Visual control of multi-rotor UAVs

Stuart Duncan

A thesis submitted in partial fulfilment
of the requirements for the degree of
Master of Engineering
in
Electrical and Computer Engineering
at the
University of Canterbury,
Christchurch, New Zealand.

2014

ABSTRACT

Recent miniaturization of computer hardware, MEMs sensors, and high energy density batteries have enabled highly capable mobile robots to become available at low cost. This has driven the rapid expansion of interest in multi-rotor unmanned aerial vehicles. Another area which has expanded simultaneously is small powerful computers, in the form of smartphones, which nearly always have a camera attached, many of which now contain a OpenCL compatible graphics processing units. By combining the results of those two developments a low-cost multi-rotor UAV can be produced with a low-power onboard computer capable of real-time computer vision. The system should also use general purpose computer vision software to facilitate a variety of experiments.

To demonstrate this I have built a quadrotor UAV based on control hardware from the Pixhawk project, and paired it with an ARM based single board computer, similar those in high-end smartphones. The quadrotor weights 980 g and has a flight time of 10 minutes. The onboard computer capable of running a pose estimation algorithm above the 10 Hz requirement for stable visual control of a quadrotor.

A feature tracking algorithm was developed for efficient pose estimation, which relaxed the requirement for outlier rejection during matching. Compared with a RANSAC-only algorithm the pose estimates were less variable with a Z-axis standard deviation 0.2 cm compared with 2.4 cm for RANSAC. Processing time per frame was also faster with tracking, with 95 % confidence that tracking would process the frame within 50 ms, while for RANSAC the 95 % confidence time was 73 ms. The onboard computer ran the algorithm with a total system load of less than 25 %. All computer vision software uses the OpenCV library for common computer vision algorithms, fulfilling the requirement for running general purpose software.

The tracking algorithm was used to demonstrate the capability of the system by performing visual servoing of the quadrotor (after manual takeoff). Response to external perturbations was poor however, requiring manual intervention to avoid crashing. This was due to poor visual controller tuning, and to variations in image acquisition and attitude estimate timing due to using free running image acquisition.

The system, and the tracking algorithm, serve as proof of concept that visual control of a quadrotor is possible using small low-power computers and general purpose computer vision software.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENT

First I would like to thank my supervisors Andrew Bainbridge-Smith and Michael Hayes for their continued help. Thanks for nudging me in the right direction, for your continued support and confidence.

I also thank my family for their unwavering support, my mother without whose resolve to remain in New Zealand I would not have even been here to take this opportunity.

And to eveyone whose belief held even when mine failed, thank you for your vote of confidence.

Chapter 1
INTRODUCTION

Unmanned aerial vehicles (UAVs) are quickly becoming a ubiquitous fixture of modern industry. They have applications in search and rescue (Doherty and Rudol, 2007, Goodrich et al., 2008), aerial photography and remote sensing (Jensen et al., 2008, Rango et al., 2009), surveillance (Rodriguez et al., 2006), and many other fields. Some specific applications include power line inspection (Li et al., 2008, 2010) and providing sensory feedback for precision agriculture (Herwitz et al., 2004, Zarco-Tejada et al., 2008, Cohen et al., 2012, Primicerio et al., 2012, Schellberg et al., 2008, Pea-Barragn et al., 2010, Zhang and Kovacs, 2012). Most UAVs require human operators, and in the case of hover-capable UAVs, manual control requires a high level of concentration and skill. Modern autopilots can assist operators by using inertial sensors for stabilization, however, visual feedback has the potential to significantly simplify operator interaction. Visual control has successfully been used to perform position hold (Meier et al., 2012, Amidi, 1996), simultaneous localization and mapping (SLAM) (Davison et al., 2007, Meier et al., 2012), and is used for other tasks such as collision avoidance (Tsalantsanis et al., 2007, Green and Oh, 2008) or odometry (Caballero et al., 2009, Achtelik et al., 2009, Smith and Dodds, 2009).

This thesis presents a multi-rotor UAV with an onboard computer for visual control. The onboard vision computer is based on a low-power ARM system on chip (SOC), which is lighter and consumes less power than the Intel based single board computers used onboard other systems (Meier et al., 2012, Stowers, 2013). A simple monocular pose estimation and visual servoing algorithm is implemented to demonstrate the capability of the system. The control software is based on the general purpose OpenCV library, showing that real-time performance is now possible from low-power hardware without using intensive hand-optimized algorithms.

1.1 MOTIVATION

Small modern UAVs offer several advantages over manned aircraft and larger UAVs. They are cheaper to build and to operate, and in the case of multi-rotor UAVs require little maintenance at all. They are safer; a crash does not necessarily endanger human life, and nearly all of them are electrically powered which mitigates the danger posed by flammable liquid fuel[1]. For multi-rotor UAVs the individual rotors are also smaller than tilt-rotor craft, which limits the rotational energy of the blades compared to tilt-rotor UAVs of the same weight (Stowers, 2013). This minimizes potential impact damage on contact with objects or with people. In some cases rotor guards (Parrot, 2013a) even allow crashing with no damage at all. Their reduced size has allowed research on a smaller scale and conveniently allows test flights to be carried out indoors. It is not surprising that a large body of work involving small multi-rotor UAVs has been generated over the last decade. This thesis aims to add to this work by demonstrating that it is now possible to run general purpose computer vision software in real time on small, cheap ARM single board computers for controlling multi-rotor UAVs in real time.

1.1.1 Autonomy Using GPS

GPS is commonly used for outdoor autonomy, however it is desirable to operate small autonomous UAVs in GPS-denied environments, particularly indoors. Many other environments lack reliable GPS signals, such as under dense forest canopies, in caves, in extraterrestrial environments (He et al., 2008), and even in built up urban environments (Zenk et al., 2011). GPS accuracy is also in the order of metres where visual algorithms can achieve position estimates with accuracies in the order of centimetres (Hightower and Borriello, 2001, Sharp et al., 2001). Visual control is therefore an attractive alternative: cameras are lightweight and low-power while providing high-resolution data from which accurate position and speed data can be used. Visual control, in the case of onboard processing, is independent of offboard systems and therefore can be used in isolated and GPS-denied environments.

1.1.2 Onboard Processing

Real time offboard processing requires a high-bandwidth low-latency data connection between the UAV and an offboard computer, and for the system to be useful the link must be wireless. The latency inherent in any communication link places stricter constraints on the performance of the offboard computer, and requires a high-power wireless transmitter onboard the UAV. The data link also limits the operating range of the UAV, and adds an additional point of failure.

Traditionally onboard processing was achieved either by using specialized hardware such as FPGAs (Fowers et al., 2007) or digital signal processing (DSP) hardware (Amidi, 1996), or by using high powered single board computers based on Intel's mobile architectures (Meier et al., 2012). The recent proliferation of high-end smartphones, however, has driven the development of small, efficient, portable computers based on

[1] Although LiPo batteries can be explosive the risk can be minimized by physical battery protection and good short-circuit protection

the ARM 'system on a chip' (SOC) architectures. Many of these are multi-core implementations and include parallel processing extensions to the ARM architecture, making them ideally suited to processing large amounts of data efficiently. Additionally, when considering visual control, smartphones nearly always have a camera attached. This has driven the development of ARM-optimized computer vision libraries such as FastCV (Qualcomm, 2013) for use in augmented reality, facial recognition, and other applications which require real-time performance from limited processing power.

These developments present an attractive alternative to offboard processing, with reduced power consumption, weight, and complexity over Intel computers which have been used in the past.

1.1.3 Multi-Rotor Craft

With the increase in mobile processing power and the availability of cheap inertial sensors and high energy density Lithium Polymer (LiPo) battery technology, multi-rotor UAVs have become popular among researchers, hobbyists, and in commercial applications. The attraction of multi-rotor craft is their simplicity: the only moving parts are the motors. Additionally their manoeuvrability, and their ability to hover, is desirable in many situations such as remote inspection. This is also convenient for performing experiments in smaller indoor environments.

1.2 RELATED WORK

Modern micro aerial vehicles (MAVs) typically weigh 0.5-1.5 kg and have payload capacities of a few hundred grams (Meier et al., 2012). Their size affords convenient and relatively safe indoor operation, this coupled with the affordability and the capabilities of modern quadrotors, has made them popular research platforms and generated a large body of research involving quadrotor UAVs. A timeline of notable progress over the last two decades is given in Figure 1.1

1.2.1 The State of the Art

Traditionally autonomous aerial vehicles have had larger payload capacities, with early UAVs weighing around 10-20 kg (Meier et al., 2012). There was some early work on visual control of tilt rotor helicopters, notably by Amidi (1996) and by Johnson et al. (1996). However there is a vast body of more recent and more relevant work based on quadrotors, it is this work that I will review here.

Most of the recent work on visual control is focused on simultaneous localization and mapping (SLAM). To facilitate high-level interaction with its environment a robot must have knowledge of its position relative to objects in its environment. SLAM algorithms attempt to construct a map of landmarks surrounding the robot while simultaneously

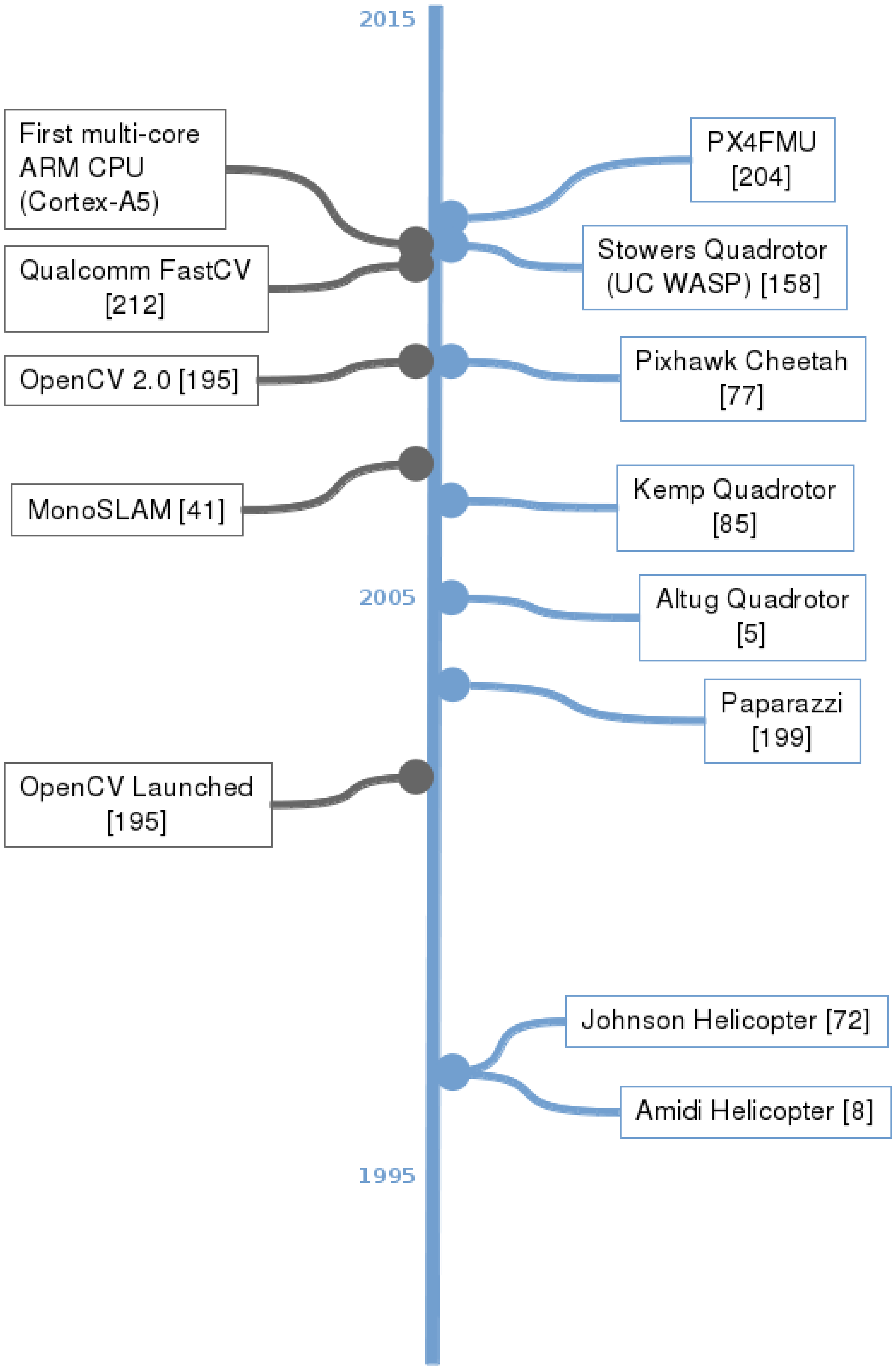


Figure 1.1: Notable developments which have driven autonomous UAV research.

tracking its pose. Complete theoretical solutions are outlined by Durrant-Whyte and Bailey (2006), so the theoretical SLAM problem is considered solved. There remain, however, many implementation challenges, most significantly scalability: SLAM maps are built using observed landmarks, which requires position estimates for each landmark. This results in a large state vector which can become intractable for large or cluttered environments. Furthermore, computing the state vector is $O(n^2)$ in the number of landmarks (Durrant-Whyte and Bailey, 2006). Nevertheless, there have been numerous successful implementations (Davison et al., 2007, Bryson and Sukkarieh, 2007, Royer et al., 2007, Artieda et al., 2009, Milford and Wyeth, 2010). Davison et al. (2007) developed the first 'pure vision' monocular SLAM algorithm, based on a sparse map of natural landmarks. Royer et al. (2007) demonstrated stereo SLAM using a three-stage algorithm where a path was recorded under manual control (stage 1), and a 3D map was generated using the recorded data (stage 2). In the third stage the quadrotor could then follow the recorded path. The system was also capable of taking paths within the mapped area differing from the one used to generate the map. SLAM has also been performed based on artificial markers; an example is the Pixhawk Cheetah system which used markers from the ARToolkit+ (ARToolkit, 2013).

Despite the popularity of SLAM, there is another class of visual control algorithm which forgoes the mapping requirement and assumes direct control of the attitude of the UAV. These algorithms typically automate simple behaviour such as maintaining a fixed position or altitude, terrain following, or collision avoidance (Stowers, 2013, Tsalantsanis et al., 2007). Collision avoidance has been demonstrated based on avoiding the centre of expansion of optical flow (Stowers, 2013). Minimizing the optical flow is a common approach to eliminate hover drift in helicopters (Meier et al., 2012, Dille et al., 2010). Another example is terrain following, that is holding a fixed altitude above the ground during a traverse, which has been demonstrated based on a statistical model of horizontal line placement (Stowers, 2013), and by using optical flow (Hérissé et al., 2010, Campbell et al., 2004) and assuming constant ground-speed. Both of these approaches have been implemented on real systems.

1.2.2 Existing Systems

Some of the earliest published work on visual control of small multi-rotor UAVs was by Altuğ et al. (2002). This system was based on an early commercial quadrotor and used an onboard and an offboard camera in conjunction with coloured markers mounted on the UAV and in the flight area. They were later able to demonstrate autonomous position hold based on pose estimation (Altuğ et al., 2003).

Work by Kemp (2006) demonstrated control based on a known 3D model of the flight area. Kemp used images from a small low-resolution onboard camera with offboard processing. To overcome the low image quality he used 2D to 3D edge matching instead of point features. Kemp was able to demonstrate stable visual servoing using this system.

One of the first successful attempts to process data onboard was by Fowers et al. (2007) who achieved real time processing by using an FPGA implementation of their control algorithm. They developed a visual approach to minimize UAV drift when hovering based on tracking Harris corners.

Blosch et al. (2010) presented one of the first implementations of visual SLAM using a quadrotor, however, the offboard processing was achieved using a USB cable for image transmission. This work was based on the AscTec Hummingbird quadrotor platform, and subsequent work by Achtelik et al. (2011) extended it to process data onboard using a bespoke onboard computer also from AscTec. Achtelik demonstrated a modified SLAM algorithm with a 10 Hz visual control loop. The system was one of the first to use the Robot Operating System (ROS, 2013) framework for robot control, and one of the first to use a general purpose computer for onboard computer vision.

Work by Williams et al. (2011) resulted in monocular visual control based on three flight modes: fast traverse, hovering, and ingress (moving through doorways or other openings). The work was tested using a recorded sequence. Williams demonstrated that pose estimates from this approach were accurate to within 0.25% of a ground truth pose estimated using a stereo imaging system.

Bills et al. (2011) used the native forwards and downwards facing cameras of the Parrot AR.Drone visually navigate previously unseen corridors, stairs, and around corners. Processing was performed offboard, with images and control commands transmitted over WiFi.

Li et al. (2011) developed a multi-robot ground-air system which used onboard processing with an AscTec quadrotor and a ground robot. Using infrared markers on the ground robot and IMU data from both robots, they were able to demonstrate pose estimation using both optical flow and visual markers.

Carrillo et al. (2011, 2012) developed a quadrotor for visual control and demonstrated hovering over a known marker using monocular vision. This work was extended to use a stereo imaging system which was able to take off, hover, and land autonomously. In both cases visual data was processed offboard.

In addition to control using onboard cameras, there has also been interest in using cameras from the perspective of an observer. The most notable developments are from the GRASP lab at the university of Pennsylvania (Univ of Penn, 2013). The GRASP flight area consists of a (5×4×3.5 m) zone monitored by a Vicon motion capture system which uses fiduciary markers mounted on the UAVs (Michael et al., 2010). This system has been used to demonstrate dynamic behaviour such as flying through a narrow slot requiring significant roll or pitch angles to allow the quadrotor through, or ‘perching’ on an inverted landing pad. GRASP has also been used to investigate the disturbances introduced when multiple quadrotors are flown in close proximity.

1.2.3 Pixhawk

The open source Pixhawk (2013) project was launched in 2009 and combined a custom high-powered single board computer with an AscTec quadrotor resulting in the Cheetah system (Meier et al., 2011, 2012). MAVLINK (Pixhawk, 2013c) and the Mavconn communication framework (Pixhawk, 2013b) were released as part of this project. Visual autonomy was demonstrated with the Cheetah autonomous quadrotor which was able to navigate using fiduciary markers from ARToolkit+ (ARToolkit, 2013).

The PX4FMU (Pixhawk, 2013a) is the second generation autopilot from Pixhawk, and was designed for the Parrot AR.Drone flight hardware. This combination provides a highly capable autopilot with low cost off-the-shelf flight hardware for a cheap and convenient research platform, albeit only suited for offboard processing.

1.3 THESIS OVERVIEW

This thesis describes a vision-capable quadrotor. Chapter 2 begins by introducing fundamental concepts of computer vision in the context of visual control of robots. Topics ranging from image formation and camera geometry to image features and optical flow are presented. Chapter 3 presents the quadrotor platform developed for the experimental work in this thesis, including a small ARM computer for onboard processing. Chapter 4 presents the implementation of the visual control software which is designed around the Mavconn communication framework and OpenCV. Chapter 5 presents visual servoing based on pose estimates from a feature tracking algorithm. The pose estimation algorithm is tested both on a recorded sequence of images, and used to demonstrate visual servoing of the quadrotor from Chapter 3. Some background theory on RANSAC and feature matching is also presented. Chapter 6 concludes this thesis with a recommendation for future system for visual research.

Chapter 2

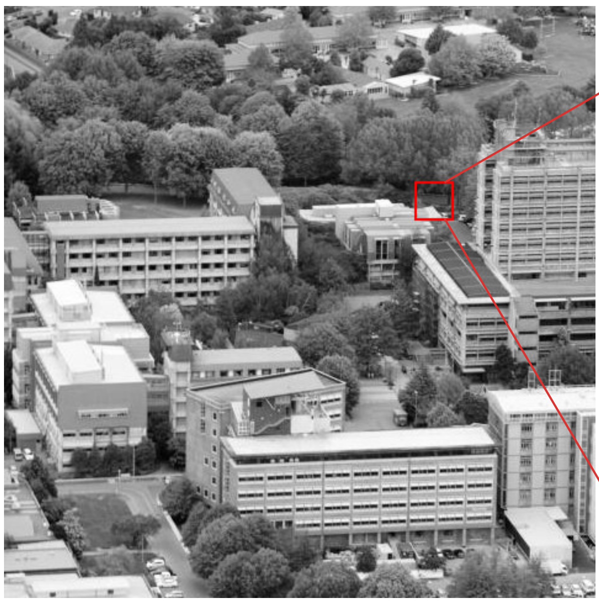
VISION AND NAVIGATION

As humans we effortlessly perceive the world as a collection of objects, surfaces, and space (Gibson, 1950). Our high level perception allows us to interact with a wide variety of objects and environments which we need not have encountered before. Human vision is also imperfect and we lack the precision of computers. Our estimates of sizes, distances, and colours are at best rough estimates. However this is sufficient to support fast and complex interactions with our environment. Computer vision seeks to give mobile robots similar perceptive abilities and enable them to interact with their surroundings in a similar manner. Computers lack any innate or learned perceptive ability that humans or other animals may possess, however, and the processing mechanics behind human vision are also poorly understood. We don't even have a precise picture of what we are trying to copy. Furthermore, the visual data available to computers consists only of arrays of noisy quantized numbers representing images which have been distorted by imperfect optical components. From this imperfect visual data we attempt to use mathematics and geometry to infer enough useful information for a robot to perform tasks such as avoiding collisions or constructing a map of an unknown area. In the context of the visual navigation of small flying robots with onboard processing there are additional constraints imposed by the limited payload capacities. Nevertheless progress has been made, as described in Chapter 1.2.

This chapter gives an overview of topics of computer vision used in visual navigation. Section 2.1 begins by describing the mathematical model of an ideal camera upon which pose estimation, stereo imaging, and other geometric operations are based. The model is then extended to include the imperfections of real cameras in Section 2.1.2. The overview of the fundamental geometric model concludes with multiple view geometry in Section 2.2. Section 2.3 then introduces the detection, description, and matching of image features, which are used to provide the point correspondences required for the methods described in Section 2.2. Section 2.4 introduces optical flow and gives some common methods used to compute flow (Section 2.4.1).

2.1 IMAGE FORMATION

Digital images are stored as arrays of numbers representing the irradiance sampled at points in a scene as illustrated by Figure 2.1. In computer vision the most common form of digital image encodes the monochromatic irradiance (called intensity images) of the scene, however, colour images are also common, and with the arrival of low-cost structured light cameras such as the Microsoft Kinect (kin, 2014) *depth* images are becoming more common. In depth images each pixel value represents the distance from the camera to the point in the scene, or the *depth* of the point. Regardless of the data representation all images are formed by sampling the irradiance of a scene over an array of sensor elements with finite areas (called pixels). This array is the only visual data available to computers.



(a)

48	57	64	44	51	78	71	48	52	74	83	67
43	50	55	41	55	74	59	46	46	70	79	51
38	50	48	44	56	54	40	47	54	64	63	56
41	57	43	56	58	38	31	47	58	55	62	47
57	71	43	52	51	38	29	39	54	47	52	40
72	75	45	40	31	21	28	42	54	53	41	44
67	56	41	27	24	27	21	24	28	43	36	26
60	68	68	57	53	54	40	42	43	52	36	35
65	61	55	54	57	64	57	66	68	66	57	46
84	75	72	72	65	71	65	61	67	67	62	50
113	91	76	77	77	74	74	78	76	80	64	52
125	107	83	81	81	81	77	80	84	88	70	53
113	109	80	79	75	86	88	83	78	85	62	55
105	106	67	66	74	79	79	85	87	91	67	57
113	105	91	64	65	97	90	80	71	69	58	63
112	112	97	82	115	127	105	110	82	74	72	104
107	117	143	189	206	202	71	80	67	78	92	106
172	178	193	225	233	226	122	51	56	61	90	74
202	206	201	224	230	225	184	46	60	71	80	57
195	190	194	213	226	231	218	102	60	72	74	74
201	206	202	205	225	231	230	175	137	176	134	121
190	200	197	196	224	224	229	217	151	191	147	104
179	192	192	199	225	230	228	225	179	126	100	54
182	184	189	198	228	219	226	209	194	195	161	63

(b)

Figure 2.1: A digital image. (a) Image rendered in greyscale, (b) the numerical values of a small patch of pixels.

Finite pixel areas of digital imaging sensors result in spatial quantization of points in images. In addition to the spatial quantization, the measured irradiance also has dynamic range limitations imposed by the sensing hardware, in some cases limited to 8-bits of precision. Quantization, and dynamic range compression introduce noise into the image. And before the light is even sampled it passes through one or more optical lenses, whose imperfections can introduce spatially varying distortions such as the barrel distortion shown in Figure 2.4. In order to model these effects and to extract useful information from an image we must start with the simple pinhole model of a camera and extend it to account for the effects mentioned above.

2.1.1 The Pinhole Model

The pinhole camera is both a physical device and a mathematical model (Hartley and Zisserman, 2004). The physical device consists of a chamber with an infinitesimal 'pinhole' in the centre of one face, through which an inverted image of the scene is projected onto the opposite face (the image plane). A lens is not required but due to the small aperture of the pinhole the light collected is insufficient to produce an images with adequate signal to noise ratios (SNR) without either a bright scene or a long integration time. The projection lines pass through the origin of the camera coordinate system, which is called the principal point. Figure 2.2 shows a geometric representation of the model. The model is similar to the physical device, but for simplicity the image plane is placed in front of the principal point. The model exhibits scale ambiguity along the projection lines: any point on the line $\bar{\mathbf{p}}\bar{\mathbf{p}}'$ is projected to the same point $\mathbf{p}$ in the image.

