Quadrotor Position Estimation using Low Quality Images

by

Ryan Gariepy

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Mechanical Engineering

Waterloo, Ontario, Canada, 2011

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

The use of unmanned systems is becoming widespread in commercial and military sectors. The ability of these systems to take on dull, dirty, and dangerous tasks which were formerly done by humans is encouraging their rapid adoption. In particular, a subset of these undesirable tasks are uniquely suited for small unmanned aerial vehicles such as quadrotor helicopters. Examples of such tasks include surveillance, mapping, and search and rescue.

Many of these potential tasks require quadrotors to be deployed in environments where a degree of position estimation is required and traditional GPS–based positioning technologies are not applicable. Likewise, since unmanned systems in these environments are often intended to serve the purpose of scouts or first–responders, no maps or reference beacons will be available. Additionally, there is no guarantee of clear features within the environment which an onboard sensor suite (typically made up of a monocular camera and inertial sensors) will be able to track to maintain an estimate of vehicle position. Up to 90% of the features detected in the environment may produce motion estimates which are inconsistent with the true vehicle motion. Thus, new methods are needed to compensate for these environmental deficiencies and measurement inconsistencies.

In this work, a RANSAC–based outlier rejection technique is combined with an Extended Kalman Filter (EKF) to generate estimates of vehicle position in a 2–D plane. A low complexity feature selection technique is used in place of more modern techniques in order to further reduce processor load. The overall algorithm was faster than the traditional approach by a factor of 4. Outlier rejection allows the abundance of low quality, poorly tracked image features to be filtered appropriately, while the EKF allows a motion model of the quadrotor to be incorporated into the position estimate.

The algorithm is tested in real–time on a quadrotor vehicle in an indoor environment with no clear features and found to be able to successfully estimate position of the vehicle to within 40 cm, superior to those produced when no outlier rejection technique was used. It is also found that the choice of simple feature selection approaches is valid, as complex feature selection approaches which may take over 10 times as long to run still result in outliers being present.

When the algorithm is used for vehicle control, periodic synchronization to ground truth data was required due to nearly 1 second of latency present in the closed–loop system. However, the system as a whole is a valid proof of concept for the use of low quality images for quadrotor position control. The overall results from the work suggest that it is possible for unmanned systems to use visual data to estimate state even in operational environments which are poorly suited for visual estimation techniques. The filter algorithm described in this work can be seen as a useful tool for expanding the operational capabilities of small aerial vehicles.

Acknowledgements

I would first like to thank my supervisor, Prof. Steven Waslander, for his support, patience, and guidance every step of the way. Also, I would like to thank Mike and April at Aeryon Labs for the great deal of assistance they have provided throughout my work.

To my colleagues at the office: Pat, Bryan, Matt, Mike, Mark, Sean, and Jaron. Thanks for being understanding when I was off flying or when I needed to get a paper done.

I would also like to thank the others in the lab. Yassir, Mike, John, P.J., Yan, Peiyi, Carlos, Arun, Sid for the huge number of suggestions and insight you have all provided, as well as your tolerance of my surprise test flights. Special thanks goes to those of you who have stood at the other end of the tether of an angry quadrotor for me.

I would like to thank everyone who's kept me sane with much needed dancing breaks, even if only for a few hours here and there. If you've taught me, followed me, led me, or helped me in any way, thank you.

Finally, to my family. I wouldn't have made it this far without your faith in me.

Dedication

This is dedicated to everyone who has worked with me, learned with me, and supported me over the years, as well as to the many people around the world who share my goal of making the world a better place. Finally, to Jeff, for your belief in the potential of all of your friends.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

1.1.1 Application

The small size, high maneuverability, and disposable nature of unmanned aerial vehicles (UAVs) makes them uniquely suited for a variety of applications. They have demonstrated the ability to lift objects [1], maneuver through tight spaces [2], and serve as remote surveillance platforms [3]. Other proposed uses of UAVs include inspection of buildings and bridges, transmission tower surveying, autonomous tracking of moving ground and sea targets, and serving as wireless network infrastructure. These latter proposed uses have not yet been rigourously demonstrated in the field.

This work focuses on "quadrotor" UAVs such as those shown in Figure 1.1 . Unlike small fixed–wing UAVs such as the commercially available Cropcam [4], quadrotors are capable of hovering in place, even in situations with unexpected wind gusts. And, unlike small unmanned helicopters like the Camcopter [5], they have no need for complex mechanical systems (i.e. the swashplate assembly), requiring only four propellers directly coupled to motors [6]. Due the small size of quadrotors and their ability to hover in place, there are many potential indoor and outdoor environments for which they are ideal. For instance, quadrotors are good platforms to employ for urban search and rescue, nuclear reactor inspection, and indoor mapping and surveying. The combination of platform capabilities and mechanical simplicity makes quadrotors uniquely suited to a wide range of inspection tasks. Of particular relevance to this work is the growing number of commercially available quadrotors which are meant for deployment by government and private industry [7, 8, 9]. Uses of these commercial quadrotors range from property assessment [7] to crime scene mapping [8].

However, effective control of quadrotors requires a reliable method of estimating vehicle position. Without such an estimation method, the low mass and undamped nature of quadrotor platforms means that they will be susceptible to position drift caused by external disturbances or slight variations in vehicle orientation. The presence of position drift means that even static obstacles can pose a risk to the vehicle. Though the vehicle is easily capable of correcting for this drift, a suitable strategy to do so cannot be developed without knowledge of the nature of the drift. Once this knowledge is available, a controller can be developed to accurately control position using PID control [6], potential fields [10], or other common control techniques. Currently, sensors such as inertial measurement units (IMUs) are able to measure key state variables such as acceleration and vehicle rotation rate directly. Unfortunately, position control requires knowledge of the full state; including position and translational velocity. The Global Positioning System (GPS) is capable of providing position measurements in open outdoor environments, but its performance degrades when the vehicle is near or under large obstacles, or is operating indoors.

Figure 1.1: Aeryon Scout [8]

If position measurements are unavailable, small UAVs lose the ability to hold position and reject disturbance forces, rendering quadrotors unusable in cluttered environments due to the potential of collision. At present, all available commercial solutions are either fully remote–controlled (requiring a human to be within line–of–sight of the vehicle) or rely on GPS receivers. Furthermore, the bulk of the research which has been done in this area either uses preinstalled infrastructure that provides GPS–like sensor information, relies on the presence of unique or known features in the environment, or uses a known map of the surroundings.

Using a computer vision approach for position estimation will greatly enhance the safety and usability of the vehicle, by enabling operation in GPS-denied environments or supplementing GPS position and velocity information with additional measurements.

1.1.2 State Estimation

Aerial vehicles typically have their state defined by the following 12–DoF vector:

$$X = \begin{bmatrix} x & y & z & \dot{x} & \dot{y} & \dot{z} & \phi & \theta & \psi & \dot{\phi} & \dot{\theta} & \dot{\psi} \end{bmatrix}^T \tag{1.1}$$

where x, y, z and their derivatives represent vehicle position and translational velocity, and ϕ, θ, ψ and their respective derivatives represent vehicle orientation and rotational rates. It should be noted that position and velocity can be expressed in a variety of ways, with the most popular being latitude–longitude–height (LLH), also known as 'geodetic' coordinates, North–East–Down (NED), and Earth–Centred, Earth–Fixed (ECEF) [11]. Likewise, vehicle orientation is typically represented as a set of Euler angles, usually of the 3-2-1 type [12].

The state vector X can be augmented by other information, such as temperature, gyro bias, motor speeds, and wind states, if these effects can be estimated given the sensor configuration, and are expected to contribute significantly to the modeling of the platform [13]. In this work, most of these additional states are abstracted away by the onboard systems of the testbed. For instance, temperature and gyro bias are incorporated into the onboard vehicle orientation estimation routine, while a responsive orientation controller subsumes the motor speeds. Wind states are not incorporated in this work, since indoor operation results in winds which vary heavily both spatially and temporally and thus cannot be easily modeled.

A typical approach used in position estimation of a quadrotor operating outdoors is to use an Extended Kalman Filter [14] to fuse noisy direct measurements of the vehicle's position (provided by GPS) with a motion model taking IMU data as control inputs. This has the benefit of always using globally–referenced position measurements. The use of globally–referenced measurements removes the possibility of the estimation error integrating over time. In comparison, using vision as a replacement for GPS requires that the features used as position references in each image frame are likewise globally–referenced. For each feature to be referenced in such a manner, the camera must be operating in a structured environment where the location of each feature is known, and each feature can be uniquely identified.

In general, this is not the case. The alternative is for visual estimators to integrate relative displacements between images over time to form an estimate of global position. Even in this situation, there is the challenge of determining correspondence between features in two separate images. Though it is not as difficult as identifying each point in the image uniquely over all time, the "correspondence problem" poses several challenges. Numerous attempts have been made to address the correspondence problem [15, 16, 17, 18]. The main focus of this work is to identify instances where incorrect correspondence of features has occurred, and to eliminate these instances as invalid measurements.

Position and orientation estimation of the vehicle can also be coupled with the global estimation of the state of the surrounding environment; a challenge referred to as the Simultaneous Localization and Mapping (SLAM) problem [19]. SLAM is a significant research area on its own, with many different ways of representing the relationship between the vehicle's state and that of the environment it is operating within. Vision-based SLAM also requires the correspondence problem to be solved and can therefore be seen as an expanded version of the position estimation problem addressed in this work. The same difficulties in feature correspondence arise for visual SLAM in environments with low quality features, and the position estimation algorithm presented in Chapter 3 can easily be adapted to the full SLAM problem as well.

1.1.3 Objective

The primary objective of this research is to develop a full state estimation technique which is applicable in GPS–denied environments which have not been preconditioned for robot operation. It will be assumed that the only sensors available for this investigation are those which are included on the Aeryon Scout quadrotor micro-UAV [8]. Specifically, this work will focus on using the onboard camera in combination with visual state estimation techniques. The information available from the rest of the vehicle sensor suite will augment the camera data. The vehicle sensor suite includes an IMU for inertial measurements and a sonar sensor for altitude measurements. Using the existing vehicle sensor suite frees up the remaining payload for other application specific uses. In addition, limiting the available equipment to that already available in the vehicle payload is desirable from an industrial perspective, as it reduces the work needed for the commercialization and technology transfer of the research.

One of the key themes throughout this work is the "featureless" or low–contrast nature of the environment. Unlike much of the prior work, the environment is not assumed to have features which are easily tracked or uniquely identified by common computer vision approaches.

Due to the many challenges that result from requiring robust operation in featureless environments, a series of assumptions have been made.

Assumption 1.1. *The ground is reasonably flat and level, with only minor deviations in the ground's surface.*

Though this condition may not be valid for every operational environment, it is a reasonable assumption for indoor environments and benign outdoor environments. This assumption removes an additional potential source of estimation error.

Assumption 1.2. *The ground plane being observed is assumed to be static with respect to a global reference frame.*

Since quadrotors are not typically deployed near people for safety and regulatory reasons, it is reasonable to assume that there will be no one moving in the ground plane. Additionally, unless it consists of grass or water, the ground itself will not contain motion.

Assumption 1.3. *The camera is kept level with the ground at all times.*

This assumption simplifies the resulting mathematics for determining feature position in 3D, but can be easily generalized to a camera with known orientation. Since Assumption 1.1 establishes that the ground itself is reasonably level, providing a camera with the ability to compensate for roll and pitch of its mount with respect to gravity will result in the camera remaining level with respect to the ground. There are many gimbal mounts available in the market which are capable of performing roll and pitch compensation with negligible lag.

Assumption 1.4. *The height of the camera from the ground is assumed to be fully known.*

There are common methods of sensing height from a flat surface (which the vehicle is operating over, as per Assumption 1.1), including SONAR and laser rangefinders. As well, if such a surface is out of range of active sensors, passive pressure sensors are commonly used in the field to establish a vertical displacement away from a reference height.

Assumption 1.5. *All of the pixels in the image correspond to points on the ground.*

This assumption is valid due to the low heights at which the vehicle is currently flown, and allows features found within each image plane to be mapped directly to locations in the corresponding 3–D global frame. When this assumption is relaxed, the algorithm will also need to incorporate SLAM techniques and operate using 3–D features, or will need to segment the image to determine which points within it are part of the ground.

Assumption 1.6. *High performance computer hardware is not available, and the system is meant to run in real time.*

This limitation on available hardware mirrors the computational resources found on the current and next–generation vehicle platforms. Such restrictions are common to aerial vehicles due to power supply and payload limits. For example, quadrotor UAVs typically have under 25 minutes of flight time with no payload and no additional power draw. The addition of multicore or otherwise higher–performance computer hardware will be evaluated in future work after bottlenecks in the existing system have been identified.

1.2 Related Work

There are many available sensors which can be used for vehicle state estimation. Inertial measurement units are typically able to provide estimates of vehicle acceleration and rotation with acceptable levels of accuracy. Integrating acceleration information to estimate position results in poor performance over time. Currently, the first and foremost method of position estimation in field robotics is the now–ubiquitious GPS system. With real–time kinematic (RTK) corrections, reliable absolute positioning can reach centimetre–level accuracy [20]. However, this kind of accuracy not only requires clear view of the sky, but an offboard fixed reference to correct for the ± 15 m expected variations in positioning information due to atmospheric effects [11]. Even using technology such as space-based augmentation services (SBAS) will not attain RTK–level precision. Expected accuracy in this case is ± 1.5 m. Any estimation technique which is designed to supplant GPS should at a minimum be able to attain an SBAS level of accuracy, as this technology is the current standard for field–deployed quadrotors.

Active sensing technology has also taken leaps forward. Light detection and ranging (LIDAR) sensors have been seeing more exposure and their ability to map an environment reliably at high accuracies and frequencies is beneficial [21]. Despite these advances, LIDAR hardware still requires high mass and relatively high power drain, which has limited its use to within the research community. In a similar way, the Microsoft Kinect sensor has also seen broad adoption among researchers and hobbyists [22]. However, it has the limitation that it only works inside, as the IR structured light grid it uses to resolve depth in images is overcome by infrared interference from the sun even on cloudy days.

With the drastic increase of UAV payload capacity and processor power, onboard vision processing is becoming an option for quadrotor state estimation. The standard for indoor control of such vehicles is to use beacons or features with known appearance in known locations in conjunction with a camera system. If such features are set up as targets which the vehicle's camera can observe, they must be placed in the environment ahead of time, and their locations and characteristics communicated to the vehicle [23, 24]. This is known as position–based visual servoing (PBVS) and can commonly be found on industrial manipulators. Unfortunately, the vehicle is limited to operating in regions where it can observe the predefined targets.

Another option is the placement of trackable markers on the vehicle which can be observed by one or more external cameras [6, 25, 26]. High–quality external local positioning systems such as OptiTrack [27] and VICON [25] are an excellent embodiment of this. The frequency and accuracy of the state estimates they provide allow impressive control techniques to be demonstrated [2, 28]. Overall, local positioning systems are popular hardware for performing controls research, as the required image processing hardware can be located offboard. It too requires modification to the environment as cameras need to be installed

ahead of time. Like the PBVS approach, the vehicle's operation is limited to areas which are within view of the cameras. Overall, both local positioning systems and PBVS approaches require modification to the surroundings, something which is unacceptable if the UAV is to be deployed in an uncontrolled and unknown environment.

The majority of these prior methods use simple techniques for tracking distinct features. For example, OptiTrack and VICON systems use IR–reflective features, IR lights, and appropriate filters to eliminate the majority of non–markers from the images, resulting in de–facto feature identification. Others use optical flow techniques such as that proposed by Lucas and Kanade [29] to track features between frames. Feature tracking is a significant improvement from a processing time standpoint on earlier "block–matching" techniques which would attempt to determine flow on a pixel–by–pixel basis. When these techniques are used, corner–like features are typically used to allow the direction of their motion to be clearly resolved [15]. It has also been proven that corner–like features are well–suited to the Lucas and Kanade optical flow algorithm [16].

Some approaches have been proposed which do not require modifications to the environment, usually because they are designed to detect unknown features in the environment. One general strategy in this area is known as "structure from motion" (SfM) [30]. Here, the environment is assumed to be static and the vehicle motion is assumed to be approximately known between image frames, providing an epipolar constraint [31]. The strategy combines these inter–frame estimates with features matched between frames to map the environment in 3–D. For example, Call uses a structure from motion approach to attempt to estimate the 3-D position of image features with respect to the vehicle in real time [32]. Merrell used a similar approach to detect and avoid obstacles [33]. If a static environment is assumed, this last method could be modified to estimate motion instead. However, SfM requires significant onboard processing power.

In general, structure from motion approaches requires widely separated views to produce the required image disparity [34], and such disparity also requires robust feature selection processes. That is, they assume that the correspondence issue first introduced in Section 1.1.2 is a solved problem. Additionally, the views SfM approaches require result in poor performance from tracking corner–like features, since these features are not robust to rotations or changes in relative distance. Though feature selectors such as "speeded up robust features" (SURF) [17] and "scale invariant feature transforms" (SIFT) [18] exist which can produce features which can be matched well between images, they require significant processing power for the generation of such features. As well, much of the work which uses complex feature selectors simply assumes that the environment is cluttered enough to produce these types of features consistently. This work does not make this assumption. A study of the suitability of various feature selectors for tracking features in uncluttered environments can be found in Section 5.1.

Image-based visual servoing (IBVS) techniques are also used for visual state estimation. In an IBVS approach, there is no clear transformation at any point between a point on the image and physical location(s) in the world. As they cannot relate image features to points in the global frame, they cannot be used for mapping applications. This family of algorithms is primarily used for control purposes, where error signals are defined based only on information within the image. Controllers have been designed to regulate features such as the position of first moments of a target within the image [23], the magnitude of the optical flow field [35], or the position of a set of linear features within the image [36]. Because of the lack of a common reference frame, a pure IBVS technique cannot improve estimates by incorporating inertial sensing or external correction data. Inertial sensing data in particular is readily available on small UAVs from additional onboard sensors. Due to this, IBVS controllers cannot incorporate real–world information and restrictions such as maximum velocities, maximum accelerations, or limitations on a vehicle's position.

