Improving the Performance of Monocular Visual Simultaneous Localisation and Mapping through the use of a Gimballed Camera

Author:
Nicholas Playle

Supervisor:
Dr. Hugh H. T. Liu
Dr. Hamidreza Bolandhemmat

A thesis submitted in conformity with the requirements
for the degree of Master of Applied Science

in the

Graduate Department of Aerospace Science and Engineering
University of Toronto

Abstract

Improving the Performance of Monocular Visual Simultaneous Localisation and Mapping through the use of a Gimballed Camera

by Nicholas Playle

Master of Applied Science
Institute for Aerospace Studies
Graduate Department of Aerospace Science and Engineering
University of Toronto
2015

In this thesis modern vision based localisation methods are discussed and contrasted with existing satellite based approaches. Shortcomings are noted and potential solutions are highlighted. A novel method of using a gimballed camera to perform visual Simultaneous Localisation and Mapping (SLAM) is proposed, along with a control algorithm to point the camera toward feature dense regions. This method is then modularly coupled with existing visual SLAM techniques allowing seamless integration across different platforms. Ground tests are performed to verify operation of the gimbal controller and rotation inverser. Results from experimental flight tests are incorporated as a final means of obtaining information to verify gimbal operation.

Contents

List of Tables

List of Figures

Acronyms

CEP Circular Error Probable. 1, 76

CoG Centre of Gravity. 9, 18, 44

DGPS Differential Global Positioning System. 1

DoF Degree of Freedom. 19, 22, 41

EKF Extended Kalman Filter. 3, 5, 7, 10, 12, 19, 23, 24, 37, 39, 40, 68

ENU East North Up. 70

FoV Field of View. vi, 27, 63

GMSF Gimballed Multi-Sensor Fusion. 43

GNSS Global Navigation Satellite System. 1, 2, 5, 15, 17, 19, 22, 23, 43, 44, 69, 76, 77, 81, 82

GPS Global Positioning System. v, 1, 3, 5, 22–24, 39, 41, 70, 76, 78

IMU Inertial Measurement Unit. 4–7, 9, 14, 16–18, 21, 22, 24, 26, 31, 36, 39, 41, 42

INS Inertial Navigation System. 1–3, 9

KF Kalman Filter. 2, 9

LCKF Loosely Coupled Kalman Filter. vii, 69–71, 76, 82

MAV Micro Aerial Vehicle. 5, 19, 30

MCPTAM Multi-Camera Parallel Tracking and Mapping. 13, 14

MEMS Micro Electro Mechanical System. 1, 2, 4, 5, 19

MSF Multi-Sensor Fusion. ii, vii, 17, 18, 21–24, 26, 36, 38–44, 48, 55, 57, 63, 64, 68–71, 75–78, 82, 84

NED North East Down. 70

PIGA Pendulous Integrating Gyroscopic Accelerometer. 1

PTAM Parallel Tracking and Mapping. ii, iii, v, vi, 3, 10–15, 17, 22–24, 26, 27, 30, 37, 38, 40, 43–48, 50–53, 55, 57–61, 63, 64, 66, 69, 71, 76–78, 81, 82, 84

PTAMM Parallel Tracking and Multiple Mapping. 13

ROS Robot Operating System. 3, 14, 38, 39, 41, 42, 55, 78

SA Selective Availability. 1

SBC Single Board Computer. 41, 70, 78

SBI Small Blurry Image. vi, 46, 48, 49, 59, 60, 64, 68, 81

SLAM Simultaneous Localisation and Mapping. ii, v, 2, 3, 9–11, 13–19, 22–43, 47–50, 52, 55, 57, 62, 63, 79–85

SSF Single Sensor Fusion. 17, 23

UAS Unmanned Aerial System. 1, 2, 4, 5, 11, 13–15, 17, 24, 30, 31, 33, 35, 37, 38, 41, 56, 61, 72, 76, 82, 85

UAV Unmanned Aerial Vehicle. 14–17, 19

VSLAM Visual Simultaneous Localisation and Mapping. vii, 3, 57, 58, 62, 69, 71, 76, 77, 79–85

Symbols

$\mathbf{M}$	Matrix $\mathbf{M}$
$\mathbf{v}$	Vector $\mathbf{v}$
$\dot{\mathbf{v}}$	Time derivative of vector $\mathbf{v}$
$\mathbf{q}_a^b$	Unit quaternion rotation from frame a to frame b
$\mathbf{p}_i^c$	Displacement of frame c with respect to frame i expressed in frame i
b_ω	Gyroscope bias
b_a	Accelerometer bias
λ	Visual scale factor
$R_z(\psi)$	Rotation matrix about axis z by value ψ
$\otimes$	Quaternion multiplication
t	Time (seconds)

Reference Frames

i	IMU Frame
w	World (inertial) Frame
v	Vision Frame
c	Camera Frame
g	Gimbal Frame

Chapter 1

Introduction

1.1 Background and Motivation

The state estimation problem has long existed in the field of robotics. Early Inertial Navigation Systems (INSs) developed during World War II for V2 rockets used Pendulous Integrating Gyroscopic Accelerometers (PIGAs) and two freely gimballed gyroscopes to control trajectory and can be thought of as the first practical use of an INS [25]. The nature of the PIGA design meant that a velocity estimate was directly available from the accelerometer, compared with modern Micro Electro Mechanical System (MEMS) designs that require external computations. The position drift varied depending on the production run but was typically 1 in 100 or 1 in 1000, yielding theoretical accuracies of $600m$ to $6km$ over a 325 second flight. Ring laser gyroscopes invented during the 1960's further served to increase INS accuracy and today modern aircraft INSs are capable of drift rates of 370 metres per hour [1] (by comparison the most accurate V2 would be on the order of 7000 metres per hour).

With the invention of GPS in the late 1970's the possibility to eliminate position drift by resetting position to known GPS co-ordinates was realised, though GPS was not fully operational until 1995. The removal of Selective Availability (SA) in 2000 meant a highly accurate (typically 2.5m Circular Error Probable (CEP) with modern non-Differential Global Positioning System (DGPS) receivers) position solution was freely available. As a result nearly all modern Unmanned Aerial Systems (UASs) and autonomous ground robots have become dependent on a reliable Global Navigation Satellite System (GNSS) position fix being constantly available.

During the mid-1980s the Simultaneous Localisation and Mapping (SLAM) problem was formalised as a method of localising a robot while building a map of the surrounding world [10]. Initially these systems relied on laser scanners, were bulky and not well suited to aerial platforms. Advances in computing have led to algorithms capable of running in real time on computers capable of fitting inside a small UAS. More recently, vision based SLAM algorithms have become popular and increases in available computing power have allowed these algorithms to be applied to mobile applications.

The integration of MEMS technology with a absolute GNSS position measurement has become a standard for both airborne and terrestrial state estimation, with INS data being used for state propagation and GNSS position and velocity readings being used for measurements in a Kalman Filter (KF). While this combination leads to accurate state estimates it has also led to a heavy dependence on GNSS. A typical state-of-the-art MEMS INS will have significant drift, potentially 150m/minute [42], and even modern high end MEMS gyroscopes cannot meet tactical or navigation grades (defined as having a position drift on the order of $1km/hr$) due to poor bias drift and angle random walk characteristics [37]. As a result of this it is generally not possible to withstand GNSS signal loss for more than a minute or two as the position and velocity estimates of the aircraft will diverge, leading to a potential loss of vehicle.

As a result of this limitation there has been an increase in research in alternative position and velocity measurement sources. Airborne SLAM applications have been investigated since the late 1990's [2] and more recently by Hrabar in [15]. Fully autonomous flights have been demonstrated by Weiss et. al. in [38, 39, 40, 41], where a multirotor navigated succesfully using only vision-based position measurements.

1.2 Objective and Scope

The objective of this thesis is to demonstrate that existing visual SLAM algorithms may be adapted to use a gimballed camera and benefit from this modification through increased algorithm robustness. A modular approach will be taken to show that the gimballed camera concept can be applied to many existing algorithms rather than tightly coupled, both to reduce integration time and computational complexity.

The project will be divided into two distinct phases. The first, algorithm design and verification phase, will describe the gimbal control methodology in detail and outline the quaternion rotation

inverser required to achieve the desired system modularity. The second phase will focus on ground-based verification of the gimballed visual SLAM algorithm and examine the result data to show the gimballed system has a higher performance than fixed camera SLAM. The second stage will culminate in a flight test to verify the overall system performance in flight conditions.

The contributions of this thesis include the development of a scene scanning algorithm designed to point a gimballed camera to regions of high feature density, and a rotation inverser allowing the visual SLAM camera to have an arbitrary rotation with respect to the vehicle frame. The rotation inverser allows the visual SLAM algorithm to provide pose estimates with respect to a vehicle-fixed frame, despite the arbitrary rotation of the camera.

This thesis is completed as a collaborative project between the University of Toronto Institute for Aerospace Studies and MicroPilot Inc.. The experimental platform is provided by MicroPilot and a MicroPilot $MP2128^{HELI2}$ autopilot is used both for INS data capture and for comparative GPS only state estimates.

1.3 Thesis Outline

Chapter 2 provides an overview of state estimation via Extended Kalman Filter (EKF) and examines details specific to use with a gimballed visual SLAM system. The motivation for using a gimballed VSLAM system will be provided in Chapter 3, including difficulties surrounding feature detection, repetitive, and sparsely textured regions. The use of a gimballed camera and gimbal control algorithm will be presented, along with how the gimbal will affect position measurements. The rotation inverser will also be discussed as a means of providing modularity to the control framework. Finally implementation-specific details of using Robot Operating System (ROS) and Parallel Tracking and Mapping (PTAM) as a framework for experimentation will be detailed. Chapter 4 will provide experimental validation, beginning with ground-based indoor stationary tests and moving to outdoor ground and flight tests. Additionally the results of the tests will be discussed. Finally, Chapter 5 provides conclusions drawn from the presented work and identifies potential future research ideas.

Chapter 2

State Estimation Overview

2.1 State Estimation Via Kalman Filter

Control and navigation algorithms operate on vehicle position, velocity and orientation data which are not directly measurable by modern sensors. As such, multiple alternative sensor data are combined via *sensor fusion* to obtain an estimate of the current UAS states, typically position, velocity and orientation with respect to the world or inertial frame. Additionally, in modern filters, estimates about the sensors themselves (such as bias) are also estimated in an attempt to obtain the most accurate state estimate possible. Since the navigation algorithms will use the estimated states it is important to ensure these estimates are as close to true values as possible, otherwise performance will not be satisfactory.

2.1.1 The Need for Sensor Fusion

At the heart of nearly all modern small UASs is a MEMS based Inertial Measurement Unit (IMU), consisting of sensors capable of measuring acceleration and angular rate on all 3 axes. The IMU is thus capable of measuring all six degrees of freedom. Velocity and position may be estimated by the single and double integration of accelerometer data. Similarly, orientation may be estimated by integrating the measured angular rates. While the IMU is capable of measuring the aircraft state at high rates with high bandwidth (typical sensors may update between $100Hz - 2000Hz$), MEMS sensors are notoriously noisy and suffer from large bias drifts, causing a rapid unbounded position drift and making them unsuitable for unaided navigation. True aviation and marine grade gyroscopes and accelerometers (as defined in [12]) are capable

of unaided navigation, however their cost is several orders of magnitude higher than MEMS sensors and their size and weight make their use on lightweight UASs impossible.

A common practice is to equip UASs with auxiliary sensors capable of measuring absolute position which allows for the use of low cost MEMS sensors. Some form of sensor fusion algorithm is required to combine the high rate but lower quality IMU measurements with slower but non drifting global position measurements. Naive approaches to sensor fusion simply reset the estimated position to the measured GPS position at regular intervals, however in general it is desirable to use more sophisticated algorithms. Recent increases in available computational power have made it practical for Micro Aerial Vehicles (MAVs) to use more advanced sensor fusion algorithms, most commonly a form of Kalman filter, which (for linear system dynamics and white noise) provides the optimal state estimate. An alternative approach to state estimation are complementary filters (colloquially known as Direction Cosine Matrix or DCM Filters) [22] which high pass gyroscope data and low pass accelerometer data as an attempt to remove gyro bias. The Ardupilot autopilot system uses a DCM for its computational efficiency, however a drawback is that no position or velocity estimates are generated, and thus those measurements come solely from GNSS at a much slower rate (typically 5Hz). Particle filters have also been used for state estimation with moderate success [4], particularly due to their ability to handle non-Gaussian MEMS noise characteristics. however they have not received widespread attention due to their relatively high computational complexity and implementation difficulty. In general, most modern aircraft state estimation schemes use a Kalman filter, with more recent implementations relying on EKFs or Unscented Kalman Filters (UKFs).

2.1.2 The Kalman Filter

A Kalman filter is a recursive estimator which uses noisy input data from multiple sources to obtain an optimal estimation of the system outputs. In the case where the system is linear and disturbed by white noise, the Kalman filter will be optimal (in the sense that the mean square error and variance of the estimates will be minimised), however the non-linear nature of the navigation equations will force the use of a linearised system model which will render the estimate non-optimal. The extended Kalman filter can be used to minimise this error by linearising the system dynamics around the current state and measurement noise at the expense of increased computational complexity.

The Kalman filter operates in state space on a system represented in Equation (2.1).

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{F}_k \boldsymbol{x}_{k-1} + \boldsymbol{B}_k \boldsymbol{u}_k + \boldsymbol{w}_k \\ \boldsymbol{z}_k &= \boldsymbol{H}_k \boldsymbol{x}_k + \boldsymbol{v}_k \end{aligned} \tag{2.1}$$

where $\boldsymbol{x}_k$ is the state vector at time k, $\boldsymbol{F}_k$ is the state transition matrix, $\boldsymbol{u}_k$ is the system input, $\boldsymbol{B}_k$ is the input coupling matrix and $\boldsymbol{w}_k$ is the process noise, which is assumed to be zero mean and Gaussian given by $\boldsymbol{w}_k = N(0, \boldsymbol{Q}_k)$, where $\boldsymbol{Q}_k$ is the process noise covariance. Similarly, for the output measurement $\boldsymbol{z}_k$, we have $\boldsymbol{H}_k$ is the measurement matrix and $\boldsymbol{v}_k$ is the observation noise, again assumed to be zero mean and Gaussian, given by $\boldsymbol{v}_k = N(0, \boldsymbol{R}_k)$ where $\boldsymbol{R}_k$ is the observation noise covariance.

In the case where the system dynamics are linear simply the well known Kalman filter equations may be used as an optimal estimator of system states. Equations (2.2) and (2.3) show the state and state covariance (denoted $\boldsymbol{P}_k$) propagation equations, respectively. These operations are carried out at the prediction rate, which in this specific case will be the IMU update rate.

$$\hat{\boldsymbol{x}}_k^- = \boldsymbol{F}_k \hat{\boldsymbol{x}}_{k-1}^+ + \boldsymbol{B}_k \boldsymbol{u}_k \tag{2.2}$$

$$\boldsymbol{P}_k^- = \boldsymbol{F}_k \boldsymbol{P}_{k-1}^+ \boldsymbol{F}_k^T + \boldsymbol{Q}_k \tag{2.3}$$

The measurement or correction equations are carried out at the measurement sensor rate, and are shown in Equations (2.4) to (2.7). Equation (2.4) calculates the *residual* $\tilde{\boldsymbol{z}}_k$ from the current state estimate and measurement matrix, which is the error between the state estimate based on IMU data and the measured states from auxiliary sensors. The *innovation covariance* $\boldsymbol{S}_k$ is then calculated in Equation (2.5), which is then used to calculate the Kalman gain $\boldsymbol{K_k}$ in Equation (2.6). Finally in Equation (2.7) the *a priori* state estimate $\hat{\boldsymbol{x}}_k^-$ is corrected to be the *a posteriori* state estimate $\hat{\boldsymbol{x}}_k^+$, which is the optimally estimated system state.

$$\tilde{\boldsymbol{z}}_k = \boldsymbol{z}_k - \boldsymbol{H}_k \hat{\boldsymbol{x}}_k^- \tag{2.4}$$

$$\boldsymbol{S}_k = \boldsymbol{H}_k \boldsymbol{P}_k^- \boldsymbol{H}_k^T + \boldsymbol{R}_k \tag{2.5}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_k^- \boldsymbol{H}_k \boldsymbol{S}_k^{-1} \tag{2.6}$$

$$\hat{\boldsymbol{x}}_k^+ = \hat{\boldsymbol{x}}_k^- + \boldsymbol{K_k} \tilde{\boldsymbol{z}}_k \tag{2.7}$$

$$\boldsymbol{P}_k^+ = (\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{H}_k) \boldsymbol{P}_k^- \tag{2.8}$$

The Kalman filter state estimates and covariance must first be initialised to some initial conditions which should represent a rough estimate of the current system state, i.e. $\hat{\boldsymbol{x}}_0 = E\langle \boldsymbol{x}_0 \rangle$ and $\boldsymbol{P}_0 = E\langle \boldsymbol{x}_0 \boldsymbol{x}_0^T \rangle$ where $E\langle \bullet \rangle$ is the expected value of the estimated variable. The filter then operates recursively on the input data stream from the IMU and performs measurement updates when available from auxiliary sensors.

As previously mentioned, the Kalman filter as described above is the optimal estimator for linear systems. As we will later see in Section 2.1.3, the inertial navigation equations yield a nonlinear system, shown in Equation (2.9).

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u}, \mathbf{w}, \mathbf{t}) \\ \mathbf{y} &= \mathbf{h}(\mathbf{x}, \mathbf{v}, \mathbf{t}) \end{aligned} \tag{2.9}$$

The most straightforward method to combat this issue is simply to linearise the system around some known trajectory and apply the standard Kalman filter. While this works well if the system trajectory is known in advance, most often this is not the case. An alternative solution is to linearise the system at each time step around the current state estimate. While this does not provide an optimal solution (nor even one that is guaranteed to converge), in practice this method is found to have satisfactory results. The resultant algorithm is called the EKF and is shown in Algorithm 1.

Note that this algorithm will be marginally more computationally expensive than the standard Kalman filter algorithm due to the linearisation step; however, for highly non-linear systems the increase in accuracy is well worth the extra computation time. Further algorithm modifications may be implemented to reduce computational complexity if required, but for brevity they will not be discussed here.

2.1.3 The Kalman Filter as Applied to Unmanned Aircraft

When the Kalman filter is based around an inertial model, the inputs to the system are considered to be the measured accelerations and angular rates from the IMU. A simple sensor model where the real accelerations and angular rates $\mathbf{a}$ and $\boldsymbol{\omega}$, respectively, consist of the measurement (denoted with a subscript m), a bias b and Gaussian white noise n, shown in Equation (2.10).

$$\mathbf{a} = \mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a \qquad \boldsymbol{\omega} = \boldsymbol{\omega}_m - \mathbf{b}_\omega - \boldsymbol{n}_\omega \tag{2.10}$$

Algorithm 1 Extended Kalman Filter Algorithm

1. Linearise System Dynamics

$$\mathbf{F}_k = \left.\frac{\partial \mathbf{f}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k^-}$$

2. Time Update

$$\begin{aligned}\hat{\mathbf{x}}_k^- &= f(\hat{\mathbf{x}}_{k-1}^+, \mathbf{u}_{k-1}) \\ \mathbf{P}_k^- &= \mathbf{F}_k \mathbf{P}_{k-1}^+ \mathbf{F}_k^T + \mathbf{Q}_k\end{aligned}$$

3. Linearise Measurement

$$\mathbf{H}_k = \left.\frac{\partial \mathbf{h}(\mathbf{x},\mathbf{u})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k^-}$$

4. Measurement Update

$$\begin{aligned}\mathbf{K}_k &= \mathbf{P}_k^- \mathbf{H}_k^T \left(\mathbf{R}_k + \mathbf{H}_k \mathbf{P}_k^- \mathbf{H}_k^T\right)^{-1} \\ \tilde{\mathbf{z}}_k &= \mathbf{z}_k - \mathbf{H}_k \hat{\mathbf{x}}_k^- \\ \hat{\mathbf{x}}_k^+ &= \hat{\mathbf{x}}_k^- + \mathbf{K_k} \tilde{\mathbf{z}}_k \\ \mathbf{P}_k^+ &= \left[\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right] \mathbf{P}_k^-\end{aligned}$$

The bias of each sensor is modelled as a non-static random process shown in Equation (2.11)

$$\dot{\mathbf{b}_a} = \mathbf{n}_{b_a} \qquad \dot{\mathbf{b}}_\omega = \mathbf{n}_{b_\omega} \tag{2.11}$$

For the purposes of aircraft state estimation, the states are typically described in Equation (2.12), which are defined in Table 2.1.

$$\mathbf{x} = \begin{pmatrix}\mathbf{p}_w^i & \mathbf{v}_w^i & \mathbf{q}_w^i & \mathbf{b}_w & \mathbf{b}_a\end{pmatrix}^T \tag{2.12}$$

It is thus somewhat intuitive to derive the differential equations which govern the state propa-

State	Meaning
$\mathbf{p}_w^i$	position of the IMU wrt. world
$\mathbf{v}_w^i$	velocity of IMU wrt. world
$\mathbf{q}_w^i$	quaternion orientation of the IMU wrt. world
$\mathbf{b}_\omega$	gyro bias estimate
$\mathbf{b}_a$	accelerometer bias estimate

TABLE 2.1: Basic states in a Kalman filter used in a UAS

gation, shown in Equations (2.13) to (2.17).

$$\dot{\mathbf{p}}_w^i = \mathbf{v}_w^i \tag{2.13}$$

$$\dot{\mathbf{v}}_w^i = \mathbf{R}_{(\mathbf{q}_w^i)}^T(\mathbf{a}_m - \mathbf{b}_a) + \mathbf{g} \tag{2.14}$$

$$\dot{\mathbf{q}}_i^w = \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m - \mathbf{b}_\omega)\mathbf{q}_i^w \tag{2.15}$$

$$\dot{\mathbf{b}}_\omega = \mathbf{n}_{b_\omega} \tag{2.16}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{b_a} \tag{2.17}$$

Where $\mathbf{g}$ is gravity, $\mathbf{R}_{(\mathbf{q}_w^i)}^T$ is the rotation matrix corresponding to the quaternion $\mathbf{q}_w^i$ and $\boldsymbol{\Omega}(\boldsymbol{\omega})$ is the quaternion kinematical matrix of $\boldsymbol{\omega}$, shown in Equation 2.18.

$$\boldsymbol{\Omega}(\boldsymbol{\omega}) = \begin{bmatrix} 0 & \omega_z & -\omega_y & \omega_x \\ -\omega_z & 0 & \omega_x & \omega_y \\ \omega_y & -\omega_x & 0 & \omega_z \\ -\omega_x & -\omega_y & -\omega_z & 0 \end{bmatrix} \tag{2.18}$$

It is important to note the differential equations listed for the velocity prediction do not take into account any centripetal acceleration terms. This implies that the IMU is located precisely on the aircraft Centre of Gravity (CoG). If it is not, centripetal acceleration will be incorrectly measured as linear acceleration which will ultimately lead to errors in the estimated states. If the centripetal contribution is small or short in duration however the overall effects should be negligible. Most commonly Kalman filters for inertial navigation are of the indirect or error state form, allowing separation of the INS and KF portions of the filter. Stability advantages also arise from this formulation as well since the INS is typically able to follow higher frequency vehicle dynamics than the KF, but perhaps the largest advantage is the fact vehicle dynamics do not have to accurately be modelled [30]. Additional details may be found in [12].