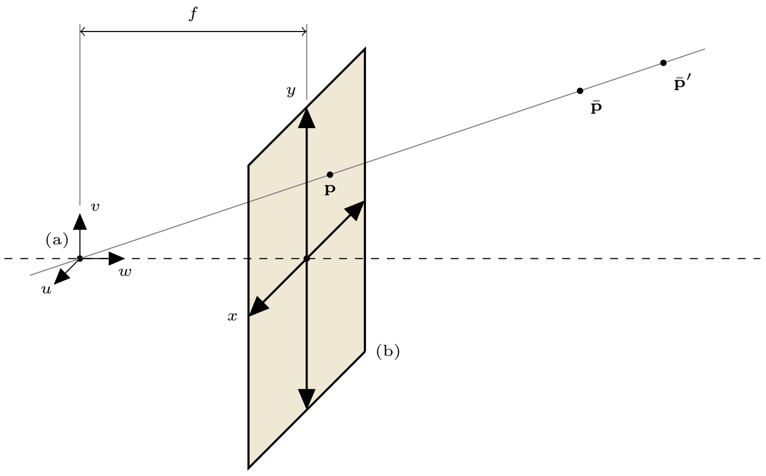


Figure 2.2: The pinhole camera model showing the position of the (a) principal point and (b) the image plane separated along along the w axis by the focal length f. $\bar{\mathbf{p}}$ $=[u,v,w]^T$ is a point in the (u,v,w) world coordinate system which is projected onto the image plane along the blue line passing through the optical centre. The imaged point $\mathbf{p}$ $=[x,y]^T$ is in the (x,y) image plane coordinate system. The model exhibits scale ambiguity: Points are unique only up to scale along the projection line so $\bar{\mathbf{p}}$ and $\bar{\mathbf{p}}'$ are both imaged at the same point $\mathbf{p}$.

Mathematically the model is simple,

$$\mathbf{p} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{w}\begin{bmatrix} u \\ v \end{bmatrix}, \tag{2.1}$$

where $\mathbf{p}$ $=[x,y]^T$ is the point in the image coordinate system and f is the focal length. $\bar{\mathbf{p}}$ $=[u,v,w]^T$ are the coordinates of the point in the camera coordinate system with the origin at the optical centre.

Because of the projective nature of the model, it is most often expressed in *homogeneous*

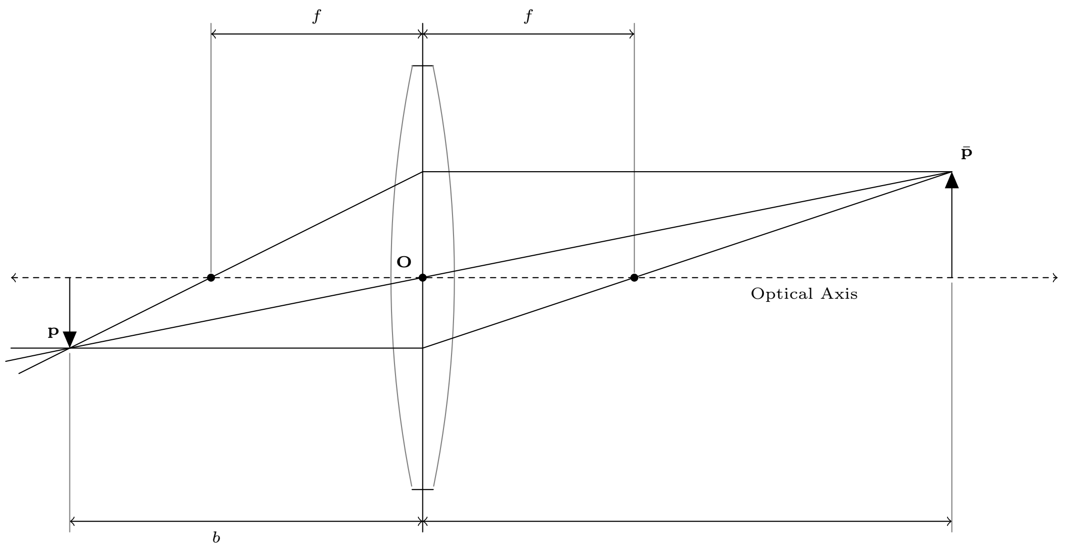


Figure 2.3: The thin lens model is an approximate model which applies to lenses whose thickness along the optical axis is negligible compared to their focal length. **O** is analogous to the principal point of the simplified model in Figure 2.2, however the camera no longer has infinite focal depth; the projection lines from $\bar{\mathbf{p}}$ diverge on either side of **p**.

coordinates (For an overview of homogeneous coordinates see Appendix A).

$$\mathbf{p} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{C}\bar{\mathbf{p}} = \begin{bmatrix} 1,0,0,0 \\ 0,1,0,0 \\ 0,0,\frac{1}{f},0 \end{bmatrix} \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \tag{2.2}$$

where **C** is the camera matrix for the ideal pinhole model. The projective representation captures the scale ambiguity of the model by expressing 2D coordinates using three elements and normalizing the coordinate so that the third element is always equal to 1. The pinhole model in this form is the basis for geometric camera operations, however, it is necessary to consider the effects of the non-ideality of real cameras.

2.1.2 Modeling Real cameras

In real cameras lenses are employed to collect more light by focusing plane waves instead of single rays. Lenses are necessary as it is impossible to produce an image with a good signal to noise ratio using the amount of light passing through an infinitesimal pinhole. A lens limits the *focal depth* of the image, as shown in Figure 2.3, and can also introduce distortions due to imperfections. In addition to lens effects, the pinhole model in (2.1) and (2.2) does not include the effects of spatial quantization or finite pixel area, or pixel skew along the x and y axes of the image plane.

In order to account for these effects and to include the position and orientation (the *pose*) of the camera in the world coordinate system the model is extended by adding three sets of parameters:

- Intrinsic parameters (Hartley and Zisserman, 2004)

  Pixel scale on each axis x_0 and y_0

  Pixel skew γ

  Focal length f

- Extrinsic parameters (Hartley and Zisserman, 2004)

  Rotation $\mathbf{R}$

  Translation $\mathbf{t}$

- Lens imperfections (distortion coefficients) (Bradski and Kaeher, 2008)

  Radial distortion (imperfect optical components)

  Tangential distortion (imperfect lens alignment)

Intrinsic and extrinsic parameters are normally expressed in matrix form and require points to be expressed in *homogeneous* coordinates.

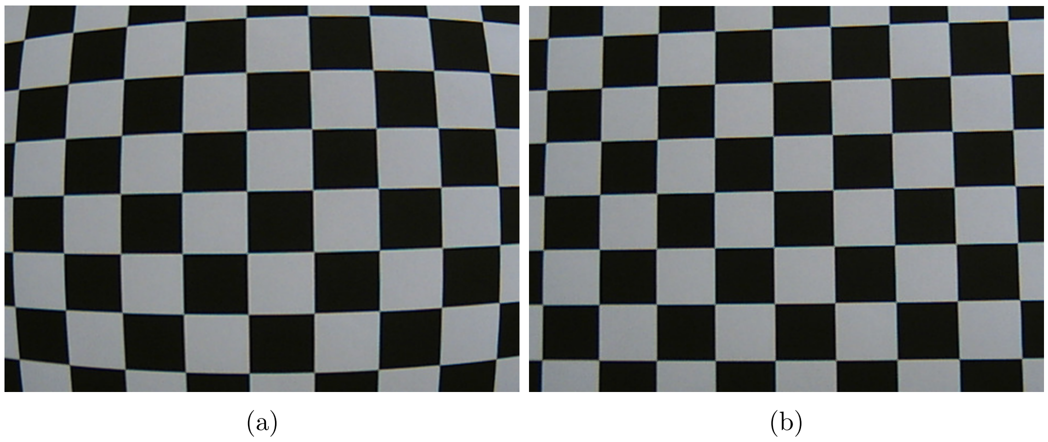


Figure 2.4: An example of an image captured with an imperfect lens (a) before and (b) after correction using the `undistort` (OpenCV, 2013c) function in OpenCV. Note that the distortion is non-uniform, and the curving of straight lines is more pronounced near the edges of the image. This is an example of barrel distortion, caused by imperfectly shaped optical components.

2.1.3 Intrinsic Parameters

Intrinsic parameters are parameters of the camera and consist of the focal length, pixel scale factors, and skew between the x and y axes. These effects are uniform over the image and are necessary to calibrate the camera in order to give its geometric quantities real units. The intrinsic parameters are encapsulated in the (3×3) matrix $\mathbf{A}$.

$$\mathbf{A} = \begin{bmatrix} \alpha_x & \gamma & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

where $[x_0, y_0]^T$ is the location of the principle point, γ is the skew coefficient between the x and y axes (usually zero), and α_x and α_y represent focal length in terms of pixels along each axis,

$$\alpha_x = f m_x, \quad \alpha_y = f m_y, \tag{2.4}$$

where m_x and m_y are scale factors which convert pixels to units of distance along each axis.

2.1.4 Extrinsic Parameters

Extrinsic parameters convert between the camera-centric coordinate system to the fixed world coordinate system. They represent the camera *pose* using a translation vector $\mathbf{t}$ and a rotation matrix $\mathbf{R}$.

$$\mathbf{R} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} \Delta u \\ \Delta v \\ \Delta w \end{bmatrix}, \tag{2.5}$$

which is usually expressed as a single 3×4 matrix $[\mathbf{R},\mathbf{t}]$,

$$[\mathbf{R}, \mathbf{t}] = \begin{bmatrix} R_{11} & R_{12} & R_{13} & \Delta u \\ R_{21} & R_{22} & R_{23} & \Delta v \\ R_{31} & R_{32} & R_{33} & \Delta w \end{bmatrix}, \tag{2.6}$$

The concatenated pose matrix in (2.6) allows the complete model to be expressed conveniently using *homogeneous* coordinates

$$\mathbf{p} = w \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{A}[\mathbf{R}, \mathbf{t}] \begin{bmatrix} u \\ v \\ w \\ 1 \end{bmatrix}, \tag{2.7}$$

where $[x, y]^T$ is in image space and $[u, v, w]^T$ is a point in world coordinate space. $\mathbf{R}$ and $\mathbf{t}$ are the extrinsic parameters which convert from the world space to camera space, and $\mathbf{A}$ is the matrix of intrinsic camera parameters.

2.1.5 Distortion Coefficients

Generally radial distortion is dominant and in most cases it is sufficient to consider only radial distortion (Zhang, 2000, Du et al., 2011), which is corrected in openCV after estimating a set of distortion coefficients k_1, k_2 and k_3 during calibration (OpenCV, 2013b)

$$\begin{aligned}\hat{x} &= x(2 + k_1 r^2 + k_2 r^4 + k_3 r^6)\\ \hat{y} &= y(2 + k_1 r^2 + k_2 r^4 + k_3 r^6),\end{aligned} \tag{2.8}$$

However, the OpenCV library also calibrates and corrects for tangential distortion (OpenCV, 2013b) by estimating p_1 and p_2

$$\begin{aligned}\hat{x} &= x + (2p_1 xy + p_2(r^2 - 2x^2))\\ \hat{y} &= y + (2p_1 xy + p_2(r^2 - 2y^2)),\end{aligned} \tag{2.9}$$

where in both (2.8) and (2.9) $\hat{x}$ and $\hat{y}$ are the corrected locations of coordinates the point observed at (x, y) coordinates of an observed point in the image. This is repeated for each pixel in the image. This approach is based on work by Zhang (2000). For all of the work in this thesis it is assumed that the camera is calibrated and the images are corrected using the above method.

2.1.6 Camera Calibration

Generally the camera parameters and the distortion coefficients are unknown and must be estimated. Camera calibration (also called resectioning) refers to the process by which the parameters are estimated. During calibration a planar object with a known pattern, such as the checkerboard in Figure 2.4, is imaged at various orientations and the expected feature locations are compared with the observed ones. In this case the features of interest are the corners of the checkerboard squares. The calibration routine uses the algorithms outlined in Zhang (2000), Du et al. (2011) to estimate the distortion coefficients of Brown's model, which are then used in (2.8) and (2.9) to correct the distorted image. Figure 2.4 shows an example of applying this method of distortion correction to the chessboard pattern used to estimate the parameters.

2.2 MULTIPLE VIEW GEOMETRY

The most common example of multiple view geometry is stereo imaging, however, the only requirement is that a common scene is viewed from multiple camera poses. Multiple views can be achieved through camera motion, assuming the scene is static. In this section, stereo imaging notation is used to denote different camera views for simplicity; $\mathbf{p}_L$ and $\mathbf{p}_R$ are the projections of $\bar{\mathbf{p}}$ into images by the left and right cameras respectively. The mathematics applies to the general case of any two images captured at different camera poses.

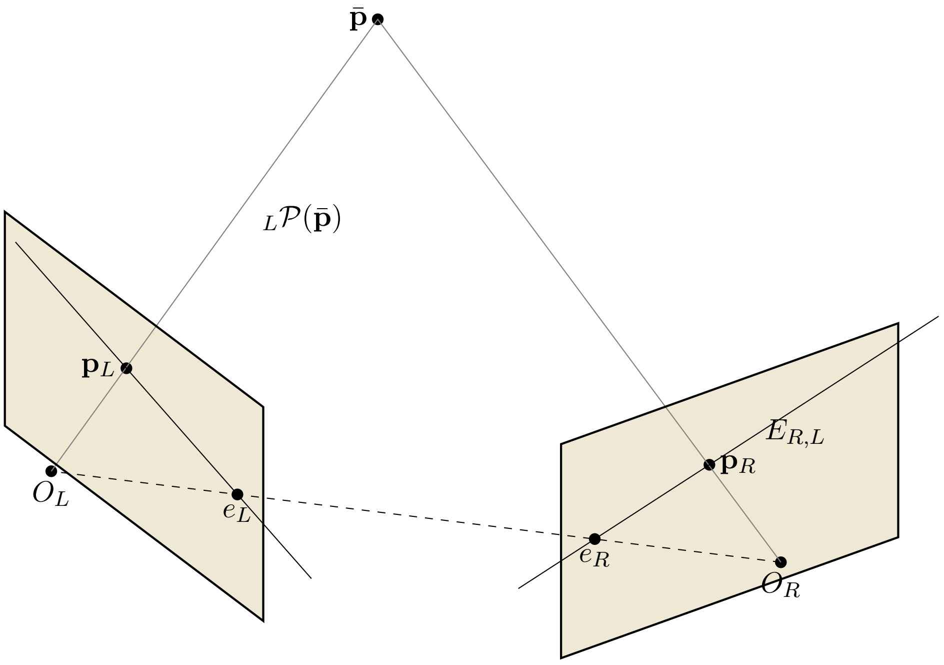


Figure 2.5: Epipolar geometry relates common points in images captured from different camera positions. This figure shows the stereo case where the point $\bar{\mathbf{p}}$ is projected onto left ($\mathbf{p}_L$) and right ($\mathbf{p}_R$) pinhole cameras with principal points at $\mathbf{O}_L$ and $\mathbf{O}_R$ respectively. The epipoles are the principal points of each view projected onto the opposite image plane. ($\mathbf{e}_L$ and $\mathbf{e}_r$). Each point has an epipolar line between the epipole and its projection onto the image plane, which is the same as the image of the points projection line onto the other image plane Here $\bar{\mathbf{p}}$ is projected onto the left hand image and $_L\mathcal{P}(\bar{\mathbf{p}})$ is imaged in the right hand plane as the epipolar line $E_{R,L}$.

Epipolar geometry is the more specific topic which relates the projections of a 3D point into multiple camera views. This is based on geometric transformations represented by matrices, and assumes points are represented using *homogeneous* coordinates.

2.2.1 Essential and Fundamental Matrices

The essential matrix was first defined by the Longuet-Higgins equation (Longuet-Higgins, 1981)

$$\mathbf{p}_L^T \, \mathbf{E} \, \mathbf{p}_R = 0, \tag{2.10}$$

where $\mathbf{p}_L$ and $\mathbf{p}_R$ are expressed in *normalized* homogeneous coordinates and correspond to the same 3D point imaged by the left and right cameras respectively. $\mathbf{E}$ is the essential matrix. Epipolar geometry can be expressed in matrix form using the essential or fundamental matrix of a system. The essential matrix relates the projection of a point in one view to its projection in another view, assuming an ideal pinhole camera.

The essential matrix is defined in terms of the rotation and translation between the camera views

$$\mathbf{E} = \mathbf{R}[\mathbf{t}]_\times, \tag{2.11}$$

where $[\mathbf{t}]_\times$ is the matrix representation of the cross product of $\mathbf{t}$,

$$[\mathbf{t}]_\times = \begin{bmatrix} 0 & -t_3 & t_2 \\ t_3 & 0 & -t_1 \\ -t_2 & t_1 & 0 \end{bmatrix}, \quad \mathbf{t} = \begin{bmatrix} t_1 \\ t_2 \\ t_3 \end{bmatrix}. \tag{2.12}$$

This can be extended to non-ideal cameras, if the intrinsic camera matrices are known

$$\mathbf{E} = \mathbf{A}_L^T \mathbf{M} \mathbf{A}_R, \tag{2.13}$$

where $\mathbf{M}$ is the fundamental matrix and $\mathbf{A}_L$ and $\mathbf{A}_R$ are the intrinsic calibration matrices for the left and right cameras respectively. $\mathbf{M}$ replaces $\mathbf{E}$ in (2.10) and extends the equation to non-ideal cameras with known parameters

$$\mathbf{p}_L^T \mathbf{M}\, \mathbf{p}_R = 0 \tag{2.14}$$

The fundamental and essential matrices are the mathematical representation of the *epipolar constraint* illustrated in Figure 2.5. That is, the projection of a point into one camera view(the left image from a stereo system for example) will be projected to a point which lies on its epipolar line in another view (the right image). The epipolar line is the image of the points projection line into the opposite view.

2.2.2 Estimating the Essential and Fundamental matrices

The essential matrix can be estimated using the eight-point algorithm presented by Longuet-Higgins (1981). As the name suggests the algorithm uses eight point correspondences to solve the Longuet-Higgins equation for the essential or fundamental matrix. Equation (2.10) is rearranged as

$$(\mathbf{p}_{Li}\mathbf{p}_{Ri})\mathbf{M} = 0, \quad i = [1, 8], \tag{2.15}$$

where $\mathbf{p}_{Li}$ and $\mathbf{p}_{Ri}$ are the *ith* corresponding points in the left and right images expressed in homogeneous coordinates, and $\mathbf{M}$ is the fundamental matrix. Eight equations of the form (2.15) are solved simultaneously for the elements of $\mathbf{M}$. This is typically achieved by expressing $\mathbf{M}$ as a column vector $\mathbf{m}$ and using (2.15) to collate the 8 point correspondences into a matrix $\mathbf{A}$

$$\mathbf{Am} = 0, \tag{2.16}$$

where

$$A_i = [xx', xy', x, yx', yy', y, x'y', 1] \tag{2.17}$$

is the *ith* row of $\mathbf{A}$, $\mathbf{p}_{Li} = [x, y]$, $\mathbf{p}_{Ri} = [x', y']$, and

$$\mathbf{m} = [M_{11}, M_{21}, M_{31}, M_{12}, M_{22}, M_{32}, M_{13}, M_{23}, M_{33}]^T, \tag{2.18}$$

This uses the rank-deficient property of $\mathbf{M}$ requiring just 8 point correspondences. This requires $\mathbf{A}$ to be at most rank 8. In practice, perturbations and incorrect point matches cause $\mathbf{A}$ to be rank 9 (or more if more than 8 points are used). To avoid ambiguity an additional constraint is added

$$\|\mathbf{m}\| = 1. \tag{2.19}$$

The eight point algorithm avoids the iterative solution which is possible with only five correspondences (Longuet-Higgins, 1981) and provides an analytical solution instead. The resulting fundamental matrix $\mathbf{M}$ is required to have rank 2, however, equations in the form of (2.15) do not generally produce matrices subject to this constraint. This constraint is typically enforced using Singular Value Decomposition (SVD) (Hartley, 1997). In practice, the *normalized* variant of the 8-point algorithm where the correspondences are normalized before estimating the matrix, with a speed up of 20 over iterative methods with similar accuracy (Hartley, 1997). An further improvement over the normalized algorithm is the use of total least-squares (TLS) Total least-squares estimate (Mühlich and Mester, 1998)

2.2.3 Planar Homography

From the model in (2.7) the projection matrix $\mathbf{A} = [\mathbf{R}, \mathbf{t}]$ has size 3×4. If all of the interest points lie on the same plane then the world coordinate system can be defined so that one component (e.g., the w component) is zero for all of the points. In this case projection matrix can be simplified by discarding one column (e.g. the third column if the w component is zero). The simplified projection matrix is the homography matrix mapping the points in the plane to the image plane of a camera. This can be used to map points between camera views, for example mapping a point in the right hand camera of a stereo imaging system to its expected location in the image plane of the left camera

$$\mathbf{p}_L = \mathbf{A}_L \mathbf{H}_{RL} \mathbf{A}_R^{-1} \mathbf{p}_R, \tag{2.20}$$

where $\mathbf{A}_L$ and $\mathbf{A}_R$ are the intrinsic camera matrices for the left and right cameras. The homography matrix is defined in terms of the rotation and translation between the two planes

$$\mathbf{H}_{RL} = \mathbf{R} - \frac{\mathbf{t}\mathbf{n}^T}{d}, \tag{2.21}$$

where $\mathbf{R}$ is the matrix describing the rotation between the two image planes and $\mathbf{t}$ is the translation. $\mathbf{n}$ is the normal vector to the plane being imaged.

2.2.4 Stereo Triangulation

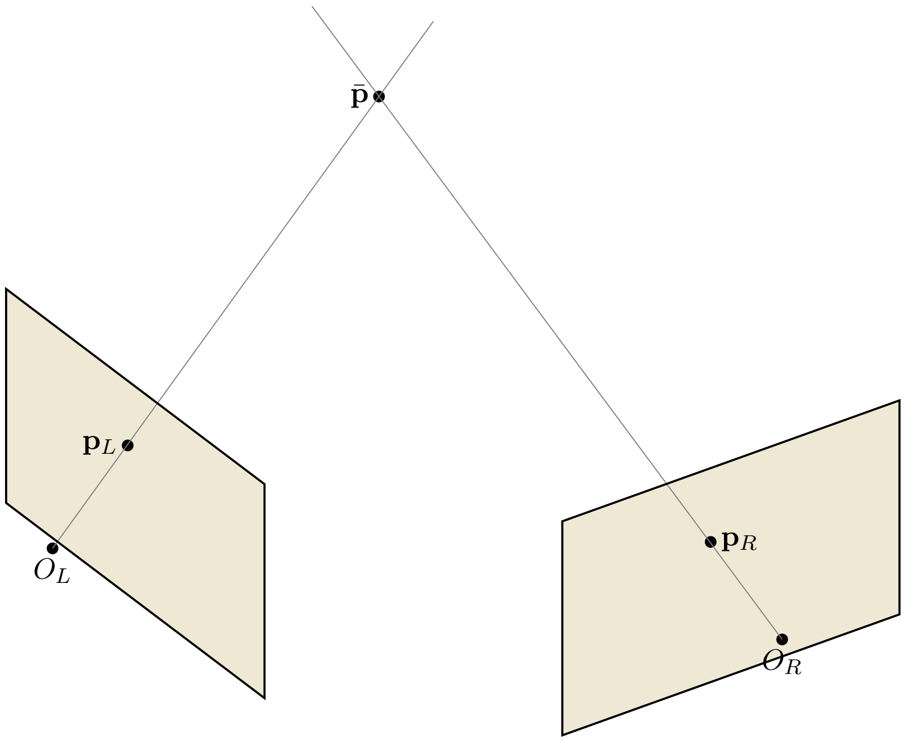


Figure 2.6: The point $\bar{\mathbf{p}}$ is projected into the left and right cameras as $\mathbf{p}_L$ and $\mathbf{p}_R$ respectively. From the camera parameters the projection lines can be calculated and the intersection of the lines can be found to determine the *depth* of the point.

From the pinhole model introduced in Section 2.1.1, the exact 3D position of a point, $\bar{\mathbf{p}}$, in the world coordinate system can be computed if the point is imaged from two different positions. Given that the camera matrices are known, the projection lines can be computed, and their intersection will give the 3D position of the point. However, pixel quantization makes it impossible to measure the exact position of the projected point in either image, so the straightforward back-projection approach will fail Hartley and Zisserman (2004).

The triangulation function is expressed as a function of the camera matrices $\mathbf{A}_L$ and $\mathbf{A}_R$, and the projection of $\bar{\mathbf{p}}$ onto each image plane ($\mathbf{p}_L$ and $\mathbf{p}_R$).

$$\hat{\mathbf{p}} = \tau\left(\mathbf{p}_L, \mathbf{p}_R, \mathbf{A}_L, \mathbf{A}_R\right) \tag{2.22}$$

where $\hat{\mathbf{p}}$ is an *estimate* of $\bar{\mathbf{p}}$.

2.2.5 Pose Estimation

The *perspective n-point* (PnP) problem refers to the estimation of the *pose* (position and orientation) of a camera given the camera parameters and a set of n point correspondences between known 3D points and their 2D projections (Lepetit and Fua, 2006). The earliest algebraic solution was found by Horaud et al. (1989), and since there has been a large body of work published on the problem (Quan and Lan, 1999, Horaud et al., 1989, Lepetit and Fua, 2006), resulting in many solutions, both iterative and analytical. This section will consider the ePnP method by Lepetit and Fua (2006), which has some notable advantages over its alternatives. ePnP is a non-iterative method which scales linearly with n, the number of point correspondences. The next best scaling non-iterative method is $O(n^2)$, and for comparable robustness to noise the next best scaling method is $O(n^5)$ (Lepetit and Fua, 2006). It also operates on any number of point correspondences so long as $n \geq 4$, and it is faster than iterative methods, but with comparable accuracy (Lepetit and Fua, 2006).