Finally, there are approaches which integrate other sensors with the camera data to produce state estimates in the global frame. Kendoul's approach uses three nested Kalman filters to estimate UAV motion with a camera and inertial sensors, allowing it to use images to correct position estimates based on inertial data. However, the proposed nested Kalman filter approach assumes small interframe motion and well–tracked features [37]. The assumption of well–tracked features is particularly sensitive, as there are many environments which will not produce such features.

A recent commercial development has successfully combined inertial data with an IBVS approach to achieve reliable position control. The Parrot AR-Drone (Figure 1.2) uses a forward–facing, a downward–facing camera, and an estimate of inter–frame yaw to estimate vehicle translation and rotation in an X–Y plane. [38]. It comes closest to a solution that addresses the main concerns identified with existing methods. The AR-Drone can handle unstructured environments, requires no external infrastructure, uses low–cost and low–mass cameras, does not require GPS technology, and does not seem to require high–quality images. However, it makes several key assumptions which are acceptable to Parrot due to its market positioning as a 'toy,' but which may not be acceptable for the inspection tasks described in Section 1.1.1. Namely:

- The image viewed in the forward plane has a suitable quantity of distinct features to track.
- When sections of the image contain more features than others, it is acceptable to override operator inputs and direct the cameras towards such sections to increase the number of features in the image.
- Objects in view of the camera are in a common plane at a significant distance from the vehicle.

Figure 1.2: Parrot AR–Drone [39]

In general, a downside of tracking features with unknown characteristics is the degree to which outliers or otherwise invalid data can corrupt the measurements. Attempting to visually estimate state in noisy conditions is quite challenging as demonstrated by previous experiments [40], since the nature of the visual estimation problem uses features matched across multiple images as measurements. Additionally, a key assumption of the commonly used Kalman filter is that the noise present in its inputs is Gaussian [14]. Even if simpler approaches such as basic averaging are used, the presence of large outliers in the measurements can add significant error to the results.

Due to this negative effect of outliers on estimation algorithms, a recent area of research centres on the filtering of visual measurement data for outliers before providing the data to a state estimator. This filtering can be used to mitigate the effect of errors in feature correspondence. A well known method for removing outliers from datasets is the Random Sample Consensus (RANSAC) algorithm originally developed by Fischler and Bolles [41]. By applying RANSAC, it is possible to remove outliers from visual measurement data before the data is incorporated into a state estimator.

Vedaldi laid the groundwork for the integration of RANSAC into traditional state estimation techniques in 2005 by merging RANSAC–based outlier rejection with an Extended Kalman filter [42], and it has been applied to a variety of practical applications since. The approach was developed to work in real–time and it assumes that features remain tracked throughout the image sequence. This assumption implicitly requires that high quality features exist, since low quality features are not consistently detectable. Kitt used a similar approach with both SIFT features and Harris corners for determining pose offsets between stereo images [43] except without fully integrating the Kalman update equations into the RANSAC iterations. Kitt noted specifically that Harris corners were beneficial from an efficiency standpoint. Scaramuzza applied the same general concept of merging RANSAC and a Kalman filter to localizing a full–sized Smart Car [44] with monocular camera data and appropriate motion constraints.

Most recently, Civera et al. explored the applicability of Vedaldi's work to reducing the dimensions required to perform visual 6–DoF pose estimation [34], where they used a similar method with additional constraints to estimate the state of a differential-drive ground vehicle. Feature correspondence over all time was found not to be required, as long as each frame shared a number of features in common with previous and following frames to avoid integrating estimation error over time. As well, their work was only a partial implementation, as they avoided the correspondence problem by manually matching points between images.

In several of cases [34, 45], an "Active Matching" strategy is used to prefilter the visual data by removing data points which are obvious outliers before they are considered by RANSAC as potential data. This strategy reduces the likelihood of wasting processor cycles on incorrect RANSAC models [46]. Active matching is a simple refinement to existing work, but should be noted due to its potential to easily improve the quality of estimates without requiring too much more computational effort.

In general, the area of visual state estimation is well studied, although many of the existing algorithms perform poorly in the face of improperly corresponded features. The existing methods are therefore not robust to operation in environments with low quality images. Indoor applications frequently result in the need to operate reliably with low quality images. A clear need exists for a visual state estimation approach which can use such images while being resilient to errors in feature correspondence.

1.3 Research Approach and Contribution

The goal of the research is to develop a method for estimating the position of a quadrotor UAV operating indoors using only its onboard sensing hardware, using the assumptions which are detailed in Section 1.1.3. The work focuses on the need to find reliable feature correspondences even when the images being used are of poor quality. Specifically, it addresses the significant amount of outlier data which is present in low–quality images and works towards mitigating the negative effect of outliers on position estimates. The algorithm which was developed combines RANSAC outlier rejection techniques and an Extended Kalman Filter (EKF) in the filtering of poor quality images, resulting in an estimator which is both faster and more robust than one which relies on a Extended Kalman Filter alone.

Overall, this work demonstrates the feasibility of a new algorithm which can use very poor quality, nearly 'featureless' images to assist in vehicle state estimation and control. This is an improvement from existing visual estimation techniques which assume that input images contain a multitude of easily trackable, high–quality features. This research is an experimental work which is focused on the challenges posed by real–world hardware. The

results of this work opens doors to the use of quadrotor UAVs in GPS–denied environments which have not been preconditioned by the deployment of markers. Additionally, the tests also demonstrate the feasibility of using camera–based navigation and mapping techniques in environments containing poorly trackable features.

The main results which will be presented in this thesis are:

- A position estimation algorithm which combines RANSAC and an EKF to allow for the use of low–quality images as the main source of measurements.
- Experimental validation that modern, computationally expensive feature selection approaches provide little gain over simpler and faster approaches when applied in low–quality environments.
- Verification that the characteristics of typical image noise requires outlier rejection techniques in addition to traditional averaging and Kalman filtering.
- Test results showing that reasonable position estimation is possible even when over 50% of image features are outliers.
- Validation that using outlier rejection in conjunction with a Kalman filter requires less overall computation than solely applying a Kalman filter.
- Flight test results from offline and online implementations of the position estimation algorithm.
- Flight test results from using the position estimation algorithm as the input into the control system of a quadrotor UAV.

Further background theory will be provided in Chapter 2, and the final algorithm produced by this work will be detailed in Chapter 3. The experimental test setup is described in Chapter 4. Results from a variety of experimental tests and configurations can be found in Chapter 5. Finally, Chapter 6 contains concluding remarks and recommendations.

Chapter 2

Background and Theory

2.1 Coordinate Frames

In the problem being examined, there are four key coordinate frames present. The first frame to be considered is a global frame, O_G, fixed to the environment around the vehicle. In most cases, the Earth–fixed reference frame is used here, with the x-axis pointing due North and the y-axis pointing due East.

The second frame used is a body–fixed frame, O_B, which is tied to the quadrotor. The x-axis in this case points "forward" and the y-axis points to the "right" of the vehicle. These two frames are related as shown in Figure 2.1, with (x_G, y_G, z_G) defining the global frames' axes and the Euler angles (ϕ, θ, ψ) defining the corresponding rotation.

Next, a camera frame, O_C, associated with a camera is mounted in a downward facing manner on a two–axis gimbal at the center of the quadrotor. The gimbal's axes correspond to the roll and pitch axes of the vehicle, allowing the onboard controller to adjust the gimbal to compensate mechanically for vehicle roll and pitch, reducing image skew and disturbance. Assuming that the roll and pitch compensation is ideal, the relationship between the body–fixed frame and the camera is shown in Figure 2.2(a), and likewise for O_G and O_C in Figure 2.2(b). The global frames' axes are denoted as in Figure 2.1, while body and camera frames are denoted by (x_B, y_B, z_B) and (x_C, y_C, z_C), respectively. The angle β in Figure 2.2 refers to a fixed angle that may exist between the camera frame and the vehicle frame due to an angular offset about z_B.

The final frame required is the image frame, which is a 2–D plane assumed to be coplanar and with a fixed offset and scale from the camera frame. The x and y axes in both the image and camera frames remain parallel with the image plane being translated by (c_x, c_y) such that the origin of the camera frame is at the centre of the image (Figure 2.3).

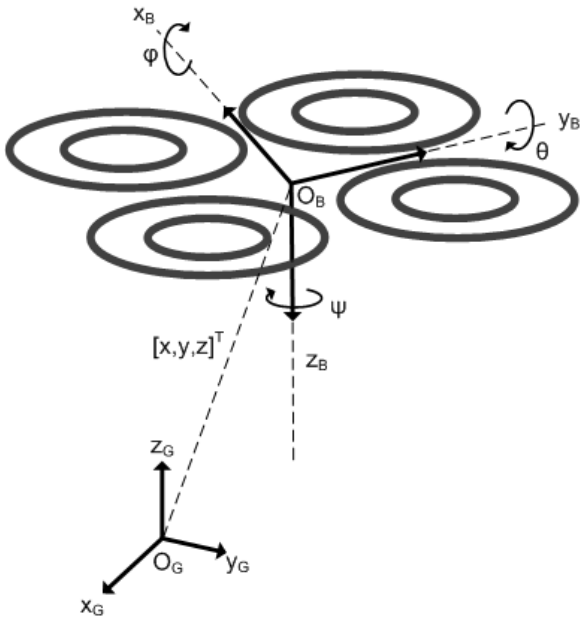


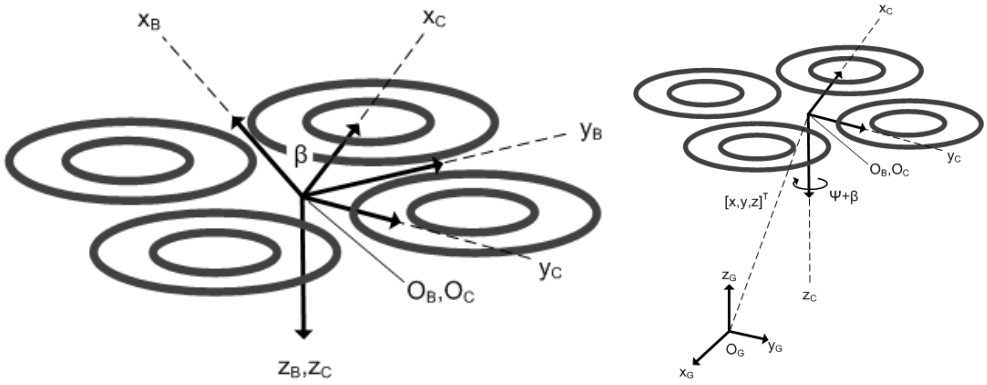
Figure 2.1: Relating the quadrotor body–fixed frame to the global frame

A pinhole camera model is used to relate points anywhere in the camera frame to points lying on the xy ('image') plane of the image frame [47]. Specifically, given a point $P_I \in \mathbb{R}^2$ in the image plane I, there are multiple possible corresponding points in O_C. Equation (2.1) expresses this ambiguity. It incorporates the above mentioned (c_x, c_y) offsets with image scaling represented by focal lengths (f_x, f_y).

$$\begin{bmatrix} P_I^u \\ P_I^v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_C^x / P_C^z \\ P_C^y / P_C^z \\ 1 \end{bmatrix} \tag{2.1}$$

The surjection between points in the camera frame and points on the image plane can be resolved by providing additional information. Figure 2.4(a) depicts the overall relationship between a point in O_G and the same point in O_C, showing this ambiguity. We will assume that the height h of the vehicle is given by the onboard height estimator and all of the features lie on the xy plane ($z = 0$). Due to these assumptions, the scale ambiguity can be resolved and a unique relationship between P_I and P_C found (Figure 2.4(b)).

Given the above, the relationship between a point $P_G \in \mathbb{R}^3$ in O_G and a point, $P_C \in \mathbb{R}^3$, in O_C is defined as follows in Equation (2.2).



(a) Body–fixed (b) Global frame

Figure 2.2: Relating body–fixed and global frames to the camera frame

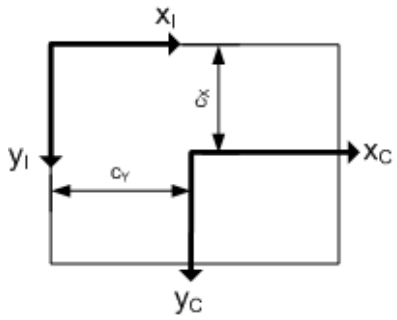
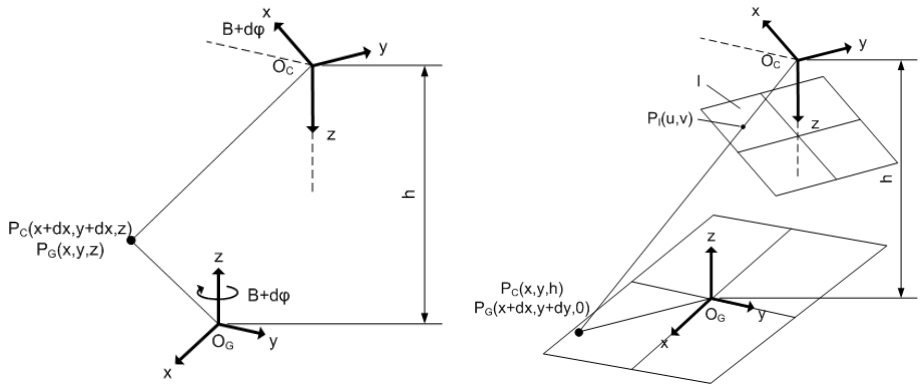


Figure 2.3: Relating the camera frame to the image frame

$$\begin{bmatrix} P_G^x \\ P_G^y \\ P_G^z \\ 1 \end{bmatrix} = \begin{bmatrix} \sin(\beta-\psi) & -\cos(\beta-\psi) & 0 & y \\ -\cos(\beta-\psi) & -\sin(\beta-\psi) & 0 & x \\ 0 & 0 & -1 & z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P_C^x \\ P_C^y \\ P_C^z \\ 1 \end{bmatrix} \tag{2.2}$$

The offset of the vehicle from the origin of O_G is denoted by x, y, and z. The vehicle yaw is indicated by ψ, and a fixed angular offset between the camera frame and the body–fixed frame, β, is as depicted in Figure 2.2.

Due to Assumptions 1.1, 1.4, and 1.5 (all pixels observed correspond to points on the ground ($P_G^z = 0$), the ground itself is flat, and the height h is reliably provided by the onboard height estimator), all of the information necessary to resolve the scale ambiguity is available. This allows a point, P_I, to be related to the global frame as follows in Equation (2.3).



(a) Projection of point with scale ambiguity

(b) Scale ambiguity resolved by setting $Z_G = 0$

Figure 2.4: Coordinate frames

$$\begin{bmatrix} P_G^x \\ P_G^y \end{bmatrix} = \begin{bmatrix} x + \dfrac{z\cos(\beta+\psi)(c_y - P_I^v)}{f_x} + \dfrac{z\sin(\beta+\psi)(c_x - P_I^u)}{f_y} \\ y + \dfrac{z\cos(\beta+\psi)(c_x - P_I^u)}{f_y} - \dfrac{z\sin(\beta+\psi)(c_y - P_I^v)}{f_x} \end{bmatrix} \tag{2.3}$$

2.2 Optical Flow

Optical flow is a very common approach for estimating motion within image sequences. It measures the relative motion of brightness patterns in a sequence of images, and ideally represents relative motion between a camera and objects in the camera's field of view. Given two images and a point in the first image, optical flow involves examining a window around the point and attempts to find a matching window in the second image. The displacement between the windows is treated as the "flow" of the point [48]. It is an iterative algorithm which makes the following assumptions:

Assumption 2.1. *The overall image intensity varies uniformly over time, resulting in pixels which have a constant intensity with respect to their neighbours.*

Assumption 2.2. *The points being tracked do not move far between consecutive images.*

Assumption 2.3. *There are unchanging surface markings viewable in both images, or matching patterns of image intensities exist.*

Assumption 2.4. *The displacement of a point's neighbours (i.e. those within a window surrounding the point) are assumed to be constant.*

The general approach is as follows:

1. Select a point, p_i, in the first image to track and a corresponding window surrounding p_i.

2. Assume that p_i shifts between images by a displacement, $d = (u, v)$, to produce a new point, $p_j = p_i + d$, in the second image.

3. Assume that the brightness (i.e. intensity) of p_j is given by Equation (2.4), which can be linearized using Taylor expansion (Equation (2.5)). Applying Assumption 2.1, this can be expressed by Equation (2.6), where I_t is the overall change in image intensity over time.

$$I(p_i, t-1) = I(p_i + d, t) \tag{2.4}$$

$$I(p_i, t-1) \approx I(p_t, t) + I_x \cdot u + I_y \cdot v \tag{2.5}$$

$$\nabla I \cdot (u, v) + I_t = 0 \tag{2.6}$$

4. Since Equation (2.6) has two unknowns for one equation, use Assumption 2.4 to generate more equations. If the window used is wxw pixels in size, there are now w^2 equations for 2 unknowns.

5. Solve Equation (2.6) for (u, v) using least–squares.

6. Compare window around p_i and p_j. If the error is below a threshold ϵ or a maximum amount of iterations has been reached, exit. Otherwise, repeat the minimization step, initializing d with the current value.

7. Once the computations have been terminated, the optical flow is the difference between the points (d).

Though it is feasible to run this algorithm on every pixel in a given image, using a sparse set of features is much less computationally intensive than attempting to compute the motion of each pixel [29]. In addition, the algorithm can use simple feature selectors such as Shi & Tomasi [16] and FAST [49, 50]. This is advantageous due to the reduced processor load of these feature selectors when compared to the computational requirements of more complex feature selectors (Section 1.2).

