Biologically Inspired Visual Control of Flying Robots

John Ross Stowers
B.E. Hons. I

A thesis presented for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

2012

ABSTRACT

Insects posses an incredible ability to navigate their environment at high speed, despite having small brains and limited visual *acuity*. Through selective pressure they have evolved computationally efficient means for simultaneously performing navigation tasks and instantaneous control responses. The insect's main source of information is visual, and through a hierarchy of processes this information is used for perception; at the lowest level are local neurons for detecting image motion and edges, at the higher level are *interneurons* to spatially integrate the output of previous stages. These higher level processes could be considered as models of the insect's environment, reducing the amount of information to only that which evolution has determined relevant. The scope of this thesis is experimenting with biologically inspired visual control of flying robots through information processing, models of the environment, and flight behaviour.

In order to test these ideas I developed a custom quadrotor robot and experimental platform; the 'wasp' system. All algorithms ran on the robot, in real-time or better, and hypotheses were always verified with flight experiments.

I developed a new optical flow algorithm that is computationally efficient, and able to be applied in a regular pattern to the image. This technique is used later in my work when considering patterns in the image motion field.

Using optical flow in the log-polar coordinate system I developed attitude estimation and time-to-contact algorithms. I find that the log-polar domain is useful for analysing global image motion; and in many ways equivalent to the *retinotopic* arrangement of *neurons* in the optic lobe of insects, used for the same task.

I investigated the role of depth in insect flight using two experiments. In the first experiment, to study how concurrent visual control processes might be combined, I developed a control system using the combined output of two algorithms. The first algorithm was a wide-field optical flow balance strategy and the second an obstacle avoidance strategy which used inertial information to estimate the depth to objects in the environment — objects whose depth was significantly different to their surroundings. In the second experiment I created an altitude control system which used a model of the environment in the Hough space, and a biologically inspired sampling strategy, to efficiently detect the ground. Both control systems were used to control the flight of a quadrotor in an indoor environment.

The methods that insects use to perceive edges and control their flight in response had not been applied to artificial systems before. I developed a quadrotor control system that used the distribution of edges in the environment to regulate the robot height and avoid obstacles. I also developed a model that predicted the distribution of edges in a static scene, and using this prediction was able to estimate the quadrotor altitude.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

PREFACE

This work began on scholarship at the Geospatial Research Centre (GRC) in early 2008 under the supervision of Dr. Andrew Bainbridge-Smith (University of Canterbury) and Dr. Steven Mills (GRC). The commercial goals of the project were the application of fixed wing unmanned aerial vehicles (UAVs) to aerial survey and mapping. My initial goal was the use of omnidirectional or wide-angle vision systems for visual control of UAVs.

At the commencement of my work quadrotor helicopters were still very new, generally used only in research, and not commercially available for a price within my budget. I also recalled a memorable lecture given during my undergraduate degree by a visiting scientist Mandyam Srinivasan. Professor Srinivasan, whose research I would go on to cite frequently, spoke on the topic of "Small minds, Smart brains: Honeybee Vision, Navigation and Cognition". The combination of these events were to shape my interests and research over the coming years.

I spent 2008 building hardware and investigating the field of biologically inspired visual control. I wanted to apply biologically plausible techniques to flying robots, and in particular, not restrict myself to strategies which only considered balancing optical flow (often called the optomotor equilibrium model, reviewed in Section 1.2.1). In the first half of 2008 I designed and constructed the first tethered quadrotor with vision capability at the University of Canterbury. Not long after, I realised this platform was insufficient and began a second revision. Through the support of the GRC I spent 4 months at the end of 2008 on exchange at École Nationale de l'Aviation Civile (ENAC), in Toulouse, France. ENAC maintains the 'paparazzi' project — at the time the most capable and popular open-source fixed-wing UAV system. During my exchange at ENAC I designed and developed, in conjunction with Antoine Drouin, the first open-source quadrotor system, the 'paparazzi booz quadrotor'. I returned from France with a working quadrotor and autopilot system.

In 2009 I added computer vision capability to the vehicle, creating the open-source 'Wasp' system; explained in Appendix A, in Stowers et al. [2010b], and in Millane et al. [2010]. Initial experiments investigated attitude control using optical flow and were successful in publication [Stowers et al., 2009, 2010a]. Unfortunately, in 2009 the GRC encountered financial difficulty stemming from the global financial crisis and a

decrease in research spending, eventually ceasing commercial operation in early 2010. As a consequence, Dr. Michael Hayes became my co-supervisor as the GRC was no longer able to employ Dr. Steven Mills.

Due to my open-source work at ENAC, I was invited to work at Eidgenössische Technische Hochschule Zürich (ETH Zürich) on the development of their custom quadrotor system: 'Pixhawk'. I spent 9 months in Zürich and continued to work on biologically inspired techniques, including; how insects perceive depth, depth from optical flow, and how concurrent biological strategies can be mapped to an artificial controller [Stowers et al., 2011d]. In early 2010 I returned to New Zealand and the Electrical and Computer Engineering department at the University of Canterbury. I continued work on the use of depth for flight control and the efficient detection of planes in depth data [Stowers et al., 2011a,b]. I also began investigating the use of edges in insect flight control.

The concluding stages of my PhD were simultaneously the most frustrating and exciting phases of my life. Late 2010 and 2011 were marred by the damaging earthquakes that struck Christchurch. While other fared worse than myself, the damage to my house, to my belongings and the inability to access my office and laboratory at University for several months affected my research and momentum to a frustrating degree. Simultaneously my work on biologically inspired flight control using edges was very fruitful and rewarding [Stowers et al., 2011c,e], ultimately leading to my current employment in the field of experimental biology, studying the visual systems of *Drosophila* at the Research Institute of Molecular Pathology (IMP).

I hope reading this thesis will enlighten you to the possibilities of biologically inspired visual flight control, and demonstrate that while the optical flow based approaches of insects have proven merit, there is utility in considering other perceptions of the world (such as edges) and through the sensible combination of simple insect inspired responses.

Papers published in this thesis in order of presentation:

J. Stowers, A. Bainbridge-Smith, M. Hayes, and S. Mills. Optical flow for attitude estimation of a quadrotor helicopter. In *Proceedings of the European Micro Aerial Vehicle Conference*, Delft, the Netherlands, 2009

J. Stowers, A. Bainbridge-Smith, M. Hayes, and S. Mills. Optical flow for heading estimation of a quadrotor helicopter. *International Journal of Micro Air Vehicles*, 1 (4):229, 2010a. doi: 10.1260/175682909790291474

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Quadrotor helicopters for visual flight control. In *Proceedings of Electronics New Zealand Conference 2010*, pages 21–26, Hamilton, New Zealand, 2010b

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Phase correlation using shear average for image registration. In *Proceedings of the 25th International Conference on*

Image and Vision Computing New Zealand, 2010c

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor. In *Proceedings of the IEEE International Conference on Mechatronics*, pages 358–362, Istanbul, Turkey, 2011a. doi: 10.1109/ICMECH.2011.5971311

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Quadrotor helicopter flight control using hough transform and depth map from a microsoft kinect sensor. In *Proceedings of the IAPR Conference on Machine Vision Applications*, pages 352–356, Nara, Japan, 2011b

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Beyond optical flow - biomimetic uav altitude control using horizontal edge information. In *The 5th International Conference on Automation, Robotics and Applications (ICARA 2011)*, Wellington, New Zealand, Dec 2011c

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Biologically inspired uav obstacle avoidance and control using monocular optical flow & divergence templates. In *The 5th International Conference on Automation, Robotics and Applications (ICARA 2011)*, Wellington, New Zealand, Dec 2011d

J. Stowers, M. Hayes, and P. Smith. A statistical model for the distribution of horizontal edges in a planar scene. In *Proceedings of the 26th International Conference Image and Vision Computing New Zealand*, 2011e

At the time of submission one further paper was under review in *IEEE Transactions on Mechatronics* and one further in preparation. Both discuss and elaborate on the work of Chapter 7.

ACKNOWLEDGEMENTS

Thankyou to my supervisors, Andrew Bainbridge-Smith and Michael Hayes for their insights and motivation, and the University of Canterbury staff for their help.

I would like to thank Andrea, and my entire family for their support throughout this PhD.

I have been fortunate to travel to, and collaborate with other institutions through my studies. I thank École Nationale de l'Aviation Civile (ENAC) and in particular Catherine Ronfle-Nadaud and Antoine Drouin for their support with the early quadrotor engineering work. Eidgenössische Technische Hochschule Zürich (ETH Zürich) also proivided me with resources and instruction at important junctures through this PhD, and I thank Friedrich Fraundorfer in particular for this.

Thankyou to Stephen Hulme who provided me with the single-board computer at no cost, and for being a willing tester of the early autopilot code.

I acknowledge the support of the Geospatial Research Centre, my co-supervisor Steven Mills, and specifically David Park who funded the first 18 months of this work.

Finally, I acknowledge the stoic and resolute people of Christchurch for their unity through a difficult 2011.

GLOSSARY

Horn and Schunck A global method for computing optical flow. 54, 65

Lucas and Kanade A local method for computing optical flow. 54, 55, 65, 95, 98

acuity The behavioural ability to resolve fine image details. iii, 18, 19, 155

Anax junius The dragonfly. 18, 19

Apis mellifera The common western or European honey bee. Usually referred to as just Apis. 1, 14, 15, 31, 33–35, 37

attitude The orientation or angular position of an object. 4, 16, 67, 69, 70, 72, 73, 75, 76, 80, 82, 89, 144, 157, 189, 193

axon A long slender projection of a nerve cell or neuron, that conducts electrical impulses away from the cell body or soma. 24, 25

Calliphora erythrocephala The blowfly. 19–21

class A taxonomic rank, lying between Phylum and Order, in the hierarchy of biological classification. 14

contralateral Taking place or originating in a corresponding part on an opposite side. On the opposite side of the body. 34, 35

dendrite The branches of a neuron conduct the electrochemical stimulation from other neural cells to the cell body. 25

diptera The order of insects possessing only a single pair of wings. 16, 20, 21

diurnal Of or belonging to or active during the day; as opposed to nocturnal. 1, 17, 18

Drosophila melanogaster The common fruit-fly. Usually referred to as just Drosophila. xx, 1, 9, 14, 15, 18–24, 28, 31, 32, 36, 94, 142, 144, 146, 148, 149, 155, 156, 158

egomotion The 3D motion of a camera within an environment. More specifically, in the field of computer vision, egomotion refers to estimating a camera's motion relative to a rigid scene. 13, 84, 91

ethology The scientific study of animal behaviour. 16

fovea In human vision; Fovea centralis. The a small depression in the retina containing only cones and where vision is most acute. 75, 77, 78, 157

gross Large; in the context of describing navigation or control, the gross behaviour is that state which the system aims to achieve over the longest time. In biological systems this is generally the search for food or a mate. 2–4, 92, 94, 96, 101, 103

haltere Small, knobbed, hind wing-like structures that function as gyroscopes in some insects. 16, 20–22, 30, 40

interneuron A non-motor or non-sensor neuron whose axons connect to other neurons within the same region of the brain or spinal cord. iii, 24, 25, 92

Lambertian A surface exhibits Lambertian reflectance if light falling on it is scattered such that the apparent brightness of the surface to an observer is the same regardless of the observer's angle of view. 53

lamina A region of the optic lobes (along with the medulla, lobula, and lobula plate). 23–25

lateral Of, at, toward, or from the side or sides. 144

lobula A region of the optic lobes (along with the lamina, medulla, and lobula plate). 22–26, 29, 33, 36

lobula plate A region of the optic lobes (along with the lamina, lobula, and medulla). 22–26, 28, 116, 118

medulla A region of the optic lobes (along with the lamina, lobula, and lobula plate). 22–26, 28, 65

Musca domestica The common house-fly. 1, 14, 15, 19, 37

neuron A specialized cell for transmitting nerve impulses, a nerve cell. Multiple neurons connect together to form a network. iii, 2, 13, 15, 22, 24–26, 28–30, 33, 34, 36, 49, 53, 90, 116, 157

ocellus Simple photoreceptors found on the head of some insects. 16, 18, 20, 144

ommatidium A single sensing unit of a compound eye. 17–19, 22, 25, 27, 28, 33, 53, 79

optomotor response A course stabilisation during free locomotion through an involuntary displacement from a straight course. The optomotor response is an innate behaviour common to all insects. Sometimes called optomotor equilibrium. 26, 33

phylum A taxonomic rank, lying between Kingdom and Class, in the hierarchy of biological classification. 14

project Terminate in, or connect to. A neuron projects from one region to another if it communicates impulses between them.. 24, 25

resolution The smallest detectable change in the quantity being measured, not the variation in general use; the number of pixels in an image. 19, 132

retinotopic The neighbourhood is respected, that is, neurons connected to neighbouring ommatidia are next to each other. iii, 22, 25, 28, 36, 89

saccade A fast movement of an eye, head or other part of an animal's body between positions of rest. 10, 32, 161

subtend The angle formed by an object at a given external point. When used in the context of vision, it is often used to describe the size and/or distance to an object. For example, to an observer on earth, the sun subtends 0.52° at its current distance and size. 42, 145

syndirectional In or towards the same directions. 26, 144

ventral Of, on, or relating to the underside of an animal or plant; abdominal. 34, 105, 144

LIST OF ABBREVIATIONS

2D two-dimensional.

3D three-dimensional.

BM block matching.

CDF cumulative distribution function.

COM centre of mass.

CPU central processing unit.

DOF degree of freedom.

EKF extended Kalman filter.

EMD elementary motion detection.

FFT fast Fourier transform.

FOE focus-of-expansion.

FOV field of view.

GPS global positioning system.

GRC Geospatial Research Centre.

HITL hardware in the loop.

IMU inertial measurement unit.

LIPO Lithium polymer.

OF optical flow.

PCB printed circuit board.

PDF probability distribution function.

PID proportional integral derivative.

RHT randomized Hough transform.

RMS root mean square.

SAD sum of absolute differences.

SBC single board computer.

SHT standard Hough transform.

SITL software in the loop.

SLAM simultaneous localisation and mapping.

SNR signal to noise ratio.

SSD sum of squared differences.

TTC time-to-contact.

UAV unmanned aerial vehicle.

USB universal serial bus.

VPN visual projection neuron.

VRML virtual reality modelling language.

CONVENTIONS

Co-ordinates
: are expressed in brackets, (x, y, x). Co-ordinates in the world reference frame are capitalised; (X, Y, Z).

Vectors
: $\mathbf{V}$, bold text with capital letters. Notable elements of vectors are indicated with subscripts. For example, let $\mathbf{V} = [x, y, z]^\mathsf{T}$ be a vector of velocities in (x, y, z) then $\mathbf{V}_x$ is the first element.

Matrices
: M, discrete, element access is indicated with $\mathrm{M}[m, n]$. By convention the letters m, n refer to row and column indices of a matrix. Sizes are given in $M \times N$.

Number Spaces
: are indicated in blackboard-bold type. $\mathbb{E}^2$, and $\mathbb{E}^3$ represent the two-dimensional (2D) and three-dimensional (3D) Euclidian space. $\mathbb{R}^n$ represents a general real-numbered space of n dimensions.

Images
: I, are treated as discrete matrices, conventionally labelled with I. Co-ordinates in the image plane are (u, v).

Computer vision geometry
: conventions are explained in Section 2.2.

Aerodynamic geometry
: conventions are explained in Appendix B. SI units are used for all quantities and when unclear, are included after the introduction of a variable using them, e.g., Ω_H $[\mathrm{rad\,s^{-1}}]$.

Biological species names
: use the two word Binomial nomenclature; the genus followed by the species. Both words are italicised, the first word (the genus) is capitalised, e.g. *Homo sapiens*.

Definitions
: are always italicised. A complete list of terms and their occurance is included in the glossary.

Acronyms
: are explained in the list of appreviations.

Chapter 1

INTRODUCTION

> “ If you wish to make an apple pie from scratch, you must first invent the universe.
>
> Dr. Carl Sagan, *Cosmos*, 1980

Consider a honeybee, fly, or small insect in flight. Even in cluttered environments and at full speed, the insect's flight behaviour is impressive. The distance between insect eyes is small and so they cannot rely on stereoscopic vision to assess the distances to surrounding objects [Collett and Harkness, 1982]. Therefore, they have developed specific visual processes to navigate their environment. These processes are responsible for controlling orientation and movement and extracting information about the 3D structure of the environment. These tasks all happen in real-time and exist in essentially all *diurnal* organisms.

Now consider a similar task performed artificially; the task of controlling a flying robot by visual means; computer vision. In order to do this one must know, just as biological organisms, much about oneself and about the surrounding environment. One begins with a flying robot and an attached camera (or cameras), but much work remains to be done before a computer vision system for robot control can be implemented.

The phrase ‘biologically inspired’ can mean many things in the field of robotics, including; mimicking the mechanical structure of biological organisms, copying their patterns of movement or behaviour, restricting robots to use the same types of sensors (including the same types of visual information), and recreation of their information processing systems [Zanker and Zeil, 2000]. In the context of flying robotics, this aspiration has had some success. For a broad review, Floreano et al. [2009] describe a number of biomimetic controllers using visual information. Similar to the work of Zufferey [2005], featured in the summaries cited above, this thesis explores the application of biologically inspired visual control strategies to flying robots. The strategies examined all have been observed or inferred from the behaviour by close study of living organisms; typically small insects such as *Drosophila melanogaster*, *Musca domestica*, and *Apis mellifera*. The goal of this thesis is to demonstrate that biologically inspired

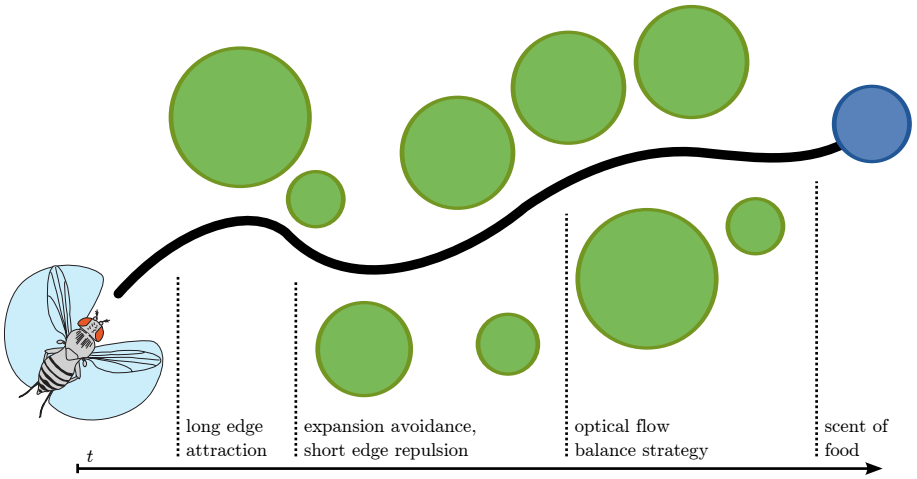
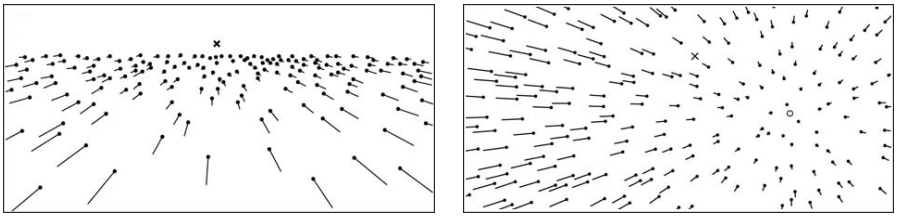


Figure 1.1: An example path of a fruit fly path towards its goal (*gross* flight behaviour). At various times during the flight different control strategies may dominate. The selection of control strategies is for illustrative purposes only.

responses and the architecture facilitating them have utility and through the sensible application and combination of such responses, a robust control system can be developed.

As we move through our environment, the image of the world formed on our retina changes. This gives an apparent motion to points in space, known as 'image motion' (or later, 'optical flow'). Representing the movement of points between two images using vectors yields Figure 1.2 and the distribution of these vectors can convey the structure of the environment. For example, looking at Figure 1.2a and drawing from our experiences of the world, one can imagine seeing such a pattern when taking off in an aeroplane. Similarly, one might associate Figure 1.2b with driving through a tunnel (the left wall of the tunnel being closer than the right). Through the use of innate knowledge and our understanding of the world, humans can construct a model sufficient for perception and action (in this example using only image motion vectors). But perception can occur on other levels too. For example, humans have an impressive ability to recognise complex attributes of the environment ranging from simple features ('corners', 'remarkable single points') to complex objects ('a chair, similar to one I own but smaller'). Thus with our complex brain; perception and understanding occur on multiple levels simultaneously drawing on many different types of visual information. This happens to various degrees in insects too. Their perception has been shaped by evolutionary forces; the limitations of their visual system and capability of their brain.

A fundamental and as yet unanswered question in insect flight is whether organisms construct an internal model of their world, or instead there exists a direct coupling between visual and motor *neurons*. The internal model hypothesis proposes that insects create a neural representation of nearby objects. The latter, direct coupling scenario proposes that steering results from neural activity without sophisticated use of stored



(a) **(b)**

Figure 1.2: Examples of image motion observed when moving through ones environment. From Werner and Chalupa [2004, ch. 84].

visual information from previous experience [Ofstad et al., 2011, Lalazar and Vaadia, 2008]. These hypotheses are not necessarily mutually exclusive; many neural responses may be combined to give a repertoire of behaviours that allow insects to successfully avoid dangerous situations, to find food, and ultimately to reproduce.

Figure 1.1 shows this scenario. It is unanswered whether the high-level *gross* flight control of an organism (moving along the example flight path) is an emergent property of multiple low-level control laws, whether top-down processes govern switching between sub-behaviours, or whether the overall pattern of behaviour is better explained by a continuously ongoing summation of low-level reflexes.

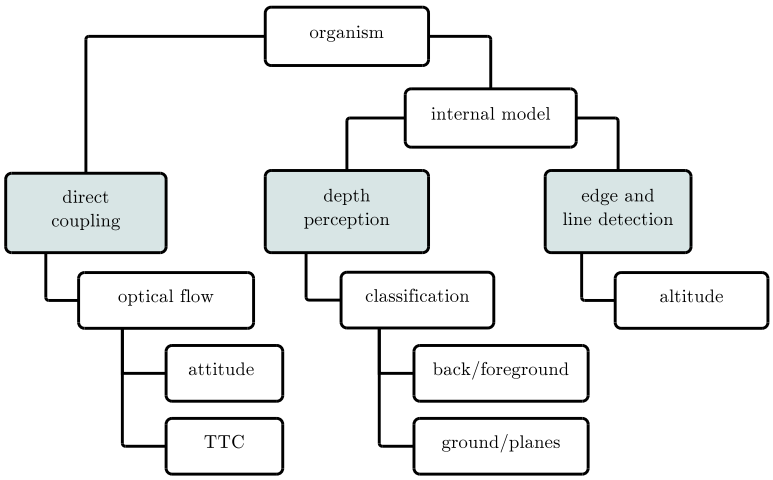


Figure 1.3: Research taxonomy of this thesis. Highlighted blocks indicate major research topics; 'direct coupling' is Chapter 5, 'depth perception' is Chapter 6, 'edge and line detection' is Chapter 7.

In this thesis I explore both the model and direct coupling hypothesis by developing and testing novel biologically inspired control techniques. A simple breakdown of the research is indicated in Figure 1.3. I consider optical flow as an important mode of visual information but do not restrict my research just to regulation of its magnitude and direction like much of the existing literature (Section 1.2.1). I consider other interpretations of optical flow such as depth-from-flow, and the interpretation of optical flow using other models such as the log-polar domain. I consider techniques closer to

the model hypothesis, those that require additional processing of the visual information such as using novel visual features including edges and lines to control flight. I investigate how concurrent combinations of these local strategies can yield *gross* flight behaviour.

This subject matter spans multiple disciplines so considerable background material is explained. The thesis proceeds as follows.

Chapter 2 explains key concepts in biology, motion vision, and computer vision including the mechanisms which insects use to observe the world and the control systems that these organisms have evolved utilising this visual information.

Chapter 3 describes optical flow and how to calculate it, while Section 3.3 introduces an efficient frequency domain method I developed.

Chapter 4 introduces the quadrotor robot platforms I developed which were used for experimental work (and whose implementation is further explained in Appendix A) and describes the engineering and architecture required for visual flight control experiments.

Chapter 5 introduces biologically inspired flight control strategies inspired by the direct coupling hypothesis. Section 5.3 describes an *attitude* estimation system that uses optical flow in the log-polar domain to measure quadrotor heading and altitude. Section 5.4 describes the time-to-contact algorithm and the well known expansion avoidance response implemented in the log-polar domain.

Chapter 6 progresses beyond direct coupling techniques and looks at biomimetic processes that model the environment using depth and not only image motion. Section 6.1 describes an indoor obstacle avoidance system that uses depth-from-optical-flow to navigate a cluttered corridor in real and simulated environments. Section 6.2 describes methods for estimating planes from depth data such as the ground plane and walls, and uses those estimates to control the flight of a quadrotor helicopter. This describes work undertaken using the Microsoft Kinect® sensor.

Chapter 7 describes biologically inspired methods that control quadrotor altitude and heading using edges and lines in the environment and not optical flow as conventionally done. Section 7.2 introduces a statistical model describing the distribution of edges in a scene.

Chapter 8 concludes with a summary of the applicability of these results and a discussion of areas of biologically inspired control that need further study.

1.1 MOTIVATION

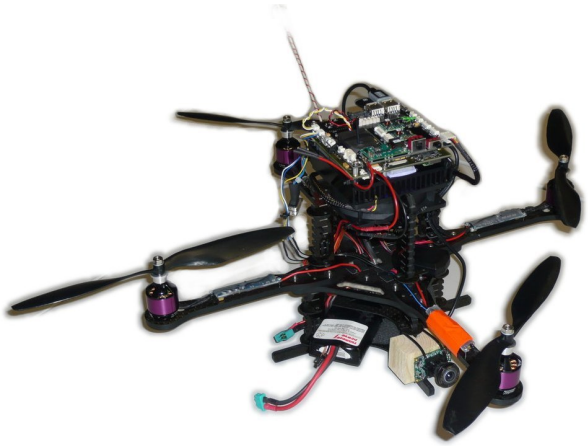
> “It is certain that there may be extraordinary activity with an extremely small absolute mass of nervous matter; thus the wonderfully diversified instincts, mental powers, and affections of ants are notorious, yet their cerebral ganglia are not so large as the quarter of a small pins head. Under this point of view, the brain of an ant is one of the most marvellous atoms of matter in the world, perhaps more so than the brain of man.
>
> Charles Darwin, *The Descent of Man*, 1871

The operation of robots in urban and indoor unknown environments has shown the necessity of robust systems to sense and navigate one's environment. Many approaches over multiple disciplines have been explored to solve this difficult problem [Siciliano and Khatib, 2008, ch. 35–37]. Simultaneously, the conflicting desires of miniaturization and increased performance have also pushed solutions to be more integrated.

Computer vision has often been applied as a solution to these problems. Computer vision algorithms have improved over the decades and Moore's law continues to hold. This has allowed smaller and more powerful computers to run more capable algorithms on smaller and smaller robots; and now even on flying robots (Figure 1.4). Due to these trends I try to run all computation in real-time on the 'wasp' quadrotors.

Much of the robotics community is focused on solving the navigation problem using simultaneous localisation and mapping (SLAM), where the control problem is often recast as a navigation problem on a map built by computer vision techniques [Siciliano and Khatib, 2008, ch. 37]. However, on the basis of the capability of biological control systems, I believe that multiple, local concurrent control systems can provide a robust basis for flying robotics. I hope to avoid a heavy dependence on a too-general purpose SLAM system. I believe that if implemented efficiently, single purpose biologically inspired controllers should augment SLAM systems, running in parallel and providing important local information such as avoidance cues and navigation recommendations for path planning.

On another note; biomimetic and biologically inspired artificial systems have frequently had a synergistic relationship with the natural sciences; with advances in one giving direction to research in the other [Koenderink and Doorn, 1987, Dahmen et al., 2001, Krapp and Wicklein, 2008]. I hope that contributions I make here could also be used to direct research in the natural sciences.



(a)

(b)

(c)

Figure 1.4: Mobile flying robots with computer vision capability developed and used in this thesis. (a) Asctec Pelican quadrotor with onboard processing. See Appendix A.4 for hardware detail and Section 6.1 for experiments using this robot. (b) 'Wasp' quadrotor with onboard and offboard processing. See Appendix A.4 for hardware detail and Section 5.3 for experiments using this robot. (c) Improved 'Wasp' quadrotor with onboard image processing. See Chapter 6 and Chapter 7 for experiments using this robot and Appendix A.4 for hardware detail.

1.2 EXISTING VISUAL CONTROL SYSTEMS

As mentioned, SLAM has emerged as the most popular architecture to solve the problem of autonomous avoidance and navigation in an unknown and unstructured environment [Siciliano and Khatib, 2008, ch. 25,35]. While SLAM is not in-scope for this thesis, this section includes a review of its basic concepts as it represents the state-of-the art in the field for onboard UAV control.

The formulation of the autonomous avoidance and navigation problem SLAM attempts to solve is;

Localization Given knowledge of the environment (a map), the robot needs to estimate its location with respect to landmarks.

Mapping Given knowledge of its position, the robot needs to map the positions of landmarks that it encounters in its environment.

SLAM The robot simultaneously maps landmarks it encounters and determines its position (as well as the position of the landmarks) using noisy sensors.

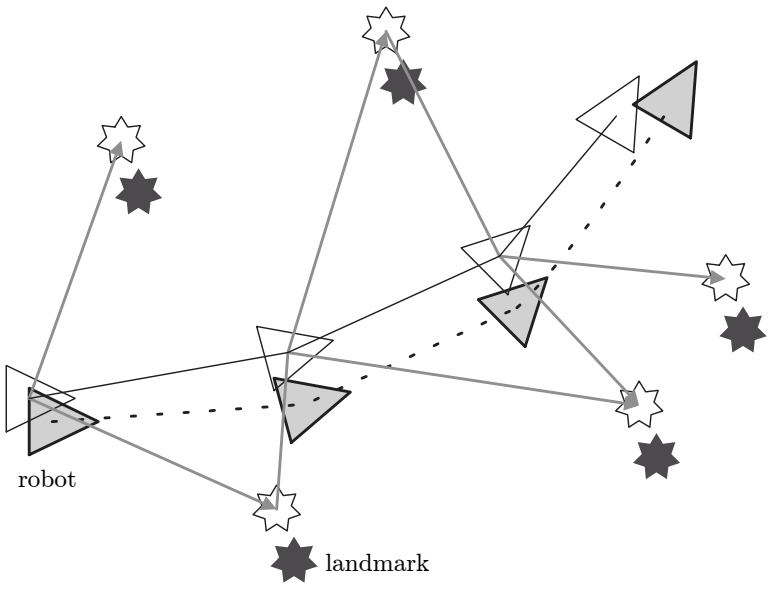


Figure 1.5: An introduction to the SLAM problem. The robot moves in its environment while observations are made of the distance to landmarks. The true locations are never known or measured directly. Estimates are shown in grey, the true location in white. Adapted from Durrant-Whyte and Bailey [2006].

Figure 1.5 shows this formulation. A SLAM algorithm is said to operate on, and maintain the state of; the robot pose and a representation or map of the environment (feature landmark locations)[1]. Due to noise associated sensor data, this state will diverge from the true real-world state through the propagation of these inherent errors.

[1] The term 'feature' refers to a specific detectable and distinguishable point in the environment. The term 'map' refers to a vector of feature location estimates. A feature is a general term for a detected landmark.

Furthermore, if a robot uses an observation of an imprecisely known object to update its position, the resulting vehicle estimate becomes correlated with the object location estimate. Likewise, correlations are introduced if an observation taken from an imprecisely known position is used to update the location estimate of a geometric feature in the map [Leonard and Durrant-Whyte, 1991]. Thus, a rigorous solution to SLAM must explicitly represent all the correlations between the estimated vehicle and geometric feature locations. This was successfully done by Smith et al. [1988] and Leonard and Durrant-Whyte [1991] who used an extended Kalman filter (EKF) to provide an optimal estimate of the robot feature positions in the presence of noise.

At a theoretical and conceptual level, SLAM can now be considered a solved problem [Durrant-Whyte and Bailey, 2006]. However, algorithmic challenges remain because a consistent full solution requires a joint state composed of the vehicle pose and every object position. This state needs to be updated following each observation and thus requires the estimator to employ a huge state vector (on the order of the number of landmarks maintained in the map) and with computation scaling $\mathcal{O}(n^2)$ with respect to the number of objects on the map.

In the last decade computer vision research has produced improved SLAM formulations and implementations that can be used in larger environments (larger maps) and in real-time (for a review see Durrant-Whyte and Bailey [2006], Bailey and Durrant-Whyte [2006]). This means that for the last few years, for small UAVs, monocular (single camera) SLAM is the optimum choice for navigation when considering performance, energy and weight constraints.

This work is exemplified recently by Weiss et al. [2011] and Tanaka et al. [2012], both of whom developed such systems. The authors presented vision-based UAV controllers using monocular SLAM technique (based on PTAM by Klein and Murray [2007]) and fusion with inertial sensors to recover map scale. Weiss et al. developed a heuristic for keyframe (and thus local map) updating and IMU fusion. Both Tanaka et al. and Weiss et al. choose optimal UAV controllers to ensure robust flight. While both systems still suffer the problem common to all SLAM algorithms; the size of the world is bounded by computational constraints via the expensive global bundle adjustment step[2], the size of the map they can work on in real-time is large enough to be eminently useful.

1.2.1 Biologically Inspired Visual Flight Control

This is not the first work to consider the control systems of flying insects as inspiration for many control strategies; for a 2005 review of the field consult Zufferey [2005], and

[2] The last step of most 3D estimation algorithms; an optimization problem of the 3D structure and camera parameters (pose, possibly distortion coefficients). Typically involves minimizing the reprojection error between the image locations of observed and predicted image points. Complexity of, in general, $\mathcal{O}(N^3)$ for N variables [Hartley and Zisserman, 2004].

for a 2009 review of the field consult Floreano et al. [2009, ch. 11–21].

The level of research activity in this area is not surprising, because despite possessing relatively small brains and simple nervous systems, flying insects provide clear demonstrations that living organisms can evolve surprisingly competent mechanisms of guidance and navigation. Unlike creatures that walk, flying animals need to control their their horizontal motion, altitude, and attitude.

Almost all the literature on biologically inspired control systems has concentrated on the use of optical flow to control flight parameters [Ruffier and Franceschini, 2005]. This section presents an overview of the state of the art in this field, including representative and important results. Additional reviews of existing work where relevant is included at the beginning of each chapter introducing a new biomimetic strategy.

Srinivasan [2006] showed that honeybees negotiate narrow gaps by balancing the speeds of the images in their two eyes. Honeybee flight speed and reaction to large disturbances is regulated by holding constant the average image velocity, as seen by the two eyes. Similarly, when insects are flying forward, the image of the ground sweeps backward across their ventral view-field and forms an 'optical flow', which depends on both the groundspeed, V_x, and the altitude, h (see Figure 7.17).

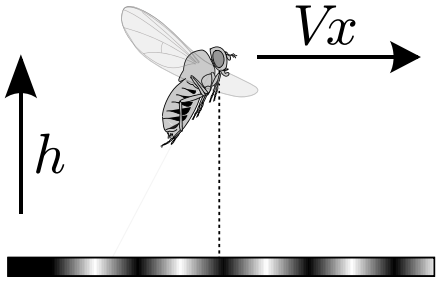


Figure 1.6: Definition of the existing variables controlled using ventral optical flow. h is the height above ground. V_x is the velocity along the x-axis in the world coordinate system. Adapted from Straw et al. [2010].

Srinivasan built several autopilot systems where the magnitude of optical flow controlled the forward speed, V_x (Figure 1.6). This allowed smooth landings on horizontal surfaces by holding optical flow constant as the surface is approached, ensuring that the flight speed is close to zero at touchdown [Srinivasan et al., 1996, 2006].

Similarly, Ruffier and Franceschini [2005] and Franceschini et al. [2007] implemented a helicopter autopilot whose visual control system adjusted the thrust to keep the ventral optical flow at a constant value; achieving terrain following, take- off, and landing. Unlike Srinivasan, the controller of Ruffier and Franceschini used optical flow to regulate altitude, h, changing forward speed, V_x, in response. The authors suggested that this model more closely matched the free-flight behaviour of honeybees, locusts, dung beetles, and mosquitoes.

Tammero and Dickinson [2002a] studied the visual cues of *Drosophila* in discrim-

inating obstacles in free flight. The authors found that image expansion is the signal that triggers each *saccade*, causing the insect to turn away from regions of high optical flow. The authors also found that the asymmetry responsible for instigating the *saccade* only defines its direction and not its magnitude. However, while in *saccade*, other visual responses including; the optomotor equilibrium response, and an immediate *saccade* in the opposite direction, are attenuated.

Other examples of biomimetic optical flow based flight control include: the control of a UAV in canyon flight [Hrabar and Sukhatme, 2004, Hrabar et al., 2005] by regulating optical flow on both sides of the aircraft; urban flight control and the avoidance of skyscrapers using a proportional controller [Muratet et al., 2005]; centering behaviour of wheeled robots using optical-flow [Serres et al., 2006] and an omnidirectional camera [Argyros et al., 2004]; automatic take-off, landing [Green et al., 2003, Ruffier and Franceschini, 2004], and obstacle avoidance by turning away from regions of high flow [Green et al., 2004].

In summary, much existing research has focused on the interpretation of optical flow at a primary level. Recent research has suggested that insects might have visual processes which utilize more than just the magnitude and direction of optical flow. For example; biological inspiration using the role of lines (and the planar ground surface) has been used for monocular vision based UAV control. Celik et al. [2009a,b] present a combined SLAM/ranging solution whereby the range and orientation of the craft is recovered from the slopes of lines in the environment. Subsequent work from the same authors [Yang et al., 2011] extended the idea of attitude determination via consideration of ground based features, but this time used a more conventional feature correspondence based technique, recovering the epipole and subsequently the UAV attitude.

1.3 CONTRIBUTION

This thesis has taken a bottom to top approach to biologically inspired control of flying robots. Utilising the 'wasp' quadrotor system I developed [Stowers et al., 2010b], I have contributed from the lowest levels of information processing to higher levels of navigation and control. At the lowest level I developed an efficient optical flow method [Stowers et al., 2010c] inspired by insect physiology. Using only optical flow I demonstrated it was possible to estimate the heading and altitude of a flying robot [Stowers et al., 2010a] efficiently in the log-polar domain. Once again, using optical flow I developed an obstacle avoidance strategy based on clustering similar flow vectors and introducing inertial information to measure distance to obstacles in the environment [Stowers et al., 2011d]. Using depth information again, this time obtained from a depth camera, I contributed a real-time control system for altitude control via ground plane detection [Stowers et al., 2011a,b]. Continuing in the theme of altitude control I developed a novel biomimetic indoor flight strategy based on the detection of edges in one's envi-

ronment [Stowers et al., 2011c], and an accompanying statistical model which predicts the distribution of edges and uses this to estimate altitude [Stowers et al., 2011e].

In general, my work uses low resolution visual information, real-time onboard computation, and similar information processing techniques as those present in biological organisms. In this way it is biomimetic; there is no learning or memory (no feature detection nor correspondences undertaken), and no map building (local or global). This work is most similar to the work of Ruffier and Franceschini [2005] and Franceschini et al. [2007], both of which used optical flow and a local strategy to control UAV altitude and attitude, but otherwise left position and navigation to a separate control architecture.

Chapter 2

VISION AND PRINCIPLES OF FLYING INSECTS

This chapter introduces background material to understand the research in this thesis. This covers two major fields; insect biology, and computer vision. The chapter proceeds as follows. Section 2.1 covers the biological principles of visual flight control, describing the sense organs insects utilise and the common behaviours observed in their flight. Section 2.2 introduces the fundamentals of computer vision including artificial images, geometry and transformation, and the computer vision pipeline.

2.1 INSECT VISION AND FLIGHT CONTROL

The mastery of flight by small insects, the first animals to evolve flight [Dickinson et al., 1999], is impressive given the limitations of their physiology. Their neural mechanisms accomplish sophisticated behaviours despite lacking many of the sensory receptors we *Homo* are accustomed to[1]. Even the insect visual system is highly constrained in relation to our own, with limitations including:

Neural limitations *Neurons* possess comparatively small dynamic ranges for representing intensities and for representing temporal changes with only spike trains. Furthermore, they can only approximate mathematical operations (such as elementary motion detection and optical flow in Section 2.1.3.2).

Ecological limitations Given that vision systems have evolved under selective pressure in the specific visual habitats of their owners, it is common that their neuronal processing strategies display characteristic adaptive properties and assumptions about their natural environment [Zanker and Zeil, 2000, ch. 0,6]. This concept is explored further in Section 2.1.4.

Geometric limitations The visual system has to extract relevant information about *egomotion*, about the 3D environment, and about moving objects in the scene from complex 2D images. The data is highly ambiguous due to elementary problems such as the *aperture problem* (described further in Section 3.1).

[1] For example, *Homo* possess a sophisticated vestibular system which allows us to sense rotational movement and linear acceleration — the inner ear.

The insect vision system is termed 'motion based' or a 'motion-vision' system. This is because, strictly speaking, relative motion between the eyes and the surroundings is a necessary prerequisite for any perception [Krapp and Wicklein, 2008]; as photoreceptors stimulated at a constant light level adapt. Thus a perfectly stabilized image results in the loss of visual perception. Despite this, motion is eminently useful and allows one to make sense of the world in various ways. For example, relative motion allows one to infer how far away objects are and breaks camouflage between motionless objects and a stationary background. Analysis of image motion is also essential for the estimation of self-motion, a critical element of flight control. For these reasons, vision is one of the most thoroughly investigated fields in insect biology; the following subsections introduce this in detail.

Section 2.1.1 describes the model organisms studied and whose behaviour I seek to replicate. Section 2.1.2 presents the sensory receptors these organisms insects possess, Section 2.1.3 shows how the information from these sensors is processed and interpreted before being utilised to drive the flight behaviour seen in insects; examples and explanations of which are included in Section 2.1.4. This tiered approach to information processing is depicted in Figure 2.2.

2.1.1 Insect Model Organisms

Insects are members of the *phylum* Arthropoda and the *class* Insecta. They are defined by the presence of an exoskeleton (external skeleton), a segmented body, and jointed appendages. Their main source of visual information is through their use of compound eyes. Insects are the most successful flying animals in Arthropoda and are characterised by a three-part body (head, thorax, and abdomen), three pairs of jointed legs, compound eyes, and two antennae.

Arthropods possesses a repertoire of behaviours mostly concerned with feeding, mating, and stability reflexes. Although this may sound unchallenging, the motor tasks involved are in some cases performed at a level of speed and perfection beyond that of any vertebrate or man-made device. Furthermore, some insects solve categorization and learning tasks complex enough to assume that these creatures possess cognitive capabilities [Krapp and Wicklein, 2008]. The combination of these behaviours and rigorous input-output analysis have made insects popular model organisms for biologists.

Model organisms are species that are consensually agreed upon to be extensively studied. The selection of model organisms is made in order to understand particular biological phenomena, with the expectation that discoveries made in the organism model will provide insight into the workings of other organisms [Fields and Johnston, 2005]. Insects selected as model organisms for further study, in particular for their flight control behaviour, include the fruit fly (*Drosophila*), the honeybee (*Apis*), and the housefly (*Musca*); these are shown in Figure 2.1.

(a) **(b)** **(c)**

Figure 2.1: The model organisms frequently studied by biologists to understand their visual flight behaviour. (a) *Drosophila*, (b) *Apis*, (c) *Musca*.

Because of their status as model organisms, the brain and visual system of *Drosophila*, *Apis*, and *Musca* have been investigated extensively over many years through the use of behavioural and electrophysical studies. The number of *neurons* in the brain varies dramatically from species to species, for example *Drosophila* has approximately 1×10^5 *neurons*, and we are still years away from understanding the insect's brain and sensory systems in detail[2]. Currently, our understanding ranges from the structure of their brains and visual system, up to models of their information processing and high level behaviours [Chapman, 1998, pt. v].

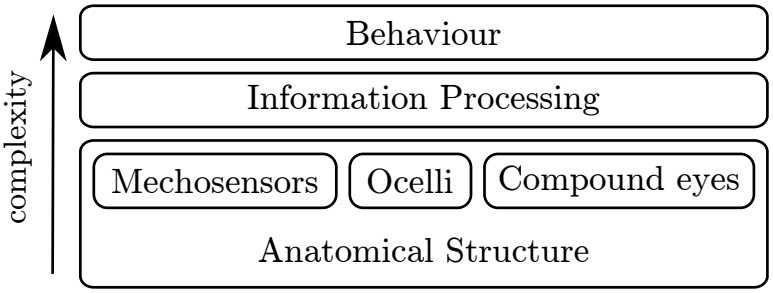


Figure 2.2: The structure of perception; information processing, behaviour, and action in the fly brain. Three levels of complexity (and hence abstraction) are identified and recreated in this thesis.

Insects process visual information in a hierarchical manner, with perception and action part of a single closed loop process rather than separate entities [Gibson, 1986, ch. 4]. Conceptually however, separating this arrangement into three levels helps when designing biomimetic implementations. Figure 2.2 shows the division of levels made in this thesis; anatomical, information processing, and behavioural.

[2] For comparison, the human brain has approximately 100×10^9 *neurons* and conservatively 225×10^{15} interactions between them [Herculano-Houzel, 2009]. In this instance we are decades, or perhaps centuries away from an understanding compatible to that we have of insects. Furthermore, it is an open question as to whether it is possible, or even desirable, to understand the human brain at the same level of detail as the insect brain.

At the first level, the anatomical description of flying insects is used by many when designing biomimetic flight vehicles; such as flapping wing and ultra light configurations [Floreano et al., 2009, ch. 11–21], or image sensors [Barrows and Miller, 2001, Barrows et al., 2002]. In this thesis the first level (Section 2.1.2) is relevant to the choice of biomimetic sensor modalities; vision and mechanosensors for the measurement of rotations and linear accelerations. At the second level is biological information processing (Section 2.1.3). The use of vision dominates here; several strong responses have been observed to crafted stimuli in neurophysiological studies or tethered flight conditions [Egelhaaf and Borst, 1993]. The third level (Section 2.1.4) has been studied through free-flight behaviour, *ethology*. These experiments have shown how insects navigate their complex environments and still manage to take full advantage of their sensor capabilities.

The remainder of this chapter discusses these three levels, introducing the required background for the artificial implementation of the behaviours discussed later. This specifically includes *attitude* and altitude control, obstacle avoidance, and course control behaviour and strategies (i.e., those useful for indoor flying robotics).

2.1.2 Sensory Receptors

Anatomical and neurophysiological study of insect model organisms has shown they possess sensory receptors that allow them to see, smell, taste, hear, and touch their environment [Klowden, 2007]. However, by a large magnitude insects rely on visual information for flight control [Chapman, 1998, pt. V]. Insects use visual feedback to perceive depth [Srinivasan et al., 1991, Tammero and Dickinson, 2002a], to control flight speed [Srinivasan et al., 1996, Srinivasan and Zhang, 2000], to detect obstacles [Egelhaaf and Borst, 1993, Tammero and Dickinson, 2002a, Maimon et al., 2008], to land [Srinivasan et al., 2000b, Borst and Bahde, 1988, Borst, 1990], to measure self-motion [Krapp and Hengstenberg, 1996, Krapp, 2000], to estimate distance travelled [Srinivasan et al., 2000a], and to regulate altitude [Straw et al., 2010]. The following subsections discuss the sensory receptors responsible for these measurements. This includes; how visual information is collected in the compound eye and in the *ocelli* (Section 2.1.2.1), the *halteres* for measuring gyroscopic information (Section 2.1.2.2), and other proprioceptive receptors that sense air movement, act as strain gauges and possibly measure acceleration (Section 2.1.2.3).

2.1.2.1 Eyes

Figure 2.3 shows the characteristic compound eyes of an insect. Compound eyes are found on most *diptera*. The true biomimetic equivalent of the compound eye is a

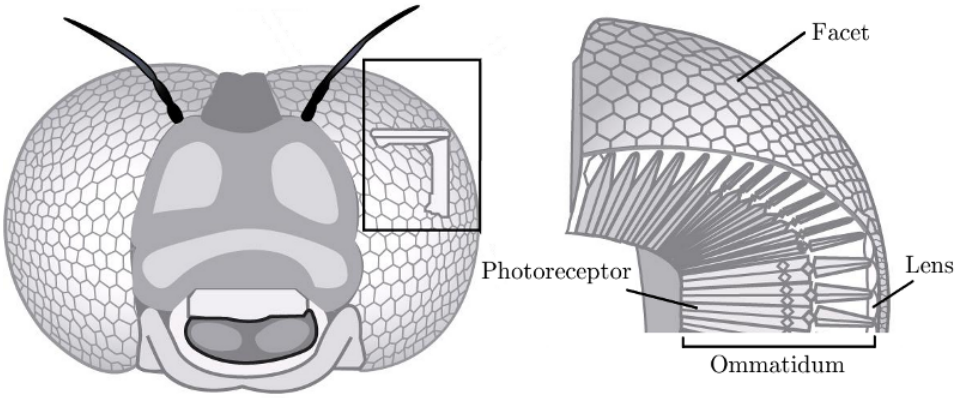


Figure 2.3: The compound eye of an insect. The eyes are composed of repeating facets, each of which functions independently of the others. Each facet is composed of a lens and one or many photoreceptors (called an *ommatidium*). Image adapted from `http://lis.epfl.ch/curvace`.

microlens[3] array.

Compound eyes are located in pairs and on the head of the insect, each compound eye is built from multiple facets called *ommatidia*, singular: *ommatidium*, arranged in a repeating pattern. At the front of each facet is a lens which admits a small part of the scene and behind lies one or more photoreceptors.

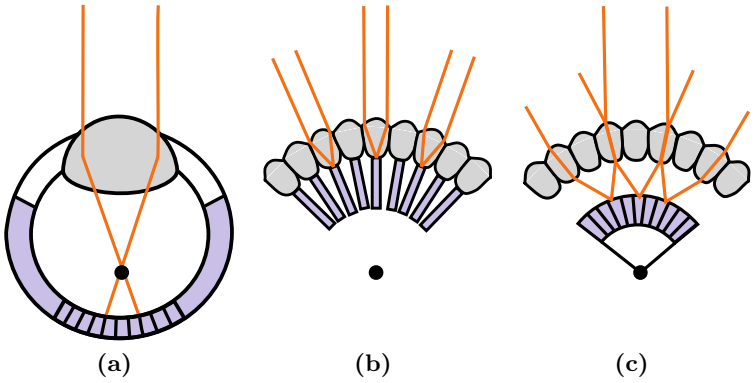


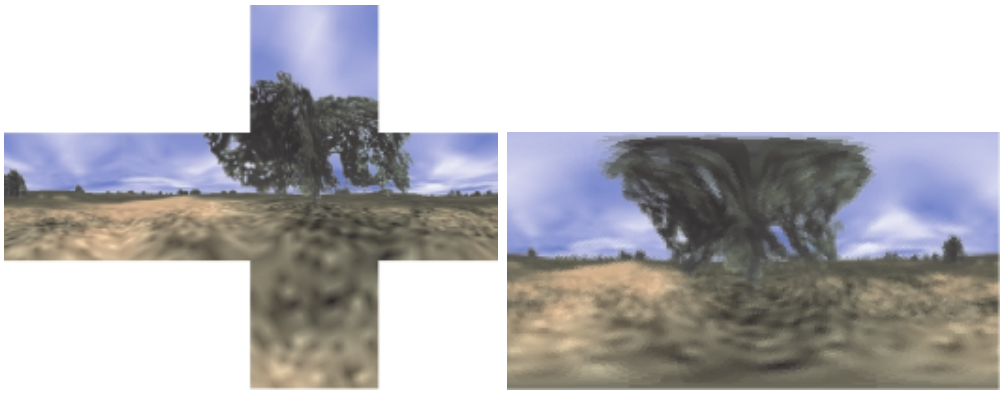
Figure 2.4: A comparison of different types of eyes. The optical centre of the eye is marked with a black circle. Photoreceptors are marked in purple, lenses are marked in grey. (a) A refractive cornea lens eye, the type usually found in vertebrates, including humans. (b) An apposition compound eye, commonly found in *diurnal* insects. (c) A refracting superposition eye, commonly found in nocturnal insects.

Compound eyes differ according to their structure and the way photoreceptors are distributed between the *ommatidia* (shown in Figure 2.4). In the case of an apposition eye, each *ommatidium* focuses only rays that are almost parallel to its long axis, so that each forms an image of only a very small part of the visual field. The image of the whole results from a combination of these partial images. In the case of a superposition eye,

[3] For more information on current research undertaken, consult the CURVACE (`http://www.curvace.net/`) project.

the sensory cells of an *ommatidium* can pick up light from a large part of the visual field so that the image received may overlap those received by as many as 30 neighbouring *ommatidia*. The superposition image thus gains in brightness but loses in sharpness compared with the apposition image [Land, 2005, Nilsson, 1989]. *Diurnal* insects have apposition eyes, whereas nocturnal insects have superposition eyes[4].

From an information processing perspective and for simplicity, I assume that each *ommatidium* corresponds to a single pixel and all *ommatidia* together capture the whole scene. The compound eye configuration allows a much wider field of view (FOV) because of the adjacent and close arrangement of *ommatidia* at different orientation instead of a single lens and a focal plane. Specifically, *Drosophila* can see in almost any direction except for the blind spot caused by their body [Neumann, 2002], shown in Figure 2.5.



(a) **(b)**

Figure 2.5: Images demonstrating the warping and field of view (FOV) of the insect compound eye. (a) Original image environment cube-map. (b) Image after warping according to compound eye model of *Anax junius*. The model specified 2562 *ommatidia*, an interommatidial angle of 4.3°, and a uniform facet spacing. Images from Neumann [2002].

Land [1997] presented a comprehensive comparison of the visual *acuity* of insects, showing that even the most capable in class, the dragonfly *Anax junius* still had $10\times$ fewer pixels than a modest webcam (VGA), and $10^4\times$ fewer than the human visual system (Table 2.1).

Although the *ommatidia* are arranged in a hexagonal array, it is useful to think of the equivalent size in terms of the standard rectangular array shape of digital cameras. For example, consider *Drosophila*— taking the square root of the number of *ommatidia* (700) is roughly equivalent to a 26×26 pixel sensor. Even the lowest quality digital camera (VGA, Table 2.1) has three orders of magnitude more photoreceptors than the *Drosophila* eye.

[4] This is a simplification. There are in fact many intermediate grades of compound eye found in insects, many forms of simple (lens) eyes in vertebrates, and many other primitive eyes; pit eyes, *ocellus*, etc. For a comprehensive review of the field, see Land and Nilsson [2002].

Organism	Number of pixels
Human retina (rods)	1.1×10^8
Canon 5D camera	2.1×10^7
Human retina (cones)	5.0×10^6
VGA resolution	3.0×10^5
Dragonfly *Anax junius*	2.9×10^4
Housefly *Musca domestica*	3.0×10^3
Fruit fly *Drosophila melanogaster*	7.0×10^2

Table 2.1: The number of pixels (or pixel equivalents, *ommatidium*) in biological and artificial vision sensors. Human vision data from Wandell [1995], insect vision data from Land [1997].

The number of pixels in the two imaging systems is not the only, nor the fairest way to compare the *resolution*[5] of insect and artificial vision systems. A better comparison of such systems considers the FOV, or more specifically the angular resolution. The angular resolution in compound eyes is determined by the interommatidial angle — the angle between adjacent *ommatidia*. The smaller the interommatidial angle the greater the distance at which objects (prey, predators, or foliage) can be resolved. Table 2.2 shows a summary of this information.

Organism	Interommatidal angle
Human retina	0.008°
Blowfly *Calliphora erythrocephala*	1.1°
Housefly *Musca domestica*	2.5°
Fruit fly *Drosophila melanogaster*	5°

Table 2.2: The interommatidal angle for various insects, compared with the angular resolution of foveated vision in humans. Human vision data from Wandell [1995], insect vision data from Land [1997].

It is remarkable that insects are capable of such impressive navigation when one considers their low-resolution eyes. This limited spatial *acuity* is a consequence of the compound eye. In order to increase spatial *acuity*, more *ommatidia* are required, however, the resolving capability of each *ommatidium* is limited by diffraction [Völkel et al., 2003]. Consequently, each lens must also be made larger — for example a compound eye with the *acuity* of the human fovea would have a radius of 11.7 m [Harrison, 2000].

Despite having poor spatial and angular resolution, insect vision far exceeds human eyes in the temporal domain. Human vision is sensitive to temporal frequencies up to 50 Hz [Wandell, 1995], whereas *ommatidium* respond to temporal frequencies as high

[5] Here I use the formal meaning of the word resolution; the smallest detectable change in the quantity being measured, and not the variation in general use; the number of pixels in an image.

as 200 Hz to 300 Hz [Dudley, 2000, p. 206]. This allows flying insects to detect rapid changes in the visual field.

Stereopsis: Insect eyes also differ from vertebrate or human eyes in their static nature. Insects have immobile eyes with fixed-focus optics. Therefore, they cannot infer the distances to objects or surfaces, from the extent to which the directions of gaze must converge to view the object, or by monitoring the refractive power that is required to bring the image of the object into focus on the retina [Srinivasan, 2006]. Even though insects do not possess the required neural apparatus for stereopsis (or triangulation) [Srinivasan, 1993], the small baseline and low resolution would ensure the precision with which insects could estimate range through binocular stereopsis would be much poorer and restricted to relatively small distances.

Ocelli: The *ocelli* are simple photoreceptive organs commonly found on insects, positioned on the dorsal side of the head between the compound eyes. Each *ocellus* consists of a single circular lens approximately 75 µm in diameter, with a visual field more than 40° in width which focuses light onto a low-resolution retina containing approximately 220 photoreceptors [Harrison, 2000]. The image produced on the *ocellus* retina is a wide-angle, unfocused view of the surroundings above and lateral to the insect.

Behaviourally, Schuppe and Hengstenberg [1993] demonstrated that insects use the *ocelli* to align their head with the centre of brightness. Dudley [2000, p. 212] and Srinivasan and Zhang [2004] reviewed the use of the *ocelli* for horizon detection. The two laterally directed *ocelli* stabilize roll by monitoring the position of the horizon on either side. The medial *ocellus* stabilizes pitch by monitoring the elevation of the horizon in the frontal field. The neural pathways mediating these reflexes remain to be investigated [Stange, 1981].

2.1.2.2 Haltres

Diptera have evolved a unique specialisation for sensing angular velocity. The hind wings of the species evolved from flight surfaces into tiny club shaped organs called *halteres*, shown in Figure 2.6, that function as angular rate gyroscopes.

The *halteres* beat up and down anti-phase to the wings at the wing beat frequency[6]. They move at nearly constant velocity during each upstroke and downstroke, covering nearly 180° [Nalbach, 1993]. At the base of the *halteres* lie campaniform sensilla, mechanoreceptors that measure strain. Specifically, they sense the periodic Coriolis[7] forces that act upon the oscillating *haltere* when the fly rotates [Dickinson, 1999]. The forces measured by the *halteres* are proportional to the angular velocity of the fly's

[6] 150 Hz in *Calliphora erythrocephala*, over 200 Hz in *Drosophila*.

[7] Coriolis effects are inertial forces acting on bodies moving in a non-inertial (rotating) reference frame.

Figure 2.6: *Halteres*, the club shaped appendage behind the wings of *diptera* used for sensing angular velocity. Shown clearly visible on a crane fly.

body. By integrating Coriolis force information over the 180° sweep of the *halteres*, and by combining signals from the two non-coplanar *halteres*, insects can measure angular rotation about all three axes.

The role of the *halteres* in insect flight was studied by Dickinson [1999]. Dickinson found that *haltere* feedback has two roles, primarily in gaze stabilization:

> “ One important role of the *haltere* is to stabilize the position of the head during flight by providing feedback to the neck motor system. [...]. Nalbach and Hengstenberg [1994] demonstrated that the blowfly, *Calliphora erythrocephala*, discriminates among oscillations about the yaw, pitch and roll axes and uses this information to make appropriate compensatory adjustments in head position ([Nalbach, 1993, Nalbach and Hengstenberg, 1994]). Such reflexes probably act to minimize retinal slip during flight, thereby stabilising the image of the external world and increasing the accuracy with which the visual system encodes motion.

The *haltere* were also found to take part in direct flight stabilization (again from Dickinson [1999]):

> “ Although the role of the *haltere* in stabilising gaze may be important, a more essential and immediate role of the *haltere* is to provide rapid feedback to wing-steering muscles to stabilize aerodynamic force moments.

Sherman and Dickinson [2004] studied *Drosophila* and found that sensory inputs from *halteres* and the visual system are combined in a weighted sum, and that the weighting structure places greater influence on feedback from the *halteres* than from the visual system.

2.1.2.3 Other Receptors

Hengstenberg [1991] demonstrated that campaniform sensilla on the fly wing are able to sense wing-loading. Dudley [2000, p. 213–215] also showed their presence on the thorax, like antennae allowed them to perceive the changing speed and direction of

flight. Harrison [2000, ch. 2.3.3] also speculated that because campaniform sensilla are found on the legs and neck of the fly, the organisms can sense the inertia of their head and limbs, and infer acceleration.

In addition to the campaniform sensilla at the base of the *halteres*, insects possess other mechanoreceptors that contribute sensory information to the organism. Sherman and Dickinson [2004] noted that posterior hairs on the neck of *Drosophila* could help in optimising flight by providing information about air movements and changes in air pressure.

2.1.3 Information Processing

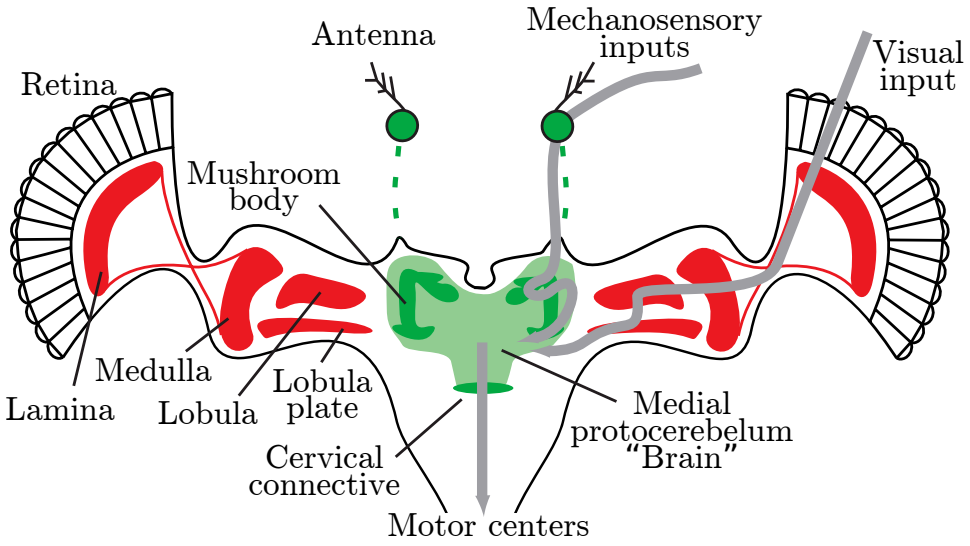
The previous section identified three sensor modalities; vision, angular velocity, and acceleration. This section focuses on the structure of the insect brain and the way information from the sense organs is combined and interpreted before passing to the motor centre; before finally being used to affect the flight of the organism.

After visual information is captured by the *ommatidia* it passes into the optic lobe. Figure 2.7a shows a simplified horizontal schematic and Figure 2.7b shows a fluorescent image of the *Drosophila* brain. The lobes on either side of the central area are by volume, larger than the entire central brain, reflecting the importance of visual information to the organism.

The laminar region of the optic lobe contains cells that exhibit transient responses to step intensity changes [Weckstrom et al., 1992] but otherwise do not perform high function behaviour. They maintain *retinotopic* organisation down to the next layer of the optic lobe, the *medulla*, before passing through the *lobula* and *lobula plate*. The function of these three parts of the optic lobe is discussed in Section 2.1.3.1.

The visual information passes from the optic lobe into the central brain, where it is integrated with other sensory information, before passing to the motor centre. Following this information flow is difficult, but by applying neuroanatomical and electrophysiological techniques, it is sometimes possible to link a specific behaviour to single *neuron*, or a chain of neuronal activity [Krapp, 2000, Krapp and Hengstenberg, 1996]. In contrast, the function of the central brain and its role in high level behaviour is less understood [Frye and Dickinson, 2003]. Specifically, how non-visual sensor modalities are integrated with visual information.

As mentioned in the introduction, there are two competing hypotheses describing the structure of insect visual control systems; the internal model hypothesis and the impulse-response hypothesis. A key prediction that distinguishes these possibilities is that a reconstruction of the instantaneous visual stimuli just prior to any steering command is sufficient to predict responses of the direct coupling case but not of the internal model hypothesis. In the extreme, particular sequences of behaviourally-generated visual stimulation predict turns in opposite directions for the two hypotheses [Straw et al.,



(a)

(b)

Figure 2.7: The *Drosophila* brain. (a) A simplified horizontal schematic. Sensory information (in grey) cascades through a hierarchy of brain centres until it reaches the central brain complex (in green). The optic lobes are shown in red. Adapted from Frye and Dickinson [2003]. (b) A 3D volume rendering of an adult brain, horizontal view. The optic lobes (*medulla*, *lobula*, and *lobula plate*) are the lobes on each side of the central area. The *lamina* (and retina) are not visible, The mushroom bodies are highlighted in the green overlay. Image adapted from `http://www.olympusfluoview.com/gallery/drosophilabrainlarge.html`.

2010]. To discriminate between these hypotheses, biological researchers measure steering behaviour of freely flying insects in response to specially designed stimuli [Straw et al., 2010, Maimon et al., 2008, Lehrer and Srinivasan, 1993, Srinivasan and Gregory, 1992].

The following subsections discusses the important regions of the insect brain, the processing undertaken by each respective region, and what behaviours are understood to emanate from them.

2.1.3.1 Optic Lobes

In insects, visual information is processed in the optic lobe before it is conveyed to the central brain.

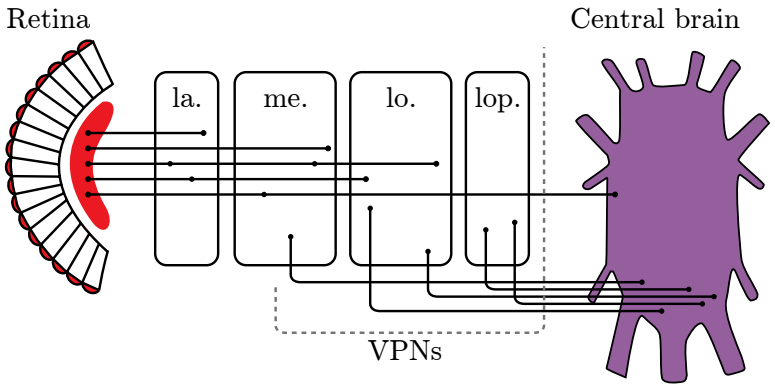


Figure 2.8: Examples of *neurons* in the optic lobe. Visual projection *neurons* (VPNs) link the *medulla* (me), *lobula* (lo), and *lobula plate* (lop) with the central brain. Adapted from Otsuna and Ito [2006].

Figure 2.8 shows a simplified schematic of the fly optic lobes[8]. Whereas a vertebrate eye contains an intense neural network within its retina, insect photoreceptor cells *project* directly to the primary visual centre of the brain, the optic lobe. The optic lobe consists of three or four aggregates of *neurons* (also called ganglia or neuropils); the *lamina*, *medulla*, and *lobula*, which are distinguishable in all the insect species. In insects such as flies, butterflies, and moths, the *lobula* is further separated into two neuropils, which are called the *lobula* and *lobula plate* [Otsuna and Ito, 2006].

Unlike *interneurons*, whose *axons* connect to other neurons in the same regions of the brain, visual projection neurons (VPNs) *project* into different brain areas. In the *Drosophila* visual system VPNs link the *medulla*, *lobula*, and *lobula plate* with the central brain. Optic lobe *interneurons* are mostly second-, third-, and fourth-order[9] *neurons* while VPNs are mostly third-, fourth-, and fifth-order *neurons*. From

[8] For a comprehensive interactive schematic of the *Drosophila* brain, see the Flybrain project. This includes the flybrain atlas (`http://www.flybrain.org/`) and the flybrain *neuron* database (`http://flybrain-ndb.iam.u-tokyo.ac.jp/index.html`)

[9] The 'order' of a *neuron* refers to the number of connections it is away from the photoreceptors.

an engineering perspective, an increasing 'order' represents a reduction in information and an increase in abstraction of the organism's environment. While *neurons* in the optic lobe are reported to extract characteristic features from visual information, such as brightness and motion, it is the VPNs which convey information to higher visual centres in the central brain[10] [Otsuna and Ito, 2006].

Retinotopic organisation is maintained through the two first neuropils down to the third one, the *lobula*, where spatial integration occurs and information from very different viewing directions are pooled together. The role of each neuropil is explained in the following paragraphs.