The premise of ePnP is to express each of the n 3D points as a weighted sum of four control points

$$\omega_i \begin{bmatrix} \mathbf{u}_i \\ 1 \end{bmatrix} = \mathbf{A} \sum_{j=1}^{4} \alpha_{ij} c_j^c, \tag{2.23}$$

where ω_i is the scalar projective parameter, $[u_i, v_i]^T$ is the 2D image point, $\mathbf{A}$ is the camera matrix, and $\mathbf{c}_j^c$ are the control points in the camera coordinate system. α_{ij} are homogeneous barycentric coordinates.

(2.23) is rearranged to a linear system of the form

$$\mathbf{M}\mathbf{x} = \mathbf{0} \tag{2.24}$$

where $\mathbf{x} = [c_1^{cT}, c_2^{cT}, c_3^{cT}, c_4^{cT}]^T$ is a 12-vector of the control points.

The solution is found in the null-space of M. For a complete description of the solution, and of choosing the right weights for the control points, see Lepetit and Fua (2006).

2.3 IMAGE FEATURES

In computer vision a 'feature' is a region or patch in an image which is likely to be uniquely identifiable. Features provide a way to identify common points in a scene from images captured at different times, with different illumination conditions, or from different camera poses (Hartley and Zisserman, 2004). Common features identified in multiple images are used to generate point correspondences, which are the basis for many geometric operations such as those described in Section 2.2.

Algorithms for working with features can be split roughly into three categories: detection, extraction, and matching algorithms. Description schemes are affected by the

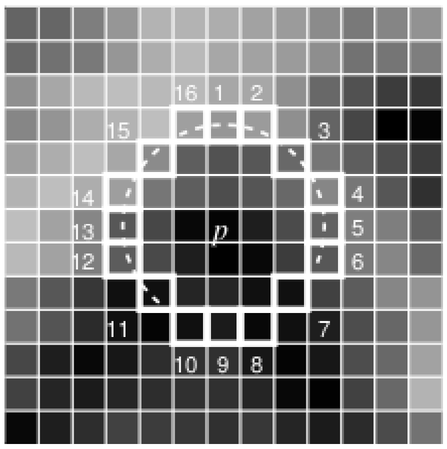


Figure 2.7: The Bresenham circle used in the FAST algorithm with the test pixels highlighted. Figure adapted from Rosten and Drummond (2006).

quality and type of the features found by the detector, so the detection and extraction stages are sometimes paired (Lowe, 2004, Bay et al., 2006, Rublee et al., 2011). Description schemes may also allow for improved matching algorithms (Ortiz, 2012), however each category is discussed separately ere. Emphasis is placed on the FAST (Rosten and Drummond, 2006) and BRIEF (Calonder et al., 2011) algorithms used for the experiments in Chapter 5, and ORB (Rublee et al., 2011) which adds rotation and scale invariance to BRIEF.

2.3.1 Detection

Features are typically located at interest points where there is a maximum or minimum in the local derivatives of the image, such as at small spots or corners (Prince, 2012). In the general case this also includes lines or edges and sometimes large blobs, however, geometric operations require precise point locations, so only corner and point detectors are used. A *de facto* standard for feature detection is Features from Accelerated Segment Test (FAST) (Rosten and Drummond, 2006) which is used for the experiments carried out in Chapter 5. FAST is based on the Accelerated Segment Test (AST) algorithm, which is also the basis for the SUSAN (Smith and Brady, 1997) detector, and the Adaptive Generic Accelerated Segment Test (AGAST) (Mair et al., 2010).

FAST detects features by comparing a centre pixel (the nucleus) to 16 pixels in a Bresenham circle (Bresenham, 1977) with a radius of $r = 3.4$ around the nucleus. An example of the test pixels surrounding a corner is shown in Figure 2.7. Features are detected when 12 or more contiguous pixels on the radius differ from the nucleus above a threshold determined by the value of the nucleus. This test is repeated at each pixel and the sum of the absolute differences between the nucleus and the pixels in the contiguous arc is calculated as the corner score V. A threshold is applied to V to select potential corners, and non-maximal suppression is then performed on the resulting candidate features to prevent detecting the same feature multiple times. This is achieved by only selecting corner features at pixels where a local extrema of V occurs (Rosten and Drummond, 2006, Neubeck and Van Gool, 2006). Non-maximal suppression is necessary because a corner or a bright spot is not necessarily a single pixel, so the surrounding pixels are likely to have high corner scores.

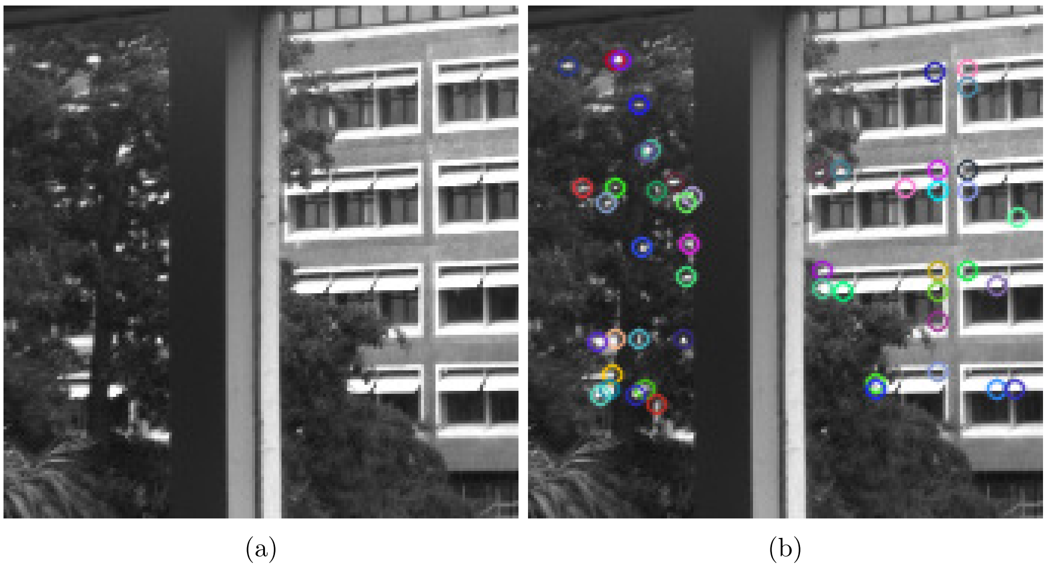


Figure 2.8: Features detected in a monochrome image. (a) The original image and (b) image with some detected features overlaid. Some points were discarded by thresholding the FAST score.

Figure 2.8 shows an example of feature detection using the FAST feature detection algorithm. Many of the bright spots and corners have been detected but many have also been excluded by thresholding the corner score V, with higher scores corresponding to higher contrast features.

2.3.2 Extraction

Feature extraction aims to extract the unique information encoded in the image patch and express it compactly for storing and comparison with other features. The compressed representation returned by a feature extractor is called the feature descriptor and usually consists of a vector of floating point values (Lowe, 2004, Bay et al., 2006) or binary strings (Calonder et al., 2011, Ortiz, 2012, Leutenegger et al., 2011). In addition, the patch orientation and scale information may also be stored in the descriptor.

The de facto standards for robust scale and rotation invariant features are SIFT (Bay et al., 2006) and SURF (Lowe, 2004) which use the floating point representation. However, more recently binary descriptors such as FREAK (Ortiz, 2012), BRISK (Leutenegger et al., 2011), ORB (Rublee et al., 2011), and BRIEF (Calonder et al., 2011) have become common. Binary descriptors are attractive particularly when processing data in real-time on hardware with limited processing power. For this reason the remainder of this section is primarily concerned with binary descriptors, however, a list of common descriptors is given in Section 2.3.2.1.

2.3.2.1 Commonly used feature descriptors

- **Scale Invariant Feature Transform (SIFT), 2004** (Lowe, 2004) is a scale and rotation invariant feature detection algorithm published by David Lowe in 2004.

- **Speeded Up Robust Features (SURF), 2006** (Bay et al., 2006) is a detector and descriptor with scale and rotation invariance.

- **Binary Robust Independent Elementary Features (BRIEF), 2012** (Calonder et al., 2011) is a lightweight feature descriptor lacking scale and rotation invariance, although it is robust to small rotation angles. It is used as the base feature descriptor for ORB, which adds rotation and scale invariance.

- **Oriented FAST and Rotated BRIEF (ORB), 2011** (Rublee et al., 2011) adds rotation and scale invariance to the BRIEF descriptor and uses the FAST feature detector in its scale pyramid. The authors present it as *'an efficient alternative to SIFT or SURF'*, which is supported by Wu et al (Wu and Lew, 2013) who found a speedup of approximately 340 over SIFT and 40 over SURF with comparable matching performance.

- **Fast Retina Keypoints (FREAK), 2012** (Ortiz, 2012) is a descriptor which uses a sampling pattern inspired by the distribution of ganglion cells in the human retina. FREAK is rotation and scale invariant, and the coarse-to-fine sampling scheme allows for improved matching performance inspired by *saccadic* search in biological vision.

Wu and Lew (2013) reviewed the above descriptors and determined their relative detection time, descriptor extraction time, and their matching performance. Table 2.1 summarizes their relative performance.

	Detection(ms)	Extraction (ms)	Total (ms)	Detection Accuracy (%)
SIFT	8.68	7.5	16.8	68.7%
SURF	0.424	0.96	1.384	46.3 %
ORB	0.604	0.022	0.626	58.5 %
FREAK	0.424	0.045	0.469	43.3 %
BRIEF	0.424	0.023	0.447	36.9 %
FAST	0.0064	–	–	–

Table 2.1: The performance of each descriptor as evaluated by Wu and Lew (2013). Detection and extraction times are per-feature and FAST is shown for a comparison with the ORB, SURF, and SIFT detectors. Wu et al used the SURF detector with FREAK and BRIEF, while SIFT uses a bespoke detection algorithm and ORB uses a modified version of FAST.

2.3.3 Binary Feature Descriptors

SIFT and SURF use large floating point vectors ($n = 256$ values for SIFT, $n = 64$ for SURF) which are compared by computing the n-dimensional Euclidean distance between the descriptor pairs. This can cause tractability problems when attempting to store large numbers of features or when extracting and comparing features in real-time on low-power computing hardware. The advantage of binary descriptors is twofold; binary descriptors consume little memory, for example BRIEF uses just 128 bits for feature description. In 32-bit environments this is 16 bytes compared to 256 and 1024 bytes for SURF and SIFT respectively, and feature comparison is reduced to calculating the Hamming distance, which can be calculated by taking the bitwise XOR of two descriptors and counting the bits in the result. These advantages make binary descriptors an attractive alternative to the more computationally expensive descriptors, particularly since binary descriptors can offer similar robustness (Rublee et al., 2011).

Binary descriptors are based on the work by Ozuysal et al. (2010) and Lepetit and Fua (2006) which show that robust image patch identifiers can be constructed using pairwise intensity comparisons. Binary descriptors are constructed using pairwise intensity comparisons from point pairs sampled over the smoothed image patch (Calonder et al., 2011, Ortiz, 2012, Leutenegger et al., 2011).

In this thesis the BRIEF descriptor is used as a lightweight feature descriptor where rotation or perspective warping can be ignored. The ORB detector and descriptor is suggested for use where where scale and rotation invariance are required.

2.3.3.1 BRIEF

The BRIEF descriptor is a binary descriptor which uses a random Gaussian sampling pattern to construct a binary string from pairwise intensity comparisons. Calonder et al. (2011) demonstrate the performance of BRIEF compared with that of SURF and U-SURF (SURF without orientation). BRIEF does not consider feature orientation and therefore is not robust to in-plane rotations or perspective warping. Its authors suggest computing a set of descriptors for rotated and perspective warps of each patch, however, this would negate the some of the speed advantages of BRIEF.

2.3.3.2 ORB

ORB is presented as directly comparable with SURF or SIFT but is intended for use in low-powered embedded environments such as onboard a small UAV. Feature scale is considered using a *global* scale pyramid for the whole image rather than on a per-feature basis. Scale invariance is achieved by filtering the Harris corner measure (Harris and Stephens, 1988). Feature orientation is computed using the intensity centroid based on image *intensity moments* (weighted averages of the intensity over the image patch)

$$\Theta = \begin{cases} \tan^{-1}\frac{m_{01}}{m_{10}} & \text{if corner} = \text{bright} \\ \tan^{-1}\frac{m_{01}}{m_{10}} + 180 & \text{if corner} = \text{dark} \end{cases}, \tag{2.25}$$

here m_{pq} is the intensity moment of the feature patch at $(x, y) = (p, q)$,

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y). \tag{2.26}$$

where $I(x, y)$ is the intensity of the patch at (x, y), and the moment is computed over pixels with in the feature patch size of $(x, y) = (p, q)$. 180° compensation is required if the corner is darker than the surrounding pixels to correct for the offset of the moments between light and dark corners. The intensity centroid C is defined in terms of the moments m_{01}, m_{10}, and m_{00},

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right). \tag{2.27}$$

The orientation is then used to rotate the sampling pattern of the BRIEF descriptor (steered BRIEF). The BRIEF sampling pattern was also augmented by a learning algorithm over a set of 300k keypoints from the PASCAL dataset PASCAL (2013) to reduce the outlier correlation (Rublee et al., 2011).

ORB has been shown by its authors to be more robust than SIFT and SURF in some cases while detection and extraction is an order of magnitude faster than SURF and more than two orders of magnitude faster than SIFT. Performance was measured using the PASCAL dataset; two image sequences exhibiting in-plane rotation and affine warping (Rublee et al., 2011). A review of descriptor extractors and descriptors for augmented reality (AR) by Wu and Lew (2013) supports these claims.

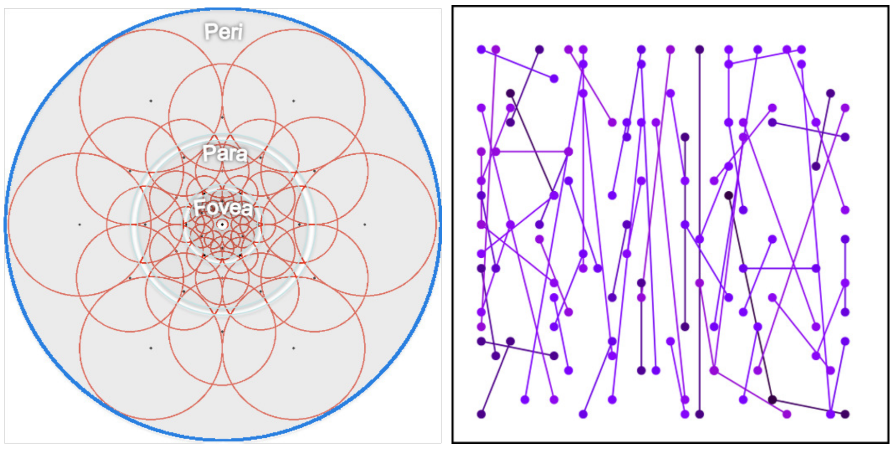


Figure 2.9: Sampling patterns used for (a) FREAK (adapted from Ortiz (2012)) and (b) ORB (adapted from Rublee et al. (2011)) descriptors. The ORB pattern is similar to the random pattern used in the BRIEF descriptor but has been optimized to minimize descriptor correlation under rotation. FREAK uses a sampling pattern where each sample point is individually smoothed, with increased resolution near the centre pixel.

2.3.3.3 FREAK

The FREAK descriptor is based on a novel coarse-to-fine sampling pattern inspired by the distribution of the ganglion cells over the human retina (Ortiz, 2012). A circular sampling pattern is used with a higher density of sample points near the centre of the patch than near the edges of the patch. In addition, the outer sample points use larger slightly overlapping smoothed patches. Orientation is estimated by summing the gradients of paired intensity measurements.

The sampling pattern allows a coarse-to-fine matching scheme, where the coarse half section of the descriptor is used to discard obvious false matches, and the fine section is used to refine the remaining potential matches to single pairs. The coarse-to-fine approach improves matching efficiency: 1.44× over BRISK, and 22× and 40× over SURF and SIFT respectively. Computing FREAK descriptors was also 77× faster than SURF and 138× faster than SIFT (Ortiz, 2012).

2.3.4 Scale and Rotation Invariance

As the camera position and orientation changes the observed feature patches are scaled and rotated. This is important for navigation where the camera pose is constantly changing, so it is necessary to ensure robustness to rotation and scaling of feature patches. Scale invariance is typically achieved by representing the image as a pyramid of ‘difference of Gaussians’ (DoG). The image is increasingly blurred, and the differences between each successive blurring form the levels of the pyramid. Features are then detected on all the layers, the descriptor is extracted from the layer which maximizes the feature detection score (Lowe, 1999, Rublee et al., 2011). In SIFT and SURF this is an integral part of the descriptor, while others such as BRISK and FREAK rely on feature detectors, such as the AGAST (Mair et al., 2010) extension to FAST, for scale invariance (Ortiz, 2012, Leutenegger et al., 2011). In this thesis the ORB descriptor is suggested for scale and rotation invariance because it has been shown to have comparable repeatability to SIFT at a fraction of the computational time (Wu and Lew, 2013, Ortiz, 2012).

2.3.5 Matching

Feature matching produces point correspondences from two sets of features extracted from different images of the same scene or object. Point correspondences are generated by finding similar features in both sets. The point correspondences provide the basis for pose and homography estimation, object recognition (Lowe, 1999), and many other operations in computer vision (Bradski and Kaehler, 2008, Brown and Lowe, 2007, Baumberg, 2000). Matching is performed by finding the nearest matches in descriptor space. In general, there will be incorrect matches below this threshold due to many features being similar, so an outlier rejection stage is typically applied after matching.

For an overview of feature matching techniques see Chapter 5.

2.3.6 Outlier Rejection

Because of the difficulty in describing a feature uniquely and efficiently, the matching process can generate false matches. And due to unavoidable uncertainties in feature locations, pose and homography estimation may be skewed by outliers.

A popular general purpose outlier rejection scheme is the Random Sample Consensus (RANSAC) algorithm (Fischler and Bolles, 1981). RANSAC outlier rejection, however, has a non-deterministic run-time because the algorithm is based on selecting a random subset of points and selecting the subset with the biggest set of inliers. Therefore, the run-time depends on the number and quality of the features available for matching, and there is no guarantee that RANSAC will find the correct solution. For a detailed overview of RANSAC outlier rejection see Section 5.2.

2.4 OPTICAL FLOW

Optical flow is the apparent 2D motion of an object across the field of view resulting from its 3D motion relative to the camera (Beauchemin and Barron, 1995, Horn and Schunck, 1981, Horn, 1986, Adiv, 1985). Motion fields and optical flow are recognized as a fundamental component of biological vision (Borst et al., 2010), particularly in insect vision (Stowers, 2013). In visual navigation optical flow can be used for dominant plane detection (Ohnishi and Imiya, 2005, 2006), collision avoidance (Green and Oh, 2008), odometry (Campbell et al., 2004, Nagatani et al., 2000), and altitude or speed control (Meier et al., 2012, Hérissé et al., 2010).

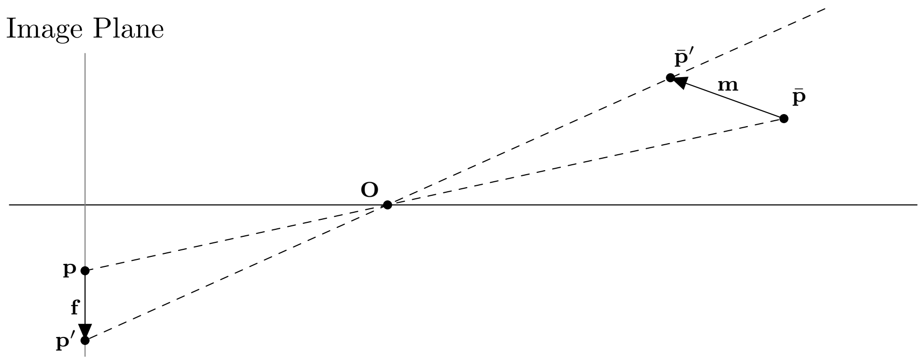


Figure 2.10: A two-dimensional illustration of the ambiguity in the relationship between optical flow and physical motion. The optical flow $\mathbf{f}$ could have been caused by observing the motion along the vector $\mathbf{m}$, but it could also have been observed for motion from $\bar{\mathbf{p}}$ to any point along the projection line from $\bar{\mathbf{p}}'$ to $\mathbf{p}'$.

The optical flow constraint equation is derived from the image intensity gradients at points in the image (Beauchemin and Barron, 1995)

$$\nabla I \cdot \mathbf{v} + \mathbf{I}_t = 0, \tag{2.28}$$

where $\nabla I = [I_x, I_y]$ is the image intensity gradient and $\mathbf{v}$ is the image velocity, or the observed optical flow. $\mathbf{I}_t$ is the time-varying component of the image intensity.

Equation (2.28) defines a line in image velocity space, and solving for $\mathbf{v}$ without other constraints can only yield the component of $\mathbf{v}$ perpendicular to the constraint line. The problem of finding the total flow using (2.28) is called the *aperture problem*, the ambiguity of the solution without additional constraints is illustrated Figure 2.11.

The aperture problem must be solved in order to find the total *local* optical flow. Usually it is solved by using additional constraints derived from the wider context of the image, or by assuming that the flow is locally continuous (Lucas et al., 1981) or globally smooth (Horn and Schunck, 1981). Section 2.4.1 provides a list of common methods.

2.4.1 Common Methods

Lucas-Kanade(Lucas et al., 1981) The Lucas-Kanade (LK) method solves the aperture problem by assuming that the flow is constant for a small neighbourhood of pixels. While it has good noise immunity, because it is a local method, it cannot be used to determine the flow in large uniform areas. This is usually fixed by using a Gaussian of pyramids to obtain increasingly more global flow fields.

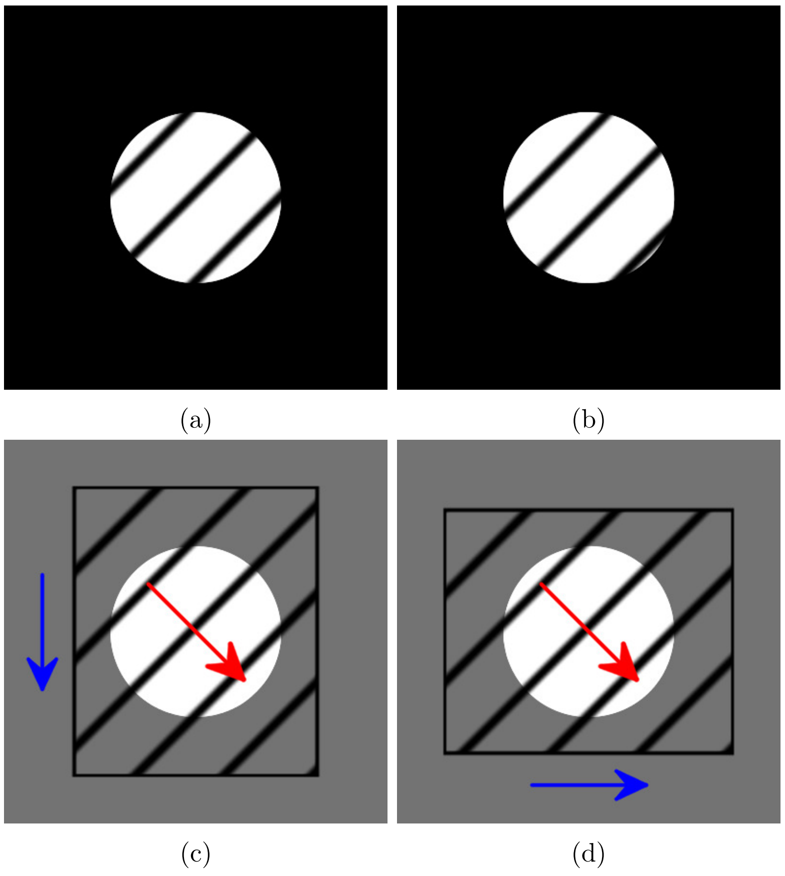


Figure 2.11: The aperture problem. (a) and (b) show to sequential frames from which the flow can be calculated. (c) and (d) show two cases with different *global* motion (blue arrows) which could have caused the *local* flow observed through the aperture (red arrows).

Horn-Schunck (Horn and Schunck, 1981) This method uses a *global smoothness* constraint to solve the aperture problem based on the assumption that optical flow is continuous nearly everywhere. It computes the dense optical flow, however, it is more susceptible to noise, and the error is worse in sequences with more discontinuities.

Farneback (Farnebäck, 2003) The Farneback method is based on the assumption that image neighbourhoods can be represented using polynomial expansion and that estimating the shift between the polynomial expansions between two frames produces a good estimate of the optical flow.

2.5 SUMMARY

This chapter has provided the necessary mathematical and computer vision background for visual control and navigation. Not all of the material in this chapter is tested, however, feature based control is demonstrated in Chapter 5. Most of the low-level algorithms have implementations in popular computer vision libraries (see Section 4.3), and so the work in this thesis is aimed at high level behaviour rather than implementation details.