The pyramidal approach implemented in the OpenCV framework is also effective at reducing the processor load of optical flow [51]. It enhances the existing process by first computing an "image pyramid" for both images. An image pyramid is a series of n images, each with half the resolution of the last (Figure 2.5). The optical flow routine is run on each pair of images at a given resolution, feeding the results of the current computation into

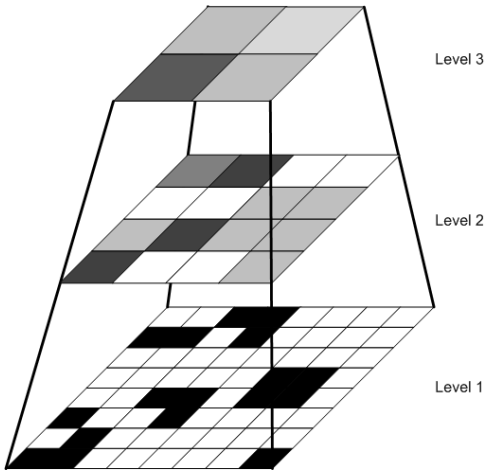


Figure 2.5: A 3–level image pyramid

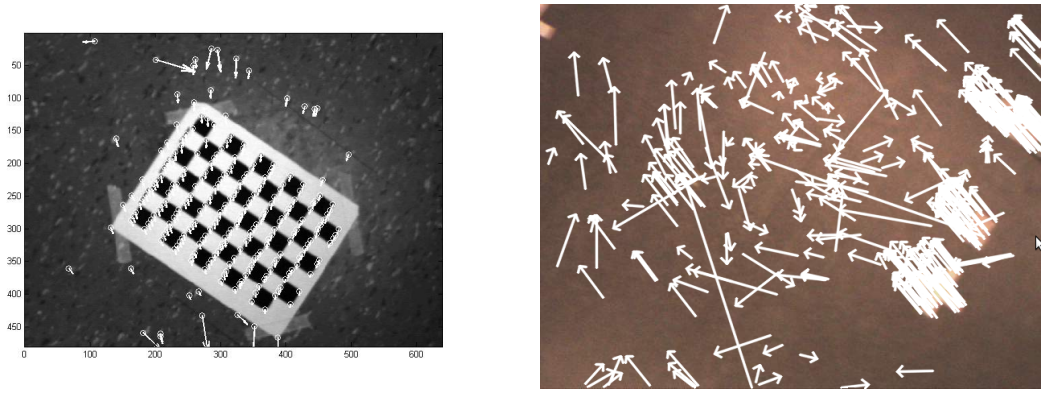
the next run as an initial condition. Using this approach, the system is able to converge to a solution much more quickly and is also more tolerant of local minima.

Once a series of flow vectors between two images have been determined, it is possible to determine the overall motion of the camera which has occurred between those images. Given additional information used to resolve the scale of features in the image, it is also possible to convert this information to real–world units. A downside does exist; a standard optical flow approach does not easily lend itself towards tracking points over more than two frames. This is caused by the use of a relatively simple feature selection process, where no attempts are made to make features invariant to scale or rotation. Without these properties, features will eventually get 'lost' when their original description fails to match their current description. This may prove to be an impediment if each point needs to be tracked over time.

In the case of low–contrast images, it is very likely that one or more resulting vectors is an "outlier" vector. These vectors are those for which accurate correspondence has not been found. Due to the gradient descent present in optical flow, such vectors will still degrade until they find another area somewhere in the image which appears representative of the area around the starting feature. They will thus indicate incorrect correspondence between areas in the two source images. This occurs with regularity with low–contrast images, no matter what the feature selection process is, something which is explored in Section 5.1. Despite the above noted downside, there has been work showing that a Shi & Tomasi selector can still find feature correspondences between images [46].

Shown below in Figure 2.6 are two examples of optical flow algorithms as run on sample data acquired from the quadrotor's onboard camera. Figure 2.6(a) is based on data acquired from test flights over a checkerboard pattern, and Figure 2.6(b) is taken

from tests run in a less feature–rich environment. 250 and 400 features were acquired, respectively. The presence of "outlier" vectors is visible in both images.



(a) Optical flow produced by vehicle motion over checkerboard target

(b) Optical flow produced by vehicle motion in test environment

Figure 2.6: Optical flow in test images

2.3 Camera Position Estimation

Given a number of optical flow vectors between images and an initial estimate of how the camera is positioned with respect to a global frame, the displacement of the camera over time can be calculated. In any two frames, I_1 and I_2, each tracked feature moves in the image plane by a certain vector $\overrightarrow{V_I}$. Since the environment is static, this motion is in fact caused by vehicle motion between the two frames. As well, due to Assumption 1.5, $\overrightarrow{V_I}$ has a corresponding vector, $\overrightarrow{V}$, which lies on the ground plane.

Equations (2.1) and (2.3) can be rearranged to form a function $T : \mathbb{R}^6 \rightarrow \mathbb{R}^2$ which relates the position of any point P_G in the world frame to a pixel, P_I, in the image given a vehicle position $[x, y, z]^T$ and yaw ψ with respect to the origin of the global frame. It is important to note in the following equations that a shift along one vector in the image in fact corresponds to the motion of the vehicle in the opposite direction.

$$\begin{bmatrix} P_I^u \\ P_I^v \end{bmatrix} = T(P_G^x, P_G^y, x, y, z, \psi) \tag{2.7}$$

Assuming every point in the image corresponds to a point on the ground, ($P_G^z = 0$), T is defined by

$$T = \begin{bmatrix} c_x - \frac{f_y(\sin(\beta+\psi)(P_G^x - x) + \cos(\beta+\psi)(P_G^y - y)}{h} \\ c_y - \frac{f_x(\cos(\beta+\psi)(P_G^x - x) - \sin(\beta+\psi)(P_G^y - y)}{h} \end{bmatrix} \tag{2.8}$$

If the vehicle's relative motion between two frames, $W = [dx, dy, dz, d\psi]^T$, is known, the flow at any point, P_I, can be determined by first applying Equation (2.3) to determine the corresponding location, (P_G^x, P_G^y), in the global frame. Then, the Jacobian of T with respect to (x, y, z, ψ) (represented as ∇T and defined in Equation (2.9)) is evaluated at the current vehicle position estimate, $[x, y, z, \psi]^T$. The vector $[x, y, z, \psi]^T$, is a subset of the elements of the state vector, X. The resulting ∇T can be used to determine how that particular point will move given vehicle motion, W, taking place over a time duration, Δt. In Equation (2.9), $\beta + \psi$ is defined as θ, $\cos(x)$ as $c(x)$, and $\sin(x)$ as $s(x)$ for brevity.

$$\bigtriangledown T = \begin{bmatrix} -\frac{f_y c(\theta)}{z} & -\frac{f_y s(\theta)}{z} & \frac{f_y((P_G^x - x)s(\theta) - (P_G^y - y)c(\theta))}{z} & -\frac{f_y((P_G^x - x)c(\theta) + (P_G^y - y)s(\theta))}{z^2} \\ -\frac{f_x s(\theta)}{z} & \frac{f_x c(\theta)}{z} & -\frac{f_x((P_G^x - x)c(\theta) + (P_G^y - y)s(\theta))}{z} & -\frac{f_x((P_G^x - x)s(\theta) - (P_G^y - y)c(\theta))}{z^2} \end{bmatrix} \tag{2.9}$$

Equations (2.3) and (2.9) combine to equal the expected new location point, (u, v), on the image plane, given its corresponding position in the global frame, P_G, vehicle motion, W, vehicle state, X, time duration between frames Δt, and initial pixel location, P_I. This is shown below in Equation (2.10).

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} x + \frac{z\cos(\beta+\psi)(c_y - P_I^v)}{f_x} + \frac{z\sin(\beta+\psi)(c_x - P_I^u)}{f_y} \\ y + \frac{z\cos(\beta+\psi)(c_x - P_I^u)}{f_y} - \frac{z\sin(\beta+\psi)(c_y - P_I^v)}{f_x} \end{bmatrix} - \nabla T W \Delta t \tag{2.10}$$

The location of a point, (u, v), at time k can also be expressed in terms of the location of (u, v) at time $k-1$ using Equation (2.11). The Jacobian, ∇T, is defined by Equation (2.9), where the vehicle position estimate at time $k-1$ is used for linearization.

$$\begin{bmatrix} u \\ v \end{bmatrix}_k = \begin{bmatrix} u \\ v \end{bmatrix}_{k-1} - \nabla T W \Delta t \tag{2.11}$$

The relative motion, (du, dv), of a location between time $k-1$ and time k is then given by Equation (2.12).

$$\begin{bmatrix} du \\ dv \end{bmatrix} = -\nabla T W \Delta t \tag{2.12}$$

The challenge is now to estimate the relative motion, W, given a set, $F = \{f_1, \ldots, f_n\}$ of n features $f_1, f_2, ...f_n$ ($f_i \in \mathbb{R}^2, \forall i \in \{0, 1, \ldots n\}$) and their corresponding motion vectors $\dot{f}_1, ...\dot{f}_n$ ($\dot{f}_i \in \mathbb{R}^2, \forall i \in \{0, 1, \ldots n\}$). Since dz is assumed to be known, ∇T can be expressed as $\nabla T = [\nabla P \ \nabla Q]$, where ∇P is the Jacobian with respect to the unknowns $dx, dy, d\psi$, and ∇Q is the Jacobian with respect to the known dz.

$$\nabla P = \begin{bmatrix} -\frac{f_y c(\theta)}{z} & -\frac{f_y s(\theta)}{z} & \frac{f_y((P_G^x - x)s(\theta) - (P_G^y - y)c(\theta))}{z} \\ -\frac{f_x s(\theta)}{z} & \frac{f_x c(\theta)}{z} & -\frac{f_x((P_G^x - x)c(\theta) + (P_G^y - y)s(\theta))}{z} \end{bmatrix} \tag{2.13}$$

$$\nabla Q = \begin{bmatrix} -\frac{f_y((P_G^x - x)c(\theta) + (P_G^y - y)s(\theta))}{z^2} \\ -\frac{f_x((P_G^x - x)s(\theta) - (P_G^y - y)c(\theta))}{z^2} \end{bmatrix} \tag{2.14}$$

A single optical flow vector can thus be related to P_G and W. Dividing the change in pixel location, (du, dv), between two consecutive frames by Δt produces a motion vector, $\dot{f}$, with components, $(\dot{f}^u, \dot{f}^v)$. Using $\dot{f}$ in conjunction with Equations (2.13) and (2.14) results in Equation (2.15).

$$-\nabla P|_{(P_G^x, P_G^y)} \begin{bmatrix} dx \\ dy \\ d\psi \end{bmatrix} = \begin{bmatrix} \dot{f}^u + dz\Delta t \nabla Q_1|_{(P_G^x, P_G^y)} \\ \dot{f}^v + dz\Delta t \nabla Q_2|_{(P_G^x, P_G^y)} \end{bmatrix} \tag{2.15}$$

By combining the entire set of features and their motion vectors into a linear system (Equation (2.16)), it is possible to solve via least–squares for $[dx, dy, d\psi]^T$. This vector is then added to the vehicle position estimate, $[x, y, z, \psi]^T$. As each flow vector is in $\mathbb{R}^2$ and the vector $[dx, dy, d\psi]^T$ is in $\mathbb{R}^3$, the minimum number of flow vectors required for such a solution is 2. However, due to the potential for errors, it is beneficial to have more than this number of vectors contribute to the least–squares calculation.

$$-\begin{bmatrix} \nabla P|_{(P_G^x, P_G^y)_1} \\ \nabla P|_{(P_G^x, P_G^y)_2} \\ \vdots \\ \nabla P|_{(P_G^x, P_G^y)_n} \end{bmatrix} \begin{bmatrix} dx \\ dy \\ d\psi \end{bmatrix} = \begin{bmatrix} \dot{f_1^u} + dz\Delta t \nabla Q_1|_{(P_G^x, P_G^y)_1} \\ \dot{f_1^v} + dz\Delta t \nabla Q_2|_{(P_G^x, P_G^y)_1} \\ \dot{f_2^u} + dz\Delta t \nabla Q_1|_{(P_G^x, P_G^y)_2} \\ \dot{f_2^v} + dz\Delta t \nabla Q_2|_{(P_G^x, P_G^y)_2} \\ \vdots \\ \dot{f_n^u} + dz\Delta t \nabla Q_1|_{(P_G^x, P_G^y)_n} \\ \dot{f_n^v} + dz\Delta t \nabla Q_2|_{(P_G^x, P_G^y)_n} \end{bmatrix} \tag{2.16}$$

Unfortunately, if image features are not consistently matched through multiple frames, this approach can allow estimation error to increase in an unbounded fashion over time. As well, if errors in feature correspondence are allowed to remain, outliers in the data will produce drastic changes in W if they exist in sufficient quantity.

The challenge then lies in choosing a set of vectors which are representative of the vehicle motion, given the presence of outlier vectors as shown in Figure 2.6 and the $O(n^3)$ nature of least–squares calculations [52].

2.4 RANSAC

Random sample consensus (RANSAC) is a technique developed for removing outliers from datasets while simultaneously fitting the data to a model of known structure [41]. It uses a set of data, F_{all}, and repeatedly chooses subsets, $F_i \subset F_{all}$. Each subset, F_i, contains m data points, where m is the minimum size required to estimate parameters M of a given model which describes F_{all}. Then, every other member, $f \in F_{all} \backslash F_i$, is evaluated to see how well the member fits with the model. If the member fits well, it is added to a consensus set, $F_c \subseteq F_{all}$. After the completion of n iterations of the above, the largest consensus set, B_c, is chosen for further refinement. The model is re–estimated by using only the features in B_c which agree with the model which originally generated B_c. Optionally, another iteration over F_{all} can be done to re–include features removed by the latest model re–estimation before a final model re–estimation is performed.

An outlier ratio, ϵ, is defined as the ratio between the number of outliers and the number of total data points in F_{all}, and p is defined as the desired probability of testing at least one outlier–free hypothesis. The outlier ratio is not known in advance, but is a property of each set, F_{all}. It has been suggested that an appropriate value, n, for the number of iterations is given by Equation (2.17) [34]. In Equation (2.17), the value of n increases with p due to the greater number of tests required to reject all outliers, and with m due to the increasing solution space of the model. Likewise, increases in ϵ also cause n to increase,

since a larger number of outliers will by necessity require more iterations before a model is generated from outlier–free data.

$$n = \frac{log(1-p)}{log(1-(1-\epsilon)^m)} \tag{2.17}$$

As an example, the RANSAC algorithm is applied to the estimation of parameters, (a, b), which define a line, $y = ax + b$. Data is selected from the set, F_{all}, shown in Figure 2.7(a). Figure 2.7(b) shows models being created (indicated by grey solid lines) from subsets F_i (indicated by large blue dots). Using the above straight–line case, each other member of F_{all} can be evaluated based on the shortest distance, d, between it and the line defined by $y = ax + b$ (indicated by grey dashed lines in Figure 2.7(b)). The consensus set, B_c, is then selected. Members of B_c which do not agree with the new model are rejected (indicated by large red dots), and the model re–estimated (Figure 2.7(c)). Finally, points removed from B_c may be re–included into B_c and the model re–estimated a final time (Figure 2.7(d), re–included feature indicated by a large green dot).

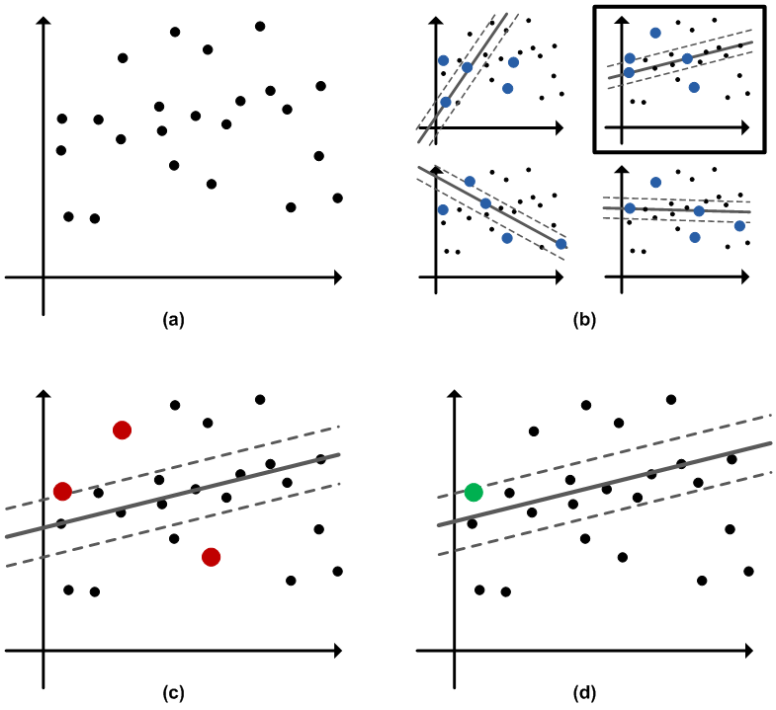


Figure 2.7: Using RANSAC to fit a straight line

To increase the robustness of the estimate, criteria such as absolute minimum size or minimum variance can be placed on B_c and checks performed before the re–estimation process. These checks can allow appropriate actions such as marking the entire dataset as invalid — to be taken when poor data is provided.

RANSAC can be further improved by combining it with the Kalman filter to allow its outlier–rejection properties to compensate for the Kalman filter's assumption of normally–distributed data [42]. In addition, the incorporation of a motion model which reduces the value of m allows the amount of iterations of the RANSAC procedure to be decreased [34]. Generally, this process can be applied to optical flow data by treating the magnitude and origin of each flow vector as a data point in $\mathbb{R}^4$ (with possible outliers) and modeling the motion of the image frame based on a subset of this flow data.