2.2 Simultaneous Localisation and Mapping

2.2.1 A Brief History of the SLAM Problem

The idea of SLAM was first detailed in the late 1980's, where the problem was formalised to be probabilistic in nature by Smith in [35] and Durrant-Whyte in [9], which dealt with the problem of simultaneously constructing a navigable map of the visible world and localising

oneself within the constructed map. A milestone in SLAM was published by Smith et. al. in [34] where it was formally shown that in order for a SLAM algorithm to find a consistent solution the estimator would need to include both the robot pose and also information on each observed feature. This implies (for filter-based SLAM) a state vector approximately the size of the number of tracked points which must be updated at every map iteration. Indeed, early SLAM algorithms exhibited quadratic growth due to the use of extended Kalman filters to estimate vehicle state, limiting their scope to applications with an abundance of computational power. In 2002, Montemerlo et. al. introduced the FastSLAM algorithm in [23]. FastSLAM differed from most previous SLAM approaches by using a particle filter in place of the more traditional extended Kalman filter, allowing for a dramatic reduction in computational complexity. EKF based SLAM approaches grow quadratically as $O(k^2)$ for k map points, whereas FastSLAM grows as $O(m \log k)$ for m particles and k map points, effectively reducing the complexity to logarithmic growth. FastSLAM 2.0 was published by Montemerlo et. al. in [24] as an extension to FastSLAM which is more accurate and stable with only one particle. FastSLAM and FastSLAM 2.0 have been successfully used in real-world applications, including Stanford's Stanley autonomous vehicle.

The problem of visual SLAM was also investigated in the late 1980's by Ayache and Faugeras in [3], where the authors applied EKF-based SLAM techniques to distance measurements taken from stereo vision cameras. Due to computational requirements, this algorithm was not able to operate in real-time, and it would be many years before real-time visual SLAM became possible. Research instead focused on visual tracking of previously known environments, off-line structure-from-motion and optical flow algorithms. The first practical visual SLAM framework was demonstrated by Davison in [6] and was achieved by using an EKF based approach and carefully managing map features so that the computations were able to remain real-time. In 2007 Davison et. al. published the MonoSLAM algorithm in [7], designed to provide 3D position from a single camera. Careful map management was once again key to keeping real-time performance of their algorithm which was demonstrated to work with up to 100 tracked features and work in room sized environments. Parallel Tracking and Mapping (PTAM) was developed by Klein and Murray in [19] and was fundamentally different from previous visual SLAM approaches due to the separation of the mapping and tracking tasks. This allowed real-time performance coupled with large map sizes, and variants of PTAM have been used on both ground and air based robots.

2.2.2 Parallel Tracking and Mapping

PTAM is a SLAM variant published by Klein and Murray in 2006 in [19]. PTAM differs from other SLAM algorithms in the fact that it separates the problem in to two separate tasks: a *tracking* thread and a *mapping* thread. PTAM was originally developed for virtual augmented reality games but found a wide following in the UAS community due to the algorithm's performance. PTAM is *key-frame* based in nature, that is, new tracked features are only added to the global map when deemed necessary by the map making thread. This is in contrast to filter-based SLAM where the map (and thus all map features) are updated at each frame. The fact that the map is not updated every frame not only reduces the computational requirements but also means that PTAM is inherently stable when stationary. Filter-based SLAM algorithms may become unstable if they observe the same scene for extended periods of time due to error accumulation in the covariance matrix. Since PTAM only updates the map when sufficiently many new features are observable it is inherently more stable. Furthermore, PTAM does not update the map if the current map tracking is deemed to be poor quality, thereby preventing erroneous measurements from corrupting the existing map [19]

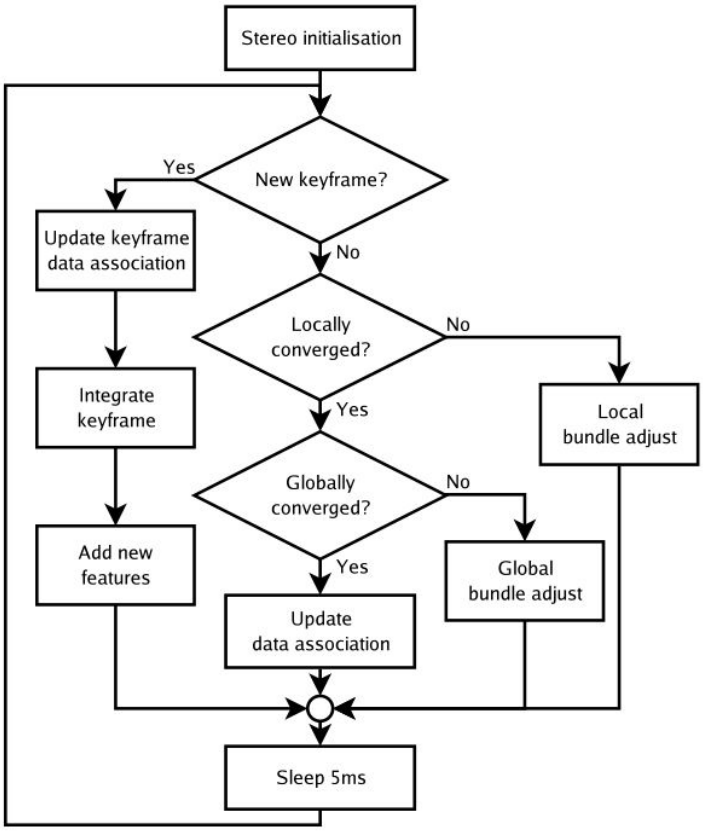


FIGURE 2.1: PTAM Mapping Thread Flow Diagram [19]

The parallel nature of PTAM allows the tracking and mapping tasks to be separated into threads that may be executed in parallel. Specifically, in PTAM the tracking thread runs at the camera frame rate and is responsible for running a feature recognition algorithm (typically a FAST corner detector, but this may be replaced by any suitable algorithm) to extract tracked points and then correlating the observed features with the camera position and orientation. Perhaps the largest advantage PTAM has over EKF-based algorithms is the fact that the tracking task is (for all intents and purposes) capable of running in a constant time regardless of map size. This implies that one could have an arbitrarily large map and maintain pose updates at the camera frame rate. This assumption is generally correct; however, the limiting factor becomes the rate at which new regions can be explored. The *mapping* thread of PTAM is responsible for generating new portions of the map. To do so, it employs a technique known as *bundle adjustment* to optimally add new points to the map and to minimise error in existing map points. In general, the bundle adjustment algorithm scales as $O(m^3 + mn)$ for m keyframes and n map points. A comparison of normalised time complexities is shown in Figure 2.2. Note that

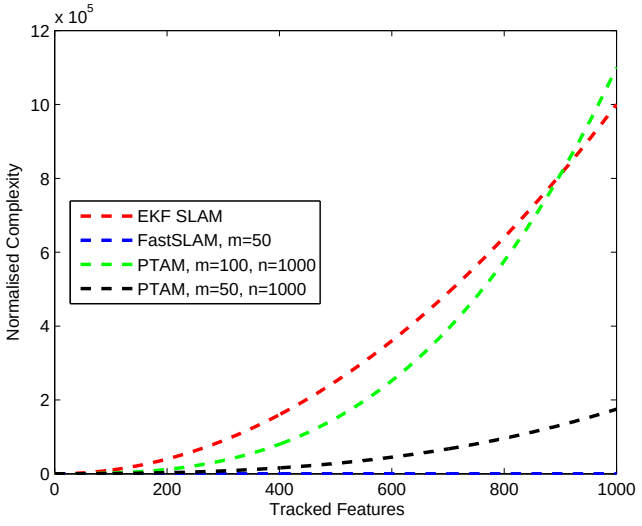


FIGURE 2.2: Complexity growth of various SLAM algorithms

through careful management of keyframes it is possible to ensure the complexity of PTAM does not exceed that of EKF-based SLAM. While it would appear that FastSLAM is more efficient than PTAM, in practice FastSLAM operates with orders of magnitude more map points and practical vision based implementations are limited to frame rates of approximately 2Hz [33].

PTAM uses a bundle adjusment algorithm to optimise the location of map points during the addition of a new keyframe to the existing map. Bundle adjustment in general solves a nonlinear least squares problem using a weighted cost function. While bundle adjustment is an optimal

solution to the optimisation of map points with Gaussian measurement error it is often observed that map point error is more widely distributed, leading to large errors introduced by outliers in the standard least squares solution. As such, PTAM employs a Tukey biweight M-estimator to reduce the effect of outliers and maintain a robust map estimate. As shown by Strasdat et. al. in [36] there are very few scenarios where filter based SLAM is able to outperform a bundle adjustment based approach, and during the test performed bundle adjustment outperforms filter based approaches per unit of computing time, implying that it is better suited for computation constrained scenarios (such as small UASs).

To overcome the map scalability issue Parallel Tracking and Multiple Mapping (PTAMM) was developed by Castle and Murray in [5] which extended PTAM to allow online switching between maps. By imposing a keyframe limit on map size the maximum computational complexity of the algorithm remains bounded while the ability to switch between maps allows PTAMM to retain a full map of explored territory. A limitation of this work is that the sub maps can expand over one another which can creates redundant map points and leads to inconsistent pose estimates.

2.2.2.1 Multi-Camera Parallel Tracking and Mapping

In [13] Harmat et. al. demonstrate Multi-Camera Parallel Tracking and Mapping (MCPTAM) which fuses data from multiple cameras to provide visual pose estimates. In this work the camera model was also modified to the Taylor omnidirectional camera model in order to accommodate cameras with greater than 180° field of view. The fields of view of the cameras may or may not overlap, however if overlap exists between cameras then the algorithm is capable of recovering the absolute metric scale, something which is not observable with any monocular based visual SLAM system.

MCPTAM is further expanded in [14] where a total of four cameras are used, two in a downward looking configuration and two in a forward looking configuration. In general MCPTAM requires an in depth inter-camera calibration procedure (when compared with vanilla PTAM) in order to obtain the camera-to-camera pose estimates. While Harmat et. al. point out that there is not a significant performance degeneration when using multiple cameras since each camera is estimating the same pose, it is noted that the performance shown in this paper is generally lower than others, possibly due to the more advanced camera model and inter-camera transforms which must be accounted for at each frame step. Using the on board computer an update rate

of 7 FPS was achieved with two cameras, however this dropped to 2 FPS when using 4 cameras (approximately a 50% performance hit).

During the extensive flight tests performed in this paper MCPTAM was run on a ground based laptop and not processed on the UAS itself. It was shown that by utilising two forward-looking cameras in conjunction with the two downward looking cameras that MCPTAM was able to maintain accurate position estimates when hovering near the ground. This is an important development since downward facing SLAM systems often have difficulties accomplishing this task due to the high apparent feature velocities seen by downward looking cameras when close to the ground (when close to an observed feature a $10cm$ translation may be $20 - 30$ pixels, compared to $2 - 3$ pixels if observed at altitude).

While there is an obvious advantage to using multiple cameras it is not generally possible to navigate with a 2-4Hz update rate, and currently no on board navigation solution has been demonstrated, and small aircraft may not be able to carry the additional cameras due to space or weight constraints. Additionally, MCPTAM does not currently employ a map size limiting approach and has only been demonstrated in a relatively small regions with minimal exploration of new regions. Since there is no upper bound on map size it is possible that PTAM would not be able to function sufficiently fast for navigation purposes when exploring new environments.

2.2.2.2 ETHZ Parallel Tracking and Mapping

ETHZ PTAM is another modified version of Klein's original PTAM. Developed by researchers at ETH Zurich, this version of PTAM has been modified to address uses in Unmanned Aerial Vehicles (UAVs). Specifically, it has been modified to run on computationally limited platforms typically found in UAVs while remaining robust to self similar textures such as grass and pavement, described in [41]. The algorithm was also modified to gracefully handle map loss and to automatically re-initialise based on last known position. ETHZ PTAM also publishes pose covariance estimates crucial to the integration with a Kalman filter framework. Further extensions have allowed ETHZ PTAM's motion model to be fused with real-time IMU data rather than the exponentially decaying motion model used in traditional PTAM. Finally, ETHZ PTAM was integrated into the Robot Operating System (ROS) framework, allowing it to be easily ported across multiple platforms. Perhaps the most important modification is the ability to limit the number of map keyframes. As previously shown in Figure 2.2 the number of

keyframes dramatically increases PTAM computational complexity due to the bundle adjustment algorithm. In order to limit computational complexity it is possible to limit the number of keyframes in a map. The strategy followed by ETHZ PTAM is to add keyframes up to some arbitrary limit k and then begin deleting keyframes that are furthest from the current position estimate. Complexity may also be reduced by placing a limit on the number of tracked features but due to the $O(m^3 + mn)$ growth of PTAM there is less benefit to reducing tracked features (linear growth) compared to keyframes (cubic growth). This allows the computational burden of PTAM to be tailored to the individual computer running the algorithm. The removal of map keyframes introduces the potential for large visual frame drift since over a large scene the behaviour of PTAM will be optical flow-like rather than a true SLAM algorithm, but it puts a hard bound on computation time for the SLAM algorithm. The drift is largely due to the fact that PTAM is deleting map points and there is no guarantee the same points will be tracked if the UAS flies over the same portion of the scene again. Over a large travelled distance (e.g. if the UAS is flying a large circular path) this error will accumulate as a drift in the position. The drift will be present on all 3 axes, not just the direction of travel, so it is important to account for this if using ETHZ PTAM for navigation. As will be discussed in Section 2.3.2 it is possible to use GNSS as a global position reference to completely eliminate the drift from the Kalman filter (visual SLAM drift is unavoidable). The drift will also be spatial in nature as PTAM will delete the keyframe data furthest away from its current position if a new keyframe is to be added to the map. Thus, if a vehicle is hovering or stationary no keyframes will be deleted and there will be no vision frame drift. This also allows PTAM to run in constant time (the time taken to run the corner detector and point tracking algorithm on a given frame) while the UAV is stationary, a large benefit for hovering vehicles. Additionally, since the map is not updated while stationary there is no risk of map corruption due to continually increasing covariance, a common problem Kalman filter based SLAM algorithms which update the map every frame. Due to these features ETHZ PTAM was chosen as the framework on which to base further research.

2.2.3 Monocular and Stereovision Systems

It is well known that absolute scale is observable on stereo camera systems by means of triangulation. Many visual SLAM algorithms have relied upon stereo vision for depth information, from the very beginning of visual SLAM research shown in [3] to more modern particle filter based approaches seen in [11]. It is thus worthwhile to ascertain the benefit of stereo vision

based approaches in comparison to monocular based approaches. A key paper by Lemaire et. al. [20] compares a filter based stereo vision SLAM algorithm with an equivalent monocular based algorithm, with experimental results from a ground based vehicle and an aerial platform. The stereo vision algorithm showed centimetre level accuracy on the ground vehicle, compared to the monocular system which showed error in the $10cm - 20cm$ range. However, when used on the aerial platform (a blimp with a 2.1m stereo baseline) both systems show comparable error, in the $1m - 10m$ range. At first these results may appear counter-intuitive, however upon closer inspection the reason for the disparity is clear. Practical stereo vision systems have limited camera resolution (typically 640×480 pixels), and thus the error in depth information will vary with distance, as shown in Figure 2.3. It is clear that stereo vision systems offer very precise

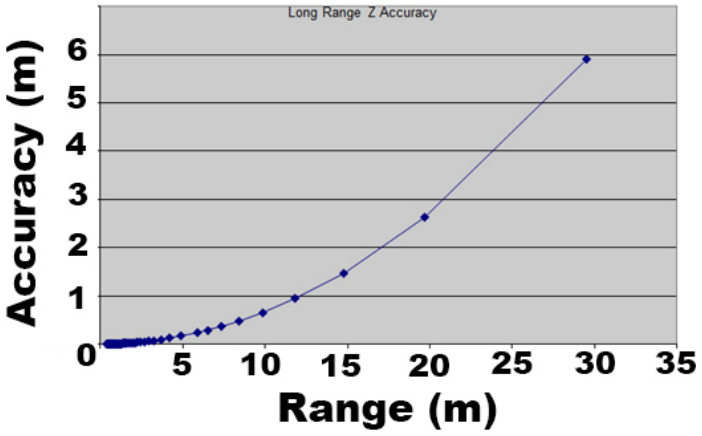


FIGURE 2.3: Depth accuracy of a Point Grey Research Bumblebee2 Stereovision Camera [28]

depth resolution (typically $1mm$ at $2m$) however at the higher altitudes UAVs typically fly the depth resolution is marginal at best. Coupled with the additional weight and processing power required to operate stereo vision cameras, the costs often outweigh the benefits. Furthermore, while it is true that standalone monocular SLAM systems are unable to observe absolute scale, by utilising the IMU present on nearly all unmanned aircraft it is possible to recover global scale, as shown by Nutzi et. al. in [27]. For these reasons, only monocular vision based SLAM systems are considered for the purposes of this research.

2.3 Kalman Filter Framework

The basic framework of an extended Kalman filter was described in Section 2.1.2 with application specific details presented in Section 2.1.3. This section will further expand on the Kalman filter

as applied to UAVs and elaborate on the various co-ordinate frames and measurement update sensors used on the UAV itself.

2.3.1 Overview of ETH Zurich Sensor Fusion

Prior work into vision based SLAM led researchers at ETH Zurich to develop two sensor fusion algorithms which are uniquely suited to processing visual SLAM data, first presented by Weiss et. al. in [38, 40]. The sensor fusion algorithms, called Single Sensor Fusion (SSF) and Multi-Sensor Fusion (MSF) perform sensor fusion from one or multiple sensor sources. Single Sensor Fusion (SSF) is based around the assumption that the sole measurement sensor is a visual SLAM algorithm while MSF assumes there is a core visual SLAM algorithm which is augmented by additional sensor sources (such as GNSS). The core of both filters are time delay compensated iterated extended Kalman filters. SSF was developed by Weiss during Ph.D research at ETH Zurich [41] and was used on a quadrotor UAV to successfully navigate outdoor trajectories without GNSS augmentation. It was later expanded to use multiple sensors by Lynen in [21] where it is currently being actively researched, upon which SSF has effectively been deprecated. For the purposes of this research, MSF will be run with a single visual SLAM measurement sensor. In this mode the position and orientation measurements come from a single sensor, PTAM and no data from GNSS, pressure altimeter or magnetic compass are used.

The filter consists of a set of *core states* which are the minimal states required to obtain a useful result. The core states are shown in Equation (2.19) and consist of the states discussed earlier in Equation (2.12) with 8 additional states. The additional states are $\mathbf{p}_i^c$ and $\mathbf{q}_i^c$, which correspond to the translation and rotation between the IMU and camera and λ, an arbitrary visual scale factor which represents the scale between the arbitrarily scaled visual SLAM data and metric units.

$$\mathbf{x} = \begin{pmatrix} \mathbf{p}_w^i & \mathbf{v}_w^i & \mathbf{q}_w^i & \mathbf{b}_w & \mathbf{b}_a & \lambda & \mathbf{p}_i^c & \mathbf{q}_i^c \end{pmatrix} \tag{2.19}$$

A visual representation of the reference frames used in the minimal implementation is shown in Figure 2.4. Green lines denote sensor measurements from the visual SLAM algorithm, orange denotes fixed transforms and grey denotes slowly varying transforms. Red lines denote variables estimated by the Kalman filter and are used for UAS control. Frame V is the visual reference frame, and all vision based measurements are made with respect to this frame. Frame C is the location of the camera centre on board the aircraft, and frame I is the location of the IMU.

Note that the MSF assumes the IMU is located at the aircraft's CoG. Frame W is the world reference frame. Without a proper world reference frame W and frame V have the same origin, however a rotation may exist between frame V and frame W. Frame W is gravity aligned.