Chapter 3
UAV HARDWARE

For the purpose of experimentation a small quadrotor UAV was built. The quadrotor is based on control hardware from the Pixhawk (Pixhawk, 2013) project, and uses the second generation PX4FMU flight control computer (autopilot). To facilitate visual control an ARM single board computer (SBC) and a single downward facing monochrome camera are mounted on the quadrotor. The quadrotor has a flight time of approximately 10 minutes, and a total weight of 980 g including the onboard computer and camera.

This chapter describes the quadrotor platform. Section 3.1 describes the actuation hardware, chassis, and power system. Section 3.2 describes the autopilot and its some of its application-specific peripheral boards. The onboard processing hardware is described in Section 3.3.1, with consideration of onboard computer perfomance. The integration of each subsystem into a vision-capable quadrotor is described in Section 3.4. A list of alternative platforms is given in Section 3.5, and a description of an alternative research system based on the commercially available Parrot AR.Drone is presented in Section 3.5.1. The chapter concludes with a summary of the system specifications in Section 3.6.

3.1 FLIGHT PLATFORM

Multi-rotor UAVs are popular research platforms because of their mechanical simplicity, their manoeuvrability, and ability to hover. An example of a four-rotor[1] configuration (quadrotor) is shown in Figure 3.1. Quadrotors consist of two pairs of counter-rotating propellers, each driven by a brushless DC (BLDC) motor mounted symmetrically around the centre plate which holds the battery, control hardware, and payload. Pitch and roll control is achieved by manipulating the balance of thrust from motors on opposite arms, and yaw is controlled by the balance of torque from the counter-rotating pairs. For a more complete overview of quadrotor control see (Hoffmann et al., 2007).

Typical open source quadrotors consist of two subsystems: The autopilot, and the drive hardware. This section is concerned with the latter.

[1] Typical open source autopilots are capable of controlling UAVs with six or eight rotors, however the four-rotor configuration is the simplest and most common

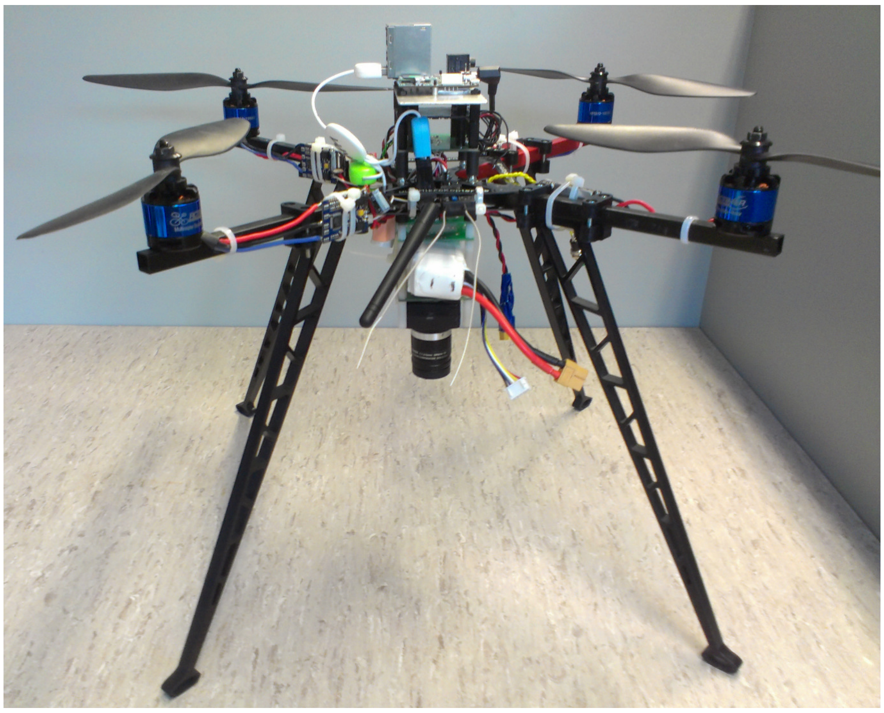


Figure 3.1: The UAV built as part of this thesis is an example of the popular quadrotor configuration.

3.1.1 Quadrotor Drive

Research by Pounds et al. (2006) and Hoffmann et al. (2007) has shown that control response latency is important for stable control. General purpose brushless motor controllers are intended for relatively low-frequency human input, and are typically designed for standard 50 Hz pulse position modulated (PPM) servo signals. The controller input filter normally has its cutoff frequency under 100 Hz. Pounds et al. (2006) showed that the latency introduced by limiting control signals to 50 Hz is, in general, insufficient for stable control. Furthermore, the PX4FMU control loop runs at 250 Hz (Pixhawk, 2014), which is five times faster than generic controllers are designed for. Therefore bespoke multi-rotor motors and controllers were chosen: HP2212-1000KV motors from RCTimer (RCTimer, 2013) paired with 30 A RCTimer motor controllers. The controllers are based on generic hobbyist hardware but are compatible with the 'SimonK' (SimonK, 2013) firmware patch to allow for 250 Hz PPM signals. The controllers are supplied with the patched firmware.

Table 3.1 shows the manufacturer specifications for the motors, motor drives, and battery. The peak thrust is 28.8 N, which allows for a total flying weight of 2.2 kg with a 30% thrust margin over hovering as suggested by Pounds et al. (2006). At maximum weight, however, the estimated flight time is reduced to 5 minutes.

3.1.2 Power supply circuit

A power control board was designed, including an arming switch which is off by default. The arming circuit is a hysteresis circuit which uses a high-current MOSFET and a momentary-on switch to toggle the state of the power supply to the motor drivers, and is based on the Paparazzi booz power distribution board (Paparazzi, 2013). In addition to manually disarming the motors, the hysteresis circuit also provides short-circuit protection.

The autopilot, onboard computer, and avionics hardware require 5 V power supply. To minimize chance of accidental control actuation, 5 V power supply is independent of the arming circuit, allowing the control hardware to operate with the motor controllers disabled. A 5 V, 2.25 A switch-mode power supply is used to provide the 5 V. This was found to be sufficient to power the autopilot, onboard computer, and the telemetry link.

Schematics of the arming and power supply circuits are attached in Appendix B

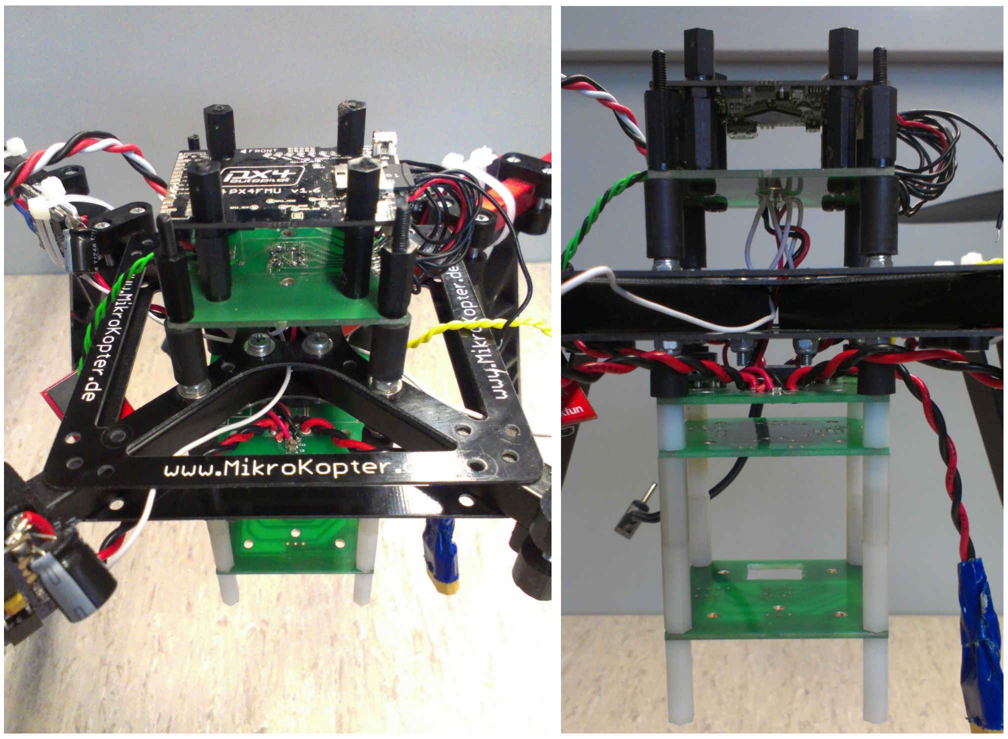
Motors		**Speed Controllers**		**Battery**	
Speed	1000 rpm/V	Current (10 s)	40 A	Voltage (nom)	14.8 V
Current (max)	≈9 A	Current (cont)	30 A	Current (cont)	77 A
Power (max)	130 W	Power (cont)	500 W	Power (cont)	1140 W
Thrust (max)	720 g			Capacity	32 Wh
Weight	59 g	Weight	23 g	Weight	280 g

Table 3.1: Manufacturer specifications for the motors (RCTimer HP2212), motor drivers (RCTimer SK-30A), and battery (4 cell LiPo). The quadrotor has a peak thrust of 28.8 N in total allowing a maximum flying weight of 2.2 kg (Pounds et al., 2006). The estimated flight time with a total weight of 1 kg is 10 minutes.

A four-cell Lithium polymer (LiPo) battery is used to supply power to all onboard systems.

3.1.3 Chassis

The Mikrokopter (Mikrokopter, 2013) chassis is a simple structure consisting of two fibreglass centre plates bolted to four square aluminium arms radiating from the centre plate at 90° intervals. The centre plate has 30 mm hole spacing compatible with the PX4FMU and with Praparazzi PCBs. The chassis is designed for 30 cm motor spacing, giving the quadrotor an overall size of 46×46 cm including rotor radius. The chassis weighs 218 g including the landing gear. Figure 3.2 shows the centre plates and flight control hardware. (with no onboard computer or camera). Rubber coupling is used between the chassis and the control boards and payloads to provide some isolation from motor vibrations.



(a) (b)

Figure 3.2: Mikrokopter showing the autopilot (top board), signal routing (below autopilot), and power distribution (below chassis) circuit boards. The bottom board has been adapted for mounting a downward-facing camera.

3.2 AUTOPILOT

The PX4FMU (Pixhawk, 2013a) is the second generation autopilot from the Pixhawk project. In 2012 it was still in development, however it was capable of stable flight and supported control via the MAVLINK library. Continued development has produced a flexible autopilot compatible with both fixed-wing and multi-rotor UAVs at the time of writing.

The PX4FMU consists of a single board which holds the inertial measurement and processing hardware. Motor control, serial communications, PPM input, and other IO is exposed via a 30-pin expansion header which is compatible with the carrier boards designed in conjunction with the PX4FMU. For this work a secondary 13-pin header is used which exposes enough IO for quadrotor control, MAVLINK communication via USART, and standard PPM control.

Several peripheral boards were designed as part of the Pixhawk project:
PXFlow: An optical flow camera which interfaces directly to the PX4FMU via a USART and enables position hold (Honegger et al., 2013).
PXIOAR: Carrier board which interfaces the PX4FMU to the Parrot AR.Drone quadrotor. It includes a 5 V, 2.25 A power supply and a XBee compatible WiFi module

	Gumstix Overo	**Odroid U2**	**PXComEx** (Pixhawk, 2012b)
CPU	700 MHz	4×1.7 GHz	2×1.86 GHz
Memory	512 MB DDR	2 GB DDR2	2 GB DDR2
Storage	SDHC	eMMC/SDHC	SATA-II
USB	1×USB 1.1	2×USB 2.0	7×USB 2.0
Weight	5.6 g	20 g	235 g
Power	8 W	10 W	27 W
FPU	No	Yes	Yes

Table 3.2: A comparison of onboard computer specifications. The PXComEx used in the pixhawk systen is included as a benchmark for high-performance onboard computers. Odroid and Gumstix power measurements were made at full CPU load.

headers.

PXIO: General purpose carrier board with an onboard failsafe microcontroller. This board is intended primarily for fixed-wing craft and includes several high-current output pins and 8 servo outputs.

3.2.1 Magnetic Sensor Interference

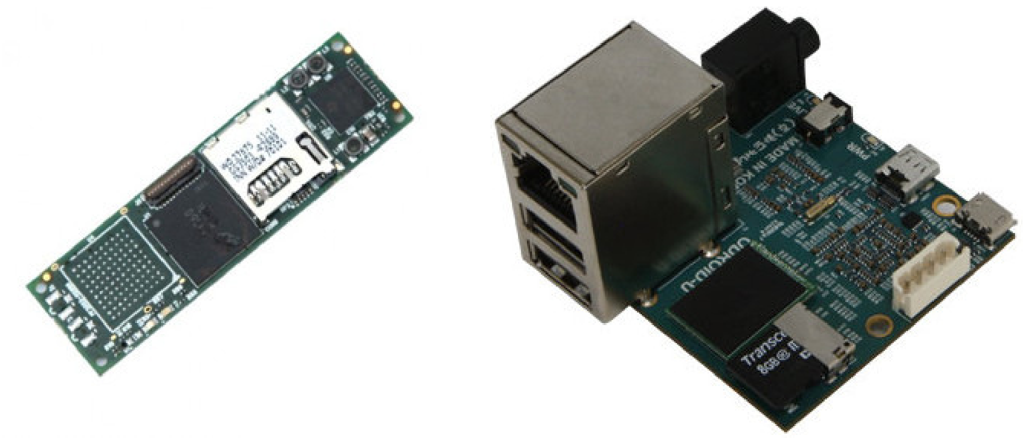
During initial test flights yaw interference of up to 40° were observed between idle and full-throttle. This was found to be the result of magnetic interference between the high-current motor supply circuit and the magnetometer. To avoid this, the power supply hardware was mounted below the chassis, and the autopilot was mounted above the chassis at the top of the board stack. A separation of 6 cm was found to be sufficient to reduce the yaw error to less than 0.5° between idle and full-throttle.

3.3 COMPUTER VISION SYSTEM

Visual control is achieved using a single camera and an onboard computer. During the course of this thesis two onboard computers were tested, and used with the Point Grey Chameleon machine vision camera.

3.3.1 Onboard Computer

The Gumstix (Gumstix, 2012) Overo Earth is a compact SBC released in 2008. It requires a separate carrier board to supply power and to expose USB, display, and networking peripherals. The original device lacks the processing power for meaningful real-time computer vision (Stowers, 2013). However, at the time of writing a second generation of hardware has been released and is based on an updated dual-core architecture. Only the first generation hardware was available so the Gumstix was only used during initial testing.

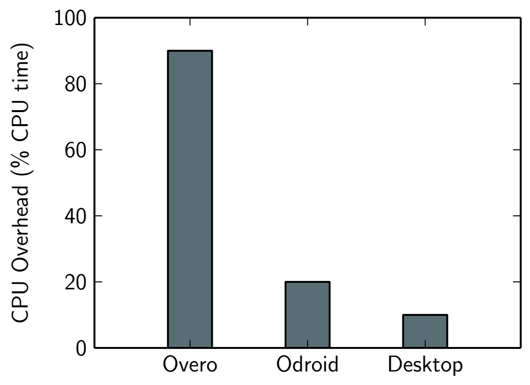


(a) (b)

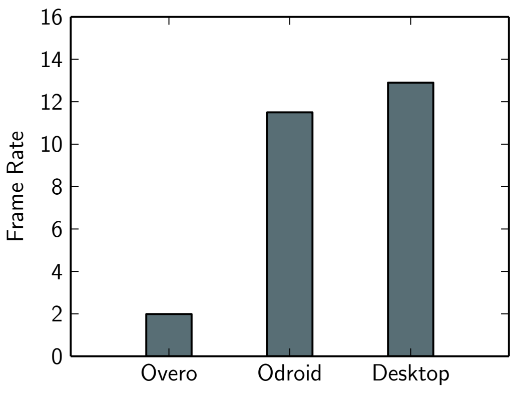
Figure 3.3: ARM single board computers used for onboard processing. (a) Gumstix Overo Earth daughter board. A carrier board is required to supply power and provide USB, networking, and other IO connectors. (b) The Odroid-U2. Power is supplied via a single 5 V DC jack. The board includes standard USB, ethernet, HDMI, and audio connecters.

Hardkernel (Hardkernel, 2013) developed the Odroid device family for developers using the Android operating system. The U family of devices is the compact version of the main line of devices. Power is provided by a single 5 V input and USB, Ethernet, and HDMI connectors are provided without the need for a carrier board.

Table 3.2 compares the specifications of each computer. The specifications of the PXComEx used onboard the Pixhawk Cheetah are also shown as an example of a high onboard computer. Clock speed, memory, and USB version are similar, however algorithm performance is defined also by the architecture, software optimizations, and other variables not shown here, so a performance comparison was carried out using the pose estimation algorithm from Chapter 5. In the performance test the algorithm is also drawing and displaying the matches as shown in Figure 5.2, which reduces the frame rate from the performance measured in Chapter 5. Figure 3.4 summarizes the performance comparison in terms of the average frame rate and the total system load. The performance of a desktop performance with specifications similar to the PXComEx is shown as a benchmark performance. All computers use the same algorithm parameters, and OpenCV version 2.4.2. From Figure 3.4, Gumstix performance is shown as severely limited, therefore further experiments are performed exclusively on the Odroid. Furthermore the frame rate on the Odroid, including the extra load of drawing and displaying visual data, is above 10 Hz which the lower bound on visual the visual control loop rate for stable quadrotor control Stowers (2013).



(a) CPU Overhead



(b) Frames Per Second

Figure 3.4: A comparison between the two onboard computers, with a desktop as a benchmark. Performance was measured for the pose estimation algorithm presented in Chapter 5 with a live display window of the matches.

3.3.2 Camera

The Point Grey Chameleon (Point Grey, 2013) is a bespoke machine vision camera based on a Sony CCD sensor. Images are transfered to the computer using the IEEE1394 protocol. Point Grey also provide a proprietary driver and software development kit, however, it is not open source and unsupported by the OpenCV camera control module.

3.4 COMPLETE SYSTEM

Figure 3.5 shows the structure of the complete system. The visual processing computer generates high level commands in the form of attitude setpoints and sends them to the autopilot. The autopilot compares the attitude setpoint to an estimate of the current attitude measured by an IMU and sends control signals to the motor controllers. The UAV has a weight of 980 g including onboard computer and camera, and has an estimated flight time of 10 minutes.

Weight	980 g
Max payload	1.2 kg
Flight time	10 min
Battery	32 Wh

Table 3.3: Specifications for the complete quadrotor platform including the onboard vision system. Flight time is calculated based on the battery capacity and assumes that most of the time will be spent at or near hover.

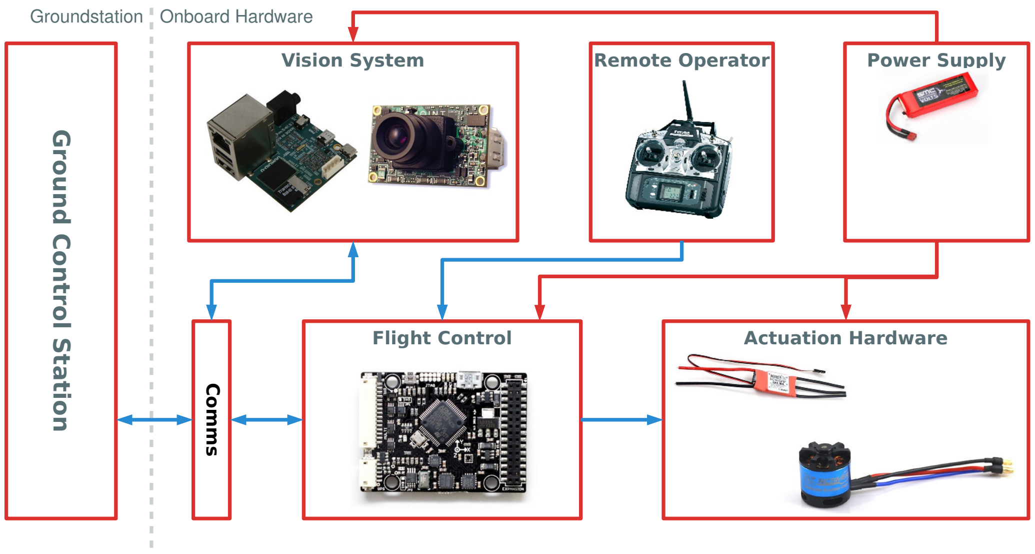


Figure 3.5: System architecture. Power is supplied by a 4-Cell LiPo battery via a power distribution board which steps the voltage down to 5 V for the visual processing hardware and autopilot. The autopilot uses control input from either a remote operator or from the onboard computer and adjusts the motor speed via the speed controllers (not pictured). Communications between the groundstation and the onboard computer allow flight data to be viewed in real-time, and also allow the operator to take control of the UAV.

3.5 ALTERNATIVE PLATFORMS

There are numerous open source and commercial alternatives ranging from complete systems such as those provide by Ascending Technologies (AscTec), to flexible autopilots and sensor systems such as the PX4FMU and ArduPilot systems. The following is a short list of systems considered when designing the quadrotor.

- **ArduPilot** (ArduPilot, 2012) is a popular platform among hobbysists due to its simplicity. Older versions have lacked the capabilities required by researchers, however recent versions are compatible with MAVLINK.

- **Paparazzi** (Paparazzi, 2013) is the original open-source autopilot, in many ways it is the predecessor of the Pixhawk system and the PX4FMU autopilot used in this thesis.

- **Mikrokopter** (Mikrokopter, 2013) is a complete multi-rotor platform consisting of control hardware, chassis, power supply, and motors.

- **Pixhawk** (Pixhawk, 2013) is the project which resulted in the PX4FMU as well as its predecessor. The aim of this project is to develop a flexible autopilot with visual control in mind.

- **Asctec** (Ascending Technologies, 2012) design multi-rotor UAVs with Intel-based onboard computers. The systems are targeted at researchers, but are costly.

3.5.1 The Parrot AR.Drone

The Parrot AR.Drone (Parrot, 2013a), shown in Figure 3.6, is a commercially available quadrotor designed for control via WiFi. Its intended use is manual control using an Android or Apple device, and it is advertised as a platform for augmented reality (AR) games. The system consists of a quadrotor with two cameras (forward and downward facing) and an ultrasonic altitude sensor, and has a flight time of 12 minutes. The proprietary control is capable of autonomous behaviour, as position hold and 'flips'. Offboard control is supported by the native autopilot via a published client API which connects to control server instance running on the native controller (Parrot, 2013b). The platform has several advantages over custom-built quadrotors and more complicated autopilots:

- Rotor-guards for operator safety and minimizing crash damage.
- Altitude and speed limits enforced by the onboard control software.
- Automatic position and altitude hold.
- Client-side control and communications API is documented for offboard control.
- Offboard control via NodeCopter (2013), including access to the image data from both cameras.
- Ready to fly.

Its relatively low cost (NZ$ 450-500), and the simplicity of the offboard control framework make the AR.Drone an ideal choice for computer vision research. However it is not designed to carry a payload, making it unsuitable for onboard processing.

3.6 SUMMARY

This section has described the UAV platform built for computer vision experiments. The quadrotor is capable of 10 minutes of continuous flight, and the onboard computer is capable of running feature based pose estimation above the 10 Hz threshold for stable visual control. With the pose estimation running the Odroid onboard computer has 3/4 of its total processing power free for other tasks. The quadrotor is also small, measuring less than 50 cm across and weighing just under 1 kg. It is therefore suitable for indoor operation. The Parrot AR.Drone was suggested as an off-the-shelf quadrotor suited for experimental visual control, although it is unsuitable for visual control with onboard processing.

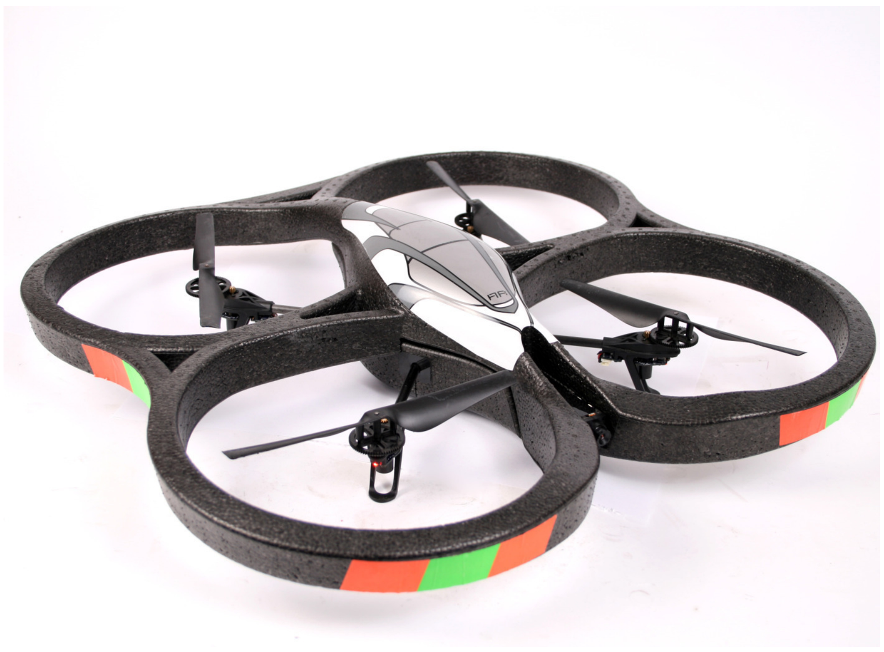


Figure 3.6: The Parrot AR.Drone quadrotor is a low-cost quadrotor which is designed for control over WiFi. A high-definition forward facing camera and a lower esolution downward facing camera, combined with a rotor guards on the chassis make it an ideal platform for indoor visual control experiments.