2.5 Kalman Filter

A Kalman filter is a state estimation algorithm which incorporates measurements received over time with a state transition model to optimally estimate a state vector. It assumes that noise is present, but assumes the noise is Gaussian. This optimal estimate minimizes the mean squared error between the state and the estimate. It is a recursive estimator; only the current state vector, the state transition model, and the current measurement are required to compute the next state vector. Typically, it is split into a "prediction" and a "correction" step. The former step represents the application of the state transition model to the state vector to generate a new estimate, and the latter step an adjustment in this estimate which combines a measurement vector with a measurement model describing how the measurement model relates to the state vector. In vehicle state estimation, the effect of the state transition model (known as the "motion model" in this application) will depend on the current vehicle control inputs. For example, the change in state over time for a car with no throttle applied will be different than the change of state over time for a car with its throttle applied at maximum.

For the 2–D position estimation of a quadrotor, there are a variety of ways which state estimation systems can incorporate knowledge about the vehicle's kinematics and dynamics. The simplest by far is a constant velocity model, which tends to serve as the default in studies where no additional information about vehicle motion is known [34] [43] [53]. Given a state vector, $X_P = [x, y, \dot{x}, \dot{y}]^T, X_P \in \mathbb{R}^4$, and a quadrotor vehicle, the constant velocity model can be expressed as follows in the global frame, where Δt is the period of the

estimator, and $[w_x, w_y]^T$ represents noise in the model caused by unknown accelerations.

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_k = \begin{bmatrix} x + \dot{x}\Delta t \\ y + \dot{y}\Delta t \\ \dot{x} \\ \dot{y} \end{bmatrix}_{k-1} + \begin{bmatrix} 0 \\ 0 \\ w_x \Delta t \\ w_y \Delta t \end{bmatrix}_k \tag{2.18}$$

Civera et al. argues that the more information used from onboard sensors, the better [34]. If it can be assumed that accurate vehicle orientation is available at all times and that wind resistance is negligible, a orientation–based model such as that provided in Equation (2.19) can be used to augment the state estimate. Acceleration due to gravity is represented by g, and (ϕ, θ, ψ) are vehicle Euler angles with respect to the global frame.

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_k = \begin{bmatrix} x + \dot{x}\Delta t \\ y + \dot{y}\Delta t \\ \dot{x} \\ \dot{y} \end{bmatrix}_{k-1} + \begin{bmatrix} 0 \\ 0 \\ g(\tan(\theta)\cos(\psi) - \tan(\phi)\sin(\psi)) + w_x \\ g(\tan(\theta)\sin(\psi) + \tan(\phi)\cos(\psi)) + w_y \end{bmatrix}_k \Delta t \tag{2.19}$$

Finally, the orientation of the vehicle can be further combined with onboard accelerometer data to improve the acceleration estimate, producing the dynamic model expressed by Equation (2.20). Values read from accelerometers along each axis, (ax, ay), is assumed to be already rotated into a frame which is level to the ground based on the orientation estimate. These accelerometer values represent drag and disturbance forces resisting the acceleration due to gravity on which the orientation estimate is based.

$$\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}_k = \begin{bmatrix} x + \dot{x}\Delta t \\ y + \dot{y}\Delta t \\ \dot{x} \\ \dot{y} \end{bmatrix}_{k-1} + \begin{bmatrix} 0 \\ 0 \\ (g\tan(\theta) - ax)\cos(\psi) - (g\tan(\phi) - ay)\sin(\psi) + w_x \\ (g\tan(\theta) - ax)\sin(\psi) + (g\tan(\phi) - ay)\cos(\psi) + w_y \end{bmatrix}_k \Delta t \tag{2.20}$$

In both the kinematic and dynamic vehicle models, incorrect estimates of orientation or acceleration may be present due to noise or sensor bias. If this is the case, the motion estimate may likewise experience variance or offset [13].

The motion estimates produced by any of these preceding vehicle models are then combined with a measurement model. This work uses optical flow vectors as measurements and a measurement model based on Equation (2.10) from Section 2.3. The current state estimate is used to predict the magnitude and direction of optical flow vectors in arbitrary locations on the image plane, which are then used to correct the motion estimates.

Using the traditional definition of a Kalman filter [14]:

$$\begin{aligned}
\hat{X_{Pk}}^{-} &= A_k \hat{X}_{k-1} + B_k u_k \\
\Sigma_k^- &= A_k \Sigma_{k-1} A_k^T + R_k \\
K_k &= \Sigma_k^- C_t^T (C_k \Sigma_k^- C_k^T + Q_k)^{-1} \\
\hat{X}_{Pk} &= \hat{X_{Pk}}^{-} + K_k (y_k - C_k \hat{X}_k^-) \\
\Sigma_k &= (I - K_k C_k) \Sigma_k^-
\end{aligned} \tag{2.21}$$

The first two equations encompass the prediction step and the last three the correction step. In this work, the measurement vector y_k is of size $2n$, representing a stacked set of n optical flow vectors $[f_1^u, f_1^v, ...f_n^u, f_n^v]^T$, with each pair having a corresponding feature located at (P_{I1}^u, P_{I2}^v). Covariance matrices, R_k and Q_k, are also defined, where the former matrix is the covariance of the motion model and the latter is the covariance of the measurement vector. The vector y_k is related to the state vector, X_P, by defining $C_k = \bigtriangledown T \Delta t$, where $\bigtriangledown T$ is the state–independent Jacobian as defined in Equation (2.9).

2.6 Control Overview

A quadrotor is controlled by varying the thrust in each of its four propellers. As shown in Figure 2.8, two of its propellers spin clockwise, and two counter–clockwise. If propeller #1 is producing a different amount of thrust then #3, the vehicle will pitch. Likewise, if propeller #2 is producing a different amount of thrust then #4, the vehicle will roll. If the combined thrust of propellers #1 and #3 is different then that of #2 and #4, the vehicle will yaw. Finally, overall climb rate is govered by the sum total of all propeller thrusts.

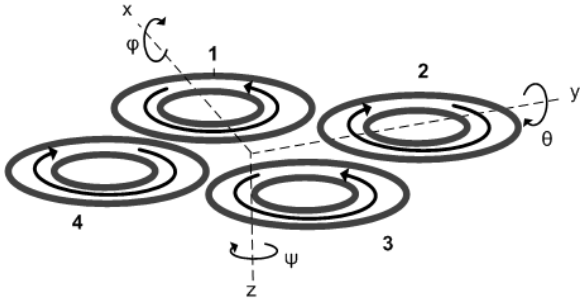


Figure 2.8: Quadrotor layout

Quadrotors are often controlled using cascaded control loops as shown in Figure 2.9, detailed extensively in [6].

The innermost loop is a very high–speed loop which controls individual propeller thrusts by varying motor voltage and can be modeled as a unity transfer function. The dynamics associated with the closed loop plant can be ignored in the model for vehicle position and orientation when vehicle state estimation and trajectory planning is being executed. Next, an orientation controller uses estimates of vehicle orientation and rotation rates to ensure that the vehicle is able to attain an arbitrary orientation within expected bounds. Finally, the outermost position controller takes a desired high–level trajectory or path and uses a model of the vehicle to generate corresponding roll, pitch, and yaw setpoints which will result in the vehicle successfully following the high–level input. In the case of the orientation and position controllers, development is done by linearizing the plant about 0° pitch and roll (ϕ and θ).

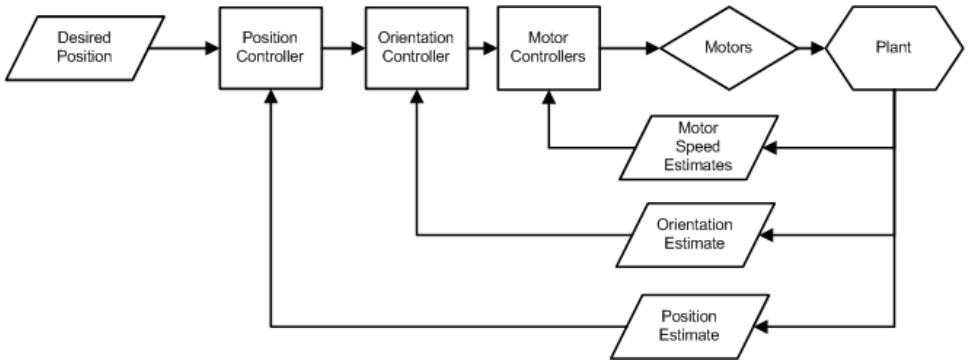


Figure 2.9: Nested control loops

This architecture is feasible due to the relative speeds of each control loop. Brushless motor control loops are able to control motors at frequencies upwards of 10 KHz, limited by the dynamics of the propeller and the electrical capabilities of the motor driver. These control loops use a technique where they energize coils within the motor at a rate proportional to the desired output speed and use a measurement of the output shaft position to guide which coil should be energized next.

Likewise, orientation controllers operate at the rate they receive orientation data from the IMU, which is typically between 50 and 200 Hz. The most important property of this particular controller is its ability to regulate both the vehicle's roll and pitch to 0° when necessary, keeping it in the air in the face of wind gusts and other disturbances. One example of a typical orientation controller is a set of PID-DD controllers which are coupled together to control output. In this case, the vehicle's dynamics are decoupled into four independent plants ($P_\phi, P_\theta, P_\psi, P_t$, governing roll, pitch, yaw rate, and climb rate, respectively). Then, an individual PID–DD controller is applied on each plant, producing the corresponding outputs $u_\phi, u_\theta, u_\psi, u_t$. The outputs are superimposed as shown in Equation

(2.22) to provide four motor outputs, $u_1 \dots u_4$, one for each propeller.

$$\begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} = \begin{bmatrix} u_\theta - u_\psi + u_t \\ -u_\phi + u_\psi + u_t \\ -u_\theta - u_\psi + u_t \\ u_\phi + u_\psi + u_t \end{bmatrix} \tag{2.22}$$

Finally, position controllers operating at 10 to 20 Hz are capable of tracking trajectories in 2–D and maintaining a hover despite disturbances. Simple PID or lead–lag controllers can be adequate for these purposes. They typically produce as control outputs desired accelerations in the global frame, (a_x, a_y). These accelerations can then be rotated into corresponding accelerations in the vehicle frame, (a_x^v, a_y^v), by Equation (2.23), where ψ is the current vehicle heading in the global frame.

$$\begin{bmatrix} a_x^v \\ a_y^v \end{bmatrix} = \begin{bmatrix} a_x \cos(\psi) + a_y \sin(\psi) \\ a_x \sin(\psi) - a_y \cos(\psi) \end{bmatrix} \tag{2.23}$$

These accelerations can then be translated into a desired orientation, (ϕ, θ), by applying Equation (2.24), the inverse of the vehicle kinematics described in Equation (2.19), where g denotes the magnitude of acceleration due to gravity.

$$\begin{bmatrix} \phi \\ \theta \end{bmatrix} = \begin{bmatrix} \tan^{-1}(a_y/g) \\ -\tan^{-1}(a_x/g) \end{bmatrix} \tag{2.24}$$

In the scenario of tracking a trajectory in 2–D without specified orientation or height, the other two inputs to the orientation controller, (ψ, climb rate), can be set arbitrarily.

Of these three control loops, there is only one which requires a position estimate as input is the outermost loop; the position controller. The information required for the other two loops can be sensed, regardless of the availability of a position estimate. The motor control loop only requires motor shaft position, and the vehicle orientation can be sensed entirely by an onboard IMU. Very little correction to these two measurements would be provided by incorporating position data. This further justifies a cascaded approach and enables the use of vision, as processing frames at 10-20 Hz is feasible.

Chapter 3

Position Estimation

3.1 Motivation

It has already been shown in Section 2.2 that only two optical flow vectors are necessary to estimate the relative motion of an aerial vehicle in an xy plane when height information is available. Existing algorithms presume that high quality features will be available in every image, but this is often not the case. The following situations often arise and lead to low quality images that can cause significant difficulties for state estimation:

- Operating in an environment which lacks patterns on surfaces reduces the total number of available image features.
- The lack of distinctiveness between existing features adds additional difficulty in determining specific feature correspondences between images.
- The low level of ambient light leads to an increase in image noise and a decrease in the reliability of optical flow calculations.

An example of the impact of reduced image and feature quality is shown in Figure 3.1. 400 features are selected as source points for optical flow. Quite a few vectors could be considered outliers, with their error going far beyond what would be expected with mere image noise. If 50% of the available vectors are outliers and only two vectors are being used to determine how the vehicle has moved, there is a significant chance that one of the two optical flow vectors used for the solution is an outlier. Even if Equation (2.16) is used and a least–squares solution is found, these outliers will greatly distort the result. In addition, the $O(n^3)$ nature of least-squares calculations suggests using all available features may be too costly to compute in real time, and the non-Gaussian form of the measurement

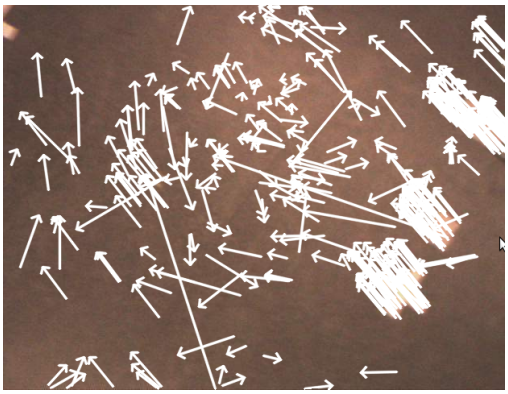


Figure 3.1: Optical flow produced by vehicle motion

noise ensures that even as $n \to \infty$, the least-squares solution may not approach the true solution.

This work proposes a solution based on RANSAC (introduced in Section 2.4). Specifically, it begins with the general framework developed by Vedaldi in [42] where a RANSAC–based approach is combined with a Kalman Filter, and continues with the objective of developing an algorithm which is applicable for the estimation of the planar displacement of a quadrotor UAV.

3.2 Algorithm Definition

The main inputs to the position estimation algorithm are monocular camera frames, and direct measurements of height and height rate. RANSAC is used to condition a set of noisy optical flow vectors to generate a filtered measurement set which is then used in an EKF–based position estimator. Providing an EKF with robustness to outliers in its measurement data is a primary benefit of this work. A secondary benefit is the lack in significant computational load when comparing the algorithm to a standard EKF.

The algorithm developed in this work follows the standard Kalman filter structure of predictions and corrections, and embeds the RANSAC method in the correction step. At each time step, the prior state estimate is propagated using a state transition model for the vehicle, and both altitude measurements and camera frames are captured for use in the measurement update.

Let Z be defined as $Z = [x, y, z, \psi, \dot{x}, \dot{y}, \dot{z}, \psi]^T)$, a subset of the elements of X defined in Section 1.1.2. This vector will be used as the state vector for the Kalman filter in place of $\hat{X_P}$ as defined in Section 2.5. At timestep k, the prediction step of the Kalman filter

applies a motion model to create $\hat{Z}_k^-$ based on $\hat{Z}_{k-1}$. If this is the first execution of the algorithm, $\hat{Z}_{k-1}$ is initialized to a known vehicle position. This initialization occurs only once per iteration of the estimator, outside the outlier rejection loop.

After the application of the motion model, the available features, F_{all}, are prefiltered based on $\hat{Z}_k^-$. Prefiltering to remove obvious outliers is known to be effective in improving accuracy [45, 46]. In this estimation algorithm, the features were prefiltered based on comparing flow vector length with the prior velocity estimate. Since sudden changes in direction are feasible, angles of flow vectors were not used as prefilter criteria.

Then, r features are randomly selected and a measurement model based on Equation (2.10) from Section 2.3 is built from these features based on their locations in the image plane and the height of the vehicle. The resulting measurement model is then used to formulate a Kalman gain K_k, which is applied along with $\hat{Z}_k^-$ and r flow vectors to generate a corrected estimate $\hat{Z}_{tmp}$. At this point, the rest of the flow vectors f which were accepted by the prefilter are compared to those generated by $\hat{Z}_{tmp}$ and the measurement model. If the error is less than a set threshold, f is added as per the standard RANSAC formulation to the consensus set F_c. This continues for n iterations.

The final $\hat{Z}_k$ is selected as the estimate corresponding to the iteration in which F_c has the largest number of elements. A minimum size for the consensus set is enforced, reducing the likelihood that an estimate based on an excessively noisy image is admitted. If no consensus set is larger than this minimum size, a zero motion vector is assumed. This assumption is due to the observation that a great deal of images where no consensus could be found resulted from a stationary camera. Once the iterations are complete, $\hat{Z}_k$ is set to the estimate found which is supported by the most features, and the covariance Σ_k is updated by the Kalman filter update equations as shown in Equation (2.21).

The combination of RANSAC and the EKF is done in the Kalman gain update step. The above approach runs the EKF during each RANSAC run, with the processor time remaining similar because the RANSAC model calulation is replaced in a nearly one–to–one way with a Kalman gain update of the same size. From here, speeding up execution further can be done by reducing the number of iterations, n, of the process. The suggested number of iterations to be used is given by Equation (2.17) in Section 2.4, and has been reprinted below.

$$n = \frac{log(1-p)}{log(1-(1-e)^m)} \tag{3.1}$$

Since there are three variables being estimated (x, y, ψ) by the algorithm and each feature is in $\mathbb{R}^2$, $m = 2$.

Finally, Equation (2.21) inverts a matrix to compute the Kalman gain, K_k. This can be mathematically sensitive with certain matrix libraries [54]. This particular part has been restructured to an $Ax = b$ form and solved for x, where $x = K_k^T$.

$$(H_k \Sigma_k^- H_k^T + R_k)^T K_k^T = H_k^T (\Sigma_k^-)^T \tag{3.2}$$

A summary of the algorithm is presented below in Algorithm 1. The motion model is defined as $\mathbf{f}(\hat{Z}_{k-1}, u_k)$, where the control input, u_k, is either a zero vector, the orientation vector of the vehicle, or the combined set of the orientation vector and the acceleration vector. These models are represented by Equations (2.18), (2.19), and (2.20), respectively. The function **prefilter_flow(**$F_{all}, \hat{Z}_k^-$**)** uses the current estimate of vehicle velocity from estimate, $\hat{Z}_k^-$, to reject elements of F_{all} which are obvious outliers. Likewise, **select_m_features(**F_{filt}, m**)**, randomly selects m features from F_{filt}. In this case, $m = 2$. During each iteration, a measurement model, C_k is created by **measurement_model(**$\hat{Z}_k^-$**)**, which uses Equation (2.10) to determine the magnitude and direction of optical flow vectors originating at each feature in F_i given a current state estimate, $\hat{Z}_k^-$. The covariance of the motion model is represented by R_k and that of the measurement model by Q_k. As well, an error function, $e(f, \overrightarrow{V})$, is defined. This function returns the difference between a provided feature, f, and a predicted flow vector, $\overrightarrow{V}$. If this difference is less than an error threshold, ϵ_f, f will be added to the consensus set, F_c. Finally, the consensus set which is selected after n iterations must not only be the largest, but must also be larger than a minimum size, Ω, for the results to be considered valid.