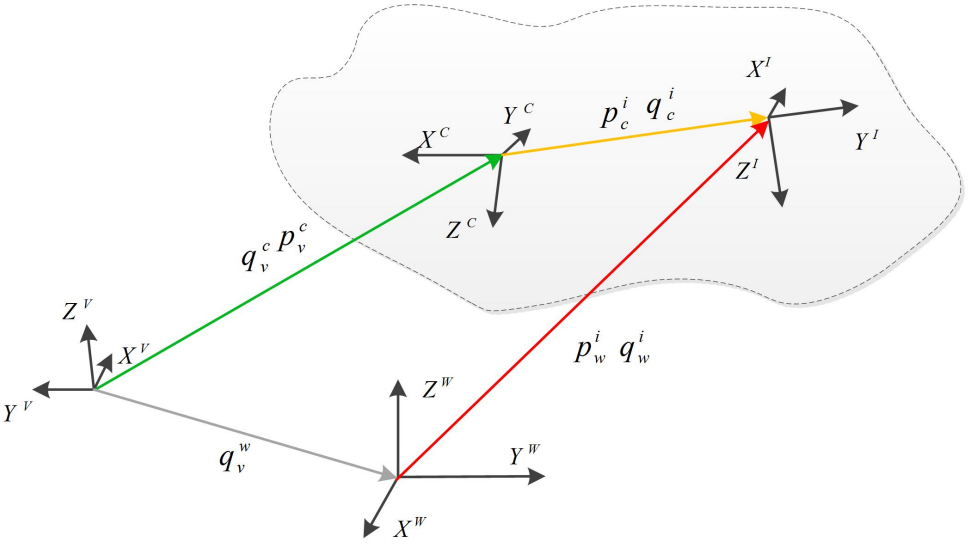


FIGURE 2.4: Co-ordinate frames in a minimal sensor implementation of MSF

The estimation of the visual scale factor is essential due to the fact that a global absolute scale is not observable from a monocular SLAM system. The filter states are predicted according to Equations (2.13) to (2.17), where the predictions for the new states are considered to propagate with no change. They are updated according to the measurement model presented by Weiss et. al. in [38], which are Equations (2.20) and (2.21) modified to take into account the scale and rotation of the visual SLAM system.

$$\mathbf{z}_p = \mathbf{p}_c^v = \mathbf{R}^T_{(\mathbf{q}_w^v)} \left(\mathbf{p}_w^i + \mathbf{R}^T_{(\mathbf{q}_i^w)} \mathbf{p}_i^c \right) \lambda + \mathbf{n}_p \tag{2.20}$$

$$\mathbf{z}_q = \mathbf{q}_c^v = \mathbf{q}_c^i \otimes \mathbf{q}_i^w \otimes \mathbf{q}_w^v \tag{2.21}$$

The measurement equations are linearised and used to update the Kalman filter state according to the algorithm shown in Algorithm 1.

It is worth noting that the camera-IMU transform is typically constant during filter operation (and in fact it is required for full observability), however the authors left the states in the filter for completeness [38]. The vision frame V is the point that the visual SLAM algorithm considers its origin. It is required that the vision frame is roughly aligned with gravity for the filter to converge quickly to correct estimates. The world-vision rotation $\mathbf{q}_v^w(k)$is estimated by

Equation (2.23) which is also used to detect failures of the visual SLAM algorithm by checking if the instantaneous value of $\mathbf{q}_v^w(k)$ is within the 3σ bounds of $\hat{\mathbf{q}}_v^w(k)$.

$$\mathbf{q}_v^w(k) = \hat{\mathbf{q}}_w^{i^{-1}}(k) \otimes \hat{\mathbf{q}}_i^{c^{-1}}(k) \otimes \mathbf{q}_v^c(k) \tag{2.22}$$

$$\hat{\mathbf{q}}_v^w(k) = med(\mathbf{q}_v^w(i)) \qquad i = k - N \rightarrow k \tag{2.23}$$

For cases where the only position sensor is visual SLAM, absolute world position is not observable. In this case, the origin of the vision frame V is considered to be coincident with the origin of the world frame W.

2.3.2 Additional sensors used in framework

The outline shown in Section 2.3.1 is only a minimal working implementation of a vision based EKF framework. In practice, modern UAVs have a much larger sensor suite available. A typical sensor payload will consist of a barometric pressure sensor, a 3 axis magnetometer and a GNSS receiver. These sensors are then used to augment the minimal implementation by providing barometric altitude measurements, absolute heading measurements and absolute world position measurements, respectively. These sensors require their own measurement matrices $\mathbf{H}_{sensor}$ and may introduce additional states. The co-ordinate frames for each sensor can be seen in Figure 2.5. Note that not all the co-ordinate frame transforms are 6 Degree of Freedom (DoF). For example, the GNSS senses only absolute position – the orientation of the sensor is irrelevant and thus not required. The following paragraphs will address how each additional sensor is handled within the sensor framework and the benefits of each sensor.

Absolute Pressure Sensor

Full size aircraft have been equipped with membrane based pressure altitude sensors since the beginning of modern aviation. In recent years MEMS-based absolute pressure sensors have shrunk to sizes well suited for MAVs. They typically have accuracy in the $0.020mbar$ range, corresponding to an altitude resolution of approximately $30cm \quad RMS$. Pressure altitude (in metres) can be obtained from a barometric pressure measurement (in millibars) through Equation (2.24), a **NOAA** standard model [26].

$$h_{alt} = 44,307.694 \left(1 - \frac{h_{press}}{1013.25}^{0.190284}\right) \tag{2.24}$$

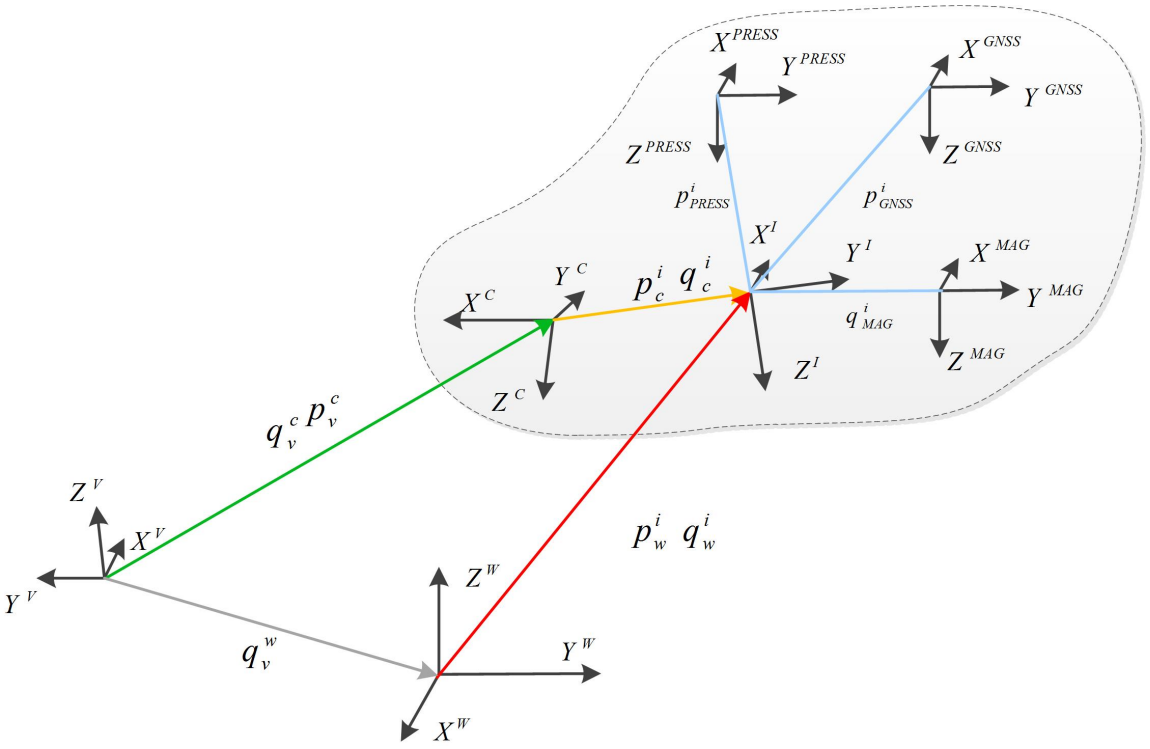


FIGURE 2.5: Co-ordinate frames in a typical implementation of MSF

Practically, the pressure sensor model will be similar to the accelerometer and gyroscope - that is to say it will consist of a pressure reading consisting of the true pressure plus some bias, as well as some zero mean Gaussian noise, shown in Equation (2.25).

$$h_{pressure} = p_z + b_{press} + n_{press} \tag{2.25}$$

Note that in general the addition of the pressure sensor does not change the observability of the system due to the fact that it introduces another state as well as a measurement. If the sensor is drift free (i.e. not affected by atmospheric changes), the sensor will add to the observability. Practically this is not the case, however over short terms the bias is constant which can aid the filter. Furthermore, it is possible to partially compensate for atmospheric changes through techniques like *QNH* (the practice of adjusting the barometric pressure to sea level) or regional pressure settings. Pressure sensors may also be used to set the local height above ground immediately after takeoff, allowing a better initial estimate of the visual scale factor and thus faster convergence.

Magnetometer

As noted by Kelly and Sukhatme in [18] and Weiss in [41], the minimal state implementation of MSF yields one unobservable state, causing states $\mathbf{x}_U = \left(\mathbf{p}_w^i\ \mathbf{v}_w^i\ \mathbf{q}_w^i\ \mathbf{p}_v^w\ \mathbf{q}_v^w\right)$ to be jointly observable. The obvious solution is to use the visual estimate of yaw; however, due to the visual frame drift errors will accumulate in the world-vision rotation $\hat{\mathbf{q}}_v^w$ and ultimately cause error in the position and velocity estimates of the filter. In the minimal sensor implementation of MSF this error is unavoidable without some measurement of global yaw. While complicated solutions such as sun sensors or a second camera as a visual compass will render the system observable, the most straightforward solution is to use a 3 axis magnetometer as a digital compass to measure the Earth's magnetic field. Following the magnetometer model presented by Weiss

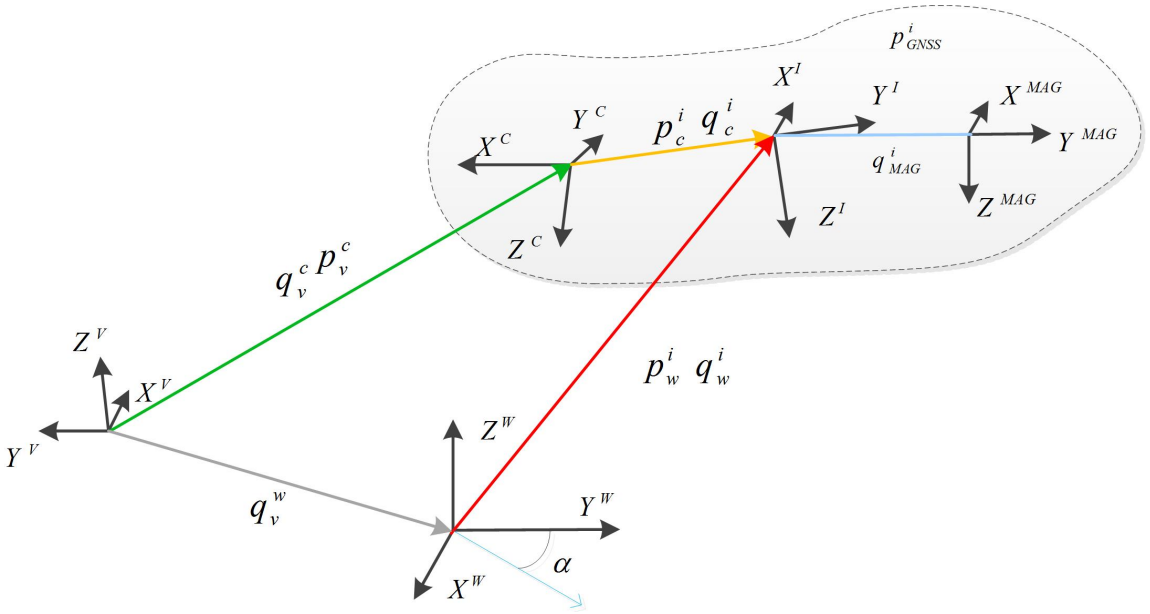


FIGURE 2.6: Co-ordinate frame of MSF with Magnetometer

in [41] five states are added to the Kalman filter framework, $\mathbf{q}_i^m$ and α, where $\mathbf{q}_i^m$ is the unit rotation quaternion from the magnetometer reference frame to the IMU reference frame and α is the magnetic inclination angle between the measured magnetic field vector and the world frame. The model presented by Weiss assumed that the α is constant which is valid over small geographic regions, however for flights on the order of hundreds of kilometres this assumption will introduce estimation errors in $\mathbf{q}_w^v$ and thus estimates in world position. The propagation

model is presented in Equation (2.26) and the measurement model presented in Equation (2.27).

$$\dot{\mathbf{q}}_i^m = 0 \quad \dot{\alpha} = 0 \tag{2.26}$$

$$\mathbf{H}_{mag} = \mathbf{R}_{(\mathbf{q}_i^m)}\mathbf{R}_{(\mathbf{q}_w^i)}m(\alpha) + \mathbf{n}_m \tag{2.27}$$

Note that while the addition of the magnetometer will allow the system to be fully observable, the visual frame origin is still defined to be the world origin and thus the system is still susceptible to the accumulation of position errors due to drifting visual frame origin. In order to eliminate these errors the system must model the visual drift state $\mathbf{p}_v^w$, however this would render the system unobservable. To fully fix this drift a form of absolute position measurement is required.

GNSS for Absolute Position Measurement

In Section 2.3.1 it was noted that the world-vision frame drift $\mathbf{p}_v^w$ is unobservable in the minimal sensor implementation of MSF due to the lack of absolute position measurements (since the visual SLAM algorithm reports location from an arbitrary origin). While it is possible to obtain 3 DoF position measurements from many sources (such as laser tracking or 6 DoF vision based motion capture systems), the most practical method is to use some form of GNSS receiver which provides absolute position measurements according to a reference datum, typically *WGS84* for GPS based systems.

The addition of an absolute position sensor yields six new states, $\mathbf{p}_i^g$ as the translation between the IMU and GNSS receiver and $\mathbf{p}_v^w$ as the drift between the visual frame origin and the world origin. The IMU-GNSS transform is considered fixed for a given airframe and the visual frame drift is considered to vary spatially and not temporally, due to the previously mentioned globally optical flow behaviour of ETHZ PTAM. As such, the state propagation equations can be written as shown in Equation (2.28).

$$\dot{\mathbf{p}}_i^g = 0 \quad \dot{\mathbf{p}}_v^w = 0 \tag{2.28}$$

The state propagation matrix is thus padded with zeros and will not add any unknown states that need to be observed. Note that due to the inclusion of the visual frame drift the measurement equation for the visual SLAM system now must take into account the newly available data. The resultant measurement equation for the GNSS sensor can be seen in Equation (2.29) and the vision based sensor in Equation (2.30). Note that the visual SLAM measurement equation

presented here differs from the one presented in Equation (2.20) due to the fact the visual frame drift is now observable.

$$\mathbf{H}_{\mathbf{p}_{gnss}} = \mathbf{p}_w^i + \mathbf{R}^T_{(\mathbf{q}_w^i)}\mathbf{p}_i^g + \mathbf{n}_g \tag{2.29}$$

$$\mathbf{H}_{\mathbf{p}_{vis}} = \lambda\left(\mathbf{p}_v^w + \mathbf{R}^T_{(\mathbf{q}_v^w)}\left[\mathbf{p}_w^i + \mathbf{R}^T_{(\mathbf{q}_w^i)}\mathbf{p}_\mathbf{i}^\mathbf{c}\right]\right) \tag{2.30}$$

The addition of absolute measurements means that the visual SLAM estimates must now be considered relative measurements. This is due to the expected variations in scale estimates from the filter, coupled with the globally optical flow type behaviour from keyframe (or map size) limited visual SLAM algorithms. The authors address the necessity of incorporating visual SLAM estimates as relative measurements in [21]. Namely, the covariance of the visual SLAM position estimates should increase with distance travelled in order to properly represent the scale error, world-vision frame rotation error and the optical flow type behaviour of the vision system. This approach allows fusion of both GNSS and vision based data, where as previous approaches in [32, 39] treat position measurements as absolute, prohibiting fusion of multiple position measurement sources. This approach does degrade theoretical performance when stationary due to the fact when far away from the origin the SLAM position estimate may not be accurate (position estimate will be off from GNSS estimate) but it will be precise (low variance from frame to frame), where GNSS data will have a relatively constant temporal drift.

2.3.3 Results from Implementations of Multi-Sensor Fusion

Both Single Sensor Fusion and Multi-Sensor Fusion have been successfully implemented and tested in both simulations and real world environments. The first working implementation of the algorithm was demonstrated by Weiss in [41], where the system was able to navigate successfully in large outdoor environments with little error. Figure 2.7 shows the results of a 500 second flight under vision navigation compared to GPS. Note there is marginal position drift over the course of the flight highlighting the global optical flow type behaviour of keyframe limited PTAM.

2.4 Conclusion

In this chapter a brief background on sensor fusion algorithms was presented. Additional it was discussed how Extended Kalman Filters are used in conjunction with the sensor suites found

FIGURE 2.7: Results from implementation by Weiss in [41], Fig. 4.35. States are converged by GPS and navigation is performed by vision measurements alone over the course of a 500 second, 360 metre flight. Resultant visual frame drift is minimal, with $p_v^w = [8.4m\ 9m\ -2.3m]^T$.

on typical unmanned aircraft. Basic sensor models for both the accelerometer and gyroscope were presented, along with state propagation algorithms for an inertially based Kalman filter. The SLAM problem was presented and various algorithms were outlined. Parallel Tracking and Mapping was chosen to be the visual SLAM algorithm due to the ability to bound computational complexity while maintaining robust map tracking. The scale observability problem with monocular cameras was discussed and it was shown that for typical UAS operations stereo vision cameras offer little to no advantage. It was noted that by fusing IMU data with monocular vision systems it is possible to observe the scale, which is the basis for further research. The modular Extended Kalman Filter algorithm developed by ETH Zurich was examined, as well as the impact of various different measurement sensors. Additionally, existing implementations of the MSF algorithm were discussed in order to provide a basis for future work. The work presented in the following chapter will outline how to utilise PTAM and MSF as a backbone framework upon which the gimbal control algorithm can be seamlessly integrated.

Chapter 3

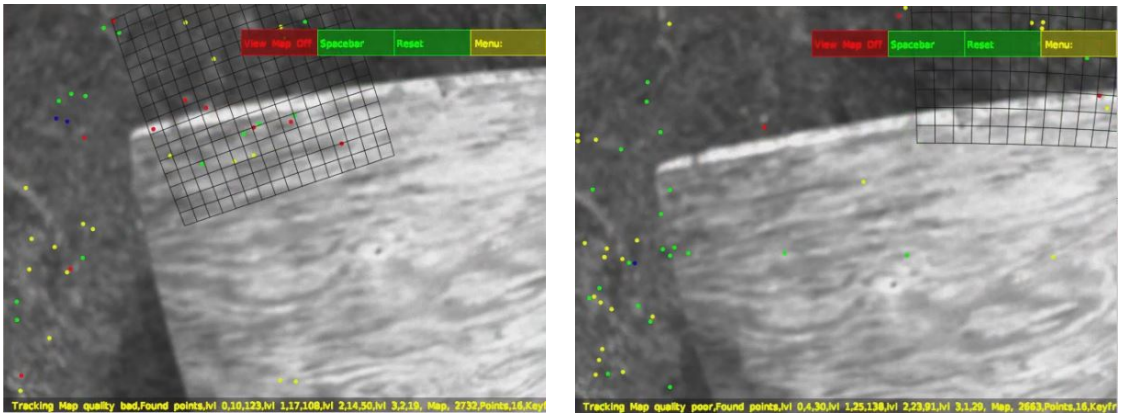
Gimballed Visual SLAM

3.1 Motivation

A camera that is rigidly attached to an aircraft is not always able to observe enough key features to allow a visual SLAM algorithm to function reliably. It may be possible to orient the aircraft in such a way to increase the amount of points visible; however, typical SLAM algorithms are not coupled with control algorithms. Furthermore, this behaviour would restrict the operations of the aircraft. A movable camera that is independently capable of orienting itself based on the features visible in the scene is thus desired.

3.1.1 Overview of Corner Tracking

All visual SLAM algorithms rely on matching key points from one frame to another. By tracking these points across multiple frames it is possible to reconstruct depth information, and thus perform SLAM. Stereo vision systems are able to construct depth information directly through feature triangulation across a known camera baseline distance, however the system must first detect trackable features. This is in contrast with laser ranging SLAM algorithms which can use object depth information without any post processing from the sensor. Thus, at the core of any robust visual SLAM algorithm lies a robust feature detection algorithm. Most feature detection algorithms rely on corner detection as a form of robust feature tracking over multiple frames. It is important that the algorithm be robust to both blur and any potential noise from the camera itself. A key paper by Rosten et. al. compares many common corner detectors for both speed and repeatability [29]. A key observation in this paper is the fact that FAST



(A) Note origin grid is on corner of table and angled left

(B) After origin jump, grid largely off table and angled right

FIGURE 3.1: Flight data taken two frames apart shows origin grid jump approximately 0.5 metres and orientation change

detectors are typically in the range of 70% repeatability, thus highlighting the need for features which can reliably be tracked.

The obvious issue with this approach is that there is no guarantee that there will be sufficient features in a given frame in order to correctly apply the tracking algorithm. If this is the case, it is not possible to estimate the camera location and the SLAM algorithm is no longer has a solution. Tightly coupled filter based algorithms may be able to predict camera location based on odometry or IMU data in an attempt to maintain a coarse estimate of location within a larger map. However, when the visual estimator is uncoupled from the filter (as is the case with MSF and PTAM), the loss of tracking means the map is usually unrecoverable and the best approach is simply to re-initialise the SLAM algorithm.