- The *lamina* lies just under the receptor layer of the eye and receives direct input from the photoreceptors. The *neurons* in this ganglion act as high-pass filters by amplifying temporal changes. They also provide a gain control functionality that ensures quick adaptation to varying background light. *Axons* from the *lamina* invert the image from front to back while projecting to the *medulla*.

  From an engineering perspective, the *lamina* provides image preprocessing functions like temporal and spatial high-pass filtering and adaptation to background light [Zufferey, 2005].

- Behavioural experiments suggest that cells in the *medulla* are responsible for local motion detection [Douglass and Strausfeld, 1996] (see Section 2.1.3.2 for a further discussion). The *retinotopic* organisation is still present in this second ganglion and there are about 50 *neurons* per *ommatidium*. Following the *medulla*, visual information passes into the *lobula* complex.

- The third optic ganglion, the *lobula* complex, is the locus of spatial convergence. Information from several thousand photoreceptors, having been preprocessed by the two previous ganglia, converge onto 60 cells in the *lobula plate* [Hausen and Egelhaaf, 1989].

  These cells, commonly known as lobular plate tangential cells (LPTCs), have many *dendrites* that receive synaptic inputs from large spatial regions of the *medulla*. This means they have large visual receptive fields (see Section 2.1.3.3). The *lobula* complex *projects* to higher brain centres and to descending *neurons* that carry information to motor centres in the thoracic ganglia.

Although VPNs have been described in various insect species, information about the detailed 3D form and shape of the VPNs is scarce. Research is currently being undertaken [Otsuna and Ito, 2006] to understand how visual information preprocessed

[10] For an exhaustive discussion and summary of the structure of the VPNs and the behaviour attributed to optic lobe *interneurons* by 'order', refer to Otsuna and Ito [2006].

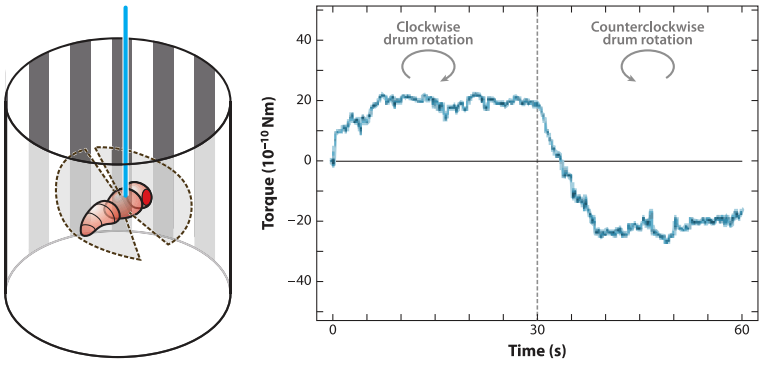
in the optic lobe is read by the *neurons* in the higher visual centres of the central brain, and how information flows from the optic lobe to the central brain.

The following sections describe the topology of optic lobe (*medulla*,*lobula*, and *lobula plate*) *neuron* connections. From the structure of these connections the organism possesses specific characteristics; such as detection of self-motion (Section 2.1.3.2) or the detection of patterns in image motion (Section 2.1.3.3). These characteristics provide the basis for higher level flight control discussed and later recreated.

2.1.3.2 Motion Detection

Although the use of image motion by insects is widely recognised as the primary visual cue [Borst et al., 2010] for in-flight navigation, an understanding of the neuronal mechanisms underlying local motion detection in the *medulla* is instructive when developing control strategies that one claims to be biomimetic.

Motion detection in insects is explained clearest through an example of its dominant form; the *optomotor response*. When a fly is tethered in the centre of a striped drum (Figure 2.9a) and the drum is rotating clockwise, the insect tries to turn clockwise too. Similarly, when the drum is moving in the opposite direction, the insect turns counter-clockwise (Figure 2.9b). This describes the *optomotor response*, consisting of a *syndirectional* reaction to the motion of the surroundings that builds up slowly over several seconds. This robust behaviour attempts to minimize retinal slip to help maintain stability in the face of external perturbations such as a gust of wind [Duistermars et al., 2007].



(a) **(b)**

Figure 2.9: (a) A demonstration of the *optomotor response* using a fly secured inside a rotating striped drum. (b) The compensatory torque generated by the fly when rotating the drum in different directions. Figures adapted from Borst et al. [2010].

These and similar behavioural experiments, coupled with recordings from the tangential cells in the *lobula*, led to the proposal of functional models describing *neurons* and their connections required for local motion detection. The best-known is the so-called correlation-type elementary motion detection (EMD) first proposed by

Hassenstein and Reichardt [1956], in which intensity changes in neighbouring *ommatidium* are correlated [Reichardt and Wenking, 1969] to detect motion. This model has been initially proposed to account for the experimentally observed optomotor response in insects [Götz, 1975, Egelhaaf et al., 1989]. This behaviour tends to stabilise the insect's orientation with respect to the environment and is evoked by the apparent movement of the visual environment.

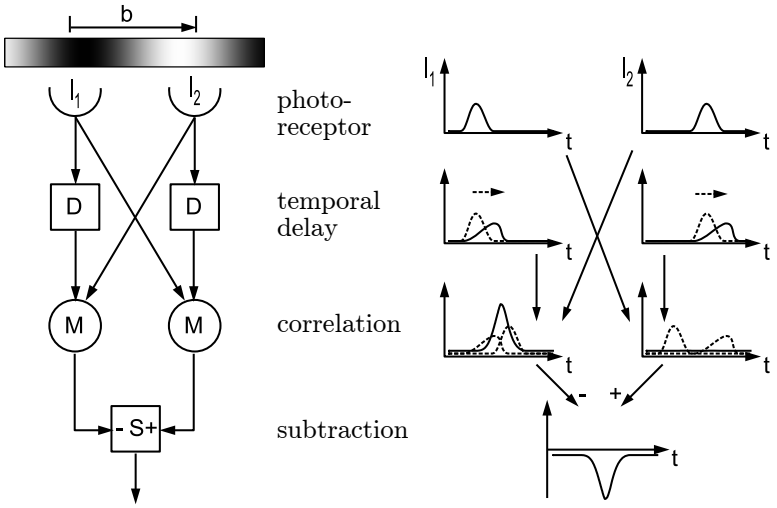


Figure 2.10: A Reichardt-type *EMD*. Two neighbouring photoreceptors detect image intensities I_1 and I_2 across a gradient b. Figure adapted from Neumann and Bülthoff [2002].

An EMD of the correlation type performs a multiplication of input signals received by two neighbouring photoreceptors as shown in Figure 2.10. Prior to entering the multiplication unit, one of the signals is delayed (usually by a first order low-pass filter), whereas the other remains unaltered. Due to these operations, the output of each multiplication unit responds preferentially to visual stimuli moving in one direction. By connecting two of them with opposite directional selectivity (with excitatory and inhibitory elements connected to a common integrating output stage), one obtains a bidirectional EMD.

It should be noted that the Reichardt detector responds maximally to a certain number of spatial periods passing by a single photoreceptor, not to a certain image speed [Srinivasan et al., 1996].

2.1.3.3 Motion Fields

When an insect moves through its environment, the patterns of motion on the retina depend on the trajectory of the organism and on the 3D structure of the surroundings. These patterns of motion contain information that tells the insect about its own motion or the distances to potential obstacles. They also depend, in a characteristic way, on the organisms' visual system; such as the eyes' field-of-view and orientation.

Koenderink and Doorn [1987] and later Dahmen et al. [2001] formalised this idea, performing numerical studies on optical flow fields to develop ideas on how insect visual systems could recover self-motion; the translation $\mathbf{T}$, and rotation, R. The 'KvD' algorithm they developed to encapsulate the relationship between a optical flow field of noisy parallax vectors $\mathbf{p}_i$ (such as those seen in Figure 1.2);

$$\mathbf{p}_i = \frac{\Delta \mathbf{d}_i}{\Delta t} = -(\mathbf{T} - (\mathbf{T} \cdot \mathbf{d}_i)\mathbf{d}_i)/\mathbf{D}_i - (\mathrm{R} \times \mathbf{d}_i). \tag{2.1}$$

Equation (2.1) shows the flow vectors $\mathbf{p}_i$ are given by the linear sum of translation and rotation vectors, and the distance $\mathbf{D}_i$ to objects at direction $\mathbf{d}_i$. The authors presented strategies for solving the set of flow vectors $\mathbf{p}_i$ to recover an estimate of rotation R' and direction of translation, $\mathbf{T}'$[11]. While (2.1) theoretically only requires five flow vectors to solve, the numerical study demonstrated that recovering self-motion is more effective if more directions are considered (a wide-field of view). Therefore, insects such as *Drosophila* (with 700 *ommatidia* and a wide field of view) are well equipped to solve the task of self-motion estimation. When doing so however, they must combine optic flow from different regions of the visual field in order to infer behaviourally significant information [Nelson and Aloimonos, 1988, Zufferey, 2005]. An example of this is shown in Figure 2.11; without a global interpretation, the highlighted local motion fields are indistinguishable. Analysis of the global motion field (of at least several different regions) is thus required.

Some form of spatial integration is known to happen after the *medulla* (where local motion detection occurs retinotopically), mainly in the *lobula plate* where tangential *neurons* receive inputs from large receptive fields [Hausen and Egelhaaf, 1989]. The *lobula plate* thus represents a major centre of motion field analysis. Some of the 60 *neurons* of the *lobula plate* are known to be sensitive to coherent large-field motion (i.e., the VS, HS, and Hx-cells), whereas other *neurons*, the figure detection cells (FD-cells), are sensitive to relative motion between small objects and background [Egelhaaf and Borst, 1993, Krapp and Hengstenberg, 1996].

As an example of the usefulness of these *neurons* at the behavioural level, there is good evidence that HS and VS-cells are part of the system that compensates for unintended turns of the fly from its course [Krapp, 2000].

2.1.3.4 Detection of Self Motion

Figure 2.11 illustrates a limitation of local motion fields. The measured local downward motion in the right lateral visual field (highlighted grey in the figure) can be generated either by upward lift translation or by roll rotation to the left. Because of such ambiguities, local motion signals cannot be used directly for motor control. These ambiguities

[11] Details of the iterative [Koenderink and Doorn, 1987] and one-shot [Dahmen et al., 2001] solutions to (2.1) are explained in the respective papers and in Krapp and Wicklein [2008].

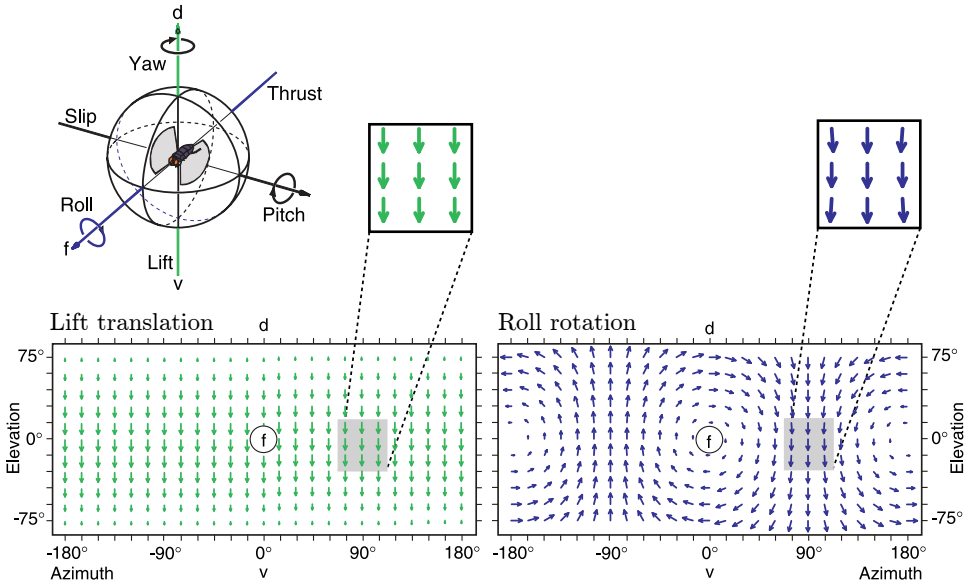


Figure 2.11: The structure of motion fields for an insect moving performing two movements; lift translation and roll rotation (directions; f, frontal; v, ventral, d, dorsal). Different motion components induce different optic flow over both eyes of the moving animal. Note the highlighted (grey box) regions of the motion field, locally, the downward tuned centre detector is excited equally in both scenarios. Yet globally the two motion fields can easily be distinguished from one another. Adapted from Krapp and Wicklein [2008].

can be overcome by a selective wide-field integration of local motion signals [Krapp et al., 1998].

Researchers found cells in the *lobula* that responded specifically to rotational optic flow fields consistent with those induced by the fly during rotations around various horizontal axes [Krapp et al., 1998]; and cells which responded to translational optic flow field [Krapp and Hengstenberg, 1996][12]. Figure 2.12 illustrates this result. The tangential cells were acting as neuronal matched filters [Wehner, 1987] tuned to particular types of global-field motion.

The first artificial implementation of this theory of visual matched filters was undertaken by Franz and Krapp [2000], garnering some success at estimating self-motion of a simulated agent. However, Krapp [2000] cautions about too simplistic interpretations of this biological model of spatial integration:

> “ [Some] approaches take for granted that the results of the local motion estimates are summed up in a linear fashion at an integrating processing stage. For insect visual systems, however, it was found that local motion analysis is achieved by elementary motion detectors whose output is not simply proportional to velocity [...] but also depends on pattern properties like spatial wavelength and contrast [...]. Hence, it remains unclear how biological sensory systems cope with highly

[12] The research found VS *neurons* are arranged retinotopically, not uniformly, resembling rotational optic flow fields that would be induced by rotations. Hx cells show the global structure of a translational flow field, however, the response fields of HS cells are more difficult to interpret since they probably do not discriminate between rotational and translational components [Krapp, 2000].

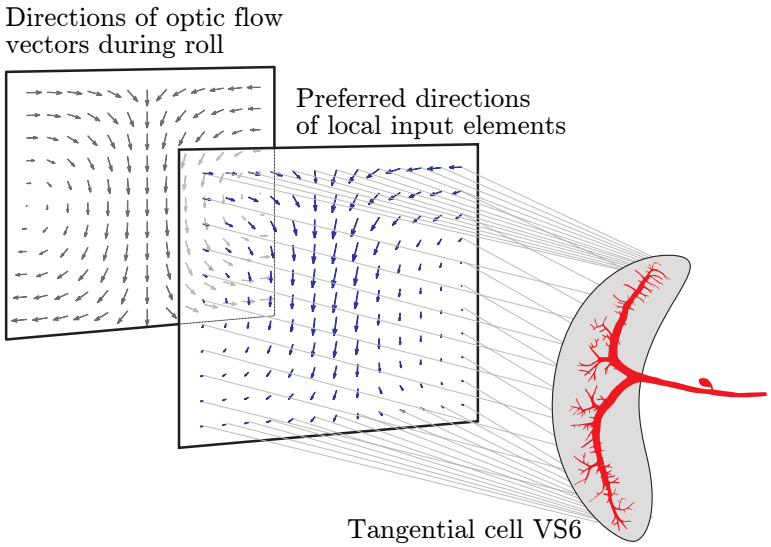


Figure 2.12: A LPTC cell (VS6) for sensing a particular pattern of self-motion. Local patterns of optic flow activate locally motion detectors with appropriate preferred directions. A wide-field *neuron* spatially integrates the signals of these motion detectors. Hence it would be most sensitive to that particular pattern of optic flow and consequently to the self-motion that caused the flow. Adapted from Krapp and Wicklein [2008].

> dynamic stimuli as encountered, for instance, by the fly during free flight. It is by no means easy to predict the signals of the tangential *neurons* under such natural conditions.

Or put simply, while the concept of large-scale spatial integration is clearly used in insects, when designing a biomimetic strategy, be careful to not just sum the output of image motion detectors (in biology; EMDs) whose output is only proportional to velocity.

Krapp [2000] also remarked that tangential *neurons* (like VS, HS, etc) cells cannot be insensitive to optic flow components induced by movements that are not their own preferred self-motion. The output of those *neurons* needs to be corrected for apparent rotations, which may be due to translational self motions and to rotations around axes other than the preferred axis[13]. One way to correct such errors is to use gyroscopic information, such as that from the haltres (Section 2.1.2.2). Krapp [2000] says again:

> “Correction signals encoding fast self-rotations may also be supplied by the *haltere* system [Nalbach and Hengstenberg, 1994]. Because the dynamic range of the *haltere* system is shifted toward higher angular velocities, it is thought to complement the visual self-motion estimation [Hengstenberg, 1991].

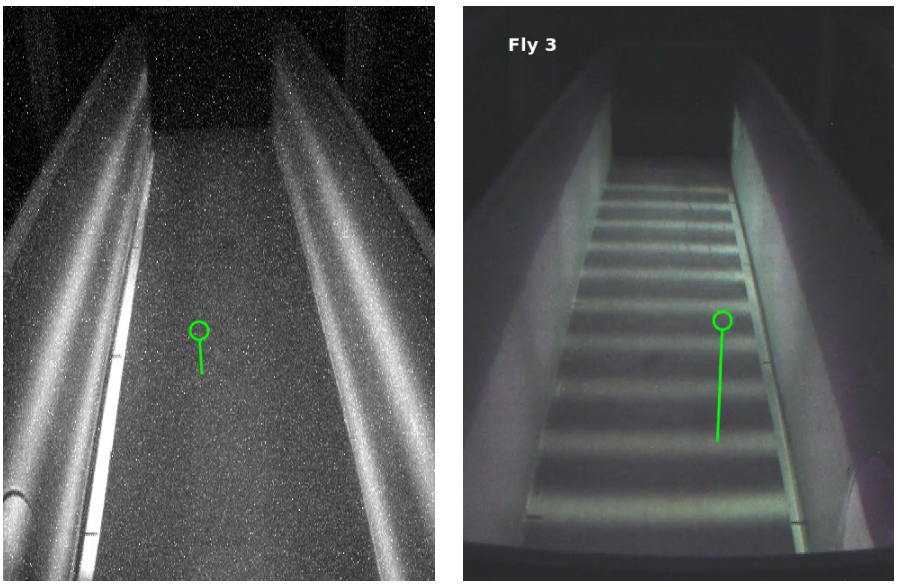
[13] This is the biological equivalent of the fundamental loss of information in computer vision, occurring when projecting a 3D scene onto a 2D plane (the sensing device, or retina). This is explained in the computer vision context in Section 2.2.1.

2.1.3.5 Edge Detection

Reichardt and Wenking [1969], in their original EMD work, showed that *Drosophila* fixate on long vertical edges in their environment. However, unlike optical flow, this phenomenon and the role of edges in other behaviours has been less extensively explored.

Lehrer et al. [1990] revealed that when landing, *Apis* pay special attention to edges which provide contrast to green-sensitive receptors. The authors demonstrated that edge-detection pays an important part in selecting the landing spot. The authors propose that edges provide cues which play an important role in guiding landing manoeuvres towards objects of interest, such as flowers.

Maimon et al. [2008] investigated the role of vertical edges on *Drosophila* turning. The authors found that while the organism was attracted to long vertical edges, it was repulsed by shorter yet similar visual stimuli. The organism could distinguish between long and short vertical edges; possessing a rudimentary object classification system. The authors proposed that attraction to longer edges would lead to feeding sites, while repulsion of short edges would help them avoid predators or other insects; the motion of *Drosophila* depended on the balance of these conflicting responses.



(a) **(b)**

Figure 2.13: A *Drosophila* subject in free-flight experiments by Straw et al. [2010]. The authors studied the altitude response of *Drosophila* (circled) to visual stimulus projected on the walls and floor of the arena. (a) *Drosophila* adjust their altitude in response to the changing vertical optical flow. (b) Most interestingly, the authors showed that *Drosophila* regulates its altitude also in response to the light/dark edge surrounding it (shown on the sides of the enclosure) and not by regulating the magnitude of ventral optical flow (pattern on the floor of the arena). Images courtesy Andrew Straw (unpublished).

Straw et al. [2010] investigated the conventional understanding of altitude control

and found that *Drosophila* utilize three reflexes to control altitude: wide-field stabilization (Section 2.1.4.2), expansion avoidance (Section 2.1.4.5), and edge tracking. The first two responses utilize optical flow in the conventional manner (explained in their respective flight behaviour section), but the third response utilized edges in a previously unseen way.

Straw et al. showed that *Drosophila* establish an altitude set point on the basis of nearby horizontal edges and tend to fly at the same height as such features. This is shown in Figure 2.13 — once a light/dark edge is introduced, the *Drosophila* control their altitude based on the position of the edge. The authors noted that this reflex is invariant to contrast sign; a light-to-dark edge is roughly as attractive as a dark-to-light edge. Straw et al. also suggest that in the natural environment it is likely that the edge response is used to approach and fly level with nearby visible objects, such as the tops of vegetation or geological features.

The result of Straw et al. was the first experimental evidence that freely flying insects adjust their altitude to the height of nearby visual features; in this case, edges.

2.1.3.6 Distance or Depth Estimation

Two visual mechanisms potentially allow for distance or depth estimation: stereo vision and motion parallax. Stereopsis will not be covered here as it has not been observed in the model organisms I consider.

Exploiting motion parallax to estimate distance is quite common [Srinivasan, 1993, Srinivasan and Zhang, 2000]. From (2.1) it is clear that the magnitude of the local vectors is inversely proportional to the distance — but only for translational movements. Subsequently this property is exploited by a number of insects [Collett and Harkness, 1982, Krapp, 2000] which perform 'peering movements' to induce translation and allow them to estimate depth.

Similarly, experiments on *Drosophila* also suggest that translational optical flow is used to estimate the distance to the walls in a textured flight arena [Tammero and Dickinson, 2002b]; they perform a *saccade* away from the arena wall based on the pattern of expansion.

2.1.4 Flight Behaviour

In the previous section I explained how insects process their limited sensory information into quantities like motion and depth and to gain information on the 3D structure of their environment. This section describes selected characteristic flight behaviours in insects and their use, how they were discovered, and how they are constructed from the low level responses previously outlined[14].

[14] For a comprehensive review of this field, consult Srinivasan and Zhang [2004].

This restricted selection of behaviours is not a complete sample of the biological literature. The behaviours were chosen because they have clear utility to my goal; allowing an autonomous robot to fly in indoor environments.

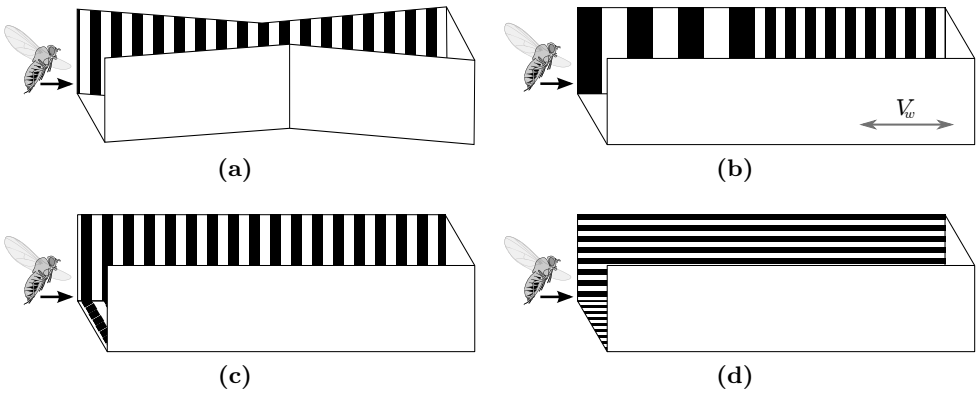


Figure 2.14: Experimental apparatus used by Srinivasan et al. to investigate a number of flight behaviours — by analysing the (in)sensitivity to spatial period of the *Apis* visual system. (a) Flight through a narrowing tunnel, insect velocity reduces with tunnel width. (b) Flight experiment investigating centering behaviour. *Apis* adjusted their horizontal position in the tunnel in response to motion of the tunnel walls, V_w, and not due to changes in the spatial frequency of the images on them. (c,d) Experiment investigating how *Apis* estimate distance flown to their food source. They do so by integrating over time, the image velocity experienced during the flight (c). However, when the tunnel image-motion cues are marked axially (d) *Apis* lose this ability. Adapted from Srinivasan et al. [2004].

2.1.4.1 Wide Field Flight Stabilization

In Section 2.1.3.2 the *optomotor response* was introduced. Through studies of the *optomotor response*, it was discovered that flies sense the direction of image movement by correlating the intensity variations registered by neighbouring *ommatidia* in the compound eye [Reichardt, 1969]. Thus, the first step of the flight stabilization pathway consist of an array of EMDs in the *lobula* that perform these correlations. Furthermore, different sets of EMDs are used to detect motion in various directions by correlating signals from *ommatidia* that are appropriately positioned relative to each other.

EMDs are connected together through motion sensitive *neurons*. Over the last 30 years, researchers have discovered multiple such *neurons* with large visual fields[15], for example, such *neuron*s were found to respond preferentially to motion in a specific direction [Hausen and Egelhaaf, 1989, Hausen, 1993], or to the fly's rotation around a specific axis [Krapp, 2000, Krapp and Hengstenberg, 1996]. It was found that these *neuron*s derive their sensitivity and selectivity by pooling signals from EMDs that have the appropriate directional selectivity in different regions of the compound eye.

[15] 'Large visual fields' is synonymous with the *neuron* being connected to many EMDs which are in-turn connected to *ommatidia* covering a wide field of view.

While, it is generally supposed [Srinivasan and Zhang, 2004] that motion-sensitive *neurons* play an important role in the insect's high level autopilot/navigation mechanism by detecting deviations from intended course and generating appropriate turning commands, the precise means by which this occurs in the brain remains elusive. Srinivasan and Zhang [2004] recognise the need for these intermediate layers to have a mediating role and put it thus;

> “Motion-sensitive *neurons* possess large visual fields, each typically covering most of one eye. Therefore, steering a straight course can only be achieved by balancing the responses of two *neurons*, each sensitive to front-to-back motion in one eye. Such a scheme works well only when the insect is flying in a symmetrically structured environment. It does not work when the insect flies along a cliff, for example, because the eye that faces the cliff experiences substantially greater image motion than does the *contralateral* eye. The only way to steer a straight course in an asymmetrical world (which is more often the rule than the exception) is to sense and compensate for image motion in only a small patch of the visual field that faces the direction along which the insect wishes to fly — the frontal visual field, for example, if the objective is to fly straight ahead.

2.1.4.2 Altitude Control

Srinivasan investigated [Srinivasan et al., 2000b, Srinivasan, 2006, Barron and Srinivasan, 2006] *Apis* altitude regulation extensively; in the context of terrain following and for landing (Section 2.1.4.6). Both strategies are based on the measurement and regulation of the angular velocity of the *ventral* optical flow field. For example, when insects are flying forward, the image of the ground sweeps backward across their *ventral* field of view, the magnitude of this image motion depends on both the forward speed, V_x, and the altitude, h.

However, in *Apis*[16] the precise relationship between altitude regulation and speed regulation is still contested [Portelli et al., 2010]. That is, do the organisms operate in a ‘altitude control’ or a ‘speed control’ flight mode. In the former the organism changes altitude to keep optical flow constant; thus changing flight speed. In the latter, *Apis* adjusts altitude to maintain the magnitude of optical flow; resulting in a change in altitude. *Apis* have also been shown to regulate altitude in response to headwinds [Riley and Osborne, 2001, Barron and Srinivasan, 2006] — although again there is some debate as to whether the parameter controlled is altitude or flight speed.

2.1.4.3 Centring Response

Using a novel experimental apparatus; a tunnel where each surface had a black-and-white grating pattern and could be moved independently (Figure 2.14) Kirchner and

[16] *Apis* are the most studied insect with respect to altitude control, other insect model organisms are less well understood

Srinivasan [1989] studied the mechanisms that insects use to balance the *contralateral* image motion.

A lower image speed on one eye caused the bee to move closer to the wall seen by that eye. A higher image speed had the opposite effect. Unlike the optomotor response, the authors showed the centring response is a more complex behaviour, demonstrating the visual system of *Apis* is capable of measuring the image velocities in the two eyes robustly and independently of spatial frequency and contrast [Srinivasan et al., 1991].

Srinivasan et al. [1991] found that when both gratings (wall) were stationary, *Apis* tended to fly along the midline of the tunnel, i.e., equidistant from the two walls (Figure 2.14c). But when one of the gratings was moved at a constant speed in the direction of the insect's flight — thereby reducing the speed of retinal image motion on one eye relative to the other — the insect's trajectories shifted toward the side of the moving grating (Figure 2.14b).

Experiments in which the contrast and the period of the gratings on the two sides were varied revealed that this centring response is robust to variations in these parameters: *Apis* continued to fly through the middle of the tunnel even when the contrast of the gratings on the two sides were substantially different or when their periods varied by a factor of as much as four [Srinivasan et al., 1991].

When one of the gratings was in motion (Figure 2.14b), the insects shifted towards or away from the moving grating (as described above) according to whether the grating moved with or against the direction of the insect's flight. These results indicate that the bees were indeed balancing the speeds of the retinal images on the two eyes and not the contrast frequencies.

2.1.4.4 Controlling Flight Speed

Using a similar experimental apparatus, Srinivasan et al. [1996] forced bees to fly through a tapered tunnel with a constant pattern on the walls (Figure 2.14a). The authors found that the insect decreased its flight speed as the tunnel narrowed. The insects did so to keep the angular velocity of the image of the walls, as seen by the eye, constant [Srinivasan et al., 1996]. Experiments also showed that when the tunnel width was doubled, the bee flew twice as fast. In the case of a uniform width tunnel, bees did not change their flight speed, even when the spatial period of the stripes lining the walls was abruptly changed [Srinivasan et al., 1996] (Figure 2.14b).

This indicates that flight speed is regulated by a visual motion-detecting mechanism that measures the angular velocity of the image independently of its spatial structure. In this respect, the speed-regulating system is similar to the system that mediates the centring response described above. Controlling flight speed by regulating image speed allows the insect to automatically slow down to a safer speed when negotiating a narrow passage or a cluttered environment.

2.1.4.5 Collision Avoidance

Figure 2.15 shows the case of an insect approaching an obstacle and the 'expansion' pattern seen by the organism. Tammero and Dickinson [2002a,b] showed that collision avoidance manoeuvres in *Drosophila* could be explained by the perception of image expansion as detected by an array of local motion detectors.

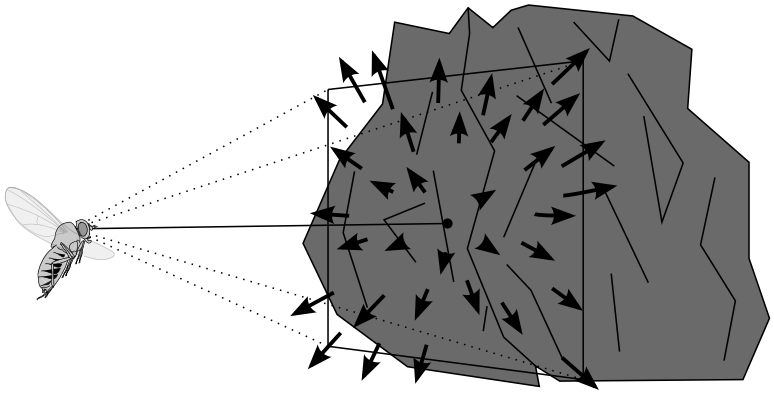


Figure 2.15: When approaching an obstacle or a surface perpendicularly, the image expands in the eye of the observer; eventually eliciting a collision avoidance response. The filled circle represents the focus-of-expansion.

Section 2.1.3.3 introduced the tangential cells in the *lobula* plate, their resemblance to matched filters, and their use for wide-field integration. However, *neurons* extracting image expansion from a *retinotopic* array of local motion detectors have not been found at the level of the *lobula* complex [Egelhaaf and Borst, 1993]. Additionally, in the cervical connective (shown in Figure 2.7a), cells are known to be sensitive to retinal image expansion. These *neurons*, which respond strongest when the insect approaches an obstacle or a potential landing site, have been proposed to be part of the neuronal circuit initiating the landing response [Borst, 1990] (for more information on the landing strategy of insects see Section 2.1.4.6).

Structurally this means that certain neural mechanisms in the *Drosophila* visual pathway are tuned to detect local image expansion, a result consistent with the detection of such *neurons* in flies [Borst, 1991].

2.1.4.6 Orchestrating Smooth Landings

The ability for insects to coordinate delicate landing responses on challenging terrain is fascinating, because for it to work well, necessitates that it conflicts with many other flight behaviours such as collision avoidance.

There are two phases of the landing process; the decision of when to initiate landing (the landing response) and the approach path to the surface (i.e., how rapidly should the insect decelerate).

Studies by Borst and Bahde [1988] and supported by Tammero and Dickinson [2002b] suggested that the initiation of the landing response depends on the spatial-frequency content and the contrast of the pattern, as well as the speed and duration of the pattern's expansion. Borst and Bahde [1988] proposed a model where the response is triggered when the time-accumulated output of an expansion-detection system, based on the correlation model [Reichardt, 1969], exceeds a preset threshold. Alternatively, Wagner [1982] proposed that *Musca domestica* may determine when to initiate a landing by calculating the time required to contact the object or surface on which they are about to land (also called time-to-contact, explained in section Section 5.4).

Srinivasan et al. [2000b] found that *Apis* tailor their landing depending on the type of surface. For example, on horizontal surfaces, bees usually perform 'grazing landings'[17]. Unlike when approaching a perpendicular surface, image expansion is weak in this scenario because the image motion is dominated by a strong front-to-back component.

Experimental analysis of landing trajectories by Srinivasan et al. [2000b] showed that the speed of flight is approximately proportional to the height above the surface. This indicates that the insect holds the angular velocity of the surface's image approximately constant as it is approached. This strategy is a simple way of decreasing the flight speed automatically and progressively, ensuring that its value is close to zero at touchdown. The strategy is plausible and advantageous; control is achieved by a simple process and without explicit knowledge of flight speed or distance from the surface.

2.1.4.7 Estimating Distance Flown

Apis mellifera have a remarkable ability to travel repeatedly between their hive and sources of food. Many people are aware of the 'waggle dance', the performance that returning honeybees put on to convey the location of food to their peers. It was originally suspected that the unit of measure encoded in the 'waggle dance', that conveyed the distance to the food source, was the total energy expended during flight[18].

This theory persisted for decades. Esch and Burns [1995, 1996], Srinivasan et al. [2000a, 1996, 1997] proposed and tested a new hypothesis; that the insects instead use visual cues to measure distance. Results showed instead that bees estimate distance flown using visual odometry; by integrating over time the image motion of the walls as registered by their eyes while they flew through the tunnel [Srinivasan et al., 1996, 1997]. For example, in one experiment the bees were unable to measure distance to food when the tunnel was marked axially (shown in Figure 2.14d). A subsequent study by Si et al. [2003] also found that the motion detecting foundation that underlies distance

[17] A grazing landing is one whereby the approach trajectory is considerably smaller than 45°.
[18] For a comprehensive review of the 'waggle dance' see Frisch [1967].

estimation seems to be robust to variations in the spatial texture and contrast of the environment.

It is not clear whether bees use optic flow information from the ventral [Esch and Burns, 1995] as well as the lateral fields of view for odometry [Si et al., 2003, Srinivasan et al., 1997]. If ventral flow is important then bees need to fly at a consistent height, or to account for the height of flight in the computation, to estimate distances reproducibly[19].

2.1.4.8 Combining Multiple Behaviours

In the insect brain, several motion-sensitive pathways exist, each with a distinct set of properties and geared to a specific visual function (such as those explained in the preceding subsections). It seems unlikely that all these systems, and other as yet undiscovered ones, operate continuously and are combined in a static manner. For example, the optomotor system has to be switched off, or its corrective commands ignored, when the insect makes a voluntary turn [Heisenberg and Wolf, 1993, Kirschfeld, 1997, Srinivasan and Bernard, 1977]. Similarly, it is also impossible to land on a surface without first disabling the collision avoidance system.

How these multiple behaviours are combined is not known, and this area remains an important and active focus of research.

2.2 COMPUTER VISION

Computer vision is a branch of computer science whose goal is to model the real world or to recognise objects from digital images. It is seeing with understanding. When we 'see', our eyes capture the image, then pass the information to the brain. The brain interprets the image and gives us the meaning of what we see.

In a computer vision system, a computer receives an image which is really just a grid of numbers from a camera or from disk. For the most part, there is no built-in pattern recognition, no automatic control of focus and aperture, and no associations with years of experience. Compared to the human vision system and despite decades of research, vision systems are still fairly naïve, depending largely on the sophistication of the computer and algorithms to make meaning of the image the camera captures.

Computer vision is also used as a tool in many other fields of research including engineering, biology, and psychology for example. Subsequently 'computer vision' has come to be known by, or associated with, a number of different terms which are worth clarifying here:

[19] The exact distance is not important, as long as two bees generate the same result.

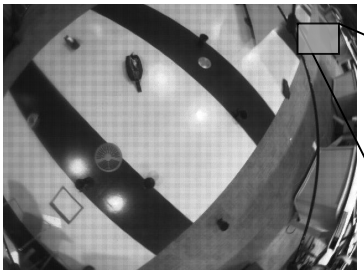
Machine vision is a somewhat outdated term which tends to refer to industrial vision applications where (usually) a single camera is used to solve a structured inspection task.

Pattern recognition refers to the recognition of structures in 2D images (usually without reference to any underlying 3D information).

Photogrammetry is the science of measurement though non-contact sensing, e.g., terrain maps from satellite images. Usually more focused on accuracy than interpretation.

Image Processing is the study of the properties of operators that produce images from other images, image filtering and related operators from image processing can be considered pre-processing steps undertaken before the more difficult task of computer vision and understanding.

Figure 2.16 shows that while humans perceive an environment or scene, what the computer 'sees' is not this information rich.



What we see;

But the computer sees this;

203	206	208	120	124	183	222	102	200	126	88	145
145	128	167	145	172	163	236	237	203	123	172	144
138	234	218	136	173	175	211	215	222	157	164	157
193	209	200	200	194	159	210	222	255	238	238	255
235	249	200	127	194	148	219	199	209	122	99	199
128	121	138	255	188	173	210	213	211	96	87	99
35	35	53	59	42	38	93	47	73	23	37	187
98	234	93	27	93	73	88	93	85	86	83	173
99	39	91	59	85	40	50	85	45	45	56	44
76	94	58	9	28	50	64	62	99	87	23	17
48	58	38	50	54	50	93	59	49	34	88	89
22	39	26	94	45	45	8	24	24	74	58	88

Figure 2.16: A comparison of perception; what humans see versus what the computer sees.

There are two types of images used in this thesis:

intensity images are images encoding light intensities; recognisable as the familiar photographs we are used to looking at. Both colour and greyscale photographs are types of intensity images. Colour images are often created by the combination of several individual intensity images of individual colours, commonly the three primary additive colours; red, green, and blue. Greyscale images are usually created by measuring the intensity of light at each pixel in a single band of the electromagnetic spectrum.

range images encode distance as intensity values and are usually captured using special optical systems, or sometimes non-visual sensors like sonar.

Any digital image, irrespective of its type, is a 2D array (matrix) of numbers as shown in Figure 2.16. This fact has two consequences [Trucco and Verri, 1998, ch. 2]:

- the exact relationship of the digital image to the physical world is determined by the acquisition process and the sensor used,
- any information contained in the images must ultimately be extracted from the 2D numerical array in which it is encoded.

This array of numbers has a large noise component and so by itself gives little information, but this matrix is all the computer 'sees'. The challenge is to turn this noisy information into perception.

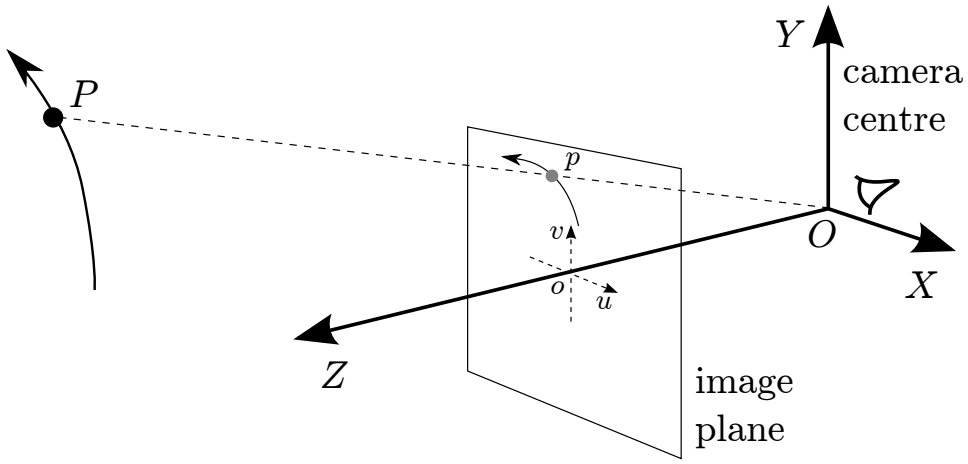


Figure 2.17: The geometry of image formation. A point P moves in 3D space and leaves a track p in the 2D image plane.

Figure 2.17 demonstrates this problem as posed thus far. Given a 2D view (image) of a 3D world, there is no unique way to reconstruct the 3D signal. Formally, such an ill-posed problem has no unique or definitive solution. The same 2D image could represent any of an infinite combination of 3D scenes, even if the data were perfect.

In the design of a practical system, additional knowledge can be used to work around the limitations imposed by visual sensors. The addition of such information, including combining inertial measurements or imposing assumptions about the world is explored in Chapter 6. For example, the former approach is also found in insects through their use of *halteres* (explained in Section 2.1.3).

This section introduces the principles of computer vision. This includes the details of intensity and depth images used in this thesis. I also describe the geometry of image formation including the geometric transformations necessary to interpret a 2D view of a 3D world.

2.2.1 Image Formation and Imaging

Any imaging system has three basic components. The process of image formation begins with rays of light from the environment entering the camera through an aperture (in the human case, pupil). The rays may be focused by an optical system such as one or more lenses. Finally, the rays strike the photosensitive device or image plane (retina) which registers light intensity.

Optical System

In order to obtain sharp images, all rays coming from a single point in the scene, P, must converge onto a single point on the image plane, p, known as 'the image of P'. If this happens, then P is 'in focus'. Focussing the image can be achieved two ways:

1. Reducing the camera's aperture down to a single point — a pinhole. This ensures that only one ray from any point in the scene can enter the camera. This creates a one-to-one correspondence between visible points and image points. The disadvantage of pinhole type imaging systems is that as the amount of light entering the system is reduced (here down to a single ray) so the time take to register enough light for a legible image increases. This long exposure time is generally impractical.
2. Introducing an optical system composed of lenses and apertures designed to make all rays coming from the same point in the scene converge onto the same image point.

In summary; Trucco and Verri [1998, p. 19] propose the goal of an optical system as:

> "An optical system can be regarded as a device that aims to produce the same image obtained by a pinhole aperture, but by means of a much larger aperture and a shorter exposure time. Moreover, an optical system enhances the light gathering power.

Thin Lenses

The principal difference between a computer vision system and a biological vision system is the lens and its effect on image formation[20] [Völkel et al., 2003]. The lens of the human eye (illustrated in Figure 2.4a) is flexible for focusing, while the lenses of the computer vision systems used in this thesis are fixed.

[20] The human eye also differs from artificial imaging systems in resolution and range. It has higher resolution; ≈ 500 Megapixels. It also has a non-uniform distribution of sensing elements (and hence resolution), higher concentrations of photoreceptor are located at the fovea. The human eye has a large dynamic range; functioning between bright sunlight and faint starlight (a range of $10^7 : 1$) through an adjustable aperture. The receptors alone have a contrast sensitivity of $10^4 : 1$ [Blackwell, 1946].

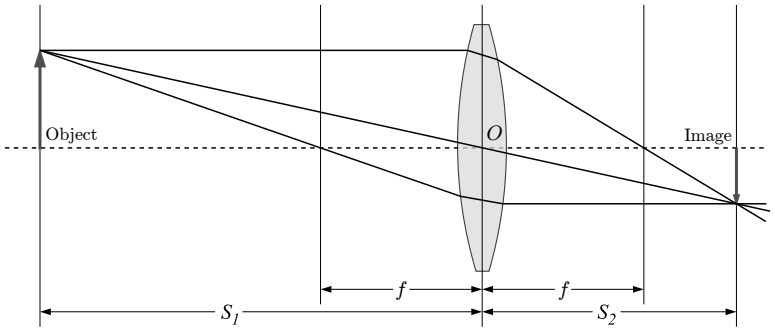


Figure 2.18: The geometry of image formation by a thin lens. Rays of light from an object pass through the lens and are focused to create an image. Figure adapted from wikia[21].

The thin lens is a simple approximation to standard optical systems[22]. Figure 2.18 shows a thin lens. The optical axis (dotted) passes through the lens centre O. In this arrangement a thin lens has two properties. A ray of light travelling parallel to the lens axis will be focused to a spot (known as the focal point) at a distance f from the lens[23]. Conversely, a point source of light placed at the focal point will be converted into a parallel beam by the lens. f is known as the focal length of the lens. From these two properties, the fundamental equation of thin lenses is:

$$\frac{1}{S_1} + \frac{1}{S_2} = \frac{1}{f}. \tag{2.2}$$

The field of view is another optical parameter that is frequently needed when comparing the properties of natural and artificial imaging systems. For a thin lens; let d be the effective diameter of the lens[24]. The field of view, w, is an angular measure of the portion of 3D space seen by the camera. It is defined as half the angle subtended by the lens diameter as seen from the focus point [Trucco and Verri, 1998, ch. 2.2],

$$\tan w = \frac{d}{2f}. \tag{2.3}$$

Figure 2.18 shows a simple 2D case of an object on the optical axis of the camera. Figure 2.17 showed a more correct, yet still contrived example; the loss of information as a point in the environment moves in 3D and we capture only its 2D projection onto the image plane (a loss of uniqueness). The problem is harder still. If perception

[21] `http://psychology.wikia.com/wiki/Lens_(optics)`

[22] More correctly, a thin lens is a lens with a thickness that is negligible compared to the focal length of the lens.

[23] This implies that different scene points will be in focus at different distances from the lens and hence at different points in the image plane. The optical lens systems of real cameras are designed so that scene points at a range of distances are imaged on or close to the image plane and therefore in focus. This range is called the 'depth of field' of the camera [Trucco and Verri, 1998, p. 21].

[24] d is called the 'effective' diameter and not the 'physical' diameter of the lens element because the aperture, placed between the scene and the lens, may prevent some rays from reaching the outside bounds of the lens.

and understanding of ones environment is the goal, then one must know where in 3D space the 2D image points lie. This requires knowledge of where one is looking (the camera orientation), the properties of the camera optics, and often simplified assumptions about the world. The following section introduces the equations linking the coordinates of points in 3D space with the coordinates of their corresponding image points.

Image Formation

Figure 2.17 shows the geometry of image formation. The aim is to link the position of points in the scene to corresponding image points. To do this one must model the geometric projection performed by the camera [Trucco and Verri, 1998, ch. 2.2]. A camera model transforms a 3D scene point $\mathbf{P} = [X, Y, Z]^\mathsf{T}$ into an image image point[25] $\mathbf{p} = [x, y]^\mathsf{T}$.

The most common model of a camera used for capturing intensity images is the 'perspective' or 'pinhole' model. From Figure 2.17; O is the 'optic centre' or 'focus of projection', the line through O and perpendicular to the image plane is the 'optical axis', the intersection of which, o, is called the 'principal point' or 'optical centre'. The Z-axis and the optical-axis are co-linear. Using the earlier definitions of P, p, and the basic equations of perspective projection[26] in the camera frame,

$$\begin{aligned} x &= f\frac{X}{Z} \\ y &= f\frac{Y}{Z}. \end{aligned} \tag{2.4}$$

This is commonly called the 'perspective camera'. Each point is scaled by its individual depth, and all projection rays converge to the optical centre. Note that (2.4) is non-linear because of the $1/f$ factor, thus it does not preserve distances between points nor angles between lines.

An affine transform is a transformation which preserves straight lines and ratios of distances. A classical approximation that turns (2.4) into linear equations is the 'affine projection weak perspective', or commonly called the 'weak perspective camera'. This model requires that the relative distance along the optical axis, δZ, between any two scene points (i.e., the scene depth) is much smaller than the average distance $\bar{Z}$ of the

[25] In the camera frame the third component of p is always equal to the focal length f, so one can write $\mathbf{p} = [x, y]^\mathsf{T}$ instead of $\mathbf{p} = [x, y, f]^\mathsf{T}$.

[26] Pinhole cameras can be modelled using two main types of projections: perspective projection, and affine projection.

points from the viewing camera[27]. In this case (2.4) becomes

$$x = f\frac{X}{Z} \approx \frac{f}{\bar{Z}}X$$
$$y = f\frac{Y}{Z} \approx \frac{f}{\bar{Z}}Y. \tag{2.5}$$

The pinhole camera model and the perspective and weak-perspective projections introduced so far are useful in practice. However, these need to be defined in an appropriate manner to allow computation using basic linear algebra operations, otherwise these would require non-linear computations. The following subsection (Section 2.2.2) describes the computations. In addition, through a more rigorous application of geometry one can map scene points to image points through an arbitrary transformation; thus the Z-axis and optical-axis need no longer be co-linear as assumed in Section 2.2.1.

2.2.2 Geometric Camera Parameters

The computer vision pipeline relates an object in space (Figure 2.17) to the image data received (Figure 2.16) through a sequence of linear algebra operations. In general, the world and pixel coordinate systems are related by a set of physical parameters such as:

- the focal length of the lens,
- the size of the pixels of the image sensor,
- the position of the principal point,
- the position and orientation of the camera.

These parameters may be classified into two categories:

Extrinsic parameters are the parameters that define the location and orientation of the camera reference frame with respect to another known world reference frame.

Intrinsic parameters are the parameters necessary to link the pixel coordinates of an image point with the corresponding coordinates in the camera reference frame.

Roughly, the flow of transforming object coordinates into pixel coordinates proceeds as shown in Figure 2.19 [Trucco and Verri, 1998, sec. 2.2–2.4].

The following subsections explain the steps in the computer vision pipeline.

Extrinsic Parameters

Extrinsic camera parameters identify the transformation between the known world reference frame (X_w, Y_w, Z_w) and the unknown camera frame (X_c, Y_c, Z_c). It is common to split this transformation into two steps, a translation $\mathbf{T}$ and a rotation R.

[27] The 'weak perspective' approximation becomes viable for $\delta Z < \bar{Z}/20$ [Trucco and Verri, 1998, p. 27].

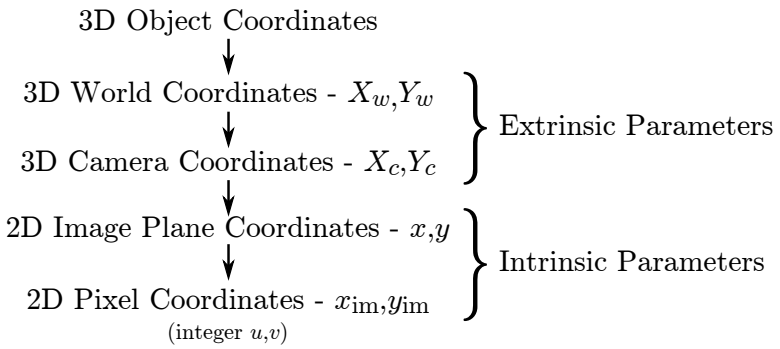


Figure 2.19: The computer vision pipeline, the standard steps for transforming the coordinates of an object in the environment into the pixel coordinates on the sensor which it is registered.

By geometric principles, determining these parameters means:

1. finding the 3D translation vector $\mathbf{T}$ between the relative positions of the origins of the two reference frames,
2. finding the 3×3 rotation matrix R that brings the corresponding axes of the two frames into alignment.

Using the extrinsic camera parameters, one can find the relation between the coordinates of a point in the world, $\mathbf{P}_w$, and the coordinates in the camera, $\mathbf{P}_c$;

$$\mathbf{P}_c = \mathrm{R}(\mathbf{P}_w - \mathbf{T}), \tag{2.6}$$

where the rotation matrix is composed of

$$\mathrm{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}. \tag{2.7}$$

Intrinsic Parameters

Intrinsic parameters characterize the optical, geometric, and digital characteristics of the camera. They extend the thin lens approximation of Section 2.2.1 and describe:

1. the perspective projection, which like the thin lens simplification, only requires the focal length f,
2. the transformation between image plane coordinates and pixel coordinates,
3. the geometric distortion introduced by the optics.

To find the transformation between image plane coordinates (x, y) and pixel coordinates $(x_{\text{im}}, y_{\text{im}})$ — sometimes called the 'image reference frame', consider Figure 2.17,

$$\begin{aligned} x &= -(x_{\text{im}} - o_x)s_x \\ y &= -(y_{\text{im}} - o_y)s_y, \end{aligned} \tag{2.8}$$

where (o_x, o_y) are the coordinates of the principal point, o, in pixels; $o_x = N/2, o_y = M/2$ is the principal point at the centre of the image. Also, s_x, s_y correspond to the effective size of the pixels in the horizontal and vertical directions (in millimetres). Therefore, f, o_x, o_y, s_x, s_y encapsulate the intrinsic parameters.

In some cases the optics introduce image distortions, often evident at the periphery of the image. Correcting for these distortions in order to recover the true values of $(x_{\text{im}}, y_{\text{im}})$ requires the estimation of additional coefficients; the radial distortion coefficients k_1, k_2, k_3, and the tangential distortion coefficients p_1, p_2. Correcting for these distortions is explained in Appendix D.2.

Camera Models Revisited

One can now link the pixel coordinates of image point with the world coordinates of the corresponding 3D point without explicit reference to the camera reference frame needed by (2.4). Substituting (2.6) and (2.8) into (2.4) yields

$$\begin{aligned} -(x_{\text{im}} - o_x)s_x &= f\frac{\mathbf{R}_1^\top(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^\top(\mathbf{P}_w - \mathbf{T})} \\ -(y_{\text{im}} - o_y)s_y &= f\frac{\mathbf{R}_2^\top(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^\top(\mathbf{P}_w - \mathbf{T})}, \end{aligned} \tag{2.9}$$

or

$$\begin{aligned} x_{\text{im}} &= fs_x\frac{\mathbf{R}_1^\top(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^\top(\mathbf{P}_w - \mathbf{T})} + o_x \\ y_{\text{im}} &= fs_y\frac{\mathbf{R}_2^\top(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^\top(\mathbf{P}_w - \mathbf{T})} + o_y, \end{aligned} \tag{2.10}$$

where $\mathbf{R}_i^\top, i = 1, 2, 3$ is a 3D vector formed by the i-th row of the rotation matrix R.

Rewriting (2.9) as a matrix product and defining two M_{im} and M_{ex}, as

$$\text{M}_{\text{im}} = \begin{bmatrix} -f/s_x & 0 & o_x \\ 0 & -f/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.11}$$

and

$$\mathrm{M}_{\mathrm{ex}} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -\mathbf{R}_1^{\mathsf{T}}\mathbf{T} \\ r_{21} & r_{22} & r_{23} & -\mathbf{R}_2^{\mathsf{T}}\mathbf{T} \\ r_{31} & r_{32} & r_{33} & -\mathbf{R}_3^{\mathsf{T}}\mathbf{T} \end{bmatrix}, \tag{2.12}$$

means the 3×3 matrix M_{im} contains only the intrinsic camera parameters and the 3×4 matrix M_{ex} contains only the extrinsic camera parameters. Expressing $\mathbf{P}_w$ in homogeneous coordinates $(x_h, y_h, w)^{\mathsf{T}}$ one obtains a linear matrix expression describing the perspective projection,

$$\begin{bmatrix} x_h \\ y_h \\ w \end{bmatrix} = \mathrm{M}_{\mathrm{im}}\mathrm{M}_{\mathrm{ex}} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{2.13}$$

or $\mathrm{M} = \mathrm{M}_{\mathrm{im}}\mathrm{M}_{\mathrm{ex}}$, where M is called the projection matrix

$$\mathrm{M} = \begin{bmatrix} m_{11} & m_{12} & m_{13} & m_{14} \\ m_{21} & m_{22} & m_{23} & m_{24} \\ m_{31} & m_{32} & m_{33} & m_{34} \end{bmatrix}. \tag{2.14}$$

The components of the vector $(x_h, y_h, w)^{\mathsf{T}}$ can be expressed as ratios of image coordinates

$$\begin{aligned} x_{\mathrm{im}} &= \frac{x_h}{w} \\ y_{\mathrm{im}} &= \frac{y_h}{w}. \end{aligned} \tag{2.15}$$

Substituting (2.15) and (2.14) into (2.13) and expressed to give the pixel coordinates directly,

$$\begin{aligned} x_{\mathrm{im}} = \frac{x_h}{w} &= \frac{m_{11}X_w + m_{12}Y_w + m_{13}Z_w + m_{14}}{m_{31}X_w + m_{32}Y_w + m_{33}Z_w + m_{34}} \\ y_{\mathrm{im}} = \frac{y_h}{w} &= \frac{m_{21}X_w + m_{22}Y_w + m_{23}Z_w + m_{24}}{m_{31}X_w + m_{32}Y_w + m_{33}Z_w + m_{34}} \end{aligned} \tag{2.16}$$

The relation of 3D points and their 2D projections can be seen as a linear transformation from the projective space $(X_w, Y_w, Z_w, 1)^{\mathsf{T}}$ to the projective plane $(x_h, y_h, w)^{\mathsf{T}}$ and ultimately to pixel coordinates (up to an arbitrary scale factor).

The projection matrix M is constructed according to the camera model chosen. The selection of the camera models of Section 2.2.1 and their respective projection matrices is described in Appendix D.1.1.

Note, for the rest of this thesis, because I deal primarily with image plane coordinates, I use (u, v) instead of $(x_{\text{im}}, y_{\text{im}})$ for clarity.

2.2.3 Depth Images

A single intensity image does not provide the distance to, nor the shape of obstacles directly, yet if one wishes to avoid obstacles, the distance to these obstacles is important. Pixel values are related to depth only indirectly; through illumination of the scene and the optical and geometric properties of the imaging system previously discussed. While Chapter 6 uses the computation and use of depth from optical flow, depth can also be measured directly if using the correct sensors; for example range sensors as used in Section 6.2. This section introduces range sensors and their properties.

Several types of sensors can produce range images:

structured light sensors illuminate the scene with a specially designed structured light pattern and then capture an intensity image of the scene using a conventional camera. The structured light can be in the form of horizontal and vertical lines, points, or geometric patterns. Depth is then determined from a single image of the reflected light by searching for the projected pattern and considering the geometric constraint between the projector and the image sensor. The Microsoft Kinect sensor used in Chapter 6 uses structured light; a projected infrared dot pattern.

time-of-flight sensors measure distance using the time-of-flight technique (similar to radar/sonar). They use a pulse of light and a custom image sensor equipped to measure with precise accuracy, the arrival time and/or phase of the reflected light pulse.

moire interferometry range sensors project light from two gratings featuring regular linear patterns onto the scene. They measure relative distance using the phase difference from the interference pattern visible in the captured image.

stereo triangulation range sensors use two or more calibrated image sensors, positioned relative to one-another at known geometry. By finding corresponding features in each sensor's image they estimate distances to these features using triangulation.

This was only a selection of range sensors, all image based and producing a 2D matrix of depth estimates. Because the sensors are image based, many of the same geometric properties described in the preceding section still apply (extrinsic parameters obviously, but some intrinsic ones too; principle point, lens distortion, etc).

The use of range information obtained from a Microsoft Kinect camera is discussed in Chapter 6. The calibration of the camera and its geometric properties are included in Appendix A.3.2.

2.3 SUMMARY

This chapter introduced a number of concepts, in particular those relating to the insect visual and mechanosensory system.

The remainder of this thesis concerns the application of biologically inspired visual principles to the control of flying robots. One should read the thesis with the following concepts in mind:

- High level behaviours are usually the amalgamation of low level information processing, sometimes combining image motion with other sensory inputs.
- High level behaviours typically integrate information covering a wide spatial range.
- Low level processes often sample the world in efficient and evolutionarily specialized ways via the connection patterns of *neurons* in the brain.
- Some dynamic combination or mediation of high level behaviours occurs in insects, yet the mechanism is not completely known. This is an active topic of biological research.
- There is great commonality between the different yet well studied insect model organisms; both at the behavioural level, and when comparing the physical structure of their brains.

Chapter 3

OPTICAL FLOW

Section 2.1 introduced many ways that insects use image motion to control their flight and navigate their environment. This chapter explains the mathematical foundations for calculating optical flow (Section 3.1) and the relevant techniques for doing so used in this thesis (Section 3.2). This chapter also describes a phase-correlation based optical flow technique I developed (Section 3.3).

Figure 2.17 shows the scenario. Each point moves along a three-dimensional (3D) path in space. When projected onto the image plane each point now produces a two-dimensional (2D) path. The instantaneous rate-of-change of the position of the point along this path is its velocity [Fleet and Weiss, 2005]. The 2D velocities for all visible points is often referred to as the; 2D motion field, image velocity, or image motion — the specific meaning of these terms is discussed in Section 3.1. The goal of optical flow estimation is to compute an approximation to the motion field from a sequence of intensity images.

Provided that optical flow is a reliable approximation to 2D image motion, it can then be used to recover the 3D motion of the visual sensor (to within a scale factor) and the 3D surface structure (shape or relative depth) of the environment. To do so requires assumptions concerning the structure of the optical flow, the 3D environment, and the motion of the sensor. Optical flow may also be used to perform motion detection, object segmentation, time-to-contact and focus-of-expansion (FOE) calculations, motion compensated encoding, and stereo disparity measurement [Beauchemin and Barron, 1995, Aires et al., 2008].

3.1 FORMULATION

The initial hypothesis in measuring image motion is that the intensity structures of local time-varying image regions are approximately constant under motion for at least a short duration [Horn and Schunck, 1981]. Formally, if $I(x, y, t)$ is the image intensity

function, with x and y spatial dimensions and t, time, then

$$I(x, y, t) \approx I(x + \delta x, y + \delta y, t + \delta t), \tag{3.1}$$

where $\delta x, \delta y$ is the displacement of the local image region after time δt (Figure 3.1).

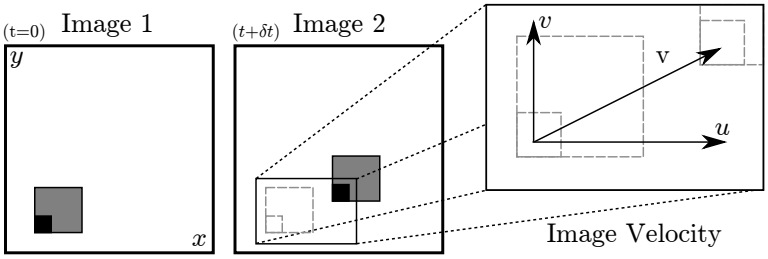


Figure 3.1: The geometry of the image intensity and image velocity vector.

Expanding the right hand side of this equation in a Taylor series yields

$$I(x + \delta x, y + \delta y, t + \delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + O^2 \tag{3.2}$$

where O^2 represents the second and higher order terms which are assumed negligible. Subtracting $I(x, y, t)$ on both sides, ignoring O^2 and dividing by δt yields

$$I_x u + I_y v + I_t = 0, \tag{3.3}$$

where $\mathbf{v} = (v_u, v_v)$ is the image velocity and $\nabla\mathbf{I} = (I_x, I_y)$ is the spatio-temporal, or partial derivatives of $I(x, y, t)$. Equation (3.3) is the 'optical flow constraint equation'; it is one linear equation in two unknowns $\mathbf{v} = (v_u, v_v)$.

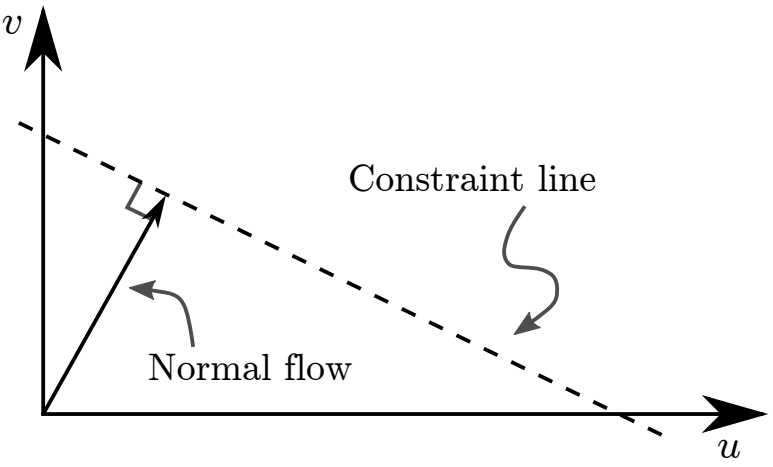


Figure 3.2: The optical flow constraint equation. At a single image pixel we get $I_x u + I_y v = -I_t$, yielding the constraint line $I_x u + I_y v + I_t = 0$. With one point we can only detect movement perpendicular to the brightness gradient; the normal flow $\mathbf{v}_\perp$. Figures adapted from Beauchemin and Barron [1995].

Figure 3.2 shows the formulation; the optical flow constraint equation is ill-posed. For one pixel, both components of $\mathbf{v}$ cannot be computed, only the normal velocity in the direction of the spatial gradient, $\mathbf{v}_\perp$. From (3.3) one can write,

$$\mathbf{v}_\perp = \frac{-I_t \nabla I}{\| \nabla I \|^2}. \tag{3.4}$$

This is known as the 'aperture problem'. In other words, a variety of contours of different orientations moving at different speeds can cause identical responses (i.e., optical flow does not represent the motion field). A real-world example of this is the 'barber-pole illusion'; it appears as though the stripes on a diagonally striped pole are moving in the direction of its vertical axis instead of around it.

The solution is to impose other constraints; such as to assume flow is consistent locally and consider the surrounding pixels. The manner in which this is done distinguishes different optical flow implementations, not surprisingly, this is similar to nature. *Ommatidia* respond to motion that occurs locally within their visual field, but because each local motion-detecting *neuron* suffers from the aperture problem, the estimates from many *neurons* need to be integrated into a global motion estimate.

Subsequently it becomes clear that for optical flow to be exactly equal to image motion a number of conditions have to be satisfied:

1. uniform illumination,
2. *Lambertian* surface reflectance,
3. pure translation parallel to the image plane,
4. sufficient intensity structure[1].

In practice these conditions are never entirely satisfied in scenery. Instead it is assumed that these conditions hold locally in the scene and therefore, locally on the image plane.

3.2 OPTICAL FLOW TECHNIQUES

There are a number of algorithms for calculating optical flow, separated into classes by similarity. In the following sections I shall summarise the classes of optical flow techniques. The boundaries between each class of methods are not always clear, however, the basic classes [Beauchemin and Barron, 1995, Barron et al., 1994] are:

1. intensity-based differential methods,
2. multi-constraint methods,
3. frequency-based methods,

[1] Imagine negotiating a world where everything is uniformly white; there would be no brightness gradient (nor any differential information) to detect.

4. correlation based methods,
5. multiple motion methods,
6. temporal refinement methods.

3.2.1 Differential Methods

Differential methods compute image velocity from spatiotemporal derivatives of image intensities, solving (3.3). Because (3.3) is ill-posed, an exact solution is not possible, so differential methods can be further split into global or local forms based on the concessions they make in order to compute optical flow.

Global methods apply additional constraint to (3.3), usually a smoothness regularization term, to compute dense optical flows over large image regions. Combined, they form a function which is minimized over the image domain. Notably popular global methods include *Horn and Schunck*.

Local methods use normal velocity information in local neighbourhoods to perform a least squares minimization to find the best fit for $\mathbf{v}$. Notably popular methods include; *Lucas and Kanade* and *Simoncelli*.

Horn and Schunck

The most well known global method was first introduced by Horn and Schunck [1981]. One advantage of the Horn-Schunck algorithm is that it yields a high density of flow vectors; when flow information missing in inner parts of homogeneous objects, it is filled in from the motion boundaries. On the negative side, it is more sensitive to noise than local methods.

The Horn-Schunck algorithm assumes smoothness in the flow over the whole image. Thus, it tries to minimize distortions in flow and prefers solutions which show more smoothness. The flow is formulated as a global energy functional which is then sought to be minimized.

These constraints were used to define an error function

$$\int_D \Big((\nabla I \cdot \mathbf{v} + I_t)^2 + \lambda^2 \ \mathrm{tr}((\nabla \mathbf{v})^\mathsf{T} (\nabla \mathbf{v})) \Big) \, \mathrm{d}\mathbf{x}, \tag{3.5}$$

over a domain of interest D, where $\mathbf{v} = (v_u, v_v)$. The solution for $\mathbf{v}$ is given as a set of Gauss-Seidel equations which are solved iteratively[2]. Uniform illumination over D, orthographic projection, and pure translational motion parallel to the scene are all conditions that must be met for the brightness consistency assumption to be satisfied.

These extensive conditions reduce the set of admissible visual events considerably and have driven research into constraints generating more applicable equations.

[2] The Gauss-Seidel method is an iterative method used to solve a linear system of equations. It is similar to the more common Jacobi method.