Since RANSAC is $O(n(2r)^3)$ (where n is the number of iterations and r is the number of optical flow vectors used to create F_c in each iteration), and an EKF is $O(s^3)$ (where s is the size of the measurement vector), one might expect that the resulting combined algorithm is the sum of both computational loads. In fact, the matrix inversion that makes RANSAC $O(n(2r)^3)$ can be combined with the $O(s^3)$ Kalman gain update. When the RANSAC inversion and the Kalman gain update are combined by replacing the model fitting process with the Kalman measurement update, a matrix inverse of the same complexity is required. The measurement vector is of size $s = 2r$ within each iteration, making the overall computational cost over all of the iterations $O(n(2r)^3)$, equivalent to the cost of RANSAC alone.

There is a final $O(s^3)$ Kalman gain update which occurs on the consensus set after the RANSAC portion has been completed, but the outlier rejection that RANSAC performs results in the set of measurements used to recalculate the Kalman gain (the largest consensus set, B_c), being much smaller than the total set of optical flow vectors F_{all}. The size difference between B_c and F_{all} which results when features are rejected is significant because of the resulting Kalman gain update being cubic in a smaller number.

The primary justification for combining RANSAC and Kalman filters was the complementary strengths of each approach. RANSAC is designed to be tolerant of outliers in a data set, but has no ability to take into account the covariances of the motion and measurement estimates. Likewise, Kalman filters are capable of fusing data from a variety of sources, but are not robust to non–Gaussian noise. The combination of both approaches provides a Kalman filter with outlier–free data and removes the need for RANSAC to take motion and measurement covariances into account.

Algorithm 1 Incorporation of Kalman Filter into RANSAC

Require: $n > 0, m > 0, \hat{Z}_{k-1}, \Sigma_{k-1}$, size$(F_{all})> 1$
  $i \Leftarrow 0$
  $B_c \Leftarrow \emptyset$
  $\hat{Z}_k^- \Leftarrow f(\hat{Z}_{k-1}, u_k)$
  $\Sigma_k^- \Leftarrow \nabla f(\hat{Z}_{k-1}, u_k)\Sigma_{k-1}(\nabla f(\hat{Z}_{k-1}, u_k))^T + R_k$
  while $i < n$ **do**
    $F_{filt} \Leftarrow$ **prefilter_flow**$(F_{all}, \hat{Z}_k^-)$
    $F_c \Leftarrow$ **select_m_features**$(F_{filt,m})$
    $C_k \Leftarrow$ **measurement_model**$(\hat{Z}_k^-)$
    $K_k \Leftarrow \Sigma_k^- C_t^T (C_k \Sigma_k^- C_k^T + Q_k)^{-1}$
    $\hat{Z}_{tmp} \Leftarrow \hat{Z}_k^- + K_k(F_c - C_k \hat{Z}_k^-)$
    for all $f \in F_{all}, f \in F_{all} \setminus F_i$ **do**
      $\overrightarrow{V} \Leftarrow$ **measurement_model**$(\hat{Z}_{tmp})\hat{Z}_{tmp}$
      if $e(f, \overrightarrow{V}) < \epsilon_f$ **then**
        $F_c \Leftarrow F_c \cup f$
      end if
    end for
    if size$(F_c) > \Omega$ **and** size$(F_c) >$ size(B_c) **then**
      $\hat{Z}_k \Leftarrow \hat{Z}_{tmp}$
      $B_c \Leftarrow F_c$
    end if
    $i \Leftarrow i + 1$
  end while
  $C_k \Leftarrow$ **measurement_model**$(\hat{Z}_{k-1})$
  $K_k \Leftarrow \Sigma_k^- C_t^T (C_k \Sigma_k^- C_k^T + Q_k)^{-1}$
  $\hat{Z}_k \Leftarrow \hat{Z}_k^- + K_k(B_c - C_k \hat{Z}_k^-)$
  $\Sigma_k \Leftarrow (I - K_k C_k)\Sigma_k^-$
  return $\hat{Z}_k, \Sigma_k, B_c$

Chapter 4

Experimental Platform

Experiments and data collection were carried out in a variety of different environments using an Aeryon Scout quadrotor as the vehicle. Initially, sample images and video were captured using only an onboard camera. As the system was refined, additional capabilities were added to provide benchmarking and additional sensor inputs. As well, the initial MATLAB–based software was refined over time into a C++ program which is capable of interfacing with the platform and running the entire estimation & control algorithm online.

4.1 Aerial Vehicle

The test platform used for this study is an Aeryon Scout quadrotor UAV as shown in Figure 1.1. It has been designed for both indoor and outdoor use, and is capable of maintaining platform stability even when it is exposed to severe wind gusts. An onboard embedded computer has access to sensor data and the ability to communicate wirelessly to a base station. The platform has an operational duration of 20 minutes at a factory weight of 1 kg. It is capable of sustained speeds of up to 14 m/s, and is commonly used for police, military, security, and environmental monitoring purposes.

The system has a set of sensors which are primarily used for stability control. A sonar ranger and pressure sensor are used for determining height above ground and climb rate, and a 6–DoF IMU equipped with 3 rate gyros, 3 accelerometers, and 3 magnetometers is used to observe inertial motion, vehicle orientation, and vehicle heading. Also onboard is a DGPS-capable GPS system for outdoor operation.

Additionally, the quadrotor is capable of being equipped with a variety of camera systems, including thermal imagers and colour cameras. A 2 MP fixed–focus colour camera with built–in tilt–compensation is mounted to the WAVELab platform. Custom code has

been written to interface with the Aeryon camera libraries and produce timestamped JPEG files for offline analysis, allowing for camera data to be easily associated with the inertial sensors.

Section 2.6 explains generally how quadrotors control their position. Of the control requirements explained in Section 2.6, the Aeryon hardware is responsible for both motor and orientation control (the two innermost loops in Figure 2.9). A position controller originally developed by Peiyi Chen of the WAVELab was implemented as part of the test environment [55]. It is a lead–lag controller which uses as input position error computed by comparing a reference position against the position estimate, and produces desired vehicle acceleration, (a_x, a_y), in the global frame. Equation (2.23) is used to translate these global accelerations into a corresponding pair, (a_x^v, a_y^v), in the vehicle frame. Then, Equation (2.24) is used to generate roll and pitch setpoints, (ϕ, θ), necessary for position control.

4.1.1 AHRS Sensors

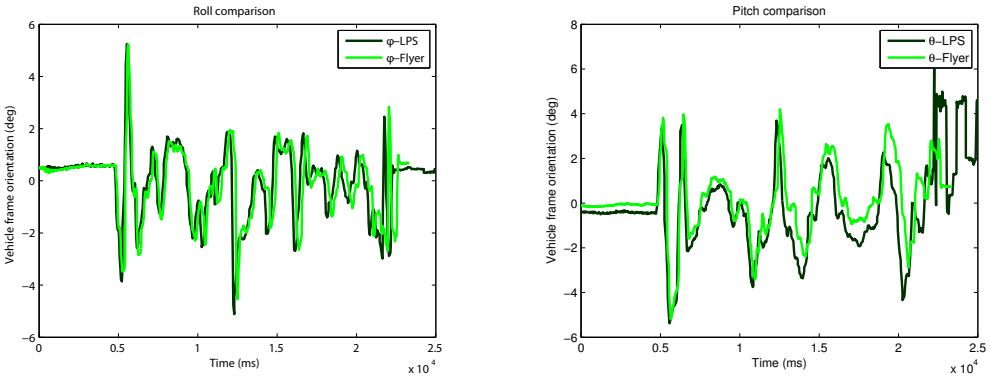
The Scout has a custom onboard IMU which feeds into a series of estimators responsible for determining orientation, acceleration, altitude, and climb rate.

A series of benchmarks were done to validate these estimators. Figure 4.1 is a comparison of the roll and pitch estimates as provided by the onboard orientation estimator with ground truth orientation data. In this particular example, the variance of the roll error is 6 deg^2, while the variance of the pitch error is 0.4 deg^2. Filter lag appears to be negligible. Using the orientation–based motion model laid out in Section 2.6, a standard deviation of $\sqrt{6}$ corresponds to an acceleration error of $\pm 0.41 m/s^2$, centred about 0. From this information, it seems that errors which could be introduced by the onboard estimator are not significant.

Figure 4.2 is a similar comparison, showing the onboard height estimate as compared to the OptiTrack ground truth. In general, the onboard estimator is accurate to within 5 cm with a variance of 1 cm^2. The height estimate does not incorporate the pressure sensor at all due to the low height of the vehicle.

4.1.2 Onboard Camera

The onboard camera is a 2 MP camera capable of taking 5 MP snapshots (Figure 4.3). Video is typically streamed over an MPEG4 connection. For experimental purposes, this was replaced with a stream of 320x240 raw greyscaled images. This allows individual images to be timestamped and also avoids the addition of compression artifacts to the images. Images are able to be reliably transmitted at 8 fps, though some variance occasionally occurs. It has been hypothesized that this variability is due to occasional network syncing.



(a) Performance of roll estimator (b) Performance of pitch estimator

Figure 4.1: Orientation estimator comparison

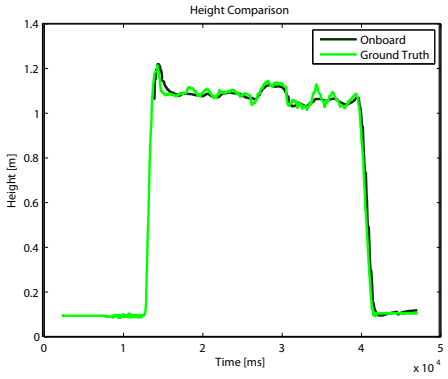


Figure 4.2: Height estimator comparison

The camera is mounted on a 2–axis gimbal which is tied directly to the vehicle's orientation estimator, allowing the system to compensate in near–real–time for vehicle pitch and roll. It is also rotated about the vehicle z axis by an angle $\beta = 45°$ with respect to the vehicle body frame to reduce the occlusion of the image by the vehicle's legs. This angle has a significant effect on the algorithm described in Chapter 3, since it factors significantly into all of the formulae found in Sections 2.1 and 2.3 which relate locations in the camera frame to locations in the global frame.

Figure 4.3: Onboard camera [8]

4.1.3 Communications

There are three ways through which the Scout can be controlled:

1. Microhard long–range modem
2. Onboard Wi–Fi
3. R/C remote

The first case is for field deployments where long–range use is required. The WAVELab platform is not so equipped at the moment. The onboard Wi–Fi provides a similar access method as the first option, though much higher bandwidth (54 Mbps vs. 1.2 Mbps). Either of these preceding methods allow for real–time command & control of the platform, including SSH and SCP access for debugging and log retrieval. Additional (non–Aeryon) ports can also be used for additional communications which is not provided by the Aeryon API (i.e. image transmission). Finally, control via a 6–channel R/C remote is an option. In all cases, this remote can be used as a safety kill switch. A software flag can also be set to place the system in a fully–manual mode where the R/C remote controls the vehicle orientation, yaw rate, and climb rate directly.

4.2 Local Positioning System

4.2.1 Network Topology

The experimental setup relies on two offboard computers participating in the same wireless network as the aerial platform as shown in Figure 4.4. The LPS Computer is responsible for receiving and interpreting ground truth data from an off-the-shelf local positioning system and providing it over a TCP/IP socket to the Estimation Computer. The Estimation

Computer displays and stores this data, and also provides a timestamped version to the Scout for onboard logging.

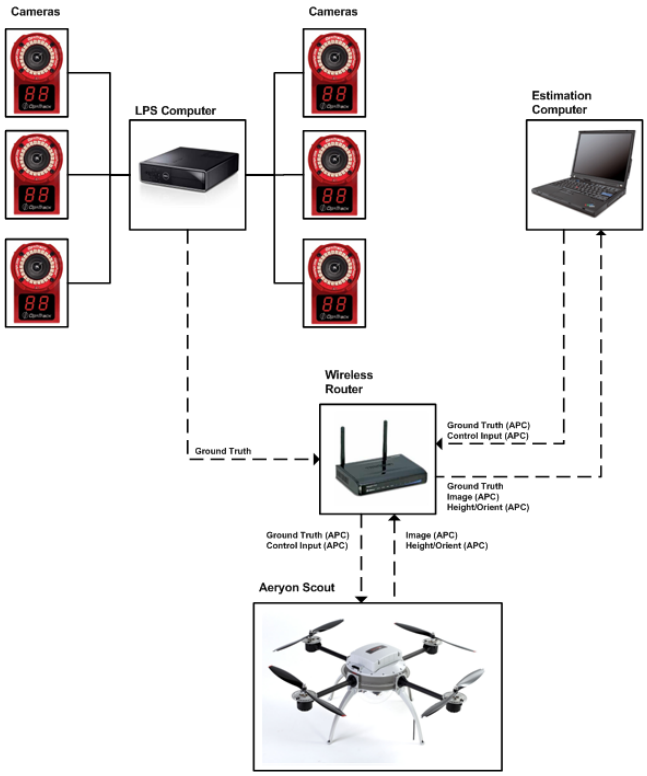


Figure 4.4: Network Diagram

The Scout itself provides a series of raw, timestamped, greyscale 320x240 images via a similar TCP/IP socket to the Estimation Computer. In addition, its full estimate of platform orientation and filtered height is provided via Aeryon's "APC" interface. The entirety of this information is processed by the Estimation Computer to produce state estimates as well as suitable roll/pitch/yaw/thrust control inputs for the Scout. In addition, the Estimation Computer also timestamps and echoes the ground truth data to the Scout in a format it can understand and log.

Ground truth is provided by a NaturalPoint OptiTrack local positioning system connected to the LPS Computer [27]. The OptiTrack system uses six near-IR cameras as shown in Figure 4.5 and passive reflectors to track the motion of rigid bodies within its capture volume. Along with its use as a reliable position reference, it has been useful for characterizing the performance of the UAV's on-board orientation estimator. Characterizations of the orientation estimates using this hardware were presented earlier in Section 4.1.1. The output from the local positioning system is the location of the origin of the

body–fixed frame, O_B, in terms of the global frame, O_G, with frames as defined in Section 2.1. Since the origin of O_B can be set arbitrarily on any point of the vehicle, it has been placed coincident to the origin of the camera frame, O_C. Though there remains a rotation between O_B and O_C, locating the two points at the same location removes the potential for an offset between the two points due to vehicle pitch and roll. If they were not located at the same location, a change in orientation of the vehicle would cause the two points to develop an offset with respect to each other in the global xy plane, adding error to the ground truth comparison.

Figure 4.5: OptiTrack camera [27]

4.2.2 Common Software Architecture

The software which runs on the Estimation Computer uses a common architecture developed by the WAVELab in late 2009. At this point in time, ROS [56] was not yet prevalent among the research community and most other options for the development of estimation algorithms were not developed with aerial vehicles in mind [57, 58]. The goal in the development was to abstract away the concepts of message passing and thread control from current and future lab members. Additionally, it was meant to encapsulate the Aeryon control API as much as possible.

In general, it uses a multithreaded Publish/Subscribe model [59]. There are three common structures in the architecture: Messages, Publisher and WaveThread. Messages are designed to be passed between threads, and are to be extended by users to contain whatever additional data is required. They also contain a small set of functions designed to allow other functions to check their types and provide identifiers which map to specific types, a feature lacking in C++ at the moment. Publisher is designed to do work with these Messages. A Publisher singleton is a threadsafe way of passing messages between threads via a subscription–based model very similar to the approach taken by Player/Stage or ROS. Finally, WaveThread is a base class which abstracts away key functions in the pthreads library, including thread creation, waiting on events, sleeping, and thread destruction.

Each WaveThread also provides a common message callback which the Publisher class knows how to deliver messages to.

To test the state estimation algorithm described in Chapter 3, the following threads are used:

- **LPS Thread:** This thread establishes communication with the LPS running on the LPS Computer and receives incoming ground truth information for reference and control purposes.
- **Estimator Thread:** The estimator thread contains and runs the position estimation algorithm.
- **Control Thread:** This module contains a position control loop which can regulate the vehicle to a stable hover or guide it through a trajectory. It can use both LPS data and position estimation data as sensor input.
- **Scout Interface Thread:** The Scout interface thread receives height and orientation data from Scout, provides LPS data and control inputs, and issues takeoff and landing commands to the Scout.
- **Image Thread:** This module receives timestamped images from Scout or reads in prerecorded images.
- **Keyboard Thread:** The keyboard thread allows operator to engage or disengage position control and swap the source of the position estimate used for vehicle control between the state estimation algorithm under test and ground truth information.