Figure 3.1 shows camera frames captured from PTAM during a flight test. In this situation the aircraft is hovering approximately $0.5 metres$ above a wooden table at a total altitude of approximately $1.5 metres$. Figure 3.1a shows a normal tracking situation where map points are tracked correctly and the reference grid is approximately gravity aligned and placed on the edge of the table. Figure 3.1b shows how PTAM handles a tracking loss. Note the reference grid is no longer gravity aligned, indicating that both position and orientation estimates from the visual SLAM algorithm are corrupted, meaning that recovery is unlikely.

3.1.2 Sources of Tracking Error

In order to improve tracking behaviour we must first understand the sources of tracking error. We will consider three major sources of tracking error, the first being lens distortion, the second due to matte surfaces and the third due to repetitive textures.

Distortion due to the Camera Lens

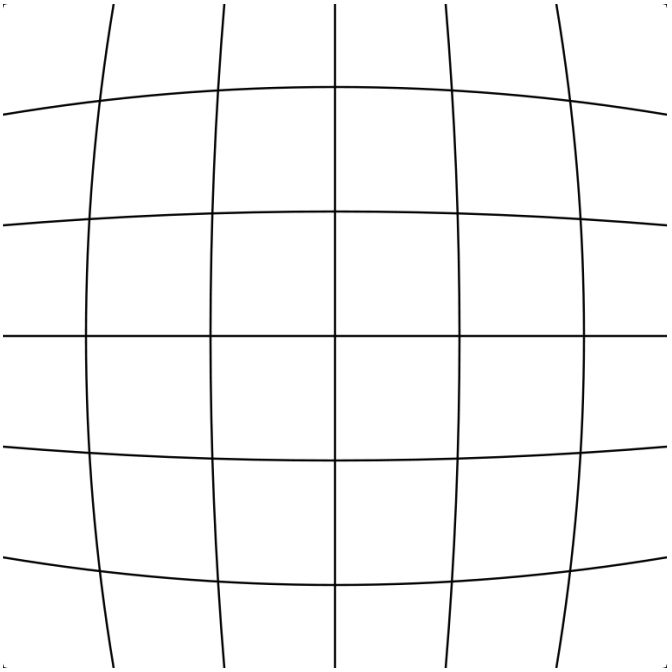
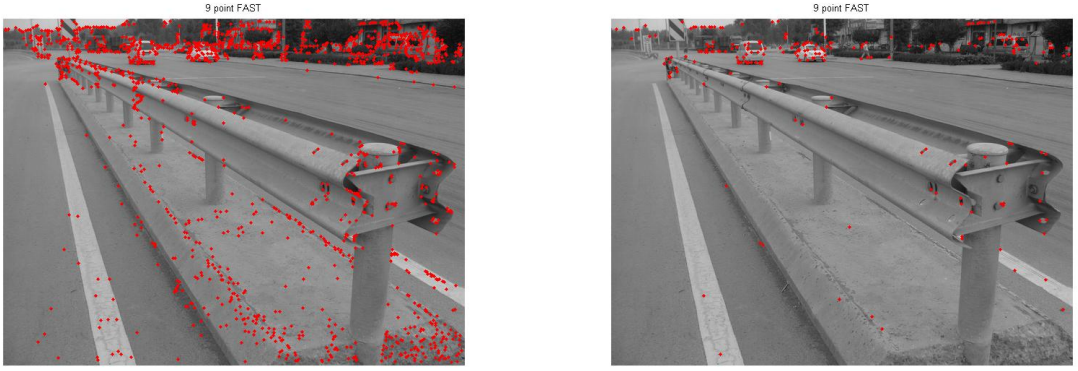


FIGURE 3.2: Barrel distortion causes straight lines to appear curved, shown by the above grid with intersecting right angle lines.

Figure 3.2 shows an image of a grid as viewed through the camera used in these experiments (a Point Grey Firefly MV with a 120° FoV lens). Note that the straight lines of the grid are distorted by the lens through an effect known as barrel distortion so that they appear curved. Typically barrel distortion is most easily visible on lower cost wide angle lenses, exactly the type used in low cost SLAM systems. If a lens with barrel distortion is not correctly calibrated there will be apparent motion between features as they move between the edge and centre of the lens. This will yield incorrect pose and orientation estimates which will ultimately lead to the instability of the SLAM algorithm. Fortunately, these distortions are easily modelled and calibration is able to largely remove the effect of barrel distortion. Many tools exist for the in field calibration of lenses, such as those in [8, 31]. PTAM uses the camera model presented in [8], though variants with large FoV cameras such as MC-PTAM use the omnidirectional



(A) Threshold = 20 (B) Threshold = 50

FIGURE 3.3: FAST9 corner detection algorithm applied to a street scene

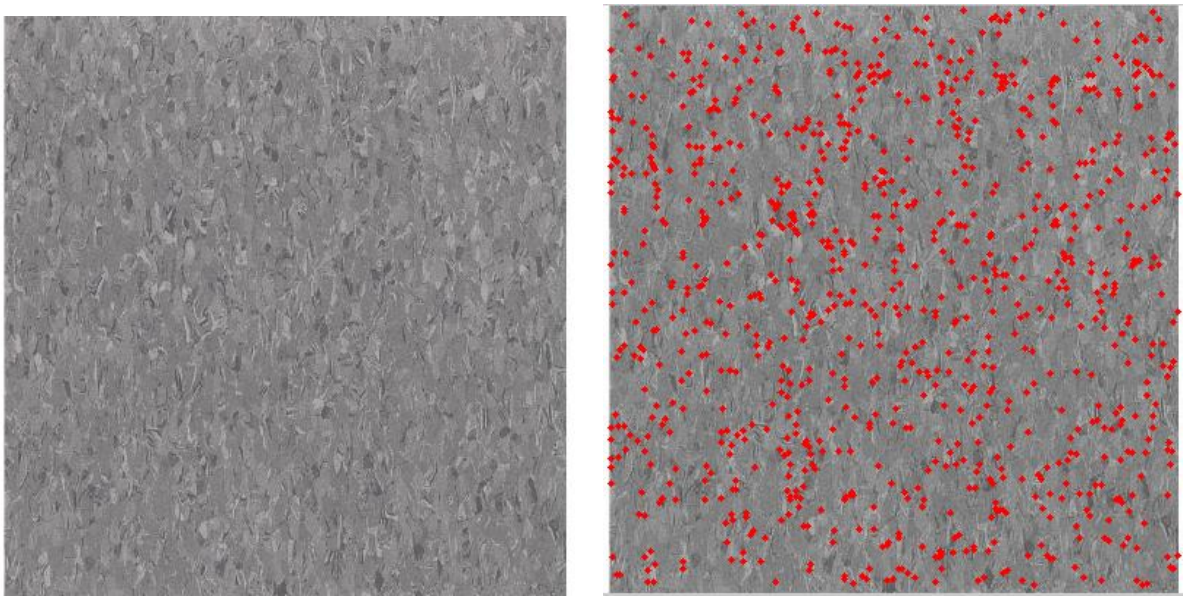
camera model presented in [31]. Since individual lenses may be calibrated we consider the error due to lens distortions to be negligible.

Lack of Trackable Features

Perhaps the greatest disadvantage visual SLAM algorithms have over their laser based counterparts is the fact that they require key features to track in order to operate, whereas laser range finders simply require a surface that reflects laser light. To illustrate this, consider the guard rail shown in Figure 3.3 which contrasts the result of the FAST9 corner detection algorithm with two different thresholds.

Figure 3.3a shows the result of a low threshold of 20. Note that many of the points detected are placed on nondistinct surfaces (such as the surface of the concrete), which means that these points will likely be unreliably tracked. In fact, attempting to track such points often leads to instability of the SLAM algorithm, so it is common to increase the threshold in an attempt to track only robust points. Increasing the threshold to 50 helps the problem; however, note that now the guard rail itself contains minimal tracked points. Furthermore, most of these points are at the very beginning of the rail and very few lie along the surface of the rail. If a world map were to be constructed using these points it is highly likely that the guard rail would not be correctly mapped, meaning the potential for collisions exists.

In this situation it may be possible to lower the threshold and still maintain reasonable tracking, however this is not always a desirable solution as it increases the computational cost and degrades map tracking.



(A) Textured commercial floor tile (B) FAST9 Corners, Threshold = 30

FIGURE 3.4: FAST9 corner detection algorithm applied to a repetitive texture

Repetitive Textures

Based on the fact that non-textured surfaces cannot be tracked one might think that a randomly textured surface would thus be ideal. In practise however this is rarely the case, as seen in Figure 3.4, where the FAST9 algorithm is applied to a textured floor tile.

Note that while the FAST algorithm detects many points (937 in this case) they are somewhat randomly distributed throughout the frame. The ultimate effect of this repetitive texture serves to confuse the tracking algorithm since instead of their being one globally optimal solution for a given image there are several locally optimal solutions. When considering reprojection errors, motion blur and uncorrected lens distortion it is not likely that the 'correct' solution will be chosen since there are many solutions which are equally likely. When the camera undergoes motion the texture will cause the position estimate to wander which may lead to instability of the visual SLAM algorithm and potentially the divergence of any filters which rely on this estimate. Attempting to increase the tracking threshold will result in a drastic loss of points since the texturing usually means all points have a similar threshold, and lowering the threshold only further increases the problem.

3.1.3 Improving Tracking Reliability

Several methods have been investigated to improve tracking robustness. A common approach is to use a forward facing camera as seen in [17], where PTAM is used with a forward looking camera for flying in an indoor environment. Two distinct advantages come from using a forward looking camera. The first is that the camera tends to look at the horizon of a scene and the second is that yawing the vehicle will expose the camera to new regions. Horizon tracking is helpful since there is generally a distinct transition between the earth and sky, however this is only of use when flying outdoors. Yawing to explore new regions is also of interest since it requires no vehicle motion, especially when compared to downward looking cameras which may require large lateral translations to expose the camera to new scenes. However, monocular vision based systems require translation between keyframes in order to initialise new map points, as depth information is gathered from the parallax associated with consecutive frames. Furthermore, while yawing is an easier manoeuvre to perform than translation it is undesirable to require specific UAS motion in order to have reliable tracking.

Multiple cameras have also been used under the assumption that it increases the likelihood of the overall system retaining tracking, even if individual cameras drop out. As previously mentioned, stereo cameras behave as monocular cameras for sufficiently long feature distances, so most often the cameras are oriented in such a way that they observe a different portion of the scene, as demonstrated by Harmat in [13] where MC-PTAM is used with two cameras in a stereo vision arrangement and a third viewing in a diametrically opposite fashion. In certain cases (such as obstacle avoidance), multiple cameras are the only way to reliably create a real-time view of the environment, as seen in [15]. For the purposes of pure navigation on a MAV multiple cameras often add a significant amount of weight to the platform, dramatically reducing both flight time and useful payload capacity.

For this investigation the algorithm developed is designed to specifically target regions with a lack of trackable features. This approach is likely to show the largest benefit of using a gimballed camera and can provide a basis to investigate different algorithms which focus on repetitive textures. The tracking algorithm developed in Section 3.2.2 uses a weighted edge method to determine the most feature rich area. In contrast, an algorithm designed to find the most trackable texture would likely need to incorporate an edge detector and a Hough or 2D Fourier transform, causing a considerable increase in complexity.

3.2 Gimballed Camera

3.2.1 Introduction

In this section we propose a novel method to use a movable gimballed camera to increase tracking robustness. Previous works consider only cameras which are rigidly attached to the aircraft and thus treat any observed camera movement as UAS movement. In general this approach works quite well however it forces the camera to undergo arbitrary and unpredictable movements. Highly dynamic UAS movements may cause tracking loss due to motion blur and the fact that most SLAM algorithms use a small search windows when reprojecting map points. It is also possible that the UAS will fly over terrain which contains minimal features (such as shorelines) or regions with repetitive textures (such as agricultural fields) which may lead to tracking instability and a degradation of precision.

To this end a system is proposed where a camera is attached to the airframe via a movable gimbal, allowing an arbitrary and variable rotation (within mechanical limits) to exist between the camera frame and the IMU frame, as illustrated in Figure 3.5. To the best of our knowledge this is the first system proposed using a gimballed camera for visual Simultaneous Localisation and Mapping purposes. The closest related work by Huh et. al [16] uses a 1D laser scanner mounted on a gimbal to acquire 2D scans of a region and feeds this data, together with data from a rigidly attached vision camera into a filter based SLAM algorithm. The scanning algorithm for this work is a continuous motion to track the visible features in the camera frame, and no motion is performed as a result of detected features.

3.2.2 Gimbal Pointing Algorithm

In Section 3.1.2 the causes of tracking error in existing visual SLAM algorithms were outlined. This section will explore how to use the gimballed camera to mitigate these errors through the development of a pointing algorithm. This pointing algorithm should point the gimbal towards the region which contains the most trackable features in the scene. To help illustrate this, consider the image shown in Figure 3.6, which represents the view from a UAS in an aerial surveying application. Note how the land contains many detected features while the water contains none. This is a large source of tracking error seen when flying over or near water, which is a largely non-trackable surface. This obviously poses a problem to a UAS with a single downward looking camera as it would not be able to fly over the water without losing visual

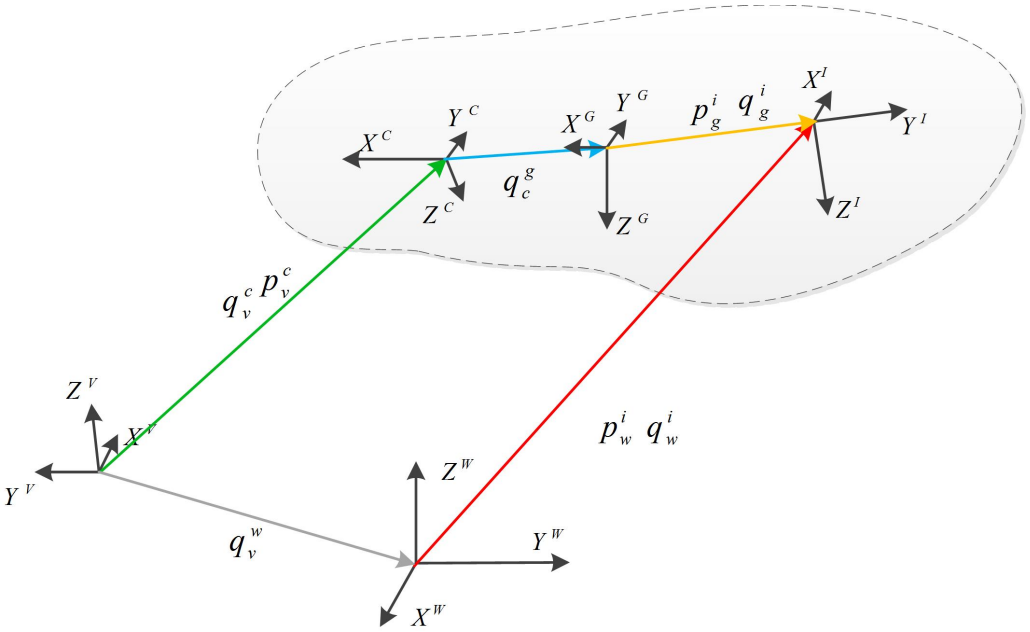


FIGURE 3.5: Co-ordinate frames in the proposed gimballed implementation of MSF

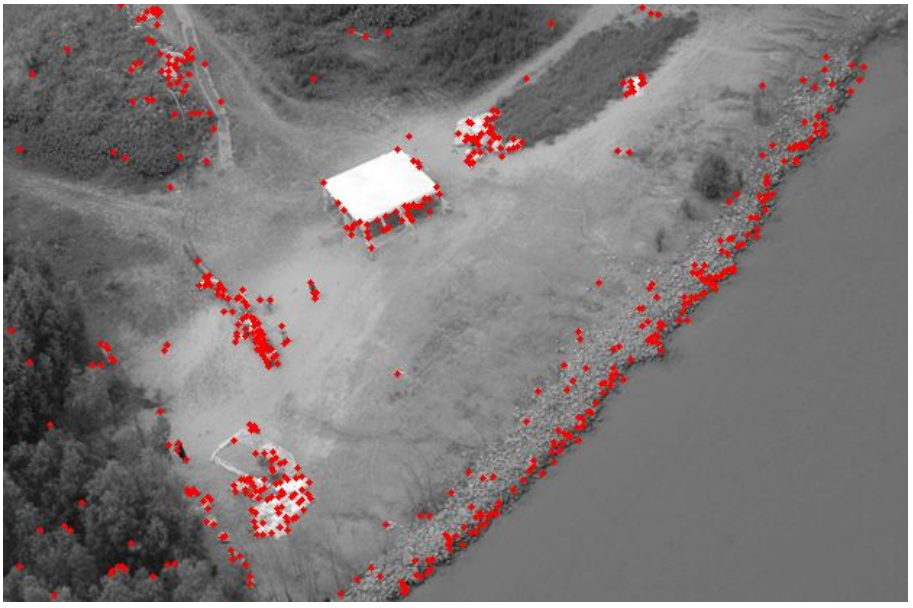


FIGURE 3.6: FAST9 Corners of Shoreline Scene

tracking. However, with a gimbal mounted camera it is possible to ensure the visual frame always has trackable features. To this end an algorithm is proposed which controls the gimbal based on corners detected by the tracking algorithm.

First, the image is segregated into three regions, the left and right regions of interest, and the central region. The regions of interest are as defined by the red outlines in Figure Figure 3.7, and are a subimage of width w_{sub} and image height h. Element wise multiplication with a weight matrix is then performed in order to compute the region's *feature score*, a measure of

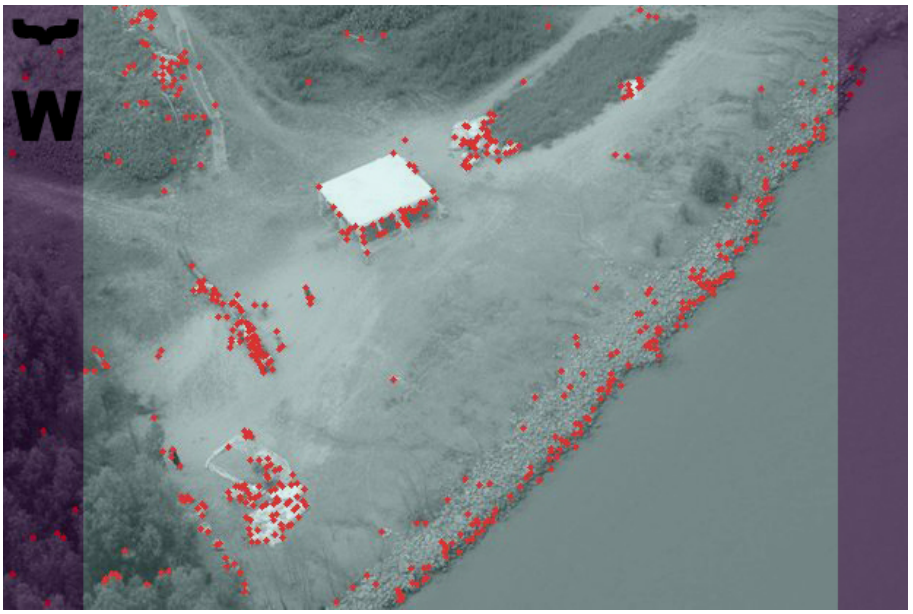


FIGURE 3.7: Image segmentation regions

the weighted feature density of the subimage. Higher scores imply that there are more trackable features in the subimage. An example weight matrix is shown in Figure 3.8, where black indicates no weight and white indicates full weight. A diagonal gradient is chosen for the weight matrix for multiple reasons. First, it is desired that the edge features receive a higher weight due to the fact that any camera rotation will drastically influence their visibility. Second, due to the forward UAS motion features near the bottom of the image frame will soon leave the camera frame, whereas features in the top of the image will slowly translate from the top to the bottom. Hence, more weight is placed on top features as they will have the longest influence over time. When combining these goals a higher weighting of the upper outer pixels results, translating to a gradient weighting matrix.