Lucas and Kanade

Local models of velocity assuming single motion patterns are also common. For example, the well known *Lucas and Kanade* [Lucas and Kanade, 1981, Lucas, 1984] algorithm uses a local constant model for $\mathbf{v}$. This is then solved as a weighted least squares solution to (3.3). Image velocity estimates are computed by minimizing

$$\sum_{\mathbf{x}\in R} W^2(\mathbf{x})(\nabla I(\mathbf{x},t)\cdot\mathbf{v}+\mathbf{I}_t(\mathbf{x},t))^2, \tag{3.6}$$

where $W(\mathbf{x})$ denotes a window function and R is a spatial neighbourhood. This weighted least-squares approach makes the algorithm comparatively robust in the presence of noise.

The *Lucas and Kanade* algorithm, like all local optical flow algorithms, does not yield a high density of flow vectors. The flow information fades out quickly across motion boundaries and the inner parts of large homogeneous areas show little or no motion [Lucas and Kanade, 1981].

The most popular variation on the *Lucas and Kanade* algorithm, and the one I use frequently in this thesis, is the pyramidal Lukas-Kanade implementation. This utilizes two related extensions to *Lucas and Kanade*;

1. Consider the assumption of small motion and the higher order terms of Figure 3.2. This is a polynomial root finding problem of which the plain *Lucas and Kanade* does one iteration of the Newton-Raphson method. To get better results in practice it is necessary to iterate multiple times on this scheme [Lucas and Kanade, 1981].
2. Consider the role of R. A small window over which to solve yields a dense flow field and does not 'smooth out' subtle differences in the motion field. However, this small window makes the algorithm prone to errors due to changes in lighting. Furthermore, in order to handle large motions, is is preferable to pick a large integration window. To solve this trade-off a multi-resolution approach is taken. The *Lucas and Kanade* algorithm is run against the target at different resolutions ('levels of a pyramid') and the intermediate results are propagated to the next level [Lucas and Kanade, 1981].

In practice window sizes of $3-7$ pixels and $2-4$ pyramidal levels (for a moderate resolution image) yield acceptably dense and accurate flow vectors for experimentation.

Another popular improvement to *Lucas and Kanade* is the feature based extension by Tomasi and Kanade [1991]. In this version a feature detection algorithm (such as Harris corners) is first run on the image. Subsequently, *Lucas and Kanade* is only run on patches surrounding these features — where by definition a detectable local gradient is present. I do not use this variation as it yields an unequal spatial distribution of

optical flow which I believe is not consistent with the regular arrangement of motion detection in insect's compound eyes.

3.2.2 Correlation-Based Methods

Numerical differentiation is sometimes impractical because of a small number of frames or poor signal to noise ratio [Barron et al., 1994]. In these situations differential or frequency approaches may not be suitable and it is wise to consider correlation based methods.

Typically, features suitable for matching such as corner points are sparse, while poor, easily matched features such as edges are dense. Even when reasonably unique features are available, establishing the correct correspondences can be problematic. In fact, a large part of the computer vision field is tasked with this problem of feature description and correspondence.

Correlation based matching approaches are less sensitive to these problems. They do not rely on the presence of significant image features and variable correlation window sizes can be used. These approaches define displacement (which is an approximation to velocity) as a shift that yields the best fit between contiguous time varying image regions.

Matching image regions often amounts to maximising a similarity measure. In particular, a correlation coefficient between two functions f and g is defined as the integral of their product

$$\int_D f(x+\delta x)g(x)\mathrm{d}x. \tag{3.7}$$

Finding δx amounts to finding the shift between f and g if $f(x+\delta x) = g(x)$.

The most naive correlation-based approach is the block matching (BM) method. In BM, a region of the current frame is placed and moved around in the previous frame using a specific search strategy. Criterion such as sum of squared differences (SSD) or sum of absolute differences (SAD) may be used as the correlation measure for determining if the region under analysis matches a corresponding region in the previous frame.

3.3 PHASE-CORRELATION FOR OPTICAL FLOW

Whereas the block matching (BM) method compares blocks by luminance matches, phase-correlation methods measure the relative shift between two images directly from their phases; typically by means of a normalised cross-correlation function computed in the 2D spatial Fourier domain [Pla and Bober, 1997]. In general where luminance based methods are susceptible to global changes in illumination; Fourier transform based implementations are not [Kuglin and Hines, 1975, Pla and Bober, 1997].

However, typical phase-correlation techniques are still computationally expensive; especially for large images as they require two 2D Fourier transforms, an inverse 2D Fourier transform, a search for the correlation peak, and a sub-pixel interpolation step [Pla and Bober, 1997].

In this section I present a new phase-correlation based technique for calculating optical flow; the shear-average phase-gradient correlation. Compared to existing methods the proposed shear-average technique removes the need for the inverse 2D Fourier transform, interpolation, and search steps.

This section proceeds as follows. Section 3.3.1 explains the principle of frequency domain image correlation. Section 3.3.1.1 introduces existing phase-correlation techniques. Section 3.3.2 explains the new shear-average algorithm for efficient optical flow estimation. Section 3.3.3 demonstrates the performance of the new approach compared with existing techniques.

3.3.1 Frequency Domain Image Correlation

The goal of image correlation is, given two images, find the displacement that maximises their similarity.

Let $\mathcal{F}_1[u, v]$ be the 2D spectrum of a 2D image $f_1[x, y]$. By convention and for the remainder of this section, (u, v) denote the spatial frequencies of $\mathcal{F}_k(\cdot)$ (and not pixel coordinates as in the rest of this thesis).

Let's denote the image f_1 as a windowed version of a larger image f,

$$f_1[x, y] = f(x - x_1, y - y_1)a(x - x_1, y - y_1), \tag{3.8}$$

where a is the aperture function centred on (x_1, y_1). Similarly, the second image f_2 can be represented as

$$f_2[x, y] = f(x - x_2, y - y_2)a(x - x_2, y - y_2). \tag{3.9}$$

Here square brackets denote the property that $f[x + mN_x, y + nN_y] = f[x, y]$ for arbitrary integers m and n. The second image can be denoted in terms of the first image as

$$\begin{aligned} f_2[x, y] = f_1[x - \Delta x, y - \Delta y]w(x, y|\Delta x, \Delta y) \\ +c(x, y|\Delta x, \Delta y), \end{aligned} \tag{3.10}$$

where $c(x, y|\Delta x, \Delta y)$ denotes the contamination due to the non-overlapping region and $w(x, y|\Delta x, \Delta y)$ denotes the common region viewed by the two images. In the Fourier

domain (3.10) transforms to

$$\mathcal{F}_2[u,v] = \mathcal{F}_1[u,v]\exp\left(-\mathrm{j}2\pi\left(\frac{u\Delta x}{N_x}+\frac{v\Delta y}{N_y}\right)\right) \odot W[u,v|\Delta x,\Delta y] + C[u,v|\Delta x,\Delta y], \tag{3.11}$$

where $\odot$ denotes circular convolution. If the shifts are small so that the overlap is large, this can be approximated by

$$\mathcal{F}_2[u,v] \approx \mathcal{F}_1[u,v]\exp\left(-\mathrm{j}2\pi\left(\frac{u\Delta x}{N_x}+\frac{v\Delta y}{N_y}\right)\right). \tag{3.12}$$

Forming the Hermitian product of the spectra yields

$$\begin{aligned} H[u,v] &= \mathcal{F}_2[u,v]\mathcal{F}_1^*[u,v], \\ &\approx |\mathcal{F}_1[u,v]|^2\exp\left(-\mathrm{j}2\pi\left(\frac{u\Delta x}{N_x}+\frac{v\Delta y}{N_y}\right)\right). \end{aligned} \tag{3.13}$$

Taking the inverse Fourier transform of $H[u,v]$ one gets the cross correlation of the two images (from the Fourier correlation theorem);

$$h[x,y] \approx ff_1[x-\Delta x, y-\Delta y], \tag{3.14}$$

where $ff_1[x,y]$ is the autocorrelation of $f_1[x,y]$ — given by the inverse Fourier transform of $|\mathcal{F}_1[u,v]|^2$. Since an autocorrelation function is always maximum at zero lag, the position of the peak in $h[x,y]$ gives the image shift Δx, demonstrated in Figure 3.3(b).

A correlation coefficient gives an indication of the reliability of the result. This is found by normalising the correlation by the square root of the product of the image energies,

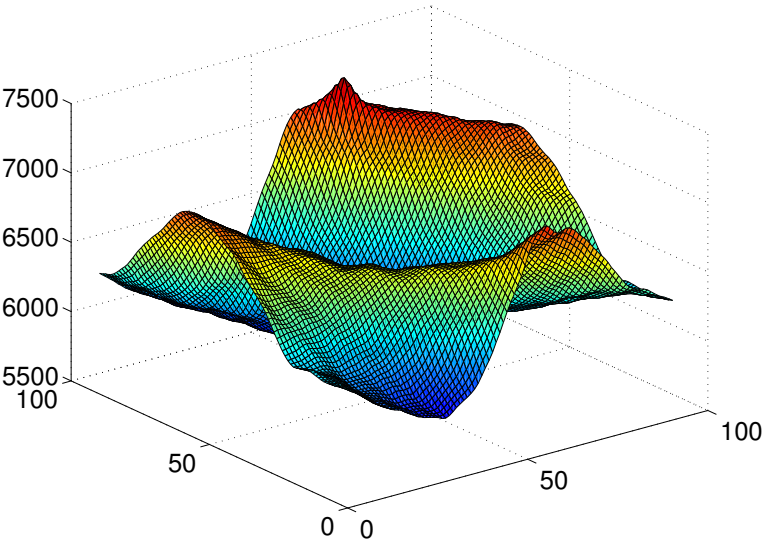
$$\rho[x,y] = \frac{h[x,y]}{\sqrt{ff_1[0,0]ff_2[0,0]}}. \tag{3.15}$$

3.3.1.1 Phase-Correlation

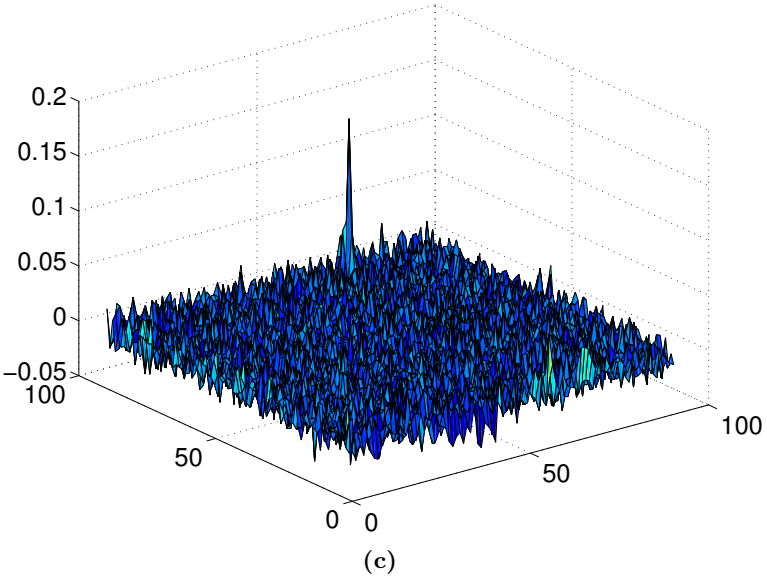
Kuglin and Hines [1975] introduced the concept of phase-correlation. This uses only the Fourier domain phase information to estimate the shift between images. When applied globally, phase-correlation is a useful technique for image registration as the method is less sensitive to scale and rotation [Kuglin and Hines, 1975]. When applied locally, phase-correlation can also be used for optical flow estimation [Fleet and Jepson, 1993, Pla and Bober, 1997] and in template matching operations [Zhang et al., 2009].

Phase-correlation normalises the Hermitian product of the image spectra H by its

(a)



(b)



(c)

Figure 3.3: (a) An image (left) and its shifted counterpart with noise. (b) The normalised cross-correlation of the two images. (c) The phase-correlation of the same two images.

magnitude $|H|$,

$$G[u,v] = \frac{H[u,v]}{|H[u,v]|} = \exp\left(\mathrm{j}\Psi[u,v]\right). \tag{3.16}$$

Here $\Psi = \arg H$ is the phase of H. The image displacement is then estimated from the shift of the peak in the inverse Fourier transform of G, demonstrated in Figure 3.3(c). The normalising step is equivalent to an inverse filter and thus enhances the bandwidth of H but requires images with a high signal to noise ratio.

Foroosh et al. [2002] extended the phase-correlation concept to subpixel accuracy and achieved improved results with a similar computational cost. They noticed that the signal power in the phase-correlation is not concentrated in a single peak but rather in several coherent peaks mostly adjacent to each other. From this they developed an analytic expression for the sub-pixel estimation of the correlation peak.

3.3.2 Shear-Average (Phase-Gradient Correlation)

The shear-average[3] algorithm [Fienup, 1989, 2001] was developed for use in synthetic aperture radar for phase error detection and has been applied in the synthetic aperture sonar field [Callow et al., 2001, Johnson et al., 1995]. The algorithm relies on the shared information in two consecutive pings[4] and works by exploiting the redundant information present.

It uses the observation that an impulse, shifted from the origin has a phase gradient in the Fourier domain proportional to the shift (shown in Figure 3.4). Thus by estimating the phase gradient directly, no inverse fast Fourier transform (FFT) and peak search is required. I propose applying the shear-average technique to estimate image shifts and optical flow.

The average phase gradient can be estimated by averaging the sheared Hermitian product of H [Fienup, 2001, 1989]. The average phase gradient in the u spatial frequency direction is

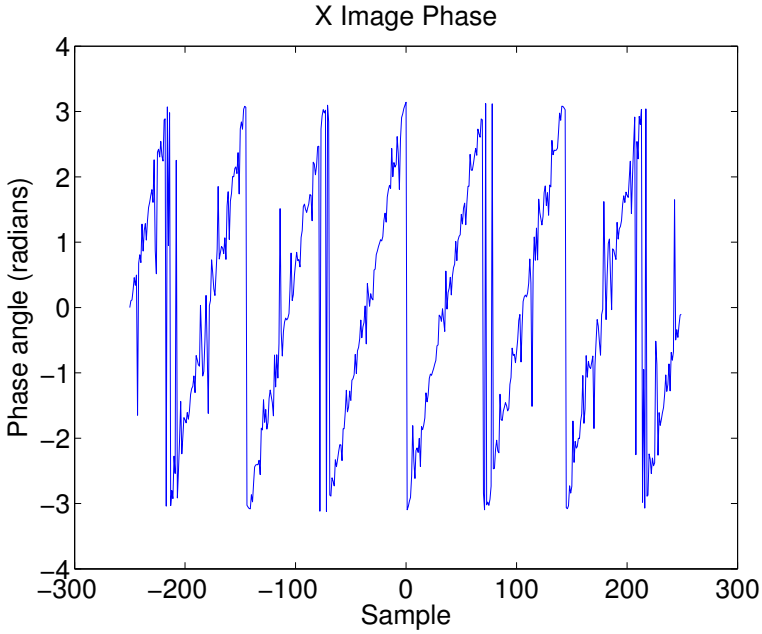
$$\Delta\theta_u = \arg\left\{\sum_{v=0}^{N_{y-1}}\sum_{u=0}^{N_{x-2}} H[u+1,v]H^*[u,v]\right\}. \tag{3.17}$$

The average phase gradient gives the estimated shift in the x direction,

$$\Delta x = -\frac{N_x \Delta\theta_u}{2\pi}. \tag{3.18}$$

[3] As named in accordance with the mathematical transform, shear.
[4] Or in computer vision; images.

(a)



(b)

Figure 3.4: A demonstration of the phase wraps and the phase gradient principle. (a) The source and shifted (by $(8, 4)$ pixels) images. (b) The phase, Ψ, of the Hermitian product, H, of image spectra along the u-axis. Notice how the 8 phase wraps (around $\pm\pi$) correspond to the original image having a shift of $\Delta x = 8$ pixels.

Similarly, the average phase gradient in the v spatial frequency direction is

$$\Delta\theta_v = \arg\left\{\sum_{v=0}^{N_{y-2}}\sum_{u=0}^{N_{x-1}} H[u, v+1]H^*[u, v]\right\}. \tag{3.19}$$

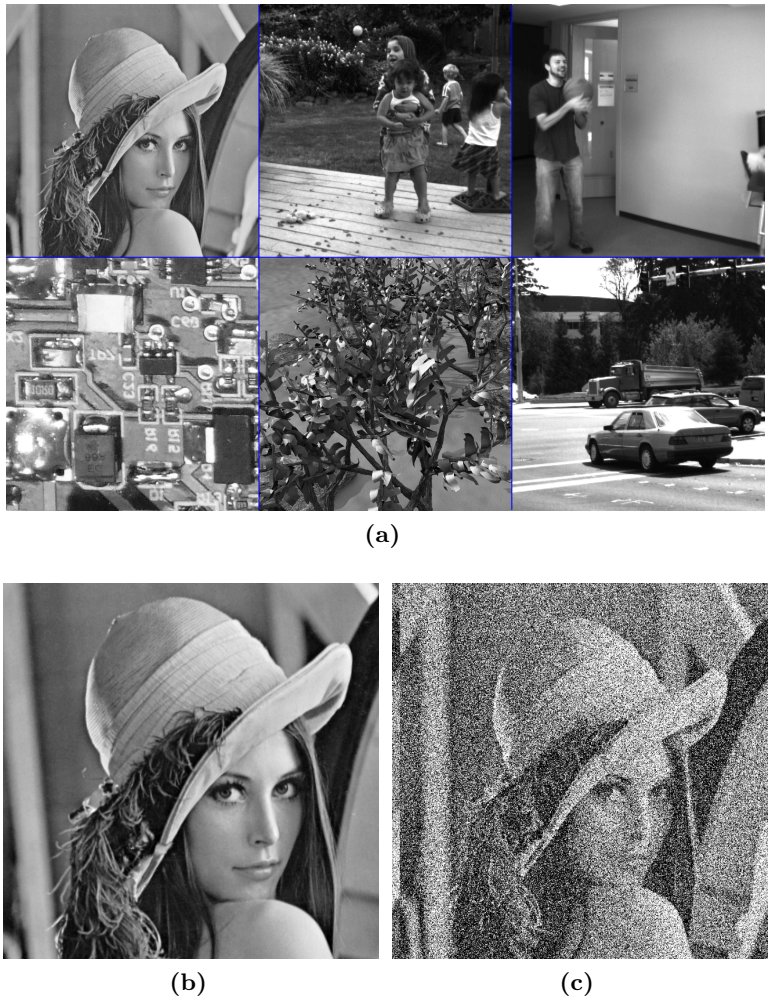
From this the estimated shift in the y direction is

$$\Delta y = -\frac{N_y \Delta\theta_v}{2\pi}. \tag{3.20}$$

The problem with this approach is that when the shift becomes large the phase is contaminated by the dissimilar parts of the images.

3.3.3 Performance

The shear-average was compared with other frequency domain correlation based algorithms; basic correlation (3.14) with and without sub-pixel interpolation, the original phase-correlation implementation of Kuglin and Hines [1975], and the improved subpixel based implementation of Foroosh et al. [2002]. The test images shown in Figure 3.5a were used in the comparison. The ranges of image shifts and rotations tested were selected based on the conventions of the computer vision field[5], however, with larger ranges because of the lower framerate and rapid dynamics of the 'wasp' quadrotor.



(a)

(b) **(c)**

Figure 3.5: The test procedure for evaluating the shear-average algorithm. (a) The set of images used for all experiments. (b) Image showing a shift of 15 pixels in x with a 5° rotation applied. (c) Image showing a shift of 15 pixels in x with noise added.

In the first test, the image was shifted by 2,5,10, and 15 pixels and white Gaus-

[5] Most optical flow review papers use the Middlebury [Baker et al., 2007] sequences (Appendix C) which have translational motion < 4 pixels per frame [Barron et al., 1994, Sec. 3]. Normal ranges of rotations were less clear. Roth and Black [2007] implemented an optical flow technique with good rotation invariance and evaluated many common algorithms against rotations $< \pm 1.5°$.

sian noise (0-mean) with increasing variance was added. The range of variance gave shifted noisy images with signal to noise ratios (SNRs) of 15 dB to 50 dB. The test was repeated 100 times at each noise variance and the results are shown in Figure 3.6.

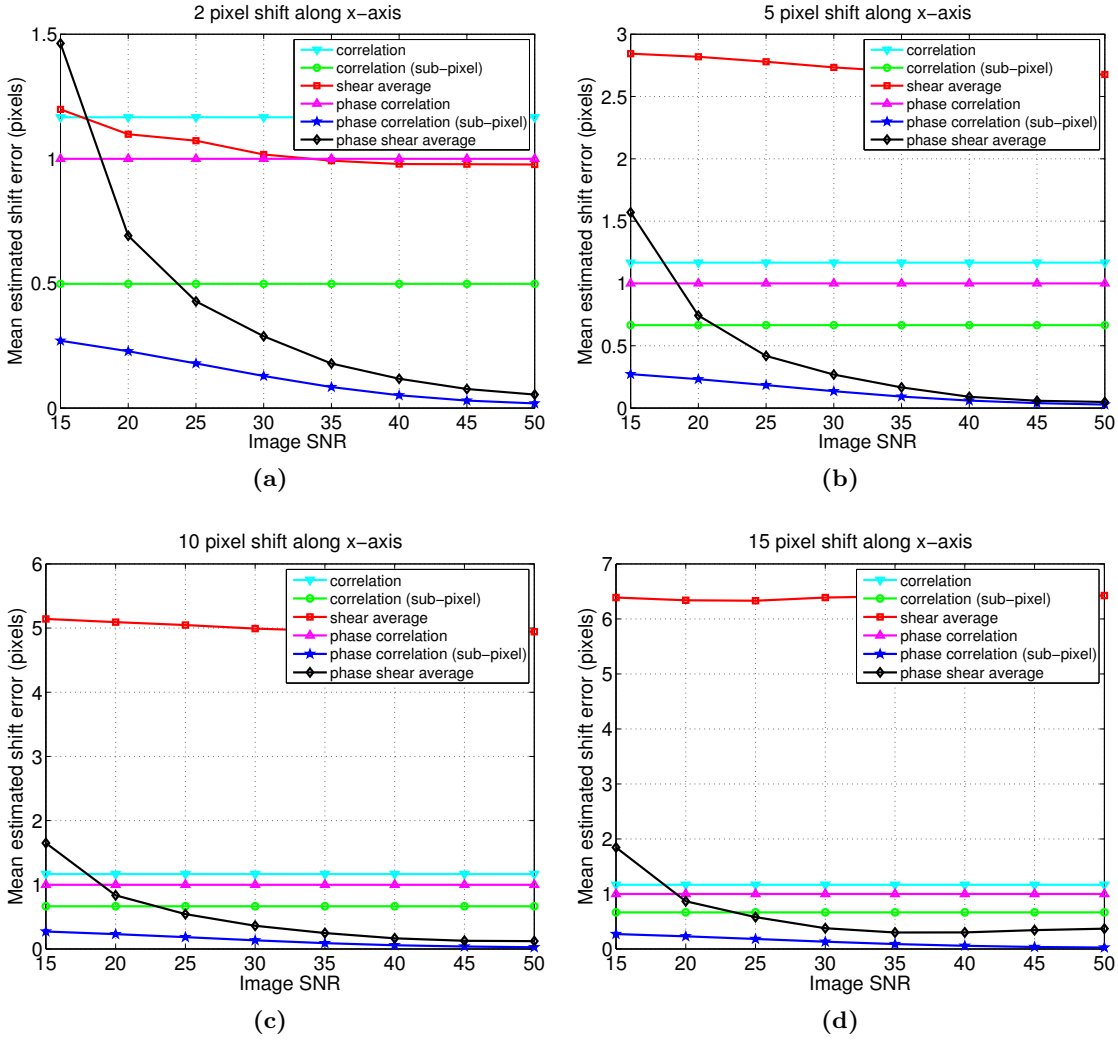


Figure 3.6: Error in estimated image shift for increasing additive noise (average of 100 runs), for shifts of 2,5,10, and 15 pixels.

In the second test, the image was once again shifted by 2,5,10, and 15 pixels, however, this time an additional rotation was applied. The rotation step also involves an interpolation, so this represents a more challenging situation for the algorithm, as the interpolation may introduce information in the transformed image that was not in the original, indistinguishable from noise. Results are shown in Figure 3.7.

Figure 3.6 demonstrates that the shear-average algorithm is able to estimate, to a subpixel accuracy, the translation of the images, just slightly worse than the Foroosh et al. [2002] implementation. The naïve phase-correlation implementation is unable to estimate translation at better than a pixel accuracy for the translations tested.

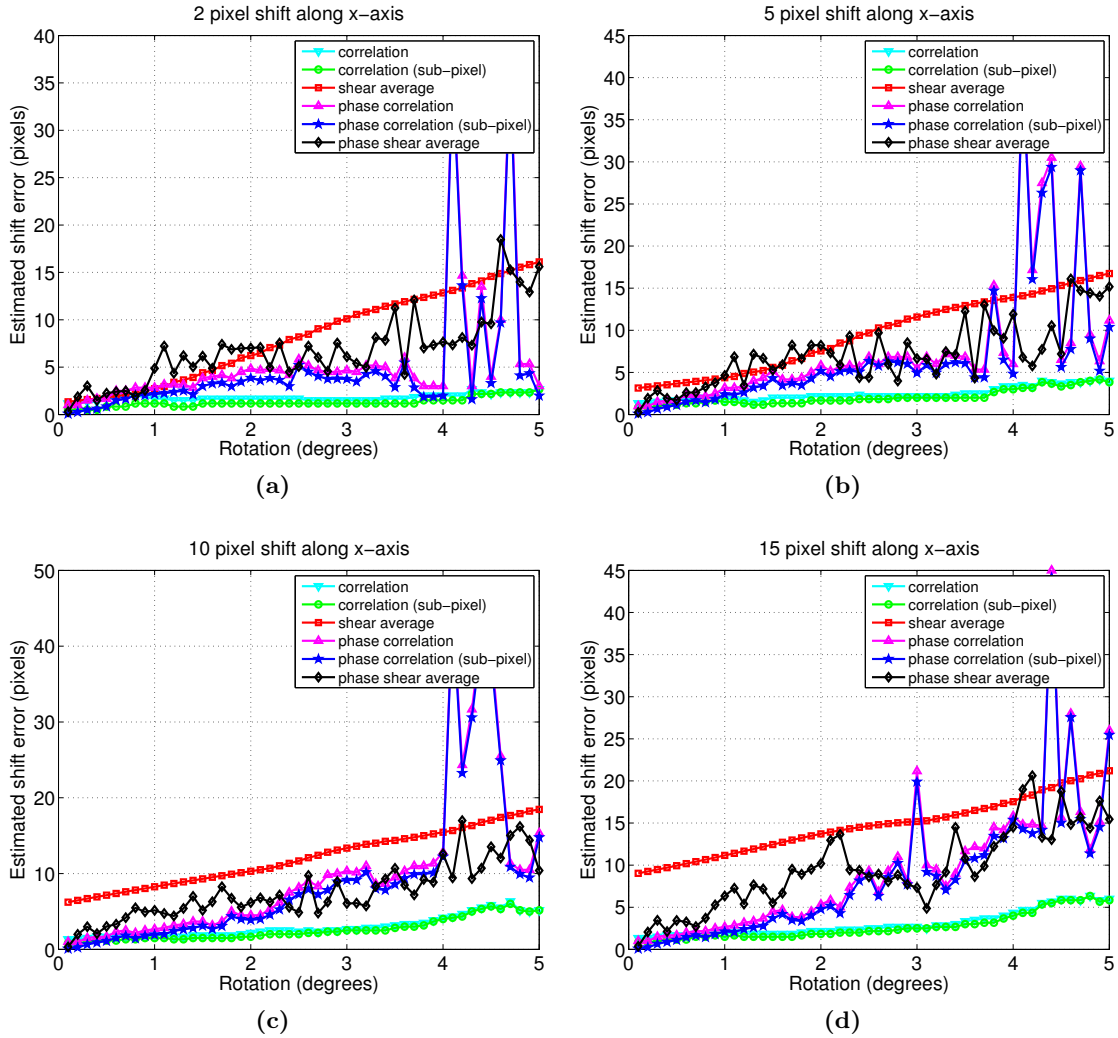


Figure 3.7: Error in estimated image shift for increasing amounts of rotation, for shifts of 2,5,10, and 15 pixels (average of the 6 images shown in Figure 3.5a). The characteristic shape of the offset error for small rotations, irrespective of the translation, was also present when the algorithm was run against other image sets.

Figure 3.7 demonstrates the shear-average algorithm is able to estimate the translation of the images better than the alternatives tested in the presence of small rotations (2–5 pixels).

Evaluating the runtime performance of the shear-average technique against others is not straightforward. The algorithm was faster than *Horn and Schunck*, and *Lucas and Kanade* for similar data, although those two algorithms are not considered state of the art anymore[6]. Furthermore, comparison of the exact numbers of arithmetic or floating point operations is almost impossible here, especially because almost all computation time in the shear-average is consumed by the FFT implementation, which is provided by the `fftw3` library. This library has a myriad of automatic enhancements chosen at runtime, which provide excellent performance [Frigo and Johnson, 2005], but make it difficult to micro-benchmark.

In conclusion, the shear-average algorithm has been compared against two other phase-correlation based implementations for image registration. In these tests, it was found that the algorithm performed with similar accuracy for image translation in the presence of noise and rotation, while having good runtime characteristics.

3.4 SUMMARY

The term optical flow comes from the optical flow constraint equation, which places a local constraint on image motion. Image velocities are the partial derivatives of image motion. The degree to which the constraint equation is satisfied and the approximations above are correct determine the degree to which optical flow approximates image motion. This thesis uses, in part, optical flow for the control of flying robots. The term optical flow will be used almost exclusively, in essence I assume that it approximates image motion to a sufficient accuracy to make control systems using optical flow possible.

Many insect behavioural strategies use image motion, but when testing biologically inspired analogues, it is important to consider the characteristics of the optical flow algorithm chosen. For example, relating to the local local motion detection occurring in the *medulla*, phase-correlation is an appropriate choice for implementing biomimetic strategies because it can be applied computationally efficiently in local patches. Furthermore, the regular (rectangular, gridded) nature of the result obtained from pyramidal *Lucas and Kanade*) also makes it suitable for implementing biomimetic strategies which operate on the premise of wide-field spatial integration of local image motion.

[6] The computer vision field has been focused on improving the accuracy of optical flow techniques, to a large degree driven by the scoreboard on the Middlebury website, which places a high importance on accuracy, but does not record runtime quantitatively (they are reported, but not controlled in any way).

Furthermore, I developed a novel frequency domain optical flow algorithm; the shear-average (phase-gradient correlation) technique which is more computationally efficient and similarly performing to existing methods. This technique will be used to estimate optical flow in other control systems presented in the forthcoming chapters.

Chapter 4

CONTROL OF FLYING ROBOTS

According to flying principle and propulsion mode, one can classify aircraft into the categories shown in Figure 4.1. This classification is important, as different flight vehicles pose different challenges for visual control, just as evolutionary pressures shaped the development of insect flight and behaviours [Dudley, 2000, ch. 6–7]. For example, Bachrach et al. [2010] implemented exploratory navigation and mapping that was only possible on a hovering platform with freedom to move in all directions.

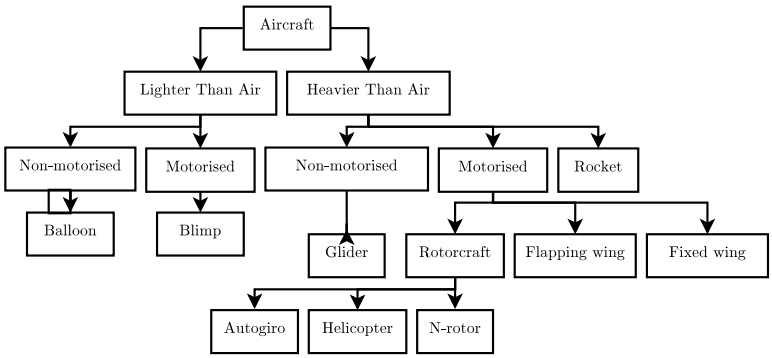
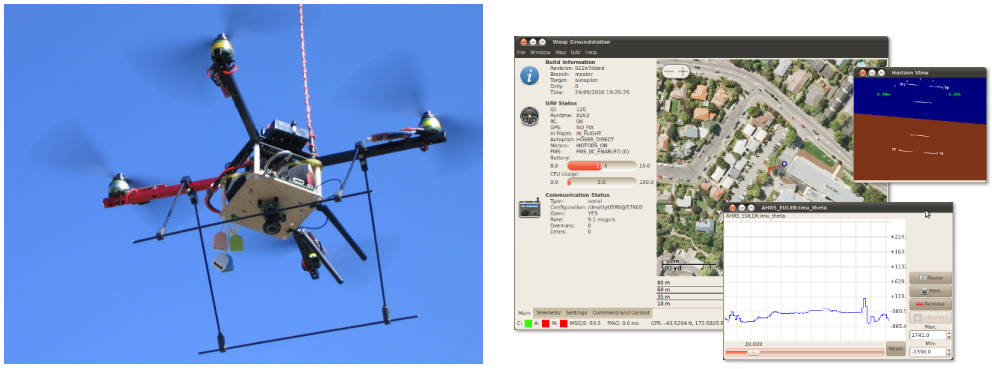


Figure 4.1: Aircraft classification by propulsion and flying principle. Quadrotor helicopters are 4-rotor rotorcraft.

Visual flight control test vehicles were historically expensive and complex platforms, often requiring large open spaces to operate. Fixed-wing aircraft also have constraints on manoeuvrability, whereas conventional helicopters are dynamically and structurally complex, expensive, and hard to control. In order to circumvent these issues I chose quadrotor helicopters to be used as the test vehicle and developed the 'wasp' system; a suite of software, hardware, and infrastructure for visual flight control research.

Figure 4.2 shows the 'wasp' system in use on a hovering quadrotor helicopter. In this scenario, the 'wasp' software autopilot provides the high-speed real-time *attitude* stabilisation of the craft, time correlated synchronised camera images, and inertial measurement unit (IMU) readings to an onboard single board computer (SBC). The SBC then executes the visual control algorithms, taking into consideration the vehicle state computed using the IMU, before sending control commands to the autopilot at



(a) (b)

Figure 4.2: (a) Quadrotor hardware and camera for onboard vision processing. (b) Groundstation software for monitoring real time controller performance.

a much slower rate. The decoupling of the two systems, both for reliability and for developmental efficiency, is facilitated by the 'wasp' system.

This chapter introduces the processes for experimenting using quadrotors. Section 4.1 describes the main idea of quadrotor dynamics, their flight properties (how they move and what movements are possible), the capabilities which make them appropriate for biomimetic visual flight control, and the considerations necessary for designing a robust system for research. Section 4.2 describes the attitude controller used to hover the craft and how higher level visual control systems were constructed which utilised it.

Appendix A includes a more detailed description of the quadrotor hardware. Appendix B describes quadrotor equations of motion, the simulation model and identification of parameters, and the control system implementation.

4.1 QUADROTOR HELICOPTERS

The first quadrotors for UAV research were developed by Pounds et al. [2002] and Bouabdallah et al. [2004a]. Today, quadrotor helicopters are a popular configuration for UAV research. The aircraft consist of four rotors in total; two pairs of counter-rotating, fixed-pitch blades located at the four corners of the vehicle and shown in Figure 4.2a.

Due to their specific capabilities, quadrotors provide a good basis for visual flight control research; research into the use of computer vision techniques for the control of the robotic system. First, quadrotors do not require complex mechanical control linkages for rotor actuation, relying instead on fixed pitch rotors and using variation in motor speed for vehicle control. This simplifies both the design and maintenance of the vehicle. Second, the use of four rotors ensures that individual rotors are smaller in diameter than the equivalent main rotor on a helicopter, relative to the airframe size.

The individual rotors therefore, store less kinetic energy during flight, mitigating the risk posed by the rotors should they collide with people or objects. Furthermore, by enclosing the rotors within a frame, the rotors can be protected from breaking during collisions; permitting flights indoors and in obstacle-dense environments with low risk of damaging the vehicle, its operators, or its surroundings.

These capabilities greatly accelerate the design and test flight process by allowing testing to take place indoors, by inexperienced pilots, with a short turnaround time for recovery from incidents. Finally, the improvement of Lithium-polymer battery technology has enabled longer flight times with heavier payloads, increasing the computational power that can be carried onboard, thus increasing the complexity of visual algorithms that can be experimented in realtime. Therefore, quadrotor helicopters fill an important role in the research arena, possessing the capability to lift large payloads while also having reduced kinetic energy, making them suitable for rapid indoor flight testing.

4.1.1 Computational Processing Considerations

Historically visual flight control of quadrotors has been performed off-board. As computing power increases and improved visual techniques are discovered, it is becoming more possible to move this computing onto the flying robot. Bachrach et al. [2010] represent the state of the art for offboard visual processing; the quadrotor is able to autonomously explore its environment, while mapping and searching along the way. The Pixhawk[1] project represents state of the art for onboard visual processing; it is able to autonomously hover and navigate a predefined map or an unknown environment.

While both onboard and offboard approaches seem distinct, they both share similar considerations:

Position control updates There is a lower bound on the acceptable rate of position control[2] if one wants to achieve stable hovering. In the author's experience[3], any vision based position controller should, assuming a reasonable degree of accuracy, provide *attitude* commands at $\geq 10\,\text{Hz}$.

State estimation A combined (visual and inertial) state estimator requires synchronised IMU and camera images for best performance.

Insensitivity to computation time The difference in computation times between simple visual processes (such as obstacle avoidance) and complex ones (such as SLAM on a large map) can differ by a factor of $10\times$ or more. For this reason, it is often a practical necessity to run these processes only as needed. Furthermore,

[1] See the project website, `https://pixhawk.ethz.ch`, for more information.

[2] Assuming that the position controller is implemented atop a PID *attitude* controller, as described in Section 4.1, low wind, and near hover conditions.

[3] Gathered while working on the Pixhawk project early January 2010, through his own experiments (unpublished) and through discussions with other researchers at micro air vehicle conferences.

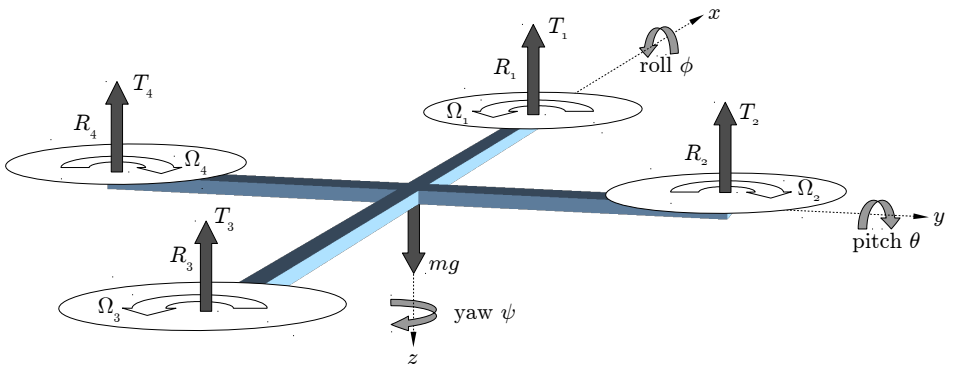


Figure 4.3: The coordinate system and free body diagram of a quadrotor helicopter.

a complex robotic system often has additional overhead in the form of processes for debugging, visualisation, and testing.

The 'wasp' system was designed to make these three considerations easy to implement.

4.1.2 Basic Concepts

Quadrotor helicopters can be well modelled [Bresciani, 2008, Bouabdallah et al., 2004a] as a structure with four rotors mounted in a cross configuration. The propellers are fixed and mounted in the same plane. The only thing that can vary is the propeller speed.

Consider the free body diagram of Figure 4.3 and let the quadrotor be at hover. All propellers rotate at the same speed, Ω_i [rad s^{-1}], to counter the force of gravity, mg.

The model has 6 degrees of freedom (DOF), but because it has only four actuators, the vehicle is still an under-actuated and dynamically unstable system [Bouabdallah et al., 2004a]. Due to the symmetric structure of the vehicle one can choose four optimal decoupled control variables as described below. These four control variables, U_i, are adjusted by the control system to allow the quadrotor to reach the desired *attitude* and altitude.

Assume the vehicle is level and at hover, with fixed propeller speed Ω_H. A positive change is represented by Δ_A, while a negative change is represented by Δ_B. The four variables are shown in Figure 4.4 and described below;

Throttle (U_1 [N])

Increasing (or decreasing) all propeller speeds by the same amount leads to a vertical force and acceleration, $\ddot{z}$, in the body frame — changing the vehicle altitude.

Roll (U_2 [N m])

Increasing (or decreasing) the left propeller speed and decreasing (or increasing)

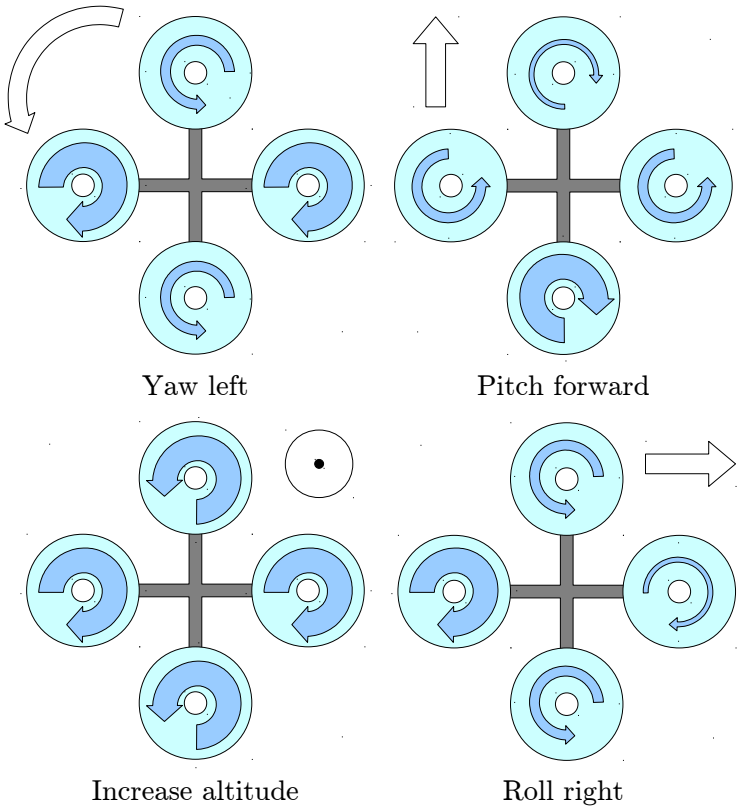


Figure 4.4: A description of quadrotor motion observed in response to different combinations of rotor speed at hover Ω_H. Arrow width is proportional to speed. Adapted from Bouabdallah et al. [2004a]

the right one leads to a torque with respect to the x-axis, causing the quadrotor to turn. If $\Delta_A \approx \Delta_B$ then the total vertical thrust is the same as when hovering; hence (to a first approximation) leading only to an acceleration in roll angle, $\ddot{\phi}$.

Pitch (U_3 [N m])

Similar to the roll command but applies a torque with respect to the y-axis. This leads to an acceleration in pitch angle, $\ddot{\theta}$.

Yaw (U_4 [N m])

Increasing (or decreasing) the front-rear propellers' speed and simultaneously decreasing (or increasing) the left-right pair. This leads to a torque with respect to the z-axis which makes the quadrotor turn. The yaw movement is generated because one opposing pair rotates clockwise and the other pair rotates counter-clockwise; hence the total torque is unbalanced. The total thrust is the same as hovering so the command leads to only an acceleration in yaw angle, $\ddot{\psi}$.

The dynamics of a generic 6 degree of freedom (DOF) rigid-body under external forces applied to the centre of mass and expressed in the body fixed frame are [Etkin

and Reid, 1995],

$$\begin{bmatrix} m\mathrm{I} & 0 \\ 0 & \mathrm{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{V}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega} \times m\mathbf{V} \\ \boldsymbol{\omega} \times \mathrm{I}\boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \boldsymbol{\tau} \end{bmatrix}, \tag{4.1}$$

where $\mathrm{I} \in \mathbb{R}^{3\times3}$ the inertia matrix, $\mathbf{V}$ is the body linear speed vector and $\boldsymbol{\omega}$ the body angular speed. The craft has mass m and is actuated by forces, $\mathbf{F}$, and torques, $\boldsymbol{\tau}$. For details of how $\mathbf{V}$ and $\boldsymbol{\omega}$ are generated from U_1 see Appendix B. For details of how I and other model constants were estimated empirically see Appendix B.1).

4.2 ATTITUDE CONTROL

When considering *attitude* control of quadrotors secondary aerodynamic effects such as blade flapping[4], thrust dependence on free stream velocity, and interference caused by the vehicle body in the slip-stream of the rotor can be disregarded for hover and slow translational flight [Pounds et al., 2006, Hoffmann et al., 2004, 2007]. Pounds et al. [2010] found that the regulation performance of simple PID control schemes was within $\pm2^\circ$ of level tracking, furthermore, that PID control offered an advantage over model based controllers due to the scheme's simplicity and potential robustness to parameter variation [Bouabdallah et al., 2004b]. Such parameter variation occurs frequently in this thesis; such as when changing the craft weight, the centre of mass, or when experimenting with different vision and computer configurations (such as described in Appendix A.4).

Navigation or obstacle avoidance can be considered a position control or velocity control problem. In the positional flight control case, Hoffmann et al. [2007] showed positional control implemented using a PID controller with the vehicle's pitch and roll as control inputs. Tilting the vehicle in any direction causes a component of the thrust vector to point in that direction, so commanding pitch and roll is directly analogous to commanding accelerations in the x-y plane. However, a key weakness of this and similar position controllers was the assumption that the velocity of the free stream and *attitude* control are decoupled. This is only true for small velocities.

Insofar as this concerns the visual flight control strategies implemented in this thesis; I conclude that positional control and navigation can be implemented on top of, at minimum, a PID *attitude* controller. By commanding the *attitude* controller with pitch and roll deviations from hover, I regulate position with good accuracy; providing the flight dynamics are slow (close to hover) and wind velocity is low (such as in indoor flight).

The details of the PID attitude controller are explained in Appendix B.4.

[4] The flexion of rotor blades due to difference in relative air velocity in translational flight.

4.3 SUMMARY

Quadrotor helicopters are a hovering aircraft controlled by symmetrically positioned counter-rotating rotors. The aircraft moves by adjusting the velocity of these rotors, which induces a change in *attitude* of the craft and subsequently a change in position.

I created the 'wasp' quadrotor system for visual flight control research. It was one of the first of its kind and allows real-time visual flight control research on a flying robot. For this thesis I used the 'wasp' system to test biomimetic flight control strategies[5].

When testing these strategies I consider the UAV at hover, thus I implemented a position controller without consideration of the detailed control system dynamics of the *attitude* controller. The PID controller was insensitive to the plant variations commonly encountered in my research; the significant changes in payload and configuration shown in Appendix A.4.

[5] With the exception of using the Pixhawk system for some experiments in Section 6.1

Chapter 5

DIRECT CONTROL

Sections 2.1.3 and 2.1.4 showed that one of the strongest visual responses in insects is the optomotor response and in conjunction, the wide-field optical flow balance strategy.

Compared with wheeled robotics, one of the difficult challenges of flight control is the need to control the vehicle *attitude*. This is especially true in quadrotor helicopters, because only through changing *attitude* can one change position.

I propose the log-polar domain as a suitable way to decouple the translational and rotational flow fields; to facilitate extraction of robot *attitude* and altitude.

The log-polar transform has two salient properties (explained in the forthcoming sections) which make it suitable for visual flight control [Tistarelli and Sandini, 1993, Manzotti et al., 2001, Traver and Bernardino, 2010];

An elegant trade-off between the mutually opposed criteria of wide field-of-view, high resolution, and minimizing the amount of data to process. Specifically, log-polar sampling reduces the size of computation up to 30 times while retaining a higher resolution in the foveal region.

Invariance to rotation and scaling when done so about the optical centre. In this scenario, patterns in the log-polar image undergo translation, preserving their shape. This is particularly for scale and rotation-invariant image alignment, pattern recognition and optical-flow.

These properties are well suited to image processing when moving in a complex environment. In this situation the optical flow fields will contain rotational and translational components that, without additional information or assumptions, are difficult to separate. The biological equivalent of this is the estimation of self motion in insects. Insects decouple the mixed rotational and translational components of image motion through wide-field integration of the output of directionally sensitive EMDs (see Section 2.1.3.3).

This chapter describes three strategies I developed for *attitude* estimation and control of the 'wasp' quadrotor. All strategies use optical flow and spatial integration of optical flow vectors in the log-polar domain.

This chapter proceeds as follows. Section 5.1 introduces the log-polar transform and explains why it is well suited for flight control. Section 5.3 describes a method using integration of optical flow vectors in the log-polar domain to estimate robot *attitude* and altitude. Section 5.4 introduces an artificial implementation of the expansion avoidance strategy — called time-to-contact (TTC), again utilising the log-polar domain. This expansion avoidance strategy is further tested in the behavioural experiments in Chapters 6 and 7.

5.1 THE LOG-POLAR TRANSFORM

The log-polar mapping is a geometrical transformation which attempts to emulate the topological reorganization of visual information from the retina to the visual cortex of primates [Tistarelli and Sandini, 1993, Traver and Bernardino, 2010].

The log-polar operation transforms an image, $\mathbf{I}[u, v]$, indexed in the Cartesian coordinate system (u, v), onto a cylinder $\mathbf{I}^*[\rho, \varphi]$, indexed in the polar coordinate system (ρ, φ). Here r denotes radial distance from optical centre (o_x, o_y), and φ denotes angle. Any (u, v) point can be presented in these polar coordinates;

$$\begin{aligned} u &= r\cos\varphi, \\ v &= r\sin\varphi, \\ \rho &= \ln r, \end{aligned} \tag{5.1}$$

where $\ln = \log_e$. Rearranging (5.1) gives the transform;

$$\begin{aligned} \varphi &= \arctan\frac{v}{u}, \\ \rho &= \ln\sqrt{u^2 + v^2}. \end{aligned} \tag{5.2}$$

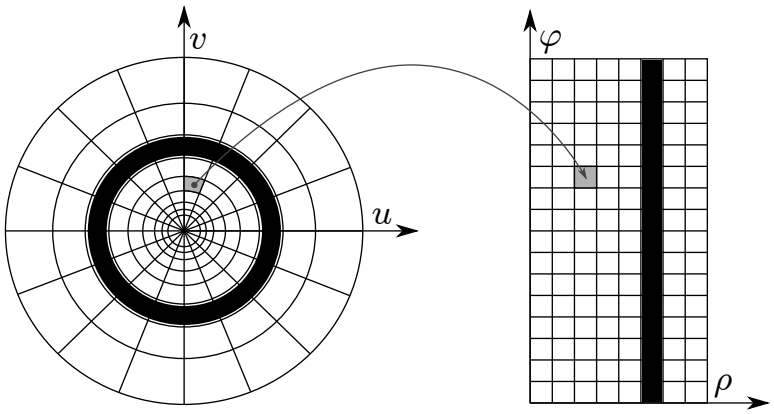


Figure 5.1: A schematic of the log-polar mapping demonstrating that the transform is multi-resolution (see grey area). Regions of different area are mapped into a uniform grid in the cylindrical image.

Consider the grey highlighted region in Figure 5.1. The area of the regions in (u, v) increase with the distance from the origin — in $\mathbf{I}$, when far from the origin, the mapping of pixels from $\mathbf{I}$ to $\mathbf{I}^*$ is many-to-one. Therefore, $\mathbf{I}^*$ is multi-resolution; its resolution decreases with the axial distance ρ [Wolberg and Zokai, 2000, Peters II et al., 1996].

Typically, due to this multi-resolution property, a small window that surrounds the transform origin in $\mathbf{I}$ is encoded at full resolution in $\mathbf{I}^*$. The region is known as the *fovea*, analogous to the human retina. In the periphery (the pixels not in the *fovea*) $\mathbf{I}$ is increasingly down-sampled since many pixels in $\mathbf{I}$ now map to a single pixel in $\mathbf{I}^*$.

This data reduction can be significant and can speed image processing [Peters II, 2000] (a variation of this concept is used in Section 6.1). Figure 5.2 shows exaggerated examples of the log-polar transform multi-resolution and invariance properties. However, the primary use of the log-polar transform in this thesis is not for its data reduction property, but for its invariance to scale and rotation; linear scaling and rotations along (x, y) are transformed into linear shifts along (ρ, φ) [Wolberg and Zokai, 2000, Peters II et al., 1996, Peters II, 2000].

The magnitude of data-reduction is controlled by the position and size of the *fovea*. Therefore, (5.2) is usually [Peters II et al., 1996] expressed as

$$\rho = M \log (r + \kappa) \tag{5.3}$$

$$r = \sqrt{(u - o_x)^2 + (v - o_y)^2} \tag{5.4}$$

$$\varphi = \text{atan2} \left(\frac{v - o_y}{u - o_x} \right), \tag{5.5}$$

where (u, v) are the image plane coordinates, (o_x, o_y) is the principal point (the notation of Section 2.2.2), M and κ are real positive parameters that code the foveal region of I. For specific details on choosing and calculating M and κ see Appendix D.3.

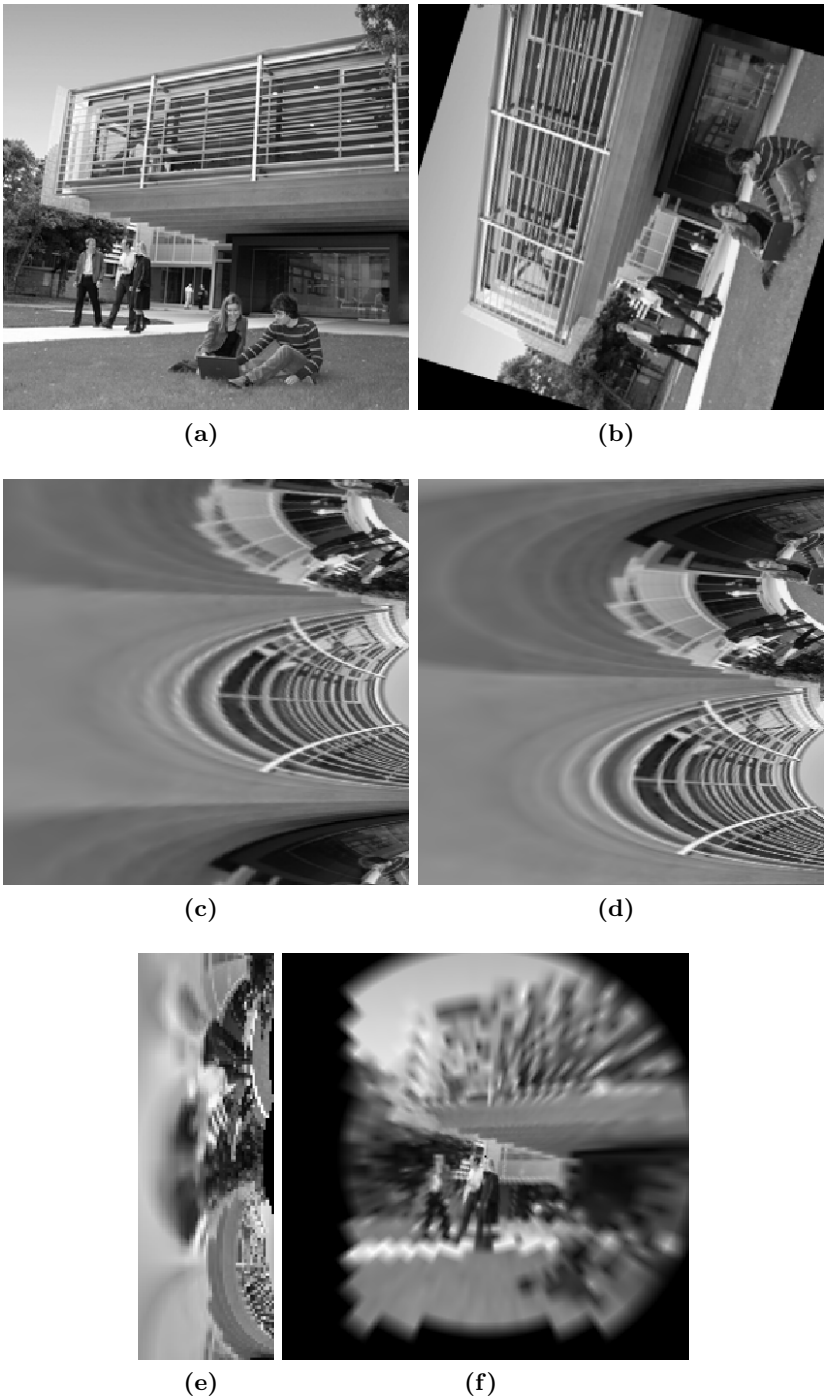


Figure 5.2: Images demonstrating the rotation-invariance and multi-resolution properties of the log-polar transform. As in Figure 5.1, the log-polar images (c),(d), and (e) vertical axis is φ and horizontal axis is ρ. (a) The original image. (b) The image rotated by 73°. (c) The original image in the log-polar domain, using a *fovea* such that spatial extrema in I map to radial extrema in I^* (see Appendix D.3). (d) The rotated image in the log-polar domain; the rotation has become a vertical translation in the φ axis. The *fovea* was set as in (c). (e) The log-polar transform of (a) using a smaller resolution in the log-polar domain of 80×160 pixels. (f) The inverse log-polar transformation of (e). Due to the multi-resolution property of the transform, only the area around the *fovea* has been recovered with high fidelity.

5.2 VARIANCE OF IMAGE REGIONS

To mimic the local receptive fields of the *ommatidia*, for most experiments, I compute optical flow over small image regions. In this case, it is more likely an image patch may have insufficient intensity structure (Section 3.1). This would yield an inaccurate optical flow estimate for the region and potentially an incorrect control outcome. Variance is one of the measures of a distribution, describing how far numbers lie from

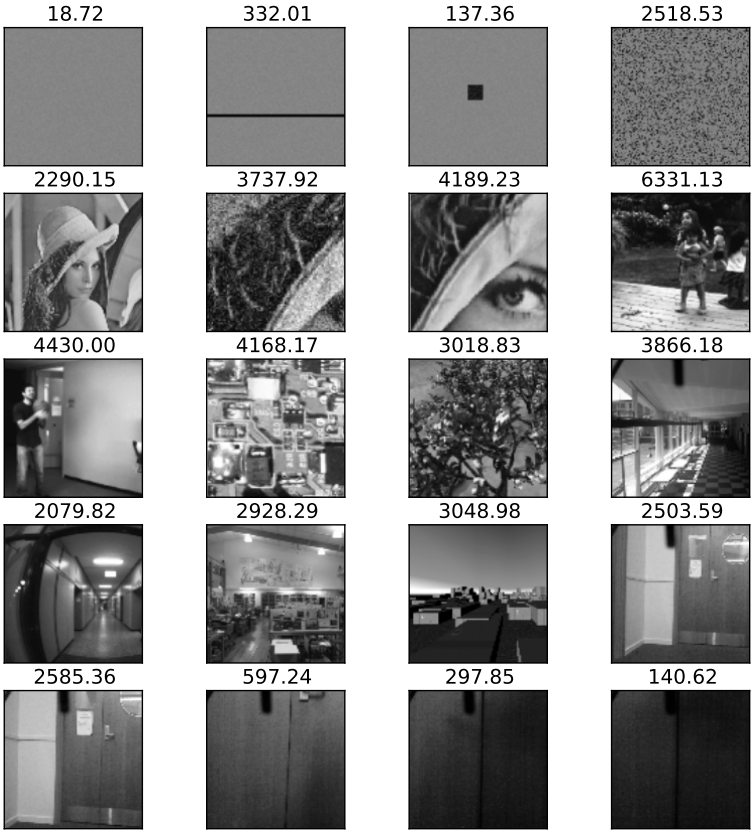


Figure 5.3: Examples of images and their variance (printed above each respectively). The first row of images are randomly generated. The last 9 images are extracted from a flight and time-to-collision experiments explained in later sections.

their mean. When applied to an image, it coarsely indicates how much structure is present. To reduce such inaccurate optical flow estimates from such regions, I do not compute optical flow if the variance in that region is too low.

I arrived at this heuristic by considering representative images from different datasets and the environments where I performed experiment, shown in Figure 5.3.

In general this value was 200 for `uint8` grey images, which I used exclusively.

5.3 ATTITUDE ESTIMATION AND CONTROL USING OPTICAL FLOW

In this section I present a biologically inspired computer vision technique to estimate the *attitude* of the 'wasp' quadrotor.

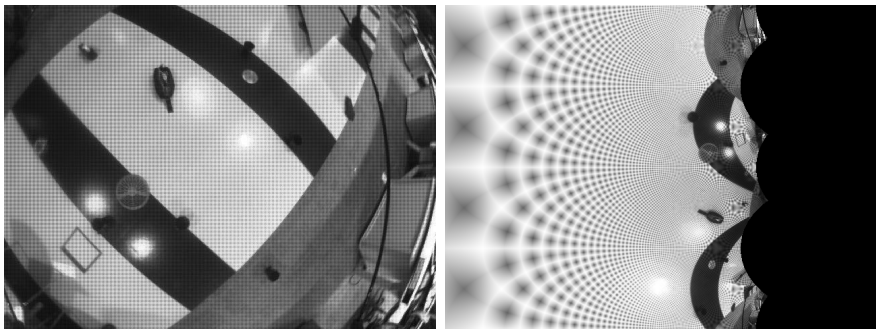
Images captured from the onboard camera record image motion due to rotation about the optical axis and translation in the direction of the optical axis. For a downward facing camera, this corresponds to estimating the heading and altitude components of the quadrotor. This is also equivalent to measuring lateral translation and pitch in an insect[1].

UAVs are usually controlled using an *attitude* estimate obtained from an IMU consisting of gyroscopes, magnetometers, accelerometers, and a barometer. However, IMUs are not without problems such as accumulating drift. There are also limitations of the global positioning system (GPS) network including potential loss of signal for prolonged periods of time. If it can be shown that vision systems are capable of similar performance then integrating them into the flight control system seems sensible.

In order to compare with inertial sensors, I measure the yaw rate (change in heading), r, in the aircraft body-frame.

5.3.1 Optical Flow and Phase-Correlation in the Log-Polar Domain

The use of phase-correlation and cross-correlation for registering image shift was introduced in Section 3.3 and the log-polar transform was introduced in the previous section.



(a) **(b)**

Figure 5.4: Downward facing images captured while the quadrotor was in hover, showing the effect of log-polar transformation. (a) Original image. (b) After log-polar transformation.

[1] Geometrically speaking, as the hemispherical compound eyes of the organism point to the side, a change in pitch induces a rotational flow field, whereas lateral translation produces an expansionary flow field.

Application of the cross-correlation function to the log-polar images $\mathbf{I}^*_k$ and $\mathbf{I}^*_{k+1}$ yields a peak in the response. For rotation only, the peak lies at an offset in the φ-axis of the cross-correlation space. Similarly a constant image scaling will yield an impulse function offset in the ρ-axis. In the case of a downward-facing camera mounted on the 'wasp' quadrotor, constant image scaling (Figure 5.4) is due to motion in the optical axis, such as that due to a change in altitude. Image rotation corresponds to a change in heading[2] of the quadrotor.

The scaling is inversely proportional to Z^2 and consequently the dynamic range over which this motion can be effectively measured can be small. Taking logarithms can improve this situation,

$$\log \Delta\rho \propto \log \Delta Z - 2 \log Z. \tag{5.6}$$

Hence the cross-correlation of the log-polar image space, $\mathbf{I}^*$, is used in practice to determine a change in altitude ($\log \Delta\rho$) and heading ($\Delta\alpha$). Here, for a down-facing camera, α and Z are the heading and altitude in the aerodynamic body-axis described in Section 4.1.

Using this result, the following section describes a novel altitude estimation and control system I developed.

5.3.2 Implementation

The quadrotor system used in these tests was the 'wasp' quadrotor with Gumstix single board computer, described in Appendix A and Appendix A.3.3. To the single board computer I attached the Firefly MV image sensor and 'fisheye lens', described in Appendix A.3.1.

Images are captured at 640×480 pixels, at 15 times per second. Each image is taken with a 15 ms shutter time. The camera is fitted to look directly at the ground.

Captured images from the camera are passed into `OpenCV` using the `libdc1394` library. The image is transformed into log-polar coordinate space, a Gaussian blur is applied, and the image is broken into several regions. The phase correlation in each region is computed via the result of a discrete Fourier transform. The discrete Fourier transform is again performed using the open source `fftw3` library.

Simultaneously, the inertial estimate is received over universal serial bus (USB) from the flight control system. Both the inertial and computed visual yaw results are recorded on a non-volatile SD card.

[2] The terms yaw and heading are often used interchangeably. Yaw is a rotation about an object's z-axis and will be used when describing the output of a specific sensor. Heading refers to the direction the aircraft is facing or tracking. Yaw and heading are coincidental and aligned, subject to calibration, in this aircraft.

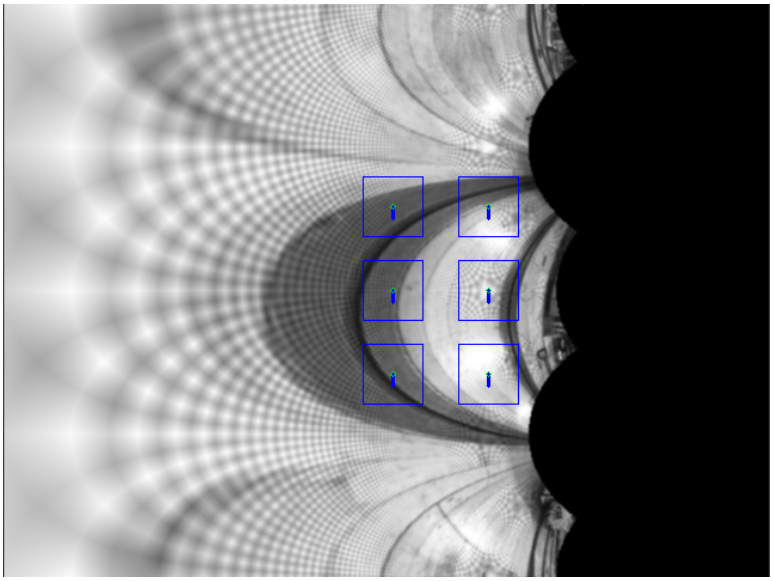


Figure 5.5: Log-polar image with regions of calculation for phase correlation. The vertical lines, enclosed in boxes indicating the calculated region, represent the magnitude vector of the image motion from the previous frame over that region. The vectors shown in this image have been scaled for readability.

5.3.3 Computation of Heading and Altitude

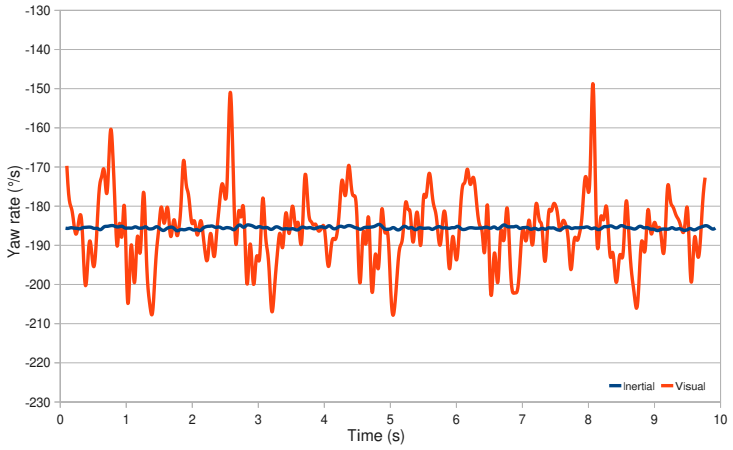
The log-polar images are processed on a small number of sub-images or regions for computational efficiency reasons. The image region locations shown in Figure 5.5 were selected based on the likelihood of significant change of the image data in order for the phase-correlation to work effectively. The omnidirectional lens magnified rotational change sufficiently to make this happen.

The total heading and altitude estimate is obtained by averaging the result — the impulse location, from each regions' estimate.

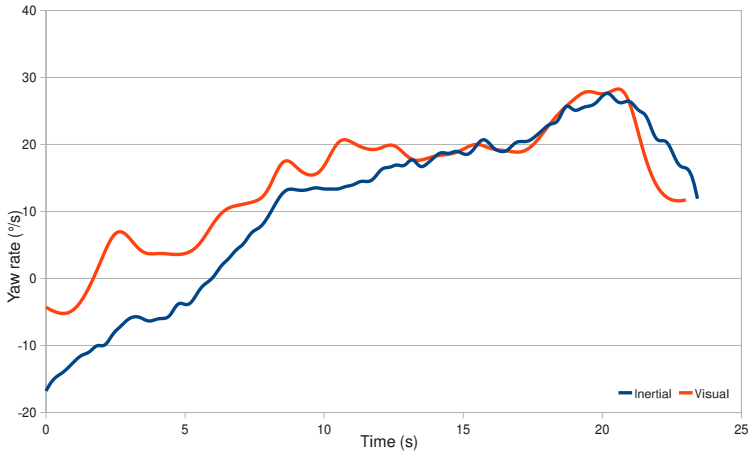
All experiments were performed with the 'wasp' quadrotor placed in *attitude* control mode. The craft is commanded through deviations from zero *attitude*. This means the craft is stable in hover without pilot input. For more information on the 'wasp' control system, see Appendix B.4.