A depiction of the communciation between these threads is shown in Figure 4.6. The external libraries the software depends on are:

- **OpenCV:** Optical flow and image display [51]
- **pthreads:** C/C++ threading
- **libapc:** Aeryon Scout control interface

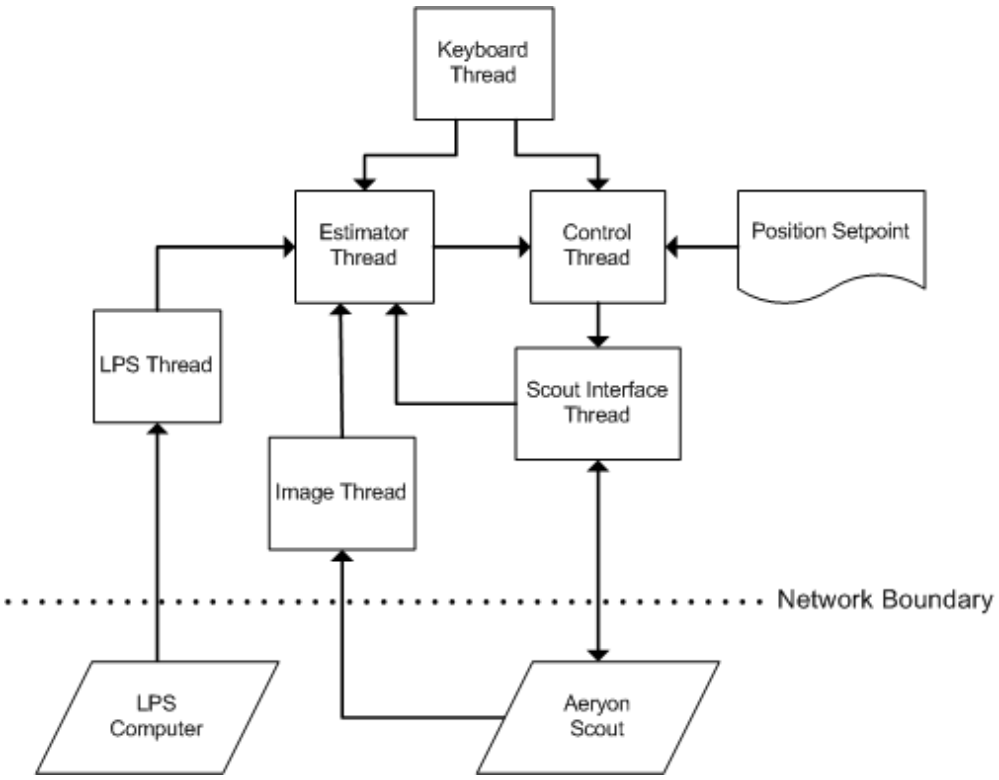


Figure 4.6: Estimation Computer Software Architecture

Chapter 5

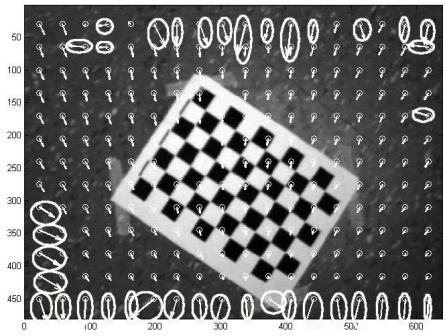
Experimental Results

Multiple types of experiments were performed to validate each part of the proposed algorithm. First, Section 5.1 examines the validity of using simple feature selection methods instead of more complex ones for optical flow, as well as the reasoning for using RANSAC. Next, Section 5.2 explores the various factors which affect processing time, including the significant effect of the wireless network. Section 5.3 shows the results of implementing the algorithm on the vehicle platform. Finally, Section 5.4 examines the outcome of using the algorithm for online vehicle position control.

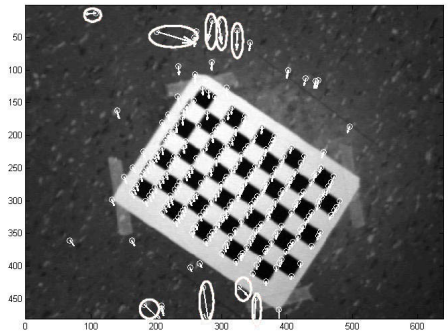
5.1 Feature Selection

Three feature selection methods have been compared, primarily based on their ability to produce features which can be tracked well between low quality image frames. The methods chosen are a naive equal grid spacing, Shi & Tomasi [16], and SURF [17], in order of computational complexity. An ideal feature selection method is one which produces trackable features without requiring significant processing time. Since this algorithm is designed for use as an estimate source for vehicle position control, feature selection methods which would prevent the algorithm from running online due to speed concerns should not be used.

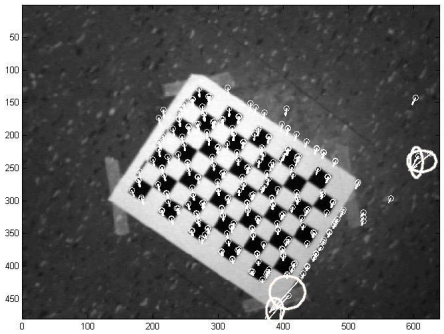
Shown below in Figure 5.1 are the results of using each feature selector in combination with pyramidal optical flow on an image containing a high-contrast black & white checkerboard target. In each case, only the most trackable 150 features have been displayed, where the definition of "trackability" is dependent on the selection algorithm. The exception is the evenly spaced features, where 150 points have been distributed equally on a square grid.



(a) Optical flow from evenly spaced features



(b) Optical flow from Shi & Tomasi features

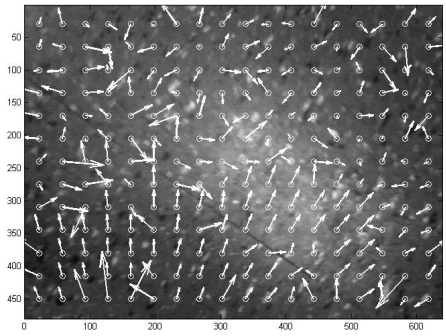


(c) Optical flow from SURF features

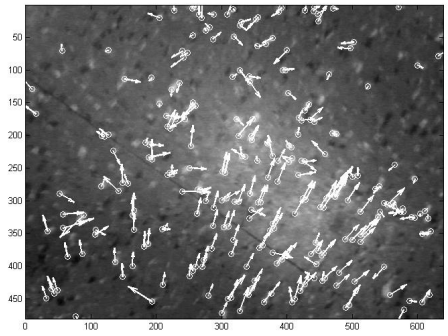
Figure 5.1: Feature selection & optical flow from a high-contrast image

Judging by the number of outliers in the vector field (circled in white), SURF produces the best results with 4 outliers. Shi & Tomasi follows with 9 outliers, and using an evenly spaced grid results in 38 outliers. The arrows superimposed on the image represent the point in the following frame which the feature corresponds to. The SURF features are nearly all consistent in defining the vector field. This is also true for the majority of the Shi & Tomasi-based features. Finally, the evenly spaced features produce the least consistent field. Ideally, there will be no outliers at all, and each flow vector will correlate with the motion of the vehicle.

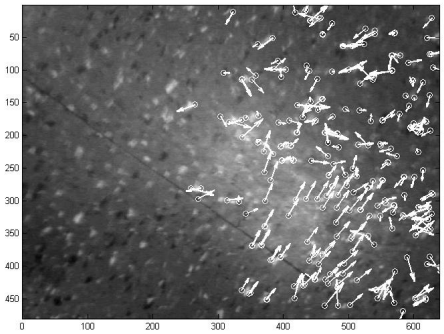
As a high-contrast environment cannot be assumed, tests were also performed using videos captured of surfaces which are largely devoid of features (cardboard, tile, clean concrete, etc). A sample of the results are shown in Figure 5.2. Here, every approach has a significant number of features which cannot be tracked well.



(a) Optical flow from evenly spaced features



(b) Optical flow from Shi & Tomasi features



(c) Optical flow from SURF features

Figure 5.2: Feature selection & optical flow from a low-contrast image

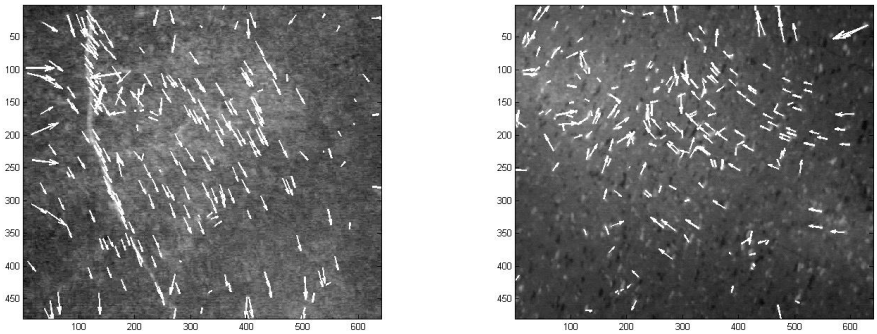
Selector	CPU Time (s)
Equal Spacing	0.003
Shi & Tomasi	0.373
SURF	5.399

Table 5.1: Processing load comparison between feature selectors

Though SURF clearly produces better results when presented with high-contrast images as input, it performs at a similar level to the others when it does not have easily trackable features. Numerous points for which correspondence errors result in poor flow vectors are still produced. However, using the MATLAB profiler and the images from Figure 5.1, its computational load was found to be higher than Shi & Tomasi by an order of magnitude. As algorithm runtime is independent of the number of image features, computational load will not change measurably between images.

Since the algorithm must be able to determine optical flow from low contrast images which do not have well-defined characteristics, there is a point at which the computational expense of complex feature selection algorithms does not provide any benefit. Therefore, given the computational cost of SURF, it is reasonable to use Shi & Tomasi instead. Shi & Tomasi is much faster, but can still take advantage of high-contrast features in images when they are present. Once the flow calculations are complete, a similarity measure is evaluated at each tracked point to determine if the result of the flow calculation is valid [16]. In order to simplify calculations, an assumption is made that each feature is only being altered by translation as opposed to undergoing a more complex affine transformation. This measure is not always effective, as the low-contrast nature of the images means that even evaluating similarity over a broad window can result in false positives, where a pixel in the image may be incorrectly classified as being "similar" to the point being tracked.

Figures 5.3(a) and 5.3(b) are more examples of the results of using optical flow with a Shi & Tomasi selector over different floor types (concrete and tile, respectively). Despite the poor quality, these figures have an underlying trend due to camera motion. Figure 5.3(a) contains flow vectors which would result from the camera moving towards the top left corner of the image, and vectors like those in Figure 5.3(b) result from counterclockwise camera motion centred around the top right corner of the image. As is expected, there are still many vectors which do not accurately represent the motion of the corresponding point in the global frame.



(a) Optical flow over a concrete floor (b) Optical flow over a tile floor

Figure 5.3: Unfiltered optical flow vectors

A primary justification for using an outlier rejection strategy before applying a Kalman filter is the fact that Kalman filters are ill–suited to non–Gaussian noise [14]. The characteristics of image noise were examined to validate the use of an outlier rejection process. Figures 5.4 through 5.6 show error histograms based on the set of total vectors in each corresponding source image frame. The source of the ground truth for each image is the

model after RANSAC has been applied. Each figure shows that both vector angle errors (a) and vector length errors (b) are non–Gaussian overall. This is most clearly shown in the various graphs of the angle errors. The distribution of the vectors ranges from a near perfect correspondence to the model to completely distributed around $[-\pi, \pi]$. This information would suggest that there are numerous outliers based on angle errors alone. As well, the rightmost graph (c) in each figure is an expansion of the centre bins of the vector length error histogram, showing a distribution which has a Gaussian bell shape. The number of outliers in terms of vector angles, and the appearance of a central Gaussian distribution with additional outliers in the distribution of the vector length errors validates the use of outlier rejection in the state estimator. Figures 5.4 through 5.6 all have angle and length outliers present.

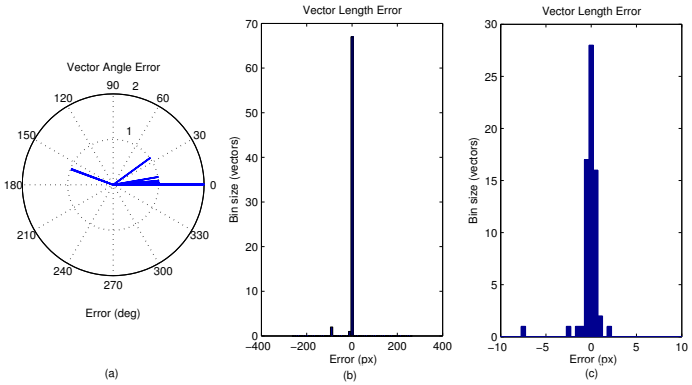


Figure 5.4: Image noise characteristics – Example # 1

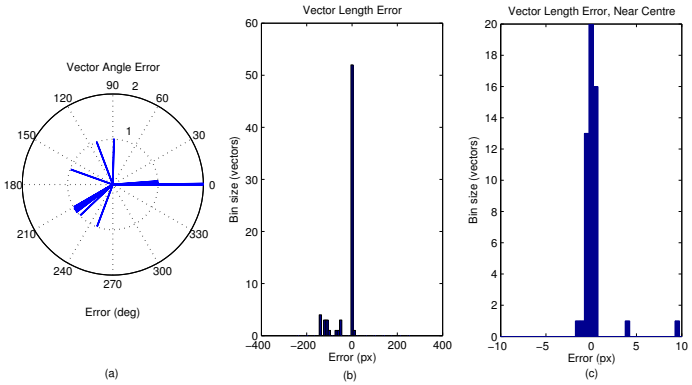


Figure 5.5: Image noise characteristics – Example # 2

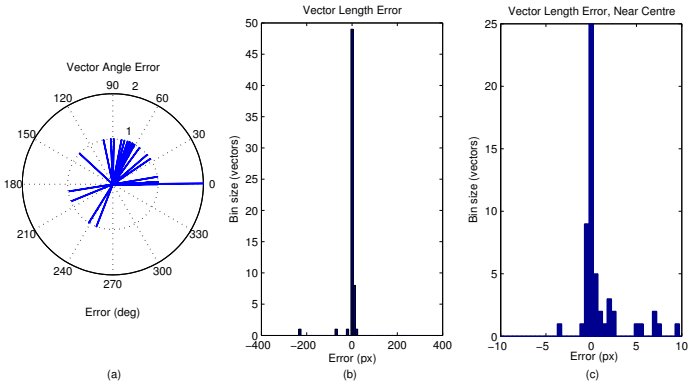


Figure 5.6: Image noise characteristics – Example # 3

The number of iterations to be used in each run of RANSAC also has an impact on the runtime and robustness of the algorithm. As described earlier, Equation (2.17) suggests a number of iterations, n, based partially on the outlier ratio, e. An attempt was made to set the environment's outlier ratio based on past data to allow the system to run faster when better quality features are present. It was found that the outlier ratio varied rapidly and unpredictably from $e < 0.1$ to $e > 0.9$ (Figure 5.7). The corresponding number of suggested iterations varies from $n < 10$ to $n > 500$. Given the rapid changes in outlier ratio, applying a strategy where the number of iterations used at the present moment adapts over time to past outlier ratios is potentially detrimental. For example, if the past outlier ratios would suggest that a small number of iterations are required and the current (unknown) outlier ratio has changed rapidly upwards, RANSAC may fail to explore a sufficient number of initial models. The resulting strategy was to fix the outlier ratio to 0.7 ($n \approx 60$), as Figure 5.7 suggests that this value is greater than is required for the majority of situations.

5.2 Processing and Communications

Communications:

If the position estimate is to be used as an input into a controller, the lag and update frequency of the system are important to study. There are three separate subsystems which impact these particular characteristics the most, denoted by thick dotted lines in Figure 5.8.

These subsystems' effects are as follows:

- **Camera library:** Due to a lack of low–level access to the camera hardware, the onboard streaming code must wait for a full frame to be captured before transmitting.

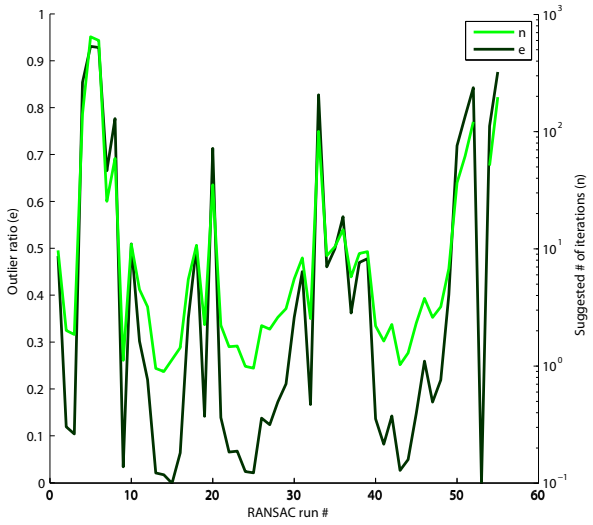


Figure 5.7: Outlier ratio over time

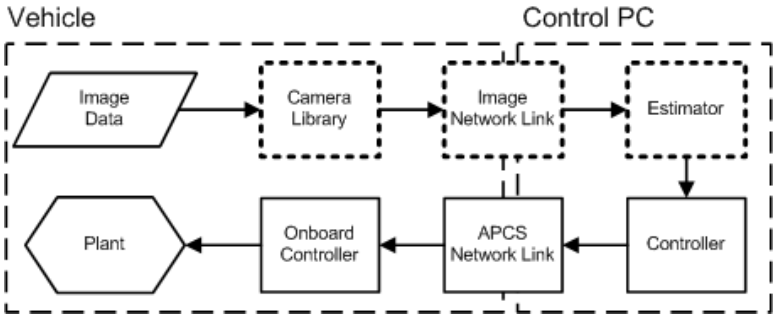


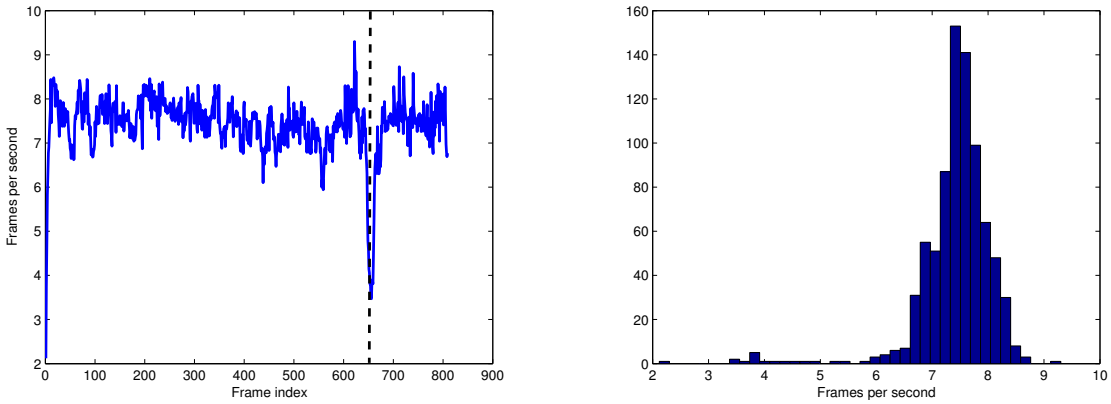
Figure 5.8: Key control and estimation subsystems

With the camera providing 320x240 images at a reported rate of approximately 17 fps, this adds a worst–case 60 ms in latency before the onboard streaming code is provided with a new frame to transmit.