Chapter 4
CONTROL SOFTWARE

In Chapter 3 a quadrotor was described, consisting of actuation hardware controlled by an autopilot, and an onboard computer for processing visual data and sending high level commands to the autopilot. The autopilot software implements an attitude control loop to stabilize the quadrotor and to match the craft attitude to the set point from either the onboard computer or from remote human operator. For the purpose of visual control the autopilot is considered a 'black box' which accepts attitude set points as the control input. Both the computer vision software and the communication framework must be robust and low-latency to facilitate stable control. A telemetry link to the ground control computer is also required for observing and logging behaviour, and taking control manually.

This chapter provides an overview of the control and communication software running onboard the quadrotor. The Mavconn communications framework, which allows transparent communications between the two onboard computers and the ground control computer, is described in Section 4.1. A brief overview of the autopilot software is given in Section 4.4, including the controller tuning procedure which must be carried out to match the autopilots attitude controller to the UAV dynamics (Section 4.4.1). Section 4.3 presents the computer vision component of the software system, which is based on the OpenCV library. Some alternatives to OpenCV are also given (Section 4.3.1), some of which are designed specifically for the ARM architecture. Section 4.5 concludes the chapter with a description of the control loop formed by autopilot and visual processing computer.

4.1 COMMUNICATION SOFTWARE

The Mavconn communications framework (Pixhawk, 2013b) is based on the MAVLINK (Pixhawk, 2013c) and LCM (MIT, 2013) libraries. MAVLINK was developed as part of the Pixhawk project (Meier et al., 2012) and released in 2009. It is now used as the primary communications protocol in several open source autopilots including the PX4FMU and Ardupilot (for a complete list see the MAVLINK website (Pixhawk, 2013c)). LCM is a set of libraries for low-latency interprocess communication designed for real time applications.

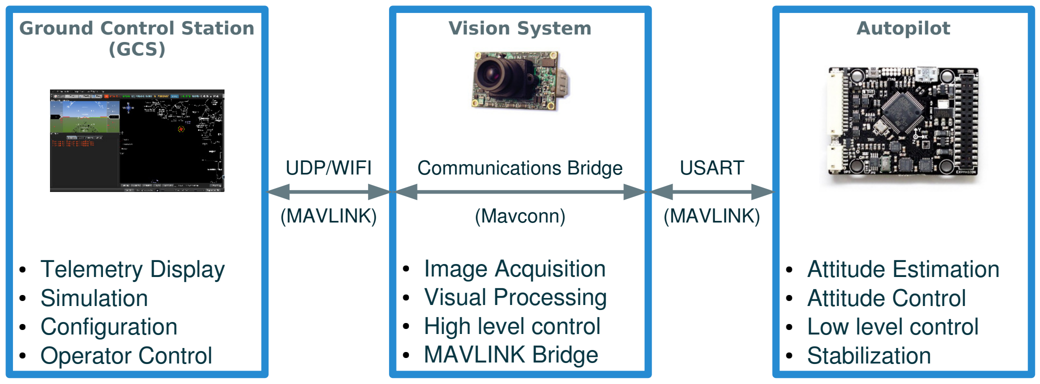


Figure 4.1: The functions performed by each of the three computers in the UAV system

Mavconn combines LCM and MAVLINK into a broadcast-subscribe communications framework with two default channels: IMAGE and MAIN. MAIN transports control, status, and telemetry data, and IMAGE transmits images acquired by the camera control module included in Mavconn. On the onboard computer efficient image transmission between acquisition and processing threads is achieved using shared memory, and transmitting the image pointer over the IMAGE channel. Drivers for communicating over standard protocols (USART and UDP, for example) are also provided.

The vision software is implemented as a multi-threaded program with the visual control module subscribing to the IMAGE channel and the control output module subscribing to the MAIN channel. Images are processed sequentially (intermediate frames are dropped if the previous frame is still bing processed when the next frame arrives), and the control output module is signaled after each successful frame.

Mavconn is capable of handling 5 cameras simultaneously, including a stereo imaging head. However only one camera is used in this thesis.

4.2 COMPUTER VISION SOFTWARE

One of the goals of this thesis is to demonstrate that real-time control is achievable using general purpose computer vision software running on a modern ARM CPU. To that end, vision software is based entirely on the OpenCV library. OpenCV is a popular open source computer vision library which includes implementations of most common computer vision algorithms as well camera and file data IO handling. Version 2.42 was used on all computers used for testing, however, it was built natively from source on each system to enable any hardware-specific optimizations.

4.3 VISUAL CONTROL

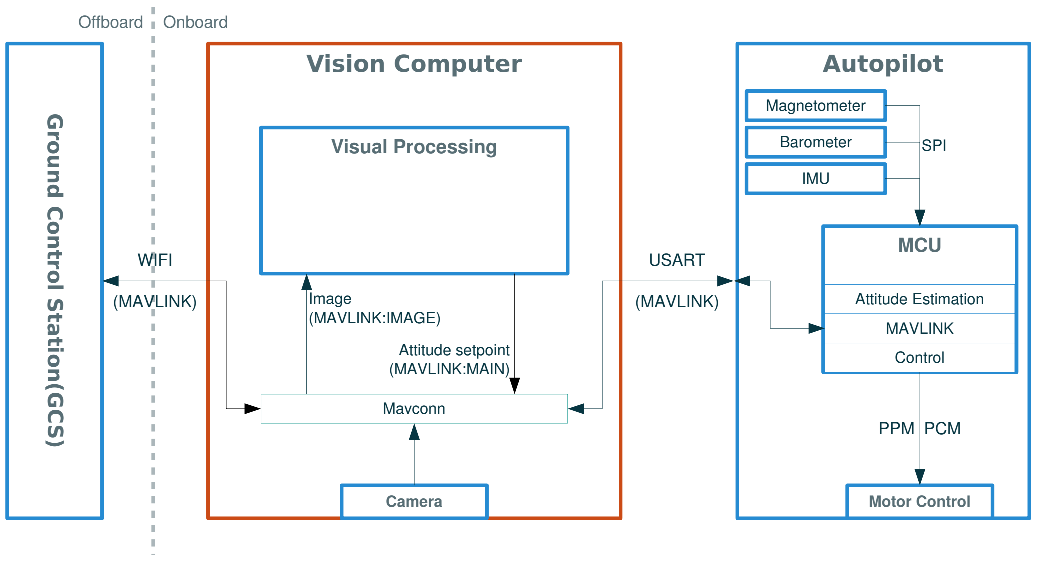


Figure 4.2: The architecture of the control software. The vision and communication software (the orange box) is the component which which this thesis is concerned with. Visual control software is implemented as a subscriber to the MAIN and IMAGE communication channels to access images and broadcast control commands. Control commands are either state changes (e.g. grounded to airborne) or attitude setpont packets.

4.3.1 Other Computer Vision Libraries

SimpleCV (SimpleCV, 2013) is a collection of libraries with a simplified Python interface. The vision libraries include OpenCV, it is intended for simplicity over performance.

BOOFCV (BoofCV, 2013) is a Java computer vision library which claims to outperform OpenCV in some areas.

UncannyCV (Uncanny Vision, 2013) is based on OpenCV with some enhancements for the ARM architecture. UncannyCV is commercial software, however a time-limited trial is available to students on request. The trial version imposes limits on the number of times certain key functions can be called in one session, limiting its uses.

FastCV (Qualcomm, 2013) is an ARM-optimized vision library developed by Qualcomm with specific optimizations for the Snapdragon line of processors.

4.4 AUTOPILOT SOFTWARE

Attitude control software runs on the autopilot with a controll loop rate of 250 Hz (Pixhawk, 2014). The quadrotor attitude is estimated from IMU measurements, and compared to the attitude setpoint. Motor control signals are generated by an independent PID controller for each axis of rotation, each of which consist of nested angle and angle rate controllers. The output is then scaled by the thrust input. The PID controllers try to minimize the error between the setpoint attitude, received from the onboard computer or from a human operator. This is achieved by controlling motor speeds to balance the thrust from each motor, and the opposing torques from each counter-rotating pair of rotors.

The PX4FMU runs the NuttX (NuttX, 2013) real-time operating system (RTOS), with the various controllers implemented at the application layer. A hardware abstraction layer provides a generic interface for each sensor on the board. Before flight the inertial sensors must be calibrated for their position relative to the chassis, and to remove any sensor bias. The sensor calibration process is described on Pixhawk wiki (Pixhawk, 2013b), and is likely to change with firmware updates. The PID control gains must also be tuned for custom quadrotor hardware, which exhibit different dynamic responses. The tuning process is based on observing the quadrotor response to manual attitude perturbations, and is described below. Quadrotors are highly unstable and under-actuated (Bouabdallah et al., 2004), and stable control depends on matching the controller to the dynamics of a particular quadrotor configuration.

4.4.1 Controller Tuning

The following tuning process for the proportional (P) and derivative (D) gains of the controller is summarized from the PX4FMU tuning guide (Pixhawk, 2013a):

1. Set all gains to zero.
2. Tune roll/pitch rates:

   Start with small P (e.g. 0.05).

   Hold quadrotor in hand with near-hovering thrust.

   Manually tilt around roll or pitch axis and observe the response.

   Increase P until mild oscillations are observed.

   Set D=0.3×P.

   Increase D until there are no oscillations.
3. Repeat #2 for yaw rate, rotating about the yaw axis instead of roll or pitch.
4. Tune roll/pitch angles.

   Start with small P (e.g. 0.05)

   Tilt about roll or pitch axis and let the quadrotor return to neutral attitude.

   Increase P until slight oscillations in the response.

   Start with D=0.3×P.

   Increase until no oscillations or overshoot are observed.

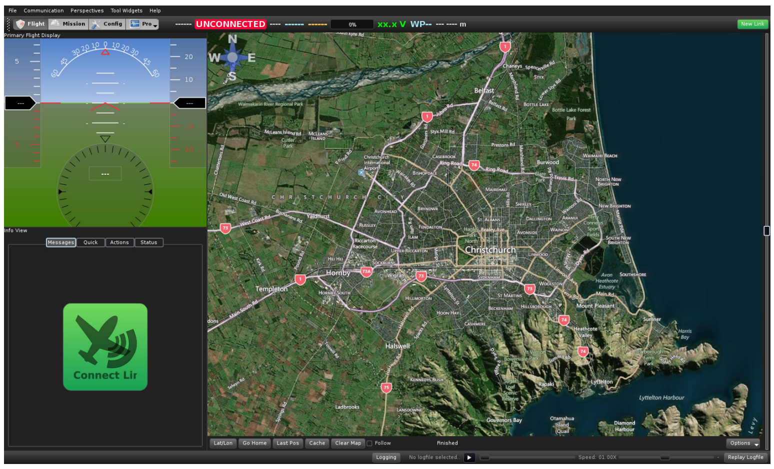


Figure 4.3: QGroundcontrol in flight plan mode. Additional functionality such as caliration, configuration, and direct control is provided by autopilot specific widgets.

5. Repeat #4 for yaw angle, rotating about the yaw axis instead of roll or pitch.

Because of the symmetry of multi-rotor UAVs the roll and pitch controllers use common gain values. It is also important to ensure that tuning is performed away from large metal structures, as these can interfere with the magnetometer.

4.4.2 Ground Control

The autopilot operator interface is called the ground control station (GCS). Ground control software displays telemetry and allows configuration and operator control from a remote computer. QGroundcontrol (Pixhawk, 2013d) is open source ground control software based on the original Pixhawk groundstation, and uses the MAVLINK protocol to receive telemetry and send commands to the autopilot. The flight-mode operator interface is shown in Figure 4.3, additional functionality such as IMU calibration and controller tuning is provided by widgets.

4.5 THE VISUAL CONTROL LOOP

The visual control loop is illustrated in Figure 4.4. Attitude estimates are broadcast by the autopilot and received by the subscribed message client. Mavconn stores images in a region of shared memory and broadcasts the image pointers on the IMAGE channel. Image clients receive the image pointers and retrieve the image from the shared memory for processing. On image retrieval the most recent attitude estimate is used as the current UAV attitude. The image and corresponding attitude estimate are then used by a visual processing module to generate a control output in the form of an attitude set point. This is then broadcast to the MAIN channel as an attitude control packet received by all other subscribers to MAIN, including the autopilot. The autopilot then

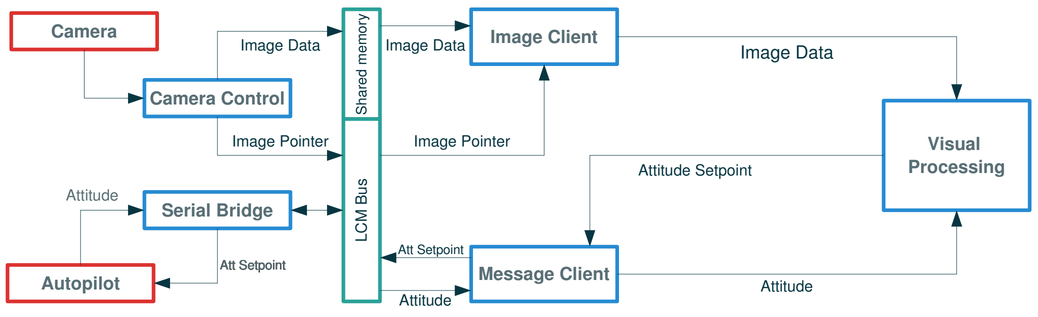


Figure 4.4: Visual control loop for control using pose estimation. Craft state is estimated visually from visual data and attitude estimates from the autopilot.

actuates the motors to minimize the error between the attitude setpoint and the IMU estimated attitude.

The delay between image capture and control output is important for stable control. Figure 4.5 shows the time delay introduced by each acquisition and processing stage. The USB transfer delay is fixed, and in (Meier et al., 2012) is measured for hardware triggered image capture. The configuration in used in Chapter 3 does not include external trigger hardware, and uses the camera in 'free running' mode, where the camera handles continually captures images and returns the most recent frame to a buffer.

4.6 SUMMARY

This section has presented the software libraries used to achieve control of a UAV. It has also shown that the most significant gains performance gains can be achieved by speeding up the visual processing software. Improvements to image acquisition and transmission, and to the control filtering, are either minimal or impossible. For the purpose of this thesis image acquisition and transmission are considered fixed delays.

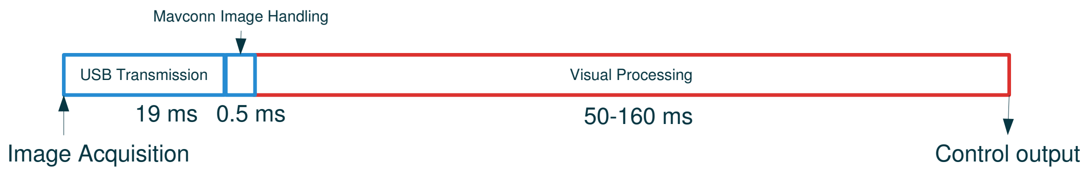


Figure 4.5: Delay from image acquisition to control output for one frame (not to scale). USB transfer and Mavconn hub delay are taken from Meier et al. (2012), and image processing time is for Chapter 5 running on the Odroid U2 onboard computer.

Chapter 5
CONTROL USING FEATURE TRACKING

Image features are used for tasks such as image registration (Zitov and Flusser, 2003, Goshtasby, 2012), pose estimation (Chaperon et al., 2011, Hmam and Kim, 2010), 3D reconstruction from multiple views (Liverani et al., 2010, Favalli et al., 2012, Varol et al., 2012), and they are a key component in SLAM algorithms (Steder et al., 2008, Artieda et al., 2009). In the context of autonomous control and navigation, image features provide a means for finding the point correspondences used for localizing a robot relative to known objects (Szeliski, 2011, Hartley and Zisserman, 2004).

This chapter presents a feature matching algorithm which relaxes the requirement for outlier rejection by tracking the location of features in image space, and minimizing the number of outliers based on a spatial constraint. The resulting set of point correspondences is used for pose estimation to perform visual servoing of the quadrotor described in Chapter 3. Section 5.1 reviews pose estimation from point correspondences and gives a method method for efficiently estimating the accuracy of the pose estimate. The RANSAC outlier rejection algorithm for pose estimation is reviewed in Section 5.2. Feature matching is presented in Section 5.3, together with the introduction of the tracking algorithm. The tracking method is intended to either replace the RANSAC stage of the pose estimation algorithm or improve its performance, the work in this chapter uses tracking to relax the RANSAC parameters. Performance and accuracy results of testing both algorithms are presented in Section 5.4 with some discussion, and the pose estimation algorithm is used for visual servoing of the quadrotor from Chapter 3. The algorithm is also used to demonstrate visual servoing using the quadrotor from Chapter 3. The chapter concludes with a summary of the results and a discussion of the limitations, advantages, and suggested improvements to the tracking algorithm in Section 5.5.

5.1 POSE ESTIMATION

Pose estimation algorithms solve the *perspective n-point* problem (PnP) to estimate the pose of the camera from a set of n point correspondences, where each correspondence consists of a 3D point in the world coordinate system and its 2D projection onto the image (Hartley and Zisserman, 2004). Point correspondences are found by extracting feature descriptors (the query points) from interest points in an image and finding the nearest matching descriptor from a set of reference features (the reference points). The reference features have known locations in the 3D world coordinate system. The pose estimation algorithm used in this work is the OpenCV implementation of the ePnP algorithm developed by Lepetit et al. (2009). ePnP was used because it is a non-iterative algorithm which offers robustness and accuracy similar similar to iterative methods, but at a reduced computational cost (Lepetit et al., 2009, Szeliski, 2011). ePnP can also be used with an arbitrary number of point correspondences provided that there are at least $n > 4$ correspondences. In contrast, many non-iterative algorithms which either require either a small fixed number of points or scale poorly as n increases. Compared to other non-iterative methods, ePnP is also more robust to noisy position measurements and has linear computational complexity ($O(n)$) compared $O(n^2)$ for the next fastest algorithm and $O(n^5)$ for the next best algorithm with comparable noise-immunity (Lepetit et al., 2009).

5.1.1 Estimating the Pose Accuracy

Due to quantization noise, illumination fluctuation, and other perturbations, it is impossible to measure the exact positions of either the reference or query features. Therefore, pose estimation algorithms find the best-fit pose for the given point correspondences. (Hartley and Zisserman, 2004) It is desirable to estimate how accurate the pose estimate is. The rigorous approach is to represent the reference and query points with Gaussian random variables with mean positions $\bar{\mathbf{p}}$ and covariance matrices $\mathbf{R}_\mathbf{p}$ (Voigt et al., 2011, Hartley and Zisserman, 2004). Hartley and Zisserman (2004) show that it is then possible to calculate an estimate of the pose accuracy using $\bar{\mathbf{p}}$ and $\mathbf{R}_\mathbf{p}$ for each feature. However, in most cases the large number of salient points makes calculating and storing the means and covariance of each point intractable (Hartley and Zisserman, 2004). An alternative is to use the reprojection error as an approximation of the pose accuracy (Voigt et al., 2011),

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} \|\mathcal{P}(\mathbf{p_i}) - \hat{\mathbf{p}}_\mathbf{i}\|, \tag{5.1}$$

where $\mathcal{P}(\mathbf{p_i})$ is the 2D projection of the 3D world point, $\mathbf{p_i}$, from the reference feature set, and $\hat{\mathbf{p}}_i$ is its matching 2D point from the query set. n is the total number of matches and the $\|\ \|$ operator computes the Euclidean norm. This is the metric used for outlier rejection, and is also used as an estimate of the pose estimate accuracy in the tracking algorithm.

5.2 RANSAC OUTLIER REJECTION

It is likely that the set of point pairs contains incorrect matches (called outliers) which make it impossible to estimate the pose accurately. Random sample consensus (Fischler and Bolles, 1981) is a general purpose outlier rejection algorithm commonly used for robust pose estimation in the presence of many outliers. The general approach is to repeatedly select a random subset of data points from the whole set and formulate a hypothesis based on the selected subset. The whole set is then compared with the hypothesis, and the data points which agree with the hypothesis (the inliers) are counted. This process is repeated with randomly selected subsets for a predetermined number of iterations, after which the algorithm returns the hypothesis corresponding to the largest group of inliers. The RANSAC algorithm for pose estimation can be summarized as follows:

```
1. From the set of all matches, $\mathbf{P}$, randomly select a subset $\mathbf{P}_R$
2. Compute pose $\mathbf{Q}$ from $\mathbf{P}_R$ (the hypothesis)
3. For all matches, $\mathbf{P}$:
       Find the per-point reprojection error $d_i$
       Compare $d_i$ to threshold $T$
         If $d_i < T$: Count as inlier
```

The reprojection error for each reference feature, d_i, is the metric used to test whether data points agree with the hypothesis

$$d_i = \|\mathcal{P}(\mathbf{p_i}) - \hat{\mathbf{p}}_\mathbf{i}\|. \tag{5.2}$$

Here $\mathcal{P}(\mathbf{p_i})$ is the projection of the reference point $\mathbf{p_i}$ using the pose estimate $\mathbf{Q}$ and $\hat{\mathbf{p}}_i$ is its matching query point.

A common variation to improve computational efficiency is to terminate the algorithm early when a sufficient number of inliers are found. While this approach is faster on average, the run time is non-deterministic. This variant is used in the OpenCV implementation of RANSAC pose estimation.

5.2.1 Minimum Iterations

The expected number of iterations for RANSAC to select a consensus set depends on the probability of selecting a good point from the consensus set and the number of inliers required for consensus,

$$E(k) = \omega^{-N}, \tag{5.3}$$

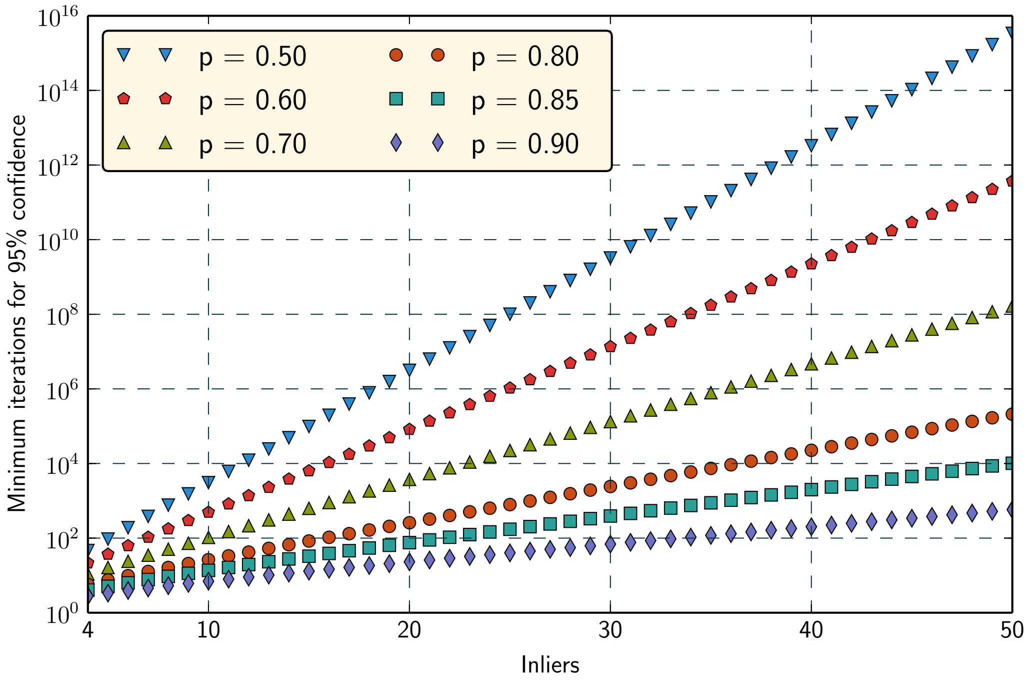


Figure 5.1: The number of iterations required for 95% confidence that at least one subset containing no oultiers has been selected. The graph was generated using (5.4) from Fischler and Bolles (1981) with $z = 0.95$ and selected values of $\omega = [0.5, 0.99]$. (5.4) implies that there are $n \geq \frac{N}{\omega}$ matches to sample from.

where k is the number of iterations required to reach a consensus of at least N inliers, with a probability ω of selecting a good point from the whole set $\mathbf{P}$. (5.3) implies that there are at least $n \geq \frac{N}{w}$ data points in total. For a confidence, z, that at least one subset of good points has been chosen, the minimum number of iterations is (Fischler and Bolles, 1981)

$$k = \frac{\log(1-z)}{\log(1-\omega^N)}, \tag{5.4}$$

where z is the confidence interval for finding the correct hypothesis. This equation is illustrated by Figure 5.1 for $z = 0.95$. (5.4) is used to determine the maximum number of iterations for which the RANSAC algorithm will run if a large enough consensus set is not found.

5.3 FEATURE MATCHING

Matching is the precursor to pose estimation, object recognition, stereoscopic triangulation, and most other point-based geometric operations (Szeliski, 2011). Feature correspondences are produced by comparing the descriptors of reference and query features and selecting the nearest pairs in descriptor space as matches (Szeliski, 2011, Hartley and Zisserman, 2004).

5.3.1 Match Criteria

There are several strategies for selecting positive matches and rejecting false ones. It is assumed that the distance between features in descriptor space correlates to feature similarity, and that the smaller the distance between descriptors the more likely they are to represent the same real point (Szeliski, 2011). The obvious approach is to select feature pairs with distances below a predetermined threshold as positive matches. However this raises the question of choosing the threshold, which varies between applications (Szeliski, 2011). This method can also find multiple matches for each query or reference feature. An alternative approach is to use the ratio of the first and second nearest neighbours in feature-space, and choose matches below a threshold. This has the advantage of favouring *unique* matches over features which similar to many other features.