A simple linear mapping is used to calculate the point weighting according to Equation (3.1), where α and β are unit-less scaling factors that adjust how rapidly the gradient rolls off in the vertical and horizontal direction, respectively. This allows the weighting to be tailored to individual circumstances, for example in fast flight points will travel through the image rapidly, so it makes more sense to have a higher α value so that the weighting focuses more heavily on the outer edges of the image rather than the front to back.

$$weight[i,j] = \frac{j}{\alpha * imageHeight} + \frac{i}{\beta * scanWidth} \tag{3.1}$$

Once the feature score has been calculated according to algorithm Algorithm 2 the resultant

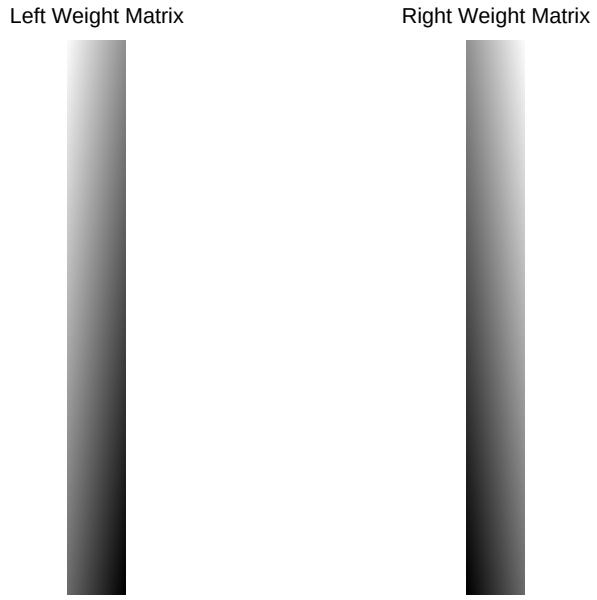


FIGURE 3.8: Gradient weighting matrices of width $w_{sub} = 25\ pixels$

feature score is used to calculate a desired change in camera tilt. A simple proportional controller is used to act on the desired angle, as shown in Equation (3.2). Note that since the feature

Algorithm 2 Calculate Feature Score of a Video Frame

$f_{score} = 0$
for $i = 1$ to w_{sub} **do**
  for $j = 1$ to h_{image} **do**
    $f_{score} = f_{score} + image[i, j] * weight[i, j]$
  end for
end for
Repeat for both left and right sub-images.
Normalise feature scores:
$leftScoreNorm = leftScore/(leftScore + rightScore)$
$rightScoreNorm = rightScore/(leftScore + rightScore)$
$F_{score} = 2 * (leftScoreNorm - 0.5)$

score has been normalised on the range $(-1..1)$ the gain factor $K_{rollGain}$ will correspond to the maximum change in roll for one camera frame, corresponding to the case where one sub image contains no features.

$$\theta_{cam}[k+1] = \theta_{cam}[k] + K_{rollGain} * F_{score} \tag{3.2}$$

The camera pointing algorithm thus updates at the camera frame rate. A servo driven gimbal with a maximum update rate of approximately $50Hz$ is used, so the camera frame rate of

$30Hz$ will not saturate the gimbal controller. If higher frame rate cameras are used then frame averaging should be employed to prevent saturation. It was previously explained how monocular SLAM algorithms require camera motion to build a 3D map of the surroundings, and how pure rotation can cause instability in the algorithm. Thus, a velocity based dead-band is employed around which no camera movement is allowed, effectively rendering the gimbal fixed. A slew rate limit is also used along with a maximum angle limiter to prevent motion blur and over-driving the gimbal, respectively. The resultant controller is visualised by the block diagram in Figure 3.9

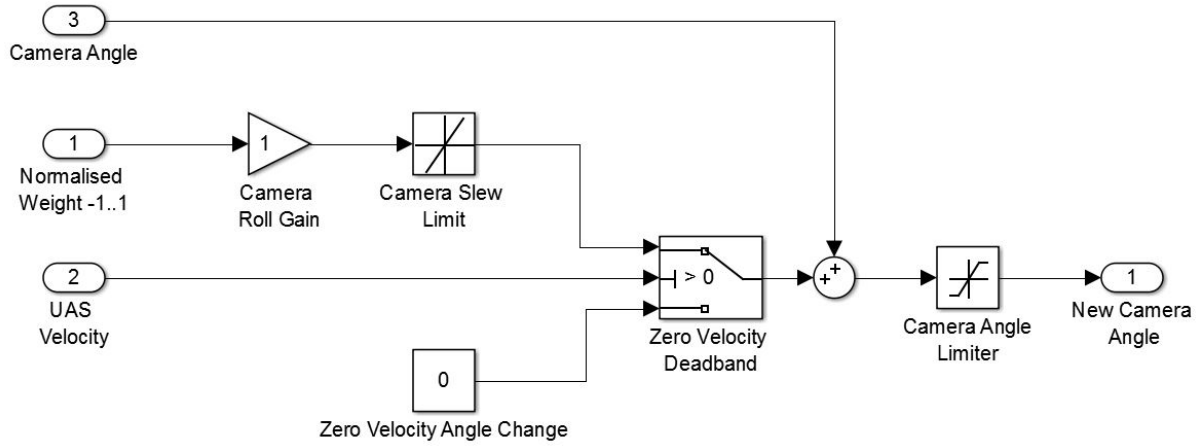


FIGURE 3.9: Block diagram of camera gimbal control algorithm

The experiment is confined to a single axis gimbal operating on the aircraft's roll axis (i.e. changing the gimbal tilt angle changes apparent roll). It is not expected that a multi-axis gimbal would offer much increased benefit for multiple reasons. Consider if the control algorithm was extended operate on the aircraft's pitch axis. The rotation would thus point the camera along the vehicles direction of travel, meaning it would only be able to explore scenes that the UAS would soon be flying over anyway. This is in contrast to a roll based control where the gimbal rotation allows new scenes to be explored.

It is important to note that this type of control is only capable of finding a local maximum number of points. For example, in Figure 3.6 it is entirely possible that there are many trackable features on the opposite shore of the river, more even than currently visible. In this scenario, the camera will never rotate to the right to be able to see these features and thus will find the local maximum of corners on the left half of the river. This is considered acceptable since in order to find the global maximum the camera may need to rotate through regions with no corners, thus introducing tracking loss.

3.2.3 Rotation Inverser

The addition of the gimbal introduces a new rotation, $\mathbf{q}_c^g$ between the camera and the gimbal, which represents the rotation between the current camera orientation and the fixed orientation of the gimbal. The rotation $\mathbf{q}_g^i$ is the rotation between the IMU and gimbal and is considered fixed, similar to the rotation $\mathbf{q}_c^i$ in the non-gimballed implementation shown in Figure 2.4. The orientation measurement of the visual SLAM algorithm now must take into account the gimbal rotation and is shown in Equation (3.3), in contrast to the non gimballed orientation measurement shown in Equation (2.21).

$$\mathbf{z}_q = \mathbf{q}_v^c = \mathbf{q}_v^w \otimes \mathbf{q}_w^i \otimes \mathbf{q}_i^g \otimes \mathbf{q}_g^c \tag{3.3}$$

Note that if these states were modelled in the Kalman filter then without any additional measurements it would render the system unobservable. As such, the rotation is not modelled in the Kalman filter algorithm but instead the rotation is subtracted from the visual SLAM measurement before it is used in the Kalman filter. This of course means that the gimbal must have position feedback in order to calculate the orientation and implement the subtraction algorithm. To this end, a gimballed visual SLAM system structure is presented in Figure 3.10. This ap-

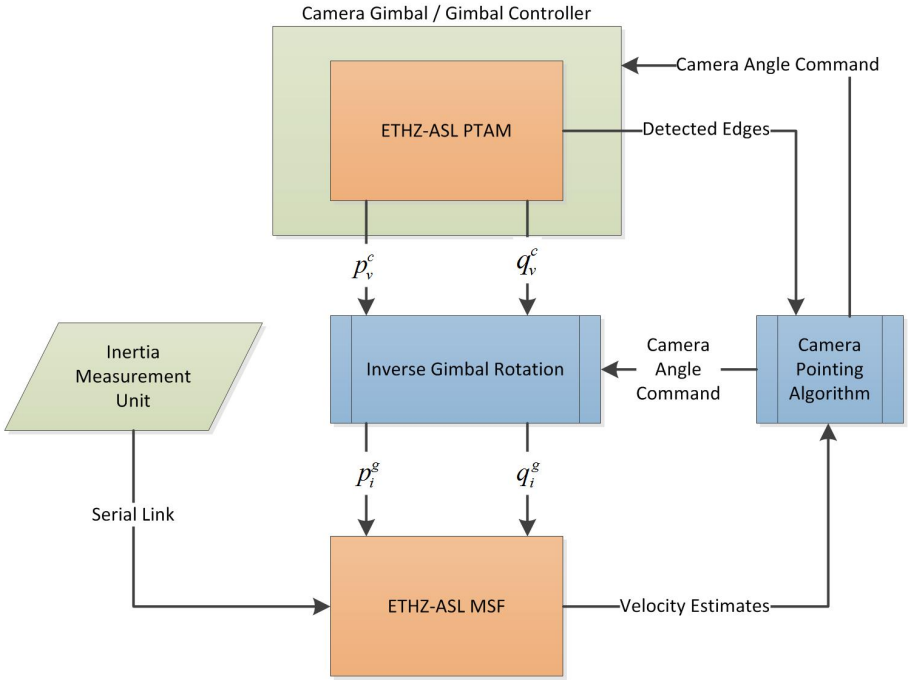


FIGURE 3.10: Block Diagram of System with Gimbal

proach will supply the MSF framework with the orientation gimbal $\mathbf{q}_g^i$ which is rigidly attached

to the airframe and will thus be indistinguishable from the original $\mathbf{q}_c^i$. The measurement to the EKF is thus as shown in Equation (3.4).

$$\mathbf{z}_q = \mathbf{q}_v^w \otimes \mathbf{q}_w^i \otimes \mathbf{q}_i^g \otimes \left(\mathbf{q}_{g_{actual}}^c \otimes (\mathbf{q}_{g_{command}}^c)^{-1}\right) \tag{3.4}$$

Note that in this situation any errors between the commanded and actual gimbal rotations (i.e. $\mathbf{q}_{c_{actual}}^g \otimes (\mathbf{q}_{c_{command}}^g)^{-1} \neq [1\ 0\ 0\ 0]$) will directly translate to errors in the estimation, so it is possible that the covariance of the visual SLAM orientation measurement may need to be increased. Encoders may serve to increase the accuracy of the gimbal position feedback but would add complexity and require calibration, and sufficiently accurate encoders are prohibitively large and expensive. It is also possible to model this rotation as states in the Kalman filter, where the prediction is simply the commanded angle and the measurement is the measured angle, however little benefit would be gained at the cost of increased computational complexity in the filtering algorithm.

Additionally, note that the position measurement from PTAM is the location of the world origin with respect to the camera. This means that as the camera rotates, the position reported by PTAM will also change as the displacement is mapped to different axes. The position reported by PTAM must thus be rotated by the opposite gimbal rotation in order to null any apparent motion. This rotation is shown in Equation (3.5). Once again, for the case where $\mathbf{q}_{c_{actual}}^g \approx \mathbf{q}_{c_{command}}^g$ the resultant position measurement can be represented as Equation (3.6)

$$\mathbf{z}_{\mathbf{PPTAM}} = \mathbf{q}_{c_{actual}}^g \otimes \mathbf{z}_{\mathbf{PGIMBAL}} \otimes \left(\mathbf{q}_{c_{actual}}^g\right)^{-1} \tag{3.5}$$

$$\mathbf{z}_{\mathbf{PGIMBAL}} \approx \left(\mathbf{q}_{c_{command}}^g\right)^{-1} \otimes \mathbf{z}_{\mathbf{PPTAM}} \otimes \mathbf{q}_{c_{command}}^g \tag{3.6}$$

3.3 Algorithm Simulation

To verify algorithm operation we simulate it on a series of images representing a video stream from the visual SLAM camera. We consider a situation where the UAS has a gimbal with a maximum roll of 20°, a $K_{rollGain} = 0.1°/frame$ and a sinusoidal velocity varying between $(0 - 0.8m/s)$ with a period of 1000 frames. The velocity is corrupted by zero mean white noise with an amplitude of $0.1m/s$, corresponding to expected noisy velocity measurements from a Kalman filter. In Figure 3.11 the results of a simulation showing the effect of a velocity deadband of $0.3m/s$ are presented. The effects of the deadband are clearly visible from frames 100-300 and

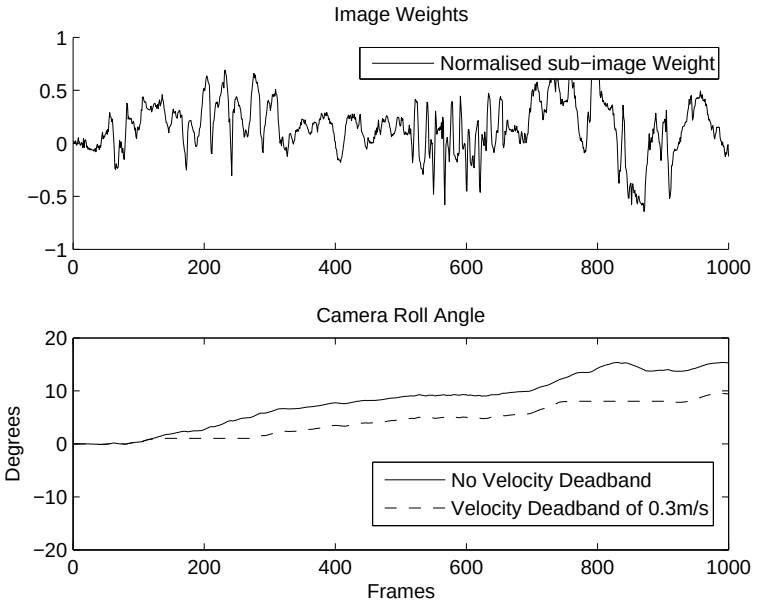


FIGURE 3.11: Simulation of gimbal control algorithm showing effect of $0.3m/s$ velocity deadband

again from approximately 700-900, where there is no change in the commanded gimbal roll. This is representative of what is expected during flights where the UAS flies from one waypoint to another and loiters at the waypoint. The velocity deadband prevents camera rotation when the UAS is reasonably stationary which reduces the chance of a map corruption. In this simulation the slew rate limit was set to $5°/s$ and due to the fact that $K_{rollGain} = 0.1°/frame$ and the frame rate is $30Hz$, resulting in a theoretical maximum slew rate of $0.3°/s$ and so the slew rate can never be reached. Figure 3.12 shows the results of the simulation when $K_{rollGain} = 0.8°/frame$ From frame 100-200 there is a rapid increase in commanded angle and how the slew rate limit affects the angle change. The effect of the hard angle limit of 20° is visible, preventing the algorithm from over-driving the gimbal. In general it is useful to note that the normalised feature score is quite noisy. This noise is mostly smoothed out by the use of sufficiently small gains, however for larger gains this may result in jerky motion. For this reason it may be useful to apply an averaging or median filter to the feature score of camera motion is notably unsteady.

3.4 Experimental Implementation

In order to function within the existing MSF and PTAM framework the gimbal control algorithm was implemented as ROS nodes. In general, a ROS node can publish and receive messages, and perform calculations. For the purposes of this research, three new nodes were created; the gimbal

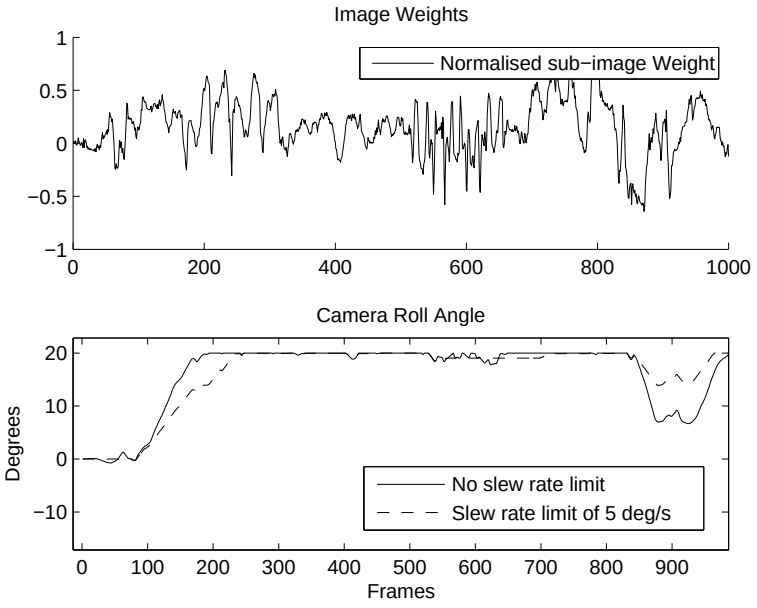


FIGURE 3.12: Simulation of gimbal control algorithm showing effect of $0.3m/s$ velocity deadband

control node, the rotation inverser node, and the IMU streaming node, shown in Figure 3.10. The gimbal control node receives features from the SLAM algorithm and velocities from the Kalman filter and outputs a gimbal control command. The rotation inverser subtracts the effect of the gimbal rotation on the position and orientation reported by the SLAM algorithm and provides an unrolled SLAM pose to the Kalman Filter. Finally, the IMU streaming node interfaces with the autopilot to receive accelerometer, gyroscope, GPS and compass data and supply it to MSF.

3.4.1 Rotation Inverser Node

The function of the rotation inverser node is to subtract the rotation of the gimbal from the orientation measurement supplied by the visual SLAM framework. We use a simple implementation shown in Equation (3.7) to subtract the gimbal rotation. We subscribe to the rotation topic published by the visual SLAM algorithm in order to obtain the vision-camera rotation $\mathbf{q}_v^c$. We then multiply this by the inverse of the commanded gimbal orientation $\mathbf{q}_c^g$ in order to obtain $\mathbf{q}_v^g$, the orientation of the gimbal with respect to the arbitrary visual frame, which is then supplied as a published ROS topic to the EKF framework as the global orientation measurement. The position measurement is implemented in Equation (3.8) as originally presented

in Equation (3.6).

$$\mathbf{q}_v^g = \mathbf{q}_v^c \otimes \mathbf{q}_c^g \tag{3.7}$$

$$\mathbf{p}_v^g = \left(\mathbf{q}_{c_c}^g\right)^{-1} \otimes \mathbf{z}_{\mathbf{pPTAM}} \otimes \mathbf{q}_{c_c}^g \tag{3.8}$$

This simplistic approach assumes the gimbal orientation is exactly the commanded orientation. In practice there will be several sources of error in this assumption. The most notable errors are the delay between the command being given and the orientation being achieved and the intrinsic positioning error, however we will consider both sources to be negligible. The command delay is negligible to due the fact that for each update we command a very small orientation change, between $0.1 - 0.8°$. When coupled with the fact modern servos have transit rates of upwards of $600°/s$ this translates to a command delay of approximately $1ms$. When coupled with the update rate of the gimbal itself the delay is on the order of $10ms$ which is well below the camera update rate of $33.3ms$. We neglect pointing error since a 12 bit digital servo will have approximately $0.04°$ resolution which is well below the variance of PTAM itself. Furthermore since any error not removed during calibration will manifest as an error in the orientation measurement we can artificially increase its covariance in order to compensate for any pointing error at the cost of minimal orientation accuracy. As was previously mentioned, sufficiently accurate optical encoders are prohibitively expensive and magnetic encoders may suffer from externally introduced errors, which would also degrade the orientation measurement.

3.4.2 Gimbal Pointing Node

The gimbal pointing node is responsible for implementing the pointing algorithm as defined in Section 3.2.2. Using the block diagrams from Figures 3.9 and 3.10 we know that this node will require the velocity estimates from the EKF and the detected corners from the tracking stage of PTAM. The velocity estimates are already publish by MSF as part of the state vector and require no modification for use. The corners are not readily available however and thus PTAM was modified to publish an additional topic containing the corners detected by the FAST9 algorithm. PTAM uses a 4 level pyramid structure to perform tracking. Level 0 corresponds to the full 640*480 image which is down-sampled to half size using nearest neighbour interpolation for Level 1, 2, and 3 meaning Level 3 contains a 'small blurry image' of 80*60 pixels. While this image is small it stands to reason that any corners found here will be significantly more robust than those found in Level 0. For completeness, the modified PTAM algorithm can be made to

output corners on all four levels, allowing the gimbal controller to use the robust points found in Level 3 or the plentiful corners found in Level 0. The output of the gimbal pointing node is the desired camera tilt angle (roll in our case), which is published as a ROS topic.

For the purposes of convenience the gain, slew limit and pyramid level are all changeable on the fly by the user. The gimbal is also able to be controlled manually, allowing the user to take total control of the gimbal and hold it at a fixed position or point it to a region of interest. The gimbal controller can be turned on and off on the fly.

3.4.3 IMU Streaming Node

For the purposes of this research, the UAS used contains a MicroPilot $MP2128^{g2HELI}$ autopilot with on board 6 DoF IMU, GPS and barometric pressure sensor. The autopilot is augmented with a 3 axis magnetometer to provide absolute heading information. The autopilot is shown in Figure 3.13. Data are transferred from the autopilot to the Single Board Computer (SBC) via a

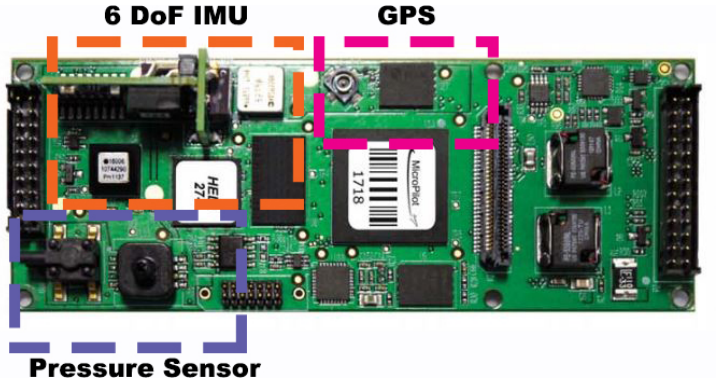


FIGURE 3.13: $MP2128^{g2HELI}$ Autopilot

serial data link. The SBC then performs the sensor fusion algorithm on the data and stores the result for post flight analysis. For the purposes of this research, GPS is used as ground truth and the compass is used for heading initialisation. They are not used in the Kalman filter for estimation or measurement updates.

The IMU streaming node is responsible for all communication with the autopilot. The autopilot is connected to the computer by means of a $38,400$ *baud* serial data link. In this case, we are supplying MSF with 6DoF IMU data from the autopilot, as well as GPS position and pressure altitude readings. The data sent from the autopilot have been temperature, scale and cross-axis compensated by calibration at the production facility. IMU data are sent at a rate of $90Hz$. GPS and pressure altitude data are sent at a rate of $4Hz$ and $5hz$, respectively. The data are

published on a topic by the node which is then subscribed to by the MSF node. This node is also responsible for updating the servo controlling the gimbal and as such the desired gimbal position from the pointing node is received by the IMU streaming node and sent to the autopilot which then updates the servo gimbal.

3.5 Conclusion

In this chapter some current limitations that exist when trying to develop a robust monocular visual SLAM algorithm were presented. Several key papers and methodologies currently used to increase the robustness of visual SLAM were outlined, along with potential limitations of these methods. A novel method of using a gimballed camera to increase the robustness of visual SLAM was then introduced and an algorithm presented for controlling the gimbal. Simulation results were presented which showed the gimbal is expected to perform in real world conditions, which will be verified in the following chapters. The implementation within the ROS framework was then discussed, with particular regard to how each node interacts with each other during run time.