- **Image Network Hardware:** Images were transmitted uncompressed over a Wi-Fi network to prevent more noise from being introduced into already low–contrast images. Assuming a 40 Mbps average bandwidth and 320x240 8-bit greyscale images, this allows for 65 frames to be transmitted per second, or 32 frames per second from the vehicle to the router and 32 frames per second from the router to the control PC. As well, the streaming library uses lossless TCP/IP for transmission, adding an unknown amount of latency due to network retransmission.

- **Estimator:** The upper bound on the frequency at which the estimator can produce new state outputs is merely the inverse of the wall–clock time required to complete one estimation run, as it is last in the critical path of the three systems detailed here. Various processing time results at different settings are presented later on in this section.

Overall, it was observed that the system as a whole was able to reliably capture and transmit images between 7 and 8 FPS, with occasional unexpected dips lower which are believed to be caused by network interruptions (denoted by the dashed line in Figure 5.9(a)).



(a) Image capture rate over time

(b) Image capture rate distribution

Figure 5.9: Image capture rate

For this image capture characterization, a dedicated network was used and the estimator was turned off. Thus, it appears that this bound is due to the camera library itself, and it appears to be caused by the choice of TCP/IP as the approach for transmitting images. UDP was implemented to attempt to mitigate this issue, but caused a severe reliability decrease. If the camera library is only able to capture images at between 7 and 8 fps and it is receiving input at 17 fps, there is approximately 75 ms in latency being added by the camera library in processing and transmission. This contributes directly to the total lag in the system, ensuring that there will always be at least $60 + 75 = 135$ ms in delay between the capture of a new image and the output of a new correction based on this image. Compared to the delay in transmitting images to the Control PC, the latency inherent in sending control commands from the controller on the Control PC to the onboard controller (Figure 5.8) is negligible.

Method	Mean Feature Set Size	Mean Est. Time (s)	Est. Time Variance (s^2)
Kalman only	78	207	$20.78 * 10^3$
Kalman only	196	1773	$5.73 * 10^6$
RANSAC	72	196	736
RANSAC	159	408	$21.78 * 10^3$

Table 5.2: Comparison against a Kalman filter only approach

Processing:

A significant benefit of the proposed algorithm lies in the reduced processor load of the state estimation algorithm when compared to an algorithm which does not filter the features for outliers. Though RANSAC sees estimation time increase with the number of features (Figure 5.10), the proposed algorithm ends up requiring as much as four times less overall time to generate an estimate. This is made more significant as the number of features (and thus the number of outlier features) increases, since they are filtered out before the time–consuming final update step which operates on every feature available to it. In comparison, least–squares will always operate on every available feature. Table 5.2 shows the effect on processor time which results from using the RANSAC–based approach in place of solving the entire feature set with a Kalman filter update. With a larger number of features, the RANSAC–based approach is over four times faster. In fact, the Kalman only approach often missed entire images when it was presented with a large number of features, due to increased processing time. With a smaller number of features, the two approaches are equivalent in speed. There are slight differences in the size of feature sets used in each comparison (78 vs. 72, 196 vs. 159) because the longer estimation time of solving the Kalman gain with the full feature set meant that it was not able to successfully process all of the images. Two rows are provided in Table 5.2 to illustrate the relative increase in time between methods as the mean feature set size increases.

It should be noted that there are worst–case limitations to the processor load reduction the proposed algorithm can produce. Specifically, if the image has a large number of features with similar optical flow vectors, the runtime of the algorithm will be worse than the approach where only a Kalman filter is used. Figure 5.10 shows a general trend towards an increase in required estimation time as the number of features in the image (including potential outliers) increases. This is due to RANSAC selecting a consensus set which is very large (with a size near or equal to the size of the full set of features) from a large initial data set and then re–estimating the model from this set. If such a consensus set is selected, the runtime will be worse. This situation is addressed by providing an adaptive feature threshold for the feature selection algorithm. If too many features are detected, the feature selection algorithm becomes more stringent in its selection. Likewise, the selection process will become more forgiving if too few features have been detected. Though the selection

process does not remove the possibility of the size of the consensus set occasionally being near to or equal to the size of the full set of features, using an adaptive threshold reduces the likelihood that the size of the full set of features will be too large to process.

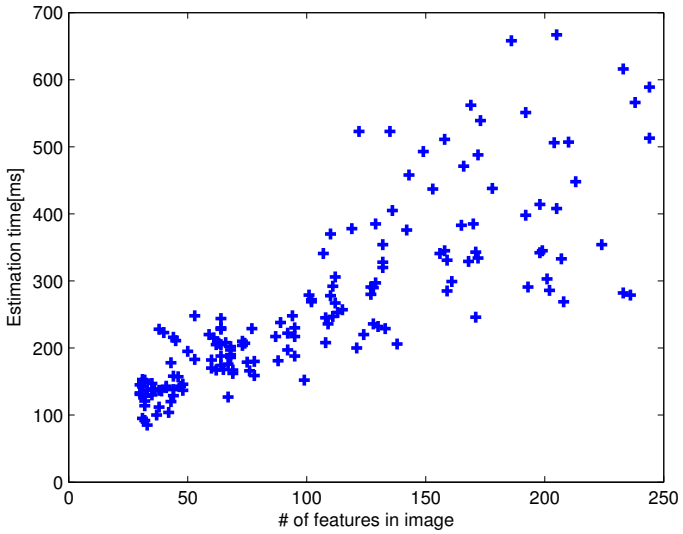
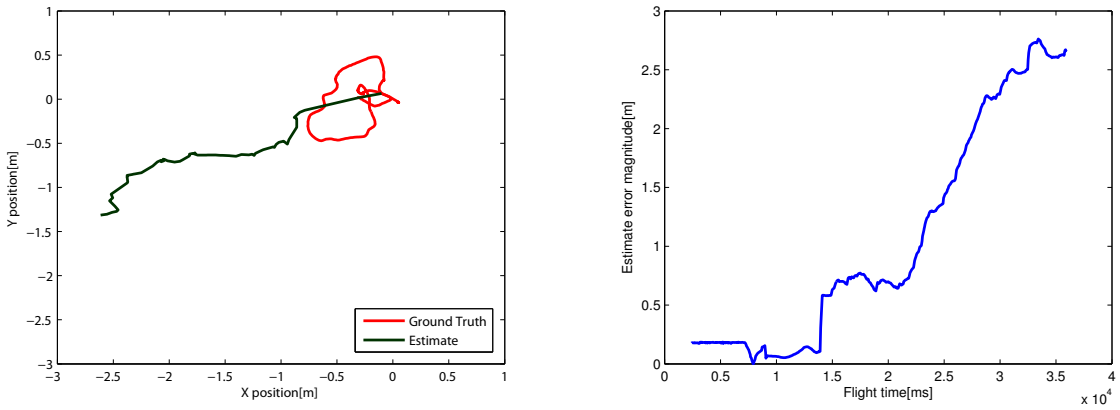


Figure 5.10: Number of flow vectors in image vs. estimation time

5.3 Estimation

The following experiments were done to validate the proposed algorithm in offline and online trials. The baseline used is a Kalman–filter based approach which does not use outlier rejection on the flow vectors (Figure 5.11). Even with a motion model in place, error rapidly increases. Figure 5.12 depicts the results of a number of variations of the state estimator based on common data from a single indoor test flight.

In all cases, 60 iterations of RANSAC were used for each correction step, and the local positioning system provided the vehicle's absolute heading. The primary difference between each state estimator is the motion model used. In the first case, a constant–velocity motion model as described in Equation (2.18) was implemented. Next, a kinematic orientation model like that laid out in Equation (2.19) was used. Finally, a dynamic model as given by Equation (2.20) was studied.

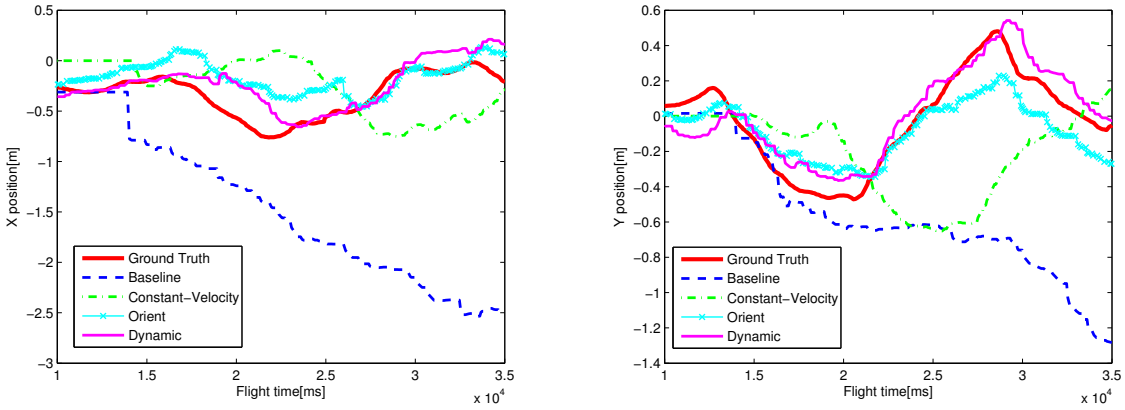


(a) Path display

(b) Estimation error

Figure 5.11: Kalman filter without outlier rejection

The results of these tests are presented in Figure 5.12. Figure 5.12(a) plots x–axis position estimates over time, along with ground truth data for reference. Likewise, Figure 5.12(b) does the same for the y–axis. The data used was the same data which was used to create Figure 5.11.



(a) X position over time

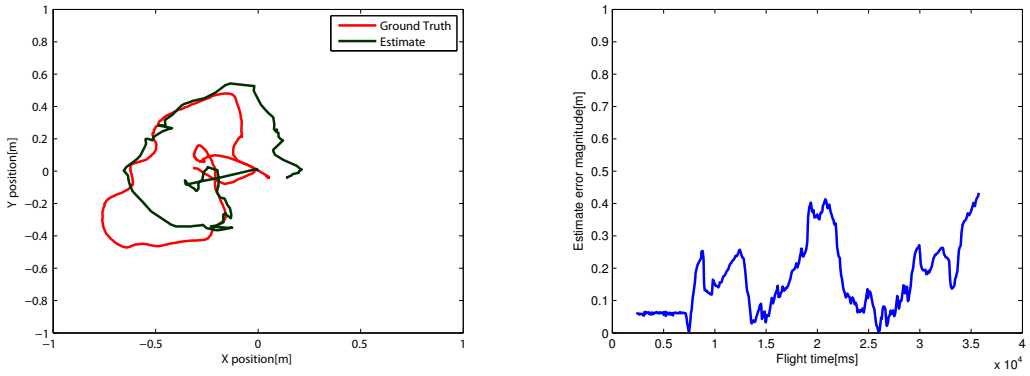
(b) Y position over time

Figure 5.12: Comparison of position estimates on a per–axis basis, using various motion models

It is clear from Figure 5.12 that the worst result is produced by the Kalman filter without outlier rejection. The constant velocity model improves on this result significantly, showing the same overall trend in motion as the ground truth, but lagging significantly

and accumulating error as time progresses. The orientation motion model and the dynamic motion model both track well ground truth over time, though there are occasional offsets in scale, and a phase shift due to latency. The best results are gained by using the dynamic motion model, likely because of the better prediction of the vehicle motion which results from the incorporation of additional sensor data.

Figure 5.13(a) compares the output of the dynamic estimator against ground truth in an xy plane, using the same test run used in Figures 5.11 and 5.12. A corresponding plot of the magnitude of estimation error in metres with respect to time is shown in Figure 5.13(b).



(a) Path display

(b) Estimation error

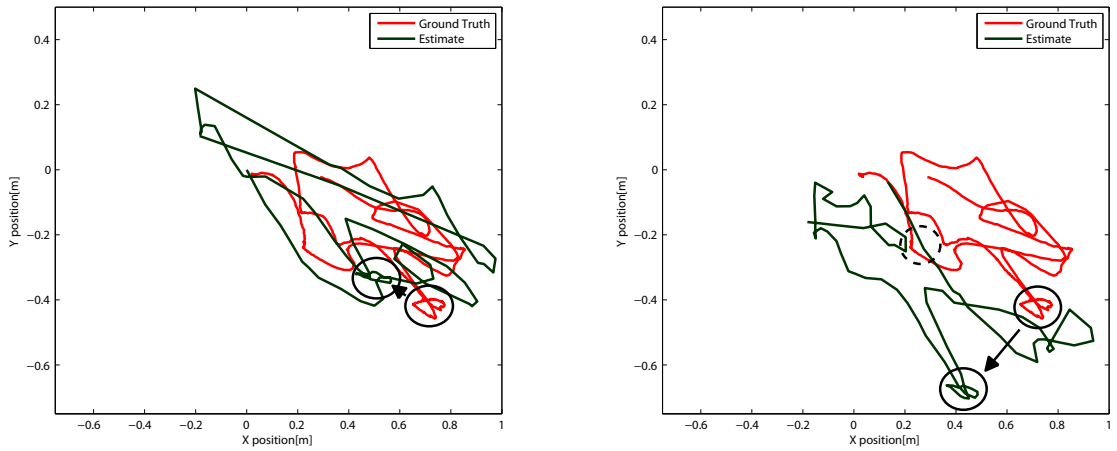
Figure 5.13: Dynamic motion model

The dynamic motion model has an worst-case total error in the above tests of 0.45 metres. In comparison, there is an error of 2.75 metres at the end of the baseline run. This latter error in the baseline also appears to have an increasing trend. Overall, the results are superior to those produced without the use of outlier rejection (Figure 5.11).

The need for an external positioning system as a reference for vehicle heading has also been examined. Figure 5.14 shows a comparison of results when the local positioning system provides vehicle heading continuously versus when the LPS simply initializes the heading estimate and allows the estimator to maintain the estimate over time.

Heading drift over time soon results in significant estimate drift, shown by the estimated path diverging down and to the left in Figure 5.14(b). This is to be expected, as a single slight angular error early on in estimation can result in a much more significant vehicle position error than the corresponding slight position error. The dashed circle in Figure 5.14(b) is the first example of this; the estimate continues towards the lower right, while the vehicle has begun moving more horizontally. This is further demonstrated by the two path sections surrounded by black circles. Though they are similar in shape, they have been

offset much more in Figure 5.14(b) than in Figure 5.14(a) due to accumulated angular error. From a practical standpoint, there are still workarounds to this deficiency when this system is to be operationally deployed. For instance, compasses can still provide a measurement of absolute heading in GPS–denied environments as long as magnetic interference is kept to a minimum.



(a) LPS as heading reference

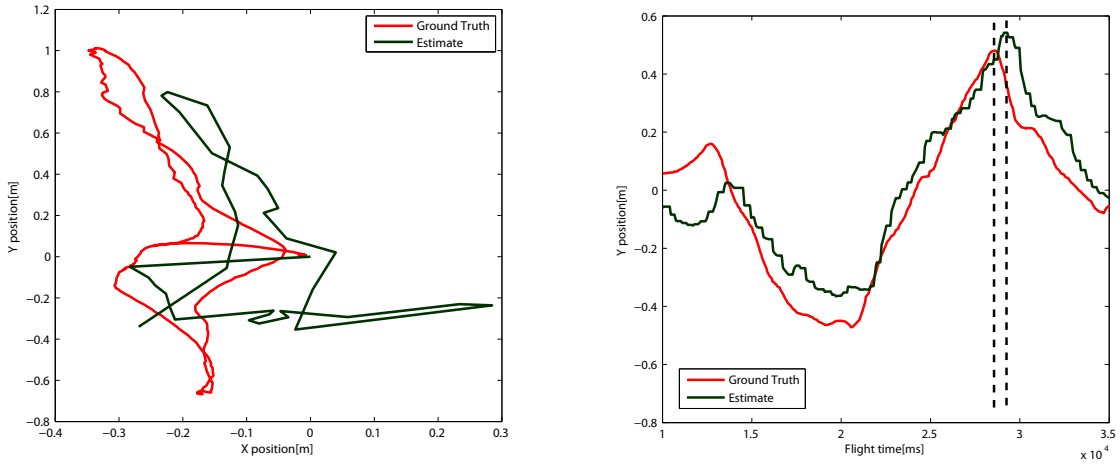
(b) No heading reference

Figure 5.14: Comparison of vehicle heading sources

The concern raised in Section 5.2 regarding the potential for lag is critical. The degree of the estimation lag has been experimentally determined by examining the Y position from the test shown in Figure 5.15(a). Plotting both estimated and ground truth positions over time shows an offset of around 900 ms between the maximum observed and maximum estimated motion along the Y axis. This has the potential to cause instability when using the estimate as the input to the vehicle's position controller.