The default strategy in OpenCV is just to return the nearest query descriptor to each reference descriptor. This is the strategy used in this work.

5.3.2 Brute-force Matching

The simplest approach is to compute the descriptor-space distance between each reference feature and each query feature, choose the nearest query descriptor to each reference descriptor. This is computationally expensive, with a computational complexity of (Hajjdiab and Laganire, 2009)

$$O(N_R N_Q H), \tag{5.5}$$

where H is the computational cost of matching one feature, and N_R and N_Q are the number of reference and query features respectively.

In this work binary descriptors are used in place of SIFT or SURF to reduce the H factor. To further improve efficiency only the N_Q and N_T factors can be altered, however it is usually undesirable to use fewer features, as this can come at a cost of lower accuracy and less robustness. Efficient alternatives to brute-force matching can be divided into two general categories (Szeliski, 2011): Hashing and index trees; and tracking methods. The first category uses efficient algorithms to search feature space, and the second uses search constraints in real world and image space.

This method can also produce incorrect matches with large reprojection errors. It can also match several reference descriptors to one query descriptor.

5.3.3 Indexed Trees and Hashing

One alternative to brute-force matching is to use an indexing structure, or hashing, to categorize the descriptors into bins. Query features are then compared to features in

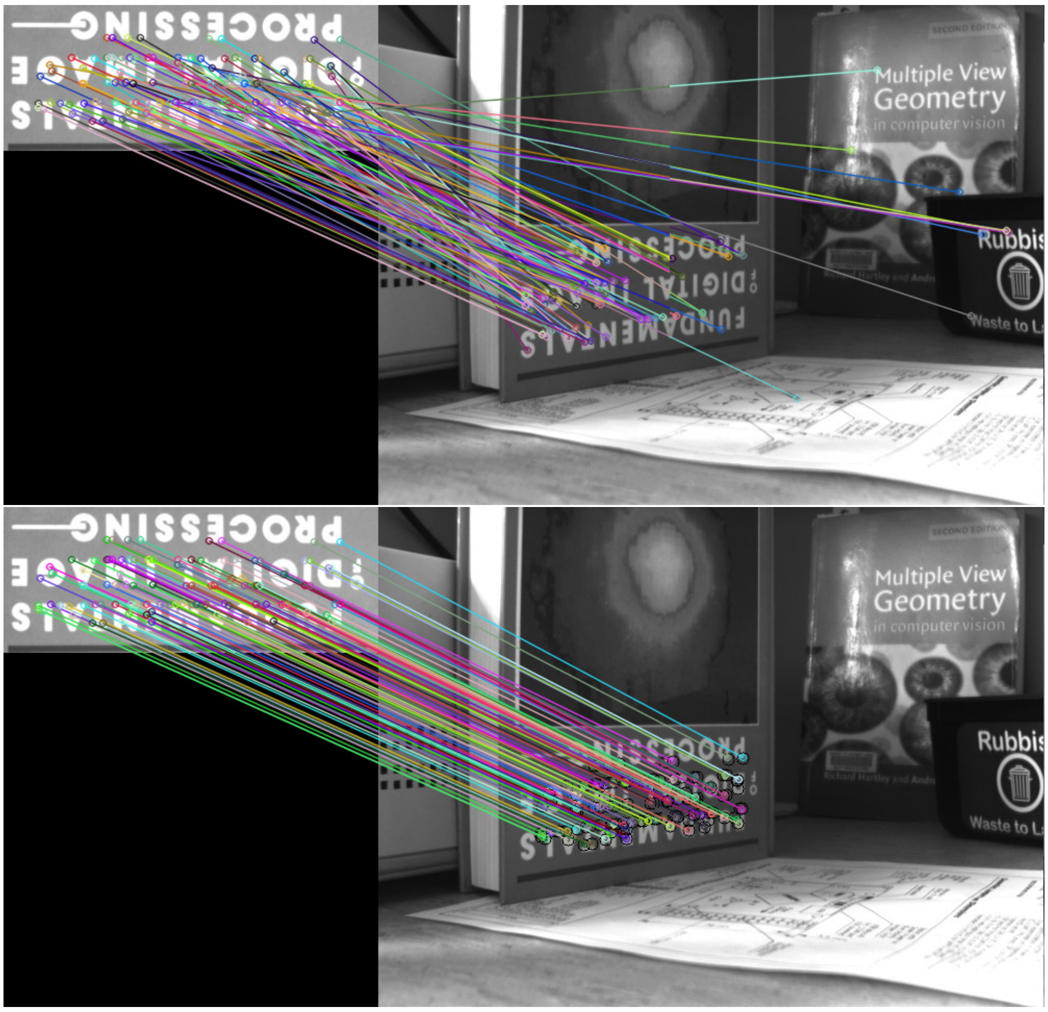


Figure 5.2: Point correspondences found by matching reference and query features. Matches are found between the reference image of the book title on the left, and features extracted in an image from the camera on the right with tracking (bottom) and without tracking (top). In the tracking example expected feature search regions are circled in black.

the nearest bins, rather than the whole set (Szeliski, 2011). In OpenCV the fast linear approximate nearest neighbour (FLANN) library is used for indexed matching (Muja and Lowe, 2009).

While these methods are efficient, their performance is often dependent on the properties of the data set (Muja and Lowe, 2009). Typically the algorithms include a training stage where the properties of the data set and some knowledge of feature distribution in descriptor space is learned. Indexing is therefore most effective when used on large datasets Szeliski (2011). For real-time computer vision a more suitable approach is to use spatial search constraints, particularly since the feature locations are likely to be known in both image space and world space.

This method is a faster way to perform brute-force matching, provided the same matching criteria is used. Incorrect matches can still occur with large reprojection errors, and several reference descriptors can be matched to one query descriptor.

5.3.4 Feature Tracking

Feature tracking can be used for efficient matching in image sequences provided they meet the following assumptions (Szeliski, 2011, Shi and Tomasi, 1994):

1. Most of the reference features are present in the query image.

2. Feature translation between sequential images is relatively small.

Both of these assumptions hold for visual servoing with a stationary marker.

Feature tracking algorithms use the locations of features in previous frames to reduce the feature search area in the current frame (Szeliski, 2011). Figure 5.3 illustrates the projection based algorithm used for feature tracking in this thesis, which uses the FAST and BRIEF feature detection and description algorithms respectively. Work performed by Dorini and Goldenstein (2010), and similar work by Voigt et al. (2011), has shown that imposing a spatial constraint on potential matches improves matching speed and robustness. Feature tracking limits potential matches based on the proximity of query features to the location of the reference feature in the previous frame, and is therefore expected to improve both robustness and processing speed. Voigt et al. (2011) used binocular vision and inertial cues to determine the region of permissible matches, and Dorini and Goldenstein (2010) focused on robust feature tracking in the general case. Hajjdiab and Laganire (2009) have shown that restricting the permissible matches based on their distance from the reference feature in an image reduces the complexity from (5.5) to

$$O(\rho N_R N_R H), \tag{5.6}$$

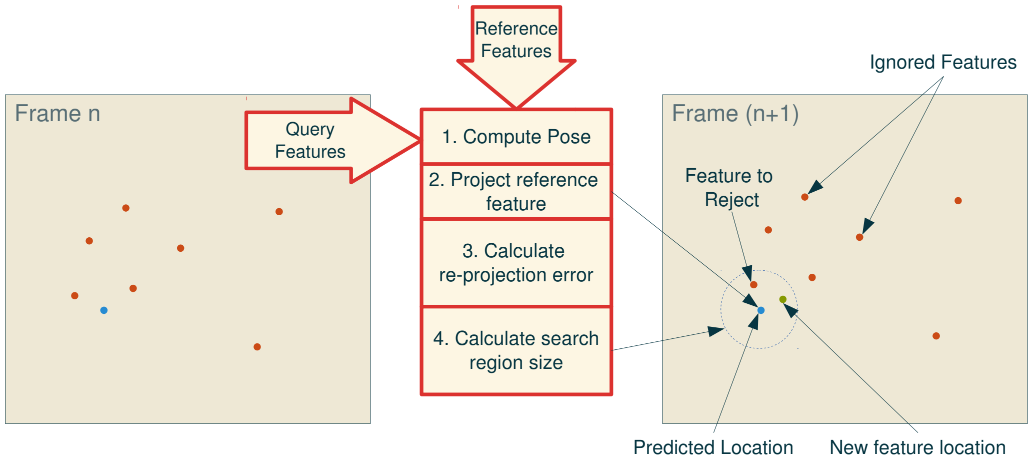


Figure 5.3: The matching algorithm ignores training features outside of the search radius for each feature query feature and the detection algorithm only searches within the search radius of the projection of each training feature. This improves matching performance by reducing ρ in (5.6). It further reduces feature search time by searching a smaller region of the image and by reducing the total number of query features N_Q.

where $\rho < 1$ is a factor defined by the spatial density of the query features and the size of the search region. Furthermore, feature detection efficiency is also improved only performing detection within the search regions.

To determine the size of the search region, the reprojection error from (5.1) is used as a measure of pose accuracy

$$r = \alpha + \sigma\delta, \tag{5.7}$$

where α is the base radius and δ is the scaling factor to apply to the reprojection error, σ, of the pose estimate from the previous frame. r is the radius around the projected reference feature, outside of which query features are ignored.

Based on equations (5.1) and (5.7) the tracking process is implemented as illustrated by Figure 5.3. The algorithm uses RANSAC pose estimation to find the initial estimate of the pose and to recover if tracking is lost. After initialization the tracking algorithm can be summarized as follows:

1. Project reference features onto frame $n + 1$ using the pose estimate from frame n.
2. Calculate the feature search and match region size using (5.7).
3. Construct an OpenCV mask of permissible matches using the location of each feature in frame n and the search region size from step 2.
4. Match features using the mask.

5. Update the pose and the reprojection error using (5.1).

6. Threshold σ to check if the marker is still visible:

   If visible: run from step 1 with the next frame.

   Else: re-initialize.

The mask for each reference feature is a Boolean vector with an entry for each query feature. The entry is true if the query feature from frame $n + 1$ is within the search radius of the reference position from frame n. In openCV the complete mask is a matrix of Boolean values of size $N_R \times N_T$. The above process is valid if the camera motion is slow relative to the frame rate; a valid assumption for visual servoing of a stationary marker.

The tracking approach will not produce matches with large reprojection errors, and introduces spatial matching criteria. In this way, similar features which are far apart in image space will not be considered potential matches, whereas in brute force or indexing they would be.

Table 5.1 shows the parameters used across both algorithms; note that the tracking method does no use RANSAC with relaxed parameters, which are tightened if the marker is lost. Tracking is also disabled if the marker is lost.

Value	**Description**
2000	Max number of iterations for RANSAC. Determined from Figure 5.1.
7	Inlier reprojection error threshold. Determined empirically.
300	Number of query features. Determined empirically.
80	Number of reference features. Determined empirically.
35	The number of inliers required for consensus. Determined empirically to match the proportion of inliers in Figure 5.4 for 80 matches.
30	Initial threshold for the FAST algorithm. The threshold is adaptive, but this was chosen as the threshold which found approximately 80 features in the reference features.
4.0	Reprojection error threshold for a lost marker
1000 cm	Z-translation jump for an incorrect pose estimate

Table 5.1: Parameters chosen for a RANSAC confidence of z=0.95, assuming approximatly 60% inliers determined empirically from Figure 5.4.

5.4 TESTING RESULTS

For visual quadrotor control the general consensus is that stable control is possible with a minimum visual control loop rate of 10 Hz (Stowers, 2013, Achtelik et al., 2011). A 10 Hz rate allows 100 ms to process each frame. During testing the processing time was measured from from image acquisition to pose estimate output, Figure 5.4 shows

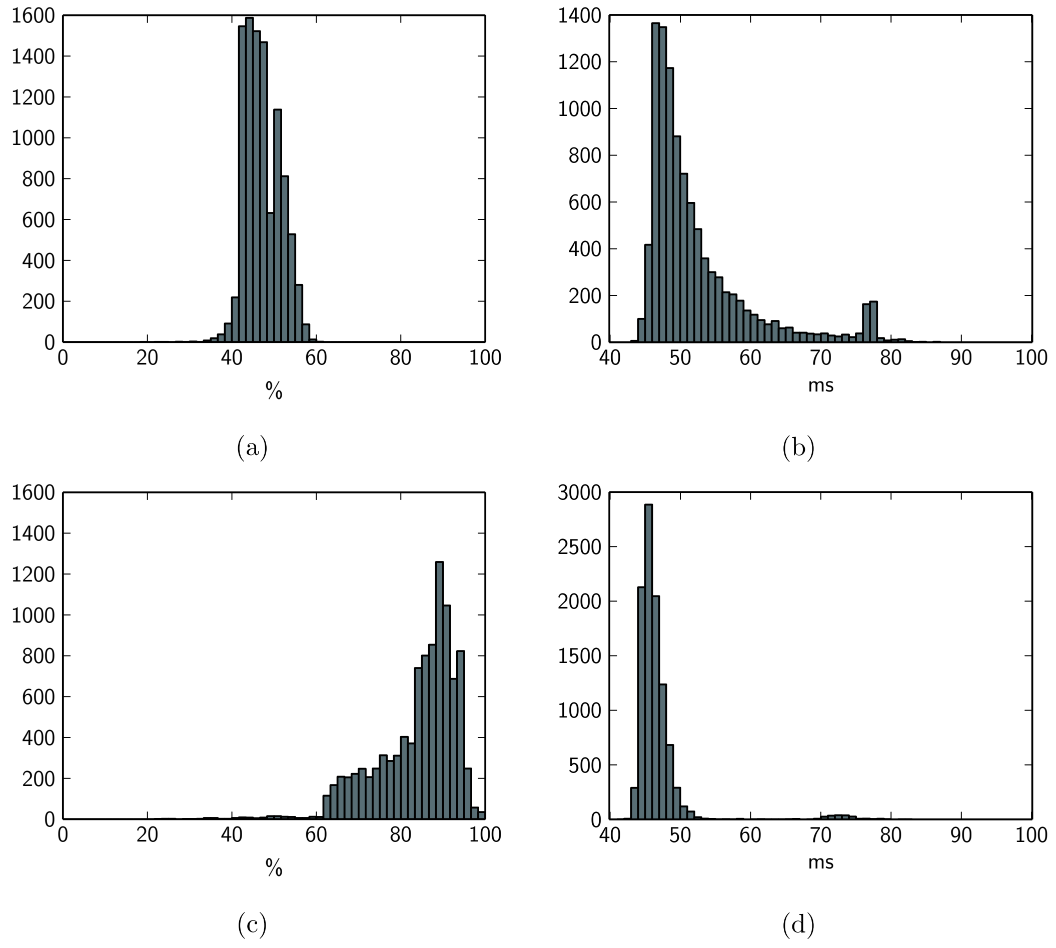


Figure 5.4: A comparison between RANSAC and tracking pose estimation. (a) and (c) show the percentage of inliers found during pose estimation stage, and (b) and (d) show the processing time per frame. The outlier groups in (c) and (d) are caused when the tracking algorithm resorts to RANSAC after loosing the marker or computing an inaccurate pose estimate.

a histogram of the frame processing times and inlier percentages for the RANSAC and tracking algorithms for a sequence of 10,000 images.

5.4.1 Processing Time

The mean processing time for the tracking algorithm is 46.7 ms, compared to the RANSAC algorithm with a mean time of 52.4 ms. The worst case times are similar (86 ms and 82 ms for RANSAC and tracking respectively), however approximately 5 % of the tracking frames required longer than 50 ms, while 47 % of the RANSAC frames did. This translates approximately to 95 % confidence that any tracking frame will be processed in under 50 ms, while the 95 % confidence processing time for RANSAC is 73 ms While the processing time was never more than the 100 ms limit, it is desirable to allow time for other tasks which might be performed for more complex control strategies. This extra time is required to avoid dropping frames and further increasing the delay between image acquisition and control output. Tracking fulfills this requirement by consistently using only half of the available time to perform some computationally intensive processing. RANSAC can only guarantee with 95 % confidence that it will not use more than 73 ms, for a given frame, offering about half as much time for other tasks when compared with the tracking algorithm.

Figure 5.4 also shows the inlier percentage for RANSAC and tracking algorithms. There is a clear improvement when tracking is enabled, however it was noted that the *number* of inliers remained the same (on average $\approx$60 inliers) but the number of matches was reduced due to tracking excluding most obvious false matches. The percentage of inliers was improved without reducing the total number of query features, which remained at 300. For RANSAC, on average, 47 % of the matches were inliers, while tracking increased the inlier percentage to 84 % and, ignoring outliers caused by returning to RANSAC, the minimum inlier percentage was 60% when feature tracking was used. From Figure 5.1 this improves required iterations for a 0.95 % confidence interval from about $O(10^{14})$ for about 47 % to $O(10^{3})$ for $\approx$60 %.

5.4.2 Pose Jitter

The pose jitter was measured for the same sequence of images, Figure 5.5 shows a histogram of the Z-axis component (the distance along the focal axis) of the pose estimate. The Z-axis component is used because for planar markers such as the one used for these tests, the Z-axis is the most variable component of the pose estimate. The data was recorded using the same image sequence and algorithm parameters as the performance data, however a second test where the algorithm was run 10,000 times on a static image from the sequence was also run. The second test was used to compare the jitter introduced by image noise to the jitter introduced by RANSAC.

The ground-truth pose has a Z-axis translation of 42 cm. For the image sequence the tracking average is rounded to 42.0 cm with a standard deviation of 0.2 cm while RANSAC gives an average of 42.4 cm with a standard deviation of 2.4 cm. For the static image the averages are the same, however tracking gives a standard deviation of 0 cm, while RANSAC has a standard deviation of 0.8 cm/ cm, RANSAC increases the standard deviation by a=1.6 cm. Furthermore the The jitter introduced by RANSAC is approximately four times the contribution from image noise (assuming standard deviation as a measure of jitter). Furthermore, noise and RANSAC jitter do not combine linearly because neither RANSAC or pose estimation are linear operations. The combined jitter of RANSAC and image noise is larger still.

5.4.3 System Load

In addition to processing speed and jitter the system load (for the Odroid SBC) was also measured using the Linux `top` utility. Both RANSAC and tracking consumed a maximum of approximately 25 % of available CPU time, including a running instance of Mavconn. This suggests that the Odroid has sufficient overhead to run similar computer vision algorithms simultaneously, and to provide more more complex behaviour in more cluttered environments.

5.4.4 Visual Servoing

Finally, the algorithm was run onboard the quadrotor and used to perform visual servoing after manual takeoff. Landing was also performed manually. Stable hovering was observed, however, external perturbations resulted in unstable behaviour which required manual intervention to correct.

The poor response perturbations is likely due to a combination of suboptimal visual controller parameters and variable delay due to acquiring images in 'free running' mode rather than using triggering. Free running was used for hardware simplicity, however this means that rather than an accurately timestamped image with a corresponding attitude estimate, the algorithm operates on the most recent image and attitude estimate. Therefore the attitude estimate might differ from the actual attitude at which the image was captured.

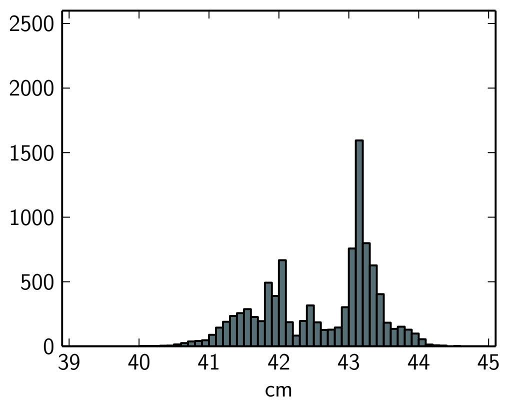
5.5 LIMITATIONS AND FUTURE WORK

The tracking algorithm relies on the pose estimate from the previous frame to compute the matching constraint. This imposes a constraint on the maximum translation between frames, and therefore the maximum camera speed. This is an undesirable constraint for UAV navigation where the camera is in motion. In its original form the algorithm handles small camera movements by simply increasing the search region, however for large movements, the search region would need to be so large that the advantage of the tracking algorithm would be lost.

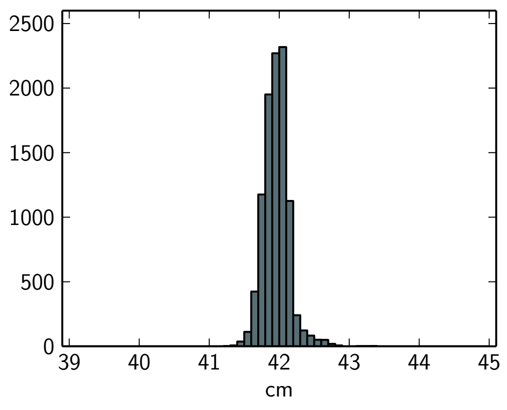
An alternative strategy is to measure the feature velocity in image space between the previous two frames, and use the velocity to translate the search region.

$$\bar{\mathbf{p}}_{\mathbf{i}} = \mathcal{P}(\mathbf{p_i})_{\mathbf{i-1}} - \mathcal{P}(\mathbf{p_i})_{\mathbf{i-2}}, \tag{5.8}$$

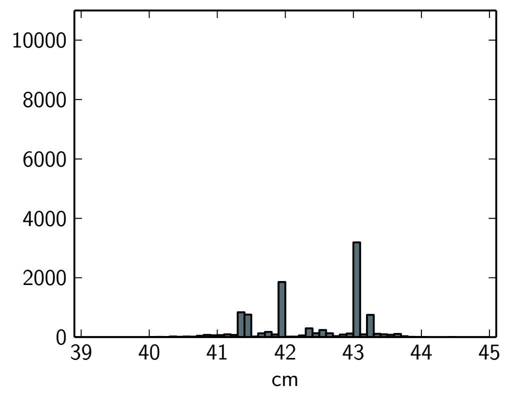
where $\mathcal{P}()_{i-1}$ and $\mathcal{P}()_{i-2}$ are the projection operations using the pose from the previous frame and the frame before it respectively. The new search region is around $\bar{\mathbf{p}}_{\mathbf{i}}$. This approach is based on the assumption that camera motion, including rotation, is continuous and therefore feature velocity in the image space must also be continuous.



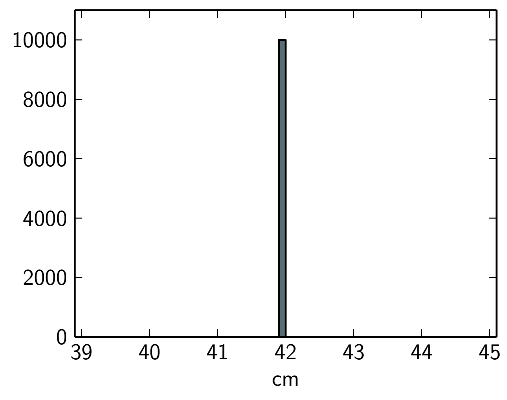
(a)



(b)



(c)



(d)

Figure 5.5: Histogram of the Z-translation of the pose estimate for a fixed camera pose at z=42 cm. (a) and (b) were recorded from the pose computed over a sequence of 10^4 images for RANSAC and tracking pose estimation respecticely. (c) and (d) were generated using data recorded by estimating the pose 10^4 times from a static reference image, also for RANSAC and tracking. The difference between the top and bottom histograms is jitter due to noise present in the image and in the feature locations, while the difference between left and right histograms is the jitter introduced by RANSAC selecting different subsets to compute the hypothesis from.

A more complicated model could be derived from the dynamics of quadrotor motion, however this simple approach may be sufficient.

Camera motion also alters the scale and rotation of the query features, whereas with a stationary camera these are approximately constant. To handle this aspect of camera motion two improvements are suggested: Use of descriptor with scale and rotation invariance, and periodically updating the reference feature descriptors as the pose changes. ORB is a scale and rotation invariant feature descriptor and detector which is based on BRIEF, and uses Gaussian pyramids for scale invariance. This is an ideal replacement for BRIEF with the current tracking algorithm. The FREAK descriptor is also a potential replacement, however its particular advantage is in the coarse-to-fine saccadic matching which may diminish if used in conjunction with tracking.

Another improvement is be to use RANSAC after coarse outlier exclusion by feature tracking. This would improve robustness, and due to the high percentage of inliers after tracking, come at minimal extra computational cost.

Finally there is no temporal filtering of the pose estimate, aside from checking for impossibly large discontinuities. The addition of Kalman filtering, or some similar filtering strategy, is expected to improve the robustness of the visual control algorithm.

5.6 SUMMARY

This chapter has presented a pose estimation algorithm which runs in real-time on a small onboard computer. This has served as a proof-of-concept for small ARM onboard computers for real-time processing of general purpose computer vision algorithms. However, the system is limited to situations where the camera is approximately stationary relative to the marker. Several suggestions were made to extend the pose estimation algorithm to allow for camera motion.

Suggestions were also made as to improving the robustness of the visual controller to external perturbations of the quadrotor pose and attitude.

A feature tracking algorithm as used in place of RANSAC outlier rejection to reduce the variability in both the pose estimate and the processing frame rate. However, the tracking approach imposes some limitations on the movement allowed between frames. Two improvements were suggested to better handle large camera displacements between frames, and feature scaling due to camera movement.

Chapter 6
CONCLUSION

The aim of this thesis was determine the suitability of small, lightweight ARM computers for real-time visual control of a quadrotor. To demonstrate this a quadrotor was built using the PX4FMU autopilot from the Pixhawk project with generic multi-rotor hardware. This was paired with the Odroid-U2 single board computer for onboard processing, and a single downwards facing camera. A bespoke power supply PCB was designed based on the Paparazzi quadrotor power distribution board. The power control circuit required manual arming of the motor drivers for safety reasons, and to allow testing of the avionics hardware without risking accidental motor actuation.