Chapter 4

Experimental Validation

4.1 Introduction

A gimbal control method to increase the number of visible features was presented in Section 3.2. The algorithm was simulated in Section 3.3 where it was run on the MonoSLAM Glow dataset. A quaternion rotation inverser was introduced to offset the change in orientation and position reported by the visual SLAM algorithm in Section 3.4.1. The first section of this chapter will serve to verify the operation of the gimbal control algorithm and quaternion rotation inverser and to demonstrate correct operation of the system after the modifications have been introduced.

The proposed visual SLAM filter will then be verified through a series of ground and flight tests. The first series of ground tests will be indoors and is intended to demonstrate that the accuracy of Gimballed Multi-Sensor Fusion (GMSF) is greater than the performance of a fixed camera setup in the controlled scenario. The second set of tests will be ground tests in an outdoor setting designed to demonstrate that GMSF is capable of accurately estimating the position, velocity and orientation states in a real-world scenario. During this test the results from the SLAM based MSF will be compared against GNSS as a form of ground truth. Finally, flight tests are conducted to show that the gimballed camera system performs adequately.

Figure 4.1 shows the test rig used for indoor and outdoor ground testing. Specifically, the setup shown is used for stationary testing and gimbal verification. For motion test the cart traverses the blue line. Since the corner detector in PTAM is not able to track the concrete floor it provides a controlled scenario to verify operation of the gimbal controller and rotation inverser. Note that for ground based testing the gimbal and camera are moved further away from the

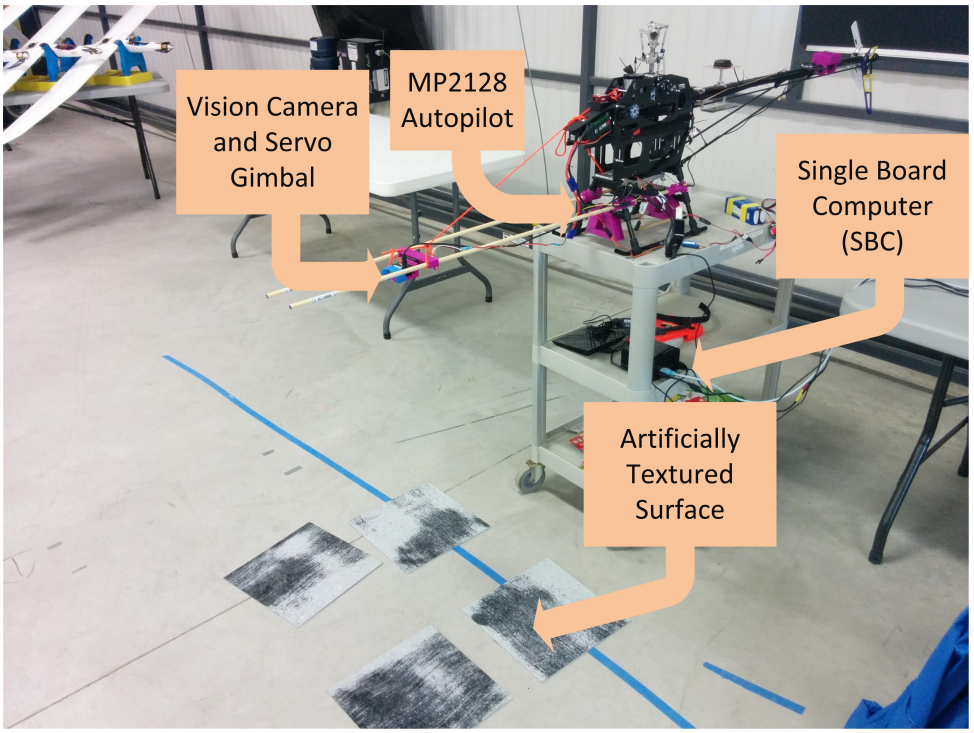


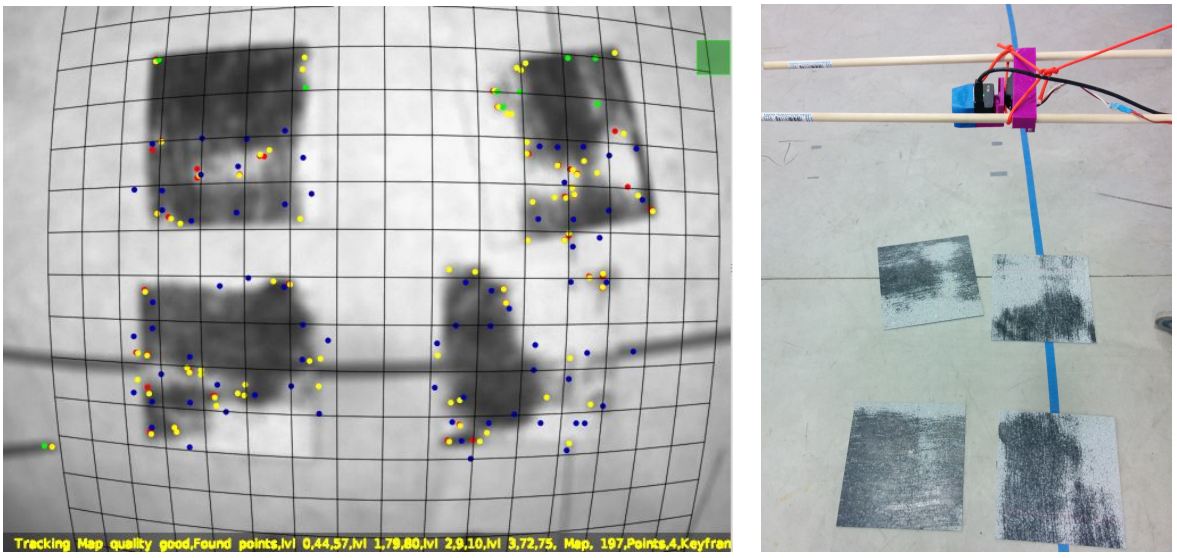
FIGURE 4.1: Indoor and outdoor ground test rig. The gimbal arm was extended so the cart wheels are not visible in the camera frame.

aircraft CoG so that the camera does not view the cart wheels which would corrupt the PTAM estimates. The camera offset is $0.8m$ in the Y direction and is updated in the MSF configuration files. The autopilot is mounted underneath the aircraft's main shaft close to the CoG.

4.2 Control Algorithm and Rotation Inverser Verification

In order to confirm correct operation of both the gimbal control algorithm and rotation inverser a series of control experiments are conducted. The gimbal control algorithm is tested by artificially removing features from a scene and observing the reaction of the camera gimbal. The rotation inverser is evaluated by examining the error in the reported orientation and position between the rotation inverser and the baseline position and orientation during the gimbal actuation. Ideally no deviation is desired, however small variations (sub 3°) are tolerable. Similarly, measured position should remain unchanged, but small (sub $5cm$) variations are permitted. These acceptable errors are well below errors present in other measurement methods such as accelerometer based tilt sensing and GNSS.

Both the gimbal control algorithm and the operation of the rotation inverser are verified using data from a single test. The indoor test environment was falsely textured with floor tiles giving



(A) View from camera after PTAM initialisation during algorithm verification

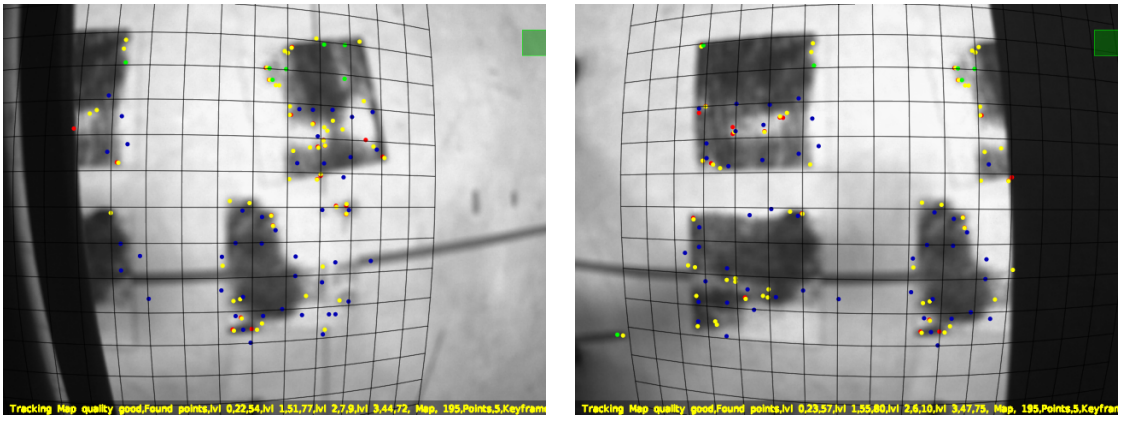
(B) View of world setup during algorithm verification

FIGURE 4.2: Experimental setup used for algorithm verification, and view from PTAM vision camera. Red denotes SBI Level 0, yellow is SBI level 1, green is SBI Level 2 and blue is SBI Level 3.

the camera distinct features to track. PTAM was then initialised with a downward facing camera to view the scene. A view after initialisation from the camera is shown in Figure 4.2a and a view of the test configuration is shown in Figure 4.2b. As seen from the camera in Figure 4.2a the gimbal will rotate the camera to the left and to the right.

The test procedure was as follows:

1. PTAM was initialised using a $10cm$ horizontal translation in the X direction (baseline distance) with the camera facing downwards and the gimbal control algorithm disabled (fig. 4.2a).
2. At $t = 5s$ the gimbal is enabled and allowed to stabilise.
3. At $t = 25s$ the features on the left of the camera's visible area are obscured and replaced by a featureless object (fig. 4.3a).
4. At $t = 35s$ the featureless object is removed from the left portion of the image.
5. At $t = 60s$ the featureless object is introduced on the right edge of the image (fig. 4.3b).
6. At $t = 70s$ the featureless object is removed from the right edge of the image.



(A) View from camera with left features obscured

(B) View from camera with right features obscured

FIGURE 4.3: Camera views while obscuring features. The gimbal control algorithm attempts to rotate the camera away from the obfuscation toward more visible features.

4.2.1 Gimbal Control Algorithm Verification

The gimbal control algorithm described in Section 3.2.2 operates by attempting to keep the weighted feature density equal on both sides of the camera's vision frame by adjusting the angle of a single axis gimbal. As a basic verification test PTAM is initialised over a sufficiently featured region and the gimbal enabled. If the feature density is consistent across the image it is expected the gimbal will not move the camera. By artificially removing features from the cameras visible region the gimbal's behaviour may be observed and thus discern if the algorithm is operating correctly. The gimbal control algorithm is verified using the data collected during the tests outlined in Section 4.2.

Figure 4.3 shows the view from the camera with features obscured. This is in contrast with the image shown in Figure 4.2a where no edges are obscured. It is important to note that the gimbal control algorithm operates on the potentially visible features directly from the corner detector, not the features currently in use by PTAM visible on screen as red, yellow, green and blue dots corresponding to SBI levels 1,2,3 and 4, respectively.

The performance of the gimbal control algorithm is evaluated by plotting the number of detected features compared with the gimbal roll, as shown in Figure 4.4. Features are artificially obscured at $t = 25s$ resulting in a drop from 1120 to approximately 700 features. The gimbal responds with a rotation toward the more feature dense area and by $t = 30s$ there are approximately 900 visible features in the frame. Due to the sparsely featured area around the test region the gimbal controller is unable to obtain the original amount of tracked features however the increase from

700 to 900 marks a 28.5% increase. At $t = 35s$ the obscured features are made visible and there is an almost immediate jump to approximately 1070 features. The gimbal controller then rotates the gimbal back to the original position and the number of features increase to approximately 1120 as before the obfuscation. During the portion of the test where the right hand features were obscured the gimbal is actuated to the left as expected. In this scenario the gimbal does not provide as substantial an improvement due to the fact there were fewer features on the left half of the test field. In fact in this scenario there is a marginal decrease in total number of features near $t = 70s$ due to the fact the control algorithm is trying to balance edge density rather than total number of features.

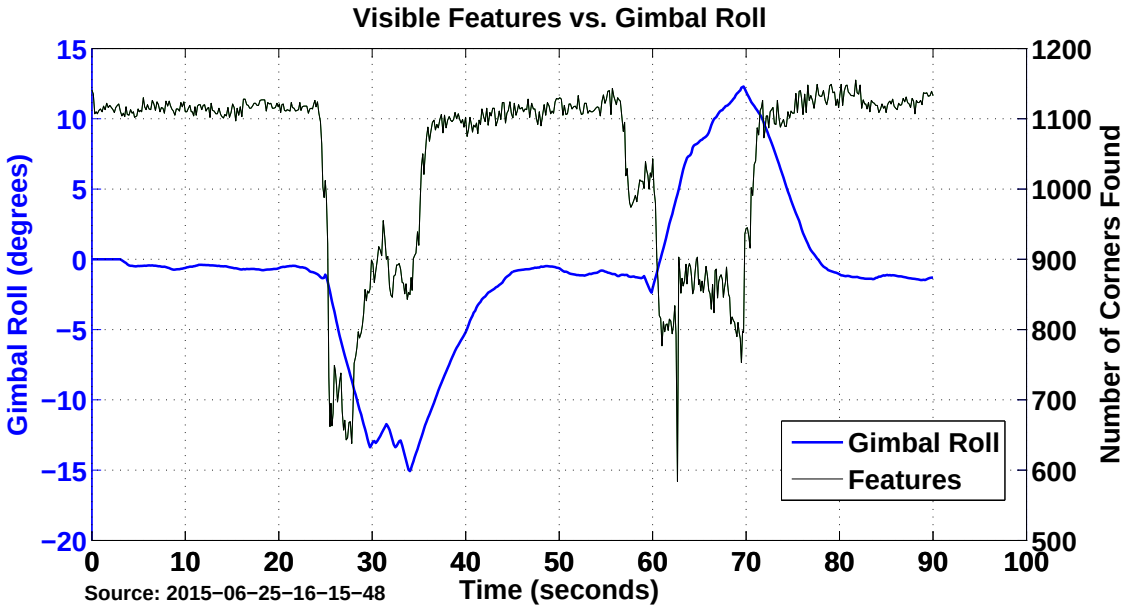


FIGURE 4.4: Visible features with respect to gimbal roll. In general the gimbal controller is able to increase the number of visible points after a portion of the visible features are obfuscated.

While verification of the gimbal control algorithm may be completed by analysing the observed features we must also analyse the ability of the gimbal control algorithm to increase the robustness of the visual SLAM algorithm. While future sections will verify this explicitly through dedicated tests we may make preliminary observations by examining the covariance of PTAM as the gimbal control algorithm operates.

Figure 4.5 shows the covariance reported by PTAM (specifically that reported by bundle adjustment algorithm). In general there is an increase in the covariance when the vision frame is partially obscured. This is expected due to the fact there are fewer features available on which to base the pose estimate. In the first portion of the test from $t = 25s - 45s$ it can be seen that

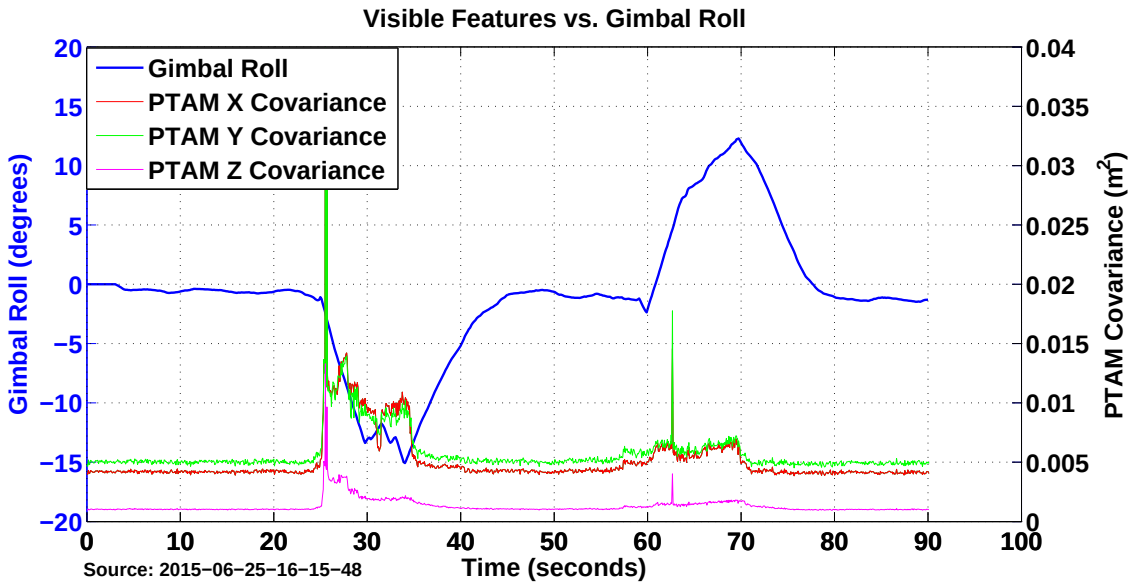


FIGURE 4.5: Visual SLAM covariance with respect to gimbal roll. The initial increase in covariance when features are obscured is mitigated by the gimbal controller.

the gimbal control algorithm is partially able to offset the increase in covariance by rotating the gimbal. However it is important to note that when the features become visible again the covariance is slightly increased from the initial values due to the gimbal roll. The covariances all return to their pre-test values after the gimbal has returned to the steady state.

In the second test the actuation of the gimbal arguably causes a slight covariance increase. This increase is largely negligible (from $0.006m^2$ to $0.007m^2$) and would not adversely affect the overall operation of either PTAM or MSF. This increase may be explained by examining the structure of the gimbal controller and seeing that it does not directly regulate covariance, rather it regulates the amount of visible features in an attempt to minimise the covariance. While these metrics tend to be strongly correlated there are situations such as this where they are not. The rotation of the camera in PTAM naturally will increase the uncertainty of the measurement, especially in this scenario where no extra map features are being added. This theory can be confirmed by examining the total points on the map and the percentage of features found.

Figure 4.6 shows the total features that make up the map and the features found per SBI level compared to gimbal roll. It is clear that while obscuring features from the camera reduces the ability of PTAM to find them it does not affect the total number of features that make up the map. When the gimbal rotates to find more features it is unable to add them to the map due to the fact a pure rotation is being performed. Recalling from Section 2.2.2 that keyframe based SLAM algorithms are unable to observe depth information without camera translation it is clear

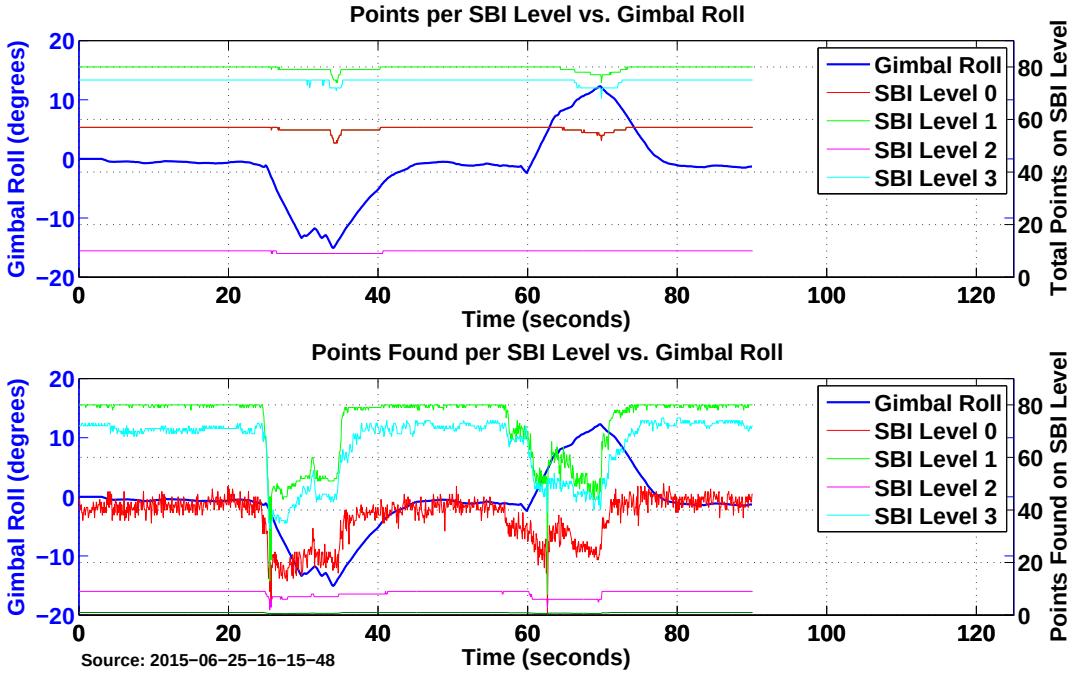


FIGURE 4.6: Total features found on each SBI level and number of features vs. gimbal roll. Note that obscuring map points does not affect the total number of points in the map (top graph) and the gimbal controller is able to increase the number of features found on a given SBI level via rotation.

to see why they are not added. The decrease in covariance during $t = 25 - 40s$ time period can also be strongly correlated with a corresponding increase in map points found. Likewise, the minor increase in covariance seen from the $t = 60 - 80s$ time period can be attributed to a decrease in total number of map points found. Finally, Figure 4.7 shows the fraction of map points found during the test. As anticipated the gimbal controller is able to increase the percentage of map points found in the first test and there is a marginal decrease in the second test, confirming the results from the previous figures.