5.4 Integrated Controls

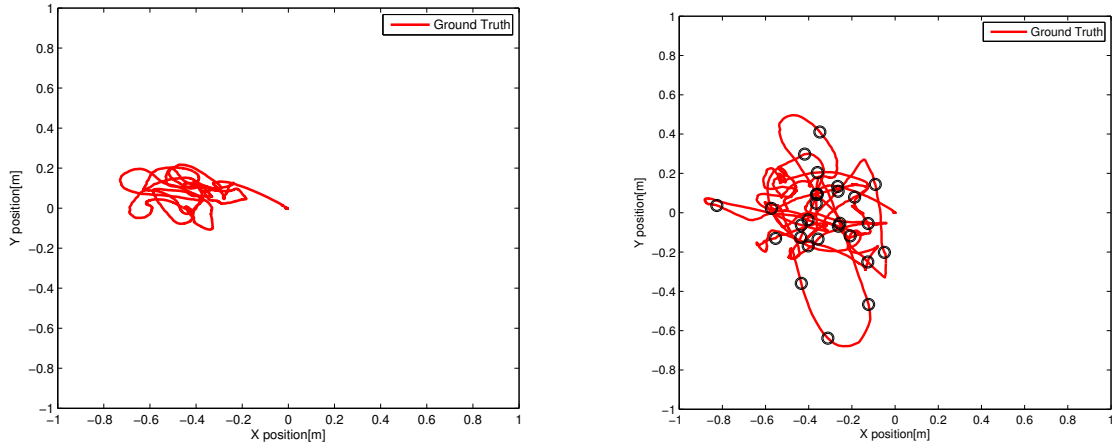
In the final set of tests, the onboard position controller takes position estimates directly from the algorithm described in this work. Figure 5.16 shows a comparison of the xy path of similar hover tests; one using the local positioning system for positioning and one using the visual state estimator.



(a) Vehicle path

(b) Vehicle Y position over time

Figure 5.15: Estimation lag



(a) Hover test – LPS

(b) Hover test – Visual Estimation

Figure 5.16: Vehicle control with visual state estimator

It is clearly apparent that the visual state estimation system is not yet suitable for controlling quadrotor UAVs in tight spaces. Though the system was able to hold position for 3-5 second durations, safety systems were needed to resync the state estimate on multiple occasions (sync locations shown by black circles) to compensate for system instability. These safety systems were added due to the restricted space available for test flights. With-

out safety systems, earlier experiments showed that the vehicle would be at risk of collision with the surroundings. It is hypothesized that the primary source of this behaviour is the estimation lag described in Section 5.2 and shown in Figure 5.15(b), with a secondary source being the general error in the position estimate. The local positioning system was reporting only 2 ms in image processing latency. Even with the slight additional lag due to the transmission and reception of this data, the degree of latency is still roughly two orders of magnitude less than that resulting from the use of the visual state estimator.

If the space available for test flights was larger, it would be possible to reduce the gains of the controller further to make it less sensitive to these errors. However, with nearly 1 second of lag in the system, a flight test area measuring approximately 2 x 3 metres, and the presence of unpredictable disturbances which require constant correction, this controller relaxation was not believed to be safe for the vehicle or its surroundings. The most effective way to improve the ability of this system to control the vehicle is likely to be by reducing system lag, which remains an area of future work.

Chapter 6

Conclusion and Recommendations

Quadrotor UAVs are well–suited for a wide variety of environments such as surveillance, mapping, search and rescue, infrastructure inspection, and disaster recovery. Many of the scenarios being proposed at the present time involve their operation in indoor or otherwise GPS–limited areas. Similarly, the unknown nature of these environments precludes the a priori installation of fixed infrastructure or reference points. Thus, new methods are necessary to avoid dependence on such technology or preconditioning.

Vision based methods are attractive, due to their low cost, low weight, and existing presence in vehicle sensor suites. Unfortunately, many of these environments are not ideal from a computer vision perspective; they may not have easily distinguishable features or may be dimly lit.

This thesis presents work which is intended to enable the use of onboard cameras for real–time estimation and control of quadrotors in such environments, without the need for extensive onboard processing or additional sensors. A position estimation algorithm has been developed, using a random–sample consensus approach in conjunction with an Extended Kalman Filter to improve the ability of the EKF to handle noisy input data, making it possible to reliably estimate vehicle motion using low quality images.. Past methods either add infrastructure or structure to the environment, or avoid the problem of noisy input data entirely. Other approaches assume predictable vehicle motion, or incur a great deal of additional computational requirements. The combination of RANSAC and the EKF was done in a way which decreased the time required to generate each estimate by a factor of 4.

A large number of simple features are used from each image with poor results being rejected by RANSAC, in contrast to the traditional approach of using a small number of features which are expensive to compute. Using simple features allows for the feature selection to take place over 10 times faster. Test infrastructure has been developed which

allows for online implementation of the algorithm on a commercial quadrotor UAV (the Aeryon Scout) using an off–the–shelf camera payload. The resulting algorithm has been integrated with a position controller, enabling its use as an estimation source for 2–D path following.

The algorithm is tested on physical hardware and found to be successfully able to estimate position of the vehicle to within 40 cm during indoor GPS–denied flights over featureless ground. The tests began with the validation of a number of assumptions about the problem. Initial results include confirmation that the noise in optical flow vectors is non–Gaussian, observations of outlier ratios changing rapidly over time, and verification that complex feature selection approaches do not provide significant added robustness. Then, a number of off– and online tests are performed to validate the algorithm, beginning from a baseline estimation which solely relies on an EKF for position estimation. Different vehicle models are used in the prediction step of the EKF to determine a suitable motion model. The algorithm is able to make the Kalman filter robust to outliers in optical flow, significantly improving on the baseline results. These results demonstrate the feasibility of using low–quality features for position estimation. The resulting position estimation algorithm is then deployed to serve as the source for a position controller for the same quadrotor.

Though the work has not been entirely successful in meeting the objective of fully controlled flight due to a control latency of nearly 1 second, avenues are apparent for future improvement.

First, it is clear that image capture frequency and latency is a major issue. To that end, it seems reasonable that this code is run fully onboard, when a computational platform is available which accommodates such development. At the moment, it is not recommended to deploy complex programs on the current processing hardware, due to its lightweight processor. It is recommended that additional processing power is added to allow for this deployment. Such an effort will also remove the current bottlenecks and delays the wireless network places on image transmission, allowing for a reduction in overall system latency, an increase in camera frame rate (leading to improved controls performance), and an increase in image resolution. If low–mass parallel processing hardware becomes available, it may also be feasible to investigate the reduction of RANSAC runtime by partitioning out the iterations to multiple cores.

Second, the assumption which constrains the camera to point directly downwards (Assumption 1.3) should be relaxed by generalizing the equations laid out in Section 2.1 and used in Section 2.3 to include roll and pitch information. This will allow the camera to point more towards the horizon, allowing for the acquisition of more information about its immediate surroundings.

Third, the use of RANSAC in combination with motion segmentation should be considered to allow for the construction of multiple models simultaneously. This capability would allow the system to be robust to the appearance of objects or walls in what has previously been assumed to be a strictly flat plane. This approach may also provide obstacle detection capabilities by considering models which do not describe horizontal planes as obstacles.

Fourth, investigation of other camera options may be beneficial. Though it is believed that processing the image onboard will provide drastic improvements in performance from the current setup through the reduction in latency and the combined increase in frame rate and resolution, additional improvements are possible. For instance, using a camera with a global shutter will mitigate image blur, while higher quality sensors will reduce image noise. As well, since no colour data is used in this work, using a greyscale camera will remove the processing required to convert the Bayer pixels to greyscale.

Finally, more rigourous experiments should be conducted in low–light conditions. Specifically, the use of near–IR cameras should be considered. Developing a system which can work with these cameras would be advantageous for commercial developers, as it would allow them to offer this technology on near–IR police and surveillance models of the vehicle.

With these modifications and extensions, it is highly probable that small aerial vehicles equipped with nothing but a camera, an IMU, and sufficient onboard processing will be able to successfully navigate in most environments, including those which are currently unsuitable due to the lack of identifiable features. This will allow vehicles to be deployed in a plethora of new flight regions, including in areas currently restricted by a lack of GPS reception. Ideally, these deployments will improve safety, comfort, and general quality of life for a wide range of people who are currently performing tasks which are unsafe, unenjoyable or unhealthy.

References

[1] N. Michael, J. Fink, and V. Kumar, "Cooperative manipulation and transportation with aerial robots," *Autonomous Robots*, vol. 30, pp. 73–86, 2011. 10.1007/s10514-010-9205-0. 1

[2] D. Mellinger, N. Michael, and V. Kumar, "Trajectory generation and control for precise aggressive maneuvers with quadrotors," in *Proceedings of the 12th International Symposium on Experimental Robotics*, (New Delhi, India), December 2010. 1, 6

[3] D. Mellinger, M. Shomin, and V. Kumar, "Control of quadrotors for robust perching and landing," in *Proceedings of International Powered Lift Conference*, (Philadelphia, PA), October 2010. 1

[4] Cropcam. `http://cropcam.com/`(current Jul 2011). 1

[5] Schiebel, "Camcopter S-100." `http://www.schiebel.net/Products/Unmanned-Air-Systems/CAMCOPTER-S-100/Introduction.aspx`(current Jul 2011). 1

[6] G. M. Hoffmann, S. L. Waslander, and C. J. Tomlin, "Quadrotor helicopter trajectory tracking control," in *Proceedings of the 2008 AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Honolulu, HI), August 2008. 1, 2, 6, 25

[7] Draganfly Innovations, Inc. `http://www.draganfly.com/`(current Jul 2011). 1

[8] Aeryon Labs, Inc. `http://www.aeryon.com/`(current Jul 2011). 1, 2, 4, 36

[9] microdrones GmbH. `http://www.microdrones.com/`(current Jul 2011). 1

[10] S. G. Ahrens, "Vision-based guidance and control of a hovering vehicle in unknown environments," Master's thesis, MIT, May 2008. 2

[11] P. Misra and P. Enge, *Global Positioning System — Signals, Measurement, and Performance*. Lincoln, MA: Ganga-Jamuna Press, 2006. 3, 6

[12] J. Farrell and M. Barth, *The Global Positioning System & Inertial Navigation*. McGraw Hill, 1998. 3

[13] N. Metni, J.-M. Pflimlin, T. Hamel, and P. Soueres, "Attitude and gyro bias estimation for a VTOL UAV," *Control Engineering Practice*, vol. 14, no. 12, pp. 1511 – 1520, 2006. 3, 24

[14] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*. Cambridge, MA: MIT Press, 2005. 3, 9, 25, 44

[15] C. Harris and M. Stephens, "A combined corner and edge detection," in *Proceedings of The Fourth Alvey Vision Conference*, (Manchester, UK), 1988. 3, 7

[16] J. Shi and C. Tomasi, "Good features to track," in *Proceedings of the 1994 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Seattle, WA), June 1994. 3, 7, 16, 41, 44

[17] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," in *Proceedings of the 9th European Conference on Computer Vision*, (Graz Austria), May 2006. 3, 7, 41

[18] M. Brown and D. Lowe, "Invariant features from interest point groups," in *Proceedings of the 2002 British Machine Vision Conference*, (Cardifff, Wales), pp. 656–665, September 2002. 3, 7

[19] R. Siegwart and I. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, MA: MIT Press, 2004. 4

[20] NovAtel, "OEMV Receivers." `http://www.novatel.com/products/gnss-receivers/oem-receiver-boards/oemv-receivers/`(current Jul 2011). 6

[21] R. He, S. Prentice, and N. Roy, "Planning in information space for a quadrotor helicopter in a GPS-denied environment," in *Proceedings of the 2008 IEEE International Conference on Robotics & Automation*, (Pasadena, CA), May 2008. 6

[22] Kinect Hacks. `http://www.kinecthacks.com/`(current Jul 2011). 6

[23] O. Bourquardez, R. Mahony, N. Guenard, F. Chaumette, T. Hamel, and L. Eck, "Image-based visual servo control of the translation kinematics of a quadrotor aerial vehicle," *IEEE Transactions on Robotics*, vol. 25, pp. 743–749, June 2009. 6, 8

[24] M. Tribou, "Recovering scale in relative pose and target model estimation using monocular vision," Master's thesis, University of Waterloo, 2009. 6

[25] N. Michael, D. Mellinger, Q. Lindsey, and V. Kumar, "The GRASP multiple micro-UAV testbed," *Robotics Automation Magazine, IEEE*, vol. 17, pp. 56 –65, September 2010. 6

[26] E. Altug, J. P. Ostrowski, and R. Mahony, "Control of a quadrotor helicopter using visual feedback," in *Proceeedings of the 2002 IEEE International Conference on Robotics & Automation*, (Washington, DC), May 2002. 6

[27] NaturalPoint, Inc. `http://www.naturalpoint.com/optitrack/`(current August 2011). 6, 37, 38

[28] S. Lupashin, A. Schollig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the 2010 IEEE International Conference on Robotics & Automation*, (Anchorage, AK), May 2010. 6

[29] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proceedings of the 7th International Joint Conference on Artificial Intelligence*, (Vancouver, B.C.), pp. 674–679, April 1981. 7, 16

[30] F. Dellaert, S. Seitz, C. Thorpe, and S. Thrun, "Structure from motion without correspondence," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, (Hilton Head Island, South Carolina), June 2000. 7

[31] V. S. Nalwa, *A Guided Tour of Computer Vision*. Addison-Wesley, 1993. 7

[32] B. Call, R. Beard, C. Taylor, and B. Barber, "Obstacle avoidance for unmanned air vehicles using image feature tracking," in *Proceedings of the 2006 AIAA Guidance, Navigation, and Control Conference and Exhibit*, (Keystone, CO), August 2006. 7

[33] P. Merrell, D. Lee, and R. Beard, "Obstacle avoidance for unmanned air vehicles using optical flow probability distributions," in *Proceedings of SPIE Mobile Robots XVII*, vol. 5609, (Philadelphia, PA), pp. 13–22, October 2004. 7

[34] J. Civera, O. Grasa, A. Davison, and J. Montiel, "1-point RANSAC for EKF filtering. Application to real-time structure from motion and visual odometry," *Journal of Field Robotics*, vol. 27, pp. 609–631, October 2010. 7, 10, 21, 23, 24

[35] B. Herisse, F.-X. Russotto, T. Hamel, and R. Mahony, "Hovering flight and vertical landing control of a VTOL unmanned aerial vehicle using optical flow," in *Proceedings of the 2008 IEEE International Conference on Robotics & Automation*, (Nice, France), September 2008. 8

[36] R. Mahony and T. Hamel, "Image-based visual servo control of aerial robotic systems using linear image features," *IEEE Transactions on Robotics*, vol. 21, pp. 227–239, April 2005. 8

[37] F. Kendoul, I. Fantoni, and G. Dherbomez, "Three nested Kalman filters-based algorithm for real-time estimation of optical flow, UAV motion and obstacles detection," in *Proceedings of the 2007 IEEE International Conference on Robotics & Automation*, (Roma, Italy), April 2007. 8

[38] H. Seydoux, M. Lefbure, F. Callou, C. Jonchery, and J.-B. Lanfrey, "Method of piloting a rotary-wing drone with automatic stablization of hovering flight," 2011. US Patent Application 12/865355. Assignee: Parrot. 8

[39] Parrot AR-Drone. `http://ardrone.parrot.com/parrot-ar-drone/usa/`(current August 2011). 9

[40] R. Gariepy and S. L. Waslander, "UAV motion estimation using low quality image features," in *Proceedings of the 2010 AIAA Guidance, Navigation and Control Conference and Exhibit*, (Toronto, ON), August 2010. 9

[41] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981. 9, 21

[42] A. Vedaldi, H. Jin, P. Favaro, and S. Soatto, "KALMANSAC: Robust filtering by consensus," in *Proceedings of the International Conference on Computer Vision (ICCV)*, (Beijing, China), pp. 633–640, October 2005. 9, 23, 29

[43] B. Kitt, A. Geiger, and H. Lategahn, "Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme," in *IEEE Intelligent Vehicles Symposium*, (San Diego, USA), June 2010. 9, 23

[44] D. Scaramuzza, F. Fraundorfer, and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," in *Proceedings of the 2009 IEEE International Conference on Robotics and Automation*, (Piscataway, NJ, USA), 2009. 9

[45] B. Williams, G. Klein, and I. Reid, "Real-time SLAM relocalisation," in *Proceedings of the International Conference on Computer Vision*, (Rio de Janeiro, Brazil), October 2007. 10, 30

[46] M. Chli and A. J. Davison, "Active matching for visual tracking," *Robotics and Autonomous Systems*, vol. 57, pp. 1173–1187, December 2009. 10, 17, 30

[47] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*. Sebastopol, CA: O'Reilly, 2008. 13

[48] C. Tomasi and T. Kanade, "Detection and tracking of point features," Tech. Rep. CMU-CS-91-132, Carnegie-Mellon University, April 1991. 15

[49] E. Rosten and T. Drummond, "Fusing points and lines for high performance tracking.," in *IEEE International Conference on Computer Vision*, vol. 2, pp. 1508–1511, October 2005. 16

[50] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European Conference on Computer Vision*, vol. 1, pp. 430–443, May 2006. 16

[51] J.-Y. Bouguet, "Pyramidal implementation of the Lucas Kanade feature tracker: Description of the algorithm," tech. rep., Intel Corporation, Microprocessor Research Labs, OpenCV Documents., 1999. 16, 39

[52] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press, 2nd ed., 2001. 21

[53] L. Clemente, A. Davison, I. Reid, J. Neira, and J. Tardos, "Mapping large loops with a single hand-held camera," in *Robotics: Science and Systems*, (Atlanta, Georgia), June 2007. 23

[54] S. Rao, *Applied Numerical Methods for Engineers and Scientists*. Prentice Hall, 4th ed., 2002. 31

[55] P. Chen, "3–d motion planning using kinodynamically feasible motion primitives in unknown environments," Master's thesis, University of Waterloo, August 2010. 34

[56] Robot Operating System. `http://www.ros.org/`(current Jul 2011). 38

[57] Microsoft, "Microsoft Robotics Developer Studio." `http://www.microsoft.com/robotics`(current Jul 2011). 38

[58] Player Project. `http://playerstage.sourceforge.net`(current Jul 2011). 38

[59] K. Birman and T. Joseph, "Exploiting virtual synchrony in distributed systems," in *Proceedings of the eleventh ACM Symposium on Operating systems principles*, SOSP '87, (New York, NY, USA), pp. 123–138, ACM, 1987. 38