An efficient feature tracking algorithm was developed and used to demonstrate the real-time capability of the Odroid. The performance of the tracking algorithm was compared to that of a RANSAC algorithm, with both used for pose estimation. Tracking was found to produce less variable results with consistently shorter processing times per frame than the RANSAC approach. However RANSAC was used to compute the initial pose for the tracking algorithm, and to re-initialize tracking if the marker was lost. Tracking was used as the basis for pose estimation, which was demonstrated by performing visual servoing of the quadrotor.

The quadrotor, onboard computer, and the tracking algorithm have served as a proof of concept that it is possible to process visual data in real time using small ARM single board computers. However some improvements are suggested for both the tracking algorithm and the quadrotor platform, and further testing is required in more cluttered environments with more complex control strategies.

6.1 RECOMMENDATIONS

Based on the author's experience, a much more attractive path for experiments in visual control is to use a ready-to-fly quadrotor rather than spending several months building one. Furthermore onboard processing is not necessary in a research context, as most experiments will be carried out indoors, within range of high-speed wireless network connections.

An ideal base platform for such work is the Parrot AR.Drone, which solves most of the quadrotor hardware problems and supports external control over WiFi via the nodecopter library. The native controller provides autonomous behaviour such as position and altitude hold, speed and altitude limits, and can even perform ‘flips’. The AR.Drone is also a low-cost platform, with a retail price of NZ$450-500 at the time of writing.

The recommended approach for the general case of experiments in visual control of multi-rotor UAVs is to perform initial testing and validation using offboard processing with a robust platform such as the AR.Drone.

Appendix A
PROJECTIVE GEOMETRY

Projective geometry is fundamental the projective operations common in computer vision. It is an extension of Euclidean geometry where an extra coordinate is added, and used to normalize the point

$$\mathbf{p} = [x, y, 1]^T \tag{A.1}$$

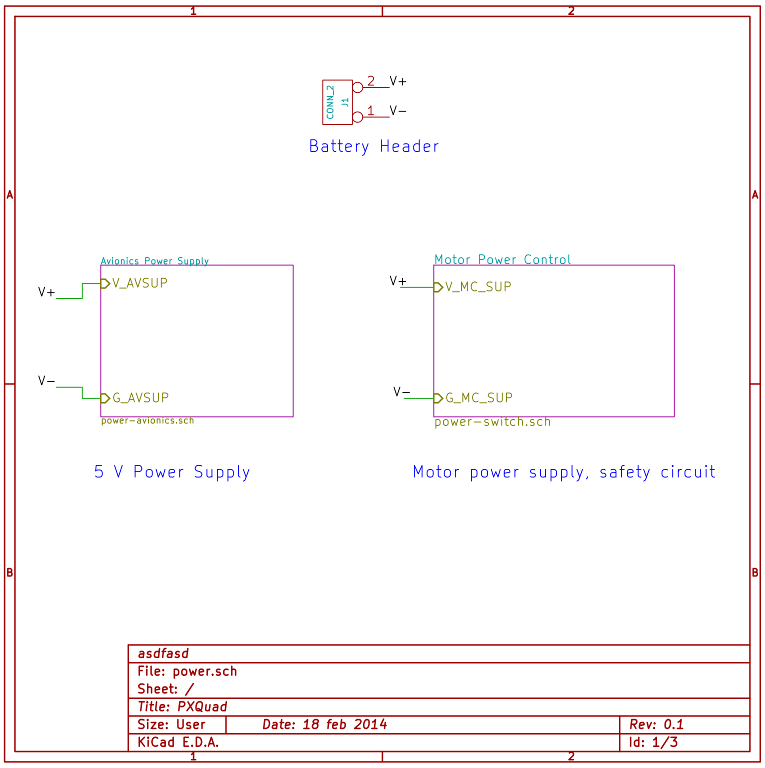
where x and y are the components of the Euclidean point $[x, y]^T$ and are scaled so that the third component[1] is always equal to 1. This expression of coordinates (called *homogeneous coordinates*) allows the expressing of points 'at infinity' by setting the last coordinate to zero

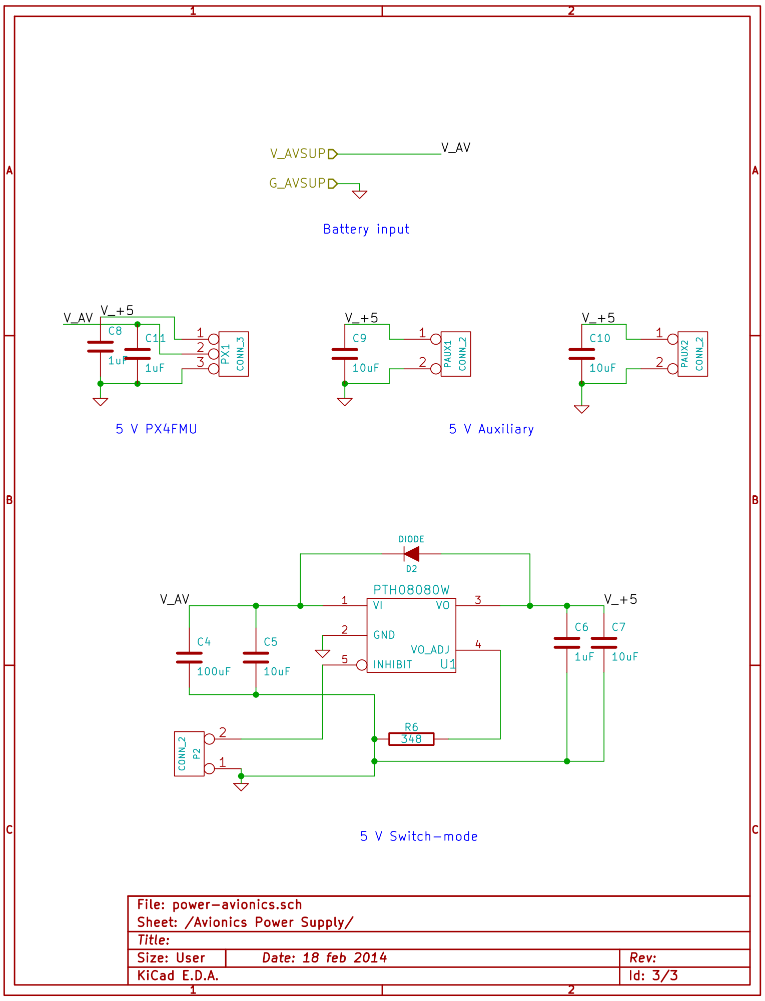
$$\mathbf{p}_\infty = [x, y, 0]^T, \tag{A.2}$$

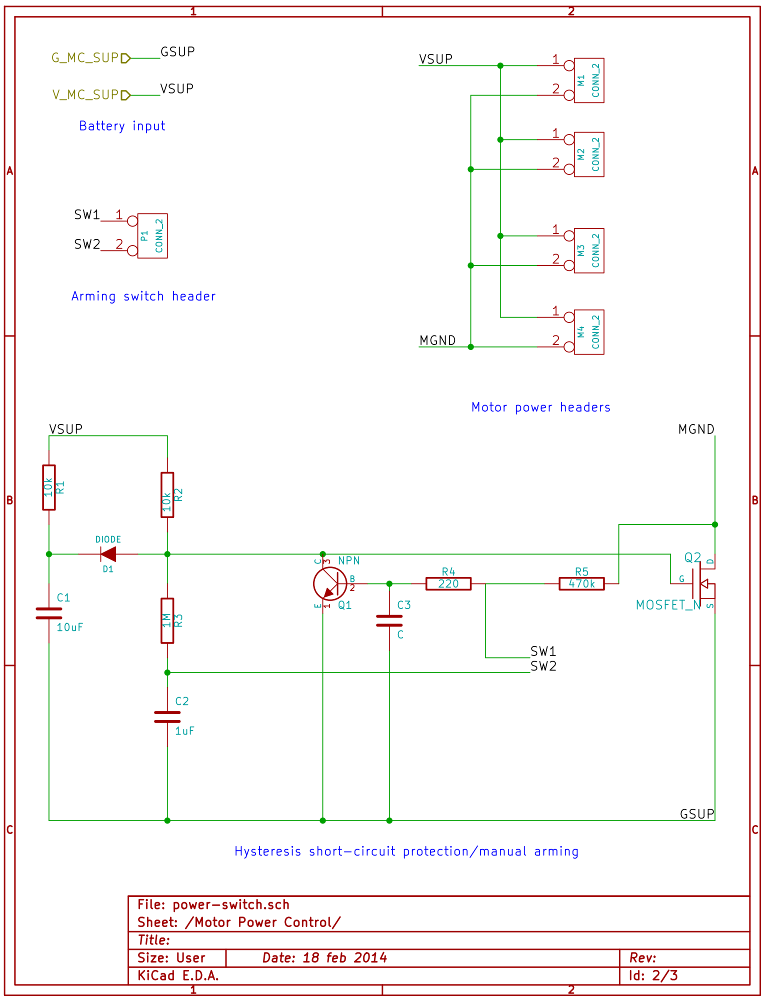
which allows parallel lines to meet in projective space. For example, parallel lines in 3D space (e.g. train tracks) appear to meet when projected onto 2D space (e.g. the point on the horizon where train tracks appear to converge).

[1] The examples here are for 2D geometry, however these are easily generalized to higher dimensions

Appendix B
AVIONICS SUPPORT HARDWARE




V_AVSUP
V_AV
G_AVSUP
Battery input
V_AV
V_+5
C8
1uF
C11
1uF
PX1
CONN_3
5 V PX4FMU
V_+5
C9
10uF
PAUX1
CONN_2
V_+5
C10
10uF
PAUX2
CONN_2
5 V Auxiliary
DIODE
D2
PTH08080W
VI
VO
GND
VO_ADJ
INHIBIT
U1
V_AV
C4
100uF
C5
10uF
V_+5
C6
1uF
C7
10uF
R6
348
CONN_2
P2
5 V Switch−mode
File: power−avionics.sch
Sheet: /Avionics Power Supply/
Title:
Size: User
Date: 18 feb 2014
Rev:
KiCad E.D.A.
Id: 3/3


G_MC_SUP
GSUP
V_MC_SUP
VSUP
Battery input
SW1 1
SW2 2
P1 CONN_2
Arming switch header
VSUP
MGND
M1 CONN_2
M2 CONN_2
M3 CONN_2
M4 CONN_2
Motor power headers
VSUP
MGND
10k R1
10k R2
DIODE D1
C1 10uF
1M R3
C2 1uF
NPN Q1
C3 C
R4 220
R5 470k
Q2 MOSFET_N
SW1
SW2
GSUP
Hysteresis short-circuit protection/manual arming
File: power-switch.sch
Sheet: /Motor Power Control/
Title:
Size: User
Date: 18 feb 2014
Rev:
KiCad E.D.A.
Id: 2/3

REFERENCES

Markus Achtelik, Abraham Bachrach, Ruijie He, Samuel Prentice, and Nicholas Roy. Stereo vision and laser odometry for autonomous helicopters in gps-denied indoor environments. In *SPIE Defense, Security, and Sensing*, pages 733219–733219. International Society for Optics and Photonics, 2009.

Markus Achtelik, Stephan Weiss, and R Siegwart. Onboard imu and monocular vision based control for mavs in unknown in-and outdoor environments. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 3056–3063. IEEE, 2011.

Gilad Adiv. Determining three-dimensional motion and structure from optical flow generated by several moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, PAMI-7(4):384–401, 1985.

Erdinç Altuğ, James P Ostrowski, and Robert Mahony. Control of a quadrotor helicopter using visual feedback. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, pages 72–77. IEEE, 2002.

Erdinç Altuğ, James P Ostrowski, and Camillo J Taylor. Quadrotor control using dual camera visual feedback. In *Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International Conference on*, volume 3, pages 4294–4299. IEEE, 2003.

Erdinç Altuğ, James P Ostrowski, and Camillo J Taylor. Control of a quadrotor helicopter using dual camera visual feedback. *The International Journal of Robotics Research*, 24(5):329–341, 2005.

Omead Amidi. *An autonomous vision-guided helicopter.* PhD thesis, Carnegie Mellon University, 1996.

Omead Amidi, Yuji Masaki, and Takeo Kanade. Research on an autonomous vision-guided helicopter. *Robotics Institute*, page 19, 1993.

Jonathan Andersh and Bernie Mettler. System integration of a miniature rotorcraft for aerial tele-operation research. *Mechatronics*, 21(5):776 – 788, 2011.

Jorge Artieda, Jos M. Sebastian, Pascual Campoy, Juan F. Correa, Ivn F. Mondragn, Carol Martnez, and Miguel Olivares. Visual 3D SLAM from UAVs. *Journal of Intelligent and Robotic Systems*, 55(4-5):299–321, 2009.

G. Astuti, G. Giudice, D. Longo, C.D. Melita, G. Muscato, and A. Orlando. An overview of the 'volcan project': An UAS for exploration of volcanic environments. *Journal of Intelligent and Robotic Systems*, 54(1-3):471–494, 2009. ISSN 0921-0296. doi: 10.1007/s10846-008-9275-9. URL http://dx.doi.org/10.1007/s10846-008-9275-9.

Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (SLAM): Part ii. *Robotics & Automation Magazine, IEEE*, 13(3):108–117, 2006.

J.L. Barron, D.J. Fleet, and S.S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994. ISSN 0920-5691. doi: 10.1007/BF01420984. URL http://dx.doi.org/10.1007/BF01420984.

Adam Baumberg. Reliable feature matching across widely separated views. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 1, pages 774–781. IEEE, 2000.

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). In *In ECCV*, pages 404–417, 2006.

S. S. Beauchemin and J. L. Barron. The computation of optical flow. *ACM Comput. Surv.*, 27(3):433–466, September 1995.

Cooper Bills, Joyce Chen, and Ashutosh Saxena. Autonomous mav flight in indoor environments using single image perspective cues. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 5776–5783. IEEE, 2011.

Michael Blosch, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Vision based mav navigation in unknown and unstructured environments. In *Robotics and automation (ICRA), 2010 IEEE international conference on*, pages 21–28. IEEE, 2010.

Alexander Borst, Juergen Haag, and Dierk F Reiff. Fly motion vision. *Annual review of neuroscience*, 33:49–70, 2010.

Tom Botterill. *Visual Navigation for Mobile Robots using the Bag-of-Words Algorithm*. PhD thesis, University of Canterbury, 2010.

Samir Bouabdallah. Design and control of quadrotors with application to autonomous flying. *Ecole Polytechnique Federale de Lausanne*, 2007.

Samir Bouabdallah and Roland Siegwart. Full control of a quadrotor. In *Intelligent robots and systems, 2007. IROS 2007. IEEE/RSJ international conference on*, pages 153–158. IEEE, 2007.

Samir Bouabdallah, Pierpaolo Murrieri, and Roland Siegwart. Design and control of an indoor micro quadrotor. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 5, pages 4393–4398. IEEE, 2004.

Odile Bourquardez, Robert Mahony, Nicolas Guenard, François Chaumette, Tarek Hamel, and Laurent Eck. Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle. *Robotics, IEEE Transactions on*, 25(3):743–749, 2009.

G. Bradski and A. Kaeher. *Learning OpenCV*. O'Reilly Media Inc, first edition, 2008.

Jack Bresenham. A linear algorithm for incremental digital display of circular arcs. *Commun. ACM*, 20(2):100–106, February 1977. ISSN 0001-0782. doi: 10.1145/359423.359432. URL http://doi.acm.org/10.1145/359423.359432.

Pascal Brisset, Antoine Drouin, Michel Gorraz, Pierre-Selim Huard, and Jeremy Tyler. The paparazzi solution. *MAV2006, Sandestin, Florida*, 2006.

Pierre-Jean Bristeau, François Callou, David Vissière, Nicolas Petit, et al. The navigation and control technology inside the ar. drone micro uav. In *18th IFAC World Congress*, pages 1477–1484, 2011.

Matthew Brown and David G. Lowe. Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1):59–73, 2007.

Mitch Bryson and Salah Sukkarieh. Building a robust implementation of bearing-only inertial SLAM for a UAV. *Journal of Field Robotics*, 24(1-2):113–143, 2007.

Fernando Caballero, Luis Merino, Joaquin Ferruz, and Aníbal Ollero. Vision-based odometry and SLAM for medium and high altitude flying uavs. In *Unmanned Aircraft Systems*, pages 137–161. Springer, 2009.

M. Calonder, V. Lepetit, M. Ozuysal, T. Trzcinski, C. Strecha, and P. Fua. BRIEF: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2011.

Jason Campbell, Rahul Sukthankar, and Illah Nourbakhsh. Techniques for evaluating optical flow for visual odometry in extreme terrain. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 4, pages 3704–3711. IEEE, 2004.

LR García Carrillo, Eduardo Rondon, Anand Sanchez, Alejandro Dzul, and Rogelio Lozano. Stabilization and trajectory tracking of a quad-rotor using vision. *Journal of Intelligent & Robotic Systems*, 61(1-4):103–118, 2011.

Luis Rodolfo García Carrillo, Alejandro Enrique Dzul López, Rogelio Lozano, and Claude Pégard. Combining stereo vision and inertial navigation system for a quad-rotor uav. *Journal of Intelligent & Robotic Systems*, 65(1-4):373–387, 2012.

D. E. Catlin. *Estimation, Control, and the Discrete Kalman Filter*. Number v. 71 in Applied Mathematical Sciences. Springer, 1988. ISBN 9780387967776.

HaiYang Chao, YongCan Cao, and YangQuan Chen. Autopilots for small unmanned aerial vehicles: a survey. *International Journal of Control, Automation and Systems*, 8(1):36–44, 2010.

Thomas Chaperon, Jacques Droulez, and Guillaume Thibault. Reliable camera pose and calibration from a small set of point and line correspondences: A probabilistic approach. *Computer Vision and Image Understanding*, 115(5):576–585, 2011.

Y. Cohen, V. Alchanatis, A. Prigojin, A. Levi, and V. Soroker. Use of aerial thermal imaging to estimate water status of palm trees. *Precision Agriculture*, 13(1):123 – 140, 2012. A good overview of recent efforts to automate sensing in agriculture.

Jonathan Courbon, Youcef Mezouar, Nicolas Guenard, and Philippe Martinet. Visual navigation of a quadrotor aerial vehicle. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 5315–5320. IEEE, 2009.

Andrew J. Davison, Ian D. Reid, Nicholas D. Molton, and Olivier Stasse. MonoSLAM: Real-time single camera SLAM. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(6): 1052–1067, 2007.

Guilherme N DeSouza and Avinash C Kak. Vision for mobile robot navigation: A survey. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(2): 237–267, 2002.

Ernst D Dickmanns. A general dynamic vision architecture for ugv and uav. *Applied Intelligence*, 2(3):251–270, 1992.

Michael Dille, Ben Grocholsky, and Sanjiv Singh. Outdoor downward-facing optical flow odometry with commodity sensors. In *Field and Service Robotics*, pages 183–193. Springer, 2010.

Patrick Doherty and Piotr Rudol. A uav search and rescue scenario with human body detection and geolocalization. In *AI 2007: Advances in Artificial Intelligence*, pages 1–13. Springer, 2007.

L. Dorini and S. Goldenstein. Unscented feature tracking. *Computer Vision and Image Understanding*, 115(1):8–15, 2010.

Xin Du, Hongdong Li, and Yunfang Zhu. Camera lens radial distortion correction using two-view projective invariants. *Opt. Lett.*, 36(24):4734–4736, Dec 2011. doi: 10.1364/OL.36.004734. URL http://ol.osa.org/abstract.cfm?URI=ol-36-24-4734.

S. Duncan, M. Hayes, and A. Bainbridge-Smith. Efficient feature based visual servoing for onboard computer vision. In *Electronics New Zealand*, 2012.

Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *Robotics & Automation Magazine, IEEE*, 13(2):99–110, 2006.

Sara Erhard, KarlE. Wenzel, and Andreas Zell. Flyphone: Visual self-localisation using a mobile phone as onboard image processor on a quadrocopter. *Journal of Intelligent and Robotic Systems*, 57(1-4):451–465, 2010. ISSN 0921-0296. doi: 10.1007/s10846-009-9360-8. URL http://dx.doi.org/10.1007/s10846-009-9360-8.

Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Image Analysis*, pages 363–370. Springer, 2003.

M. Favalli, A. Fornaciai, I. Isola, S. Tarquini, and L. Nannipieri. Multiview 3d reconstruction in geosciences. *Computers Geosciences*, 44(0):168–176, 2012.

Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981.

Spencer G Fowers, Dah-Jye Lee, Beau J Tippetts, Kirt D Lillywhite, Aaron W Dennis, and James K Archibald. Vision aided stabilization and the development of a quadrotor micro uav. In *Computational Intelligence in Robotics and Automation, 2007. CIRA 2007. International Symposium on*, pages 143–148. IEEE, 2007.

Xiao-Shan Gao, Xiao-Rong Hou, Jianliang Tang, and Hang-Fei Cheng. Complete solution classification for the perspective-three-point problem. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(8):930–943, 2003.

Joel George, Sujit P. B., and J.B. Sousa. Search strategies for multiple UAV search and destroy missions. *Journal of Intelligent & Robotic Systems*, 61(1-4):355–367, 2011. ISSN 0921-0296. doi: 10.1007/s10846-010-9486-8. URL http://dx.doi.org/10.1007/s10846-010-9486-8.

J. J. Gibson. *The Perception of the Visual World*. Houghton Mifflin, 1950.

AliHaydar Goktogan, Salah Sukkarieh, Mitch Bryson, Jeremy Randle, Todd Lupton, and Calvin Hung. A rotary-wing unmanned air vehicle for aquatic weed surveillance and management. *Journal of Intelligent and Robotic Systems*, 57(1-4):467–484, 2010. ISSN 0921-0296. doi: 10.1007/s10846-009-9371-5. URL `http://dx.doi.org/10.1007/s10846-009-9371-5`.

J. Gomez-Balders, P. Castillo, J. Guerrero, and R. Lozamo. Vision based tracking for a quadrotor using vanishing points. *Journal of Intelligent Robot Systems*, 65(1-4): 361–371, 2012.

Michael A Goodrich, Bryan S Morse, Damon Gerhardt, Joseph L Cooper, Morgan Quigley, Julie A Adams, and Curtis Humphrey. Supporting wilderness search and rescue using a camera-equipped mini uav. *Journal of Field Robotics*, 25(1-2):89–110, 2008.

A. Goshtasby. *Image Registration*. Springer, 2012.

William E Green and Paul Y Oh. Optic-flow-based collision avoidance. *Robotics & Automation Magazine, IEEE*, 15(1):96–103, 2008.

Nicolas Guenard, Tarek Hamel, and Robert Mahony. A practical visual servo control for an unmanned aerial vehicle. *Robotics, IEEE Transactions on*, 24(2):331–340, 2008.

JoseAlfredo Guerrero and Yasmina Bestaoui. UAV path planning for structure inspection in windy environments. *Journal of Intelligent & Robotic Systems*, 69 (1-4):297–311, 2013. ISSN 0921-0296. doi: 10.1007/s10846-012-9778-2. URL `http://dx.doi.org/10.1007/s10846-012-9778-2`.

Hassan Hajjdiab and Robert Laganire. Complexity analysis of featured-based image matching. *World Academy of Science, Engineering and Technology*, 51:281–284, 2009.

Jinlu Han, Yaojin Xu, Long Di, and YangQuan Chen. Low-cost multi-UAV technologies for contour mapping of nuclear radiation field. *Journal of Intelligent & Robotic Systems*, 70(1-4):401–410, 2013. ISSN 0921-0296. doi: 10.1007/s10846-012-9722-5. URL `http://dx.doi.org/10.1007/s10846-012-9722-5`.

Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Manchester, UK, 1988.

R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2004. ISBN 9780521540513.

R.I. Hartley. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(6):580–593, 1997. ISSN 0162-8828. doi: 10.1109/34.601246.

Ruijie He, Sam Prentice, and Nicholas Roy. Planning in information space for a quadrotor helicopter in a gps-denied environment. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 1814–1820. IEEE, 2008.

Bruno Hérissé, Tarek Hamel, Robert Mahony, and François-Xavier Russotto. A terrain-following control approach for a VTOL unmanned aerial vehicle using average optical flow. *Autonomous Robots*, 29(3-4):381–399, 2010.

SR Herwitz, LF Johnson, SE Dunagan, RG Higgins, DV Sullivan, J Zheng, BM Lobitz, JG Leung, BA Gallmeyer, M Aoyagi, et al. Imaging from an unmanned aerial vehicle: agricultural surveillance and decision support. *Computers and electronics in agriculture*, 44(1):49–61, 2004.

Stanley R Herwitz, Lee F Johnson, JC Arvesen, Robert Higgins, Joseph Leung, and Stephen Dunagan. Precision agriculture as a commercial application for solar-powered unmanned aerial vehicles. In *AIAA 1st Technical Conference and Workshop on Unmanned Aerospace Vehicles*, 2002.

Jeffrey Hightower and Gaetano Borriello. A survey and taxonomy of location systems for ubiquitous computing. *IEEE computer*, 34(8):57–66, 2001.

Hatem Hmam and Jijoong Kim. Optimal non-iterative pose estimation via convex relaxation. *Image and Vision Computing*, 28(11):1515 – 1523, 2010.