4.2.2 Rotation Inverser Verification

The gimbal control algorithm is enabled and the gimbal rotation is controlled automatically. Position and orientation reported by the visual SLAM algorithm are observed and comparted to that reported by the rotation inverser. Since the camera is not physically being moved the ideal scenario is no change in the position or orientation as reported by the rotation inverser, however due to small misalignments in the test setup and non ideal camera lens calibration it is expected that there will be small deviations. The data from the tests are examined to ensure

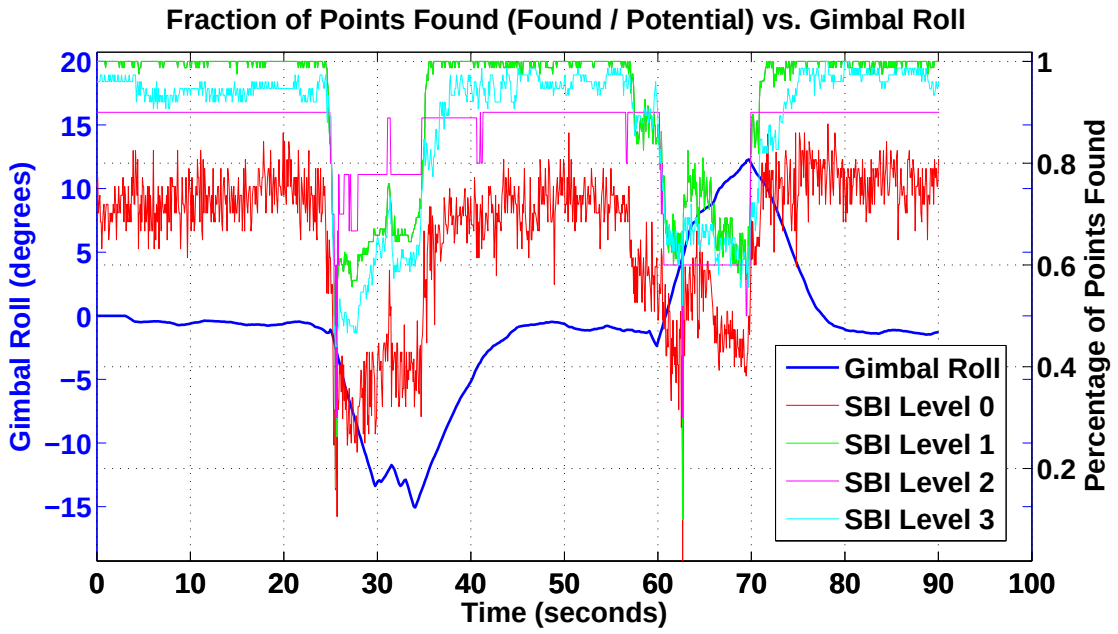


FIGURE 4.7: Fraction of features found vs. gimbal roll. The gimbal controller is able to marginally increase the total fraction of features found and confirm results from previous figures.

that any deviation is sufficiently small as to not disturb the state estimator or SLAM algorithm during future tests.

Quaternion Orientation Inverser Verification

Figure 4.8 shows the performance of the orientation portion of the quaternion rotation inverser. Note that the graph shows the orientation represented as a unit quaternion rotation. While it is not intuitive to interpret orientation as a quaternion it is important to verify the output as the quaternion orientation is the data fed to the Kalman filter.

As expected, the rotation is reported by the visual SLAM algorithm and largely negated by the rotation inverser. For analysis the quaternion output is converted to Tait-Bryan Euler angles and the results are presented in Figure 4.9. Intuitively it is expected that the only angle change should be reported in the roll channel since the gimbal is only acting on the roll axis of the camera. Again, the operation of the rotation inverser is verified by confirming that the unrolled Euler angles only differ in the roll channel of the measurements. Any discrepancies would indicate that the rotation inverser is acting on the pitch or yaw channels which is undesirable.

It is clear from Figure 4.9 that the inverse quaternion operation is being performed correctly. The gimbal rotation is further verified correct by PTAM as minimal amounts of rotation are

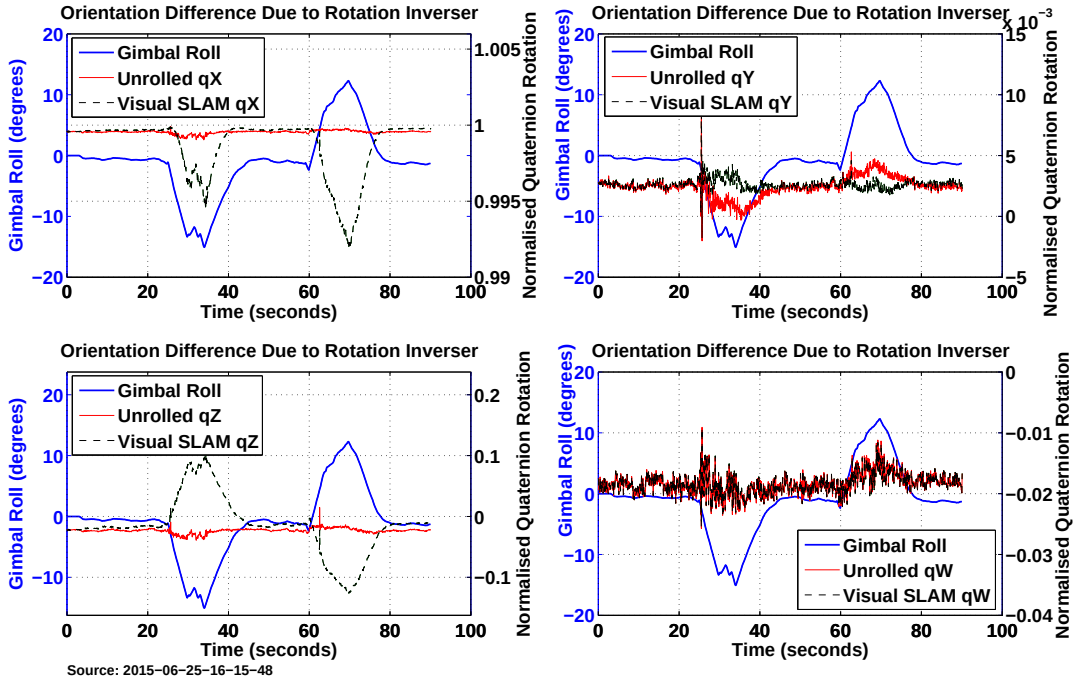


FIGURE 4.8: Operation of quaternion rotation inverser. Note that only q_x and q_z have notable components, implying the rotation largely takes place in the roll channel.

mapped onto the pitch and yaw axis, implying the camera is aligned correctly with the gimbal frame. For further analysis the error between the initial orientation estimate and the rotation reported by the rotation inverser are plotted. Since the test setup was stationary during the test it can be assumed that any difference in orientation is a direct result of gimbal movement.

Figure 4.10 shows the error between the initial Euler angles and those measured during the course of the test. Pitch and yaw measurement error during gimbal actuation are largely negligible with pitch and yaw errors having a mean of 0.017° and 0.099° with a maximum of 0.81° of 0.95°, respectively. When considering the standard deviation of the yaw channel is 0.207° it becomes clear that the impact of the gimbal is minimal. The roll channel shows the largest error mean of 0.211° and a maximum of 4.142°. The roll channel error shows a strong correlation with gimbal actuation which may imply the rotation inverser is not entirely removing the rotation commanded by the gimbal. To further investigate this premise the commanded gimbal roll is compared to the roll measured by PTAM, shown in Figure 4.11.

The initial gimbal roll is subtracted from the PTAM roll and compare how the error propagates over the course of the test. When the gimbal is initially actuated at $t = 25s$ it can be seen that the error begins to increase to approximately 2.5°. While this is a notable deviation it

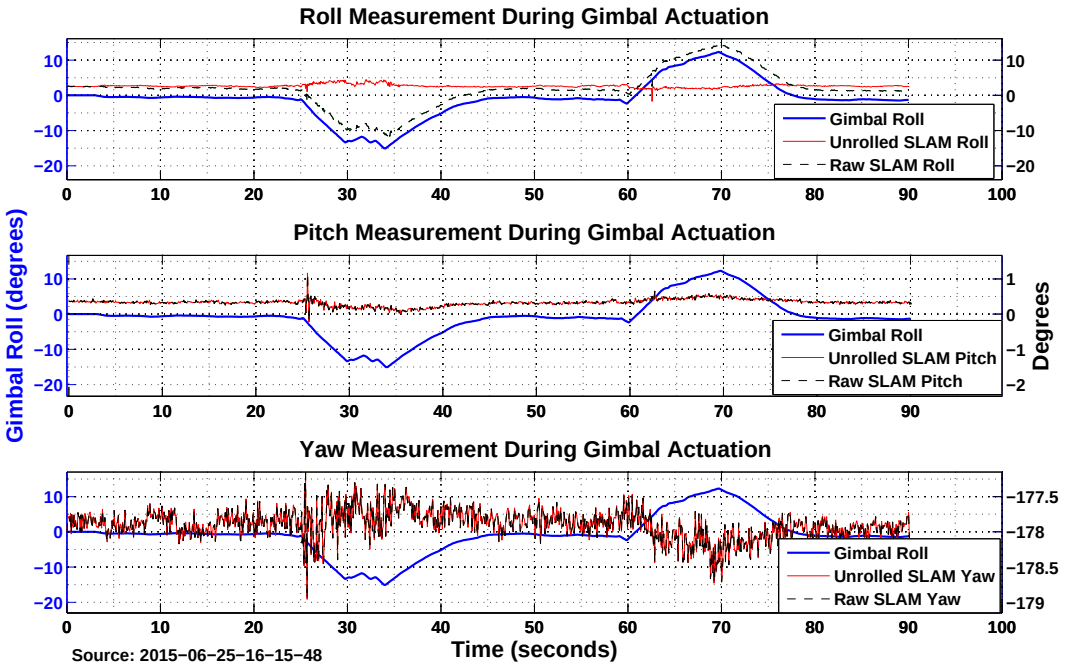


FIGURE 4.9: Operation of quaternion rotation inverser presented in degrees. Once again only the roll channel reports a significant change.

is noted that once the features are no longer obfuscated at $t = 35s$ the roll error dramatically decreases. This implies that the error is caused by error from PTAM rather than error from the rotation inverser. Furthermore the opposite actuation from $t = 60s - 80s$ shows significantly less error - if the error was due to the rotation inverser it is expected to be symmetric about the axis of rotation.. Regarding Figure 4.4 from Section 4.2.1 it can be seen that in the first test case obscuring the features on the left side of the frame results in a drastically reduced visible set of points. Since only points on the right portion of the frame would be visible PTAM must calculate the roll estimate from points with a much shorter baseline, resulting in higher variance. No such problems occur for the pitch and yaw channels since we are not reducing the observability in those directions. Finally, the higher error in this channel overall may be explained by the fact that the servo output jumps in $0.5\mu sec$ steps, corresponding to a potential $0.5°$ jump in position during each gimbal update.

Position Inverser Verification

Since PTAM reports visual SLAM location as the position of an arbitrary world frame with respect to the camera frame the effect of the gimbal rotation must also be removed from the

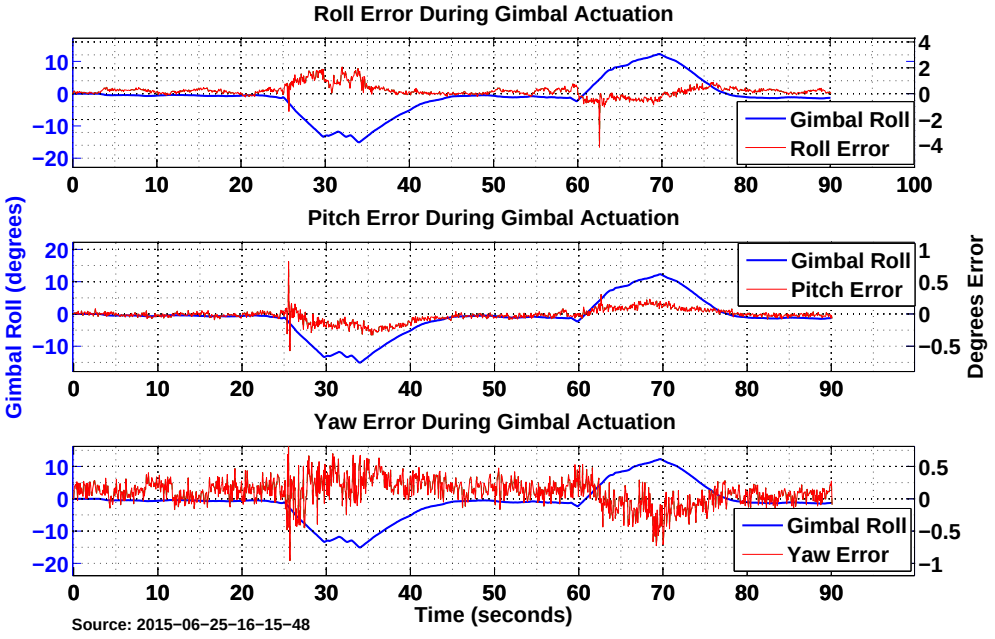


FIGURE 4.10: Error between initial orientation measurement and measurements during test. In general the largest roll error trend is 2.5°, with pitch and yaw being largely unaffected.

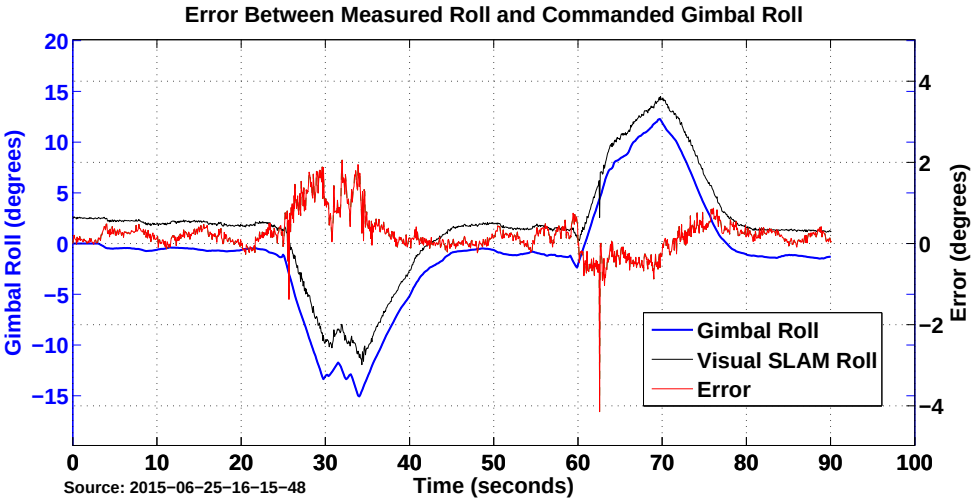


FIGURE 4.11: Error between commanded gimbal roll and roll measured by PTAM. Note that the error is largely correlated with the obfuscation of features rather than the gimbal roll, implying accurate operation of the rotation inverser.

position reported by PTAM. Figure 4.12 shows how position reported by PTAM varies during gimbal actuation despite the camera remaining stationary. The rotation inverser is largely able to remove the effect of the rotation on the reported position. Once more it is noted that the error in position is more strongly correlated with the obfuscation of features rather than gimbal roll, implying the error is due to the lack of visible features.

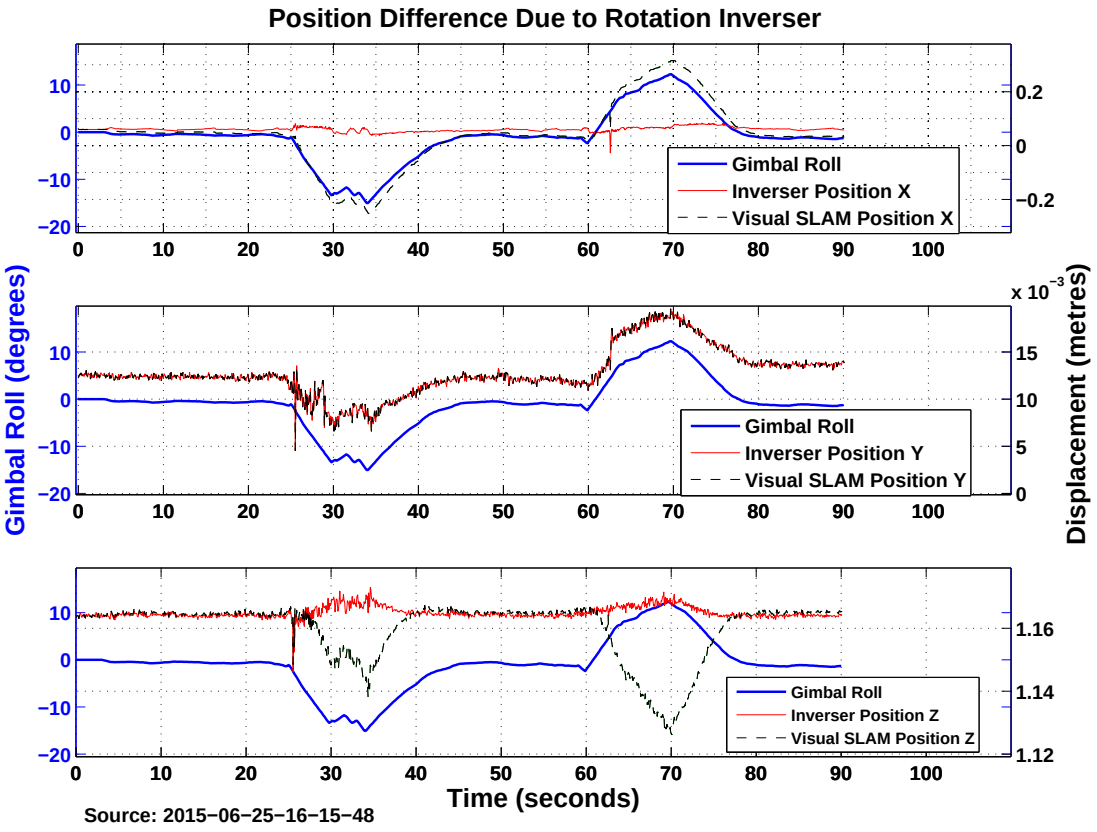


FIGURE 4.12: Position variation during gimbal actuation. Note in general the position remains unchanged and the rotation inverser acts only in roll.

Figure 4.12 is used as a means to further verify the correct quaternion operation is performed on the input data. Since the yaw is roughly 180° there should not be a significant change in the Y position, which is reflected in the data in the plots. In general, the inverse operation cancels out the change in X and Y position as expected. Error is defined as the offset from the initial position and the results are presented in Figure 4.13.

The largest errors are $8cm$, $0.78cm$ and $0.97cm$ for X, Y, and Z position, respectively. If outlying spike at sample $933, t \approx 63s$ is removed the largest error on X displacement is a more reasonable $2.26cm$. Once more, the X position estimate is expected to vary the most due to the fact removing the points from the left and right edges of the image degrades the observability in the X axis. The mean of the errors are $0.00021cm$, $0.00062cm$ and $0.17cm$ for X, Y, and Z with a standard deviation of $0.0082\sqrt{cm}$, $0.0026\sqrt{cm}$ and $0.0019\sqrt{cm}$ respectively.

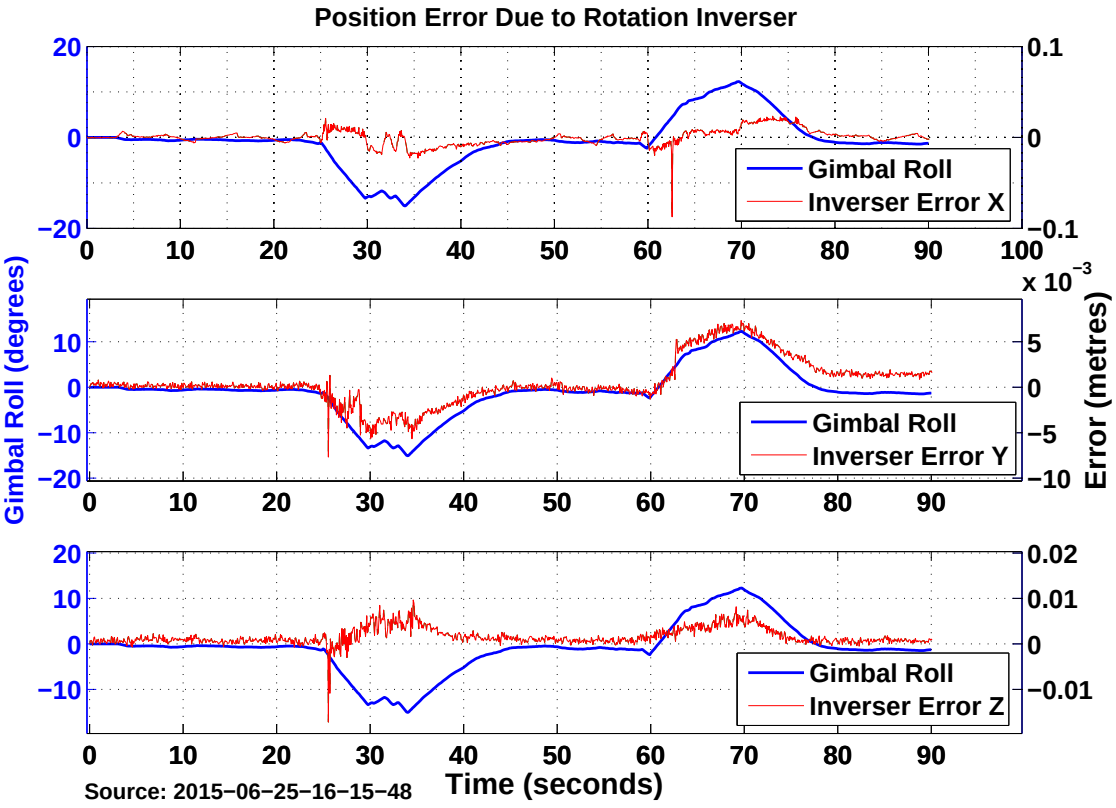


FIGURE 4.13: Position error during gimbal actuation. As expected the largest error is in the X channel and is 8cm, with the other channels having sub centimetre error.

While the error increases during gimbal actuation the net effect is minimal. Coupled with the method MSF uses of incorporating PTAM position measurement covariance in a relative fashion [21] the net effect on the state estimates will be negligible.