5.3.4 Test Results

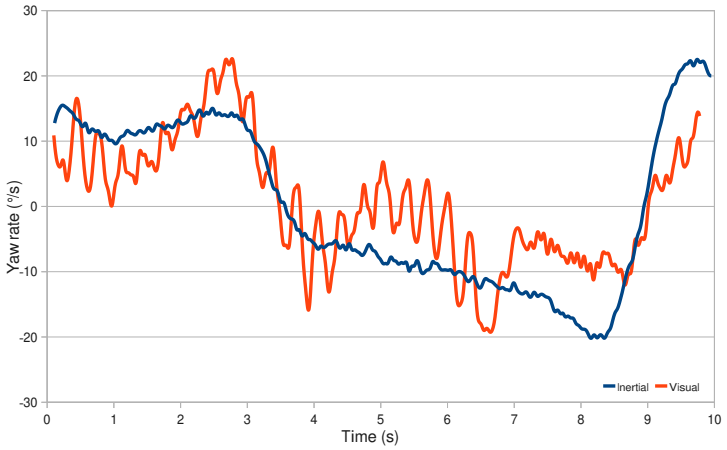
To calibrate the system the quadrotor was first mounted on a mechanical platform, whose rotational velocity was known and constant, shown in Figure 5.6a. The optical flow estimate was calculated realtime by the SBC at 10 Hz, limited by the computational power of the OMAP3503 processor. The inertial estimate was calculated at 100 Hz by the central processing unit (CPU) and sent to the SBC, where it was recorded for comparison with the optical estimate. Comparison of the two techniques was performed offline.



(a)



(b)



(c)

Figure 5.6: The estimation of quadrotor yaw rate from inertial measurement unit and image processing pipeline. (a) Quadrotor mounted on rotating platform. (b) Quadrotor flown in stable flight. (c) Quadrotor flown in aggressive flight in a busy environment

Figure 5.6 shows the estimated yaw rate of the quadrotor over the flight. The inertial estimate is measured using the rotational velocity obtained from the yaw-axis gyroscope. The visual estimate is obtained from the total φ-displacement of phase-correlated samples in the log-polar image domain.

The visual estimate is related to the inertial baseline through a constant, whose value was determined by fitting the visual estimate to the inertial measurements taken when the quadrotor was mounted on the rotating platform, Figure 5.6a. This constant relates the perceived change in image brightness through eqs. (5.5) and (5.6) yielding the yaw rate *egomotion* of the quadrotor.

Following the initial calibration, the quadrotor was then flown manually in a stable hover. Changes in altitude and translation were kept at a minimum while yawing the quadrotor fore and back, Figure 5.6b and Figure 5.6c.

Results obtained in real time, using phase-correlation on log-polar transformed images can provide yaw rate estimates for a quadrotor helicopter. These real-time estimates could be fed back to the quadrotor control system as an aid, or possibly a replacement, for measurements traditionally obtained from an inertial measurement unit. The same scaling constant was applied to all results and shows that the visual estimate of the yaw rate correlates with the actual yaw rate as reported by the inertial measurement system. It is plausible that an evolved neurological relationship could embody this correlation; giving the possibility that organisms could use wide-field integration of optical in the manner demonstrated here.

5.4 TIME-TO-CONTACT EXPANSION AVOIDANCE

Time-to-contact (TTC) is a technique for estimating the time when collision with a visible object will occur using only visual information. It is used in robotic vision and visual control because knowledge of the robot velocity or its initial distance from the object is not required. Based on the equations for the expansion of the optical flow field it is possible to calculate the number of frames (not time per se) remaining before contact with an object [Lee, 1976] and [Trucco and Verri, 1998, sec. 8.1]. Time-to-contact is a formalisation of the expansion/collision avoidance reflex (or the visual portion only, not the reflex) found in insects and explained in Section 2.1.4.5. It is an efficient response for the organisms as it does not require solving the general case of *egomotion* in order to exhibit useful behaviour [Camus, 1995].

This section describes the related concepts of time-to-contact and focus of expansion as they used frequently through the rest of my work.

Figure 5.7 describes the optical geometry. Let the camera face the same direction as the direction of motion. As it moves it does so towards an imaginary point known as the focus-of-expansion (FOE), since this is the point from which the optical flow

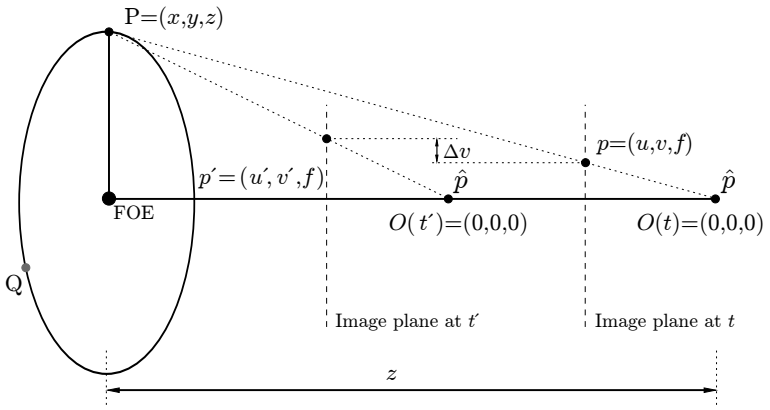


Figure 5.7: The optical geometry of focus-of-expansion (FOE) and time-to-contact (TTC). The camera (with focal length f) at origin, O, moves along a straight path to O' at t'.

diverges.

One can calculate the time-to-contact by tracking a point in the image frame between frames. A point of interest $\mathbf{P}$ at location (x, y, z) is projected through the focus of projection, $\hat{p}$ centred at the origin of the coordinate system $(0, 0, 0)$. $\mathbf{P}$ is fixed in space and does not move. The camera mounted on the robot moves forward with a velocity $\frac{\mathrm{d}x}{\mathrm{d}t}$. Point $\mathbf{P}$ corresponds to $\mathbf{p} = [u, v, f]^\mathsf{T}$, a point on the image plane with (u, v) pixel coordinates and f the focal length of the lens. As the robot and hence the camera origin move closer to $\mathbf{P}$, the position of $\mathbf{p}$ in the image plane changes.

The TTC can be determined by considering the equilateral triangles;

$$\frac{v}{f} = \frac{y}{z}, \tag{5.7}$$

with f a constant value, differentiating with respect to time

$$\dot{v} = \frac{\dot{y}}{z} - y\left(\frac{\dot{z}}{z^2}\right). \tag{5.8}$$

Since $\mathbf{P}$ is fixed and static in the environment, set $\dot{y} = 0$. Substitute $y = vz$;

$$\dot{v} = -v\left(\frac{\dot{z}}{z}\right), \tag{5.9}$$

and divide by v before taking the reciprocal

$$\frac{v}{\dot{v}} = -\frac{z}{\dot{z}} = \tau. \tag{5.10}$$

The unit-less quantity τ is known as time-to-contact.

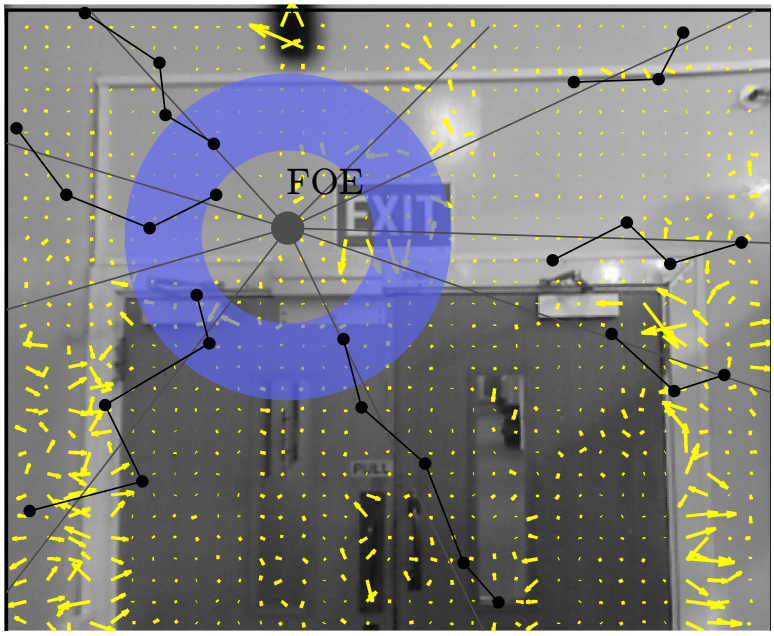


Figure 5.8: A schematic of the principles of focus-of-expansion and time-to-contact. When making a direct approach to a flat surface, all optical flow vectors emanate from a single point; the focus-of expansion.

5.4.1 Computing the Focus of Expansion

The FOE is demonstrated in Figure 5.8. It is the point in an image from which all optical flow vectors appear to diverge [Trucco and Verri, 1998, ch. 8]. The FOE is a quantity which is used throughout this thesis; in this section I explain how it is computed from measurements of optical flow [Guissin and Ullman, 1991].

Ideally, one could simply take the intersection of any two optical flow vectors, but due to measurement errors this would be inaccurate [Camus, 1995], so a large number of flow vectors are considered. This takes advantage of the (x, y) component representation of the optical flow vectors. In the case of forward translational motion and a single object filling the field of view, the FOE may be calculated by averaging the x and y components of all the optical flow vectors, with each component being assigned a unit pixel value regardless of its actual magnitude, and treating the average as an offset from the line of sight, which is assumed to be straight-forward.

Calculate the magnitude, M of the optical flow V in each (x, y) direction

$$\begin{aligned} \mathrm{M}_u &= \mathrm{V}_u^2 \\ \mathrm{M}_v &= \mathrm{V}_v^2. \end{aligned} \tag{5.11}$$

The index of the minimum sum is the point from which image motion appears to

diverge. This is calculated for each direction,

$$\begin{aligned}\mathbf{FOE}_u &= \arg\min_i \sum_{i=0}^{N} \mathrm{M}_u \\ \mathbf{FOE}_v &= \arg\min_i \sum_{i=0}^{N} \mathrm{M}_v. \end{aligned} \tag{5.12}$$

Here, $\mathbf{FOE} = [\mathbf{FOE}_u, \mathbf{FOE}_v]^\mathsf{T}$, a vector specifying the focus-of-expansion (FOE) in the image (u, v) dimensions.

5.4.2 Efficient Calculation of Time-to-Contact

Figure 5.7 and (5.10) show that the expected divergence of any point along a circle of a given radius should be equal; e.g., given the assumption of a flat surface and a direct approach the divergence of any point q should be equal to that of point p.

From this observation, Camus [1995] suggested averaging the optical flow measurements along the circumference of any circle of a given radius, centred at the FOE, to get a single real-valued measurement for $\dot{v}$ of (5.10).

The quantity v in (5.10) is then simply the radius of that particular circle. Finally, a single measurement of TTC is calculated as $\tau = v/\dot{v}$. The number of independent TTC measurements available is only limited by the image size and the position of the FOE.

Tistarelli and Sandini [1993] had a similar observation; the time-to-contact of a point on the retinal plane only affects the radial component of the optical flow. From this it is sufficient instead of summing around the radius of a particular circle, to sum values along the φ axis in the log-polar transformed image (see Figure 5.1).

5.4.3 Integration and Flight Test Results

To validate the time-to-contact algorithm, I performed several experiments with the 'wasp' quadrotor approaching a corridor wall, and computed the time-to-contact from these real image sequences.

Figure 5.9 shows the performance of the algorithm. The different trials have been shifted so impact corresponded to frame zero. The log-polar based algorithm correctly recovered an estimate for time-to-contact, τ, that corresponded to the true frame number at which impact occurred.

Given an efficient method to calculate time-to-contact values, one must transform these into a behavioural command to be executed by the control system. In 'wasp', this command was an 'avoid' signal, which set the forward velocity of the craft to zero.

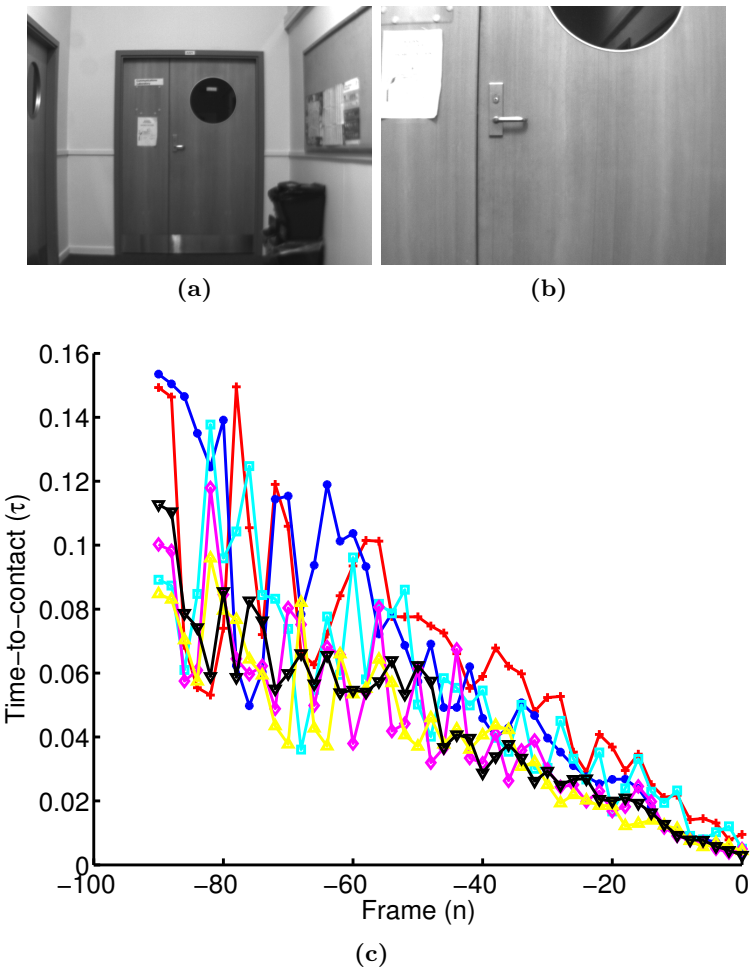


Figure 5.9: The performance of the time-to-contact algorithm in a number of flights. Each curve represents an individual trial where the 'wasp' quadrotor was flown towards the wall. (a,b) Two subsequent images of the wall towards which the quadrotor was flown. (c) The predicted time-to-contact value (τ).

This process is demonstrated graphically in Figure 5.10, and comprises the following steps;

1. for each new frame calculate the time-to-contact, τ
2. maintain a running history of 5 previous frame numbers, n, and time-to-contact estimates, τ
3. using these 5 pairs of values, perform a linear regression to calculate the slope, m, intercept, b, and p-value
4. using the slope of the line, calculate the x-intercept relative to the current frame number
5. if the x-intercept is $< F_{\max}$ frames in the future and the line estimate is robust ($p < 0.05$), generate an 'avoid' command.

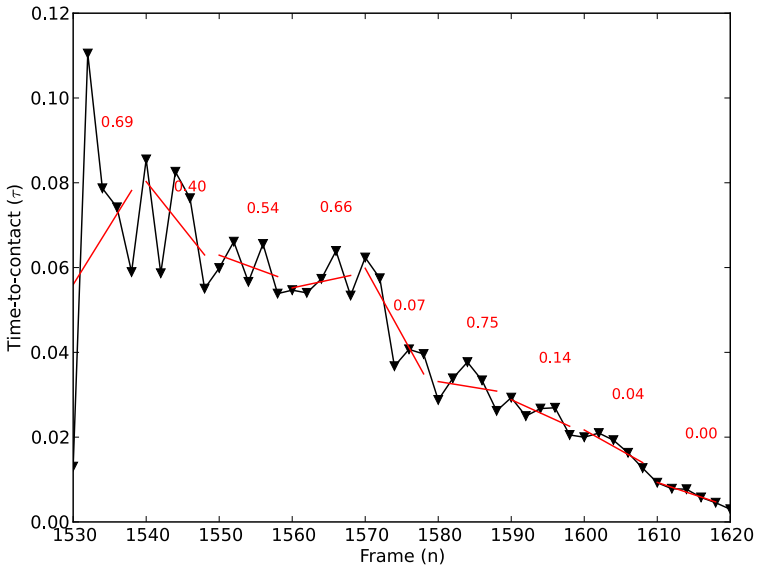


Figure 5.10: An example of the on-line process for evaluating time-to-contact estimates in order to generate an 'avoid' command. The red-lines indicate a running (history of last 5 frames) linear regression and associated p-values. If $p < 0.05$ and the calculated x-intercept is imminent then an 'avoid' command is generated.

The value $F_{\max}$ was tuned iteratively, based on the UAV weight, motor power, and frame-rate, to be $F_{\max} = 30$.

The motivation for reducing the quantitative time-to-contact estimate down to a qualitative binary command ('avoid' or 'not avoid') was two-fold. Firstly, to be consistent with nature, where the avoidance response is thought to be winner-take-all in insects. Secondly, to reduce false positive 'avoid' commands due to incorrect time-to-contact estimates.

Like the optical flow algorithms on which it was built, experiments showed the time-to-contact implementation was susceptible to noise, of which a large part was due to poor SNR in the input image. To mitigate the effects of this, I do not estimate TTC if the image has insufficient variance, as described in Section 5.2.

The robustness of FOE calculations can be improved by assigning vector components values for robustness, i.e., vectors close to the FOE are not penalized because their actual magnitudes are small.

5.5 SUMMARY

Direct control; or generally, the use of optical flow without the use of a model to capture important points of the environment is the traditional example of biologically inspired visual control.

In this chapter I used the log-polar representation of image motion to estimate quadrotor *attitude* and to build a time-to-contact system. Just as a *retinotopic* ar-

rangement of *neurons* in the optical lobe (Section 2.1.3.3) is used for wide-field interpretation of image motion I find the log-polar representation efficient for interpreting motion fields. I construct an obstacle avoidance strategy based on detecting objects whose relative distance is different to that which surrounds them.

I presented an implementation of the time-to-contact algorithm; in the conventional Cartesian coordinate system, and in the log-polar coordinate system. The time-to-contact algorithm is the artificial equivalent to the insect collision/expansion avoidance response and I use it as a building block in creating the other control systems explained in the forthcoming chapters.

Chapter 6

CONTROL USING DEPTH

The previous chapter presented the use of optical flow for attitude estimation, altitude estimation, and flight control. This chapter extends that work to consider other interpretations of the optical flow field. These include; using inertial information to aid estimating depth from optical flow, and utilizing the depth reading from a Microsoft Kinect sensor. Both estimates are used to control quadrotor flight.

Consider the optical flow measured as a UAV flies through the world (Figure 6.1). The optical flow field can be interpreted as the *egomotion* of the observer, or in terms of the 3D environment. An expanding flow field with the focus of expansion in the front can signify an impending collision [Srinivasan, 2006, Barrows et al., 2002]. Similarly, a high velocity patch embedded in a low velocity surrounding may indicate a nearby object in front of a more distant background. In a static environment, knowing the motion of the observer, one can relate optical flow to depth allowing further interpretation. For example, large regions of similar optical flow (and depth) may describe geometric structures such as planes (like the ground plane).

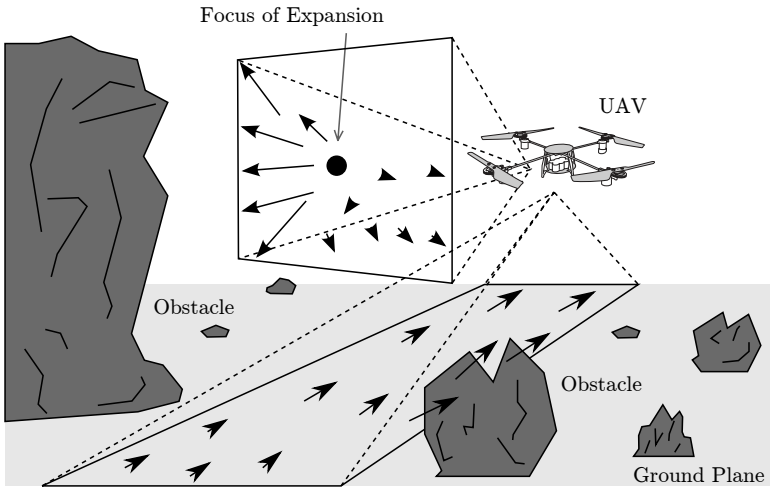


Figure 6.1: A schematic demonstrating patterns of optical flow observed as an observer moves through the environment. The structure and interpretation of these patterns is used to direct specific control strategies.

This chapter presents two different control strategies for the 'wasp' quadrotor that, through the imposition of constraints or the addition of more sensor data, allow greater meaning to be assembled from image data.

This chapter proceeds as follows. Section 6.1 presents a control strategy that uses two different interpretations of optical flow; the relative depth to obstacles, and the detection of regions of depth different to their surroundings. The combination of these processes are combined in a manner similar to the concurrent biological responses described in Chapter 1, to command the 'wasp' quadrotor. Section 6.2 utilises a Microsoft Kinect depth-camera (Section 2.2.3) to estimate the ground plane in the quadrotor's environment, and uses this estimate to control the robot altitude. The biological rationale for each control strategy is presented in the relevant section in turn.

6.1 DEPTH AND OPTICAL FLOW BALANCE STRATEGY

This section describes a biologically inspired obstacle avoidance and navigation strategy (that I call 'depth and optical flow balance strategy') for flying robots that uses two concurrent control processes whose sum determine quadrotor flight. The goal of this work was to demonstrate the application of concurrent biologically inspired processes in real-time, on the robot was not only possible, but the combination of the two processes led to improved behaviour over individual processes used alone.

The implementation was tested against open-loop datasets and real indoor flight. In both situations it was able to control the craft heading and pitch during flight, avoiding both corridor walls and oncoming obstacles.

6.1.1 Biological Basis

As described in Section 1.2.1, experiments have shown that insect exploratory or goal orientated *gross* flight behaviour is interspersed with instantaneous control responses such as object avoidance. Furthermore, considering *interneurons* outside of the visual system, Frye and Dickinson [2004] showed that motor output reflects the linear superposition of visual and olfactory inputs in Drosophila. I propose linearly superposing an obstacle avoidance and a navigation controller.

Specifically relevant in this section is the work of Tammero and Dickinson and Srinivasan et al.. Tammero and Dickinson [2002a] demonstrated that fruit flies avoid obstacles by turning away from regions with high levels of optical flow (OF). Srinivasan et al. [1996] showed that honeybees balanced the level of lateral OF in order to stay equidistant from the flanking walls.

Figure 6.2 shows a possible version of this scenario. A vertical-edge attraction strategy draws the insect towards a fence pale, however, as the fence fills more of the

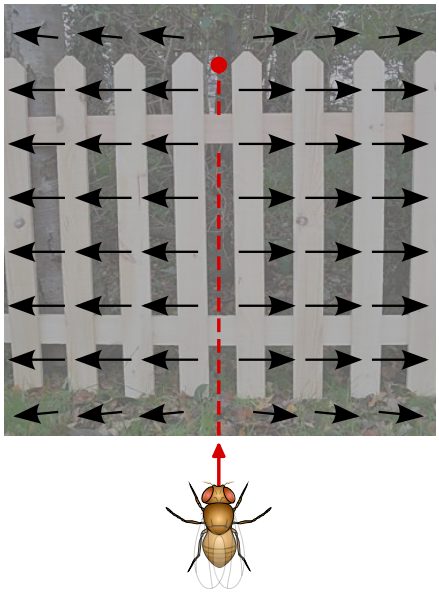


Figure 6.2: A schematic demonstrating plausible concurrent control strategies in insects; mediation of vertical-edge attraction and the optical flow balance strategy.

insect's vision, the optical flow balancing strategy dominates to keep the insect from collision.

In an obstacle avoidance context, the important measurement is depth to the objects one wishes to avoid. Optical flow as a proxy for measuring depth in ones environment was shown in the biological context in Section 2.1.3.6.

To recover a more accurate depth estimate I augment visual with inertial information. This too is biologically plausible [Sherman and Dickinson, 2004]; as discussed in Section 2.1.2.2.

6.1.2 Previous Work

For ground based vechicles, Coombs et al. [1998] and Camus et al. [1996] introduced the idea of navigation based on flow field divergence for wheeled robots. Here, the control responses were generated by balancing optical flow vectors according to a supplied template.

Hrabar et al. [2005] implemented a combined navigation and obstacle avoidance system. The authors used lateral cameras and optical flow regulation for navigation, stereo cameras and triangulation was implemented to compute distances towards detected objects. These two strategies had high computational complexity and required a powerful and heavy onboard computer.

Lateral obstacle avoidance controllers for a hovercraft using two one-dimensional lateral optical flow sensors have been developed in by several researchers [Srinivasan et al., 1996, Serres et al., 2008].

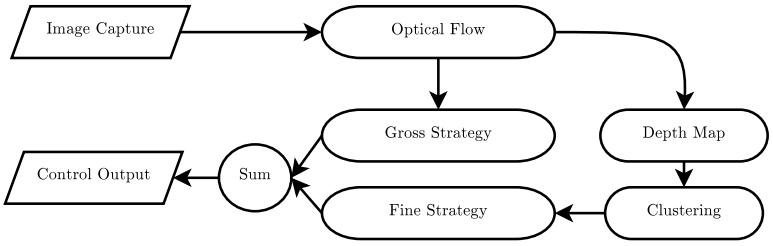


Figure 6.3: The control system architecture of the 'depth and optical flow balance strategy'.

Zufferey et al. [2007] implemented a single-camera frontal collision-avoidance strategy computing by the divergence of OF. OF has also been used for altitude control by several [Ruffier and Franceschini, 2008, Green et al., 2003] authors.

Neumann et al. [1997] trained a neural network to navigate a virtual corridor in 3D by learning the motion of the observer, sufficient to build a background model of the scene.

6.1.3 Implementation

I begin by estimating the distance to obstacles in the environment; constructing a depth map. This depth map is generated by measurement of optical flow in a uniform grid across the image. Grid regions are each analysed singularly, each one's depth is estimated by comparing their observed motion with the motion of the craft as measured by the inertial measurement unit.

Two control processes run simultaneously on the robot, a *gross* strategy uses divergence of optical flow vectors about the FOE, balancing these according to a predefined divergence template. A fine strategy looks for outliers from the estimated depth map. The sum of these processes generates a control impulse to steer a quadrotor helicopter away from obstacles.

The divergence template (*gross* strategy) was informed by Camus et al. [1996] and Coombs et al. [1998]. The fine strategy was informed most closely by Hrabar et al. [2005, Fig. 4]. However, instead of using stereo vision to compute frontal regions, I use independent estimates of depth from the horizontal and vertical components (u, v) of optical flow measurements[1]. The implementation of these two processes is explained in the forthcoming sections.

Figure 6.3 shows the system architecture. Experiments were undertaken using the ETH quadrotor system (Appendix A.4). 640×480 pixel images are captured from a forward facing Firefly MV camera (Appendix A.3.1), mated with a wide angle 180° fisheye lens. Onboard processing was performed using the Kontron SBC (Appendix A.3.4).

[1] inspired by the separable nature of the *Drosophila* horizontal and vertical pathways.

6.1.3.1 Computing Optical Flow

The Pyramidal *Lucas and Kanade* (Section 3.2.1) algorithm computes the optical flow across multiple regions of the image of equal size and spacing, called N_{plk}. This was a uniform grid of 12 columns and 9 rows. Considering the image was 640×480 this gave a region of interest per optical flow computation node of 53×53 pixels. The variance (Section 5.2) for each region is calculated and low values (regions with insufficient texture) are excluded from subsequent processing.

The *Lucas and Kanade* algorithm was chosen (Section 3.2.1), and the open source implementation from `OpenCV`, `cvCalcOpticalFlowPyrLK`, was used. The algorithm was performed over the N rectangular regions with a window size of 10×10.

6.1.4 Gross Optical Flow Process

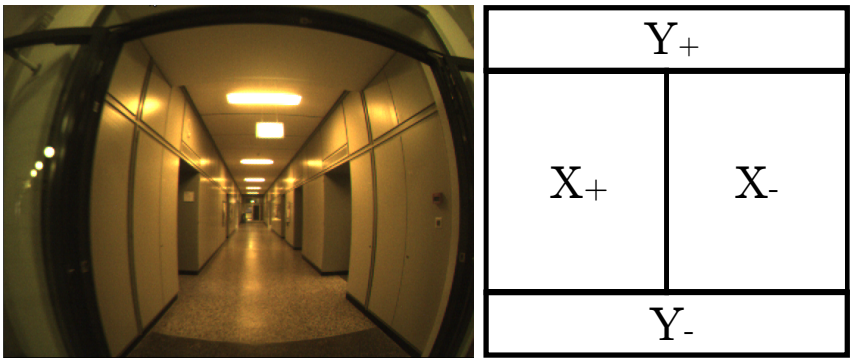


Figure 6.4: A schematic of the divergence template for indoor flight. (a) Indoor image. (b) Divergence template for wall, ceiling, and floor avoidance to control altitude Z, and lateral position Y.

I begin by extending the template approach suggested by Camus et al. [1996], Coombs et al. [1998] to two dimensions; to control both heading and altitude. The design of the divergence template shown in Figure 6.4 means that large motion on one side of the image (such as that observed when approaching a planar surface like a wall or ceiling from an oblique angle) should generate an opposing force. The shape of the motion template is chosen to constrain the forces. Rotation about the roll-axis creates flow vectors that are oriented tangentially, rather than radially, which effectively cancel out [Coombs et al., 1998]. Unlike Coombs et al., I use the motion template to implement a ground and ceiling avoidance response in parallel with the horizontal aspect.

The *gross* force vector is determined using;

$$\begin{cases} F_{gx} = \sum\limits_{x,y \in X_- \cup X_+} v_n(x,y) \\ F_{gy} = \sum\limits_{x,y \in Y_- \cup Y_+} v_n(x,y) \end{cases}, \tag{6.1}$$

where $v_n(x,y)$ is the image velocity, i.e., optical flow, computed over the grid region whose centre lies at x, y.

The total *gross* force $\mathbf{F}_g$ is computed as the sum of flow vectors in the sub-regions X_- and X_+ defined by the divergence template shown in Figure 6.4. A similar computation occurs for the sub-regions Y_- and Y_+.

6.1.5 Fine Optical Flow Process

Baraldi et al. [1989] showed that in the case of a translating camera and a static environment, a relative depth map of the scene can be generated from the known camera velocity using (6.2),

$$Z = V\frac{D}{I}, \tag{6.2}$$

where V is the velocity of the moving camera, D is the distance of the point (x,y) on the image plane from the FOE, I is the amplitude of the flow in (x,y), and Z is the depth of the point in the scene projected in (x,y).

Markel et al. [2002], Lopez et al. [2003] showed that by using an IMU a depth map can be estimated in the presence of more complex camera motion, such as pitching and yawing of the craft. I use the implementation derived by Markel et al. to estimate this depth map before performing a clustering step to identify obstacles to avoid.

6.1.5.1 Computation of Depth from Optical Flow

Figure 6.5 describes the geometry of the system. A model relating the optical flow components and the known object location in the image plane to the translational and rotational motion of the vehicle can be constructed [Markel et al., 2002]. The Markel et al. model describes a single object, however, I apply the concept to each of the N regions over which optical flow is calculated. As previously stated, the environment is assumed stationary.

The translational velocity $\mathbf{V}_t$ and angular rates ω of the camera in the camera frame of reference are taken from the IMU and given by the column vectors;

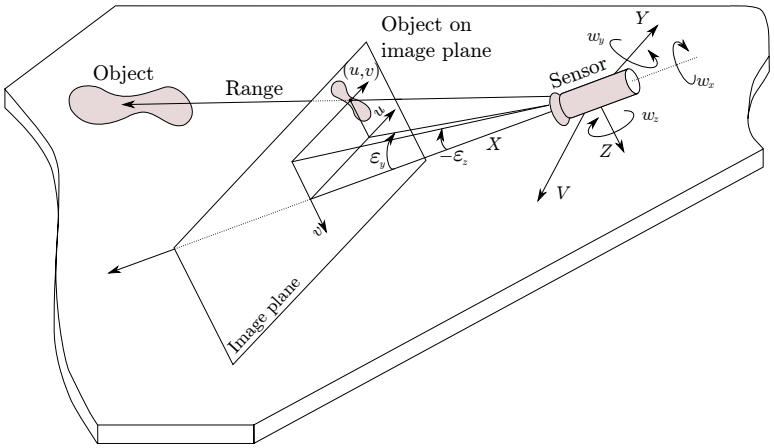


Figure 6.5: Geometry description showing the reference frames of the camera and control system.

$$\begin{aligned} \mathbf{V}_t &= [U, V, W]^\mathsf{T} \\ \omega &= [\omega_x, \omega_y, \omega_z]^\mathsf{T}. \end{aligned} \tag{6.3}$$

For a stationary object, the total velocity due to camera translation and rotation in the camera frame of reference is given by

$$\mathbf{V}_p = \mathbf{V}_t - \omega \times \mathbf{R}_p, \tag{6.4}$$

where

$$\mathbf{R}_p = [X, Y, Z]^\mathsf{T} \tag{6.5}$$

is the relative position of the object p with respect to the camera range-to-target, with the X axis along the optical axis, and

$$\mathbf{V}_p = [\dot{X}, \dot{Y}, \dot{Z}]^\mathsf{T}, \tag{6.6}$$

is the total velocity in the camera frame of reference. In component form the total velocity is given by

$$\begin{bmatrix} \dot{X} \\ \dot{Y} \\ \dot{Z} \end{bmatrix} = - \begin{bmatrix} U \\ V \\ W \end{bmatrix} - \begin{bmatrix} Z\omega_y - Y\omega_z \\ X\omega_z - Z\omega_x \\ Y\omega_x - X\omega_y \end{bmatrix}. \tag{6.7}$$

The location of the object in the image (pixel location) is given by the angles ϵ_y

and ϵ_z, as a function of range components

$$\begin{aligned}\frac{Y}{X} &= \tan(\epsilon_y) \approx \epsilon_y \\ \frac{Z}{X} &= \tan(\epsilon_z) \approx \epsilon_z,\end{aligned} \tag{6.8}$$

where $\approx$ represents the standard 'small angle' approximate equality. Because obstacles to avoid lie in the centre of the frame as approached, the small angle approximation employed here is reasonable.

Using the definition of the optical flow velocity components from (3.3) — (v_u, v_v), at the angular pixel location (ϵ_y, ϵ_z), taking the derivatives of (6.8) yields

$$\begin{aligned}(u, v) &\triangleq (\dot{\epsilon}_y, \dot{\epsilon}_z) \\ &= (\frac{\dot{Y}}{X} - \frac{Y\dot{X}}{X^2}, \frac{\dot{Z}}{X} - \frac{Z\dot{X}}{X^2}).\end{aligned} \tag{6.9}$$

Substituting (6.7) and (6.8) into the optical flow components given by (6.9), then separating the translation and rotational components yields the desired relationship between the optical flow velocities and the motion parameters for the case of stationary environment;

$$\begin{aligned}v_u &= \frac{1}{X}(-V + U\epsilon_y) + \omega_x\epsilon_z + \omega_y\epsilon_y\epsilon_z - \omega_z(1 + \epsilon_y^2) \\ v_v &= \frac{1}{X}(-W + U\epsilon_z) + \omega_x\epsilon_y + \omega_z\epsilon_y\epsilon_z + \omega_y(1 + \epsilon_z^2).\end{aligned} \tag{6.10}$$

Again, in (6.10) all parameters are assumed known except the down range component, X, and the optical flow components v_u and v_v for each pixel. From the *Lucas and Kanade* calculations, the optical flow components v_u and v_v are available, thus giving two estimates of range for each region in the image. Re-arrangement gives;

$$\begin{aligned}X_u &= \frac{-V + U\epsilon_y}{v_u - \omega_x\epsilon_z - \omega_y\epsilon_y\epsilon_z + \omega_z(1 + \epsilon_y^2)} \\ X_v &= \frac{-W + U\epsilon_z}{v_v + \omega_x\epsilon_y + \omega_z\epsilon_y\epsilon_z - \omega_y(1 + \epsilon_z^2)}.\end{aligned} \tag{6.11}$$

These calculations are repeated for each region in the image yielding a relative depth map for the environment.

6.1.5.2 Clustering of Computed Depth Map

Once the depth map is computed, the estimated depth is grouped into clusters using the k-means clustering algorithm. K-means clustering is a method of cluster analysis which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean.

Each observation is $\mathbf{x} = [X_u, X_v]^\mathsf{T}$, calculated in (6.11). The set of observations $(x_1, x_2, \ldots, x_n)$ contains the depth estimates for all grid regions in the image. Clustering aims to partition the observations into k sets, where $(k < n)$ and $S = S_1, S_2, \ldots, S_k$ so as to minimize the within-cluster sum of squares.

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \boldsymbol{\mu}_i\|^2 . \tag{6.12}$$

The most common algorithm uses an iterative refinement technique. Given an initial set of k means $(m_1^{(1)}, \ldots, m_k^{(1)})$, which may be specified randomly or by some heuristic, the algorithm proceeds by alternating between an assignment step and an update step.

Assignment Step

Assign each observation to the cluster with the closest mean (i.e., partition the observations according to the Voronoi diagram generated by the means).

$$S_i^{(t)} = \left\{ \mathbf{x}_j : \left\| \mathbf{x}_j - \mathbf{m}_i^{(t)} \right\| \leq \left\| \mathbf{x}_j - \mathbf{m}_{i^*}^{(t)} \right\| \text{ for all } i^* = 1, \ldots, k \right\}.$$

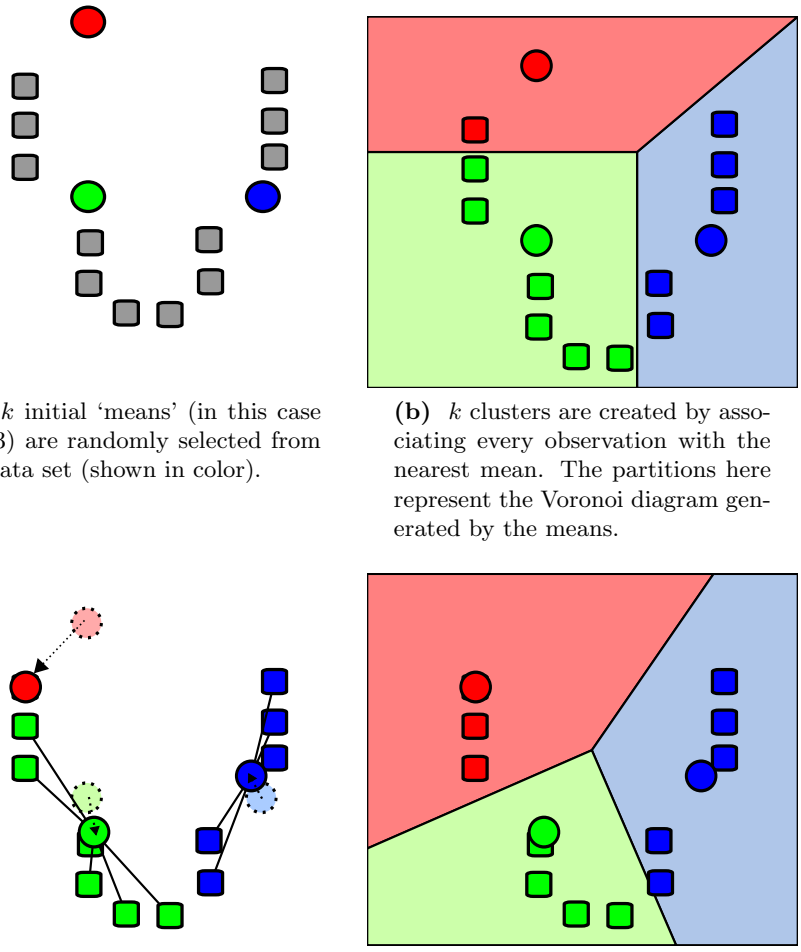
Update Step

Calculate the new means to be the centroid of the observations in the cluster.

$$\mathbf{m}_i^{(t+1)} = \frac{1}{|S_i^{(t)}|} \sum_{\mathbf{x}_j \in S_i^{(t)}} \mathbf{x}_j .$$

The algorithm is deemed to have converged when the assignments no longer change. However, because k-means is a heuristic algorithm, there is no guarantee that it will converge to the global optimum; the result depends on the initial clusters. Because the algorithm is usually fast, I run it several $(n = 3)$ times with different starting conditions. This is demonstrated visually using Figure 6.6.

Because k-means is a general clustering technique, its application depends on the set of observations $(x_1, x_2, \ldots, x_n)$. I am interested in differentiating objects based on their depth, thus each observation is $\mathbf{x} = [X_u, X_v]^\mathsf{T}$; the depth estimated from the optical flow velocity components calculated in (6.11). μ_i is the mean of points in the S_i^{th} cluster.



(a) k initial 'means' (in this case $k = 3$) are randomly selected from the data set (shown in color).

(b) k clusters are created by associating every observation with the nearest mean. The partitions here represent the Voronoi diagram generated by the means.

(c) The centroid of each of the k clusters becomes the new means.

(d) Steps (a) and (b) are repeated until convergence has been reached

Figure 6.6: A demonstration of the standard k-means algorithm.

6.1.5.3 Responding to Clusters

Equation (6.13) is applied to calculate the total fine force $\mathbf{F}_f$ from k detected obstacles (clusters, $S_1 \ldots S_k$). Unlike Hrabar et al. [2005], whose system responded to any obstacle detected, I introduce additional weighting terms to respond more strongly to large and nearby obstacles located around the focus-of-expansion (FOE), as it is these that pose the most immediate risk of collision if one is translating forward.

$$\mathbf{F}_f = \sum_{i=1}^{k} \frac{k_l}{S_i - \mathbf{FOE}} \times \text{size}(S_i) \times k_s \times ((D_{\max} - \text{depth}(S_i)) \times k_d)\,, \tag{6.13}$$

Here $\times$ indicates this is a vector operation performed on both (x, y) elements of the clusters. S_i is the centre of the cluster, size(S_i) is the size (number of grid regions) of the cluster, and depth(S_i) is the average depth to the i^{th} cluster respectively. $D_{\max}$ is the maximum distance to consider obstacles (so nearby obstacles are penalised more).

FOE is the focus-of-expansion (for details of how this is calculated see Section 5.4.1).

The scaling factors k_l, k_s, k_d, and $D_{\max}$ were experimentally derived and relate to the number of grid regions being analysed in the preceding step and the motion properties of the craft. For example, the large positive values of k_s and k_l were chosen to give the quadrotor a sufficiently large turn command that it avoided obstacles in front when flying forward at its' normal indoor velocity. The first gating term, $k_s = 0$, was chosen to minimise the effects of noise due to the incorrect assignment of small clusters from low optical flow magnitude grid regions. The second gating term, $k_l = 0$ was chosen to prevent large clusters at the edge of the image, such as the corridor walls, being counted twice — once in the gross process and again detected as obstacles here.

$$
\begin{aligned}
k_l &= \begin{cases} 0 & \text{if } S_i > 300 \text{ pixels from } \mathbf{FOE} \\ 1 & \text{otherwise} \end{cases} \\
k_s &= \begin{cases} 0 & \text{if size}(S_i) \leq 3 \text{ grid regions} \\ 5 & \text{otherwise} \end{cases} \\
k_d &= 10.
\end{aligned} \tag{6.14}
$$

In principle this control law is susceptible colliding with objects located at the FOE, where (6.13) generates zero corrective force (due to the contribution of $\frac{k_l}{S_i - \mathbf{FOE}}$). However, in practice this was not encountered in testing due to the inherent noise in the system; this is an unstable point and therefore, any deviation of the object's position away from the FOE results in a corrective force driving it further away.

6.1.6 Computation of Net Control Impulse

The total output force, $\mathbf{F}_t$, from the divergence template (*gross* process), $\mathbf{F}_c$, and the clustered depth map (fine process), $\mathbf{F}_f$, are then weighted to form the net control impulse,

$$\mathbf{F}_t = k_c \mathbf{F}_c + k_f \mathbf{F}_f. \tag{6.15}$$

The x component of the total force, $\mathbf{F}_{\text{tx}}$ is used to control the craft heading, while the y component, $\mathbf{F}_{\text{ty}}$ is used to control pitch. The weighting factors k_c and k_f were determined experimentally.

Additionally, a 10-element ($n = 10$), moving average filter is applied to the *gross* control impulse, $\mathbf{F}_c$

$$\mathbf{F}_c = \mathbf{F}_{c-1} - \frac{\mathbf{F}_{c-n}}{n} + \frac{\mathbf{F}_c}{n}. \tag{6.16}$$

6.1.7 Test Results

Data from the Rawseeds *Bicocca_2009-02-26a* dataset [Bonarini et al., 2006, Ceriani et al., 2009] were used to perform open-loop experiments; tuning the contributions (k_c and k_f of (6.15)) of the concurrent control processes to the final output.

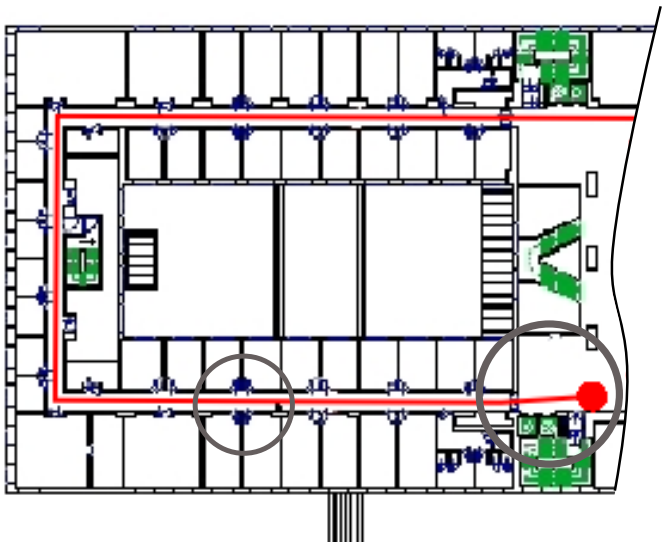
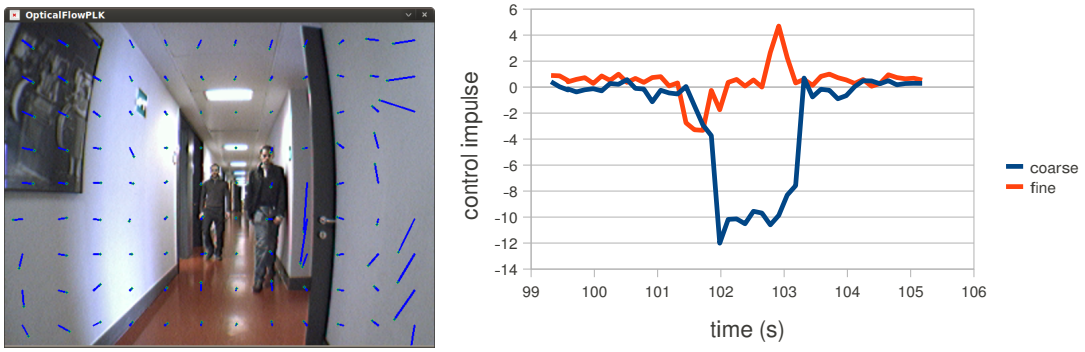


Figure 6.7: An overview of the Rawseeds *Bicocca_2009-02-26a* dataset. The dataset begins at the red circle and progresses clockwise through the indoor environment. Data from the corridor (small circle in the centre) was used in Figure 6.8. Data from the large open space (large circle in grey on the right of the figure) was used in Figure 6.9.

Figure 6.8 shows the contribution of each control process for a corridor section in the Rawseeds data (small circle, Figure 6.7). Note the two approaching obstacles (people) in the frame causes impulses in opposite directions. Initially the rightmost obstacle causes a large response by the coarse process due to its' size and proximity to the wall, while the second smaller obstacle is clustered and identified by the fine process.

Figure 6.9 shows another open-loop experiment from the same dataset; however this time it uses data from an open section of the Rawseeds data (large circle, Figure 6.7). Figure 6.9a indicates the robot's path as it moves through this section. Figure 6.9b shows the output of the control system as it traverses the path, based on calculating the optical flow shown in Figure 6.9c. Consider the images taken at $t = 18\,\mathrm{s}$ and $t = 24\,\mathrm{s}$ and the generated control impulse; at $t = 18\,\mathrm{s}$ the system generates a moderate strength corrective impulse towards the corridor entry, and once traversing the corridor the impulse is mostly neutral, keeping the robot in the centre of the corridor.

Several closed-loop experiments were also performed. These were undertaken at ETH in the corridor environment shown in Figure 6.4 and used the ETH quadrotor system described in Appendix A.4. The control system was given authority over heading and limited authority over pitch. Unlike the Rawseeds data analysis test, the real



(a) **(b)**

Figure 6.8: Evaluating the contribution of the *gross* and fine optical flow processes on Rawseeds data. (a) Observed image flow at $t = 99\,\text{s}$. (b) Contribution to total control impulse of heading, $\mathbf{F}_{\text{tx}}$, by *gross* and fine processes from $t = 99\,\text{s}$ to $t = 105\,\text{s}$. Negative values represent a turn to the left, positive values represent a turn to the right.

flight performance was disappointing. The gross process was able to navigate the corridor and avoid the walls, but oscillated in a stable limit-cycle manner between opposite sides of the corridor. Unfortunately, the large magnitude of motion from the limit cycling exceeded the signal from small obstacles, so they were not detected reliably while oscillating[2]. However, if the quadrotor was in a stable hover, small obstacles were detected.

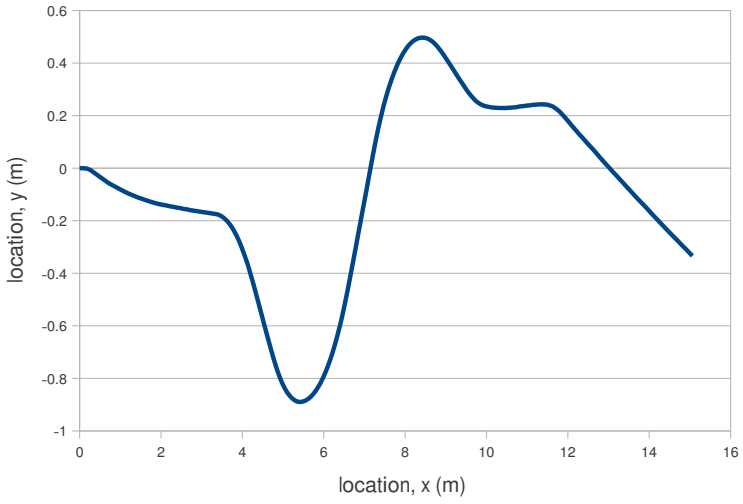
Considerable time was spent tuning k_c and k_f during these experiments; which in my experience, relate to the dynamics of the craft and to the structure of the environment. For example, the onboard PID heading controller[3] of the ETH quadrotor was highly damped, so the magnitude of turn command required was large, however, this induced limit-cycle oscillations as previously noted. k_f was related to the number of obstacles in the environment and the speed at which they approached. For example, one common feature of indoor flight was that people often approached the craft, which required quicker turning responses than if the obstacles were stationary. If the testing was being undertaken at night, with fewer obstacles, a lower value of k_f could be used.

6.1.8 Conclusions and Future Work

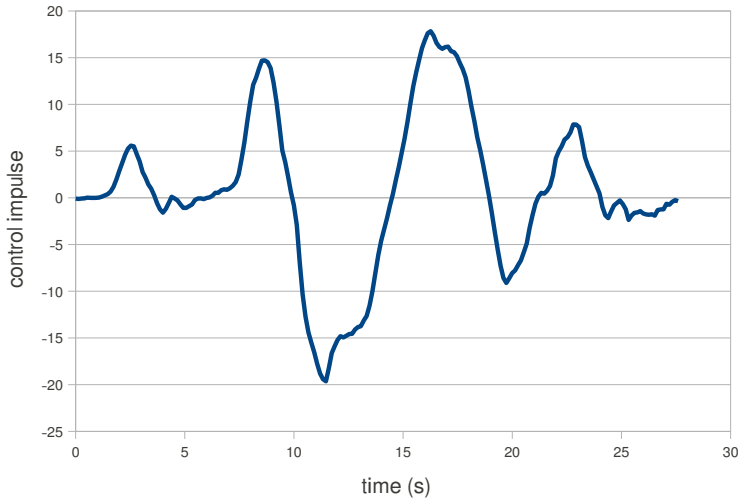
The combination of simulation and real flight testing showed this biologically inspired approach has merit. The current implementation is able to generate correct control impulses to steer the quadrotor around large obstacles, such as corridor walls and also avoid smaller obstacles as they are approached.

Real flight testing showed it necessary to dampen rapid control responses using a moving-average filter, (6.16), often in response to noisy image data. This was due

[2] In this regard, testing in a narrow corridor was a challenging flight environment.
[3] At that time the autopilot was provided by Asctec and not able to be customized.



(a)



(b)

(c)

Figure 6.9: Evaluating the performance of the obstacle avoidance and navigation control system on Rawseeds data for the period $t = 0 \ldots 25\,\mathrm{s}$. (a) Robot position detail. (b) Generated control force $F_t(x)$. (c) Optical flow at (left to right, top to bottom) 8,11,14,16,18 and 24 seconds.

to rapid craft movement causing blurred images, eventually leading to failure of the control system. This damping reduced the utility of the control system however. Such damping is not without biological precedent however, as described in Section 2.1.4.8, insects must frequently inhibit conflicting behavioural responses.

Due to the dependence of k_c and k_f on the scene in which the quadrotor was flying, in a real biological system or in an extension of this work, their value should be adjusted at runtime to suit the broad nature of the environment (how many obstacles are expected, how nearby are walls).

One of the attractive features of a system of this architecture was the highly structured input to the control system. A regular grid of optical flow regions, Figure 6.9c, is well suited to experimenting with other control architectures.

6.2 GROUND AND PLANE DETECTION

To navigate their environment; flying animals need to control their altitude as well as their horizontal motion. The absolute altitude (above sea level) matters little; it is the height above ground, h, that is important. Previous authors have used *ventral* optical flow to estimate altitude [Neumann and Bülthoff, 2002, Webb, 2007] and for terrain following [Ruffier and Franceschini, 2004, Srinivasan et al., 2006]. I propose that a model based approach utilising knowledge that the ground is locally flat can be used to efficiently estimate altitude.

This chapter introduces a method for finding the ground plane in range data. In these experiments I use the Microsoft Kinect sensor, which uses a structured light method to measure depth directly.

I present a sampling strategy and algorithm for finding the dominant plane below the observer (presumed to be the ground plane) and use the resulting information to control the quadrotor altitude. Aspects of the sampling strategy are biologically inspired by the wide-field integration patterns of LPTC cells (Section 2.1.3.4).

6.2.1 Previous Work

The application of plane detection in robotics is not very common - the preference given to more general mapping schemes such as SLAM. However, for humanoid robots detection of the inclination of the ground plane is necessary for robust locomotion. Okada et al. [2001], Lim et al. [2008] apply iterative Randomized Hough transform on depth data (sonar) for this task with some success.

Most Hough transform applications have been in the field of photogrammetry, aerial surveying, medicine, and geography. Bi et al. [2009] is the application of the Hough transform for the detection of planes in helical single slice CT (computed tomography) scans.

Sarti and Tubaro [2002] is a thorough introduction in the use of the 3D Hough transform for the detection of 3D planes in binary data; in this case volumetric density data obtained from X-ray tomography of bedrock.

Belmans et al. [2009] and Deklerck et al. [2010] utilise the Hough transform in order to detect orthogonal planes in range data. Given three orthogonal planes it is shown that it is possible to calibrate the camera's extrinsic parameters.

There are many works discussing the different variations on the Hough transform in an algorithmic context. Kälviäinen et al. [1995], Belmans et al. [2009] both include discussions of extensions to the Hough transform including; the probabilistic Hough transform, the progressive probabilistic Hough transform, and the inverse progressive probabilistic Hough transform.

The most similar work is that of Borrmann et al. [2011], who discuss several Hough transform techniques for plane detection in point clouds. They introduce a new accumulator design with better cache coherency and access attributes that speeds up computation when the number of points is very high.

A review of different Hough transform techniques is provided in Section 6.2.3.

6.2.2 Plane Geometry

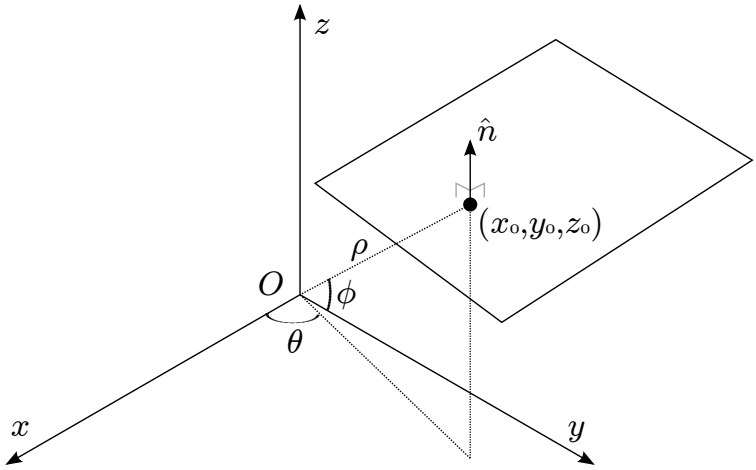


Figure 6.10: The representation of a plane and its normal, $\hat{\mathbf{n}}$, in 3D space. $\mathbf{x_0} = (x_0, y_0, z_0)$ is a point on the plane.

Consider Figure 6.10. A plane can be represented [Vince, 2004, sec. 1.15] by the signed distance ρ to the origin of the coordinate system and the slopes m_x and m_y in the direction of the x-axis and y-axis, respectively:

$$z = m_x x + m_y y + \rho. \tag{6.17}$$

However, this representation is prone to numerical issues. To avoid problems with

infinite slopes when trying to represent vertical planes (a common problem) the normal form of the plane is preferred. The equation of the plane is then given by

$$\rho = \mathbf{p} \cdot \mathbf{n}, \tag{6.18}$$

$$\rho = p_x n_x + p_y n_y + p_z n_z, \tag{6.19}$$

where $\mathbf{p} = (p_x, p_y, p_z)$ is a point on the plane and $\mathbf{n} = (n_x, n_y, n_z)$, a normal vector perpendicular to the plane. The distance to the origin is ρ.

It is convenient to specify planes in Hessian normal form. If $|\mathbf{n}| = 1$ and $\rho \geq 0$, the unit normal vector $\hat{\mathbf{n}} = (n_x, n_y, n_z)$ is

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{\| \mathbf{n} \|}, \tag{6.20}$$

and its components

$$n_x = \frac{a}{\sqrt{a^2 + b^2 + c^2}} \tag{6.21}$$

$$n_y = \frac{b}{\sqrt{a^2 + b^2 + c^2}} \tag{6.22}$$

$$n_z = \frac{c}{\sqrt{a^2 + b^2 + c^2}}, \tag{6.23}$$

giving the constant

$$\rho = \frac{d}{\sqrt{a^2 + b^2 + c^2}}, \tag{6.24}$$

and thus the Hessian normal form of the plane is

$$\rho = \hat{\mathbf{n}} \cdot \mathbf{x}. \tag{6.25}$$

Consider the angles between the normal vector, $\mathbf{n}$, and the coordinate system of Figure 6.10. In spherical polar form, angles θ and ϕ, (6.25) can be written as

$$\rho = p_x \cos\theta \sin\phi + p_y \sin\theta \sin\phi + p_z \cos\phi, \tag{6.26}$$

with the components of $\hat{\mathbf{n}} = (n_x, n_y, n_z)$,

$$\begin{aligned} n_x &= \cos\theta \sin\phi \\ n_y &= \sin\theta \sin\phi \\ n_z &= \cos\phi. \end{aligned} \tag{6.27}$$

Hence, θ, ϕ, ρ define the plane.

6.2.2.1 Finding the Equation of a Plane from 3 Points

Given a list of points $\mathcal{P}$ in 3D, one can find the equation of the plane spanning any 3 non-colinear points [Vince, 2004, sec. 1.15]. Let $\mathbf{p_1} = (x_1, y_1, z_1)$, $\mathbf{p_2} = (x_2, y_2, z_2)$, and $\mathbf{p_3} = (x_3, y_3, z_3)$ (Figure 6.11).

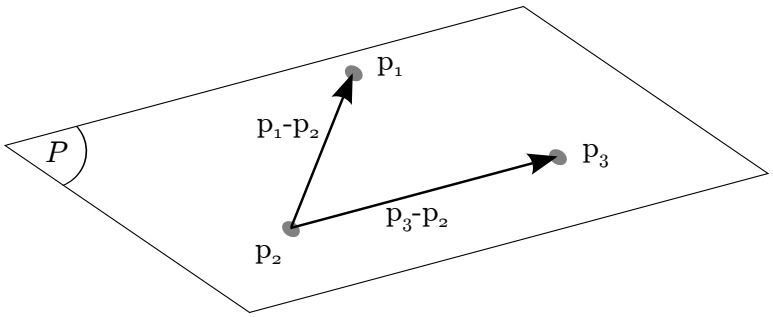


Figure 6.11: Constructing a plane from three points.

The normal to the plane spanned by these points is calculated with the cross product

$$\mathbf{n} = (\mathbf{p_3} - \mathbf{p_2}) \times (\mathbf{p_1} - \mathbf{p_2}). \tag{6.28}$$

The unit vector to the plane is thus

$$\hat{\mathbf{n}} = \frac{\mathbf{n}}{\|\mathbf{n}\|}, \tag{6.29}$$

where $\hat{\mathbf{n}} = (n_x, n_y, n_z)$ and using (6.19), ρ is computed as

$$\rho = \hat{\mathbf{n}} \cdot \mathbf{p_1}. \tag{6.30}$$

Rearranging (6.27) I compute the remaining plane parameters

$$\phi = \cos^{-1} n_z \tag{6.31}$$

$$\theta = \sin^{-1} \frac{n_y}{\sin \phi}. \tag{6.32}$$

6.2.3 The Hough Transform

The Hough transform [Hough, 1962] is a method for detecting parametrised objects in data; such as lines or ellipses in the 2D case and planes in the 3D case. The principle of this technique is the mapping of a set of points initially defined in Euclidean space into another parameter space; often called the Hough space. In doing so, certain geometric primitives can be detected with improved computational efficiently. In practice, the Hough transform converts a complex pattern detection problem in the image space into a more manageable peak detection problem in the parameter space [Sarti and Tubaro, 2002].

For example, consider the normal form of a line;

$$\rho = x\cos\theta + y\sin\theta, \tag{6.33}$$

where θ and ρ are the parameters of the normal passing through the origin (Figure 6.12).

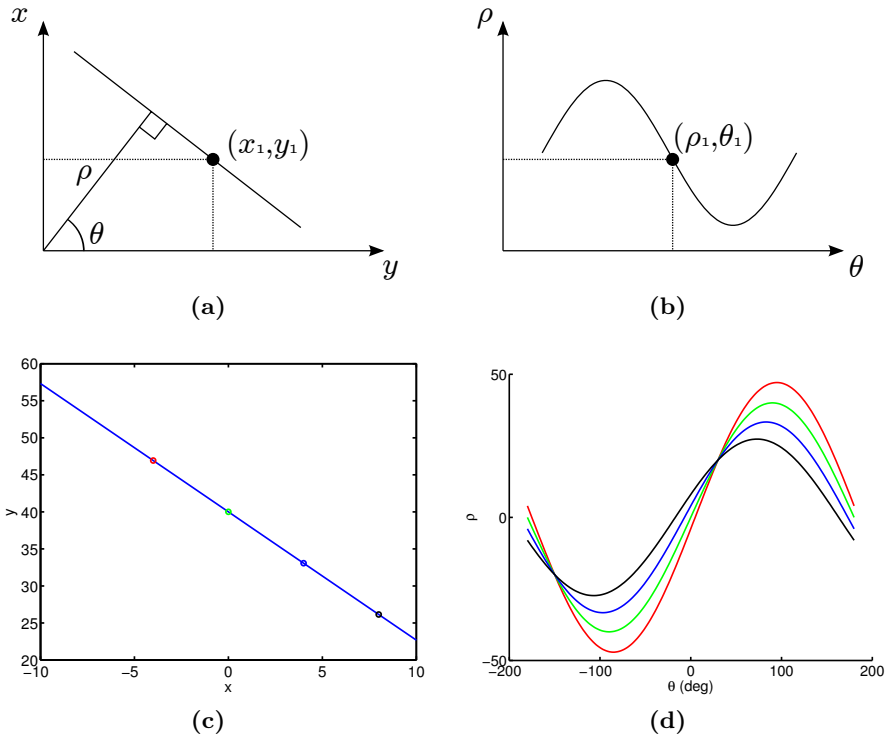


Figure 6.12: A comparison of a line in Cartesian space and Hough space (a,b). A demonstration of how each point on the same line causes an intersection in parameter space (c,d). (a) The line P and its normal in 2D space. (b) A point on the same line in the parameter space, represented using the normal form. (c) A line and several points along that line in 2D space. (d) The same points now parameter space; note the intersections of the curves, for each x, y there exist two possible solutions, forward and back of $\rho = 0$.

The parameter space in this case is $\theta, \rho \in \mathbb{R}^2$ and one point $\mathbf{p_1} = (x_1, y_1)$ in 2D space represents a sinusoid in parameter space (Figure 6.12). Detection of multiple sinusoid curves passing through the same point in parameter space allows one to detect straight lines in 2D space.

The same concept can be applied in the 3D case to detect planes in range data [Overby et al., 2004, Vosselman et al., 2001]. Consider the plane in (6.26) whose parameter space is (θ, ϕ, ρ). One plane in 3D space represents a sinusoidal surface in the parameter space (Figure 6.13). Detection of multiple similar sinusoidal surfaces allows one to detect planes in 3D space.

For detection the parameter space is discretized into N_θ, N_ϕ, N_ρ values for θ', ϕ', ρ'. A datastructure called an accumulator (A) stores a score for each of these cells.

In the standard Hough transform (SHT) each point votes for all sets of parameters (θ', ϕ', ρ') on which it may lie. After all points have been processed, the cells with the highest values represent the most prominent planes; those that cover the most points of the cloud.

The SHT has many limitations for real-time use. It has high computational complexity; for an $M \times N$ image it is of the order $O(MNN_\theta N_\phi N_\rho)$ [Xu and Oja, 1993]. The determination of peaks in the accumulator is difficult; discretisation means planes may occupy many adjacent cells, so a sliding window type search must be used to find the most prominent regions. It also has high memory requirements since a 3D accumulator has $N_\theta \times N_\phi \times N_\rho$ cells for storage.

6.2.3.1 The Hough Transform for 2D and 3D

Consider a 2D Hough transform; for example a line is represented in $\mathbb{E}^2$ space as

$$y = a \cdot x + b, \tag{6.34}$$

where (a, b) are the parameters of the line. The line can be represented by a point with coordinates (a, b) in the parameter space $\mathbb{R}^2$; $a, b \in \mathbb{R}^2$. Similarly, a point (x_i, y_i) in $\mathbb{E}^2$ is represented by a line in $a, b \in \mathbb{R}^2$ as

$$b = -x_i \cdot a + y_i. \tag{6.35}$$

However, if the line has the form $x =$ constant then it cannot be present in parameter space $a, b \in \mathbb{R}^2$, because the y-axis coefficient is zero. For this reason the normal form of the line is used;

$$\rho = x \cos\theta + y \sin\theta, \tag{6.36}$$

where θ and ρ are the parameters of the normal passing through the origin, and are constant for one line. Hence the parameter space in this case is $\theta, \rho \in \mathbb{R}^2$ and one point $p_1 = (x_1, y_1)$ in 2D space represents a sinusoid in parameter space — shown in Figure 6.12. During analysis, detection of multiple sinusoid curves passing through the same point in parameter space allows one to detect straight lines in 2D space.

The same concept can be applied in 3D space [Overby et al., 2004, Vosselman et al., 2001] to detect planes in range data. Consider one plane belonging to 3D space $\mathbb{E}^3$ can be represented by a point (a, b, c) in parameter space $a, b, c \in \mathbb{R}^3$,

$$z = a \cdot x + b \cdot y + c. \tag{6.37}$$

Analagous to the reasons explained in the 2D case above (instead the z-axis coefficient is equal to zero) and further justified in Overby et al. [2004], the normal form of the

plane is preferred

$$0 = a \cdot x + b \cdot y + c$$
$$\rho = x \cos\theta \cos\phi + y \sin\theta \cos\phi + z \sin\phi, \tag{6.38}$$

where θ,ϕ and ρ are the parameters of the plane normal passing through the origin (Figure 6.10) and are constant for one plane.

Hence the parameter space in this case is $\theta, \phi, \rho \in \mathbb{R}^3$, and one plane P in 3D space represents a sinusoidal surface in the parameter space. Detection of the peaks of the sinusoidal surfaces thus allows detection and yields the parameters of planes in 3D space. Figure 6.13 demonstrates the representation of a plane and its value in the Hough accumulator.