Gabriel M Hoffmann, Haomiao Huang, Steven L Waslander, and Claire J Tomlin. Quadrotor helicopter flight dynamics and control: Theory and experiment. In *Proc. of the AIAA Guidance, Navigation, and Control Conference*, pages 1–20, 2007.

Dominik Honegger, Lorenz Meier, Petri Tanskanen, and Marc Pollefeys. An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications. In *Intl. Conf. Robotics and Automation*, 2013.

Radu Horaud, Bernard Conio, Olivier Leboulleux, and Bernard Lacolle. An analytic solution for the perspective 4-point problem. *Computer Vision, Graphics, and Image Processing*, 47(1):33–44, 1989.

B. Horn. *Robot Vision*. Mit Electrical Engineering and Computer Science Series. MIT Press, 1986. ISBN 9780262081597.

Berthold KP Horn and Brian G Schunck. Determining optical flow. *Artificial intelligence*, 17(1):185–203, 1981.

Seth Hutchinson, Gregory D. Hager, and Peter I. Corke. A tutorial on visual servo control. *Robotics and Automation, IEEE Transactions on*, 12(5):651–670, 1996.

Austin M Jensen, Marc Baumann, and Yang Quan Chen. Low-cost multispectral aerial imaging using autonomous runway-free small flying wing vehicles. In *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, volume 5, pages V–506. IEEE, 2008.

Hailin Jin, Paolo Favaro, and Roberto Cipolla. Visual tracking in the presence of motion blur. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 18–25. IEEE, 2005.

EN Johnson, PA DeBitetto, CA Trott, and MC Bosse. The 1996 mit/boston university/draper laboratory autonomous helicopter system. In *Digital Avionics Systems Conference, 1996., 15th AIAA/IEEE*, pages 381–386. IEEE, 1996.

Christopher Kemp. *Visual Control of a Miniature Quad-Rotor Helicopter*. PhD thesis, University of Cambridge, 2006.

D. Kragic and H. I. Christensen. Robust visual servoing. *The International Journal or Robotics Research*, 22(10-11):923–939, 2003.

V. Kyrki, D. Kragic, and H. I. Christensen. Measurement errors in visual servoing. *Robotics and Autonomous Systems*, 54(10):815 – 827, 2006.

Seok-Han Lee, Jae-Young Lee, and Jong-Soo Choi. Lens distortion correction using a checkerboard pattern. In *Proceedings of The 7th ACM SIGGRAPH International Conference on Virtual-Reality Continuum and Its Applications in Industry*, VRCAI '08, pages 44:1–44:2, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-335-8. doi: 10.1145/1477862.1477917. URL http://doi.acm.org/10.1145/1477862.1477917.

Vincent Lepetit and Pascal Fua. Keypoint recognition using randomized trees. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1465–1479, 2006.

Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An accurate O(n) solution to the PnP problem. *International Journal of Computer Vision*, 81(2): 155–166, 2009.

Stefan Leutenegger, Margarita Chli, and Roland Y. Siegwart. BRISK: Binary robust invariant scalable keypoints. *Computer Vision, IEEE International Conference on*, 0:2548–2555, 2011.

Wei Li, Tianguang Zhang, and K Klihnlenz. A vision-guided autonomous quadrotor in an air-ground multi-robot system. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2980–2985. IEEE, 2011.

Zhengrong Li, Yuee Liu, Ross Hayward, Jinglan Zhang, and Jinhai Cai. Knowledge-based power line detection for uav surveillance and inspection systems. In *Image and Vision Computing New Zealand, 2008. IVCNZ 2008. 23rd International Conference*, pages 1–6. IEEE, 2008.

Zhengrong Li, Yuee Liu, Rodney Walker, Ross Hayward, and Jinglan Zhang. Towards automatic power line detection for a uav surveillance system using pulse coupled neural filter and an improved hough transform. *Machine Vision and Applications*, 21(5):677–686, 2010.

Chia-Kai Liang, Li-Wen Chang, and Homer H Chen. Analysis and compensation of rolling shutter effect. *Image Processing, IEEE Transactions on*, 17(8):1323–1330, 2008.

Tony Lindeberg. Feature detection with automatic scale selection. *International journal of computer vision*, 30(2):79–116, 1998.

Alfredo Liverani, Francesco Leali, and Marcello Pellicciari. Real-time 3d features reconstruction through monocular vision. *International Journal on Interactive Design and Manufacturing (IJIDeM)*, 4(2):103–112, 2010.

H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004.

Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. In *IJCAI*, volume 81, pages 674–679, 1981.

Alexis Lussier Desbiens and MarkR. Cutkosky. Landing and perching on vertical surfaces with microspines for small unmanned air vehicles. *Journal of Intelligent and Robotic Systems*, 57(1-4):313–327, 2010. ISSN 0921-0296. doi: 10.1007/s10846-009-9377-z. URL http://dx.doi.org/10.1007/s10846-009-9377-z.

Elmar Mair, Gregory D. Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *Proceedings of the 11th European conference on Computer vision: Part II*, ECCV'10, pages 183–196, Berlin, Heidelberg, 2010. Springer-Verlag.

Eric Marchand and Franois Chaumette. Feature tracking for visual servoing purposes. *Robotics and Autonomous Systems*, 52(1):53 – 70, 2005.

Lorenz Meier, Petri Tanskanen, Friedrich Fraundorfer, and Marc Pollefeys. Pixhawk: A system for autonomous flight using onboard computer vision. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 2992–2997. IEEE, 2011.

Lorenz Meier, Petri Tanskanen, Lionel Heng, Gim Hee Lee, Friedrich Fraundorfer, and Marc Pollefeys Pollefeys. PIXHAWK: A micro aerial vehicle design for autonomous flight using onboard computer vision. *Autonomous Robots*, 33(1-2):21–39, 2012.

Nathan Michael, Daniel Mellinger, Quentin Lindsey, and Vijay Kumar. The grasp multiple micro-uav testbed. *Robotics & Automation Magazine, IEEE*, 17(3):56–65, 2010.

Krystian Mikolajczyk and Cordelia Schmid. Scale & affine invariant interest point detectors. *International journal of computer vision*, 60(1):63–86, 2004.

M. Milford and G. Wyeth. Persistent navigation and mapping using a biologically inspired slam system. *The Internoational Journal of Robotics research*, 29(9):1131–1153, 2010.

David M. Mount, Nathan S. Netanyahu, and Jacqueline Le Moigne. Efficient algorithms for robust feature matching. *Pattern recognition*, 32(1):17–38, 1999.

Matthias Mühlich and Rudolf Mester. The role of total least squares in motion analysis. In *Computer VisionECCV98*, pages 305–321. Springer, 1998.

Marius Muja and David G Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *VISAPP (1)*, pages 331–340, 2009.

Marius Muja and David G. Lowe. Fast matching of binary features. In *Computer and Robot Vision (CRV)*, pages 404–410, 2012.

RobinR. Murphy, Eric Steimle, Michael Hall, Michael Lindemuth, David Trejo, Stefan Hurlebaus, Zenon Medina-Cetina, and Daryl Slocum. Robot-assisted bridge inspection. *Journal of Intelligent & Robotic Systems*, 64(1):77–95, 2011. ISSN 0921-0296. doi: 10.1007/s10846-010-9514-8. URL http://dx.doi.org/10.1007/s10846-010-9514-8.

Keiji Nagatani, Satoshi Tachibana, M Sofne, and Yutaka Tanaka. Improvement of odometry for omnidirectional vehicle using optical flow information. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 1, pages 468–473. IEEE, 2000.

Andrew E Neff, DongBin Lee, Vilas K Chitrakaran, Darren M Dawson, and Timothy C Burg. Velocity control for a quad-rotor uav fly-by-camera interface. In *SoutheastCon, 2007. Proceedings. IEEE*, pages 273–278. IEEE, 2007.

Alexander Neubeck and Luc Van Gool. Efficient non-maximum suppression. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 850–855. IEEE, 2006.

U. Niethammer, M. R. James, S. Rothmund, J. Travelletti, and M. Joswig. Uav-based remote sensing of the super-sauze landslide: Evaluation and results. *Engineering Geology*, 128:2–11, 2012.

Hyondong Oh, Dae-Yeon Won, Sung-Sik Huh, DavidHyunchul Shim, Min-Jea Tahk, and Antonios Tsourdos. Indoor UAV control using multi-camera visual feedback. *Journal of Intelligent & Robotic Systems*, 61(1-4):57–84, 2011. ISSN 0921-0296. doi: 10.1007/s10846-010-9506-8. URL http://dx.doi.org/10.1007/s10846-010-9506-8.

Naoya Ohnishi and Atsushi Imiya. Featureless robot navigation using optical flow. *Connection Science*, 17(1-2):23–46, 2005.

Naoya Ohnishi and Atsushi Imiya. Dominant plane detection from optical flow for robot navigation. *Pattern Recognition Letters*, 27(9):1009–1021, 2006.

Naoya Ohnishi and Atsushi Imiya. Independent component analysis of optical flow for robot navigation. *Neurocomputing*, 71(10):2140–2163, 2008.

Raphael Ortiz. FREAK: Fast retina keypoint. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, CVPR '12, pages 510–517, Washington, DC, USA, 2012. IEEE Computer Society.

Mustafa Ozuysal, Michael Calonder, Vincent Lepetit, and Pascal Fua. Fast keypoint recognition using random ferns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(3):448–461, 2010.

Jungkeun Park, Sunggyu Im, Keun-Hwan Lee, and Jeong-Oog Lee. Vision-based SLAM system for small UAVs in GPS-denied environments. *Journal of Aerospace Engineering*, 25(4):519–529, 2011.

Jos M. Pea-Barragn, Francisca Lpez-Granados, Montserrat Jurado-Expsito, and Luis Garca-Torres. Sunflower yield related to multi-temporal aerial photography, land elevation and weed infestation. *Precision Agriculture*, 11(5):568 – 585, 2010.

P Pounds, R Mahony, P Hynes, and J Roberts. Design of a four-rotor aerial robot. In *Proc. 2002 Australasian Conference on Robotics and Automation*, volume 27, page 29, 2002.

Paul Pounds, Robert Mahony, and Peter 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.

Jacopo Primicerio, Salvatore Filippo Di Gennaro, Edoardo Fiorillo, Lorenzo Genesio, Emanuele Lugato, Alessandro Matese, and Francesco Primo Vaccari. A flexible unmanned aerial vehicle for precision agriculture. *Precision Agriculture*, 13(4):517–523, 2012. Springer.

Simon J. D. Prince. *Computer Vision: Models, Learning and Inference*. Cambridge University Press, 2012. ISBN 1107011795.

Kari Pulli, Anatoly Baksheev, Kirill Kornyakov, and Victor Eruhimov. Real-time computer vision with OpenCV. *Communications of the ACM*, 55(6):61–69, 2012.

Long Quan and Zhongdan Lan. Linear n-point camera pose determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):774–780, 1999.

Rene Racine. Altitude, elevation, and seeing. *Publications of the Astronomical Society of the Pacific*, 117(830):401–410, 2005. JSTOR.

Kambiz Rahbar and Hamid Reza Pourreza. Inside looking out camera pose estimation for virtual studio. *Graphical Models*, 70(4):57 – 75, 2008.

Albert Rango, Andrea Laliberte, Jeffrey E Herrick, Craig Winters, Kris Havstad, Caiti Steele, and Dawn Browning. Unmanned aerial vehicle-based remote sensing for rangeland assessment, monitoring, and management. *Journal of Applied Remote Sensing*, 3(1):033542–033542, 2009.

BryceB. Ready and ClarkN. Taylor. Inertially aided visual odometry for miniature air vehicles in gps-denied environments. *Journal of Intelligent and Robotic Systems*, 55(2-3):203–221, 2009. ISSN 0921-0296. doi: 10.1007/s10846-008-9294-6. URL http://dx.doi.org/10.1007/s10846-008-9294-6.

Pedro A Rodriguez, William J Geckle, Jeffrey D Barton, John Samsundar, Tia Gao, Myron Z Brown, and Sean R Martin. An emergency response uav surveillance system. In *AMIA Annual Symposium Proceedings*, volume 2006, page 1078. American Medical Informatics Association, 2006.

Paul L. Rosin. Measuring corner properties. *Computer Vision and Image Understanding*, 73(2):291 – 307, 1999. ISSN 1077-3142. doi: http://dx.doi.org/10.1006/cviu.1998.0719. URL http://www.sciencedirect.com/science/article/pii/S1077314298907196.

Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, volume 1, pages 430–443, May 2006.

Cyril Roussillon and Simon Lacroix. High rate-localization for high-speed all-terrain robots. In *Communications, Computing and Control Applications (CCCA), 2012 2nd International Conference on*, pages 1–8. IEEE, 2012.

Eric Royer, Maxime Lhuillier, Michel Dhome, and Jean-Marc Lavest. Monocular vision for mobile robot localization and autonomous navigation. *International Journal of Computer Vision*, 74(3):237–260, 2007.

Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to SIFT or SURF. In *International Conference on Computer Vision*, Barcelona, 11 2011.

S. Salazar, H. Romero, R. Lozano, and P. Castillo. Modeling and real-time stabilization of an aircraft having eight rotors. *Journal of Intelligent and Robotic Systems*, 54(1-3):455–470, 2009. ISSN 0921-0296. doi: 10.1007/s10846-008-9274-x. URL http://dx.doi.org/10.1007/s10846-008-9274-x.

Atheer L. Salih, M. Moghavvemi, Haider A. F. Mohamed, and Khalaf Sallom Gaeid. Flight PID controller design for a UAV quadrotor. *Scientific Research and Essays*, 5(23):3660–3667, 2010.

Davide Scaramuzza. Performance evaluation of 1-point-RANSAC visual odometry. *Journal of Field Robotics*, 28(5):792–811, 2011.

Jürgen Schellberg, Michael J Hill, Roland Gerhards, Matthias Rothmund, and Matthias Braun. Precision agriculture on grassland: Applications, perspectives and constraints. *European Journal of Agronomy*, 29(2):59–71, 2008.

Gerald Schweighofer and Axel Pinz. Robust pose estimation from a planar target. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(12):2024–2030, 2006.

Nicu Sebe, Qi Tian, Etienne Loupias, M Lew, and T Huang. Evaluation of salient point techniques. In *Image and Video Retrieval*, pages 367–377. Springer, 2002.

Keith Sevcik and Paul Oh. Testing unmanned aerial vehicle missions in a scaled environment. *Journal of Intelligent and Robotic Systems*, 54(1-3):297–305, 2009. ISSN 0921-0296. doi: 10.1007/s10846-008-9267-9. URL http://dx.doi.org/10.1007/s10846-008-9267-9.

Courtney S Sharp, Omid Shakernia, and S Shankar Sastry. A vision system for landing an unmanned aerial vehicle. In *Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on*, volume 2, pages 1720–1727. IEEE, 2001.

Jianbo Shi and Carlo Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

David Hyunchul Shim, Jae-Sup Han, and Hong-Tae Yeo. A development of unmanned helicopters for industrial applications. *Journal of Intelligent and Robotic Systems*, 54(1-3):407–421, 2009. ISSN 0921-0296. doi: 10.1007/s10846-008-9272-z. URL http://dx.doi.org/10.1007/s10846-008-9272-z.

R. Sim and G. Dudek. Learning environmental features for pose estimation. *Image and Vision Computing*, 19(11):733 – 739, 2001.

Devin Smith and Zachary Dodds. Visual navigation: image profiles for odometry and control. *Journal of Computing Sciences in Colleges*, 24(4):168–179, April 2009.

Stephen M Smith and J Michael Brady. SUSAN- a new approach to low level image processing. *International journal of computer vision*, 23(1):45–78, 1997.

M Srinivasan, SW Zhang, M Lehrer, and T Collett. Honeybee navigation en route to the goal: visual flight control and odometry. *Journal of Experimental Biology*, 199 (1):237–244, 1996.

B. Steder, G. Grisetti, C. Stachniss, and W. Burgard. Visual SLAM for flying vehicles. *Robotics, IEEE Transactions on*, 24(5):1088–1093, 2008.

John Stowers. *Biologically inspired visual contol of flying robots*. PhD thesis, University of Canterbury, 2013.

John Stowers, Michael Hayes, and Andrew Bainbridge-Smith. Quadrotor helicopters for visual flight control. In *Proceedings of Electronics New Zealand Conference*, pages 21–26, 2010.

John Stowers, Michael Hayes, and Andrew Bainbridge-Smith. Altitude control of a quadrotor helicopter using depth map from microsoft kinect sensor. In *Mechatronics (ICM), 2011 IEEE International Conference on*, pages 358–362. IEEE, 2011.

Hauke Strasdat, JMM Montiel, and Andrew Davison. Scale drift-aware large scale monocular SLAM. In *Robotics: Science and Systems*, volume 1, page 4, 2010.

Hauke Strasdat, J.M.M. Montiel, and Andrew J. Davison. Visual SLAM: Why filter? *Image and Vision Computing*, 30(2):65 – 77, 2012.

Richard Szeliski. *Computer vision: algorithms and applications*. Springer, 2011.

Kazuo Tanaka, Hiroshi Ohtake, Motoyasu Tanaka, and Hua O Wang. Wireless vision-based stabilization of indoor microhelicopter. *Mechatronics, IEEE/ASME Transactions on*, 17(3):519–524, 2012.

A. Tsalantsanis, K. Valavanis, and A. Yalcia. Vision based target tracking and collision avoidance for mobile robots. *Journal of Intelligent Robot Systems*, 48(2):285–304, 2007.

Joo Valente, David Sanz, Jaime Del Cerro, Antonio Barrientos, and Miguel de Frutos. Near-optimal coverage trajectories for image mosaicing using a mini quad-rotor over irregular-shaped fields. *Precision Agriculture*, 14(1):115 – 132, 2013.

A. Varol, A. Shaji, M. Salzmann, and P. Fua. Monocular 3d reconstruction of locally textured surfaces. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(6):1118–1130, 2012.

R Voigt, J Nikolic, C Hürzeler, S Weiss, L Kneip, and R Siegwart. Robust embedded egomotion estimation. In *Proc. of The IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, USA, September 2011.

Richard Voyles and JulieA. Adams. Editorial: Security, search and rescue robotics. *Journal of Intelligent & Robotic Systems*, 64(1):3–6, 2011. ISSN 0921-0296. doi: 10.1007/s10846-011-9613-1. URL http://dx.doi.org/10.1007/s10846-011-9613-1.

Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Monocular-slam–based navigation for autonomous micro helicopters in gps-denied environments. *Journal of Field Robotics*, 28(6):854–874, 2011.

Brian Williams, Nicolas Hudson, Brent Tweddle, Roland Brockers, and Larry Matthies. Feature and pose constrained visual aided inertial navigation for computationally constrained aerial vehicles. In *Robotics and automation (ICRA), 2011 IEEE international conference on*, pages 431–438. IEEE, 2011.

Stephen Williams and Ayanna M. Howard. Developing monocular visual pose estimation for arctic environments. *Journal of Field Robotics*, 27(2):145–157, 2010.

Song Wu and Michael Lew. Evaluation of salient point methods. In *Proceedings of the 21st ACM International Conference on Multimedia*, MM '13, pages 685–688, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2404-5. doi: 10.1145/2502081.2502179. URL http://doi.acm.org/10.1145/2502081.2502179.

Shaowu Yang, Sebastian A. Scherer, and Andreas Zell. An onboard monocular vision system for autonomous takeoff, hovering and landing of a micro aerial vehicle. *Journal of Intelligent and Robotic Systems*, 69(1-4):499–515, 2013.

Pablo J Zarco-Tejada, José AJ Berni, L Suárez, and E Fereres. A new era in remote sensing of crops with unmanned robots. *SPIE Newsroom*, pages 2–4, 2008.

Shannon N Zenk, Amy J Schulz, Stephen A Matthews, Angela Odoms-Young, JoEllen Wilbur, Lani Wegrzyn, Kevin Gibbs, Carol Braunschweig, and Carmen Stokes. Activity space environment and dietary and physical activity behaviors: a pilot study. *Health & place*, 17(5):1150–1161, 2011.

Chunhua Zhang and John Kovacs. The application of small unmanned aerial systems for precision agriculture: a review. *Precision Agriculture*, 13(6):693 – 712, 2012.

Tianguang Zhang, Ye Kang, Markus Achtelik, Kolja Kuhnlenz, and Martin Buss. Autonomous hovering of a vision/imu guided quadrotor. In *Mechatronics and Automation, 2009. ICMA 2009. International Conference on*, pages 2870–2875. IEEE, 2009.

Zhengyou Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(11):1330–1334, 2000.

Barbara Zitov and Jan Flusser. Image registration methods: a survey. *Image and Vision Computing*, 21(11):977 – 1000, 2003.

INTERNET RESOURCES

libdc1394: The API for IEEE1394/firewire cameras. http://damien.douxchamps.net/ieee1394/libdc1394/, Jun 2013.

Xbox kinect — full body gaming and voice control. http://www.xbox.com/en-US/kinect, Feb 2014.

Arch Linux. Arch linux for ARM computers. www.archlinuxarm.org, Sep 2012.

Arch Linux. Archlinux. www.archlinux.org, Oct 2013.

ArduPilot. ArduPilot uav autopilot. www.ardupilot.com, Mar 2012.

ARToolkit. ARToolkit Home Page. http://www.hitl.washington.edu/artoolkit, Dec 2013.

Ascending Technologies. Ascending technologies uavs. www.asctec.de, Nov 2012.

BoofCV. BoofCV computer vision library. www.boofcv.org, Jul 2013.

Gumstix. Gumstix single-board computers. www.gumstix.com, Jun 2012.

Hardkernel. Hardkernel android development hardware. www.hardkernel.org, Aug 2013.

Mikrokopter. Mikrokopter multi-rotor uav. www.mikrokopter.de, Dec 2013.

MIT. LCM - Lightweight Communication and Marshalling. https://code.google.com/p/lcm/, Aug 2013.

NodeCopter. Nodecopter control library for the parrot AR.Drone. www.nodecopter.com, Dec 2013.

Node.JS. Node.js framework. www.nodejs.org, Dec 2013.

NuttX. NuttX Real-Time Operating System. http://nuttx.org/, Nov 2013.

OpenCV. OpenCV computer vision library. www.opencv.org, Mar 2013a.

OpenCV. OpenCV computer vision library. www.opencv.org/doc/tutorials/calib3d/camera_calibration/camera_calibration.html, Mar 2013b.

OpenCV. OpenCV computer vision library. www.opencv.org/modules/imgproc/doc/geometrictransformations.html#undistort, Mar 2013c.

Oxford Datasets. Oxford wall dataset. http://www.robots.ox.ac.uk/~vgg/data/data-aff.html, Jun 2013.

Paparazzi. Papparazziuav. www.paparazzi.enac.fr, Feb 2013.

Parrot. Parrot AR.Drone. ardrone2.parrot.com, Dec 2013a.

Parrot. Parrot AR.Drone software development kit. ardrone2.parrot.com/developer-zone, Dec 2013b.

PASCAL. PASCAL dataset. http://pascallin.ecs.soton.ac.uk/challenges/VOC/databases.html#VOC2006, Nov 2013.

Pixhawk. Pixhawk cheetah uav. https://pixhawk.ethz.ch/micro_air_vehicle/quadrotor/cheetah, Nov 2012a.

Pixhawk. PXComExpress onboard computer. www.pixhawk.ethz.ch/electronics/base_large#pxcomexcom_express, Nov 2012b.

Pixhawk. Multirotor PID Tuning Guide - PX4 Autopilot Platform. http://pixhawk.org/users/multirotor_pid_tuning?s[]=tuning, Apr 2013a.

Pixhawk. MAVCONN communications framework. https://pixhawk.ethz.ch/software/mavconn/start, Aug 2013b.

Pixhawk. MAVLink Micro Aerial Communication Protocol - QGroundcontrol GCS. http://qgroundcontrol.org/mavlink/start, Jun 2013c.

Pixhawk. The Pixhawk project. wwww.pixhawk.ethz.ch, Nov 2013.

Pixhawk. PX4FMU autopilot. pixhawk.ethz.ch/px4/modules/px4fmu, Sept 2013a.

Pixhawk. Calib. http://pixhawk.org/users/sensor_calibration?s[]=sensor&s[]=calibration, Apr 2013b.

Pixhawk. PXIO4AR AR.Drone carrier board for the PX4FMU. www.pixhawk.ethz.ch/px4/modules/pxio4ar, Aug 2013c.

Pixhawk. QGroundControl groundstation software. www.qgroundcontrol.org, Aug 2013d.

Pixhawk. Thread Priorities - PX4 Autopilot Platform. http://pixhawk.org/dev/thread_priorities, Jan 2014.

Point Grey. Point Grey Chameleon®. http://ww2.ptgrey.com/USB2/chameleon, Apr 2013.

QuadShot. Quadshot. http://transition-robotics.com/collections/quadshots, Sep 2013.

Qualcomm. FastCV computer vision library. https://developer.qualcomm.com/mobile-development/mobile-technologies/computer-vision-fastcv, May 2013.

RCTimer. RCTimer products. www.rctimer.com, Jul 2013.

ROS. ROS.org—Powering The Worlds Robots. http://www.ros.org/, Dec 2013.

SimonK. SimonK brushless controller firmware patch. github.com/sim-/tgy, Jun 2013.

SimpleCV. SimpleCV computer vision library. www.simplecv.org, Jan 2013.

Sony. Playstation® Eye Camera — PS3 Accessories. http://us.playstation.com/ps3/accessories/playstation-eye-camera-ps3.html, Jun 2013.

Uncanny Vision. Uncanny vision. www.uncannyvision.com, May 2013.

Univ of Penn. Homepage — GRASP Laboratory. https://www.grasp.upenn.edu/, Dec 2013.