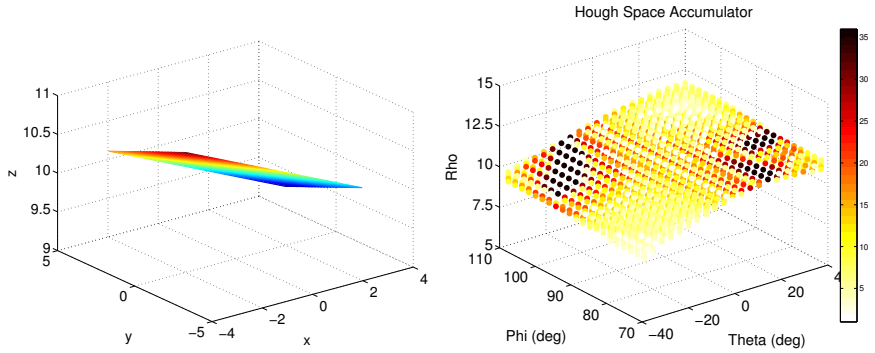


Figure 6.13: A comparison of a plane in Cartesian space and as represented in the Hough accumulator array. (a) Presentation of plane P. (b) Hough accumulator array of plane P.

6.2.3.2 The Randomized Hough Transform

The randomized Hough transform (RHT) [Xu et al., 1990] based plane detection begins with the premise that a single plane can be determined uniquely with three points from the range data. These three points in 3D space are mapped into one point in the parameter space corresponding to the plane spanned by the three points.

Algorithm 1 explains the implementation. The parameter space $\theta, \phi, \rho \in \mathbb{R}^3$ is discretized into N_θ, N_ϕ, N_ρ values. In each iteration, three points $\mathbf{p_1}$, $\mathbf{p_2}$, and $\mathbf{p_3}$ are randomly selected from $\mathcal{P}$. Using the technique described in Section 6.2.2.1; eqs. (6.28), (6.30) and (6.32), the parameters, (θ, ϕ, ρ), of the plane corresponding to these three points is calculated.

The corresponding cell in the accumulator $A(\theta, \phi, \rho)$ is incremented. If a threshold T_A is exceeded by the score in the cell then a plane is detected. Otherwise, the algorithm continues until all the points have been processed or a maximum number of iterations T_I is reached.

Algorithm 1 Randomized Hough Transform

```
repeat
  choose a point, p1, from the input list at random.
  repeat
    choose a point, p2, in a neighbourhood around p1
    choose a point, p3, in a neighbourhood around p1
  until points p1,p2,p3 satisfy the distance constraint
  calculate the plane, P, spanned by these points
  increment accumulator recording the parameters of P
  if accumulator value for P is > T_A then
    mark P as detected
  end if
  mark points points p1,p2,p3 as tested
until sufficient planes detected
```

The computational complexity of finding the biggest plane with area m is approximately $O(\min(m^3 T_A, T_I))$, which is independent of the size of image and quantification steps [Xu and Oja, 1993, Ding et al., 2005]. Compared to the SHT, the RHT can detect planes more efficiently.

6.2.4 Implementation

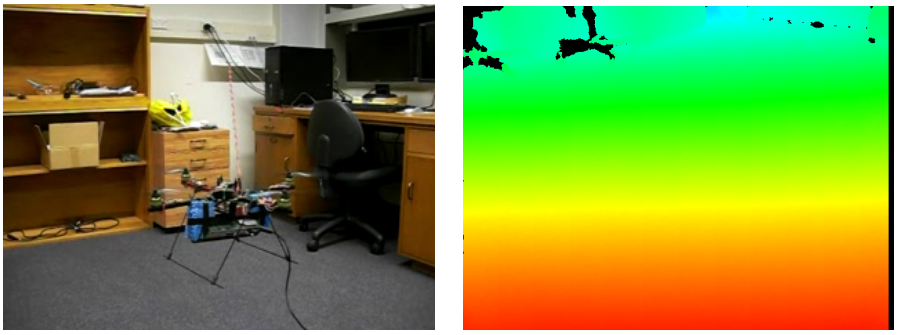
The algorithm was verified initially on synthetic data (images and depth generated via ray-tracing, Appendix C.3) and subsequently on real-collected datasets of the flight test environment (Figure 6.14a).

Closed-loop experiments were undertaken using the 'wasp' quadrotor configured with the SBC as described in Appendix A.4. To capture depth images a Microsoft Kinect sensor (Appendix A.3.2) was used. The sensor was mounted below, and looking at an oblique angle towards the front, as shown in Figure 6.14. A calibration (Appendix A.3.2.1) and median filter (Appendix A.3.2.2) was applied to the depth images before computing the Hough transform.

Given a calibrated Kinect depth camera (Appendix A.3.2.1) the locations of a 3D point $\mathbf{p} = [x, y, z] \in \mathbb{E}^3$ can be recovered. From the pixel coordinates (u, v), and using (2.8), the elements of $\mathbf{p}$ are;

$$\begin{aligned} x &= (u - c_x)\frac{\text{depth}(u,v)}{f_x} \\ y &= (v - c_y)\frac{\text{depth}(u,v)}{f_y} \\ z &= \text{depth}(u,v), \end{aligned} \tag{6.39}$$

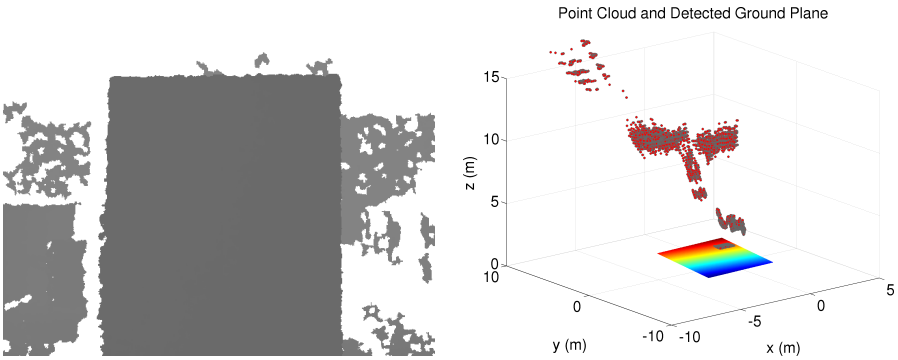
where depth is (A.5) and the lens parameters c_x, c_y, f_x, f_y are described in Table A.3. Figure 6.15 shows values of (x, y, z) for a calibrated Kinect image. With three such



(a) **(b)**

Figure 6.14: (a) The modified Kinect mounted on the 'wasp' quadrotor helicopter. (b) The false color depth image measured.

calibrated points obtained in this way, I calculate a candidate plane that spans them using eqs. (6.30) and (6.32) as described in Section 6.2.2.1, before using the RHT to find the true planes in the depth data.



(a) **(b)**

Figure 6.15: A Kinect image and the corresponding calibrated points in 3D space. (a) An image taken of a workshop scene, using the Kinect. A large flat plane is placed in front of the camera at a distance of 1 m. (b) Pixels in the depth image calibrated and projected into 3D (grey and red circles). The RHT has successfully detected the plane at 1 m (brightly coloured).

The biggest plane has the largest probability of being detected and because of the orientation of the Kinect camera (looking predominately at the ground), the biggest plane should be the ground plane.

Figure 6.16 shows the algorithm applied to a single image from a sequence captured during flight. Visible in Figure 6.16b, the RHT has reconstructed the ground plane from the point cloud, Figure 6.16a.

Suppose $\frac{1}{m^{\text{th}}}$ of the points in $\mathcal{P}$ lie on the biggest planar surface. The probability of the three randomly selected points simultaneously belonging to the biggest plane is thus $\frac{1}{m^3}$.

I first implemented the RHT using MATLAB, as described in Section 6.2.3.2, but

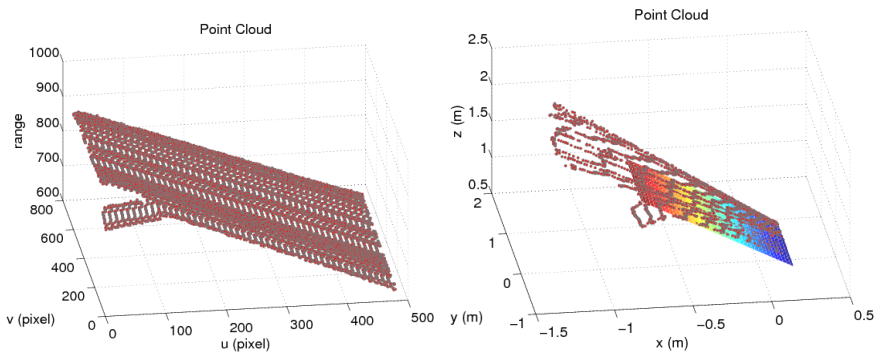


Figure 6.16: Processing stages in detecting the ground plane. (a) Range data point cloud in the camera frame. (b) Point cloud transformed into world coordinate frame with the detected ground plane overlaid.

with several notable differences. Unlike the canonical implementation of Xu et al. [1990], my implementation does not back-calculate and remove from $\mathcal{P}$ all points that correspond to $A(\theta, \phi, \rho)$ once T_A is reached. I only want to detect the largest plane (the ground plane) and therefore, the algorithm terminates once T_A is met.

From Section 6.2.3.2, algorithm 1, and in particular when detecting only the dominant (ground) plane, the important variables which control the behaviour of the algorithm are; the number of buckets (discretized into N_θ, N_ϕ, N_ρ values), and the threshold, T_A, at which a plane is considered detected. Figure 6.17 demonstrates this. The accumulator threshold and the number of buckets both affect the runtime linearly; this is intuitive, the more planes need to be tested, the longer the algorithm takes. All four variables affect the accuracy of the result. T_A determines the robustness of the result, not necessarily the absolute accuracy; a low value means incorrect planes, possibly due to noise, may be detected in the data, a high value requires more agreement in the dataset. The number of buckets has a direct relationship on the absolute accuracy of detection; this is as expected as the number of buckets determines the resolution (because of quantization) of the result. From these experiments I found the values of $T_A = 10$, $I = J = K = 200$ to be acceptable.

The MATLAB RHT prototype was reimplemented using `C++`. From the recommendations of Borrmann et al. [2011] and based on the low number of planes tested when detecting the ground plane (Figure 6.17), I implemented a custom accumulator. Unlike the author's work (an improved mapping to an in-memory array to preserve resolution) I chose a sparse tree structure; the first two layers storing θ and ϕ and the leaves of the tree storing accumulator values per ρ. This structure is very efficient for the use-case of detecting the dominant (ground) plane.

Once I have an accurate estimate of the plane parameters (θ, ϕ, ρ) from the RHT I calculate the altitude, h, above the ground. More correctly, if the Kinect is located

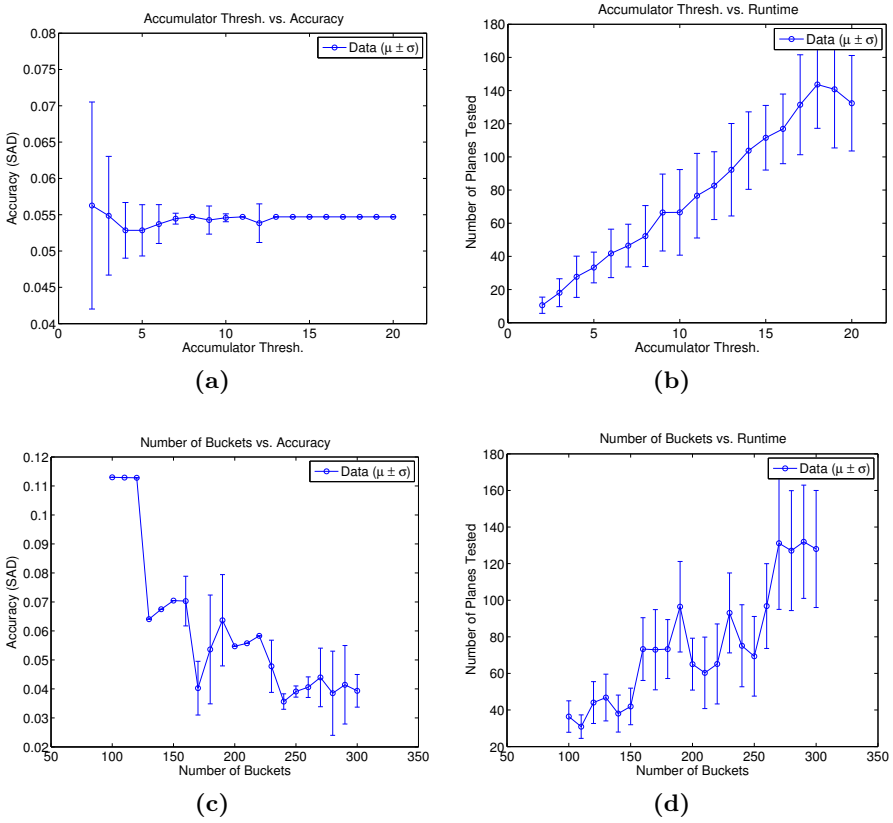


Figure 6.17: The effect of the critical parameter on the RHT; the accumulator threshold, T_A (a,b), the number of buckets, N_θ, N_ϕ, N_ρ (c,d). SAD is a measure of accuracy; the sum of absolute differences in the plane parameters, θ, ϕ, ρ, relative to the parameters of the true plane. The number of planes tested is synonymous with the algorithm runtime as it is the trigonometry calculations which take the majority of the time. The dataset was gathered in my office, an image and detection of the ground plane can be seen in Figure 6.14 and Figure 6.16 respectively. Each test was performed 50 times ($n = 50$), these results show the mean and standard deviation. (a) The effect of T_A versus the accuracy of plane detection. (b) The effect of T_A versus the number of candidate planes. (c) The effect of the number of buckets, N_θ, N_ϕ, N_ρ, versus plane detection accuracy. (s) The effect of the number of buckets, N_θ, N_ϕ, N_ρ, versus the number of candidate planes.

at the origin O in Figure 6.10 then

$$h = \rho / n_z. \tag{6.40}$$

Appendix B.4.1.3 explains the 'wasp' quadrotor altitude controller which uses the barometer and sonar (if attached). In this experiment it was replaced by another PID controller (Appendix B.4.1, (B.16)) running on the SBC. From (B.16); the PID output controls the thrust, $u(t) = U_1$, the estimated altitude is that calculated from the parameters of the ground plane, $\hat{Z} = h$, and $K_p = 5$ and $K_I = 1$ are the proportional and integral control gains determined experimentally. The SBC executed the control system at $\frac{1}{20\,\text{Hz}} = 50\,\text{ms}$.

6.2.4.1 Biologically Inspired Non-Random Hough Transforms

The 'random' choice of 3 candidate points also has an affect on the runtime of the algorithm. A 'naive' strategy of choosing 3 points independently located anywhere in the image is not optimal. This result is intuitive; 3 points which are randomly chosen yet lie very close together in space will be more susceptible to noise (both random and due to quantization). At the other extreme, points randomly chosen that lie far apart may not be co-planar[4] due to the structure of the environment — if one wishes to detect only the dominant plane, this situation should be minimised as this represents a wrong result. Therefore, it follows that the method of sampling points from which to compute their plane is important to the absolute accuracy and runtime of the algorithm.

Sections 2.1.3.3 and 2.1.3.4 explained how insects use non-random and non-uniform sampling; encoded via the connection patterns of *neurons* in the LPTC cells to distinguish translational and rotational components of the optical flow field. Similarly, this section presents strategies for improving the efficiency of ground plane detection by selecting points for the Hough transformation in a non-random manner. The following describe the non-random sampling strategies employed and the results of each.

Limited Local Sampling: Borrmann et al. [2010] introduced the idea of a 'distance criterion'; the basis of which is that points closer together are more likely to belong to the same plane than those more distant. On the other hand, points which are too closely located negatively affect the accuracy of the result. The authors offered this only as a general principle, not specifying the implementation. One can think of two domains in which to enforce this criterion; in Cartesian 3D world coordinates and in 2D image plane coordinates.

[4] I mean co-planar in the environment; because strictly speaking, the 3 points chosen randomly by the RHT algorithm are of course assumed to be co-planar when a candidate plane is fit to them.

Cartesian coordinates: Once the location of points in (X, Y, Z) has been recovered using the calibrated Kinect, depth points which lie far away in 3D space are ignored. Formally, only points where $\text{dist}(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}) \leq \text{dist}_{\max}$, where dist is the Euclidian point-to-point-to-point distance of all three points and $\text{dist}_{\max} = 4\,\text{m}$ are considered as candidate planes for the RHT. The value of 4 m was chosen empirically as the maximum flying height in the test environment and as less than the maximum sensor range of the Kinect. Enforcing the distance criterion in this domain has a non-negligible runtime cost; in evaluating (6.39) one must compute the camera transformation and particularly, the calibration/scaling of the Kinect depth measurement, (A.5).

A better approach would be to apply a comparable and useful 'distance criterion' test in the image coordinate system.

Image coordinates: Given that I need to only detect the dominant (ground) plane and that it should consistently fill a large proportion of the image, I pre-compute many 3-tuples of image coordinates and from this pool I select tuples until the T_A is reached and a plane is detected. By ensuring the 3-tuples are co-located closely in the image plane I reciprocate the intent of the 'distance criterion' suggested above.

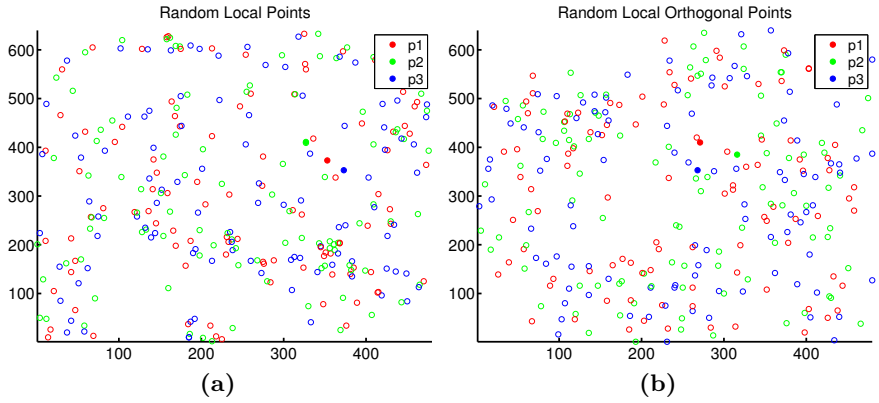


Figure 6.18: A comparison of local sampling strategies for plane detection. 100 3-tuple $(\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3})$ are shown as circles. Solid points (centre-right) indicate a single tuple. (a) The 'local (image)' sampling strategy. (b) The 'orthogonal' sampling strategy..

I compared two strategies for pre-selecting these 3-tuples in the image plane with the conventional 'naive' approach. Examples of these strategies are shown in Figure 6.18.

The first strategy, 'local (image)', uses 3 points; the first chosen from a uniform distribution over the entire image, the second two points are chosen from Gaussian distributions centred about the first. This is essentially the distance criterion pre-emptively applied in the image domain at the sampling stage (when one assumes the dominant plane constitutes most of the image).

The second strategy, 'orthogonal', uses 3-tuples of points chosen to be predominately orthogonal and near to each other, distributed uniformly over the image. This is reminiscent of the sampling patterns in the *lobula plate*; I designed this arrangement to maximise accuracy of dominant plane detection while minimising runtime. After choosing the first point randomly (uniform distribution), the second point is placed at a randomly chosen (uniform distribution) angle from the first, at a distance chosen from a gamma distribution. This distribution is non-symmetric and peaks a positive distance from the first point. This ensures the second point is located reasonably close to, but not too near the first point. The third point is orthogonal to the first two at the same random distance.

Figure 6.19 demonstrates that the 'orthogonal' sampling strategy performs just as well as the distance criterion suggested by Borrmann et al. [2011], while performing less computation.

6.2.5 Altitude Control Results

The `C++` RHT was run on the 'wasp' Kontron SBC to control the altitude in real time. The implementation was as previously described, using the orthogonal sampling strategy.

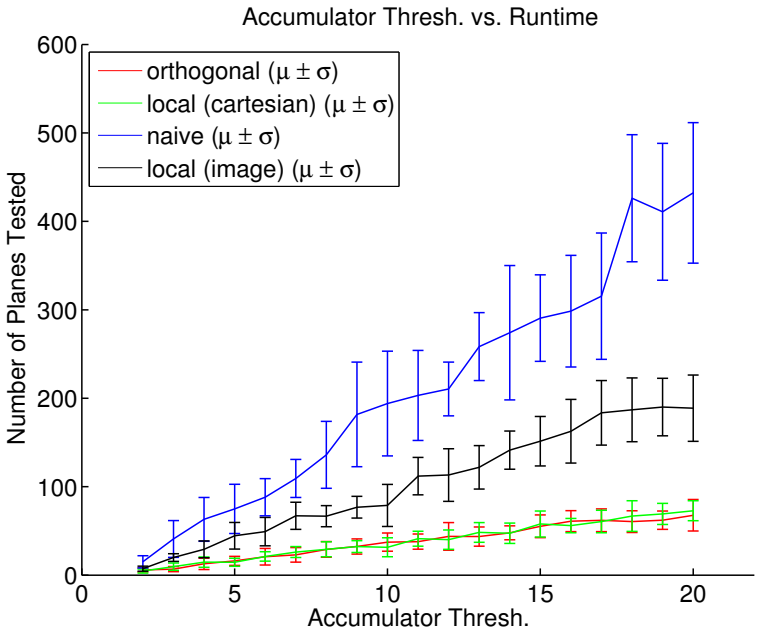
Figure 6.20 shows the performance of the control system. The quadrotor was commanded to hover at fixed altitudes above the laboratory floor. The oscillation in altitude shows that while the PID controller is not optimal for this task, it allowed the quadrotor to regulate attitude (at different setpoints) for the duration of the tests (40 s).

The execution time and the memory usage are important considerations when using the algorithm in a real-time system. Because of the random nature of the sampling the runtime and memory usage can vary between iterations.

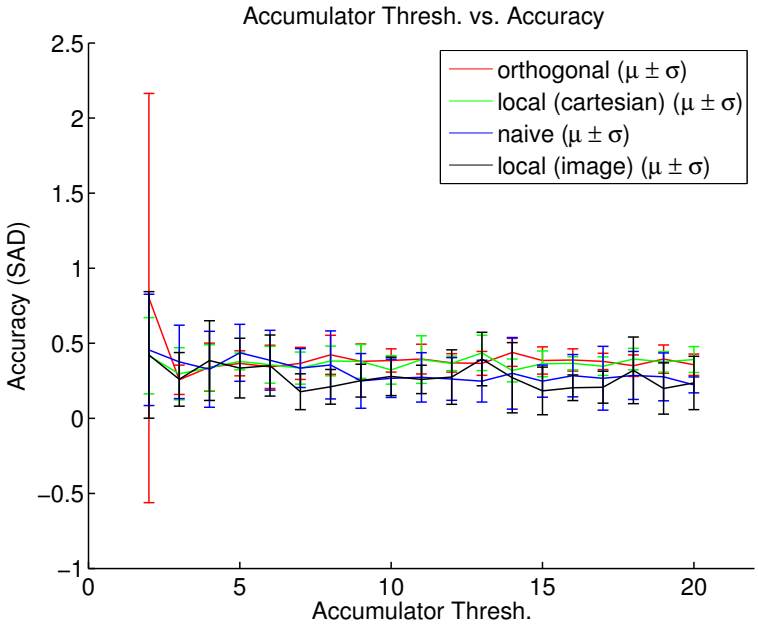
Figure 6.21 shows the execution time to find the ground plane in a sequence of test images, repeated 20 times per image. This demonstrates that the execution time of the RHT algorithm is 0.5 ms. Figure 6.21c shows the execution time when the ground is shown in a diminishing proportion of the image. While a slight trend is present, the dominant property is that only a marginally longer runtime occurs despite the ground only being visible in a small proportion of the image. This constant runtime demonstrates the effect of the importance sampling approach taken in selecting points to consider.

6.3 SUMMARY

This chapter described two visual approaches for controlling flight. Section 6.1 presented a strategy whereby the output of two concurrent visual control processes was



(a)



(b)

Figure 6.19: The accuracy and runtime of different sampling strategies for plane detection. 'Naive' is sampling from the whole image, 'local (cartesian)' is the application of the distance criteria in 3D space. 'local (image)' is the distance criteria in the image domain, and 'orthogonal' is the biologically inspired strategy. The test image was that shown in Figure 6.14b. (a) The number of planes tested before detection (the runtime). (b) The accuracy of the plane detected, SAD as per Figure 6.17 .

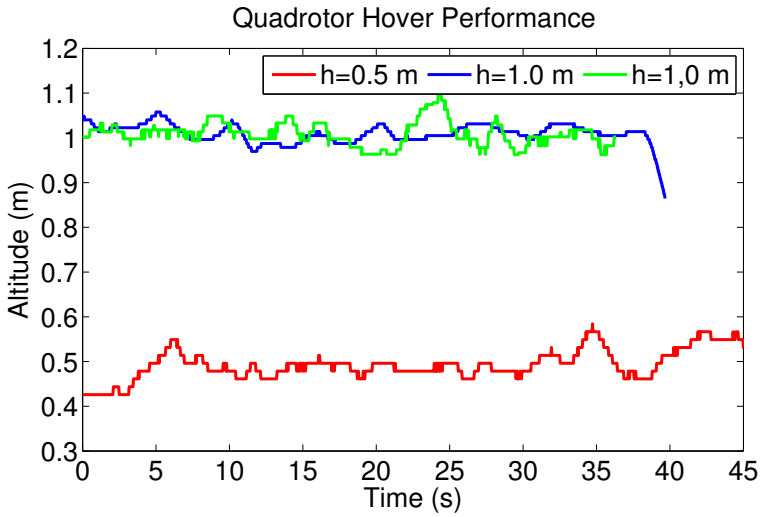


Figure 6.20: Hovering performance for autonomous flights at $h = 0.5\,\text{m}$ and $h = 1.0\,\text{m}$. Values plotted are the estimated altitude from the RHT. The stepped appearance of the plot is due to the quantization of the RHT.

combined to control the heading of a quadrotor helicopter. One process balanced the optical flow from a forward facing camera about the FOE using a general divergence template. The second estimates depth to obstacles in the environment and searches for irregular regions of depth. The way in which the processes are combined is reminiscent of the concurrent visual processes in operation in insects. The control system is able to successfully navigate an indoor corridor in simulated and real data.

Section 6.2 presented a computationally efficient and real-time implementation of plane detection. This utilised the representation of, and the search for, planes in the Hough space. A biologically inspired sampling strategy was shown to improve the performance of the algorithm and reduce computation time. The altitude controller constructed from these principles was able to control quadrotor altitude by detecting the ground plan in images obtained from a Microsoft Kinect in real-time.

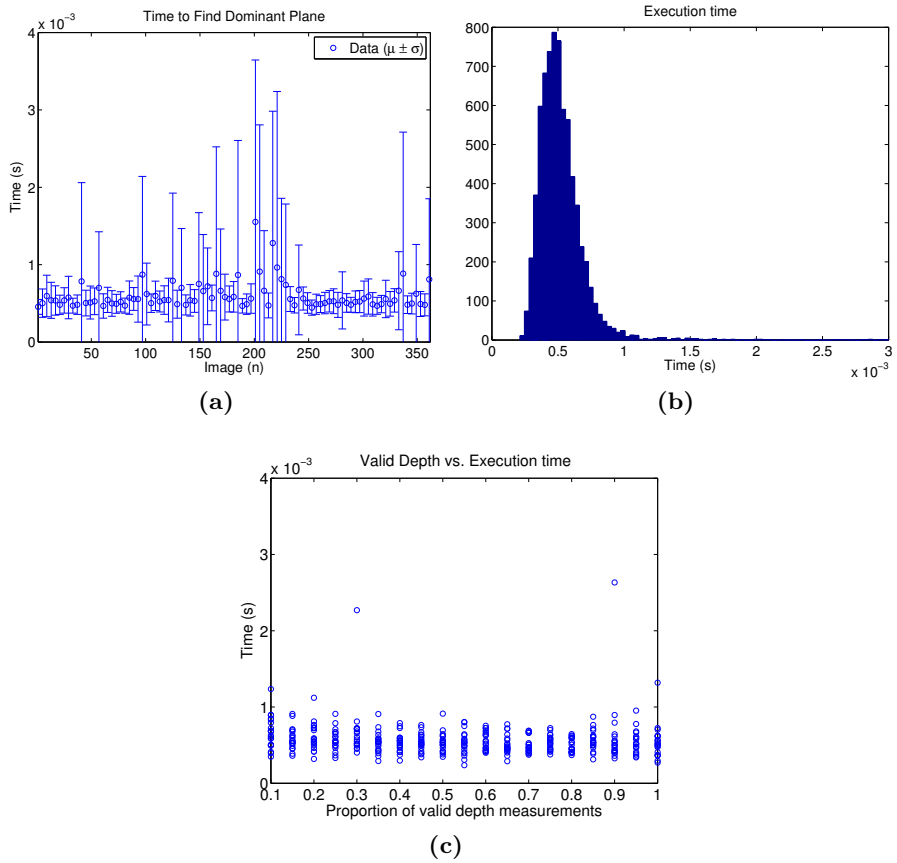


Figure 6.21: Execution times for finding the ground plane in depth images. (a) Mean and standard deviation to find ground plane in test sequence of images. (b) Histogram of the same result. (c) The range of execution time for detecting the ground in the presence of increasing numbers of invalid image points.

Chapter 7

CONTROL USING EDGES AND LINES

This chapter describes a biologically inspired control and avoidance strategy which analyses horizontal edges in an image and is able to successfully control the flight of a quadrotor helicopter in simulated and real environments.

I explained in Section 1.2.1 that conventional biologically inspired UAV flight control uses regulation of optical flow fields. Such designs were motivated by research undertaken on insects that showed they too predominately use optical flow to control their flight (Section 2.1.4). However, recent research [Straw et al., 2010, Maimon et al., 2008] has refined the hypothesis that other visual features are also considered for flight control. More specifically, edges and lines in the visual field have been shown to assist in controlling flight behaviour; both altitude and heading.

This chapter proceeds as follows. Section 7.1 describes image processing techniques for detecting edges in images. Section 7.2 develops a statistical model that describes the distribution of edges in a planar scene, as might be seen by an insect at hover. Section 7.3 describes the biological basis for the altitude control strategy in greater detail. Section 7.4 introduces a number of altitude control strategies based on the distribution of edges and explains how they were developed through simulated flight experiments. The chapter concludes with Section 7.5, the results of flight control experiments and a discussion of how to apply edge based altitude controllers in future.

7.1 EDGE DETECTION

Edges are often associated with the boundaries of objects in a scene. In an image, an edge is a curve that follows a path of rapid change in image intensity (brightness) and it can be shown [Barrow and Tenenbaum, 1981] that under general assumptions about image formation, discontinuities in image brightness are likely to correspond to:

- discontinuities in depth,
- discontinuities in surface orientation,
- changes in material properties,
- variations in scene illumination.

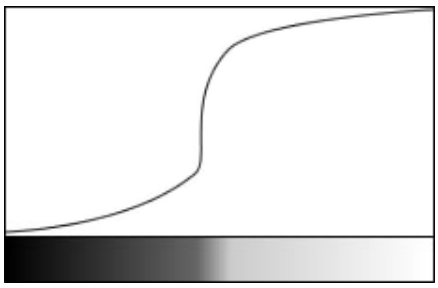


Figure 7.1: The intensity profile of an example edge. The bottom section is the input image, the top shows the image intensity for each pixel.

Consider image intensity about an edge (Figure 7.1). The intensity value ranges from low (black) to high (white). The intensity changes rapidly toward the middle of the figure, shown by a vertical intensity gradient. Now consider the derivatives of image intensity shown in Figure 7.2,

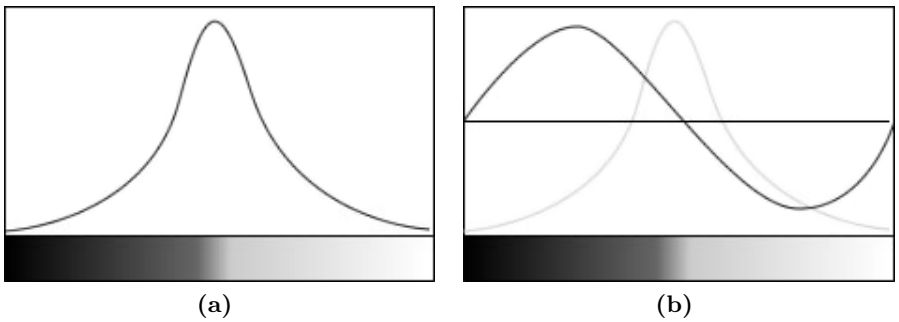


Figure 7.2: Image intensity derivatives of Figure 7.1. (a) The first derivative of image intensity. (b) The second derivative of image intensity.

It is easier to detect an edge (an abrupt change in intensity) by analysing the intensity derivatives [Trucco and Verri, 1998, sec. 4.2]. Those that use the first derivative (Figure 7.2a) are called gradient based edge detectors, those that use second derivative (Figure 7.2b) are called Laplacian based edge detectors.

Gradient based edge detectors look for places where the first derivative of the intensity has a larger magnitude than some threshold. Laplacian based techniques do not require an explicit threshold; by using the second derivative they can detect edges as the zero crossing represents a change in slope of the first derivative. However, due to the additional derivative step required by Laplacian based detectors, they are more susceptible to noise.

7.1.1 Image Convolution and Gradient Detection

The common step in both gradient and Laplacian based edge finding techniques is the calculation of image (brightness) derivatives using differential[1] operators. Previously I assumed the images were continuous, however, this is not the case as digital images are sampled and discrete. Subsequently I approximate differentiation through first order differences. This can be achieved using a convolution filter.

Applying the convolution operation to an image results in a modified version of the input. The parameter that controls this operation is called the convolution kernel. Through the design of an appropriate kernel specific regions of an image can be amplified or attenuated. For example, the simple kernel

$$\mathrm{L} = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}, \tag{7.1}$$

has an averaging effect, giving the image a slight blur, as show in Figure 7.3. Notice how the sum of the 9-element kernel is 1.0 — when designing a convolution kernel the sum of the elements of L should be 1.0. If the kernel sum is not 1.0, the resulting image will be brighter or darker [Trucco and Verri, 1998, p. 56].

(a) **(b)**

Figure 7.3: Demonstrating the effect of convolving an image with a blurring kernel. (a) The original image. (b) The modified image.

There are many popular kernels used for edge detection [Ziou and Tabbone, 1998]; gradient based convolution kernels include the Sobel and Prewitt operators, and Laplacian based operators use kernels, known as Laplacian kernels, of various sizes.

The Sobel edge detector calculates the first derivatives of image intensity separately

[1] The terms 'gradient operator' and 'differential operator' are often used interchangeably with the former specifically in reference to the vector calculus field, Gradient, of the same name.

for the X and Y axes, using the following two kernels,

$$\mathrm{L}_x = \begin{bmatrix} -1 & 0 & -1 \\ -2 & 0 & -2 \\ -1 & 0 & -1 \end{bmatrix}, \tag{7.2}$$

$$\mathrm{L}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}. \tag{7.3}$$

The gradient, G, in each direction is calculated using

$$\mathrm{G}_x = \mathrm{L}_x \odot \mathrm{I} \tag{7.4}$$

$$\mathrm{G}_y = \mathrm{L}_y \odot \mathrm{I}, \tag{7.5}$$

where I is the input image and $\odot$ denotes convolution. The gradient magnitude M is thus

$$\mathrm{M} = \sqrt{\mathrm{G}_x^2 + \mathrm{G}_y^2}, \tag{7.6}$$

and the direction of the gradient at each point is

$$\phi_{\mathrm{G}} = \tan^{-1} \frac{\mathrm{G}_y}{\mathrm{G}_x}. \tag{7.7}$$

The Sobel kernels have a smoothing effect, so they are less sensitive to noise than some techniques and avoid needing to introduce an additional Gaussian blurring step.

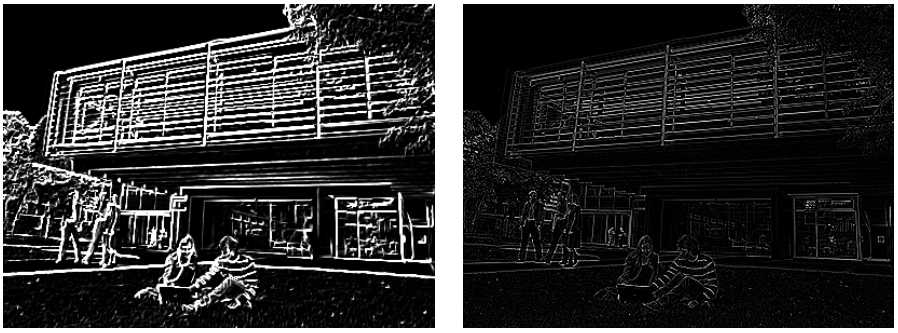
Unlike the Sobel edge detector, the Laplacian edge detector uses only one kernel[2]. It calculates second order derivatives in a single pass,

$$\mathrm{M} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \odot \mathrm{I}. \tag{7.8}$$

Figure 7.4 shows the magnitude of the detected edges, M, given when convolving a sample image (Figure 7.4a) with Sobel and Laplacian operators. One can now take the M image highlighting edges and apply a threshold, k_s, to reduce it to a binary image, B,

$$\mathrm{B} = \begin{cases} 1 & \text{if } \mathrm{M}[m,n] \geq k_s \\ 0 & \text{otherwise} \end{cases}. \tag{7.9}$$

[2] While the kernel used here is designed to detect horizontal and vertical lines only, there are other Laplacian kernels that also detect diagonal lines. Some of these can be seen in Table 7.1.



(a) **(b)**

Figure 7.4: Demonstrating the effect of the Sobel and Laplacian kernels for detecting horizontal and vertical edges. (a) The result of the Sobel operator. (b) The result of the Laplacian operator.

There are several recommended approaches to choosing an appropriate threshold, k_s, depending on the kernel used. For example, the recommended approach for thresholding a Sobel-generated gradient image is to set k_s to the root mean square (RMS) value of the gradient image [Pratt, 2007, ch. 15].

The most popular and widespread edge-detection technique is the Canny method. The Canny method differs in that it uses two different thresholds (to detect strong and weak edges), including the weak edges in the output only if they are connected to strong edges. Therefore, this method is less likely than the others to be fooled by noise and more likely to detect true weak edges, albeit at the cost of higher computational complexity [Canny, 1986].

Other kernels to detect lines at different orientations are possible and are shown in Table 7.1.

7.2 A STATISTICAL MODEL OF IMAGE EDGES

In this section, a statistical model in the edges in an image detected by a camera looking across a planar scene is developed.

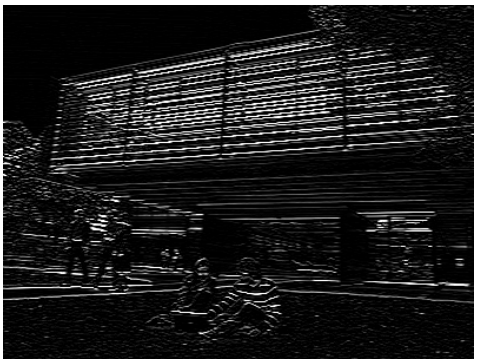
The model was inspired by my observations from images captured onboard the 'wasp' quadrotor. That configuration is represented by the geometry visible in Figure 7.5. The observations were as follows;

1. Because of limited resolution on the camera, edges close together will not be distinguished.

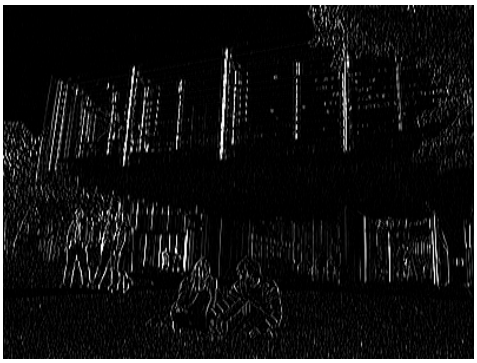
2. Edges are failed to be detected independently of other edges; they are missed because they are in close proximity.

These points are synonymous with the important properties of Poisson processes (Section 7.2.1 explains this in more detail). Using this insight, a model for estimating

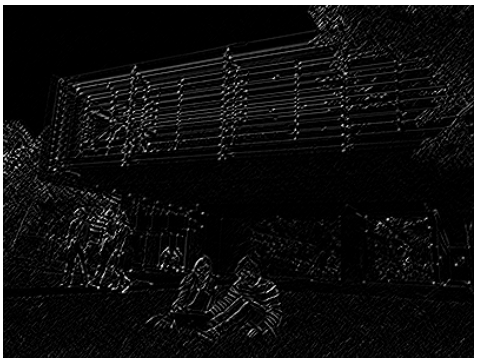
$$\mathrm{L} = \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}$$



$$\mathrm{L} = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}$$



$$\mathrm{L} = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}$$



$$\mathrm{L} = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$

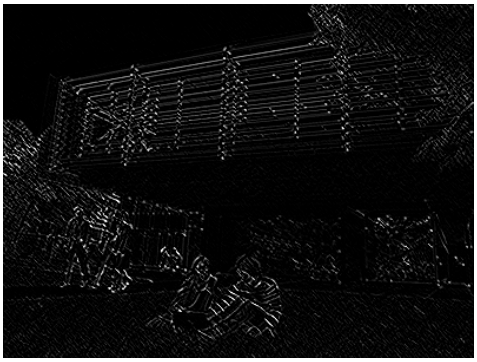


Table 7.1: Example convolution kernels and the resulting image. The kernels were selected to detect lines at different orientations; horizontal, vertical, 45°, and 135° respectively.

altitude using edges detected in an image of the scene was created. The model imposes the following formal assumptions;

1. Edges are assumed to be distributed along the plane as a Poisson process with intensity λ (this is the mean number of edges per metre).

2. The scene is planar or at least has small height variations compared to the camera height, h, above the scene.

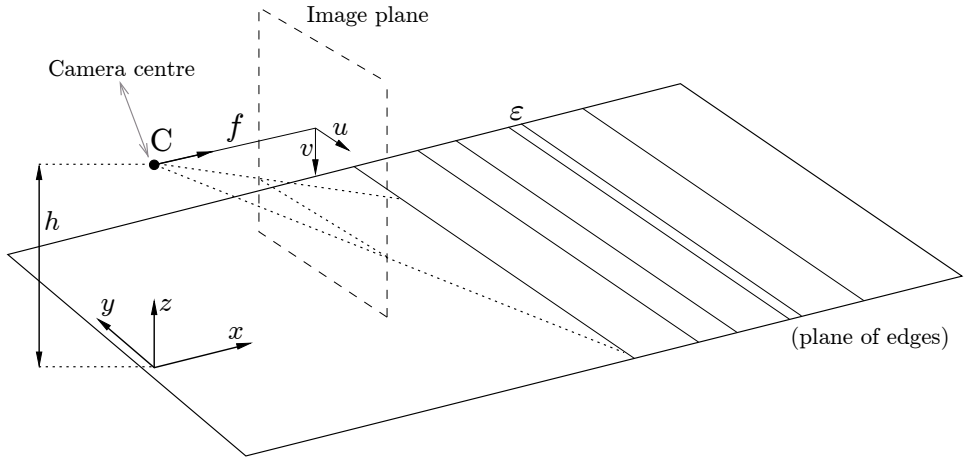


Figure 7.5: The geometry of the problem. A camera, C, looks out over a plane containing many edges, ε. The camera has a focal length, f, and is at an altitude, h, above the plane. The camera image plane coordinates are measured in pixels, (u, v).

7.2.1 Poisson Processes

The Poisson distribution [Ross, 1983] and the underlying Poisson process was discovered by Siméon-Denis Poisson in 1837. The author's original work described the properties of random variables counting number of discrete occurrences of an event, k, over a discrete time interval. Applications of the Poisson distribution include modelling: birth defects and genetic mutations; car accidents, traffic flow and gap distance; reliability engineering, machine and component failures; photons arriving at a telescope; and the nuclear decay of atoms.

Consider the last example as illustrative, although by definition the others obey the same underlying process; once a particle decays, it does not decay again.

More rigorously, the basic-form Poisson process is a continuous-time counting process, $N(t), t \geq 0$, with the following properties:

1. The number of successes in two disjoint time intervals is independent.

2. The probability of a success during a small interval is proportional to the entire length of the interval.

Compare these properties with the detection of edges in an image of a planar scene, explained in the previous section. The consequence of these properties yield:

1. The probability distribution of the number of events in an interval is a Poisson distribution.

2. The probability distribution of the waiting time until the next occurrence is an exponential distribution.

3. The occurrences are distributed uniformly on any interval.

The Poisson probability distribution function is defined as

$$P(k) = \frac{\lambda^k e^{-\lambda}}{k!}, \tag{7.10}$$

where $k \in 0, 1, 2, 3, ...$ and represents the number of occurrences of an event (known as the 'rate parameter'). λ is a positive real number equal to the expected number of occurrences during the given interval.

The two most common variations of Poisson processes are the homogeneous and non-homogeneous forms. Both relate to different properties of λ. A homogeneous Poisson process $N(t)$, yields a Poisson distribution $P(k)$ with associated parameter $\lambda\tau$ such that the number of events in the interval $(t, t + \tau]$ follows

$$P[(N(t + \tau) - N(t)) = k] = \frac{e^{-\lambda\tau}(\lambda\tau)^k}{k!}, \tag{7.11}$$

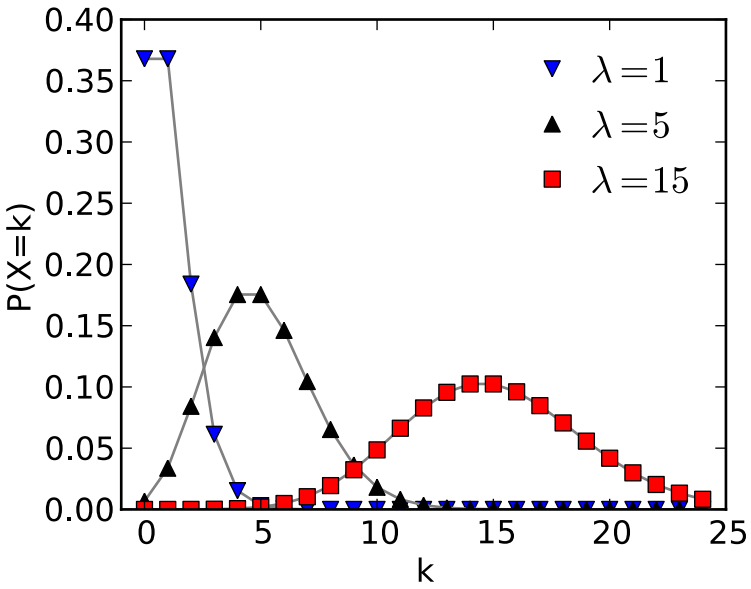
where $N(t + \tau) - N(t) = k$ is the number of events in time interval $(t, t + \tau]$. A non-homogeneous Poisson process accounts for the rate parameter changing. In this case the rate parameter is replaced with a generalized rate function, $\lambda(t)$. The expected number of events is given by

$$\lambda_{a,b} = \int_a^b \lambda(t)\,\mathrm{d}t. \tag{7.12}$$

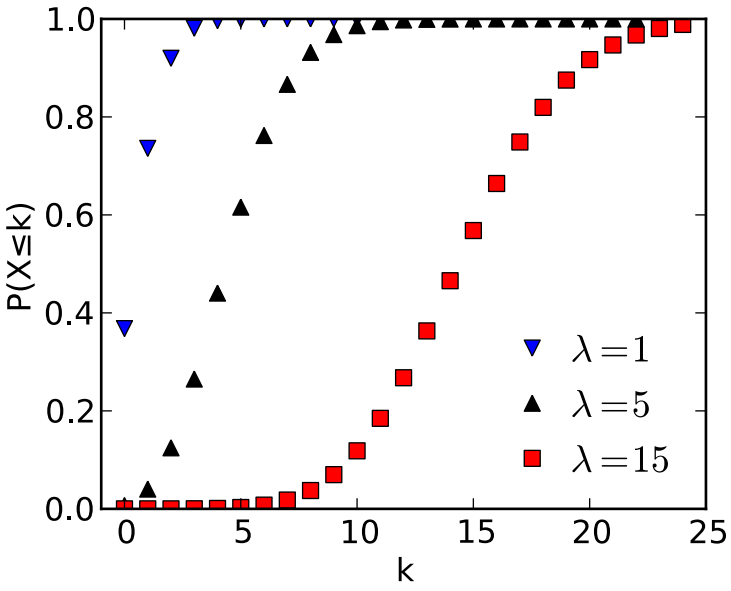
Thus the number of events in the interval $(a, b]$ follows a Poisson distribution with associated parameter $\lambda_{a,b}$

$$P[(N(b) - N(a)) = k] = \frac{e^{-\lambda_{a,b}}(\lambda_{a,b})^k}{k!}, \tag{7.13}$$

where $N(b) - N(a)$ is the interval $b - a$. The probability distribution function (PDF) and cumulative distribution function (CDF) of a Poisson process is shown in Figure 7.6 for different values of λ.



(a)



(b)

Figure 7.6: The characteristics of a Poisson process. (a) The probability distribution function (PDF). The horizontal axis is k, the number of occurrences of an event. The function is only defined at integer values of k. (b) The cumulative distribution function (CDF). The horizontal axis is k, the number of occurrences of an event. The CDF is discontinuous at the integers of k and flat everywhere else (a variable that is Poisson distributed only takes on integer values).

7.2.2 System Model

Using a pin-hole camera model [Hartley and Zisserman, 2004], a scene coordinate (x, y) is mapped to image coordinates (u, v) using

$$u = \frac{fy}{x}, \tag{7.14}$$

$$v = \frac{fh}{x} = \frac{k}{x}, \tag{7.15}$$

where f is the focal length of the camera and h is the height of the camera above the scene. These parameters can be combined into a single scale factor, k. If the maximum image vertical coordinate is $v_{\max}$, the closest visible scene position is

$$x_{\min} = \frac{k}{v_{\max}}. \tag{7.16}$$

If the camera has a vertical *resolution*[3] δ, then multiple edges in the region $x_{\min} \leq x < x_1$ are not resolvable, where

$$\begin{aligned} x_1 &= \frac{k}{v_{\max} - \delta} \\ &= \frac{k}{\frac{k}{x_{\min}} - \delta}. \end{aligned} \tag{7.17}$$

I assume that the first such edge, the edge closest to $x_{\min}$, will be detected but subsequent edges in the region $x_{\min} \leq x < x_1$ are not detected.

A simple non-homogeneous Poisson process is used to model the detected edges where the intensity $\lambda(x)$ varies spatially. The probability of detecting an edge, ε, in the region $(x, x + \Delta x)$, where $x_{\min} \leq x < x_1$, is the probability that there is an edge inside $(x, x + \Delta x)$ and no other detected edge in the preceding *resolution* interval. Mathematically, this can be expressed as

$$\begin{aligned} &P\left(\varepsilon_D \in (x, x + \Delta x)\right) \\ &\quad = P\left(\varepsilon \in (x, x + \Delta x)\right) \times P\left(\varepsilon \notin (x_{\min}, x)\right) \\ &\quad \approx (\lambda \Delta x) \times P\left(\varepsilon \notin (x_{\min}, x)\right) \\ &\quad \approx \lambda \Delta x \exp\left(-\lambda (x - x_{\min})\right). \end{aligned} \tag{7.18}$$

Here, ε denotes an actual edge and ε_D denotes a detected edge. Hence, by definition, the intensity of the non-homogeneous Poisson process at x is

$$\lambda(x) = \lambda \exp\left(-\lambda (x - x_{\min})\right). \tag{7.19}$$

[3] Here I use the formal meaning of the word resolution; the smallest detectable change in the quantity being measured, not the variation in general use; the number of pixels in an image.

Now consider a more distant edge at $x > x_1$. The probability of detecting this edge in the image is

$$\begin{aligned} &P\left(\varepsilon_D \in (x, x+\Delta x)\right) \\ &\quad = P\left(\varepsilon \in (x, x+\Delta x)\right) \times P\left(\varepsilon \notin (x_2, x)\right) \\ &\quad = \lambda \Delta x \exp\left(-\lambda(x - x_2)\right), \end{aligned} \tag{7.20}$$

where

$$\begin{aligned} x_2 &= \frac{k}{v+\delta} \\ &= \frac{k}{\frac{k}{x}+\delta}. \end{aligned} \tag{7.21}$$

So, the equivalent intensity for the non-homogeneous Poisson process is

$$\lambda(x) = \lambda \exp\left(-\lambda\left(x - \frac{k}{\frac{k}{x}+\delta}\right)\right). \tag{7.22}$$

Let the random variable, X, denote the position of a detected edge in a scene, then the empirical CDF of the edge position is given by

$$F_X(x) = P(X \leq x) \tag{7.23}$$

$$= \lim_{N\to\infty}\left\{\frac{\sum_{i=1}^{N} C_i(x)}{\sum_{i=1}^{N} C_i(\infty)}\right\} \tag{7.24}$$

$$= \frac{E\left\{C_i(x)\right\}}{E\left\{C_i(\infty)\right\}}, \tag{7.25}$$

where $C_i(x)$ denotes the cumulative number of detected edges for the i$^{\text{th}}$ scene. Hence, the CDF can be expressed as [Ross, 1983]

$$F_X(x) = \frac{m(x)}{m(\infty)}, \tag{7.26}$$

where

$$m(x) = \int_{x_{\min}}^{x} \lambda(u)\,\mathrm{d}u, \tag{7.27}$$

represents the mean number of detected edges in the region $x_{\min} \leq x$. Finally, differ-

entiating the CDF yields the PDF as

$$f_X(x) = \frac{m'(x)}{m(\infty)} = \frac{\lambda(x)}{m(\infty)}$$
$$= \begin{cases} \frac{\lambda}{m(\infty)} \exp\left(-\lambda(x - x_{\min})\right) & x_{\min} \leq x \leq \frac{k}{\frac{k}{x_{\min}} - \delta} \\ \frac{\lambda}{m(\infty)} \exp\left(-\lambda(x - \frac{k}{\frac{k}{x} + \delta})\right) & x > \frac{k}{\frac{k}{x_{\min}} - \delta} \end{cases}. \tag{7.28}$$

From (7.15), the random scene variable, X, maps to the random image variable V using

$$V = \frac{k}{X}. \tag{7.29}$$

Hence, the probability distribution function for an image edge, $f_V(v)$, is related to the probability distribution function for a scene edge, $f_X(x)$, by

$$f_V(v) = f_X\left(\frac{k}{v}\right)\left|\frac{-k}{v^2}\right|, \tag{7.30}$$

and thus,

$$f_V(v)$$
$$= \begin{cases} \frac{\lambda k}{m(\infty)v^2} \exp\left(-\lambda\left(\frac{k}{v} - x_{\min}\right)\right) & x_{\min} \leq \frac{k}{v} \leq \frac{k}{\frac{k}{x_{\min}} - \delta} \\ \frac{\lambda k}{m(\infty)v^2} \exp\left(-\lambda\left(\frac{k}{v} - \frac{k}{v+\delta}\right)\right) & \frac{k}{v} > \frac{k}{\frac{k}{x_{\min}} - \delta} \end{cases}$$
$$= \begin{cases} \frac{\lambda k}{m(\infty)v^2} \exp\left(-\lambda k\left(\frac{1}{v} - \frac{1}{v_{\max}}\right)\right) & v_{\max} - \delta \leq v \leq v_{\max} \\ \frac{\lambda k}{m(\infty)v^2} \exp\left(-\lambda k\left(\frac{1}{v} - \frac{1}{v+\delta}\right)\right) & v < v_{\max} - \delta \end{cases}. \tag{7.31}$$

7.2.3 Validating the Model

The model was verified in a virtual environment (Figure 7.7a), created using the VRML modelling language (Appendix C.2). The experiments involved moving the virtual camera (configured with a known focal length f) in the vertical axis and recording the image, $\mathbf{I}$, at each instant.

To test the model prediction I use the cumulative sum of image edges. I first calculate the vertical image gradient by convolving the image with the Sobel kernel,

$$\mathrm{G}_v = \mathbf{I} \odot \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix}. \tag{7.32}$$

Here, $\mathbf{I} = \mathbf{I}[m, n]$ is the input image, $\mathrm{G}_v = \mathrm{G}_v[m, n]$ is the vertical gradient image (in the v-axis), and $\odot$ denotes convolution.

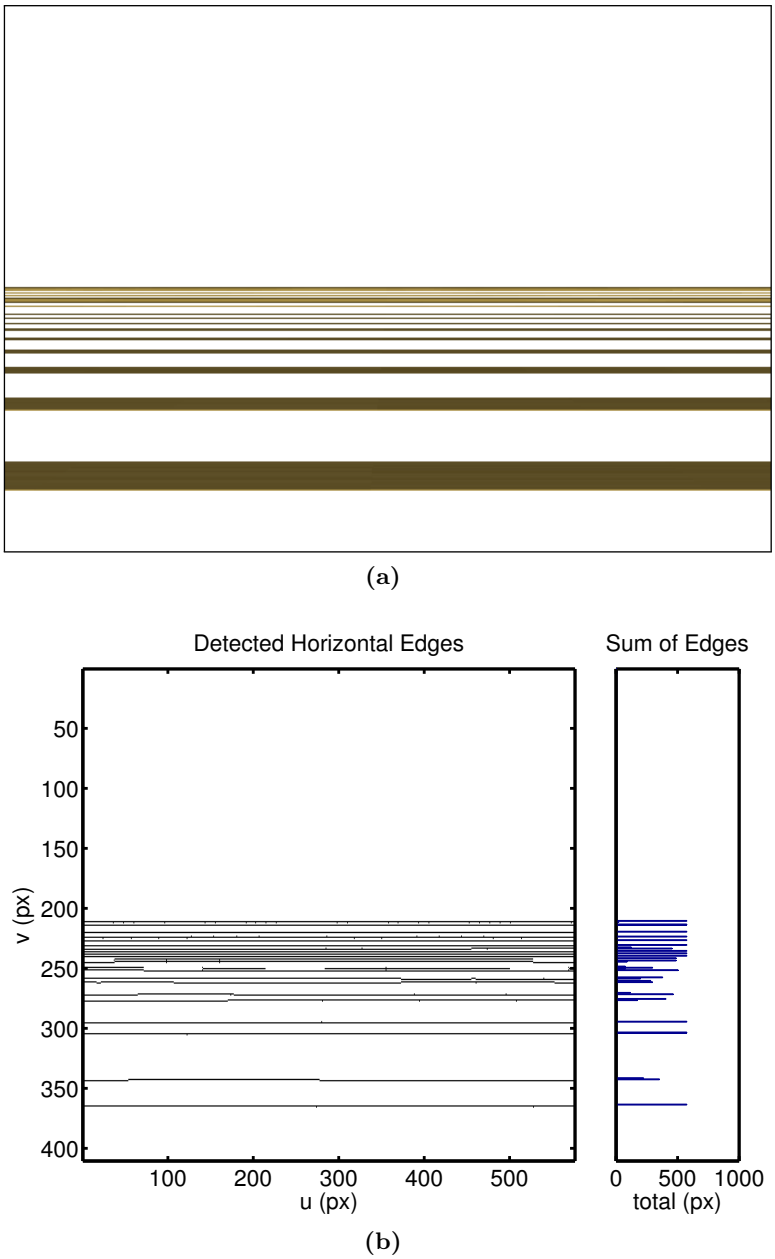


Figure 7.7: The virtual simulation environment used for validation of the probabilistic model of edges. (a) An image from the virtual scene (note, this has be recoloured, replacing the black background with white for readability). (b) The result of the edge detection algorithm (left), $\mathbf{B}_v$, and the row-wise (v) sum of edges (right), $\mathbf{E}_v$.

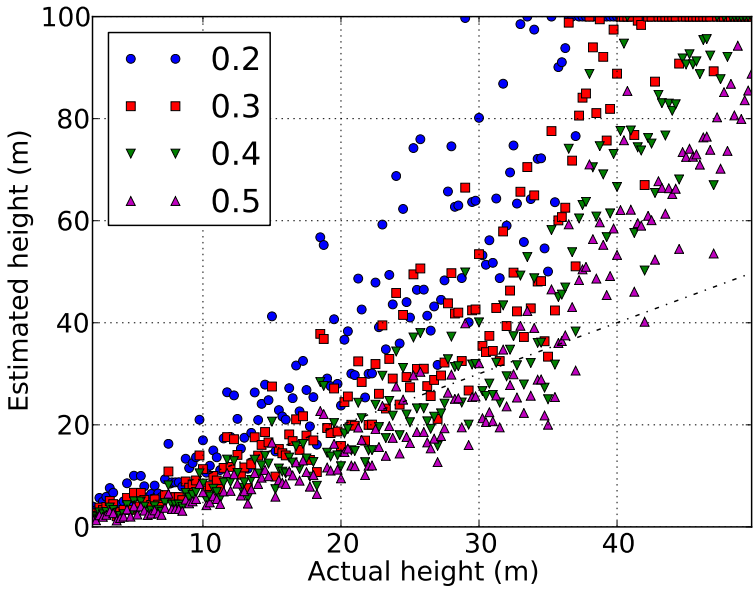


Figure 7.8: A comparison of estimated camera height versus actual camera height as a function of λ with a camera field of view 60°.

I extract a binary image from the gradient image by thresholding,

$$\mathbf{B}_v[m,n] = \begin{cases} 1 & \text{if } \mathbf{G}_v[m,n] \geq k_s \\ 0 & \text{otherwise} \end{cases}. \tag{7.33}$$

Here k_s is the RMS value of the gradient image, the recommended approach as described in Section 7.1. This results in an image with only horizontal images remaining (Figure 7.7b).

The quantized gradient is then summed in the horizontal direction,

$$\mathbf{E}_v[m] = \sum_{n=1}^{N} \mathbf{B}_v[m,n]. \tag{7.34}$$

This represents the number of horizontal edges in the image (Figure 7.7b). Finally, a running sum of $\mathbf{E}_v$ is performed to estimate the empirical cumulative distribution of horizontal edges,

$$\mathbf{C}_v[n] = \frac{\sum_{j=1}^{m} \mathbf{E}_v[j]}{\sum_{j=1}^{M} \mathbf{E}_v[j]}. \tag{7.35}$$

Equation (7.31) showed that the height estimate depends on two model parameters, λ, describing the number of edges in the environment, and k, which includes the camera focal length, f. The following experiments explore the accuracy of the model as those values change[4].

[4] More precisely, I change the virtual camera FOV in the VRML using (C.1) and as explained in (2.3), FOV is very closely related to focal length.

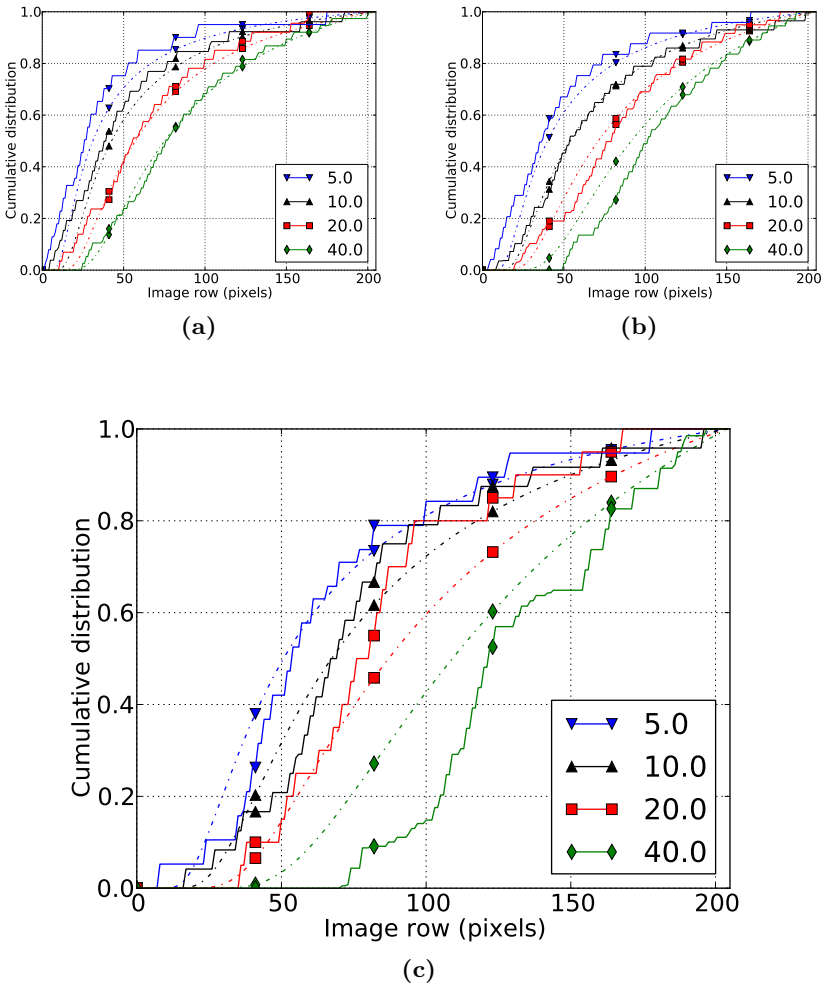


Figure 7.9: A comparison of estimated cumulative probability distribution with the model (dashed) as a function of height h for $\lambda = 0.35$: (a) camera field of view 120°, (b) camera field of view 90°, (c) camera field of view 60°. The performance of the model improves when using a camera with a larger field of view.

Figure 7.8 shows the effect of the parameter λ on the height estimate when matching the empirical data to the statistical model. In this experiment, images from the virtual world were collected using a camera field of view of 60°. In this configuration, $\lambda = 0.35$ was found to best estimate the height in the simulated environment.

Using this value, further simulations were performed using virtual cameras with different fields of view, 60°, 90°, 120°. Figure 7.9 shows the model-predicted CDF, from (7.31), plotted against the real cumulative sum of edges in the image, from (7.35). The model fits reasonably well with a larger field of view but gets poorer with a decreasing field of view.

Figure 7.10 extends this work, estimating height by fitting the model CDF to the empirical CDF. Notice that while the larger field of view plots under-estimate height,

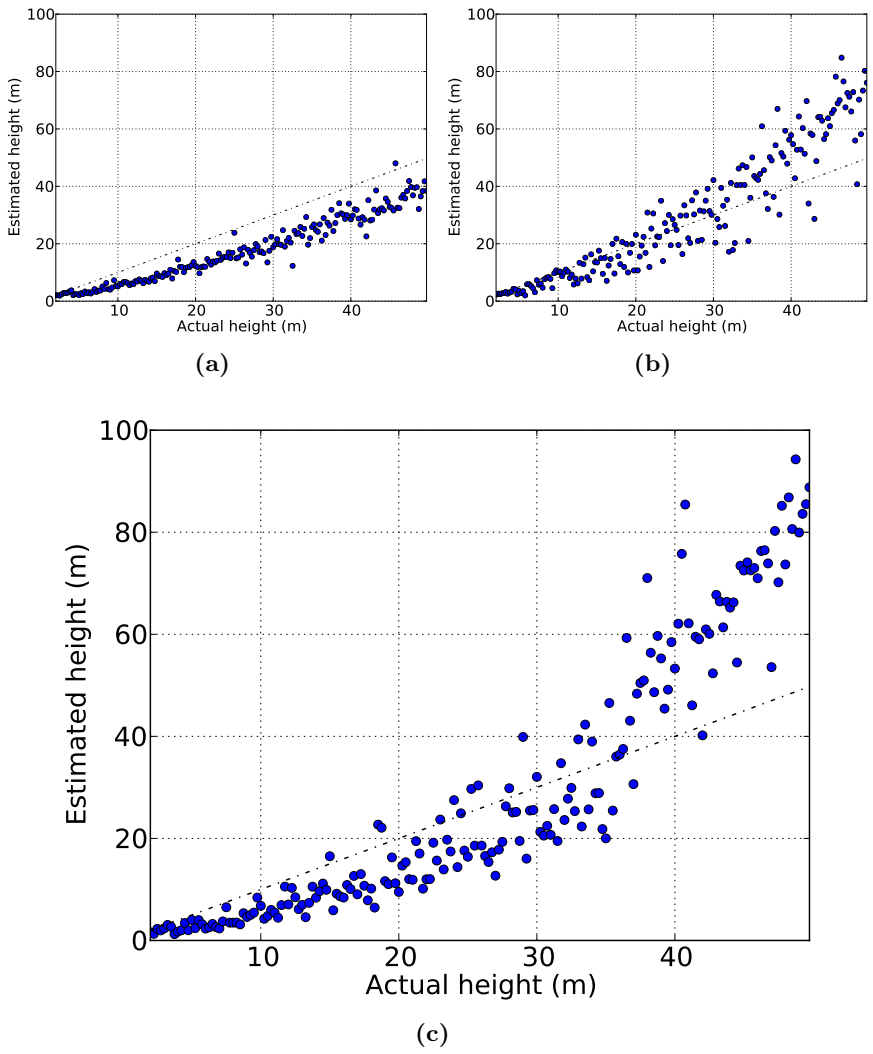
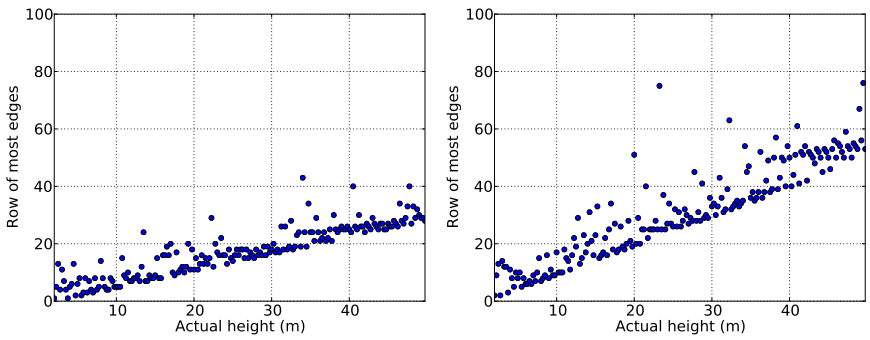


Figure 7.10: A comparison of estimated camera height versus actual camera height with $\lambda = 0.35$. (a) camera field of view 120°, (b) camera field of view 90°, (c) camera field of view 60°. The variance of the height estimate decreases when using a camera with a larger field of view.

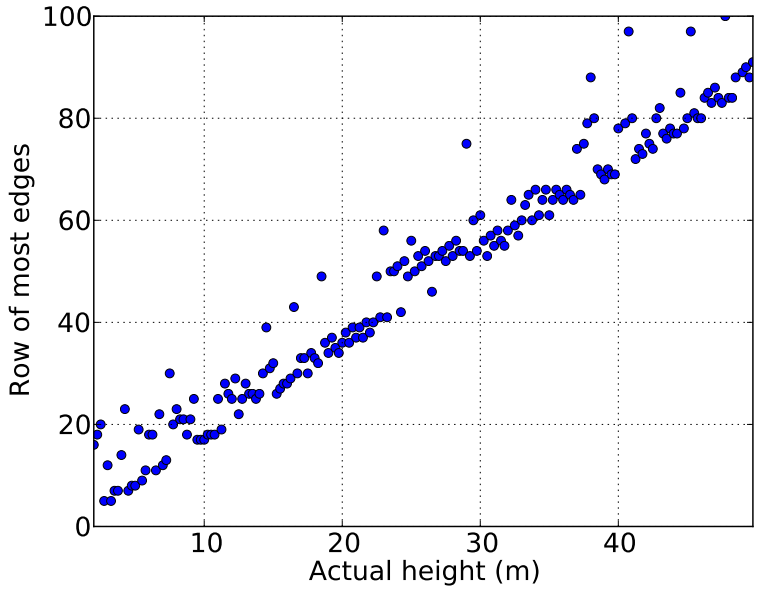
the variance of their estimate is lower.

An interesting observation made during simulation is that the peak of the empirical histogram (for example seen on the right hand panel of Figure 7.13a) appears to be a linear function of camera height. Figure 7.11 tests this idea, showing that altitude can be estimated by by detecting this peak alone. There is a small bias that needs to be subtracted. It is expected that the proportionality factor is a function of λ; this is yet to be confirmed.

Differentiating the modelled probability distribution function against the image coordinate v predicts a square-root dependence of the position of the mode versus



(a) (b)



(c)

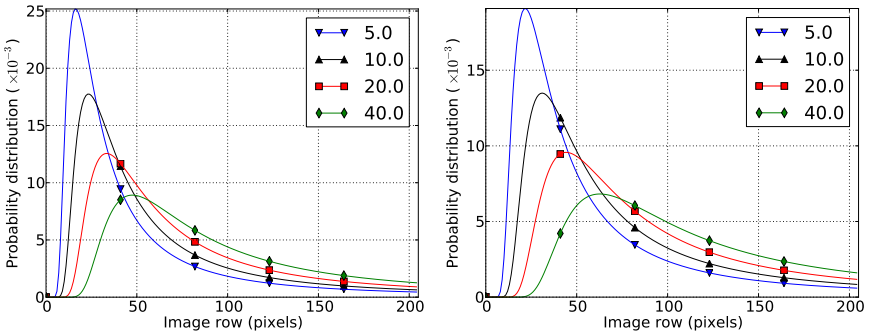
Figure 7.11: A comparison of the image row with the most edges versus the camera height with $\lambda = 0.35$. (a) camera field of view 120 degrees, (b) camera field of view 90 degrees, (c) camera field of view 60 degrees.

camera height, (visible in Figure 7.12),

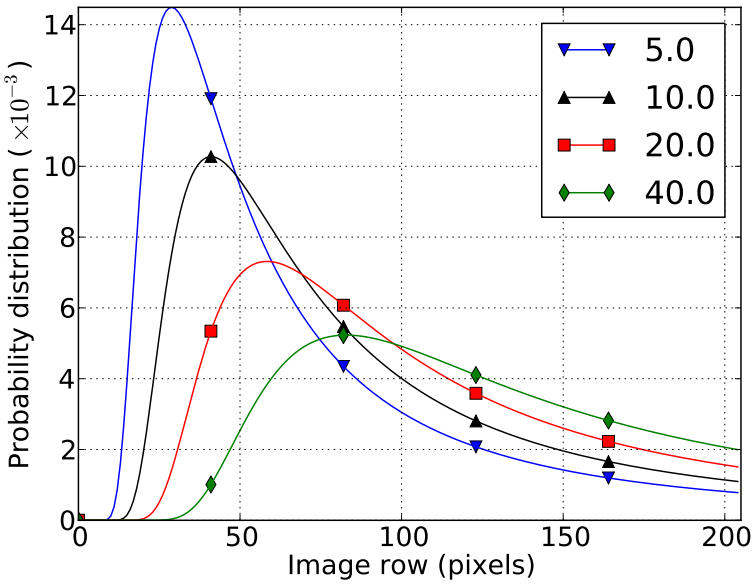
$$v_{\text{mode}} \approx \sqrt{k\lambda\delta}. \tag{7.36}$$

Thus in summary, while the statistical model is a only a simple continuous first approximation, it manages to capture the behaviour of the system under certain conditions. The model also features a biologically promising characteristic — increasing the field of view of the camera improves the accuracy and decreases the variance of the model estimate and as described in Section 2.1.2.1, insects have large FOV eyes.

The following sections apply the model to real data.



(a) (b)



(c)

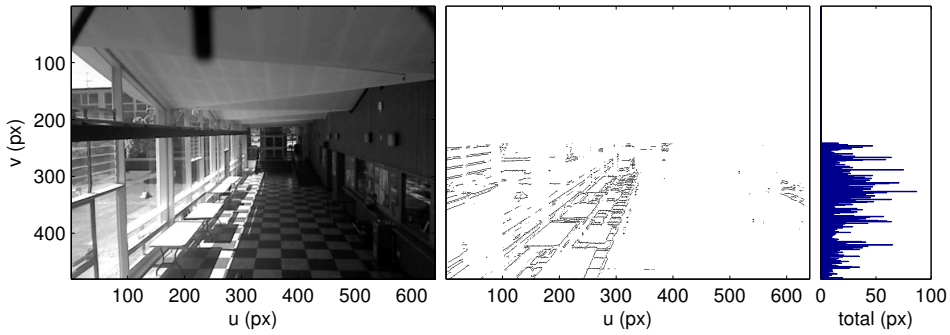
Figure 7.12: The modelled probability distribution functions as a function of camera height h for $\lambda = 0.35$: (a) camera field of view 120°, (b) camera field of view 90°, (c) camera field of view 60°.

7.2.4 Altitude Estimation in Real Scenes

To validate the model of the previous section, a number of flight experiments were undertaken; trials 1 and 2 were undertaken in an indoor environment (Figure 7.13a), and trials 3 and 4 were performed outside (Figure 7.13b).

In all trials, the 'wasp' quadrotor was flown at a range of different altitudes and images were captured. The system was setup in the "Wasp' Kontron Kinect' configuration of Appendix A.4. Equation (7.15) requires the Kinect visible camera to be calibrated as described in Table A.3, with $f = 525\,\text{px}$.

Figure 7.13 shows representative images from two test flights. Because the model describes an infinite plane extending out to the horizon, I assume the horizon is in the centre of the image and disregard information below that point (visible in Figure 7.13a).



(a)

(b)

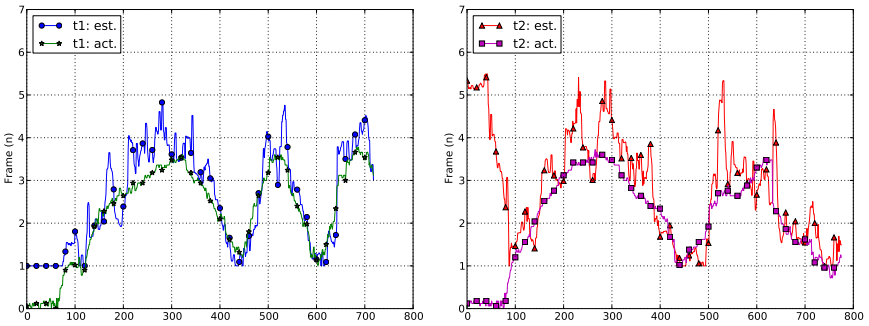
Figure 7.13: Images from flight testing the edge statistics altitude model. (a) An indoor scene. On the left is the captured image, in the middle is the edges detected using the Sobel algorithm, and on the right is the sum of edges along the u-axis. Only the bottom half of the image (below the 'horizon') is considered; see text for explanation. (b) An outdoor scene.

Using (7.31), the altitude is solved numerically[5] by fitting the CDF to the cumulative sum. Solutions for many values of λ were calculated.

Figure 7.14 and Figure 7.15 show the results of experiments in two distinct real environments. In the first environment $\lambda = 3.1$ was found to best describe the data over multiple trials. Similarly in the second experiment, $\lambda = 1.5$ was found to be best.

In both experiments the value of λ derived from the first trial in the environment, could be used to predict the altitude of the quadrotor in subsequent trials in the same environment. This demonstrates that it is an innate property of the environment and even if estimating its value from a single image is a nebulous task, once obtained by any means it can be used to predict altitude.

[5] Implementation in `python-numpy` and `python-scipy` using `sp.optimize.fminbound`.



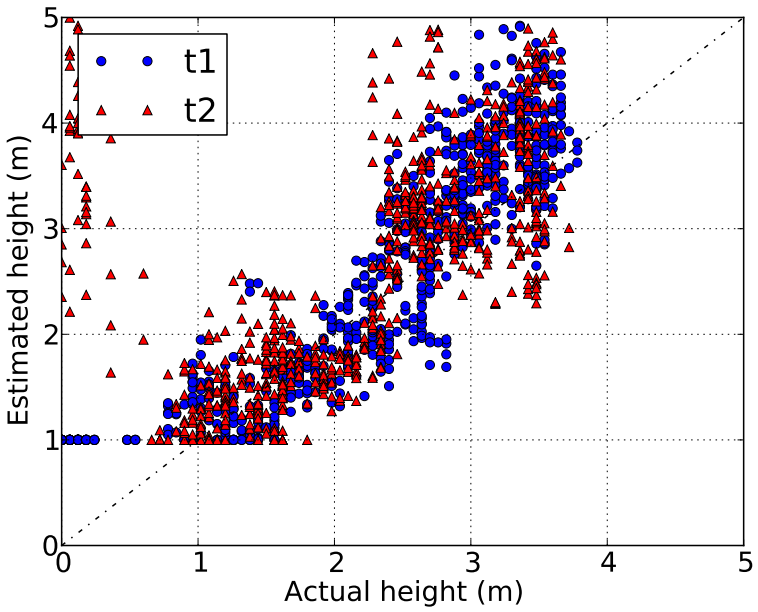


Figure 7.14: Altitude estimation using edge statistics of a real indoor scene; In both trials $\lambda = 3.1$. (a) Trial 1 estimated versus actual altitude. (b) Trial 2 estimated versus actual altitude. (c) A comparison of the estimated and actual altitude.

7.2.5 Conclusion

The results show that the derived model captures many features of the real detection process; in Figure 7.9 the model is able to approximate the distribution of edge information in a simulated planar scene, and in Figure 7.10 the model is able to estimate the height given a distribution of edges.

These two results demonstrate that it is possible to extract sufficient information from edges in an image in order to estimate the camera height.

In the biological context from which this work was inspired, it is interesting to consider what accuracy the algorithm requires in order to be of utility to an organism. An analogous situation is the avoidance response in *Drosophila* (Section 2.1.4.5), despite

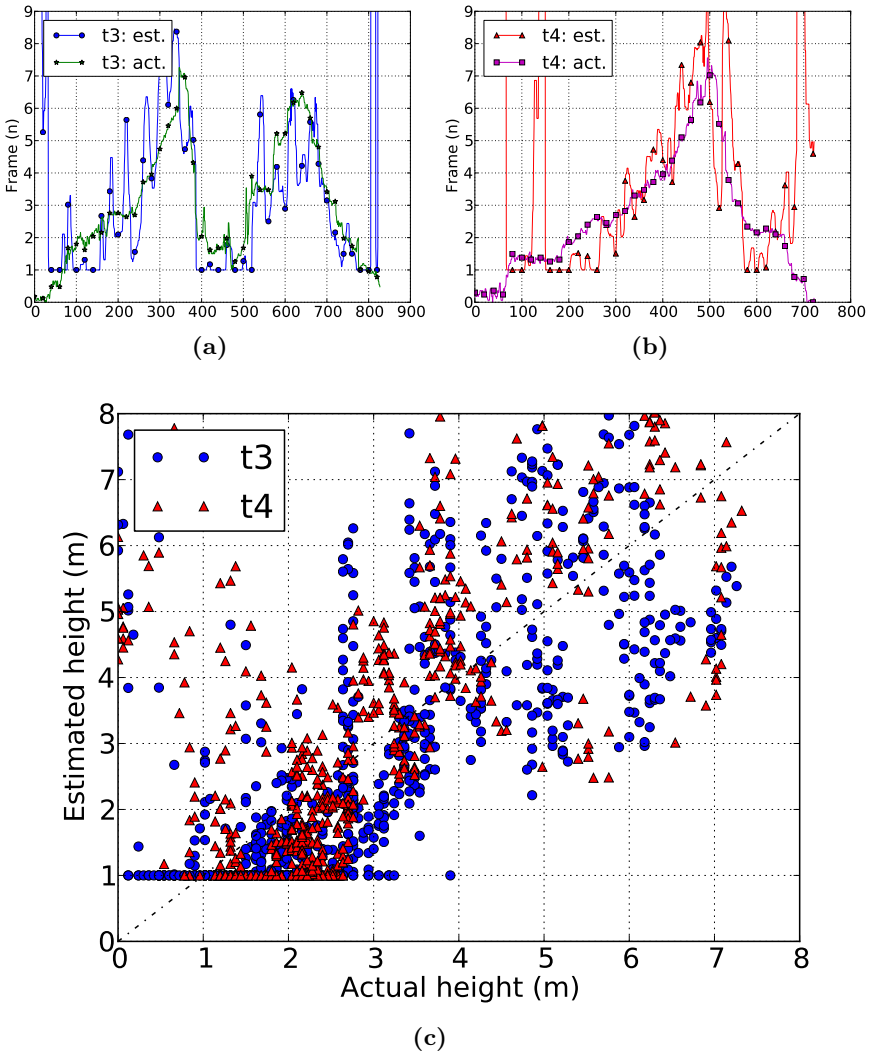


Figure 7.15: Altitude estimation using edge statistics of a real outdoor scene; In both trials $\lambda = 1.5$. (a) Trial 3 estimated versus actual altitude. (b) Trial 4 estimated versus actual altitude. (c) A comparison of the estimated and actual altitude.

the insect continually measuring image motion, the avoidance response seems to be active only when optical flow exceeds a predetermined threshold. Comparatively, the model-predicted height looks suitable for such decisions.

1. Qualitatively Figure 7.9 shows the model can distinguish large from small heights.
2. Quantitatively, Figure 7.10 shows the model can provide a reasonable height estimate at low altitudes.

Additionally, given it plausible that an organism could evolve an estimate for λ for environments relevant to its' survival, it could subsequently use this value and the model outlined, to estimate its altitude. Furthermore, the performance of the model improves when using a wider field of view camera, compatible with insects with wide field of view eyes.

Further work is required to improve the model to predict the observed linear height dependence of the empirical histogram mode. Additional work remains to assess the model performance when the scene is non-planar, in non-ideal lighting conditions, or when λ is unknown. Even in these scenarios, the results of Figure 7.10 are encouraging. If λ can be assumed constant (such as over a small number of frames) the agreement between the model-predicted and actual height at low values is again reminiscent of time-to-contact approaches [Horn et al., 2007] which emulate the biological expansion avoidance response.

7.3 BIOLOGICAL BASIS

Section 2.1.4.2 described the conventional understanding of altitude control; that most insect flight behaviours are in response to changes in image motion, in particular, *ventral* optical flow (Figure 1.6). Neurologically, interpretation of structure from image motion is undertaken by the optic lobe, and the central brain (Section 2.1.3.1). Even given the decoupling self from scene motion, there is still a question of how an altitude control signal is generated.

While investigating altitude control of *Drosophila*, Straw et al. [2010] found an additional mode of visual information processing used to control altitude. The authors confirmed that *Drosophila* respond to wide-field motion with *syndirectional* velocity changes such that vertical, forward, and *lateral* visual motions elicit movement in the same direction (i.e., the optomotor response). The authors also confirmed that *Drosophila* also avoid strong ventral expansion (i.e., the collision avoidance reflex). Most novel however, the authors also found that *Drosophila* adjust their altitude on the basis of nearby visual features — controlling their altitude based on the visual location of the dominant horizontal edge in the image.

For more discussion of the role of edges in visual flight control, see Section 2.1.3.5.

One parallel to this work could be biologically inspired UAV *attitude* control systems which use horizon detection [Ettinger et al., 2002, Thurrowgood et al., 2009, de Croon et al., 2011]; however, biologically speaking, these replicate the *ocelli* and are neither line nor edge detection schemes.

To my knowledge this is the first system that utilises edge and line information as the primary and only means of visual control.

7.3.1 Mapping Edges to a Real Controller

The Straw et al. [2010] result was novel but raises interesting challenges as it does not address practical issues necessary for a biomimetic implementation. This section discusses those issues and presents rationale for the simulation and implementation decisions explained in the forthcoming sections.

Straw et al. [2010] did not investigate the role of depth and whether it affected which edges the flies responded to, nor did they find how the edge based altitude controller is mediated with other control systems. In flight, multiple horizontal edges of various sizes and distance will be visible and the experiments offered no insight into how flies choose among them. The authors did note "close parallels between the reflexes used to control altitude and those used to control horizontal course" and so I might reasonably consider results studying horizontal course for inspiration.

Maimon et al. [2008] looked at horizontal course control and found that flies used edges to classify objects into long and short classes[6], and the attraction or repulsion of these objects was related to their size in the visual field of the insect (the angle the object *subtends*). Conventionally we call an arrangement of edges a line and we may categorize objects into classes using the line length on its border. However, there is no strict need nor biological basis to calculate line geometry; biological studies have only addressed the responses to objects irrespective of their length[7]. This is fortunate; I prefer control systems that could feasibly be represented in the brains of insects and the detection of lines from edges is computationally expensive[8].

From these two results and a desire to create a computationally efficient controller, I approximate horizontal or vertical lines as the sum of edges along the respective image axis — the row with the largest sum of edge pixels is likely to contain the longest line along this axis. This approximation is more correct the larger or closer the object is, as then lines on it will *subtend* more of our visual field. I validate this approximation against the Straw et al. [2010] result in Section 7.4.1.

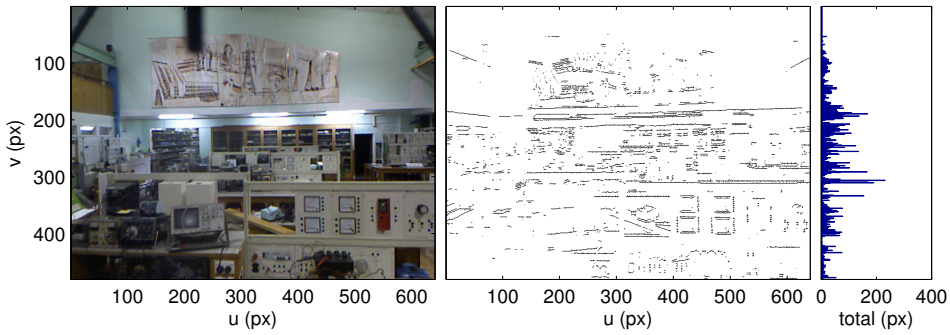
Consider the images in Figure 7.16. In a real environment I observed that we see a characteristic distribution of horizontal edge information. The most relevant information (from the perspective of objects we may wish to avoid) is closer to us and at a similar altitude to our eyes. This is shown on the right of the figure by the total sum of all edges along the vertical axis.

Looking directly ahead, if we wanted to avoid all obstacles in our path, one strategy may be to try to place all the visual information below us (consider how featureless the sky is). If we wanted to fly at an altitude that is likely to contain interesting objects, we should instead keep those horizontal regions of the image at or near some fixed position, such as the horizon or the centre of the frame. In a static environment, with the distribution of edges unchanging, by keeping the longest line (or largest sum in the v-axis) at a constant angle from the centre of the frame we can control our altitude. In

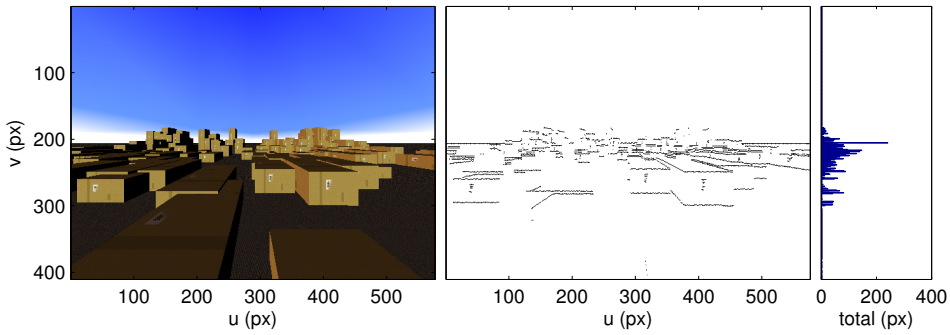
[6] Edges describe only a local feature, defined in image processing terms as a discontinuity in brightness. Lines in this context describe a global feature; the arrangement of edges in a formal pattern.

[7] Maimon et al. [2008] do offer a comparison to a similar result found in houseflies where feature detecting cells respond to vertical contours which might be described as vertical lines.

[8] The Hough transform method for detecting lines is $O(N^2)$ for N edge points and can have large memory requirements if searching for lines with a wide range of orientations [Shapiro and Stockman, 2001].



(a)



(b)

Figure 7.16: Images demonstrating edge information and detection. On the left is the original image, $\mathbf{I}$, in the centre is the binary image showing horizontal edges, $\mathbf{B}_v$. To the right is the sum of horizontal edges in the v-axis, $\mathbf{E}_v$. (a) A real image captured from the quadrotor helicopter while flying in a laboratory. (b) An artificial image from our simulated virtual world.

addition, the ability to hold lines at fixed angles from the centre of the field of view is reminiscent of similar capabilities *Drosophila* used when approaching or avoiding lines of varying length, described in Maimon et al. [2008].

From these insights and the biological literature, I hypothesize that a control system using only a measure of horizontal edges may be useful as an altitude control strategy and simultaneously as a means for choosing an optimal height to fly when exploring a new environment.

7.4 EDGE BASED ALTITUDE CONTROL SYSTEMS

In this section I develop an altitude control strategy for the 'wasp' quadrotor which uses only horizontal edges detected in an image of its environment (Figure 7.17).

Initial experimentation was undertaken in MATLAB using the custom VRML visual simulation environment (Appendix C.2) and quadrotor model (Appendix B.1). This simulation was chosen to provide the necessary visual fidelity and the correct simulation of camera motion during manoeuvring.

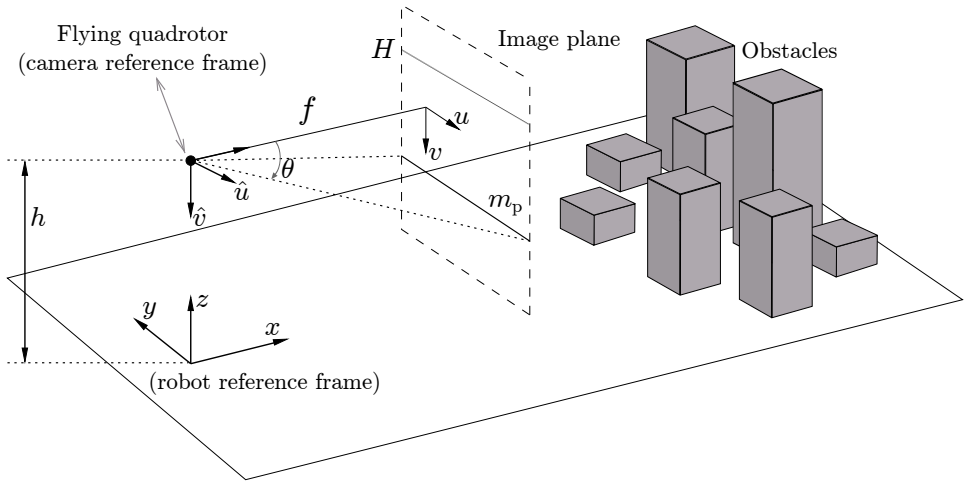


Figure 7.17: A description of the geometry of the image and control system. f is the focal length of the lens, h is the altitude. m_p represents the row, v, in the image, $\mathbf{I}$ (image plane, u, v), containing the most horizontal information. H represents the horizon. The quadrotor position in the world is measured in x, y, z.

Real flight testing used the 'wasp' quadrotor, configured with the Kontron SBC and Microsoft Kinect sensor. Flight testing took place inside the cluttered workshop shown in Figure 7.18.

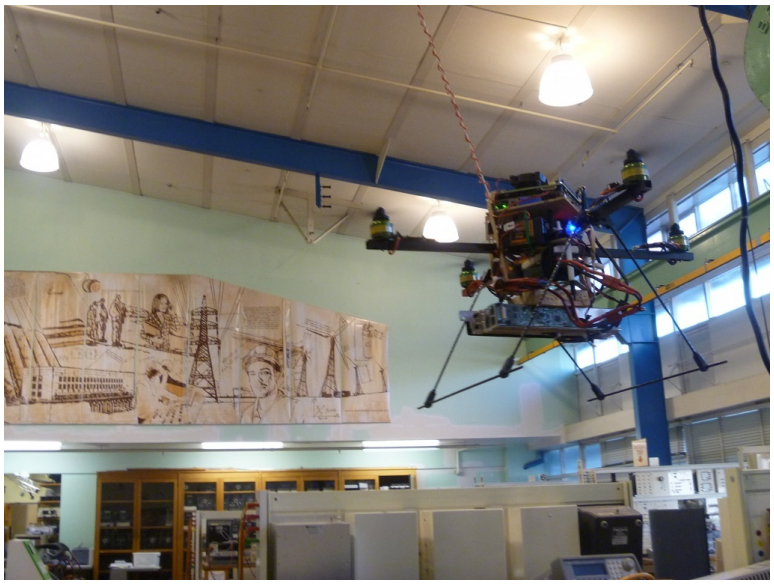


Figure 7.18: An image showing the quadrotor in flight during an altitude control experiment.

7.4.1 Edge Detection

In the absence of neurophysical evidence[9] on how *Drosophila* or other insects detect edges I stick to the conventional computer vision techniques for detecting edges explained in Section 7.1.1. Specifically, by calculating the image gradient in the vertical, v, direction using convolution with the Sobel kernel of (7.2). From eqs. (7.4) and (7.9) I calculate the binary image $\mathbf{B}_v = \mathbf{B}_v[m, n]$ in the v-axis.

The binary image is then summed in the horizontal direction,

$$\mathbf{E}_v[m] = \sum_{n=0}^{N-1} \mathbf{B}_v[m, n]. \tag{7.37}$$

This is a measure of horizontal information content in the image (Figure 7.16, right). The row, m_p, is that with the largest sum of horizontal information and our approximation to the row containing the longest line. It is calculated using

$$m_p = \arg\max_m \mathbf{E}_v[m]. \tag{7.38}$$

I recreated the Straw et al. [2010] result to test these two hypothesis;

- that the largest sum of edges is a reasonable approximation of the longest line in an image (while avoiding more computationally expensive line / contour fitting techniques such as the Hough transform)
- that such a sum is reasonably static relative to the craft's altitude and thus suitable for an altitude control system.

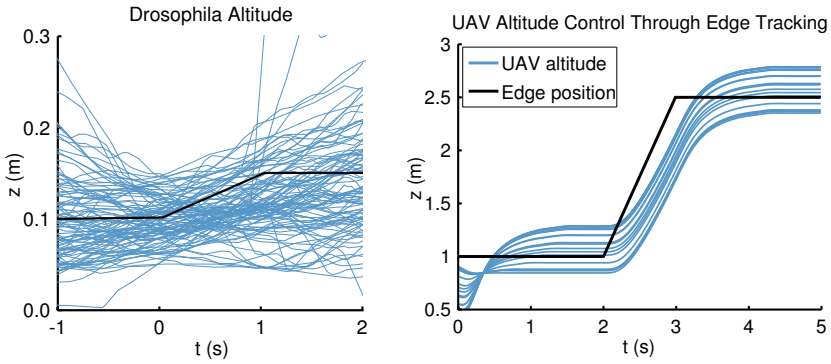
Using VRML, I constructed a virtual model of the Straw et al. [2010] experiment (Figure 2.13b); a long rectangular arena with textured walls and floor. The quadrotor was flown into this virtual environment and at $t = 2$ the position of the horizontal edge was changed. Figure 7.19 shows the resulting quadrotor altitude as the control system adjusted the height to keep m_p in the middle of the image.

7.4.2 Evaluating Altitude Control Systems

Similar to how Dittmar et al. [2010] demonstrated insects can place objects relative to a detected horizon, I attempted to control altitude by keeping the image row with the largest sum, m_p, at a constant location in the image. I call this the 'maximum peak avoidance strategy'.

I make two simplifications; I assume that the insect is level. I also use a sliding-window filtered version of $\mathbf{E}_v$ to reduce noise. I subsequently construct a controller to

[9] Analogous to how early research showed that EMDs were the means by which insects sense image motion.



(a) (b)

Figure 7.19: A recreation of the Straw et al. result to test the control system and to validate the detection of long horizontal edges. (a) *Drosophila* (light traces, one per trial) adjust their altitude in response to the changing position of a horizontal edge (dark trace). An image from this experiment can be seen in Figure 2.13b. Data courtesy Andrew Straw. (b) Results from an artificial recreation of the same scenario. The quadrotor (light traces, Monte Carlo simulation of different starting altitudes) adjusts its altitude in response to the changing position of a horizontal edge (dark trace).

minimise the error signal

$$\Delta = m_s - m_p, \tag{7.39}$$

where m_s is the row where the longest horizontal line is expected. The m_s parameter effectively determines the craft's target altitude and the trajectory taken over obstacles.

Δ is used as the process variable for a proportional-integral (PI) altitude controller

$$h = k_p \times \Delta + k_i \int \Delta \, \mathrm{d}t, \tag{7.40}$$

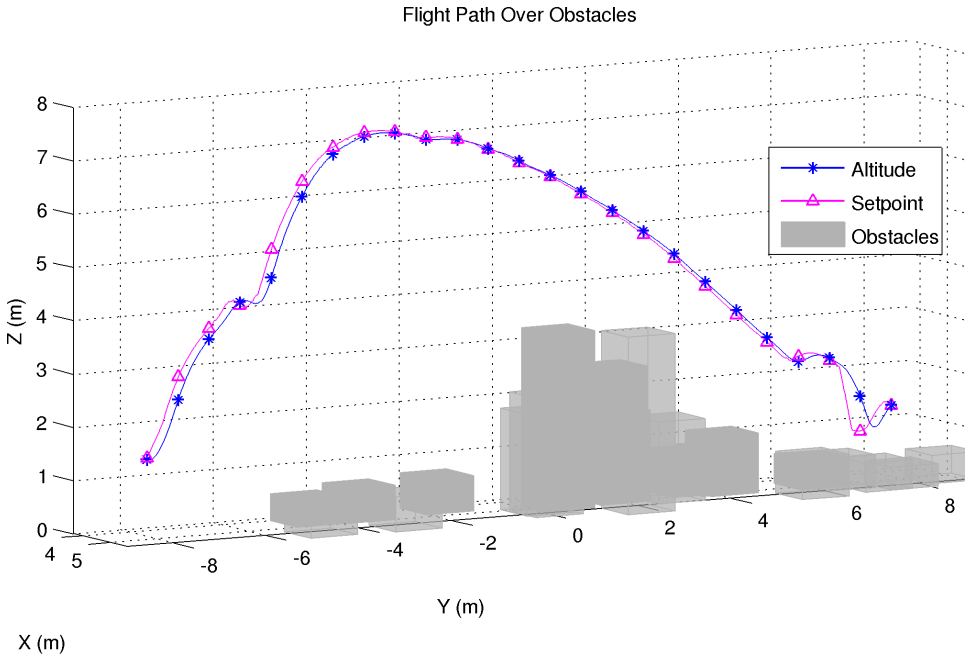
where h is the altitude sent to the autopilot, and k_i and k_p are experimentally determined control gains based on the dynamics of the craft.

To validate this controller I used my simulated VRML virtual environment of Appendix C.2. The simulation environment was designed to contain multiple obstacles at varying heights, laid out in such a way that the controller would need to ensure the craft flew above them during approach and descended once the obstacles were cleared. I chose boxes for the obstacles and distributed them semi-randomly[10] in the environment. A sample image from the virtual environment is shown in Figure 7.16b.

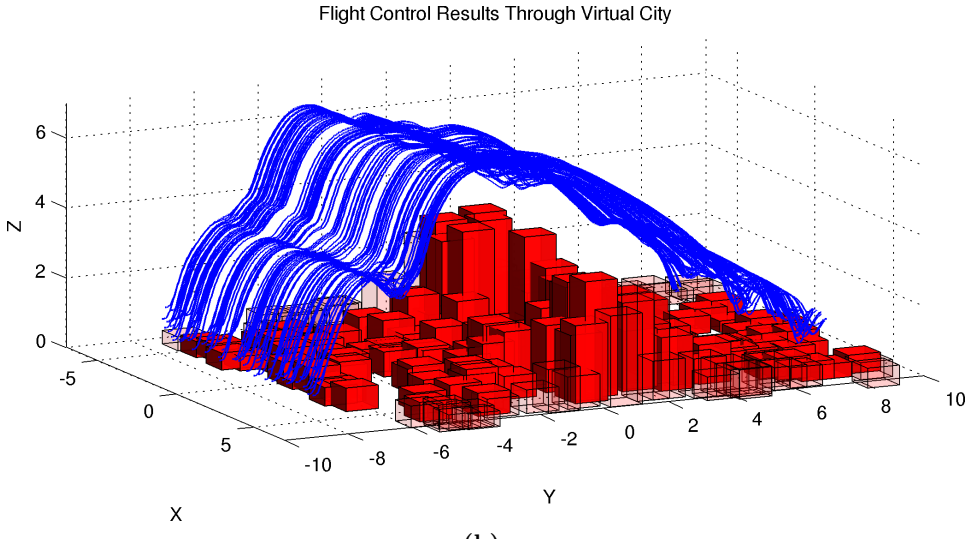
For testing, the UAV autopilot maintained a constant heading and forward speed from the starting point until the end of the simulation. Simulation occurred at 25 Hz, an image was captured, the 'maximum peak avoidance strategy' was run and the quadrotor model was iterated with new h.

Figure 7.20 shows the result of the experiment. A 100 run Monte Carlo simulation

[10] The distribution of heights is a sinusoidally modulated exponential, mirrored in the x-axis and can be seen in Figure 7.20.



(a)



(b)

Figure 7.20: Simulated results for the 'maximum peak avoidance strategy'. (a) A single trial ($n = 35$) representing the normal behaviour of the control system — all obstacles are avoided. The plot shows the recorded altitude of the quadrotor model and the autopilot setpoint, h. Obstacles not on the path of the quadrotor are rendered semi-transparent in this figure for clarity. Only a small number of the total obstacles are rendered (also for clarity) — in the simulation the virtual camera perceives many more buildings as shown in Figure 7.16b (b) Complete results of a 100 trial Monte Carlo simulation.

was chosen with each trial beginning from a randomly chosen x, constant y location. The performance of this control system was consistent; all runs avoided the buildings, the controller ensured a path well clear of the obstacles. The strategy descended once clear of the obstacles. However, as shown in Figure 7.20, by giving equal weighting to distant edges as to near edges, the 'maximum peak avoidance strategy' caused the simulated quadrotor to fly unnecessarily high. This behaviour was unwanted and also inconsistent with the limitations of insect visual systems [Zanker and Zeil, 2000] which cannot resolve such features at large distances [Maimon et al., 2008].

To test if imposing these limitations on my controller would improve the performance, I removed the ability for the system to resolve edges at large distances by using the depth estimate from the Kinect sensor. This should ensure that the craft does not fly unnecessarily high and should be more consistent with the limited resolution of insect vision systems. I call this new algorithm the 'maximum near-peak avoidance strategy'.

The 'maximum near-peak avoidance strategy' is constructed as follows. Let $\mathbf{D}$ be a depth image with each pixel containing the distance between the craft and objects in the environment. I modify the binary edge image $\mathbf{B}_v$ according to;

$$\mathbf{B}_v[m,n] = \begin{cases} 0 & \text{if } \mathbf{D}[m,n] \geq D_{\max} \\ \mathbf{B}_v[m,n] & \text{otherwise} \end{cases}, \tag{7.41}$$

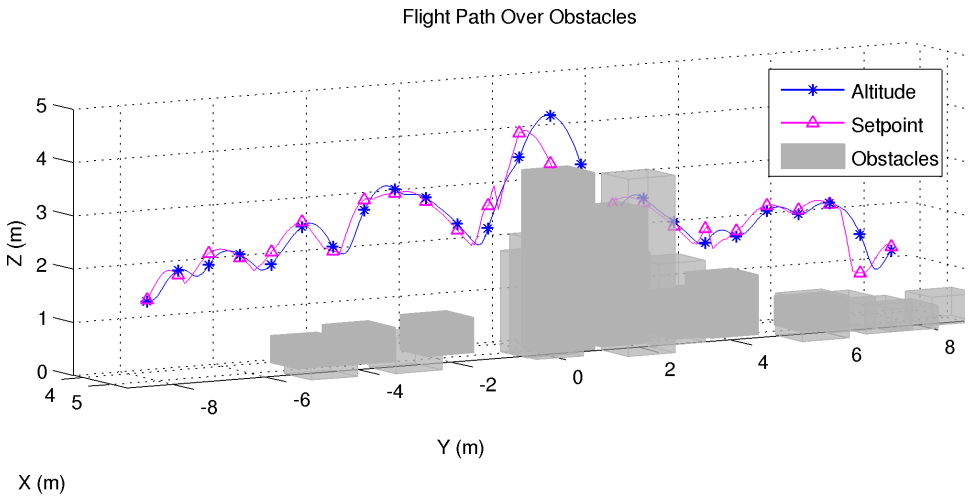
where $D_{\max} = 3\,\text{m}$, a number chosen experimentally, far enough from the craft that it gives a reasonable time to climb over obstacles but not so far that it caused unnecessarily high flight.

Figure 7.21 shows the result of the 'maximum near-peak avoidance strategy'. In this instance the craft was able to avoid obstacles but not fly unnecessarily high above them. When descending towards objects (seen in the right of Figure 7.21a), sometimes a strong response and overshoot is observed. This is in part due to the simplistic PI controller (some overshoot is always expected) and also due to the edge detection algorithm not detecting edges until they were close to the camera which elicited a very strong proportional response.

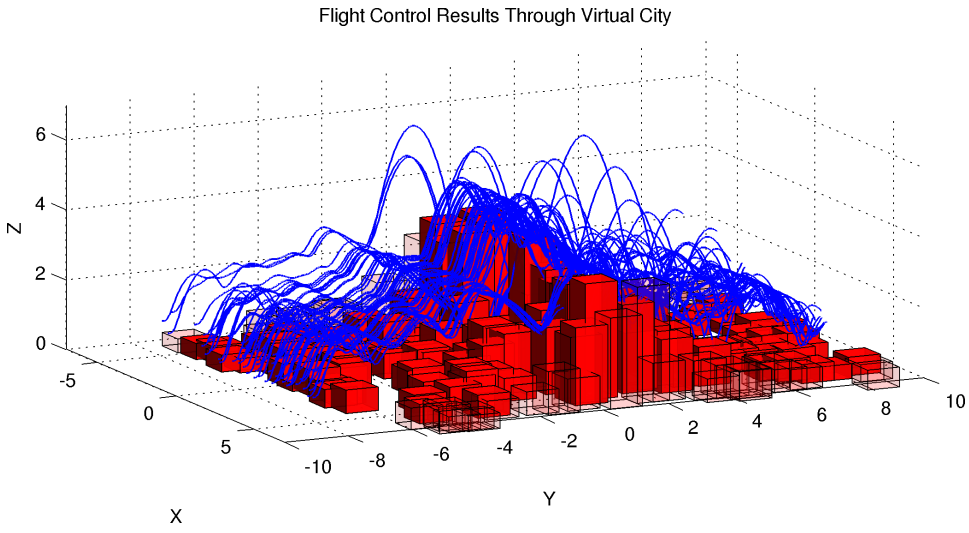
These successful simulations demonstrated the controller has merit. The following section applies the controller to real data.

7.5 ALTITUDE CONTROL RESULTS

To recreate the 'maximum near-peak avoidance strategy' I needed a way to eliminate distant edges. There are numerous techniques for obtaining depth from a sequence of images, for initial flight testing, I extended our previous work and used the Microsoft Kinect [Stowers et al., 2011b] sensor which provided measurements of distance to ob-



(a)



(b)

Figure 7.21: Simulated results for the 'maximum near-peak avoidance strategy'. (a) A single trial ($n = 35$) representing the normal behaviour of the control system — all obstacles are avoided. (b) Complete results of a 100 trial Monte Carlo simulation.

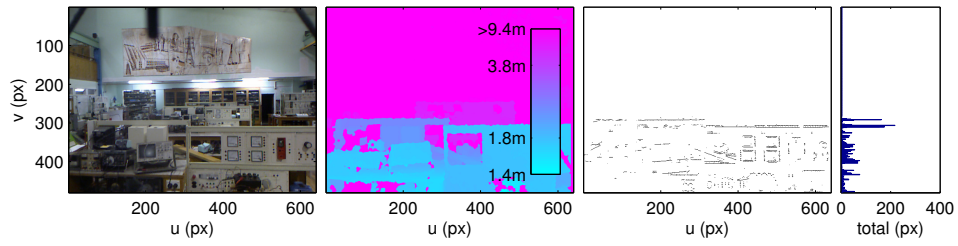


Figure 7.22: A sample frame from a flight experiment. From left to right; the image, $\mathbf{I}$, the depth map from the Kinect, $\mathbf{D}$, the binary images showing horizontal edges within 3 m of the quadrotor, $\mathbf{B_v}$, and the sum of those edges in the $\mathbf{E}_v$.

jects, $\mathbf{D}$, directly. Using $\mathbf{D}$ as outlined in (7.41), I removed distant edges. The effect of this on $\mathbf{B}_v$ is apparent in Figure 7.22.

The quadrotor was placed in an 'attitude hold' autopilot mode. The altitude, h, was provided using the 'maximum near-peak avoidance strategy'. Position in x, y was controlled by the UAV autopilot. The experimental environment was a cluttered workshop containing three benches (visible in Figure 7.16a). The UAV was commanded to maintain a slight forward attitude such that it would approach the benches (obstacles). The autopilot held approximately constant heading and forward speed.

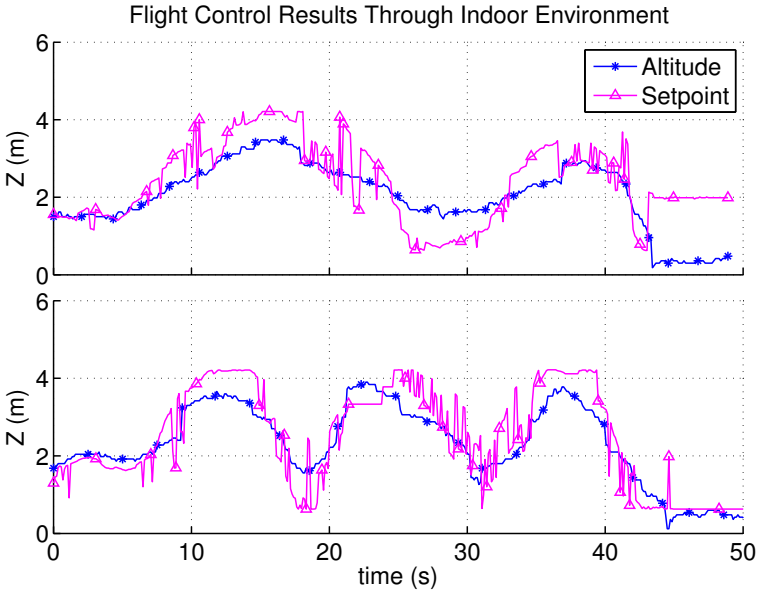
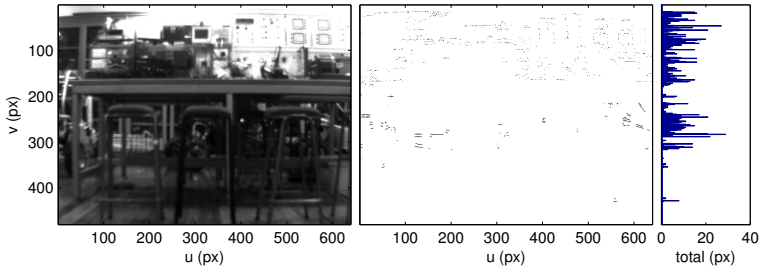


Figure 7.23: Two flight test results from the 'wasp' quadrotor. Each test begins with the helicopter in flight, the control system was enabled at $t = 0\,\text{s}$ and proceeds until $t = 45\,\text{s}$. Both plots show the current altitude and the altitude setpoint given by the 'maximum near-peak avoidance strategy'.

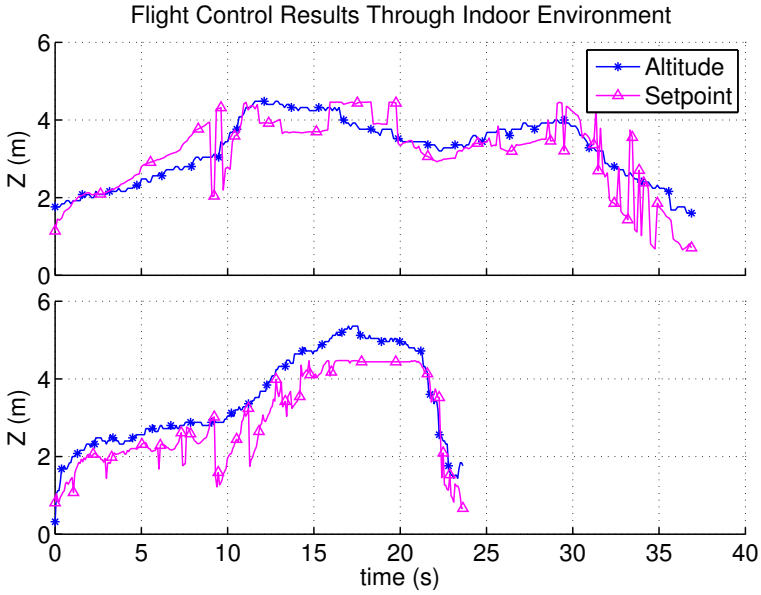
Figure 7.23 shows the result of this experiment. As the UAV moved toward the object and it became visible (or more correctly, its edges became visible) the 'maximum near-peak avoidance strategy' requested a change in altitude to steer the craft over the benches without collision.

Requiring the cumbersome Microsoft Kinect for depth measurements is not the only way to ignore distant objects. By exploiting the depth-of-field property of optical lenses and selecting an appropriate threshold for edge detection, it is possible to remove them from $\mathbf{B}_v$.

Figure 7.24 shows flight tests using this approach. Using a Firefly MV imaging sensor and attaching a variable focus lens (focal length $\approx 1.1\,\text{mm}$) I was able to adjust the focus point to be $\approx 1\,\text{m}$ to $2\,\text{m}$ in front of the UAV. This was sufficient to prevent distant edges from detection; thus approximating the effect of $D_{\max}$ in the previous section.



(a)



(b)

Figure 7.24: Flight tests showing the performance of 'maximum near-peak avoidance strategy' on images which have reduced depth-of-field. (a) An image showing the effect of focusing the lens closer to the UAV, thus reducing the depth-of-field; the edges closer to the camera dominate $\mathbf{E}_v$. (b) The result of two flight trials. In the first trial the UAV successfully climbs over the benches. In the second trial, the UAV climbs over the first bench but overshoots the setpoint and flies too high (I limit the maximum commanded altitude) so I end the test after $t = 23\,\text{s}$.

Figure 7.24 confirmed that the obstacle avoidance and altitude control behaviour first demonstrated in simulated experiments, and later by using the Kinect depth map, can be recreated using a lens with a short depth-of-field. The results also showed (in particular Figure 7.24(b)), that the choice of $D_{\max}$ is important, and a value of $\geq 1\,\text{m}$ to $2\,\text{m}$ should be selected. Such a range was found to provide a control signal strong enough to avoid obstacles at this distance, without excessive overshooting.

7.5.1 Tuning the Controller

The tuning parameters of the control strategy; $D_{\max}$ (the distance threshold) and m_s (the relative altitude setpoint in the image) are both experimentally derived. In my

experience these are a function of the UAV dynamics and not the scene geometry; the rate at which it can climb, and the rate it moves towards obstacles. A heavier or faster moving UAV should define a larger $D_{\max}$ (and perhaps control gain) so it has time to see and climb over obstacles as it approaches them. Similarly, m_s relates the geometry of the image sensor to the scene. For an image plane perpendicular to the ground (i.e., a camera that looks forward) m_s should be located below the midline of the camera's field of view. However, if the image sensor is mounted looking slightly toward the ground then m_s could be increased toward the horizon (for example). In the experiments shown, $m_s = 150\,\text{px}$, or half below the midline. This corresponds to $\approx 10^\circ$ on the Kinect (48°) total FOV, and $\approx 15^\circ$ on the Firefly MV ($\approx 60^\circ$ FOV).

I believe the intrinsic relationship between the UAV dynamics and these tuning parameters is consistent with an evolutionary view of how similar behaviour constants should be related to the animal in which they evolved. That is; such parameters should be useful in more than one environment, but specifically tuned to the organism.

7.6 CONCLUSION AND FUTURE WORK

Both simulated and real flight test results have demonstrated that simple computer vision techniques can reliably detect edges in the environment. Furthermore, by understanding properties of the distribution of these edges, an efficient altitude control system can be developed.

I began by recreating the Straw et al. [2010] result, demonstrating that the largest row-wise sum of edges is a reasonable approximation of the longest line in an image and that this feature is sufficiently stable to control the altitude of a quadrotor. Motivated by this result, I investigated whether other properties of the row-wise sum of edges could be used to build efficient controllers. Rather than representing the longest line per se, I demonstrated that this maximum row-wise sum also represents obstacles that a flying robot should avoid ('maximum peak avoidance strategy'). Extending on this result I showed that by applying biological constraints (the limited visual *acuity* of insects), and intentionally defocusing the lens, a more effective control strategy can be implemented. From these experiments I conclude that 'maximum near-peak avoidance strategy' is an effective biomimetic strategy for avoiding obstacles and controlling altitude in an unknown environment.

Straw et al. suggested that the evolutionary justification for *Drosophila* tracking horizontal edges may be to provide landing spots or areas that may contain food. Similarly, I suggest that 'maximum near-peak avoidance strategy' provides a reasonable choice of altitude for exploration in an unknown environment, one that contains high information content if the primary UAV task is searching and one that takes the craft above obstacles and avoids collision.

The algorithm is computationally efficient; the input image is low resolution, the edge-detection convolution kernel is small (and readily parallelizable [Khalid et al., 2011]), and the line detection is a sum and an arg max operation.

Like real biological systems this strategy should operate in conjunction with other concurrent control processes. For example, it could run at faster-than-realtime; providing an altitude reference or obstacle avoidance cue while an onboard SLAM algorithm is performing its expensive bundle adjustment step.

There is scope for improvement in the system I have implemented. Like Tanaka et al. [2012], I should consider a more robust (or perhaps a lead compensated) attitude and altitude controller to prevent overshoot. Optical tuning should be undertaken using a variable-focus, variable-aperture, variable-zoom lens to better constrain the depth of field. Given depth information, one could investigate weighting the contribution to Δ ((7.39), (7.40)) by the distance of the edge from the horizon (or other setpoint) in a non-linear fashion. This could give a smoother response as approaching an obstacle. Similarly, one could also weight edges in the centre of the field-of-view higher than those at the edge of the image. In some environments, such as negotiating a clear path between obstacles on either side, this would give a better response by not increasing altitude unnecessarily.

7.7 SUMMARY

This chapter demonstrated new ways that edge information in images of an organisms environment can be used to implement a biologically inspired control system. Section 7.2 presented a statistical model which demonstrated it is possible — by analysing the distribution of edges in a scene — to extract absolute altitude if one has some biologically plausible assumptions about the scene structure.

Section 7.4 showed a biomimetic altitude control and avoidance strategy that used the distribution of horizontal edges in an image to navigate an indoor environment. Accounting for depth-of-field, this distribution is sufficiently static to provide a convenient local set point for altitude in a way similar to *Drosophila* [Straw et al., 2010].

Chapter 8

CONCLUSIONS

This thesis has presented a number of biologically inspired visual control systems for flying robots. To test each idea I simulated, constructed, and implemented each strategy on a quadrotor system which I developed. Some hypotheses have been more successful than others; the edge-based altitude controller of Section 7.4 was sufficiently robust to control quadrotor altitude and to avoid obstacles. Other work, such as the time-to-contact log-polar implementation formed just a small part of a larger system. Principles such as; considering new ways to sample data efficiently, and the use of different domains to amplify or attenuate certain properties of image, occurred in several forms in different implementations.

This chapter reviews the contributions of this thesis and describes areas of future work to be undertaken.

Chapters 1 and 2 presented the capabilities of biological and existing computer vision systems, and how the economy and efficiency of the insect visual system can provide useful inspiration for designing artificial control algorithms. Chapter 3 introduced image motion and its estimation using optical flow. This included a discussion of existing and common optical flow algorithms and presented the novel technique I developed; the shear-average (phase-gradient correlation) technique. Chapter 4 introduced quadrotor helicopters and the 'wasp' system I designed. The 'wasp' system and quadrotors are further explained in Appendices A and B.

Chapter 5 describes *attitude* control systems using optical flow. I find that image processing in the log-polar domain is computationally efficient and allows extraction of translational and rotational components of image motion. The log-polar domain and images represented in it, share many attributes of foveated imaging systems and the connection patterns (defining the receptive field) of *neurons* in insect visual systems. The multi-resolution property of the transform reduces the amount of data to process. By concentrating the high resolution parts around the focus of expansion it is easy to implement an efficient collision avoidance time-to-contact (TTC) system.

Chapter 6 investigates the role of depth in flight control. Section 6.1 describes a control system combining two local strategies; a divergence template for navigation

using wide-field optical flow balancing, and an obstacle avoidance strategy using depth. This system was able to navigate a simulated world, but its performance was not sufficient for stable operation in real flight testing. Section 6.2 presented a real time altitude control system based on detecting the ground plane in depth measurements. A biologically inspired sampling strategy promoted efficient selection of possible ground-borne points and reduced computation time. I was the first one to demonstrate quadrotor control using a Kinect (only 2 weeks after release), and the implementation and application of the randomized Hough transform (RHT) in this way was a novel first.

Chapter 7 describes work using the detection and distribution of edges (and lines) in images to control quadrotor flight. Section 7.2 introduces a statistical model based on the sampling properties of image sensors and the geometric properties of many natural scenes which can estimate altitude. The implementation was robust, performing well for multiple flights in several environments. The model is computationally efficient, biologically plausible, and hints at a possible way that real organisms could use information in this way to estimate altitude. The model was simulated and tested on real flights with real data successfully. Section 7.4 is another altitude controller from a similar idea; that the distribution of edges in the environment is locally static and reasonably predictably distributed. By detecting the structure of the distribution of these edges a robot can maintain altitude and avoid oncoming obstacles. Unlike strategies based on image motion the implementation does not require self motion. It is also biologically plausible; computationally efficient and non iterative. To the best of my knowledge, the 'maximum near-peak avoidance strategy' was the first edge-based biologically inspired UAV control system. It was able to successfully replicate the altitude control behaviour [Straw et al., 2010] of *Drosophila* and control a quadrotor helicopter. I also understand the treatment of edges in an image as a Poission process, and the application of this to estimate real world state, is a novel first.

The complete 'wasp' quadrotor, flying as discussed in (Chapter 7), did so with software and multiple biologically inspired control algorithms running. The TTC collision avoidance strategy was ever-present, ready to prevent further forward motion if collision was imminent. While I also retained a manual override, the final implementation regularly demonstrated to me that biological controllers have merit, if only they could be combined in an efficient, quantitative and principled way. The following section describes the future work and research on biologically inspired control I believe necessary.

8.1 FUTURE WORK

When implemented, each idea showed limitations and areas for improvement. While some of those limitations were presented in the relevant chapters, they are expanded on here. In addition I also include recommendations for hardware changes necessary

to test future hypotheses.

8.1.1 Hardware

In the years since starting this thesis a number of commercial and open-source quadrotor systems have become available. Given sufficient financial support I would switch to one of the Asctec branded research quadrotors, likely the 'pelican' (see Appendix A.5). After personally flying many commercial quadrotors I believe the Asctec attitude controller to be very stable (and with a well specified navigation command interface) and would continue to do research based on this. The 'pelican' includes sufficient payload capability to carry additional cameras and processing power for a complete panoramic field-of-view.

Often I was hindered by the lack of an external system to provide true measurements of the quadrotor state[1]. This made quantitative comparison and tuning of control loops in a thorough manner difficult.

At times I was limited by the performance of the 'wasp' attitude controller and would consider the presence of a more robust attitude controller on a commercial quadrotor quite an attractive reason to move away from my custom system. While PID based systems like mine are robust to changes in the model parameters they do not make aggressive autonomous flight easy; tending to err on the side of over-damping and incurring slow dynamics. Because biological responses like obstacle avoidance are quite strong (dominating other concurrent strategies) and I was unable to test such responses on the 'wasp' system due to this over-damping (and also actuator saturation from carrying excessive weight).

Since mid 2010 the Robot Operating System (ROS)[2] project has improved greatly and become the de facto standard for robot middleware. While I have several reservations about their architecture, the ROS project contains implementations of almost everything one would need to perform quadrotor research. I would suggest rebasing what 'wasp' software is still useful (the GUI and vision code) on top of the ROS middleware and continuing research using that.

8.1.2 Direct Control

I was not entirely satisfied with the performance of the TTC system. It was overly susceptible to noise; thus to make it reliable I had to reduce its integration in the whole 'wasp' system down to a single binary state of 'avoid' or 'not avoid'. This work should be re-applied using the approaches outlined in Tistarelli and Sandini [1993]. In that work the authors take a more rigorous approach to the log-polar transformation

[1] A motion capture system, such as those made by `http://www.vicon.com/` are commonly used by other Universities

[2] `http://www.ros.org/wiki/`

and formulation of TTC; computing it in addition to a relative distance measurement (I will come back to this point shortly). The complete formulation only involves image-derived parameters; like velocity and its derivative, and no motion parameters. The optical flow does not have to be differentiated to recover the translational flow nor does the FOE position have to be computed. These are very important features of a TTC algorithm because, as I found, the rotational velocity and the FOE are computed indirectly from the optical flow and are therefore, both subject to errors. To the former point, the formulation of an estimate of relative depth in the Cartesian coordinate system (Section 6.1.5.1, (6.11)) can also be accomplished in the log-polar domain using the approach of Tistarelli and Sandini.

In addition, the formulation of relative (and absolute given inertial information) distance in the log-polar domain would be helpful to estimate altitude in a more thorough manner. This would be appropriate to extend the work discussed in Section 5.3 which otherwise only considered heading (and this disregarded one of the axes in the log-polar domain).

If using a frequency domain optical flow computation technique (such as Section 3.3) it would also be wise to investigate the work of Srinivasan [2000], a FFT based method for extracting the FOE. Staying in the frequency domain and considering the biological implications, I suggest a study of Kern et al. [2005] with respect to its applications on artificial systems be completed. In that work the authors suggest that image motion, specifically retinal image flow evoked by translation (as observed by lateral facing eyes) of the blowfly contains information about not only yaw, but about nearness of obstacles in the environment. Kern et al. suggests that by combining opposing facing HSE cells (cameras in this case) the translational optic flow components can be enhanced. In particular; the summation of the responses almost exclusively signifies forward velocity, while differences between the responses almost exclusively signifies the sideward and yaw velocities. The signals can be separated by low-pass and band-pass filtering, respectively. Perhaps yaw, forward, and sideward velocity can be extracted efficiently by considering it in the frequency domain and using these simple operations.

8.1.3 Edge Based Controllers

The edge based biologically inspired control strategy was an exciting discovery and presents many opportunities for future work. I shall separate this work into two classes; improvements to the statistical model of edge distribution and the use thereof, and further development of the ‘maximum near-peak avoidance strategy’.

One important limitation of the Poission edge model (Section 7.2) is the necessity of assuming the parameters of the distribution, or as the affect the image; the number of horizontal lines in the environment. In Section 7.6 I suggest that given a low resolution

visual system that cannot see great distances, the local structure of the environment, and the evolution of an insect to inhabit it, it may be realistic that the organism has evolved suitable values for such parameters. I also propose another means to extract a useful and quantitative altitude estimate. Section 7.2.4 showed that the Poission parameters are constant per scene, I suggest that a saccadic flight strategy may allow extraction of these unknown parameters. Furthermore, due to their constant nature; the model could be used to extract a time-to-contact (TTC) like quantity by considering the change in altitude estimate as the quadrotor moves up or down.

I noted in Section 7.6 the improvement on performance that (predictably) only considering nearby edges had. Two other strategies should be investigated to achieve this. A variable-focus, variable-aperture, variable-zoom lens should be used for experiments to understand the relation between depth-of-focus and the control parameters $D_{\max}$ and m_s (see Section 7.5.1. Relative distance estimation could also be used to constrain the depth-of-focus. This relative depth estimate could be obtained by considering inertial data and optical flow in an approach similar to Section 6.1, or a comparable formulation in the log-polar domain [Tistarelli and Sandini, 1993]. Weighting edges in the centre of the field-of-view, or closer to the insect, could give a smoother altitude response.

Recently a new generation of micro optical sensors has been produced by Dr. Geoffrey Barrows (of Barrows and Miller [2001], Barrows et al. [2002]). Using these sensors[3] I believe that a real-time wide field of view edge-based altitude controller could be implemented on a very small flight platform and microcontroller using the principles I outlined. This would be a novel first.

Regarding the biological plausibility of this mode of control; Gerke et al. [2011] recently demonstrated an exciting helicopter controller which detects edges from an array of EMDs. The Borst laboratory also continues to investigate EMDs and their applicability to UAV control [Plett et al., 2012]. I suggest applying the edge based controller using an array of EMDs to further the argument that it is biologically plausible.

8.2 FINAL REMARKS AND THE APPLICABILITY OF BIOLOGICALLY INSPIRED CONTROL

The goal of implementing control systems with the robustness and flexibility of those found in nature still remains. In this work I have investigated and created new ways of analysing visual information to create quadrotor controllers. The inspiration for these control systems has always come from the field of biology, from behavioural experiments that shed light on the structure of insect visual control systems. I have strived not just to replicate the work of others, which primarily used image motion / optical flow, but

[3] `http://centeye.com/`

to extend it. The highlight of this work was a biologically inspired system which used edges in the environment to control a quadrotor helicopter.

Fundamentally I remain uncertain of the utility of biomimietic control systems when used in conjunction with traditional visual control architectures. If one has already implemented large parts of a SLAM pipeline I believe they should embrace that architecture and do as much navigation and control in the recovered world coordinate frame. While the limitations of large maps with respect to SLAM remain, present-day algorithms are more than sufficient for local control.

However, if one wishes to take a biologically inspired view, there are well tested algorithms that can be combined with traditional approaches. Terrain following using ventral optical flow has been well examined by many authors and the prevalence of a ventral facing camera on mapping and surveying UAVs makes the system readily implementable. Time-to-contact approaches have also been well tested and continue to be use on real robot systems. More complex combinations of biologically inspired controllers remain problematic. Early work in this area by Poggio and Reichardt [1976], Poggio et al. [1991] finds two concurrent controllers (one related to velocity and one related to position), when linearly combined, explain certain aspects of insect flight. The problem however, is that there must be many more controllers cooperating in parallel on a real insect, and understanding their interaction so artificial systems might replicate it remains an unknown and active field of study. Passivity based approaches have been suggested as one way to compose control laws in a way that provably maintains certain properties with a certain definition of stability being one of them.

In addition to biologically inspired control systems, I have often made use of biologically inspired sampling strategies and alternative coordinate systems in which one can represent the environment (log-polar, Hough space, etc). I encourage others to consider such approaches when performing visual control as they present techniques for reducing the computational cost of many algorithms.

Regardless of the immediate utility of biologically inspired controllers, the research into their structure and function still has value. In addition to providing inspiration for engineering, I believe the mimicry of biological systems can be seen as an experimental tool to support other fields of research like biology and behavioural neuroscience; both to validate findings and to propose new hypotheses.

Appendix A

EQUIPMENT

All experimental work was undertaken using a collection of custom hardware and software I developed at the University of Canterbury, and in conjunction with two other institutions; L'Ecole Nationale de l'Aviation Civile (ENAC)[1] and ETH Zürich [2]. The system, henceforth referred to collectively as 'wasp', was constructed for flight experiments necessary for this thesis.

In the last few years, a number of commercial and research oriented quadrotor and visual flight control platforms have become available for purchase. With hindsight, I may have chosen to use one of these systems instead of electing to build my own, however, due to the time and budget constraints at the time, this was not an option. Nevertheless, comparisons with other quadrotor systems will be made at the appropriate places within this chapter and a review of popular alternatives is included in Appendix A.5.

This chapter will begin by introducing the structure of the 'wasp' system (Appendix A.1), its architecture (Appendix A.1.1), and focusing on the aspects of the 'wasp' software that are most useful for visual flight control research (Appendix A.1.2). Appendix A.2 and Appendix A.3 describe the quadrotor hardware (the frame, electronics, controller, etc), and the vision hardware (cameras, lenses, etc). Appendix A.4 describes the different quadrotor configurations — the combinations of different hardware used for the flight experiments. The chapter concludes with a comparison between the 'wasp' system and other flight control experimental platforms (Appendix A.5).

A.1 THE WASP SYSTEM

'Wasp' is a collection of software and hardware (collectively referred to as the 'wasp' system), available under an open source, free software license, that facilitates rapid implementation of control systems for robotics research. 'Wasp' has been successfully

[1] `http://www.enac.fr/`
[2] `http://www.ethz.ch/`

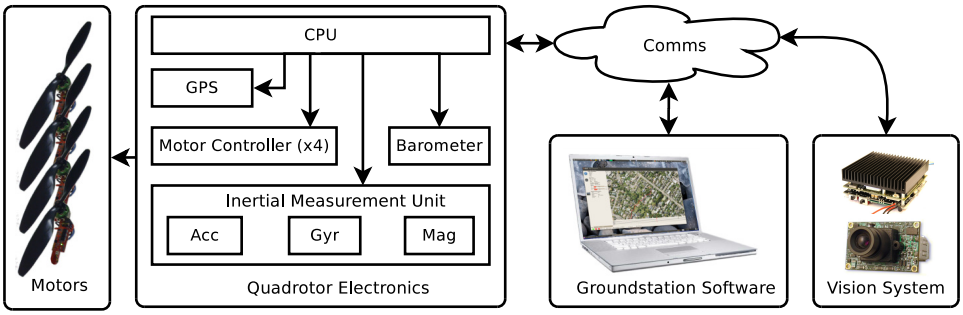


Figure A.1: The 'wasp' system and its application to a quadrotor helicopter.

used to control quadrotor helicopters, fixed wing aircraft and rockets. The system includes a number of components:

- quadrotor autopilot featuring attitude, hover and altitude hold modes,
- fixed-wing autopilot with GPS position control capability,
- groundstation software for visualisation, command and control of UAV flights,
- software in the loop simulation for fixed wing and quadrotor flight vehicles,
- calibration software for IMU and magnetometer,
- hardware designs for quadrotor autopilot, IMU, GPS, and Gumstix based single board computer,
- utility libraries of software for visual flight control,

I derived 'wasp' from the Paparazzi[3] (via ENAC) project in 2008. The two projects have subsequently diverged in capability, design and focus, although code still flows between both projects occasionally. The Pixhawk[4] system at ETH shares code from both 'wasp' and Paparazzi.

A.1.1 Architecture

Figure A.1 shows the key elements of the 'wasp' system. The method of bridging all the elements together is through the communications infrastructure; a standardised message description language. The language makes it easy to add new messages containing information one might want to send between robots, from robots to the groundstation, and between the vision system and the real time attitude controller. C-code for parsing, sending and receiving these messages is generated programmatically. In dynamic languages, it is easy to parse this message description directly (such as is done in the `Python` groundstation).

[3] `http://paparazzi.enac.fr/`
[4] `http://pixhawk.ethz.ch/`

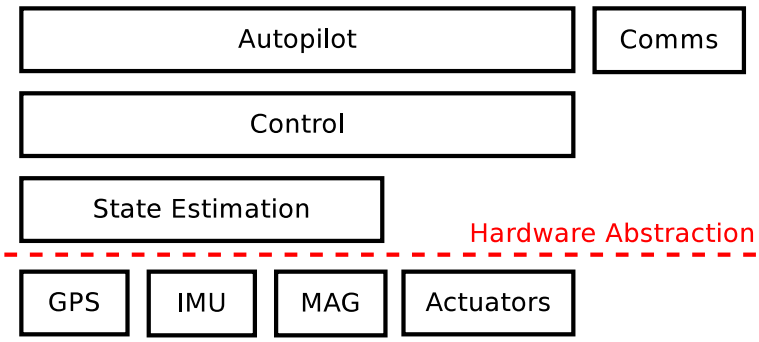


Figure A.2: The 'wasp' onboard software architecture. Each block has defined interfaces for code re-use on other aircraft, and other hardware.

The transport of these messages is sufficiently abstracted, and their content opaque, that they are utilised to allow visual control code to run either on a desktop PC or a single board computer on the quadrotor. More specifically and for example, 'wasp' defines a command messages instructing the autopilot which attitude it should hold; as explained earlier, this is a sufficient platform upon which to implement visual control algorithms.

A.1.2 Software

The 'wasp' software is a mixture of `C/C++` and `Python` code and includes the attitude controller that runs on the quadrotor hardware, and computer vision software to run on the SBC.

The attitude controller includes an extended Kalman filter and a complementary filter state estimator. Onboard code is abstracted in terms of common elements of robotic control, Figure A.2, making re-use on different aircraft relatively easy. For example, in collaboration with colleagues, I developed hardware implementations for ARM7 (section Appendix A.2.2), Gumstix (via a user-space Linux application), and control systems for fixed-wing airplanes and rockets.

Additionally, 'wasp' also includes utilities for use on Linux. These include libraries for preparing synchronized IMU measurements and images acquired from various cameras. This allows fluid mixing of visual control code, it can either run on the same SBC as the attitude controller, or separately.

A computer vision system must not only be capable of providing useful information, but able to provide these in a timely manner to have effective control. Because I aspired to execute computer vision code onboard this imposed another constraint; hardware small and light enough to be mounted on the vehicle.

To achieve these conflicting requirements software was typically developed in the following manner;

1. A prototype was created on the host computer using `Python` or `MATLAB`. Simulations were performed using the tool appropriate for the chosen language and development environment (Appendix C).

2. A beta version was implemented in `C/C++` and software-in-the-loop simulations were undertaken to check performance and consistency with the prototype.

3. An initial hardware-in-the-loop test was performed. On some occasions the code was tested on the SBC using the actual imaging hardware. On other occasions the code was tested on a more powerful computer using images captured using the SBC and transmitted to the host computer over wireless or USB.

4. The `C/C++` version was tested on the quadrotor using a hardware-in-the-loop approach. Depending on the experiment this meant giving limited authority to the controller while under the supervision of a manual pilot, running the system while tethered to the ground.

5. The finished software was deployed onto the SBC for flight testing.

The flexibility and decoupling of function (computer vision processing, real-time control) from computer node was an important feature of development, and one that many other robotic developers realised. The following section compares prominent robot development systems from other laboratories.

A.2 QUADROTOR HARDWARE

A custom quadrotor was constructed in association with ENAC for experiments necessary to this thesis. Figure A.1 shows that the quadrotor features two major electronic subsystems, the hard real-time flight control system, and the soft real-time vision processing system.

This section describes the construction of the 'wasp' quadrotor hardware and electronics. For more detail see Drouin and Warmers [2008], which describes the similar Paparazzi system. For schematics see the 'wasp' website[5].

A.2.1 Processing Electronics

The flight control system is based around a 32bit LPC2148 ARM7 micro-controller running at 60 MHz. This processor runs the computationally intensive control and state estimation algorithms. The quadrotor is flown by radio control in stability augmented mode, or autonomously commanded using a remote ground-station.

The ARM7 micro-controller and related hardware are physically arranged in a stacked manner. As can be seen in Figure A.3, the top PCB contains the GPS receiver

[5] `http://www.waspuav.org/`

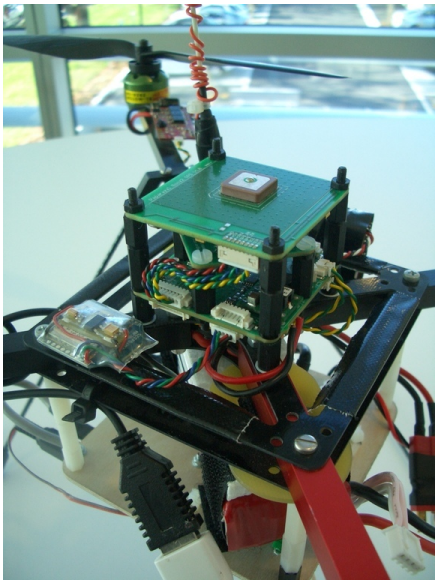


Figure A.3: The 'Wasp' flight control electronics, featuring global positioning system (GPS) receiver (top printed circuit board (PCB)), inertial measurement unit (IMU) (centre PCB, partially obscured), and central processing unit (CPU) (bottom PCB). To the left of the stack is the XBee wireless radio.

and antennae, the middle board contains the inertial measurement uni, and the bottom PCB contains the CPU, power supply, barometer and connectors.

Two forms of communication with the quadrotor are present. An XBee 2.4 GHz modem is used for transmission of flight telemetry and commands to the ground-station. A model aircraft radio receiver is used for commanding stability augmented flight using a traditional model aircraft radio transmitter. The receiver signal is analysed directly by the CPU before generating the necessary control signals for the motor controllers - all fight, even manual, is fly-by-wire.

A.2.2 Quadrotor Chassis and Construction

The quadrotor hardware consists of a cross-frame made of square aluminum tubes joined by GFC plates in the center. This design has proved to be robust, usually only requiring a propeller replacement after a crash. On this frame are mounted four brushless motor / propeller combinations, four brushless motor controller, the avionics and the battery. Two opposed mounted propellers with a distance of 0.37 m are counter-clockwise rotating and the other ones are clockwise rotating.

All control is obtained by manipulating the speed of these propellers. Two opposing pairs of propellers rotate in opposite directions. There are no other mechanical parts to control the flight direction and the speed of the quadrotor.

A.2.3 Quadrotor Drive Section

Pounds et al. [2010] showed actuator performance is a critical element in attitude controller performance. Based on these recommendations I chose Roxxy2824-34 motors with the characteristics listed in Table A.1.

Speed	1000 rpm/V
Power (max)	90 W
Thrust (max)	580 g
Weight	48 g
Dimensions	28.8 × 26 mm

Table A.1: Characteristics of the Roxxy2824-34 motor.

These motors are driven by brushless controllers from Mikrokopter[6] and Speedy-BL[7]. These controllers were selected after experimenataion [Drouin and Warmers, 2008] and based on the findings of Pounds et al. [2010], Hoffmann et al. [2007].

The authors showed that the performance of the motor controllers greatly affects the performance of the rotor system, in both efficiency and dynamics. The performance of the propulsion system is also affected by the refresh rate and synchronous/asynchronous nature of the commands sent to the controller. Commercial hobby brushless controllers typically use a servo PPM signal with a refresh rate of 40 Hz, introducing a 25 ms pure delay. The chosen controllers use digital communications for a refresh rate up to 1 kHz, reducing the delay to 1 ms. Simulation by Hoffmann et al. [2007] and my own experimental experience agree that this parameter plays a significant role in the performance of the craft.

The brushless controllers are connected via I2C. The I2C interface has a data rate of 400 kHz and control loop refresh rate of 200 Hz. The motor and controller generate, in combination with EPP1045 10×4.5 propellers, a maximum thrust of 5.6 N for each motor. All electronics are powered by 3-cell Lithium polymer (LIPO) batteries (11.4 V, 2100 mA h, 180 g).

A.2.4 Inertial Measurement Unit

A second PCB holds the inertial measurement unit. The IMU contains 3 single-axis gyroscopes (ADXR300), 2 dual-axis accelerometers (ADXL320), a 3-axis magnetometer (PNI MS2100) and a barometer (MPX6115). All are sampled at 200 Hz. The barometer is is used in conjunction with the accelerometer and GPS to provide an altitude estimate through a Kalman estimator (see Appendix B.4.2).

[6] `http://www.mikrokopter.de/ucwiki/en/BrushlessCtrl`
[7] `http://www.speedy-bl.com/speedybl-e.htm`

'Wasp' uses North East Down (NED) frame, that is positive x is pointing to the front, positive y to the right and positive z down.

A.2.4.1 Inertial Calibration

Inertial calibration is important for AHRS performance. For the magnetometer, it is also important that the calibration be performed in the fully assembled vehicle, with all systems powered. This is the so-called 'hard-iron' calibration and allows compensating any constant parasitic magnetic field generated by the vehicle.

The calibration process was developed at, and in conjunction with, ENAC. It consists of finding a set of neutrals; $\ddot{x}_0, \ddot{y}_0, \ddot{z}_0$, and scale factors, k_x, k_y, k_z for each sensor. This is formulated as,

$$\begin{bmatrix} \ddot{x} \\ \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} * \left(\begin{bmatrix} \ddot{x}_m \\ \ddot{y}_m \\ \ddot{z}_m \end{bmatrix} - \begin{bmatrix} \ddot{x}_0 \\ \ddot{y}_0 \\ \ddot{z}_0 \end{bmatrix} \right), \tag{A.1}$$

where $\ddot{x}, \ddot{y}, \ddot{z}$, are the true acceleration readings in x, y, z and $\ddot{x}_m, \ddot{y}_m, \ddot{z}_m$ are those measured by the sensors.

The principle of the calibration is as follows. An accelerometer, on a vehicle at rest measures a constant vector (the opposite of gravity) in the earth frame, expressed in the vehicle frame.

$$DCM * \begin{bmatrix} 0 \\ 0 \\ -9.81 \end{bmatrix} = \begin{bmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & k_z \end{bmatrix} * \left(\begin{bmatrix} \ddot{x}_m \\ \ddot{y}_m \\ \ddot{z}_m \end{bmatrix} - \begin{bmatrix} \ddot{x}_0 \\ \ddot{y}_0 \\ \ddot{z}_0 \end{bmatrix} \right), \tag{A.2}$$

where DCM is a rotation matrix that converts between earth frame and body frame. It will change when we change the orientation of the vehicle. Nevertheless, a rotation conserves the norm of a vector. We can thus obtain the following scalar equation that doesn't depend on the vehicle orientation

$$9.81 = (k_x(\ddot{x}_m - \ddot{x}_0))^2 + (k_y(\ddot{y}_m - \ddot{y}_0))^2 + (k_z(\ddot{z}_m - \ddot{z}_0))^2. \tag{A.3}$$

By recording multiple measurements in different orientations one can solve for the set of scale factor and neutral giving the norm closest to 9.81.

To calibrate gyroscopes a record player turntable (or some other method for rotating an object at a known speed) is needed. By placing the IMU at opposing orientations such that it rotates in opposite directions about the same axis, 2 points can be collected. A 3-point linear fit (including the resting, zero) point can be made.

The scale factor for the barometer is calculated using a simple linear fit to a number of measurements recorded at known altitudes.

A.3 VISION HARDWARE

Two types of vision hardware were used for experiments, the Firefly MV machine vision camera hosting a number of lenses, and the Microsoft Kinect.

Specific research undertaken on the Kinect included additional calibration and analysis, so that hardware is presented in more detail in Appendix A.3.2. The firefly MV sensor and lens configurations are included in Appendix A.3.1.

A.3.1 Firefly MV Sensor

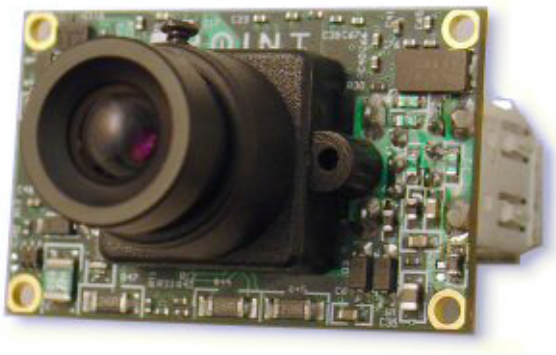


Figure A.4: The Firefly MV machine vision camera.

The Firefly MV[8] USB machine vision camera (Figure A.4) is produced by Point Grey research. The camera features a 1/3" Micron image sensor with 752×480 pixels and a global shutter, essential for sharp images in high dynamic environments. This camera was chosen due to the availability of high quality Linux drivers that allow efficient image transfer over USB with little CPU usage; on both ARM and x86 Linux platforms. The Firefly MV can be see mounted on the quadrotor in Figure A.9.

The Firefly MV accepts C-mount lenses. Two lenses were used with the Firefly MV and the determination each lens' intrinsic properties was made using the conventional 'checkerboard' method [Hartley and Zisserman, 2004, ch. 7]. The parameters of each lens used are listed in Table A.2.

The 'fisheye lens' was a Sunex DSL219A miniature fisheye (wide angle) model. This lens has an effective focal length of 1.8 mm. When mated with the 1/3" image sensor it delivers a 160° horizontal field of view.

The 'normal lens' was of unspecified origin.

[8] `http://www.ptgrey.com/products/fireflymv/fireflymv_usb_firewire_cmos_camera.asp`

	Fisheye Lens	Normal Lens
c_x	3.234×10^2	3.195×10^2
c_y	2.752×10^2	2.554×10^2
f_x	4.448×10^2	2.738×10^3
f_y	4.796×10^2	2.779×10^3
k_1	-3.271×10^{-1}	2.190×10^{-1}
k_2	1.129×10^{-1}	-1.599×10^1
p_1	-1.945×10^{-3}	-1.239×10^{-2}
p_2	-1.196×10^{-3}	-2.643×10^{-3}
k_3	N/E	N/E

Table A.2: Intrinsic calibration values for lens used with the Firefly MV machine vision camera. N/E means the parameter was not estimated.

A.3.2 Kinect Image Sensor

The Microsoft Kinect (Figure A.5a) is a low cost computer vision peripheral, released November 2010, for use with the Xbox 360 game system as a game controller. The device can be modified to obtain, simultaneously at 30 Hz, a 640×480 pixel monochrome intensity coded depth map and a 640×480 RGB video stream.

(a) **(b)**

Figure A.5: (a) An unmodified Kinect. From left to right the sensors are; the IR projector, RGB camera, and monochrome (depth) camera. (b) The Light Coding^{TM}IR pattern projected onto the environment.

The Kinect sensor connects to the PC/XBOX using a modified USB cable.[9]. However, the USB interface remains unchanged, and subsequent to the Kinect release, the protocol[10] was decoded and software to access the Kinect was created.

The Kinect features two cameras, a Micron MT9M112 640×480 pixel RGB camera and a 1.3 Megapixel monochrome Micron MT9M001 camera fitted with an IR pass filter. Accompanying the monochrome IR camera is a laser diode for illuminating the scene. The depth map has 11-bit resolution and the video hardware 8-bit resolution. Both

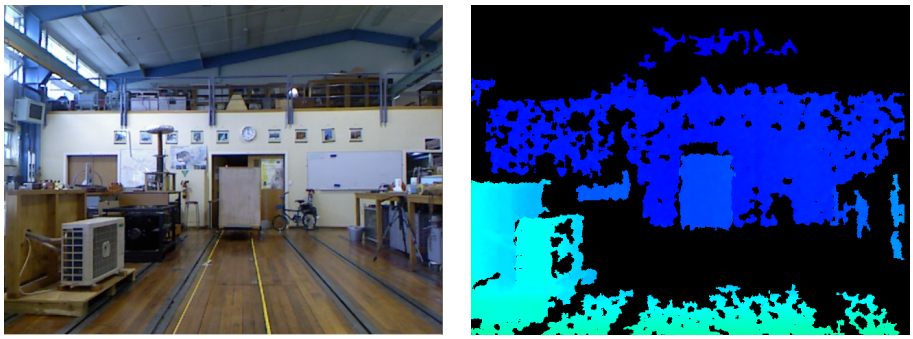
[9] required to provide additional current

[10] gratefully started by the OpenKinect project; `https://github.com/OpenKinect`

cameras deliver 640×480 pixel images at 30 Hz and have an angular field of view of 58° horizontally, 45° vertically and 70° diagonally. The spatial, (x, y), resolution (at 2 m from the sensor) is 3 mm while the depth, z, resolution at the same distance is 10 mm.[11]

A.3.2.1 Calibration of the Kinect Sensors

Depth Camera Calibration: The depth camera returns an 11–bit number (raw values in the range $0 \ldots 2047$) which needs further processing in order to extract the true depth from the sensor. A calibration procedure was performed whereby a number of reference images were captured at known distances (Figure A.6).



(a) **(b)**

Figure A.6: The calibration environment for testing. The board in the centre of the frame is placed 650 cm from the image plane. (a) Image captured from RGB camera. (b) False coloured depth map from depth camera.

This process was repeated multiple times over varied ambient light conditions in order to check the insensitivity of the depth measurement to environmental conditions. The results of this calibration procedure is shown in Figure A.7.

A second order Gaussian model was found to be an appropriate fit ($r^2 = 0.9989$) for the data over the calibrated range. Let depth(x) be the true depth from the image sensor, and x the raw range value, then

$$\text{depth}(x) = a_1 * exp(-((x - b_1)/c_1)^2) \tag{A.4}$$

$$+ a_2 * exp(-((x - b_2)/c_2)^2), \tag{A.5}$$

[11] provided by the PrimeSense reference design for the PS1080 chipset used in the Kinect. `http://www.primesense.com/?p=514`

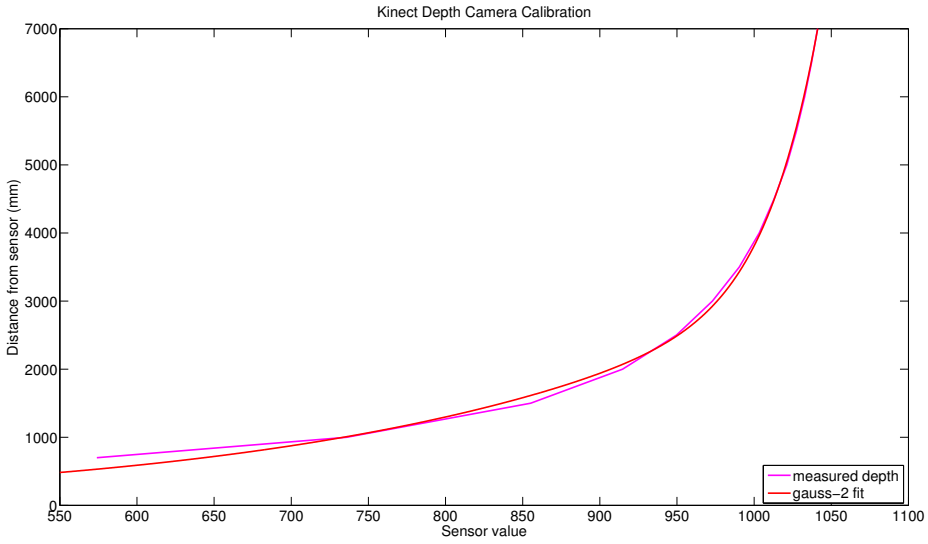


Figure A.7: Kinect depth camera calibration results and line of best fit.

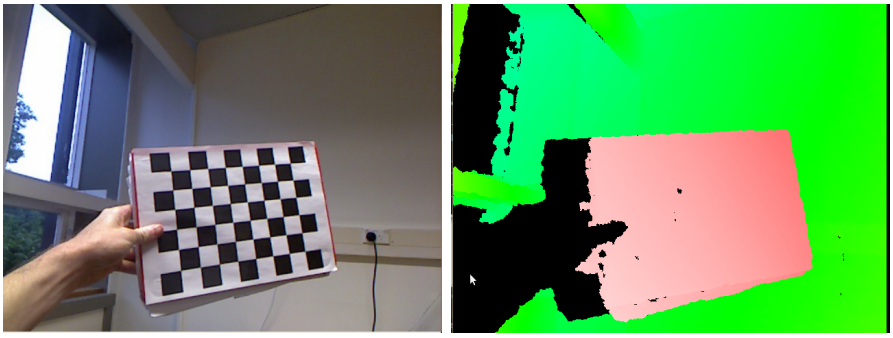
gives the true where

$$
\begin{aligned}
a_1 &= 3.169 \times 10^4 \\
b_1 &= 1338.0 \\
c_1 &= 140.4 \\
a_2 &= 6.334 \times 10^{18} \\
b_2 &= 2.035 \times 10^4 \\
c_2 &= 3154.0.
\end{aligned}
$$

It can be seen from the calibration results (Figure A.7) that the depth map is accurate and repeatable over the $0.4 \ldots 7.0\,\mathrm{m}$ range. Additionally, if the Kinect is unable to estimate the depth to certain regions in the image, those pixels are filled with the value 2047, making it easy to ignore these pixels from further image analysis.

Camera Calibration and Alignment: The intrinsic and extrinsic parameters of both cameras must be known so their images may be represented in a single co-ordinate system. Standard stereo computer vision techniques illustrated in Figure A.8 were used to perform this calibration [Hartley and Zisserman, 2004].

The RGB camera intrinsic parameters was calculated using the chessboard approach and the implementation from `OpenCV` (`cvFindChessboardCorners` was used). Calibration of the intrinsic parameters for the depth camera was performed by manually picking out the chessboard corners from the depth image. The calibration results for the two cameras are shown in Table A.3.



(a) (b)

Figure A.8: The chessboard approach for calculating the camera intrinsic parameters for both two cameras. (a) and (b) Manual matching of 4 points in both images (the corner of the chessboard) in order to calculate R and T matrices.

	RGB Camera	Depth Camera
c_x	2.62×10^2	3.51×10^2
c_y	3.29×10^2	3.02×10^2
f_x	5.22×10^2	5.80×10^2
f_y	5.25×10^2	5.38×10^2
k_1	2.45×10^{-1}	-2.01×10^{-1}
k_2	-8.39×10^{-1}	9.82×10^{-1}
p_1	-2.05×10^{-3}	-7.72×10^{-4}
p_2	1.49×10^{-3}	4.89×10^{-3}
k_3	8.99×10^{-1}	-1.38

Table A.3: Intrinsic calibration values for the Kinect RGB and depth cameras.

The extrinsic parameters, the physical relationship between the cameras, was computed by manually matching the outline of the chessboard between the frames. The rotation (R) and translation (T) matrices were thus computed to be;

$$R = \begin{bmatrix} 9.99 \times 10^{-1} & 1.39 \times 10^{-3} & -1.83 \times 10^{-2} \\ -1.88 \times 10^{-3} & 9.99 \times 10^{-1} & -1.32 \times 10^{-2} \\ 1.74 \times 10^{-2} & 1.20 \times 10^{-2} & 9.99 \times 10^{-1} \end{bmatrix}$$

$$T = \begin{bmatrix} 2.09 \times 10^{-2} \\ -7.12 \times 10^{-4} \\ -1.34 \times 10^{-2} \end{bmatrix}.$$

A.3.2.2 Median Filtering

The Kinect sensor depth measurements are sometimes corrupted by isolated noise, particularly noise cause from reflection of the structured light projection. Fortunately this noise is markedly different from the near uniform plane typically surrounding it, so a median filter can be used to suppress its influence on the depth map.

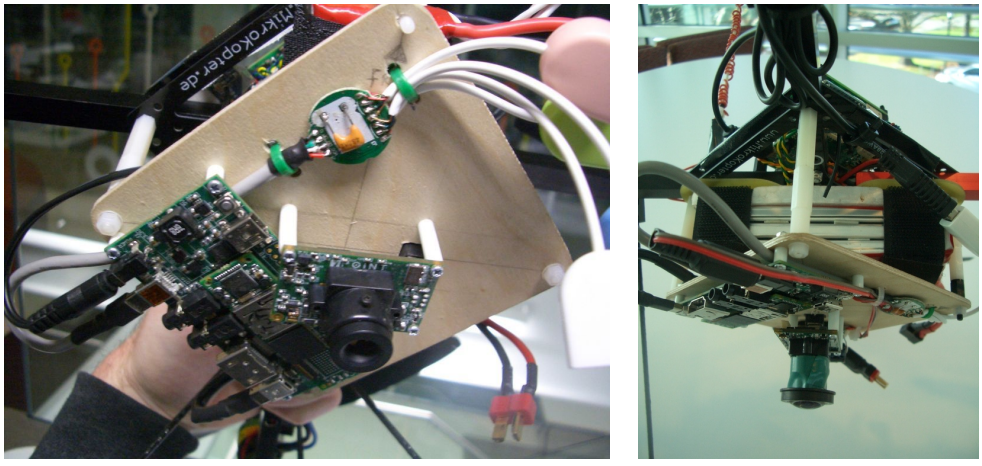
The median filter is an effective method that can suppress isolated noise without blurring sharp edges. Specifically, the median filter replaces a point by the median of all points in the surrounding neighbourhood:

$$y[m,n] = \text{median}\left\{x[i,j], (i,j) \in w\right\}, \tag{A.6}$$

where w represents a neighbourhood around a location (m,n) in the data. When using the Kinect I apply a 5 pixel window 2D median filter over all depth measurements to minimise the influence of erroneous readings.

Computation of the median result can be expensive for large windows and images. The constant time median filter [Perreault and Hebert, 2007][12] was used throughout.

A.3.3 ARM9 Embedded Computer



(a) **(b)**

Figure A.9: The Gumstix single board computer and Firefly MV camera mounted under the 'wasp' quadrotor. The camera is mounted pointing towards the ground and is shown with two different lenses attached. (a) The 'normal' lens. (b) The 'fisheye' lens.

Considerable flight testing and experiments were undertaken using the popular Gumstix single board computer. The 'Overo Earth' model chosen includes a TI OMAP3503 processor, clocked at 600 MHz and also includes 256 MB of flash and RAM.

[12] Source code available online; `http://nomis80.org/ctmf.html`.

Flight code was compiled using the OpenEmbedded toolchain and copied to a microSD card, from which the Gumstix then booted.

The Gumstix features two USB ports; one was used to communicate with the 'wasp' attitude controller via a common usb-serial adapter, and the other (USB host port) was used to power and access the Firefly MV camera.

Experimentation revealed a number of shortcomings with this approach and ultimately the Gumstix was replaced with a more powerful SBC. The principle limitations were;

Insufficient processing power was a problem for some algorithms. While it was possible to compute optical flow using pyramidial Lukas-Kanade on a 320×240 pixel image in real time, this consumed approximately 50% of the CPU leaving little resources for concurrently running other visual processes.

Inefficient USB transfer was a consistent problem using the Firefly MV cameras. Simply capturing video from the cameras consumed considerable CPU power.

Slow disk I/O was a limitation when collecting date for later processing. The peak write speed onto the microSD card was 3 MB/s and occasionally writing became completely stalled while the kernel flushed the write buffers.

A.3.4 Intel x86 Single Board Computer

The second single-board computer used was the a custom designed module featuring a Kontron[13] microETXexpress-PC single board computer (SBC). The SBC included an Intel Core2 Duo 1.86 GHz, 2 GB DDR3 1066 MHz RAM and an Intel GS45 chipset with Intel GMA X4500 GPU (Figure A.10).

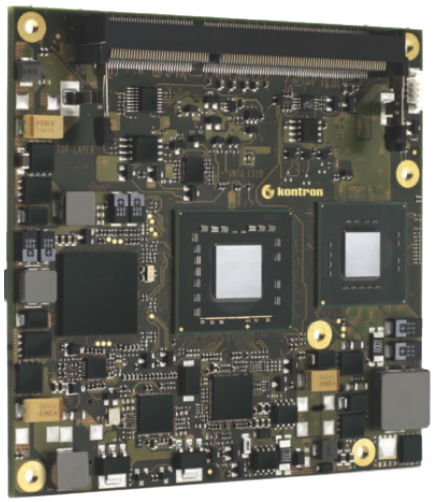


Figure A.10: The Kontron single board computer mounted on the 'wasp' quadrotor.

[13] `http://de.kontron.com`

The development of this module was undertaken by Christian Dobler and supervised by Lorenz Meir and myself at ETH Zürich.

The SBC communicates with the 'wasp' attitude controller via a USB-serial bus, built onto the custom module. All other interfacing is via the 8 USB ports present on the module. The SBC filesystem is stored on a solid-state-disk.

Performance Comparison and Benchmarking

After 18 months development using the Gumstix, the move to an x86 SBC was primarily to overcome the performance shortcomings of the previous platform; slow disk write performance made writing (or reading) images to disk impossible[14], and insufficient processing power limited framerates and image resolution.

Figure A.11 shows the disk and computational performance of the Gumstix, SBC and my desktop computer[15]. Disk benchmarks were performed using bonnie++[16] and CPU benchmarks were performed using nbench[17].

The results show the Kontron SBC performance was a marked improvement on the Gumstix, especially when combined with an SSD.

A.4 QUADROTOR CONFIGURATIONS

Due to the number of vision and processing payloads developed and the way the capabilities of the 'wasp' system evolved throughout this thesis, there were a handful of commonly used quadrotor configurations used experimentally. This section provides a brief summary (Table A.4) and description of those configurations.

	Wasp Gumstix	*Wasp Kontron*	*Wasp Kontron Kinect*
SBC	Gumstix	Kontron	Kontron
Camera	Point Gray	Point Gray	Kinect
Weight	700 g	1230 g	1925 g
Flight Time	15 min	10 min	6 min
Onboard Processing	✗	✓	✓
Camera Orientation	down	forward	forward

Table A.4: A summary of the 'wasp' quadrotor configurations used for visual flight control experiments.

[14] such as for forensics, post-experiment re-analysis, and hardware-in-the-loop testing using stored data.

[15] with an Intel 2 Duo CPU (E6850, 3 GHz and 7200 rpm SATA2 hard disk.

[16] `http://www.coker.com.au/bonnie++/`

[17] `http://www.tux.org/~mayer/linux/bmark.html`

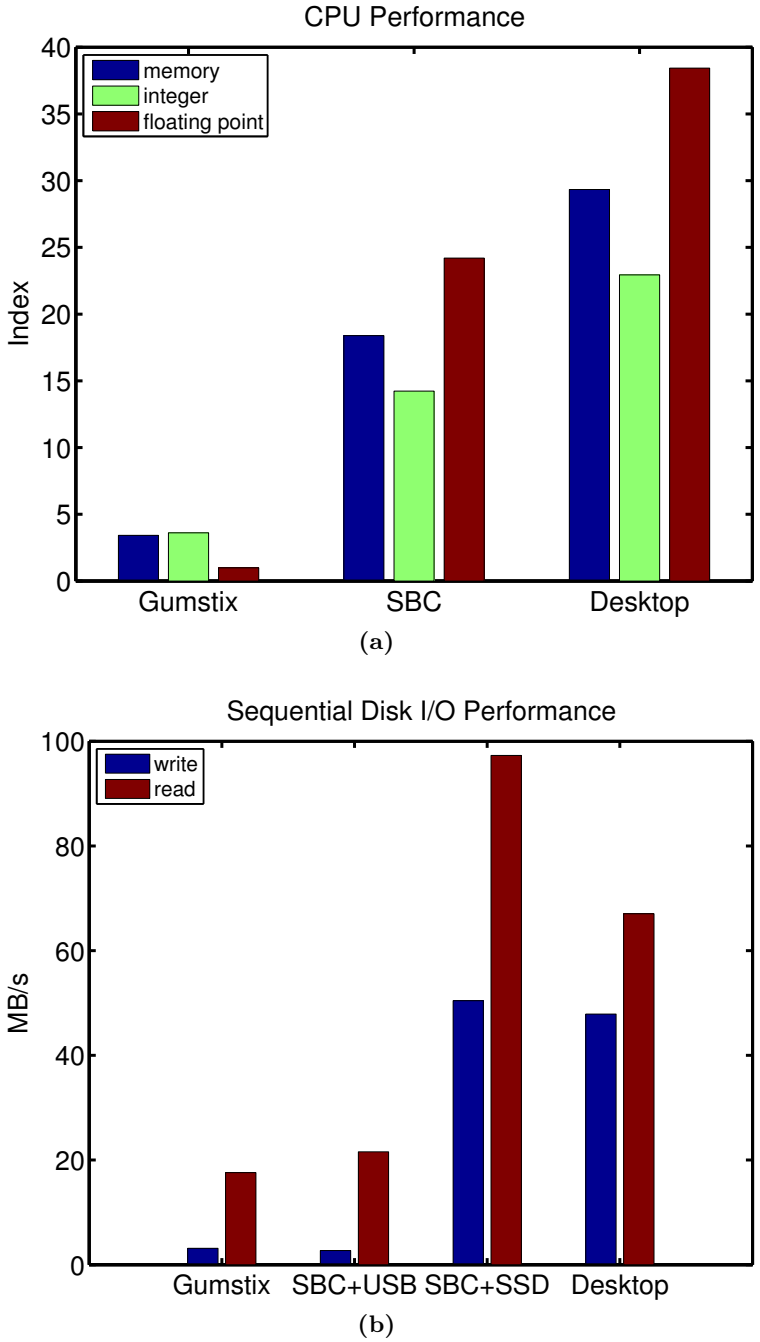


Figure A.11: A performance comparison of onboard and offboard computer systems. (a) Computational performance; memory bandwidth, integer and floating point numerical operations. The Index is relative to the 'reference' nbench system. (b) Disk performance for sequential reads and writes. Sequential block operations were chosen as this workload resembles reading and writing many medium sized images to disk. The SBC was tested against two storage types; a USB 'memory stick' and a solid-state-disk. The Gumstix disk was a standard SD card.

Wasp Gumstix

Figure A.12: Wasp Gumstix quadrotor configuration. The airframe has a total flying weight of 700 g and typical flight time of 15 minutes.

Figure A.12 shows the Wasp Gumstix configuration. The flight control system the ARM7 microprocessor based system described in Appendix A.2.1). A separate vision processing system, including camera and lens, is implemented on a Gumstix Overo SBC. The vision system communicates to the flight control via a USB interface.

Wasp Kontron

Figure A.10 shows the 'wasp' Kontron configuration. The flight control system is the ARM7 construction of Appendix A.2.1. The altitude estimator has been aided with a SONAR based rangefinder.

The notable features of this configuration is that it features the custom designed SBC module featuring a Kontron[18] microETXexpress-PC. This SBC is described in Appendix A.3.4 and had sufficient power for realtime computer vision based flight. This configuration was utilised extensively in the final year of this research.

The 'wasp' Kontron Kinect configuration was similar, only replacing the Point Gray camera for a Microsoft Kinect, as visible in Figure A.13.

ETH Quadrotor System

While at ETH Zürich, before the Pixhawk quadrotor was completed, I used the Pelican quadrotor from Ascending technologies for experimental work. The Pelican is a research focused design with a protected internal cage for mounting electronics, and capability to lift 500 g payload. As ordered, the Pelican features an Intel Atom single board computer, however, this was replaced with the more powerful Kontron SBC (Appendix A.3.4).

[18] `http://de.kontron.com`

(a) **(b)**

Figure A.13: The 'wasp' Kontron quadrotor configurations. (a) With a forward facing Microsoft Kinect. (b) With a forward facing firefly MV.

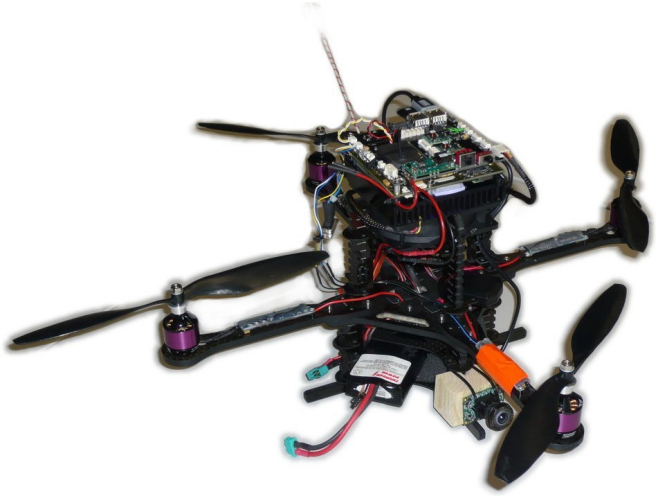


Figure A.14: The ETH quadrotor with Kontron SBC (top).

A single, forward facing, Point Grey Firefly MV USB camera is fitted with a wide angle, 180° lens. This captures 640 × 480 pixel images at 30 Hz from the quadrotor helicopter.

A.5 THIRD PARTY QUADROTOR PLATFORMS

Multiple universities and commercial companies are currently researching visual flight control, often using quadrotor helicopters as part of that research. The following list includes notable contributors to this field. It is instructive to consider each groups technical solutions to visual flight control challenges. Technical terms not explained below (and where appropriate), similarities and differences with the 'wasp' system, will be further expanded in a comparison (section A.5.1).

Pixhawk The Pixhawk[19] *cheetah* quadrotor is a custom quadrotor platform, featuring

[19] `http://pixhawk.ethz.ch/micro_air_vehicle/quadrotor/cheetah`

a bare-metal attitude controller and IMU, Core2Duo single board computer, and supporting software. Visual flight control algorithms run onboard.

CityFlyer The CityFlyer[20] project uses the Asctec *pelican* quadrotor for real time attitude control, and custom higher level control processes integrate with the system using the ROS framework.

GRASP The GRASP[21] at University of Pennsylvania uses the Asctec *hummingbird* quadrotor frame and a custom controller for attitude stabilization. Visual control is performed off-board, indirectly through the use of a Vicon[22] system.

Universitat Tubingen The Universitat Tubingen[23] autonomous flying robots research project uses the Asctec *hummingbird* quadrotor, and visual control is performed onboard using a microcontroller and low resolution infrared tracking via a Nintendo wii camera.

sFly The sFly[24] project at ETH Zürich focusses on visual flight control and coordination of multiple robots. The quadrotor platform is provided by Asctec. Visual flight control occurs onboard using the same SBC as described in Pixhawk.

Ascending Technologies (Asctec) Asctec[25] is a commercial company providing a number of large quadrotors for the research arena. The *pelican* and *hummingbird* models are frequently seen in literature as the Asctec provided attitude controller is robust and the real time telemetry available is well suited to integration with visual flight control systems, for either onboard or offboard processing.

MIT The robust robotics group[26] at MIT uses an Asctec *pelican* quadrotor and a distributed off-board vision systemBachrach et al. [2010].

Draganflyer One of the oldest commercially available quadrotor companies, Draganflyer[27] no longer offers research focused platforms and as such is not seen in research much anymore.

Microdrones Microdrones[28] also offer a number of commercial quadrotor systems, but do not offer a current open source API, nor a research focused model.

[20] `http://robotics.ccny.cuny.edu/blog/node/20`
[21] `http://www.grasp.upenn.edu/`
[22] Vicon is a motion capture system that is capable of providing real time localisation in 6DOF of multiple bodies at 200 Hz
[23] `http://www.ra.cs.uni-tuebingen.de/forschung/flyingRobots/welcome_e.html`
[24] `http://projects.asl.ethz.ch/sfly/doku.php`
[25] `http://www.asctec.de/`
[26] `http://groups.csail.mit.edu/rrg/research.html`
[27] `http://www.draganfly.com/`
[28] `http://www.microdrones.com/`

Microcopter MikroKopter[29] is the largest open-source quadrotor system and provides another excellent basis for research. The typical suite of Mikrokopter software includes an autopilot with attitude and GPS position modes, and groundstation software.

Paparazzi The paparazzi project is an established system hardware and software, featuring autopilot implementations and groundstation software for fixed wing, and more recently through the 'booz' project, quadrotor helicopters too. It does not offer visual flight control facilities. 'Wasp' was originally derived from paparazzi code in 2008. The current paparazzi project quadrotor is the 'Lisa', which is based on the improved STM32 microcontroller. A commercial version of the 'Lisa' system is available as the qudshow[30].

Other Open Source Projects There are many other open-source quadrotors, often based on arduino (Atmel ATMEGA series microcontroller). Systems such as the 'ardupilot', while cheap, in my experience do not have robust attitude control systems.

A.5.1 Comparison With Other Systems

The Pixhawk and MIT quadrotor systems also utilise an intermediate message description language, but extend the concept further by defaulting to peer-to-peer or broadcast-subscription transport models. This is more able to facilitate distributed computation on multiple nodes (such as different computers running different vision algorithms), but it is debatable whether mandating such a transport makes things simpler in the case of a single onboard computer.

The GRASP and CityFlyer projects utilise ROS[31]. ROS provides numerous building blocks implemented atop of a standardised message description and transport layer. This allows the use of tested highly functional blocks such as SLAM, laser scanners, groundstation software, and much more. ROS is a robust and healthy open source project, but is of an order of magnitude more complex[32] and capable than the 'wasp' system.

[29] `http://www.mikrokopter.de/`
[30] `http://thequadshot.com/`
[31] `http://www.ros.org/wiki/`
[32] In terms of lines of code, levels of abstraction and time to learn

Appendix B

QUADROTOR MODELLING AND CONTROL

In order to perform simulated flight experiments I implemented the quadrotor model described by Bouabdallah et al. [2004a], Bresciani [2008]. This chapter describes the implementation, and methods for determining the necessary coefficients for simulation. I use the standard aerodynamic notation [Etkin and Reid, 1995] which defines the symbols of Table B.1.

Symbol	Description	Unit
x, y, z	position, axis (body frame)	m
$\dot{x}, \dot{y}, \dot{z}$	velocity in body frame	$\mathrm{m\,s^{-1}}$
$\ddot{x}, \ddot{y}, \ddot{z}$	acceleration in body frame	$\mathrm{m\,s^{-2}}$
θ, ϕ, ψ	rotation; pitch, roll, yaw	rad
$\dot{\theta}, \dot{\phi}, \dot{\psi}$	rotational velocity	$\mathrm{rad\,s^{-1}}$
$\ddot{\theta}, \ddot{\phi}, \ddot{\psi}$	rotation acceleration	$\mathrm{rad\,s^{-2}}$
Ω	propeller speed	$\mathrm{rad\,s^{-1}}$
I_x, I_y, I_z	body inertia	$\mathrm{N\,m\,s^2}$
J	motor and propeller inertia	$\mathrm{N\,m\,s^2}$
$\boldsymbol{\tau}$	torque on airframe body	N m
b	thrust factor	$\mathrm{N\,s^2}$
d	drag factor	$\mathrm{N\,m\,s^2}$
l	lever arm	m
K_e	motor back EMF constant	$\mathrm{V\,s\,rad^{-1}}$
K_m	motor torque constant	$\mathrm{N\,m\,A^{-1}}$
R	motor internal resistance	Ω
τ_d	motor load	N m
v	motor input voltage	V

Table B.1: Aerodynamic symbols.

B.1 QUADROTOR MODEL

Recall from Section 4.1.2 that the dynamics of a generic 6 DOF rigid-body can be specified as

$$\begin{bmatrix} m\mathrm{I} & 0 \\ 0 & \mathrm{I} \end{bmatrix} \begin{bmatrix} \dot{\mathbf{V}} \\ \dot{\boldsymbol{\omega}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\omega} \times m\mathbf{V} \\ \boldsymbol{\omega} \times \mathrm{I}\boldsymbol{\omega} \end{bmatrix} = \begin{bmatrix} \mathbf{F} \\ \boldsymbol{\tau} \end{bmatrix}. \tag{B.1}$$

The quadrotor accelerates[1] according to the following sequence of equations,

$$\begin{cases} \ddot{X} &= (\sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi)\dfrac{U_1}{m} \\ \ddot{Y} &= (-\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi)\dfrac{U_1}{m} \\ \ddot{Z} &= -g + (\cos\theta\cos\phi)\dfrac{U_1}{m} \\ \ddot{\phi} &= \dot{\theta}\dot{\psi}\dfrac{I_y - I_z}{I_x} - \dfrac{J_r}{I_x}\dot{\theta}\Omega + \dfrac{l}{I_x}U_2 \\ \ddot{\theta} &= \dot{\phi}\dot{\psi}\dfrac{I_z - I_x}{I_y} - \dfrac{J_r}{I_y}\dot{\phi}\Omega + \dfrac{l}{I_y}U_3 \\ \ddot{\psi} &= \dot{\phi}\dot{\theta}\dfrac{I_x - I_y}{I_z} + \dfrac{U_4}{I_z} \end{cases}. \tag{B.2}$$

The system inputs are related to the propeller speeds,

$$\begin{cases} U_1 &= b(\Omega_1^2 + \Omega_2^2 + \Omega_3^2 + \Omega_4^2) \\ U_2 &= b(-\Omega_2^2 + \Omega_4^2) \\ U_3 &= b(-\Omega_1^2 + \Omega_3^2) \\ U_4 &= d(-\Omega_1^2 + \Omega_2^2 - \Omega_3^2 + \Omega_4^2) \\ \Omega &= -\Omega_1 + \Omega_2 - \Omega_3 + \Omega_4 \end{cases}, \tag{B.3}$$

driven by DC motors (subscript $_m$) whose dynamics may be approximated by

$$J\frac{\mathrm{d}\omega_m}{\mathrm{d}t} = -\frac{K_e K_m}{R}\omega_m - \tau_d + \frac{K_m}{R}v. \tag{B.4}$$

Introducing a propeller model (subscript $_p$), and combining K_e and K_m (they have the same value, albeit different units due to their physical coupling and the electrical and mechanical power balance) (B.4) becomes

$$J\dot{\omega}_p = -\frac{K_m^2}{R}\omega_p - d\omega_p^2 + \frac{K_m}{R}v. \tag{B.5}$$

Since this equation is non-linear, for solving (control) one should linearise about a fixed

[1] for a detailed treatment see Castillo et al. [2005, ch. 3] and Bouabdallah et al. [2004a])

point (e.g., hover), ω_H. Linearising and collecting constants yields

$$\dot{\omega}_P = A_P\omega_P + B_P v + C_P, \tag{B.6}$$

where A_P [rad s^{-1}] is the linearised propeller speed coefficient, B_P [rad^2 s^{-2} V^{-1}] is the linearised input voltage coefficient, and C_P [rad^2 s^{-2}] is the linearised constant coefficient. Their values are defined as

$$A_P = -\frac{K_m^2}{JR} - \frac{2d}{J}\omega_H \tag{B.7}$$

$$B_P = \frac{K_m}{JR} \tag{B.8}$$

$$C_P = \frac{d}{J}\omega_H^2. \tag{B.9}$$

These three parameters define the dynamics of the motor system. In vector form

$$\dot{\mathbf{\Omega}} = A_P\mathbf{\Omega} + B_P\mathbf{v} + C_p. \tag{B.10}$$

Given this model and the values of the unknown coefficients b, d, J and I, one can build a control system and simulator for a quadrotor helicopter.

B.2 COEFFICIENTS DETERMINATION

First principle numerical calculations were used to estimate the rotational and body moments of inertia; J, and I. I separated the quadrotor into several basic elements and calculated their values accordingly.

- The cross-frame structure was modelled as two thin rectangular parallelepipeds (Figure B.1a).
- The electronics and batteries were modelled as a single rectangular parallelepiped.
- The four motors were modeled as solid cylinders (Figure B.1b)

B.2.1 Rotational Moment of Inertia

The rotational moment of inertia of the motor and propeller assembly is

$$J = J_p + J_m, \tag{B.11}$$

where J_p is the propeller and J_m is the motor. From Bresciani [2008], the propeller can be modelled as a flat plate,

$$J_b = \frac{1}{12}M_B(W_B^2 + L_B^2), \tag{B.12}$$

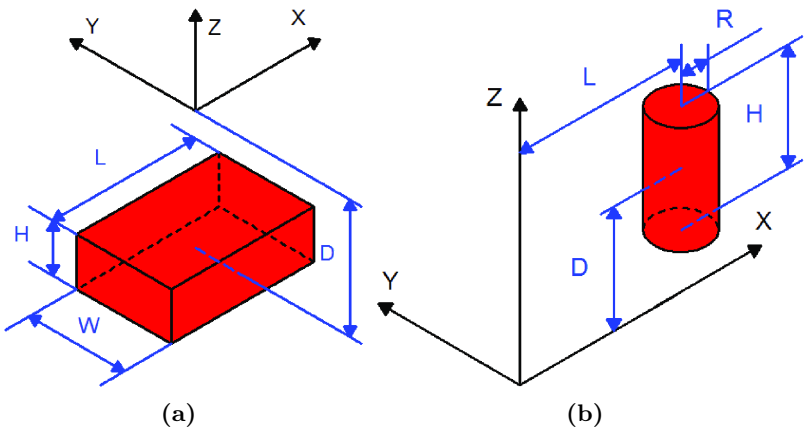


Figure B.1: Simple geometrical models for calculating moments of inertia. The centre of mass (COM) is at the centre of the marked coordinate system. (a) A cylinder, used to represent the motors. (b) A box, used to represent the electronics and battery. Figures adapted from Bresciani [2008].

and the motor as a solid cylinder,

$$J_m = \frac{1}{2} M_m R_m^2. \tag{B.13}$$

B.2.2 Body Moment of Inertia

Because the quadrotor shows a high degree of symmetry the 3×3 I matrix can be simplified to its diagonal elements; $I_X X$, $I_Y Y$, $I_Z Z$ – the moment of inertia of an object about an axis, when rotated about that axis. From Bresciani [2008], the modelled elements of the quadrotor can be represented as simple geometries (Figure B.1).

The electronics and batteries are represented by a box with mass M at a distance D from the COM. The moments of inertia of such an arrangement is

$$I_x = M\left(\frac{W^2}{12} + \frac{H^2}{12} + D^2\right),$$
$$I_y = M\left(\frac{L^2}{12} + \frac{H^2}{12} + D^2\right),$$
$$I_z = M\left(\frac{L^2}{12} + \frac{W^2}{12}\right).$$

The cross frame has been modeled as square tubing ($H = W$). The moments of

inertia of this arrangement is

$$I_x = M\left(\frac{W^2}{6} + D^2\right),$$
$$I_y = M\left(\frac{L^2}{12} + \frac{H^2}{12} + D^2\right),$$
$$I_z = M\left(\frac{L^2}{12} + \frac{W^2}{12}\right).$$

The four motors are modeled as solid cylinders. Calculation was performed for one motor (M_1) and transposed to the others due to the symmetrical nature of the vehicle. The motor moments of inertia are defined as

$$I_{M1x} = M\left(\frac{R^2}{4} + \frac{H^2}{12} + D^2\right),$$
$$I_{M1y} = M\left(\frac{R^2}{4} + \frac{H^2}{12} + L^2 + D^2\right),$$
$$I_{M1z} = M\left(\frac{R^2}{2} + L^2\right).$$

The body moments of inertia for one motor can be applied to the others

$$I_{M1x} = I_{M2y} = I_{M3x} = I_{M4y},$$
$$I_{M1y} = I_{M2x} = I_{M3y} = I_{M4x},$$
$$I_{M1z} = I_{M2z} = I_{M3z} = I_{M4z}.$$

B.2.3 Aerodynamic Coefficients

To determine the value of the thrust factor b, I collected measurements of the motor voltage and current, and propeller thrust and speed while fixed to a load cell. For the Roxxy motor and EPP1045 propeller described in Appendix A.2.3 the results are shown in Figure B.2.

The thrust factor is determined by the slope of this curve,

$$d = \frac{\mathrm{d}T}{\mathrm{d}\omega^2}. \tag{B.14}$$

Because the drag factor is a function of speed and the quadrotor is driven by an input voltage, it should be noted that this function is also linear.

The drag factor, d was extracted from the UIUC Propeller Database[2].

[2] `http://www.ae.illinois.edu/m-selig/props/propDB.html`

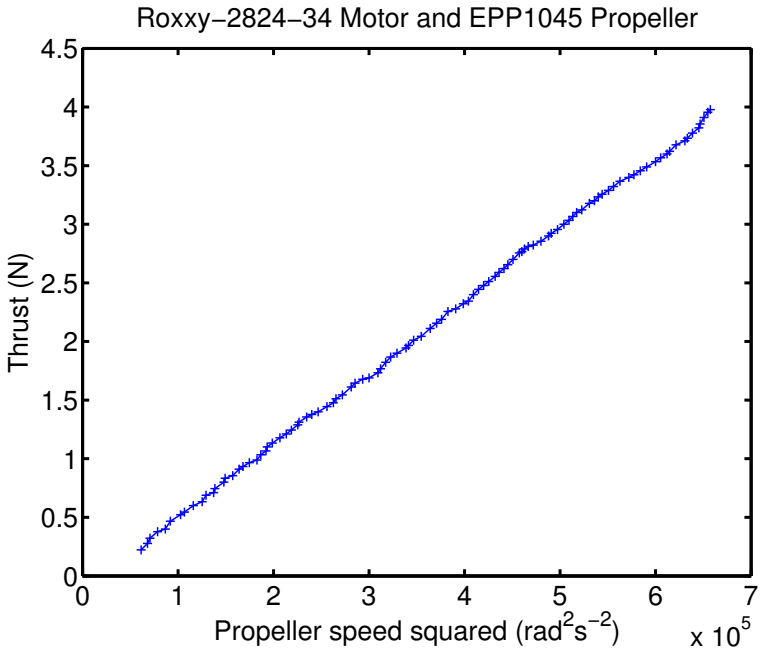


Figure B.2: Propeller and motor test; thrust at different speeds.

B.2.4 Summary of Model Coefficients

For the quadrotor configurations described in Appendix A.4 I calculated the values of Table B.2.

B.3 QUADROTOR SIMULATOR

I wrote a quadrotor simulator[3] based on qrsim[4]. The simulator is implemented in MATLAB and C.

The simulator is a direct-kinematics/direct-dynamics type. When configured with the quadrotor model coefficients[5] as calculated in the previous section, the quadrotor state is calculated by double-integrating the accelerations generated from propeller rotation, and adding these to the previous state; (B.2),(B.3),(B.4). For details of the Newton-Euler equations describing this integration refer to Etkin and Reid [1995].

Primarily the C version of the quadrotor simulator is used for hardware in the loop (HITL) and software in the loop (SITL) simulation of the quadrotor attitude controller. The MATLAB version was used for simulation of visual control strategies (such as Chapter 7). Figure B.4 shows an example trajectory from the simulator demonstrating quadrotor speed control.

[3] `https://github.com/nzjrs/simple-quadrotor-simulator`
[4] `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/Resources/`
[5] for the assembled configuration (Appendix A.4)

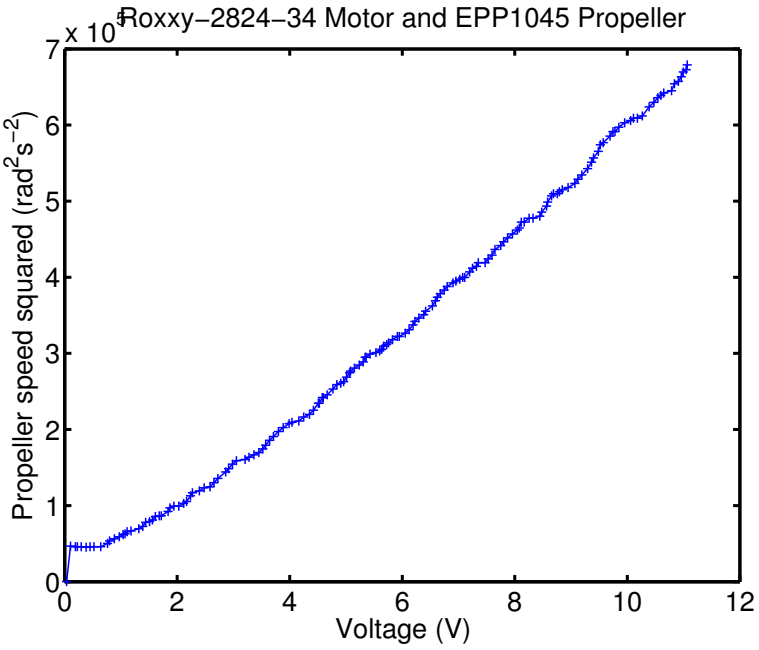


Figure B.3: Propeller and motor test; speed at different voltages.

B.4 AUTOPILOT CONTROL SYSTEM

The goal of the autopilot control system is to calculate the motor voltages, (B.3), which act to maintain the quadrotor *attitude* and altitude. The process of solving (B.2),(B.3), and (B.4) is also known as inverse-kinematics and inverse-dynamics. However, solutions to the kinematic equations are not always possible nor unique, so a number of assumptions are made [Bresciani, 2008];

1. secondary angular terms (gyroscopic effects and Coriolis forces) are assumed to be negligible due to the craft being at or near hover.
2. the angular accelerations are measured in the body fixed frame and are not necessarily equal to the Euler angle accelerations which determine the attitude in the world frame. At hover however, the transfer matrix between the two frames is near-identity; so I assume the accelerations measured can be directly related to the Euler angle accelerations. That is, I assume the gyroscopes measure $\hat{\theta} \approx q$, $\hat{\phi} \approx p$, and $\hat{\psi} \approx r$ directly.
3. I implement only *attitude* and altitude control; so equations describing X and Y position are not considered.

	I_x	I_y	I_z
onboard electronics	2.93×10^{-4}	2.93×10^{-4}	6.00×10^{-5}
SBC	4.36×10^{-3}	4.36×10^{-3}	6.67×10^{-4}
battery	7.93×10^{-5}	5.07×10^{-4}	5.15×10^{-4}
cross-frame part 1	7.92×10^{-6}	2.61×10^{-4}	2.53×10^{-4}
cross-frame part 2	2.61×10^{-4}	7.92×10^{-6}	2.53×10^{-4}
motor 1	2.76×10^{-5}	1.95×10^{-3}	1.93×10^{-3}
motor 2	1.95×10^{-3}	2.76×10^{-5}	1.93×10^{-3}
motor 3	2.76×10^{-5}	1.95×10^{-3}	1.93×10^{-3}
motor 4	1.95×10^{-3}	2.76×10^{-5}	1.93×10^{-3}
total	3.95×10^{-3}	3.95×10^{-3}	7.72×10^{-3}

(a)

	I_x	I_y	I_z
onboard electronics	2.93×10^{-4}	2.93×10^{-4}	6.00×10^{-5}
SBC	1.01×10^{-3}	1.05×10^{-3}	6.67×10^{-4}
battery	7.93×10^{-5}	5.07×10^{-4}	5.15×10^{-4}
cross-frame part 1	7.92×10^{-6}	2.61×10^{-4}	2.53×10^{-4}
cross-frame part 2	2.61×10^{-4}	7.92×10^{-6}	2.53×10^{-4}
motor 1	2.76×10^{-5}	1.95×10^{-3}	1.93×10^{-3}
motor 2	1.95×10^{-3}	2.76×10^{-5}	1.93×10^{-3}
motor 3	2.76×10^{-5}	1.95×10^{-3}	1.93×10^{-3}
motor 4	1.95×10^{-3}	2.76×10^{-5}	1.93×10^{-3}
total	1.66×10^{-3}	2.12×10^{-3}	1.15×10^{-3}

(b)

Table B.2: Model coefficients per quadrotor configuration. (a) The 'wasp' Kontron configuration (Appendix A.4) (b) The 'wasp' Gumstix configuration (Appendix A.4)

Subsequently, (B.2) is reduced to [Bouabdallah et al., 2004b],

$$\begin{cases} \ddot{Z} &= -g + (\cos\theta\cos\phi)\dfrac{U_1}{m} \\ \ddot{\phi} &= \dfrac{U_2}{I_x} \\ \ddot{\theta} &= \dfrac{U_3}{I_y} \\ \ddot{\psi} &= \dfrac{U_4}{I_z} \end{cases} . \tag{B.15}$$

Solving (B.15) can be depicted as four components, Figure B.5.

Table B.3 defines the symbols used in the following sections.

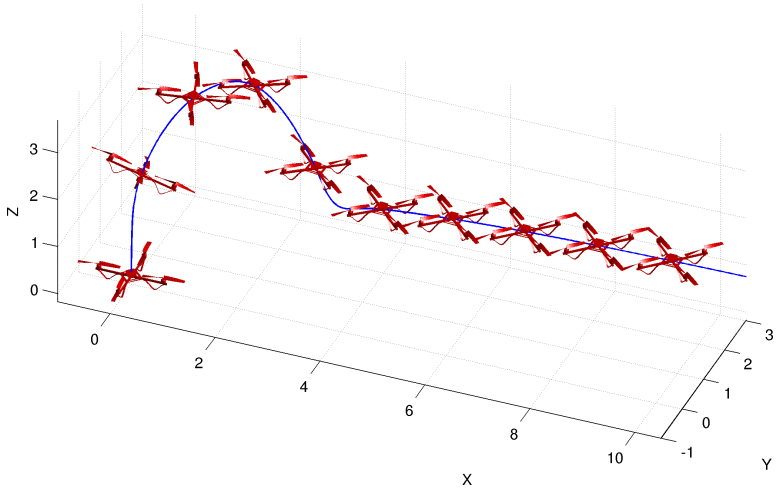


Figure B.4: Example output from quadrotor simulator.

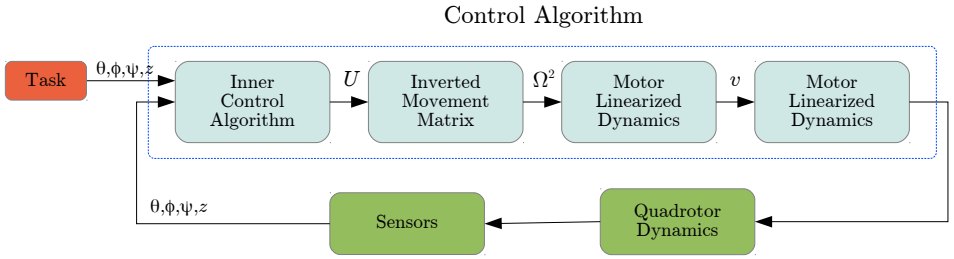


Figure B.5: Example quadrotor control system functional diagram.

B.4.1 PID Control

Bouabdallah et al. [2004b] showed the four cases of (B.15) can each be regulated by proportional integral derivative (PID) controllers when the quadrotor is near hover conditions. Figure B.6 shows the structure of a PID controller.

The PID controller is described by

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{d}{dt} e(t), \tag{B.16}$$

where u is a generic control variable, e is the instantaneous error between the task setpoint r and the plant output y. $e(\tau)$ is the accumulated instantaneous error over time τ. K_p is the proportional coefficient, K_i is the integral coefficient and K_d is the derivative coefficient.

B.4.1.1 Yaw Control

There are two implementations for yaw, ψ, control on 'wasp'; one for manual flight, and the other for autonomous flight with heading commands provided by a higher level controller. The manual flight mode is designed to be similar in feel to control of a

Symbol	Description	Unit
X, Y, Z	position (earth frame)	m
p, q, r	angular velocity about x, y, z (body axis)	$\mathrm{rad\,s^{-1}}$
u, v, w	velocity in x, y, z (body axis)	$\mathrm{m\,s^{-}1}$
$\dot{u}, \dot{v}, \dot{w}$	acceleration in x, y, z (body axis)	$\mathrm{m\,s^{-}2}$

Table B.3: Aerodynamic symbols of measured quantities.

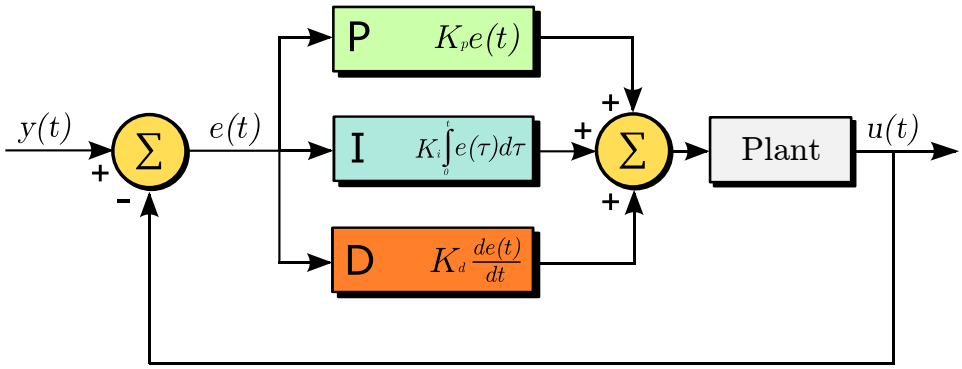


Figure B.6: The structure of PID control.

traditional RC helicopter. This is implemented as a proportional-only PID controller

$$U_4 = K_{py}(u_{\dot{y}} - r), \tag{B.17}$$

where $u_{\dot{y}}$ is the yaw-rate setpoint and r is the measured yaw-rate from the yaw-axis gyroscope. Recall that in the near-hover assumption (item 2, Appendix B.4) I assume the inertial sensors measure in the body frame and $\dot{\psi} \approx r$.

In the autonomous flight case, yaw control is implemented using a proportional-integral PID controller

$$\begin{aligned} e_y &= u_y - \hat{\psi} \\ U_4 &= K_{py} e_y + K_{iy} \int_0^t e(\tau)\, d\tau, \end{aligned} \tag{B.18}$$

where e_y is the yaw-error and $\hat{\psi}$ is the estimated yaw (see Appendix B.4.2).

B.4.1.2 Pitch and Roll Control

Pitch, u_p, and roll, u_r, control are implemented using two nested proportional-only PID controllers [Ghadiok et al., 2011]; an attitude controller that maps the angle setpoint, $u_{p,r}$, to a rate setpoint, $u_{\dot{p},\dot{r}}$ for the rate controller that in turn calculates $U_{3,2}$ to control

the propeller speed. For the pitch case (roll is identical, substitute appropriately),

$$u_{\dot{p}} = K_{\dot{p}}(u_p - \hat{\theta}) \tag{B.19}$$

$$U_3 = K_p(u_{\dot{p}} - \dot{\theta}), \tag{B.20}$$

where u_p is the pitch angle setpoint, $\hat{\theta}$ is the estimated pitch angle, and $\hat{\theta}$ is the measured pitch-rate from the gyroscopes. This can be combined into one equation

$$\begin{aligned} U_3 &= K_p\left(K_{\dot{p}}(u_p - \hat{\theta}) - \dot{\theta}\right) \\ &= K_{p1}(u_p - \hat{\theta}) - K_{p2}\dot{\theta}, \end{aligned} \tag{B.21}$$

where K_{p1} and K_{p2} are positive control parameters. Functionally this form resembles a proportional-derivitive controller (and feedback stability is the same) but is slightly easier to tune[6]. Proportional-derivitive control has been shown as suited for *attitude* control of quadrotors before [Hoffmann et al., 2007].

B.4.1.3 Altitude Control

Altitude control is implemented using a proportional-integral controller PID controller

$$\begin{aligned} e_Z &= u_Z - \hat{Z} \\ U_1 &= K_{pZ}e_Z + K_{iZ}\int_0^t e(\tau)\,d\tau, \end{aligned} \tag{B.22}$$

where e_Z is the altitude-error in the world frame, and $\hat{Z}$ is the estimated altitude using the barometer and sonar.

B.4.2 Kalman Filtering Attitude Data

The Kalman filter is a recursive and optimal estimator of the state of a system from indirect, inaccurate and uncertain observations. To estimate state, only the state from the previous time step and the current measurement are needed (i.e., recursive). Providing the measurement noise is Gaussian the Kalman filter minimises the mean square error of the estimated parameters [Kalman, 1960] (i.e., optimal).

For the general case, and theoretical background on the filter see Welch and Bishop [2006]. This section provides the formulation of the Kalman filter as used in 'wasp'. I utilise the three two-dimensional Kalman filters; two for estimating attitude from gyroscopes and accelerometers, one for estimating heading from a gyroscope and compass, and one for filtering altitude data.

[6] both in my experience and on advice from `http://www.st.ewi.tudelft.nl/~gemund/Courses/In4073/Resources/`.

Let $\mathbf{x}$ be the state vector and $\mathbf{z}$ the measurement vector for a process; then

$$x_{k+1} = Ax_k + Bu_k$$
$$z_k = Hx_k$$

is the noise free state space representation. A is the state transition matrix, B is the control matrix for an input u_k, and H is the output matrix. The Kalman filter is a two-process; a time update step to project the next state

$$\hat{x}_k^- = A\hat{x}_{k-1} + Bu_{k-1}$$
$$P_k^- = AP_{k-1}A^\mathsf{T} + Q,$$

and a measurement update step to update the estimate with new information

$$K_k = P_k^- H^\mathsf{T}(HP_k^- H^\mathsf{T} + R)^{-1}$$
$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$
$$P_k = (I - K_k H)P_k^-.$$

P is the error covariance, Q is the process noise covariance, R is the measurement error covariance, and K is the Kalman gain.

An optimal estimate of attitude (when in near-hover conditions) can be obtained by combining measurements from the onboard accelerometers (via the gravity vector, incorrect if in an accelerating reference frame) and axis-appropriate gyroscope. Inertial sensors however, particularly gyroscopes, are prone to drift and thus this drift should be corrected [Hoffmann et al., 2007]. On each of the pitch and roll axes I use a Kalman filter to correct for this drift and calculate an estimate of attitude. The following is an explanation for the pitch estimator, the roll estimator is identical except for substituting for the axis-appropriate accelerometers and gyroscope.

A pitch attitude estimator with state vector $\mathbf{x} = [\theta, q_{\text{bias}}]^\mathsf{T}$ has measurement vector $\mathbf{u} = [q]$, state transition matrix

$$A = \begin{bmatrix} 1 & -\mathrm{d}t \\ 0 & 1 \end{bmatrix}, \tag{B.23}$$

and measurement process

$$z_k = \begin{bmatrix} 1 & 0 \end{bmatrix} \mathbf{x}_k + \text{noise} \tag{B.24}$$

That is, I can measure the angle using the gravity vector

$$z_k = \text{atan2}(\dot{u}, \dot{w}). \tag{B.25}$$

The Kalman predict step for this system is then

$$\mathbf{x}_{k+1} = \begin{bmatrix} 1 & -\mathrm{d}t \\ 0 & 1 \end{bmatrix} \mathbf{x}_k + \begin{bmatrix} \mathrm{d}t \\ 0 \end{bmatrix} q + \text{noise} \tag{B.26}$$

$$\begin{aligned} P_k^- &= AP_{k-1}A^\mathsf{T} + Q \\ &= \begin{bmatrix} 1 & -\mathrm{d}t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\mathrm{d}t & 1 \end{bmatrix} + \begin{bmatrix} Q_a & 0 \\ 0 & Q_g \end{bmatrix} \end{aligned} \tag{B.27}$$

$$= \begin{bmatrix} P_{11} - (P_{12} + P_{21} - P_{22}\mathrm{d}t) + Q_a & P_{12} - P_{22}\mathrm{d}t \\ P_{21} - P_{22}\mathrm{d}t & P_{22} + Q_g \end{bmatrix} = \begin{bmatrix} Pe_{11} & Pe_{12} \\ Pe_{21} & Pe_{22} \end{bmatrix}. \tag{B.28}$$

P is the error covariance matrix, Q is the noise covariance matrix and the entries of P_e have been added for notational convenience. The Kalman update step with Kalman gains $K_k = [K_{11}, K_{22}]^\mathsf{T}$ is thus

$$\begin{aligned} K_k &= P_k^- H^\mathsf{T} (H P_k^- H^\mathsf{T} + R)^{-1} \\ &= \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \left(\begin{bmatrix} 1 & 0 \end{bmatrix} \begin{bmatrix} Pe_{11} & Pe_{12} \\ Pe_{21} & Pe_{22} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} + R \right)^{-1} \\ &= \frac{1}{P_{11} + R} \begin{bmatrix} Pe_{11} \\ Pe_{21} \end{bmatrix} \end{aligned} \tag{B.29}$$

$$\begin{aligned} \hat{x}_k &= \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-) \\ &= \hat{X}_k^- + K_k(\mathrm{atan2}(\dot{u}, \dot{w}) - \theta^-) \end{aligned} \tag{B.30}$$

$$P_k = \begin{bmatrix} Pe_{11}(1 - K_1) & Pe_{12}(1 - K_1) \\ Pe_{21} - K_2 Pe_{11} & Pe_{22} - K_2 Pe_{12} \end{bmatrix}. \tag{B.31}$$

The critical filter parameters, Q_a and Q_g are tuned emperically.

Appendix C

VISUAL SIMULATION

In order to quantitatively evaluate visual algorithm performance, some method should be available to allow repeated execution of the same control system in different conditions. There are two common solutions to this problem; the use of static datasets captured in the real world, and the use of a simulated virtual reality environment. In this thesis I have used both approaches.

The simulated approach involves two components; simulation of the virtual environment in which the quadrotor flies, and simulation of the dynamical model of the quadrotor vehicle. Three simulators were developed; programatically generated VRML via MATLAB, a MATLAB ray-tracer, and a modified version of the FlightGear[1] flight simulator. The static datasets are commonly used by computer vision researchers. In this section I explain in greater detail all the visual simulation tools used.

C.1 STATIC DATASETS

The computer vision community has provided a number of datasets in the last decade. Typical datasets consist of, at minimum, a sequence of images and additional sensor readings. They may be real or synthetic, while the additional sensor readings might include groundtruth data from and inertial measurement unit (IMU) or an external positioning system such as differential GPS (DGPS).

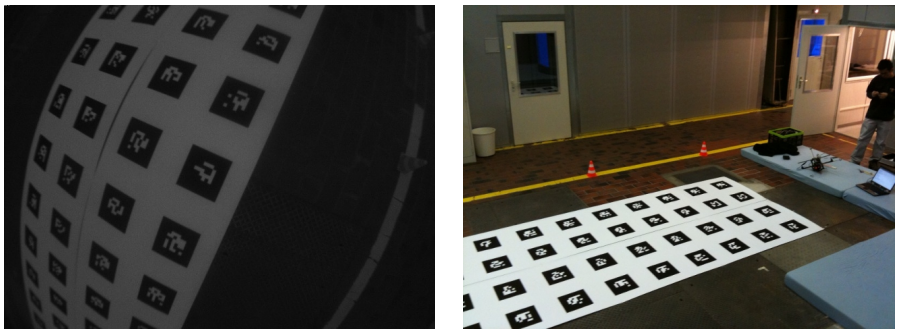
In the fields of visual control and in particular optical flow, the following datasets are in widespread use (Figure C.1);

Rawseeds

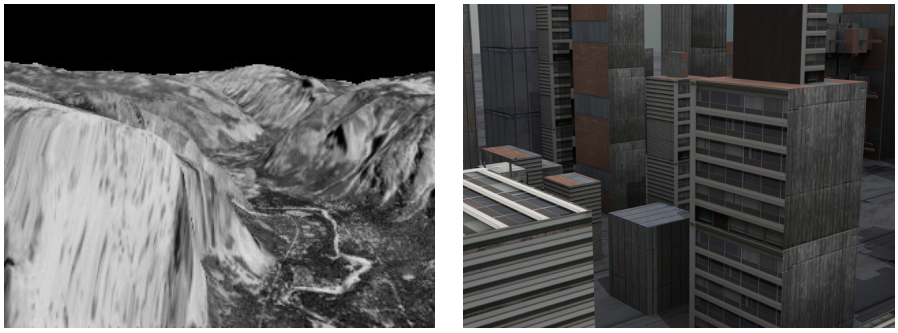
The Rawseeds [Bonarini et al., 2006, Ceriani et al., 2009] dataset describes the images and associated groundtruth for a wheeled mobile robot moving along both indoor and outdoor paths. The ground truth position information is includes the trajectory of the robot during the session, the executive drawings describ-

[1] `http://www.flightgear.org`

(a) (b)



(c) (d)



(e) (f)

Figure C.1: Sample images (a),(c), and image capture environment (b),(d) for the Rawseeds and sFly datasets respectively. Sample images from the middlebury dataset, yosemite and urban sequences, (e), (f), respectively.

ing the environments explored by the robot[2]. In addition, each dataset includes a list of positions of suitable features extracted from the executive drawings. The groundtruth data was collected by means that are separate from the sensors mounted aboard the robot, in order to be completely independent from the dataset.

sFly MAV

The sFly [Lee et al., 2010] dataset includes images and associated groundtruth position and orientation information for a flying MAV robot. Unlike the wheeled robot datasets, sFly uses a flying MAV which give image sequences with much greater degree of movement between frames.

Radish

The Radish[3] dataset is another image sequence of a wheeled robot with accompanying groundtruth data for position and heading.

Middlebury optical flow

The Middlebury [Baker et al., 2007] dataset features groundtruth optical flow measurements for a wide variety of natural and synthetic scenes. This is the eminent dataset for evaluating optical flow algorithms, but sees little use outside of this area. The length of each scene is typically short (¡ 20 frames).

The main limitation of the aforementioned datasets is that they are static, they can only be used to test control systems in a limited fashion; by proceeding in time as the dataset was captured. In effect this only allows one to evaluate the visual algorithms and not the control system as a whole. In order to obtain a full system simulation one must simulate the dynamic model of the UAV and the environment in which is flies.

C.2 VRML USING MATLAB

For 3D modelling, multiple representation formats are available, including; virtual reality modelling language (VRML), X3D, and O3D. The former two are open-standards managed by the Web3D consortium[4] and are well specified with a diverse set products implementing them. The primary reason for choosing VRML was because it was supported in MATLAB and by a number of open-source editing and viewing software packages.

[2]While most datasets are described as providing the position of the robot, what is actually useful is the position of the image sensor. Fortunately Rawseeds and other datasets (unless specified otherwise), include the necessary transformation to compute the image sensor motion from the robot motion

[3]`http://radish.sourceforge.net/`

[4] X3D is the successor to VRML.

C.2.1 The VRML Language

VRML is a text based for three-dimentional scenes. It allows the specification of polygon vertices and edges, along with the application of surface color, UV mapped textures, shininess, and transparency over those polygons. It also allows inclusion of animation, interactivity, and programmable behaviour in the virtual environment. There are two versions of VRML, VRML97 (also known as VRML2.0) replaced VRML1.0. I exclusively use the VRML97 dialect.

VRML uses a scene-graph structure. The scene is represented as a tree. The leaves of the tree hold nodes such as;

- geometry nodes,
- cluster nodes to organise other nodes into groups,
- geometric transformations to groups of nodes (scale, rotate, translate),
- appearance and material nodes for changing how the encapsulating node is displayed.

A 'hello world' sample of VRML is included in listing C.1.

```
#VRML V2.0 utf8

NavigationInfo {
   headlight    TRUE
   type         "EXAMINE"
}

Shape {
    geometry Box{}
    appearance Appearance {
        material Material {
            diffuseColor 0.5255 0.7216 0.0431
        }
    }
}
```

Listing C.1: A sample VRML world

Geometry nodes include simple geometric primitives such as **Box** {}, **Sphere** {}, **Cone** {}, **Cylinder** {}, and more complex objects like **IndexedFaceSet** {}, and **IndexedLineSet** {}.

Material notes One can style nodes using; reflection from a coloured matt surface (diffuseColor), reflection from a coloured reflective surface (specularColor, shininess), luminous objects (emissiveColor), and reflection of ambient light (ambientIntensity).

Lighting nodes

DirectionalLight Located infinitely far away. Illuminates with parallel light rays. Only affects objects below it in the scene graph.

PointLight Located at a point in space. Illuminates everything within a maximum range and which is below it in the scene graph.

Spotlight Like a PointLight, but the light is confined to a specified cone.

Viewpoint nodes Specify the position and orientation of the camera. Multiple viewpoints are allowed in the scene, and the position and orientation of the Viewpoint[5](camera) can be moved according to any of the transform nodes. By adjusting the properties of the viewpoint node one can approximate a real camera. The following nodes can be applied to a Viewpoint node:

fieldOfView specifies an angle in radians. Beginning with a 45° field of view by default, means the camera can see about 1/8 th of the full 360° panorama[6]. For example, a very zoomed-in field-of-view is about 30°, most cheap c-mount camera lenses are 45–60°, wide angle lenses are 90° and above. By convention, fish-eye or panoramic lenses are 160–180°. The widest possible field-of-view is 180°.

position specifies the user position in the coordinate system which the Viewpoint is defined.

orientation determines the direction at which the user is looking, it specifies a rotation relative to the default orientation which points along the z-axis in the negative direction.

The `fieldOfView` node, in combination with the size of captured image, can be used to match a VRML virtual camera with a physical camera's focal length, f.

$$\frac{displaywidth}{displayheight} = \frac{\tan(FOV_{horizontal}/2)}{\tan(FOV_{vertical}/2)} \tag{C.1}$$

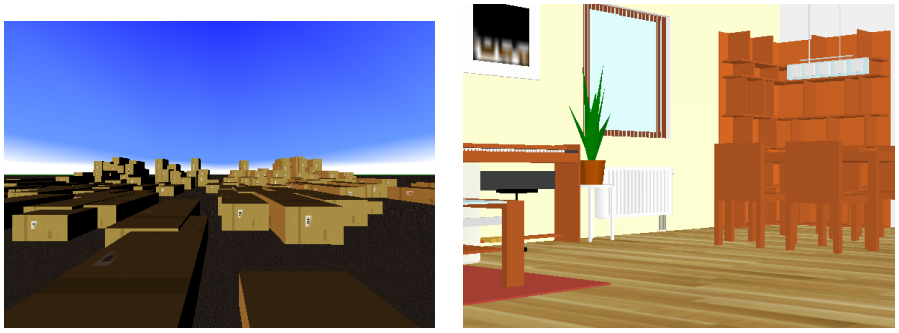
where the smaller of display width or display height determines which angle equals the `fieldOfView` (the larger angle is computed using the relationship described above). The larger angle shall not exceed π and may force the smaller angle to be less than `fieldOfView` in order to sustain the aspect ratio.

[5]In VRML 1.0, you have PerspectiveCamera and OrthogonalCamera nodes. In VRML 2.0 this was replaced by the viewpoint node and the viewpoint is always perspective.

[6] VRML expresses the `fieldOfView` parameter in radians.

C.2.2 Generating VRML For Simulation

A number of 'base' virtual environments were created authored using White Dune[7]. These contained details of the world units and scale, the sky and ground, the lighting, the definition of the virtual camera and parameters, and the inclusion of surface texture images. For Monte-Carlo simulations, such as those using in Chapter 7 the 'base' environment was amended to include additional randomly generated content. The standard MATLAB virtual reality toolbox was used to render the generated VRML and to move the virtual-camera through the environment. Sample images from these pseudo-random scenes are shown in Figure C.2.



(a) **(b)**

Figure C.2: Sample images from the randomly generated VRML for testing control strategies. (a) An outdoor city environment. (b) An indoor office environment.

C.3 RAY-TRACED DEPTH IMAGES USING MATLAB

For visual control experiments it was useful to have not just a visual representation of the scene, but true measurements of depth from the image plane to objects in that scene. Ray-tracing is a method for generating images through tracing the path of light through pixels in the image plane and simulating the effects of the beams encounters with virtual objects in that scene. As a feature of this technique, one obtains both true depth, and realistic images of the scene. Unfortunately ray-tracing is computationally expensive, so it is not suitable for rendering scenes in real-time, however, for simulation operation at slower than real-time is not a concern.

Figure C.3 shows an urban scene generated using the raytracer. The raytracer was extended and modified from the myRaytracer[8] code. It supports the following features;

- support for arbitrary geometric primitives, anything that can be expressed as $x, y, z = f(t)$.

[7] `http://vrml.cip.ica.uni-stuttgart.de/dune/`
[8] `https://sites.google.com/site/chumerin/projects/myraytracer`

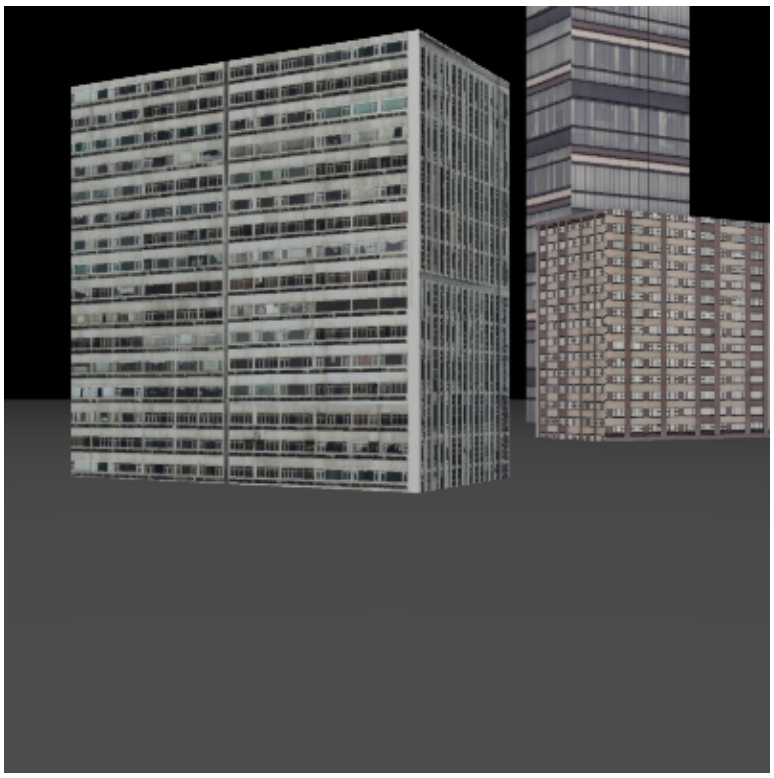


Figure C.3: Sample images from the ray-traced virtual city.

- support for arbitrary number of pinhole cameras in the environment
- support for generation of disparity maps between arbitrary camera pairs
- support for recording hit-points
- rendering speed; 45 s per frame at 300×300 px resolution with $2\times$ anti-aliasing, single camera, 3 minute per frame at 500×500 px resolution.
- cameras, light sources and objects can all be expressed as a function of time so arbitrary paths can be animated in the scene.
- expressed in the standard aerodynamic co-ordinate frame, all distances expressed in meters and angles in radians.

Appendix D

CONCEPTS IN COMPUTER VISION

This chapter includes the linear algebra formulations for the scalar camera projection equations introduced in Section 2.2. When implementing any computer vision algorithm the linear algebra formulations are always preferred owing to the greater computation efficiency with which they can be implemented.

D.1 GEOMETRIC CAMERA PARAMETERS

Using matrix notation we rearrange (2.8),

$$\begin{aligned} x_{\text{im}} &= -\frac{x}{s_x} + o_x \\ y_{\text{im}} &= -\frac{y}{s_y} + o_y \end{aligned} \tag{D.1}$$

to

$$\begin{bmatrix} x_{\text{im}} \\ y_{\text{im}} \\ 1 \end{bmatrix} = \begin{bmatrix} -1/s_x & 0 & o_x \\ 0 & -1/s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}. \tag{D.2}$$

Expressing (2.6) in matrix form. If $\mathbf{P}_c = [X_c, Y_c, Z_c]^\mathsf{T}$ and $\mathbf{P}_w = [X_w, Y_w, Z_w]^\mathsf{T}$, then

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} X_w - T_x \\ Y_w - T_y \\ Z_w - T_z \end{bmatrix}, \tag{D.3}$$

or

$$\begin{aligned} X_c &= \mathbf{R}_1^\mathsf{T}(\mathbf{P}_w - \mathbf{T}) \\ Y_c &= \mathbf{R}_2^\mathsf{T}(\mathbf{P}_w - \mathbf{T}) \\ Z_c &= \mathbf{R}_3^\mathsf{T}(\mathbf{P}_w - \mathbf{T}), \end{aligned} \tag{D.4}$$

D.1.1 Camera Model Projection Matrices

D.1.1.1 The Perspective Camera Model

Assuming $o_x = o_y = 0$ and $s_x = s_y = 1$

$$\mathrm{M}_p = \begin{bmatrix} -fr_{11} & -fr_{12} & -fr_{13} & f\mathbf{R}_1^\mathsf{T}\mathbf{T} \\ -fr_{21} & -fr_{22} & -fr_{23} & f\mathbf{R}_2^\mathsf{T}\mathbf{T} \\ r_{31} & r_{32} & r_{33} & -\mathbf{R}_3^\mathsf{T}\mathbf{T} \end{bmatrix}. \tag{D.5}$$

which can be verified

$$\begin{aligned} \mathbf{P} &= \mathrm{M}_p\mathbf{P}_w \\ &= \begin{bmatrix} -f\mathbf{R}_1^\mathsf{T} & f\mathbf{R}_1^\mathsf{T}\mathbf{T} \\ -f\mathbf{R}_2^\mathsf{T} & f\mathbf{R}_2^\mathsf{T}\mathbf{T} \\ \mathbf{R}_3^\mathsf{T} & -\mathbf{R}_3^\mathsf{T}\mathbf{T} \end{bmatrix} \begin{bmatrix} \mathbf{P}_w \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -f\mathbf{R}_1^\mathsf{T}(\mathbf{P}_w - \mathbf{T}) \\ -f\mathbf{R}_2^\mathsf{T}(\mathbf{P}_w - \mathbf{T}) \\ \mathbf{R}_3^\mathsf{T}(\mathbf{P}_w - \mathbf{T}) \end{bmatrix}, \end{aligned} \tag{D.6}$$

after homogenisation yields

$$\begin{aligned} x &= -f\frac{\mathbf{R}_1^\mathsf{T}(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^\mathsf{T}(\mathbf{P}_w - \mathbf{T})} \\ y &= -f\frac{\mathbf{R}_2^\mathsf{T}(\mathbf{P}_w - \mathbf{T})}{\mathbf{R}_3^\mathsf{T}(\mathbf{P}_w - \mathbf{T})}. \end{aligned} \tag{D.7}$$

which is identical to (2.10).

D.1.1.2 The Weak Perspective Camera Model

$\bar{P}$ is the centroid of the object (the average distance from the camera).

$$\mathrm{M}_{\mathrm{wp}} = \begin{bmatrix} -fr_{11} & -fr_{12} & -fr_{13} & f\mathbf{R}_1^\mathsf{T}\mathbf{T} \\ -fr_{21} & -fr_{22} & -fr_{23} & f\mathbf{R}_2^\mathsf{T}\mathbf{T} \\ 0 & 0 & 0 & \mathbf{R}_3^\mathsf{T}(\bar{P} - \mathbf{T}) \end{bmatrix}. \tag{D.8}$$

which can be verified

$$\begin{aligned}\mathbf{P} &= \mathrm{M}_{\mathrm{wp}}\mathbf{P}_w \\ &= \begin{bmatrix} -f\mathbf{R}_1^\mathsf{T} & f\mathbf{R}_1^\mathsf{T}\mathbf{T} \\ -f\mathbf{R}_2^\mathsf{T} & f\mathbf{R}_2^\mathsf{T}\mathbf{T} \\ 0 & \mathbf{R}_3^\mathsf{T}(\bar{P}-\mathbf{T}) \end{bmatrix} \begin{bmatrix} \mathbf{P}_w \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} -f\mathbf{R}_1^\mathsf{T}(\mathbf{P}_w-\mathbf{T}) \\ -f\mathbf{R}_2^\mathsf{T}(\mathbf{P}_w-\mathbf{T}) \\ \mathbf{R}_3^\mathsf{T}(\bar{P}-\mathbf{T}) \end{bmatrix}, \end{aligned} \tag{D.9}$$

after homogenisation yields

$$\begin{aligned} x &= -f\frac{\mathbf{R}_1^\mathsf{T}(\mathbf{P}_w-\mathbf{T})}{\mathbf{R}_3^\mathsf{T}(\bar{P}-\mathbf{T})} \\ y &= -f\frac{\mathbf{R}_2^\mathsf{T}(\mathbf{P}_w-\mathbf{T})}{\mathbf{R}_3^\mathsf{T}(\bar{P}-\mathbf{T})}. \end{aligned} \tag{D.10}$$

The Affine Camera Model The affine camera model has a number of properties different to other representations;

- the entries of the projection matrix are completely unconstrained
- the affine model does not appear to correspond to any physical camera
- leads to simple equations and appealing geometric properties
- does not preserve angles but does preserve parallelism

This yields the projection matrix

$$\mathrm{M}_a = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ 0 & 0 & 0 & a_{34} \end{bmatrix}. \tag{D.11}$$

While not used here, affine projections are numerically simpler and used in some structure from motion algorithms.

D.2 IMAGE DISTORTION DUE TO OPTICS

Radial distortions are a result of the lens shape, whereas tangential distortions are the result of the alignment of optical elements and the assembly of the camera as a whole.

Radial distortion is 0 at the optical center and increases toward the periphary. This distortion can be characterised by the first few terms of a Taylor series expansion

around $r = 0$ [Hartley and Zisserman, 2004, ch. 7]. The location of a distorted point (x_d, y_d) will be rescaled by;

$$\begin{aligned} x_{\text{im}} &= x_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{\text{im}} &= y_d(1 + k_1 r^2 + k_2 r^4 + k_3 r^6), \end{aligned} \tag{D.12}$$

where (x_d, y_d) are the coordinates of the distorted points ($r^2 = x_d^2 + y_d^2$). k_1, k_2 and k_3 are intrinsic parameters. too but will not be considered here.

Tangential distortion is characterized by an additional two parameters, p_1 and p_2;

$$\begin{aligned} x_{\text{im}} &= x_d + \left(2p_1 y_d + p_2(r^2 + 2x_d^2)\right) \\ y_{\text{im}} &= y_d + \left(p_1(r^2 + 2y_d^2) + 2p_2 x_d)\right). \end{aligned} \tag{D.13}$$

D.3 CUSTOMIZING THE LOG-POLAR TRANSFORM

If one is not interested in the data-reduction properties of the log-polar transform; instead requiring only the rotation and scale invariance, the recommended approach [Peters II et al., 1996] is to compute M so that spatial extrema in I map to radial extrema in I^*. This is done as follows. Begin by writing r from (5.5) as a function of ρ;

$$r = e^{k\rho} - 1, \tag{D.14}$$

where

$$k = \frac{\log r_{\max} + 1}{\rho_{\max}}. \tag{D.15}$$

Then, the parameter $M = 1/k$ and

$$\rho = \frac{1}{k} \log (r + 1). \tag{D.16}$$

For a detailed treatment of the log-polar transform see Peters II et al. [1996], Peters II [2000].

BIBLIOGRAPHY

K. R. T. Aires, A. M. Santana, and A. A. D. Medeiros. Optical flow using color information: preliminary results. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 1607–1611, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-753-7. doi: 10.1145/1363686.1364064.

A. A. Argyros, D. P. Tsakiris, and C. Groyer. Biomimetic centering behavior [mobile robots with panoramic sensors]. *Robotics Automation Magazine, IEEE*, 11(4):21–30, dec. 2004. ISSN 1070-9932. doi: 10.1109/MRA.2004.1371612.

A. Bachrach, R. He, and N. Roy. Autonomous flight in unknown indoor environments. *International Journal of Micro Air Vehicles*, 1(4):217, 2010. doi: 10.1260/175682909790291492.

T. Bailey and H. Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, 2006.

S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. J. Black, and R. Szeliski. A database and evaluation methodology for optical flow. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pages 1–8, 14-21 2007. doi: 10.1109/ICCV.2007.4408903.

P. Baraldi, E. De Micheli, and S. Uras. Motion and depth from optical flow. In *Proceedings of the 5th Alvey Vision Conference*, pages 205–208, Reading, UK, 1989.

A. Barron and M. V. Srinivasan. Visual regulation of ground speed and headwind compensation in freely flying honey bees (apis mellifera l.). *Journal of Experimental Biology*, 209(5):978–984, 2006. doi: 10.1242/jeb.02085.

J. L. Barron, D. J. Fleet, and S. S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43, 1994. doi: 10.1007/BF01420984.

H. G. Barrow and J. M. Tenenbaum. Interpreting line drawings as three-dimensional surfaces. *AI*, 17(1-3):75–116, August 1981.

C. Barrows, G. Neely and K. Miller. Optic flow sensors for mav navigation. *Fixed and Flapping Wing Aerodynamics for Micro Air Vehicle Applications*, 195:557–574, 2001.

G. L. Barrows, J. S. Chahl, and M. V. Srinivasan. Biomimetic visual sensing and flight control. In *Proceeding of the Bristol UAV Conf*, pages 159–168, 2002. doi: 10.1.1.125.9786.

S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433–466, 1995. ISSN 0360-0300. doi: 10.1145/212094.212141.

W. Belmans, T. Schaeps, and B. Jansen. Pseudo automatic camera extrinsic estimation using 3d hough transform. In R. Magjarevic, J. Sloten, P. Verdonck, M. Nyssen, and J. Haueisen, editors, *4th European Conference of the International Federation for Medical and Biological Engineering*, pages 567–570. Springer Berlin Heidelberg, 2009. doi: 10.1007/978-3-540-89208-3_135.

W. Bi, Z. Chen, L. Zhang, and Y. Xing. Fast detection of 3d planes by a single slice detector helical ct. In *Nuclear Science Symposium Conference Record (NSS/MIC), 2009 IEEE*, pages 954–955, 242009-nov.1 2009. doi: 10.1109/NSSMIC.2009.5402465.

H. R. Blackwell. Contrast thresholds of the human eye. *J. Opt. Soc. Am.*, 36(11): 624–632, Nov 1946. doi: 10.1364/JOSA.36.000624.

A. Bonarini, W. Burgard, G. Fontana, M. Matteucci, D. G. Sorrenti, and J. D. Tardos. Rawseeds: Robotics advancement through web-publishing of sensorial and elaborated extensive data sets. In *In proceedings of IROS'06 Workshop on Benchmarks in Robotics Research*, 2006.

D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter. A data structure for the 3d hough transform for plane detection. In *Proceedings of the 7th IFAC Symposium on Intelligent Autonomous Vehicles (IAV '10)*, Lecce, Italy, 2010.

D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Res.*, 2(2):32:1–32:13, mar 2011. ISSN 2092-6731. doi: 10.1007/3DRes.02(2011)3.

A. Borst. How do flies land? *BioScience*, 40(4):292–299, 1990. ISSN 00063568.

A. Borst. Fly visual interneurons responsive to image expansion. *Zoologische Jahrbücher (Physiologie)*, (95):305–313, 1991.

A. Borst and S. Bahde. Visual information processing in the fly's landing system. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 163(2):167–173, 1988. ISSN 0340-7594. doi: 10.1007/BF00612426.

A. Borst, J. Haag, and D. F. Reiff. Fly motion vision. *Annual Review of Neuroscience*, 33(1):49–70, 2010. doi: 10.1146/annurev-neuro-060909-153155.

S. Bouabdallah, P. Murrieri, and R. Siegwart. Design and control of an indoor micro quadrotor. volume 5, pages 4393–4398, apr. 2004a. doi: 10.1109/ROBOT.2004.1302409.

S. Bouabdallah, A. Noth, and R. Siegwart. Pid vs lq control techniques applied to an indoor micro quadrotor. volume 3, pages 2451–2456, sep. 2004b. doi: 10.1109/IROS.2004.1389776.

T. Bresciani. Modelling, identification and control of a quadrotor helicopter. Master's thesis, Lund University, Lund, Sweden, 2008.

H. J. Callow, M. P. Hayes, and P. T. Gough. Noncoherent autofocus of single-receiver broad-band synthetic aperture sonar imagery. volume 1, pages 157–162, 2001. doi: 10.1109/OCEANS.2001.968698.

T. Camus. Calculating time-to-contact using real-time quantized optical flow. Technical Report 14, Max-Planck-Institut fuer biologische Kybernetik, Arbeitsgruppe Buelthoff, 1995.

T. Camus, D. Coombs, M. Herman, and T.-H. Hong. Real-time single-workstation obstacle avoidance using only wide-field flow divergence. In *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*, volume 3, pages 323–330, Aug 1996. doi: 10.1109/ICPR.1996.546964.

J. Canny. A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-8(6):679–698, nov. 1986. ISSN 0162-8828. doi: 10.1109/TPAMI.1986.4767851.

P. Castillo, R. Lozano, and A. E. Dzul. *Modelling and control of mini-flying machines*. Advances in industrial control. Springer, 2005. ISBN 9781852339579. URL `http://books.google.co.nz/books?id=-1-UXnzDaxoC`.

K. Celik, S.-J. Chung, M. Clausman, and A. K. Somani. Biologically inspired monocular vision based navigation and mapping in GPS-denied environments. In *Proceedings of the AIAA Aerospace Conference*, Seattle, Washington, 2009a.

K. Celik, S.-J. Chung, M. Clausman, and A. K. Somani. Monocular vision SLAM for indoor aerial vehicles. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1566–1573, oct. 2009b. doi: 10.1109/IROS.2009.5354050.

S. Ceriani, G. Fontana, A. Giusti, D. Marzorati, M. Matteucci, D. Migliore, D. Rizzi, D. G. Sorrenti, and P. Taddei. Rawseeds ground truth collection systems for indoor self-localization and mapping. *Autonomous Robots*, 27(4):353–371, 2009. doi: 10.1007/s10514-009-9156-5.

R. F. Chapman. *The insects: structure and function*. Cambridge University Press, 4 edition edition, 1998. ISBN 9780521578905.

T. S. Collett and L. I. K. Harkness. Depth vision in animals. In D. Ingle, M. A. Goodale, and R. J. W. Mansfield, editors, *Analysis of visual behavior*, pages 111–176. MIT Press, Cambridge, MA, 1982. ISBN 9780262090223.

D. Coombs, M. Herman, T.-H. Hong, and M. Nashman. Real-time obstacle avoidance using central flow divergence, and peripheral flow. *Robotics and Automation, IEEE Transactions on*, 14(1):49–59, Feb 1998. ISSN 1042-296X. doi: 10.1109/70.660840.

H.-J. Dahmen, M. O. Franz, and H. G. Krapp. Extracting egomotion from optic flow: Limits of accuracy and neural matched filters. In J. M. Zanker and J. Zeil, editors, *Motion Vision - Computational, Neural, and Ecological Constraints*. Springer Verlag, Berlin Heidelberg New York, 2001.

G. C. H. E. de Croon, C. De Wagter, B. D. W. Remes, and R. Ruijsink. Sky segmentation approach to obstacle avoidance. In *Aerospace Conference, 2011 IEEE*, pages 1–16, march 2011. doi: 10.1109/AERO.2011.5747529.

R. Deklerck, B. Jansen, X. L. Yao, and J. Cornelis. Automated estimation of 3d camera extrinsic parameters for the monitoring of physical activity of elderly patients. In R. Magjarevic, P. D. Bamidis, and N. Pallikarakis, editors, *XII Mediterranean Conference on Medical and Biological Engineering and Computing 2010*, volume 29, pages 699–702. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-13039-7_176.

M. H. Dickinson. Haltere-mediated equilibrium reflexes of the fruit fly, drosophila melanogaster. *Philos. Trans. R. Soc. Lond., B, Biol. Sci.*, 354(1385):903–16, 1999. doi: 10.1098/rstb.1999.0442.

M. H. Dickinson, F.-O. Lehmann, and S. P. Sane. Wing rotation and the aerodynamic basis of insect flight. *Science*, 284(5422):1954–1960, 1999. doi: 10.1126/science.284.5422.1954.

Y. Ding, X. Ping, M. Hu, and D. Wang. Range image segmentation based on randomized hough transform. *Pattern Recognition Letters*, 26(13):2033–2041, 2005. ISSN 0167-8655. doi: 10.1016/j.patrec.2005.02.007.

L. Dittmar, W. Stürzl, E. Baird, N. Boeddeker, and M. Egelhaaf. Goal seeking in honeybees: matching of optic flow snapshots? *Journal of Experimental Biology*, 213 (17):2913–2923, 2010. doi: 10.1242/jeb.043737.

J. K. Douglass and N. J. Strausfeld. Visual motion-detection circuits in flies: Parallel direction- and non-direction-sensitive pathways between the medulla and lobula plate. *The Journal of Neuroscience*, 16(15):4551–4562, 1996.

A. Drouin and A. K. H. Warmers. Waypoint navigation, stabilization and sensor drift compensation of vtol mav. page 2008, 2008.

R. Dudley. *The biomechanics of insect flight: form, function, evolution.* Princeton University Press, 2000. ISBN 9780691044309.

B. J. Duistermars, D. M. Chow, M. Condro, and M. A. Frye. The spatial, temporal and contrast properties of expansion and rotation flight optomotor responses in drosophila. *J Exp Biol*, 210(18):3218–3227, 2007. doi: 10.1242/jeb.007807.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *Robotics Automation Magazine, IEEE*, 13(2):99–110, june 2006. ISSN 1070-9932. doi: 10.1109/MRA.2006.1638022.

M. Egelhaaf and A. Borst. Motion computation and visual orientation in flies. *Comp. Biochem. Physiol. Comp. Physiol.*, 104(4):659–73, 1993.

M. Egelhaaf, A. Borst, and W. Reichardt. Computational structure of a biological motion-detection system as revealed by local detector analysis in the fly's nervous system. *J. Opt. Soc. Am. A*, 6(7):1070–1087, Jul 1989. doi: 10.1364/JOSAA.6.001070.

H. Esch and J. Burns. Distance estimation by foraging honeybees. *Journal of Experimental Biology*, 199(1):155–62, 1996.

H. E. Esch and J. E. Burns. Honeybees use optic flow to measure the distance of a food source. *Naturwissenschaften*, 82(1):38–40, 1995. ISSN 0028-1042. doi: 10.1007/BF01167870.

B. Etkin and L. D. Reid. *Dynamics of Flight: Stability and Control.* Wiley, 3rd edition, 1995.

S. M. Ettinger, M. C. Nechyba, P. G. Ifju, and M. Waszak. Vision-guided flight stability and control for micro air vehicles. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2134–2140, 2002. doi: 10.1109/IRDS.2002.1041582.

S. Fields and M. Johnston. Cell biology. whither model organism research? *Science*, 307(5717):1885–6, 2005. doi: 10.1126/science.1108872.

J. R. Fienup. Phase error correction by shear averaging. In N. Falmouth, editor, *Signal Recovery and Synthesis III*, pages 134–137. Optical Society of America, June 1989.

J. R. Fienup. Detecting moving targets in sar imagery by focusing. *Aerospace and Electronic Systems, IEEE Transactions on*, 37(3):794–809, July 2001. ISSN 0018-9251. doi: 10.1109/7.953237.

D. Fleet and Y. Weiss. Optical flow estimation. In N. Paragios, Y. Chen, and O. Faugeras, editors, *Mathematical models for Computer Vision: The Handbook*, pages 239–258. Springer, 2005.

D. J. Fleet and A. D. Jepson. Stability of phase information. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(12):1253–1268, dec. 1993. ISSN 0162-8828. doi: 10.1109/34.250844.

D. Floreano, J.-C. Zufferey, M. V. Srinivasan, and C. Ellington. *Flying Insects and Robots*. Springer, Berlin, 2009. ISBN 978-3-540-89392-9.

H. Foroosh, J. B. Zerubia, and M. Berthod. Extension of phase correlation to subpixel registration. *Image Processing, IEEE Transactions on*, 11(3):188–200, mar. 2002. ISSN 1057-7149. doi: 10.1109/83.988953.

N. Franceschini, F. Ruffier, and J. Serres. A bio-inspired flying robot sheds light on insect piloting abilities. *Curr. Biol.*, 17(4):329–35, 2007. doi: 10.1016/j.cub.2006.12.032.

M. Franz and H. Krapp. Wide-field, motion-sensitive neurons and matched filters for optic flow fields. *Biological Cybernetics*, (83):185–197, 2000.

M. Frigo and S. G. Johnson. The design and implementation of fftw3. *Proc. of the IEEE*, 93(2):216–231, 2005. Special issue on "Program Generation, Optimization, and Platform Adaptation".

K. Frisch. *The dance language and orientation of bees*. Belknap Press of Harvard University Press, 1967.

M. A. Frye and M. H. Dickinson. A signature of salience in the drosophila brain. *Nature neuroscience*, 6(6):544–6, 2003. Copyright Nature Publishing Group Jun 2003; Jun 2003; English; 544-6; 230494120; NANEFN; New York; Frye, Mark A; Dickinson, Michael H; 1008672541; 17849061; 68927; NNRS; 10.1038/nn0603-544; 12771957; NTPGNNRSnn0603-544.

M. A. Frye and M. H. Dickinson. Motor output reflects the linear superposition of visual and olfactory inputs in drosophila. *Journal of Experimental Biology*, 207(1): 123–131, 2004. doi: 10.1242/jeb.00725.

P. K. Gerke, J. Langevoort, S. Lagarde, L. Bax, T. Grootswagers, R. J. Drenth, V. Slieker, L. Vuurpijl, P. Haselager, I. G. Sprinkhuizen-Kuyper, M. Otterlo, and G. C. H. E. d. Croon. Biomav: bio-inspired intelligence for autonomous flight. In *Proceedings of the International Micro Air Vehicle conference and competitions, 12-15 September (IMAV 2011)*, 2011.

V. Ghadiok, J. Goldin, and W. Ren. Autonomous indoor aerial gripping using a quadrotor. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 4645–4651, sept. 2011. doi: 10.1109/IROS.2011.6094690.

J. J. Gibson. *The ecological approach to visual perception.* Resources for ecological psychology. Lawrence Erlbaum Associates, 1986. ISBN 9780898599596.

K. G. Götz. The optomotor equilibrium of the drosophila navigation system. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 99(3):187–210, 1975. ISSN 0340-7594. doi: 10.1007/BF00613835.

W. E. Green, P. Y. Oh, K. Sevcik, and G. Barrows. Autonomous landing for indoor flying robots using optic flow. In *in ASME International Mechanical Engineering Congress and Exposition*, pages 1347–1352, 2003.

W. E. Green, P. Y. Oh, and G. Barrows. Flying insect inspired vision for autonomous aerial robot maneuvers in near-earth environments. *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, 3:2347–2352, 2004. ISSN 1050-4729. doi: 10.1109/ROBOT.2004.1307412.

R. Guissin and S. Ullman. Direct computation of the focus of expansion from velocity field measurements. In *Visual Motion, 1991., Proceedings of the IEEE Workshop on*, pages 146–155, Oct 1991. doi: 10.1109/WVM.1991.212776.

R. R. Harrison. *An Analog VLSI Motion Sensor Based on the Fly Visual System.* PhD thesis, California Institute of Technology, 2000.

R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.

B. Hassenstein and W. Reichardt. Systemtheoretische analyse der zeit-, reihenfolgen- und vorzeichenauswertung bei der bewegungsperzeption des rüsselkäfers chlorophanus. *Z. Naturforsch*, (11):513—524, 1956.

K. Hausen. The decoding of retinal image flow in insects. In F. Miles and J. Wallman, editors, *Visual Motion and Its Role in the Stabilization of Gaze*, pages 203–235. Elsevier, Amsterdam, 1993.

K. Hausen and M. Egelhaaf. Neural mechanisms of visual course control in insects. In D. Stavenga and R. Hardie, editors, *Facets of Vision.* Springer-Verlag, Berlin/Heidelberg, 1989.

M. Heisenberg and R. Wolf. The sensory-motor link in motion-dependent flight control of flies. *Rev Oculomot Res*, 5:265–83, 1993.

R. Hengstenberg. Gaze control in the blowfly calliphora: a multisensory, two-stage integration process. *Seminars in Neuroscience*, 3(1):19–29, 1991. ISSN 1044-5765. doi: 10.1016/1044-5765(91)90063-T.

S. Herculano-Houzel. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 3(0), 2009. ISSN 1662-5161. doi: 10.3389/neuro.09.031.2009.

G. Hoffmann, D. G. Rajnarayan, and S. L. Waslander. The stanford testbed of autonomous rotorcraft for multi agent control (STARMAC). In *Proceedings of the 23rd Digital Avionics Systems Conference*, 2004.

G. M. Hoffmann, H. Huang, S. L. Waslander, and C. J. Tomlin. Quadrotor helicopter flight dynamics and control theory and experiments. In *AIAA Guidance, Navigation and Control Conference*, Hilton Head, South Carolina, aug 2007. AIAA.

B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

B. K. P. Horn, Y. Fang, and I. Masaki. Time to contact relative to a planar surface. In *Intelligent Vehicles Symposium, 2007 IEEE*, pages 68–74, June 2007. doi: 10.1109/IVS.2007.4290093.

P. Hough. Method and means for recognizing complex patterns. US Patent 3069654, 1962.

S. Hrabar and G. S. Sukhatme. A comparison of two camera configurations for optic-flow based navigation of a uav through urban canyons. volume 3, pages 2673–2680, Sept.-2 Oct. 2004. doi: 10.1109/IROS.2004.1389812.

S. Hrabar, G. S. Sukhatme, P. Corke, K. Usher, and J. Roberts. Combined optic-flow and stereo-based navigation of urban canyons for a uav. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3309–3316, aug. 2005. doi: 10.1109/IROS.2005.1544998.

K. A. Johnson, M. P. Hayes, and P. T. Gough. A method for estimating the sub-wavelength sway of a sonar towfish. *Oceanic Engineering, IEEE Journal of*, 20(4): 258–267, oct. 1995. ISSN 0364-9059. doi: 10.1109/48.468240.

R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal Of Basic Engineering*, 82(Series D):35–45, 1960.

H. Kälviäinen, P. Hirvonen, L. Xu, and E. Oja. Probabilistic and non-probabilistic hough transforms: overview and comparisons. *Image and Vision Computing*, 13(4): 239–252, 1995. ISSN 0262-8856. doi: 10.1016/0262-8856(95)99713-B.

R. Kern, J. H. van Hateren, C. Michaelis, J. P. Lindemann, and M. Egelhaaf. Function of a fly motion-sensitive neuron matches eye movements during free flight. *PLoS Biol*, 3(6):–171, 05 2005. doi: 10.1371/journal.pbio.0030171.

N. E. A. Khalid, S. A. Ahmad, N. M. Noor, A. F. A. Fadzil, and M. N. Taib. Analysis of parallel multicore performance on sobel edge detector. In *Proceedings of the 15th WSEAS international conference on Computers*, pages 313–318, Stevens Point, Wisconsin, USA, 2011. World Scientific and Engineering Academy and Society (WSEAS). ISBN 978-1-61804-019-0.

W. Kirchner and M. Srinivasan. Freely flying honeybees use image motion to estimate object distance. *Naturwissenschaften*, 76(6):281–282, 1989. ISSN 0028-1042. 10.1007/BF00368643.

K. Kirschfeld. Course control and tracking: orientation through image stabilization. *EXS*, 84:67–93, 1997.

G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In *Proc. Sixth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, Nara, Japan, November 2007.

M. J. Klowden. *Physiological systems in insects*. Elsevier/Academic Press, 2007. ISBN 9780123694935.

J. J. Koenderink and A. J. Doorn. Facts on optic flow. *Biological Cybernetics*, 56(4): 247–254, 1987. ISSN 0340-1200. doi: 10.1007/BF00365219.

H. G. Krapp. Neuronal matched filters for optic flow processing in flying insects. *Int. Rev. Neurobiol.*, 44:93–120, 2000.

H. G. Krapp and R. Hengstenberg. Estimation of self-motion by optic flow processing in single visual interneurons. *Nature*, 384(6608):463–6, 1996. doi: 10.1038/384463a0.

H. G. Krapp and M. Wicklein. Central processing of visual information in insects. In *The Senses: A Comprehensive Reference*, volume 1, pages 131–203. Elsevier, Oxford, United Kingdom, 2008.

H. G. Krapp, B. Hengstenberg, and R. Hengstenberg. Dendritic structure and receptive-field organization of optic flow processing interneurons in the fly. *Journal of Neurophysiology*, 79(4):1902–1917, 1998.

C. Kuglin and D. Hines. The phase correlation image alignment method. September 1975.

H. Lalazar and E. Vaadia. Neural basis of sensorimotor learning: modifying internal models. *Curr. Opin. Neurobiol.*, 18(6):573–81, 2008. doi: 10.1016/j.conb.2008.11.003.

M. F. Land. Visual acuity in insects. *Annu. Rev. Entomol.*, 42:147–77, 1997. doi: 10.1146/annurev.ento.42.1.147.

M. F. Land. The optical structures of animal eyes. *Curr. Biol.*, 15(9):R319–23, 2005. doi: 10.1016/j.cub.2005.04.041.

M. F. Land and D. E. Nilsson. *Animal eyes*. Oxford animal biology series. Oxford University Press, 2002. ISBN 9780198575641.

D. N. Lee. A theory of visual control of braking based on information about time-to-collision. *Perception*, 5(4):437–59, 1976.

G. Lee, M. Achtelik, F. Fraundorfer, M. Pollefeys, and R. Siegwart. A benchmarking tool for mav visual pose estimation. Singapore, December 2010.

M. Lehrer and M. V. Srinivasan. Object detection by honeybees: Why do they land on edges? *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 173(1):23–32, 1993. ISSN 0340-7594. doi: 10.1007/BF00209615.

M. Lehrer, M. V. Srinivasan, and S. W. Zhang. Visual edge detection in the honeybee and its chromatic properties. *Proceedings of the Royal Society of London. B. Biological Sciences*, 238(1293):321–330, 1990. doi: 10.1098/rspb.1990.0002.

J. J. Leonard and H. F. Durrant-Whyte. Simultaneous map building and localization for an autonomous mobile robot. In *Proceedings IROS91, IEEE/RSJ International Workshop on Intelligent Robots and Systems*, pages 1442–1447, nov 1991. doi: 10.1109/IROS.1991.174711.

S.-J. Lim, I.-M. Ahn, and D.-J. Kang. Detection of local plane areas based on stereo range data for safe driving of mobile robot. In *Control, Automation and Systems, 2008. ICCAS 2008. International Conference on*, pages 2167–2170, oct. 2008. doi: 10.1109/ICCAS.2008.4694457.

J. Lopez, M. Markel, N. Siddiqi, and G. Gebert. Performance of passive ranging from image flow. In *Proceedings of the 2003 International Conference on Image Processing (ICIP 2003)*, pages 929–932, Sept. 2003. doi: 10.1109/ICIP.2003.1247116.

B. D. Lucas. *Generalized Image Matching by the Method of Differences*. PhD thesis, Robotics Institute, Carnegie Mellon University, July 1984.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. pages 674–679, 1981.

G. Maimon, A. D. Straw, and M. H. Dickinson. A simple vision-based algorithm for decision making in flying drosophila. *Current Biology*, 18(6):464–470, 2008. doi: 10.1016/j.cub.2008.02.054.

R. Manzotti, A. Gasteratos, G. Metta, and G. Sandini. Disparity estimation on log-polar images and vergence control. *Computer Vision and Image Understanding*, 83 (2):97–117, 2001. ISSN 1077-3142. doi: 10.1006/cviu.2001.0924.

M. Markel, J. Lopez, G. Gebert, and J. Evers. Vision-augmented gnc: Passive ranging from image flow. In *AIAA-2002-5026*, Monterey, California, 2002. AIAA Guidance, Navigation, and Control Conference and Exhibit, Aug. 5-8.

A. Millane, M. Hayes, and J. Stowers. Embedded linux controlled quadrotor helicopter. In *Proceedings of Electronics New Zealand Conference 2010*, pages 27–32, Hamilton, New Zealand, 2010.

L. Muratet, S. Doncieux, Y. Briere, and J.-A. Meyer. A contribution to vision-based autonomous helicopter flight in urban environments. *Robotics and Autonomous Systems*, 50(4):195–209, 2005. ISSN 0921-8890. doi: 10.1016/j.robot.2004.09.017. Biomimetic Robotics.

G. Nalbach. The halteres of the blowfly calliphora. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 173(3):293–300, 1993. ISSN 0340-7594. doi: 10.1007/BF00212693.

G. Nalbach and R. Hengstenberg. The halteres of the blowfly calliphora. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 175(6):695–708, 1994. ISSN 0340-7594. doi: 10.1007/BF00191842.

R. C. Nelson and J. Aloimonos. Finding motion parameters from spherical motion fields (or the advantages of having eyes in the back of your head). *Biological Cybernetics*, 58(4):261–273, 1988. ISSN 0340-1200. doi: 10.1007/BF00364131.

T. R. Neumann. Modeling insect compound eyes: Space-variant spherical vision. In *Proceedings of the Second International Workshop on Biologically Motivated Computer Vision*, BMCV '02, pages 360–367, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-00174-3.

T. R. Neumann and H. H. Bülthoff. Behavior-oriented vision for biomimetic flight control. In *Proceedings of the EPSRC/BBSRC International Workshop on Biologically Inspired Robotics: The Legacy of W. Grey Walter*, pages 196–203, HP Labs Bristol, UK, 2002.

T. R. Neumann, S. A. Huber, , and H. H. Bulthoff. Minimalistic approach to 3d obstacle avoidance behavior from simulated evolution. In *Proceedings of the 9th International Conference on Artificial Neural Networks*, 1997.

D.-E. Nilsson. Vision optics and evolution. *BioScience*, 39(5):298–307, 1989. ISSN 00063568.

T. A. Ofstad, C. S. Zuker, and M. B. Reiser. Visual place learning in drosophila melanogaster. *Nature*, 474(7350):204–207, Jun 2011. ISSN 0028-0836. doi: 10.1038/nature10131.

K. Okada, S. Kagami, M. Inaba, and H. Inoue. Plane segment finder: algorithm, implementation and applications. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 2120–2125, 2001. doi: 10.1109/ROBOT.2001.932920.

H. Otsuna and K. Ito. Systematic analysis of the visual projection neurons of drosophila melanogaster. i. lobula-specific pathways. *The Journal of Comparative Neurology*, 497 (6):928–958, 2006. ISSN 1096-9861. doi: 10.1002/cne.21015.

J. Overby, L. Bodum, E. Kjems, and P. M. Ilsøe. Automatic 3d building reconstruction from airborne laser scanning and cadastral data using hough transform. *Int. Arch. of Photogrammetry and Remote Sensing*, 35, part B3, 2004.

S. Perreault and P. Hebert. Median filtering in constant time. *IEEE Transactions on Image Processing*, 16(9):2389, 2007. doi: 10.1109/TIP.2007.902329.

R. A. Peters II. On the computation of the discrete log-polar transform. 2000.

R. A. Peters II, M. Bishay, and T. Rogers. On the computation of the log-polar transform. 1996.

F. Pla and M. Bober. Estimating translation/deformation motion through phase correlation. In *Proc. of the 9th International Conference on Image Analysis and Processing-Volume I*, pages 653–660, London, UK, 1997. Springer-Verlag. ISBN 3-540-63507-6.

J. Plett, A. Bahl, M. Buss, K. Kühnlenz, and A. Borst. Bio-inspired visual ego-rotation sensor for mavs. *Biological Cybernetics*, 106(1):51–63, 2012. ISSN 0340-1200. URL `http://dx.doi.org/10.1007/s00422-012-0478-6`. 10.1007/s00422-012-0478-6.

T. Poggio and W. Reichardt. Visual control of orientation behaviour in the fly: Part ii. towards the underlying neural interactions. *Quarterly Reviews of Biophysics*, 9(03): 377–438, 1976. doi: 10.1017/S0033583500002535.

T. Poggio, A. Verri, and V. Torre. Green theorems and qualitative properties of the optical flow. In *MIT AI Memo*, 1991.

G. Portelli, F. Ruffier, and N. Franceschini. Honeybees change their height to restore their optic flow. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 196(4):307–313, Apr 2010. ISSN 0340-7594. doi: 10.1007/s00359-010-0510-z.

P. Pounds, R. Mahony, P. Hynes, and J. Roberts. Design of a four-rotor aerial robot. Auckland, New Zealand, November 2002.

P. Pounds, R. Mahony, and P. Corke. Modelling and control of a quad-rotor robot. In *Proceedings Australasian Conference on Robotics and Automation 2006*. Australian Robotics and Automation Association Inc., 2006.

P. Pounds, R. Mahony, and P. Corke. Modelling and control of a large quadrotor robot. *Control Engineering Practice*, 18(7):691–699, 2010. ISSN 0967-0661. doi: 10.1016/j.conengprac.2010.02.008. Special Issue on Aerial Robotics.

W. K. Pratt. *Digital image processing: PIKS Scientific inside*. Wiley-Interscience, 2007. ISBN 9780471767770.

W. Reichardt. Movement perception in insects. In *Processing of optical data by organisms and by machines*, pages 465–93. New York: Academic Press, 1969.

W. Reichardt and H. Wenking. Optical detection and fixation of objects by fixed flying flies. *Naturwissenschaften*, 56(8):424–5, 1969.

J. R. Riley and J. L. Osborne. Flight trajectories of foraging insects: observations using harmonic radar. In I. P. Woiwood, D. R. Reynolds, and C. D. Thomas, editors, *Insect Movement: Mechanisms and Consequences. Proceedings of the Royal Entomological Society's 20th Symposium*, pages 129–157, London, UK, 2001. doi: 10.1079/9780851994567.0129.

S. M. Ross. *Stochastic processes*. John Wiley and Sons, New York, 1983.

S. Roth and M. Black. On the spatial statistics of optical flow. *International Journal of Computer Vision*, 74(1):33–50, 2007. ISSN 0920-5691. doi: 10.1007/s11263-006-0016-x.

F. Ruffier and N. Franceschini. Visually guided micro-aerial vehicle: automatic take off, terrain following, landing and wind reaction. In *Proceeding of the 2004 IEEE International Conference on Robotics and Automation*, volume 3, pages 2339–2346, 2004. doi: 10.1109/ROBOT.2004.1307411.

F. Ruffier and N. Franceschini. Aerial robot piloted in steep relief by optic flow sensors. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1266–1273, sept. 2008. doi: 10.1109/IROS.2008.4651089.

F. Ruffier and N. H. Franceschini. Optic flow regulation: the key to aircraft automatic guidance. *Robotics and Autonomous Systems*, 50(4):177–194, 2005. doi: 10.1016/j.robot.2004.09.016.

A. Sarti and S. Tubaro. Detection and characterisation of planar fractures using a 3d hough transform. *Signal Processing*, 82(9):1269–1282, 2002. ISSN 0165-1684. doi: 10.1016/S0165-1684(02)00249-9.

H. Schuppe and R. Hengstenberg. Optical properties of the ocelli of calliphora erythrocephala and their role in the dorsal light response. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 173(2):143–149, 1993. ISSN 0340-7594. doi: 10.1007/BF00192973.

J. Serres, F. Ruffier, and N. Franceschini. Two optic flow regulators for speed control and obstacle avoidance. In *Biomedical Robotics and Biomechatronics, 2006. BioRob 2006. The First IEEE/RAS-EMBS International Conference on*, pages 750–757, feb. 2006. doi: 10.1109/BIOROB.2006.1639180.

J. Serres, D. Dray, F. Ruffier, and N. Franceschini. A vision-based autopilot for a miniature air vehicle: joint speed control and lateral obstacle avoidance. *Auton. Robots*, 25(1-2):103–122, August 2008. ISSN 0929-5593. doi: 10.1007/s10514-007-9069-0.

L. Shapiro and G. Stockman. *Computer Vision*. Prentice Hall, 1st edition, 2001.

A. Sherman and M. H. Dickinson. Summation of visual and mechanosensory feedback in drosophila flight control. *Journal of Experimental Biology*, 207(1):133–142, 2004. doi: 10.1242/jeb.00731.

A. Si, M. V. Srinivasan, and S. Zhang. Honeybee navigation: properties of the visually driven 'odometer'. *Journal of Experimental Biology*, 206(8):1265–1273, 2003. doi: 10.1242/jeb.00236.

B. Siciliano and O. Khatib, editors. *Springer Handbook of Robotics*. Springer, 1st edition edition, 2008.

R. Smith, M. Self, and P. Cheeseman. A stochastic map for uncertain spatial relationships. In *Proceedings of the 4th international symposium on Robotics Research*, pages 467–474, Cambridge, MA, USA, 1988. MIT Press. ISBN 0-262-02272-9.

M. Srinivasan, S. Zhang, M. Lehrer, and T. Collett. Honeybee navigation en route to the goal: visual flight control and odometry. *Journal of Experimental Biology*, 199 (1):237–44, 1996.

M. Srinivasan, S. Zhang, and N. Bidwell. Visually mediated odometry in honeybees. *J. Exp. Biol.*, 200(Pt 19):2513–22, 1997.

M. V. Srinivasan. How insects infer range from visual motion. *Rev Oculomot Res*, 5: 139–56, 1993.

M. V. Srinivasan. Small brains, smart computations: Vision and navigation in honeybees, and applications to robotics. *International Congress Series*, 1291:30–37, 2006. ISSN 0531-5131. doi: 10.1016/j.ics.2006.01.055.

M. V. Srinivasan and G. D. Bernard. The pursuit response of the housefly and its interaction with the optomotor response. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 115(1):101–117, 1977. ISSN 0340-7594. doi: 10.1007/BF00667788.

M. V. Srinivasan and R. L. Gregory. How bees exploit optic flow: Behavioural experiments and neural models [and discussion]. *Philosophical Transactions: Biological Sciences*, 337(1281):–253, 1992. ISSN 09628436.

M. V. Srinivasan and S. Zhang. Visual motor computations in insects. *Annual Review of Neuroscience*, 27(1):679–696, 2004. doi: 10.1146/annurev.neuro.27.070203.144343.

M. V. Srinivasan and S. W. Zhang. Visual navigation in flying insects. *Int. Rev. Neurobiol.*, 44:67–92, 2000.

M. V. Srinivasan, M. Lehrer, W. H. Kirchner, and S. W. Zhang. Range perception through apparent image speed in freely flying honeybees. *Visual Neuroscience*, 6 (05):519–535, 1991. doi: 10.1017/S095252380000136X.

M. V. Srinivasan, S. Zhang, M. Altwein, and J. Tautz. Honeybee navigation: Nature and calibration of the 'odometer'. *Science*, 287(5454):851–853, 2000a. doi: 10.1126/science.287.5454.851.

M. V. Srinivasan, S. W. Zhang, J. S. Chahl, E. Barth, and S. Venkatesh. How honeybees make grazing landings on flat surfaces. *Biol Cybern*, 83(3):171–83, 2000b.

M. V. Srinivasan, S. W. Zhang, J. S. Chahl, G. Stange, and M. Garratt. An overview of insect-inspired guidance for application in ground and airborne platforms. *Proceedings of the Institution of Mechanical Engineers Part G Journal of Aerospace Engineering*, 218(6):375, 2004. doi: 10.1243/0954410042794966.

M. V. Srinivasan, S. Thurrowgood, and D. Soccol. An optical system for guidance of terrain following in uavs. *Video and Signal Based Surveillance, 2006. AVSS '06. IEEE International Conference on*, pages 51–51, 2006. doi: 10.1109/AVSS.2006.23.

S. Srinivasan. Extracting structure from optical flow using the fast error search technique. *International Journal of Computer Vision*, 37(3):203, 2000. doi: 10.1023/A:1008111923880.

G. Stange. The ocellar component of flight equilibrium control in dragonflies. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 141(3):335–347, 1981. ISSN 0340-7594. doi: 10.1007/BF00609936.

J. Stowers, A. Bainbridge-Smith, M. Hayes, and S. Mills. Optical flow for attitude estimation of a quadrotor helicopter. In *Proceedings of the European Micro Aerial Vehicle Conference*, Delft, the Netherlands, 2009.

J. Stowers, A. Bainbridge-Smith, M. Hayes, and S. Mills. Optical flow for heading estimation of a quadrotor helicopter. *International Journal of Micro Air Vehicles*, 1 (4):229, 2010a. doi: 10.1260/175682909790291474.

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Quadrotor helicopters for visual flight control. In *Proceedings of Electronics New Zealand Conference 2010*, pages 21–26, Hamilton, New Zealand, 2010b.

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Phase correlation using shear average for image registration. In *Proceedings of the 25th International Conference on Image and Vision Computing New Zealand*, 2010c.

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor. In *Proceedings of the IEEE International Conference on Mechatronics*, pages 358–362, Istanbul, Turkey, 2011a. doi: 10.1109/ICMECH.2011.5971311.

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Quadrotor helicopter flight control using hough transform and depth map from a microsoft kinect sensor. In *Proceedings of the IAPR Conference on Machine Vision Applications*, pages 352–356, Nara, Japan, 2011b.

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Beyond optical flow - biomimetic uav altitude control using horizontal edge information. In *The 5th International Conference on Automation, Robotics and Applications (ICARA 2011)*, Wellington, New Zealand, Dec 2011c.

J. Stowers, M. Hayes, and A. Bainbridge-Smith. Biologically inspired uav obstacle avoidance and control using monocular optical flow & divergence templates. In *The 5th International Conference on Automation, Robotics and Applications (ICARA 2011)*, Wellington, New Zealand, Dec 2011d.

J. Stowers, M. Hayes, and P. Smith. A statistical model for the distribution of horizontal edges in a planar scene. In *Proceedings of the 26th International Conference Image and Vision Computing New Zealand*, 2011e.

A. D. Straw, S. Lee, and M. H. Dickinson. Visual control of altitude in flying drosophila. *Current Biology*, 20(17):1550–1556, 2010. ISSN 0960-9822. doi: 10.1016/j.cub.2010.07.025.

L. F. Tammero and M. H. Dickinson. The influence of visual landscape on the free flight behavior of the fruit fly drosophila melanogaster. *J. Exp. Biol.*, 205(Pt 3): 327–43, 2002a.

L. F. Tammero and M. H. Dickinson. Collision-avoidance and landing responses are mediated by separate pathways in the fruit fly, drosophila melanogaster. *Journal of Experimental Biology*, 205(18):2785–2798, 2002b.

K. Tanaka, H. Ohtake, M. Tanaka, and H. O. Wang. Wireless vision-based stabilization of indoor microhelicopter. *Mechatronics, IEEE/ASME Transactions on*, 17(3):519–524, june 2012. ISSN 1083-4435. doi: 10.1109/TMECH.2011.2181532.

S. Thurrowgood, D. Soccol, R. Moore, D. Bland, and M. V. Srinivasan. A vision based system for attitude estimation of uavs. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5725–5730, oct. 2009. doi: 10.1109/IROS.2009.5354041.

M. Tistarelli and G. Sandini. On the advantages of polar and log-polar mapping for direct estimation of time-to-impact from optical flow. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 15(4):401–410, Apr 1993. ISSN 0162-8828. doi: 10.1109/34.206959.

C. Tomasi and T. Kanade. Detection and tracking of point features. Technical report, Carnegie Mellon University, April 1991. CMU-CS-91-132.

V. J. Traver and A. Bernardino. A review of log-polar imaging for visual perception in robotics. *Robotics and Autonomous Systems*, 58(4):378–398, 2010. ISSN 0921-8890. doi: 10.1016/j.robot.2009.10.002.

E. Trucco and A. Verri. *Introductory Techniques for 3-D Computer Vision*. Prentice Hall, 1998.

J. A. Vince. *Geometry for Computer Graphics: Formulae, Examples and Proofs*. Springer, 1 edition, 2004. ISBN 1852338342.

R. Völkel, M. Eisner, and K. J. Weible. Miniaturized imaging systems. *Microelectron. Eng.*, 67–68(1):461–472, June 2003. ISSN 0167-9317. doi: 10.1016/S0167-9317(03)00102-3.

G. Vosselman, E. Dijkman, K. W. B. Reconstruction, L. Altimetry, and H. Transform. 3d building model reconstruction from point clouds and ground plans. *Int. Arch. of Photogrammetry and Remote Sensing*, pages 37–43, 2001.

H. Wagner. Flow-field variables trigger landing in flies. *Nature*, 297(5862):147–148, May 1982. doi: 10.1038/297147a0.

B. A. Wandell. *Foundations of Vision*. Sinauer Associates, Inc., 1995.

B. Webb. Insect behaviour: Controlling flight altitude with optic flow. *Current Biology*, 17(4):–124, 2007. ISSN 0960-9822. doi: 10.1016/j.cub.2006.12.008.

M. Weckstrom, M. Juusola, and S. B. Laughlin. Presynaptic enhancement of signal transients in photoreceptor terminals in the compound eye. *Proceedings: Biological Sciences*, 250(1327):83–89, 1992. ISSN 09628452.

R. Wehner. 'matched filters' –- neural models of the external world. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 161(4):511–531, 1987. ISSN 0340-7594. doi: 10.1007/BF00603659.

S. Weiss, D. Scaramuzza, and R. Siegwart. Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

G. Welch and G. Bishop. An introduction to the kalman filter. Technical Report TR 95-041, University of North Carolina at Chapel Hill, 2006. URL `http://www.cs.unc.edu/~welch/kalman/kalmanIntro.html`.

J. S. Werner and L. M. Chalupa. *The visual neurosciences*, volume 2 of *The Visual Neurosciences*. MIT Press, 2004. ISBN 9780262033084.

G. Wolberg and S. Zokai. Robust image registration using log-polar transform. In *Proc. of the IEEE Int. Conf. Image Processing*, volume 1, pages 493–496, 2000. doi: 10.1109/ICIP.2000.901003.

L. Xu and E. Oja. Randomized hough transform (rht): basic mechanisms, algorithms, and computational complexities. *CVGIP: Image Underst.*, 57(2):131–154, March 1993. ISSN 1049-9660. doi: 10.1006/ciun.1993.1009.

L. Xu, E. Oja, and P. Kultanen. A new curve detection method: randomized hough transform (rht). *Pattern Recogn. Lett.*, 11(5):331–338, May 1990. ISSN 0167-8655. doi: 10.1016/0167-8655(90)90042-Z.

J. Yang, D. Rao, S.-J. Chung, and S. Hutchinson. Monocular vision based navigation in GPS-denied riverine environments. In *Proceedings of the AIAA Infotech@Aerospace Conference*, St. Louis, Missouri, 2011.

J. M. Zanker and J. Zeil. *Motion Vision: Computational, Neural, and Ecological Constraints*. Springer, 1st edition, 2000.

G. Zhang, M. Lei, and X. Liu. Novel template matching method with sub-pixel accuracy based on correlation and fourier-mellin transform. *Optical Engineering*, 48(5):057001, 2009. doi: 10.1117/1.3125425.

D. Ziou and S. Tabbone. Edge detection techniques - an overview. *International Journal of Pattern Recognition and Image Analysis*, 8:537–559, 1998.

J.-C. Zufferey. *Bio-inspired Vision-based Flying Robots*. PhD thesis, EPFL, Lausanne, 2005.

J.-C. Zufferey, A. Klaptocz, A. Beyeler, J.-D. Nicoud, and D. Floreano. A 10-gram vision-based flying robot. *Advanced Robotics*, 21(14):1671–1684, 2007. doi: 10.1163/156855307782227417.