4.2.3 Conclusion

The gimbal control algorithm described in Section 3.2 was implemented as a series of ROS nodes and verified through experimental testing. The tests show that the gimbal pointing algorithm works as expected and is able to communicate with the other nodes and autopilot and control the servo gimbal. The control law was verified in a controlled environment and reacts as expected. Both the orientation and position portion of the rotation inverser were investigated and found to work as expected. The tests also showed that the modifications made to PTAM do not affect its overall function. Minor additional uncertainty and position noise is added to the pose measurements from the visual SLAM algorithm however the stability of PTAM remained unchanged.

4.3 Ground Based Testing

4.3.1 Introduction

For preliminary verification of algorithm viability a series of ground based tests are preformed. Indoor tests represent a controlled scenario where the scene observed by the camera can be carefully managed for the presence or absence of trackable features. This allows the construction of an environment where the advantages of having a gimballed camera may be maximised. Outdoor ground based tests allow verification of the framework functionality in a real-world environment. A drawback of ground based testing is that the camera is closer to the observed scene and thus the relative distance travelled by the features (in terms of pixels per frame) is significantly higher than flight at altitude, creating a difficult tracking scenario. However this has the advantage of creating a more strenuous test situation, meaning if the algorithm is operational during ground testing it is more likely to be functional during flight verification.

Two indoor test cases are considered: the first test consists of enabling the gimbal while the UAS is partially over a previously mapped region. The gimbal control algorithm should rotate the camera towards the previously mapped region, recovering map points and increasing estimation reliability. Note that allowing the camera to rotate while stationary only works for previously mapped areas (such as a hovering helicopter or quadrotor) since no new features are able to be added during this pure camera rotation. The second test consists of moving the UAS along a single axis where the frame seen by the camera is only partially filled with trackable features, where we expect the gimbal controller to rotate the camera towards the more trackable features.

4.3.2 Stationary Testing

A test region was laid out as shown in Figure 4.14. After initialisation the test setup is moved around in a square pattern to create a map of the surrounding area prior to beginning the test. The camera is then positioned in such a way that half of the image is of the featureless region which contains minimal trackable points. The other half of the frame contains map points from the map building portion of this test. The gimbal controller is then enabled and allowed to rotate the gimbal according to the relative feature density seen in the image. Two tests are conducted to ensure repeatable results.

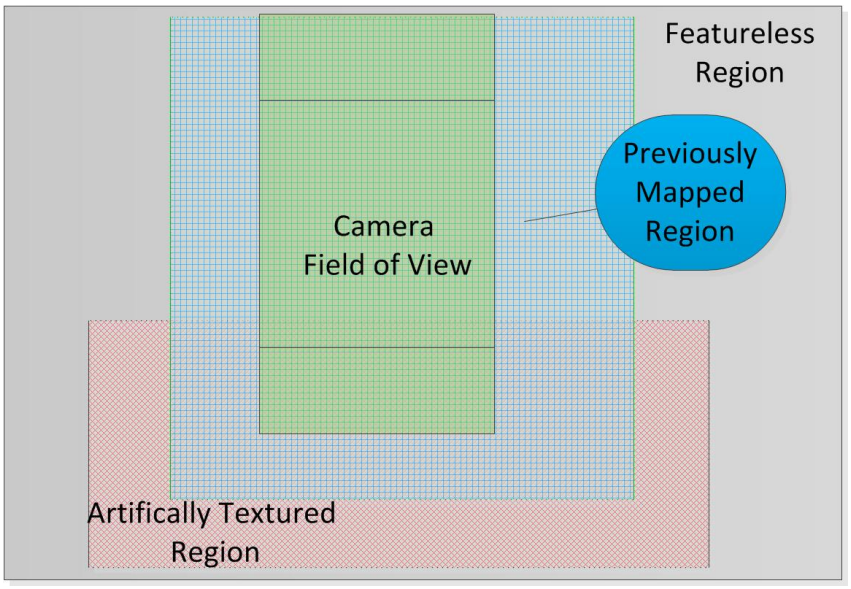
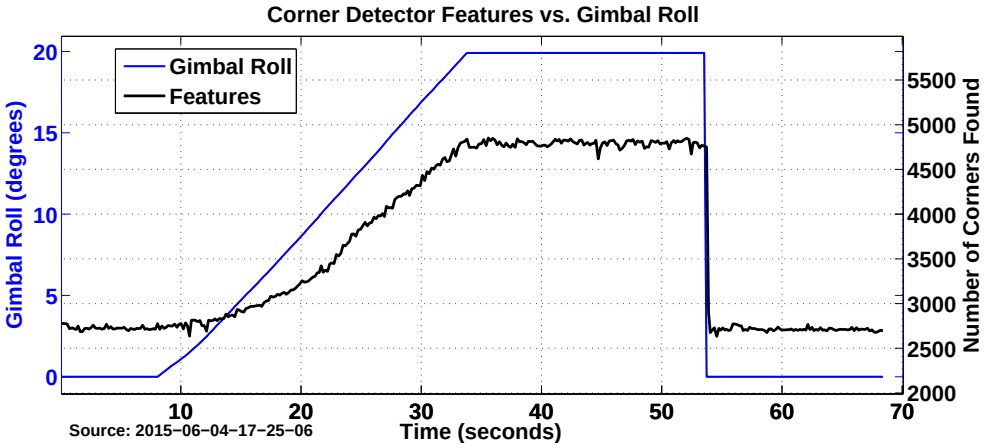


FIGURE 4.14: Stationary indoor test scenario

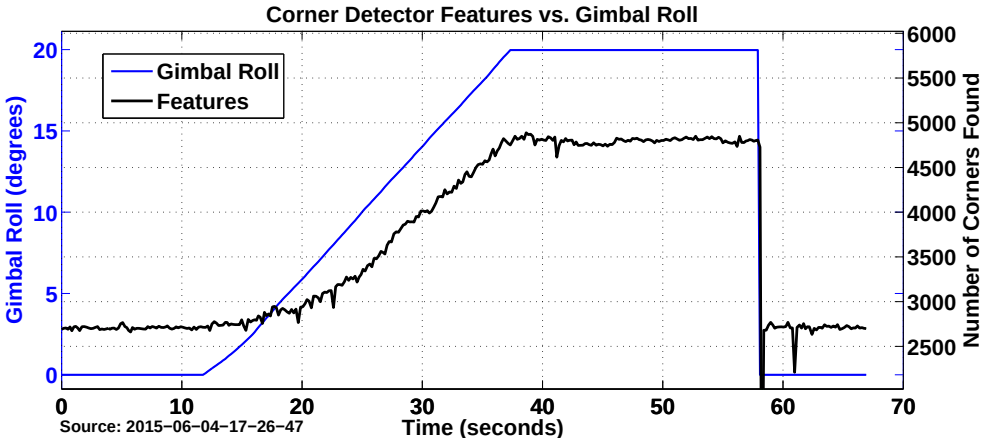
Several metrics are considered when evaluating the viability of the gimbal control algorithm to improve the stability of PTAM. First, the effect of gimbal roll on extracted features from the corner detection algorithm are presented for the stationary test in Figure 4.15. It is important to note that this metric will not directly measure the performance affect on the visual SLAM algorithm due to the fact that the measure is merely points detected by the corner detection algorithm (in this case FAST16), not of features actually being tracked by the VSLAM algorithm itself. Nonetheless it is an important metric as without sufficient points presented to the SLAM algorithm reliable tracking will not be achieved.

In both tests there is a strong correlation between gimbal roll and the number of features detected by the corner detection algorithm. In this carefully constructed scenario both tests start off detecting approximately 2750 features. As the gimbal roll increases to its maximum there is a corresponding increase in the number of points tracked, topping out at approximately 4800 features, an increase of 74.5%. While this is a controlled scenario it does serve to highlight the fact that the gimbal control algorithm is capable of providing substantial increases to the amount of features detected by the corner detection algorithm which are ultimately presented to the VSLAM algorithm as candidates for addition to the map.

It is also important to consider how the gimbal roll affects the accuracy of the visual SLAM algorithm. As a second metric the covariance measurement reported by the bundle adjustment algorithm contained within PTAM is considerd. The covariance reported by PTAM is used directly by MSF, so a decrease in visual SLAM covariance will result in a corresponding decrement



(A) Features extracted from FAST16 corner detector during test 2015-06-04-17-25-06

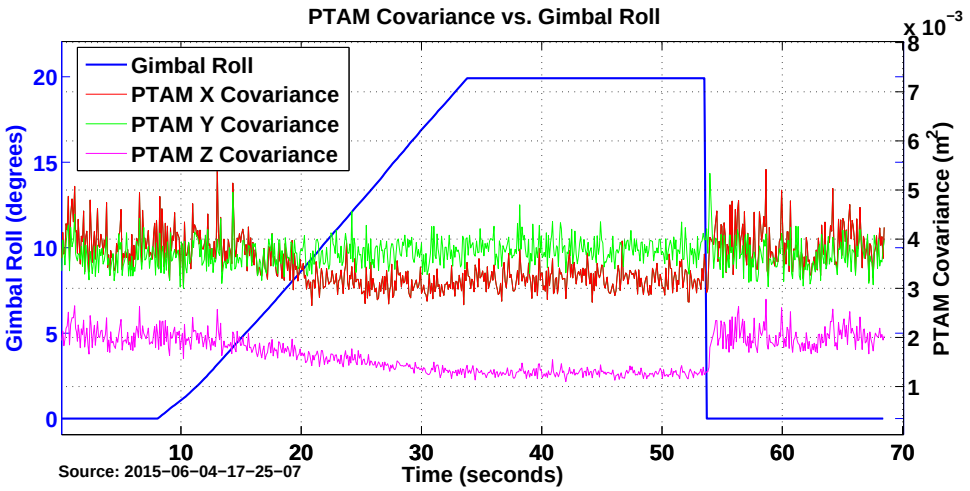


(B) Features extracted from FAST16 corner detector during test 2015-06-04-17-26-47

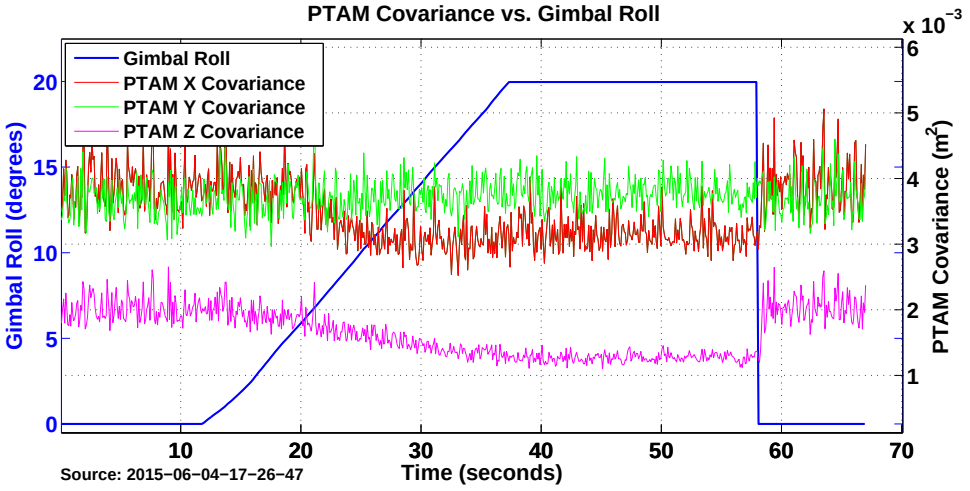
FIGURE 4.15: Features detected by gimbal control algorithm during controlled stationary scenario. Note that there is a strong correlation between visible features and gimbal roll

in the covariance reported by the Kalman filter. Figure Figure 4.16 shows how the covariance varies during the gimbal actuation.

From the plot it is clear to see that there is a strong correlation between gimbal actuation and PTAM covariance. In general there is a noted decrease in covariance during gimbal actuation, implying an increase in the overall accuracy of the VSLAM system. Covariance in the visual Z direction decreases from $2 \times 10^{-3}m^2$ to $1.2 \times 10^{-3}m^2$ while X covariance decreases from $4 \times 10^{-3}m^2$ to $3 \times 10^{-3}m^2$. This corresponds to a decrease of 40% and 25%, respectively. The Y covariance remains relatively unaffected by gimbal movement. This is easily explained by examining the high level observability of the system. With the downward facing camera there are points along the left half frame running in a top to bottom fashion. The top to bottom of the camera frame corresponds to the Y axis of PTAM and the left-to-right axis corresponds to PTAM's X axis. At the beginning of the test there are many top-to-bottom points allowing for



(A) Covariance during stationary test 2015-06-04-17-25-06

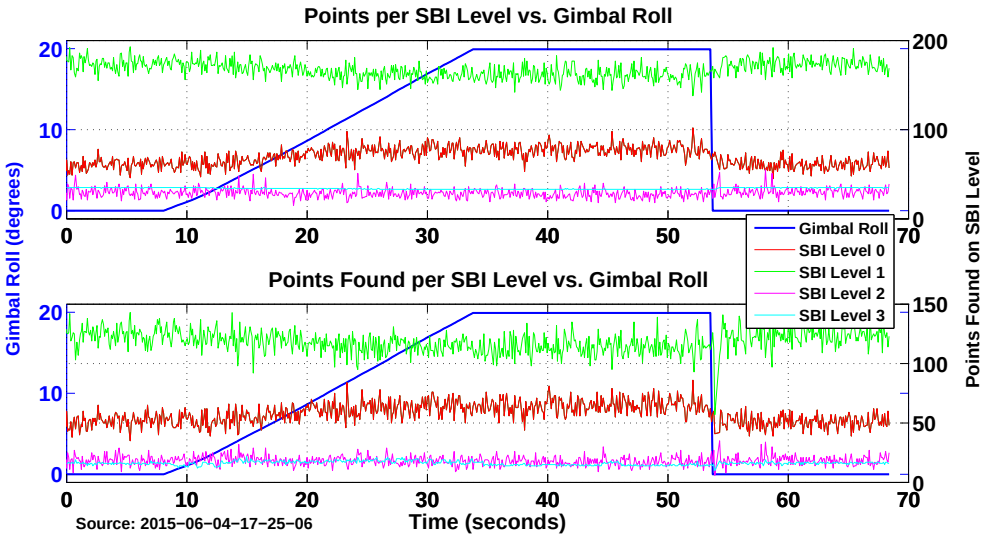


(B) Covariance during stationary test 2015-06-04-17-26-47

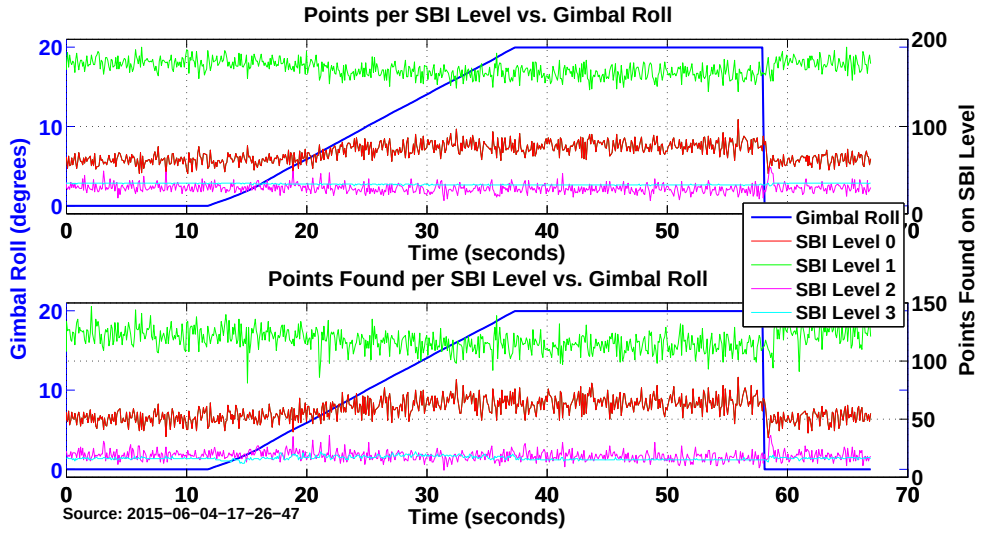
FIGURE 4.16: Covariance plots for tests in controlled stationary indoor scenario. There is a general trend of decreasing covariance during gimbal actuation.

a good estimate of the Y position and orientation whereas there are relatively few left-to-right points. After the rotation of the gimbal there is an increase in left-to-right points allowing an overall increase in the system accuracy. Similarly, Z covariance decreases because there are more total points allowing for a more accurate relative distance measurement to be made about the Z axis. There is also note a small decrease in the standard deviation of the Z covariance during the time when the gimbal is rolled.

The total number of and number of found features on each SBI level are examined. This is an important metric since the features are what PTAMs tracking algorithm actually uses to find its location. It is possible that arbitrary rotations of the gimbal may cause loss of tracking, due to the gimbal rotating to featureless or unmapped regions. Thus, it must be verified that the gimbal control algorithm does not cause a decrease in the total number of observed points. If the



(A) Tracked feature information during stationary test 2015-06-04-17-25-06. Top is total points on a given SBI level and bottom is found points on a given SBI level.

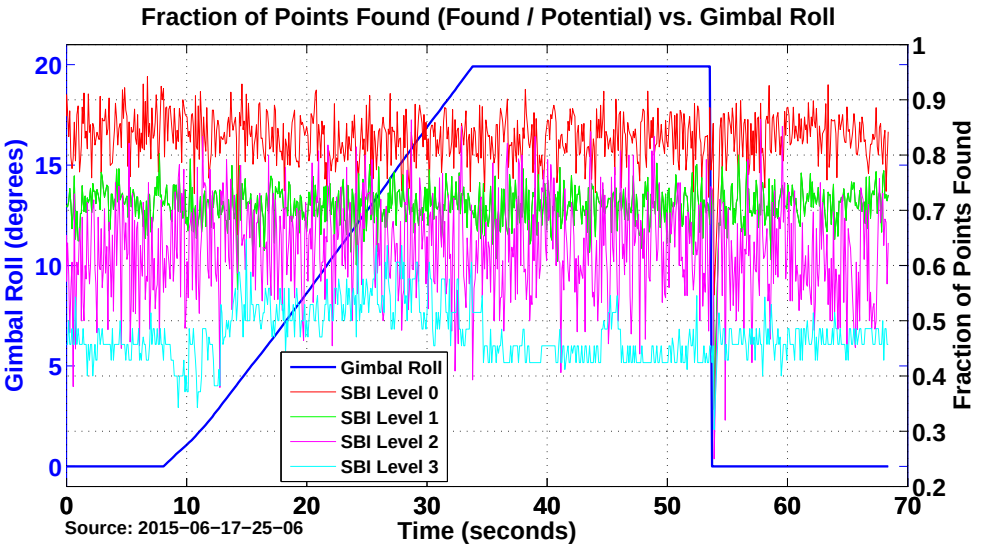


(B) Tracked feature information during stationary test 2015-06-04-17-26-47. Top is total points on a given SBI level and bottom is found points on a given SBI level.

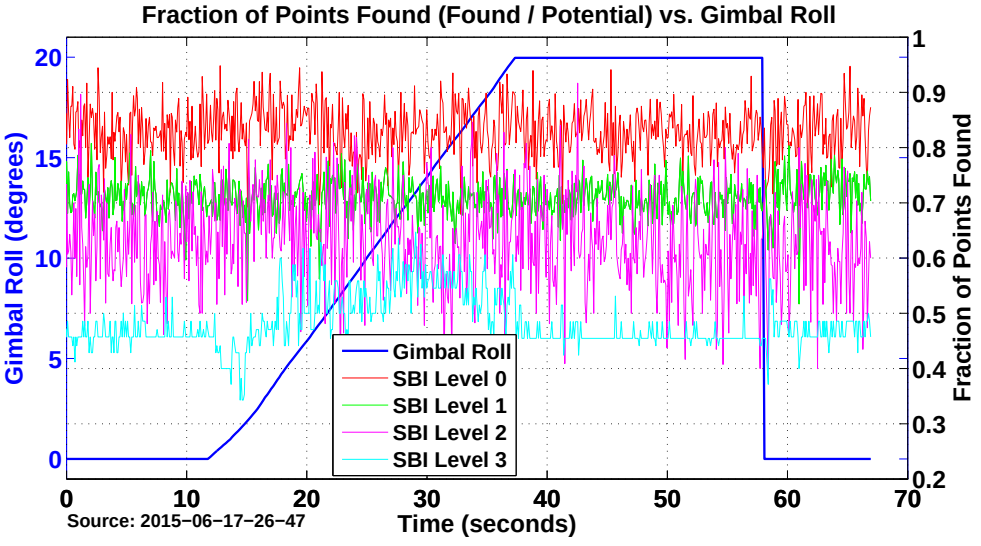
FIGURE 4.17: Total number of features and found number of features on SBI levels during gimbal actuation

gimbal control algorithm causes a loss of tracked points it may lead to divergence of the filter. Since the gimbal control algorithm operates on the features tracked by the corner detection algorithm and not the points tracked by PTAM, the gimbal controller may indeed increase the total number of features in the frame while at the same time causing tracking loss in PTAM. Results from this test are presented in Figure 4.17. Recalling that no depth information can be extracted from features found by the corner detector during a pure rotation it is clear why the number of points tracked by PTAM does not increase – no new keyframes can be added to the map. More importantly, there is no notable decrease in the number of points tracked during

gimbal actuation, showing that gimbal motion does not appear to be destabilising PTAM. This is also an important observation as it shows that the camera may be gimballed while the UAS is stationary, but only if it will be observing a previously mapped area. It is important to limit prevent the camera from rotating to unexplored regions during stationary periods since PTAM will not be able to add any map points in the newly visible region.



(A) Fraction of tracked features found during stationary test 2015-06-04-17-25-06



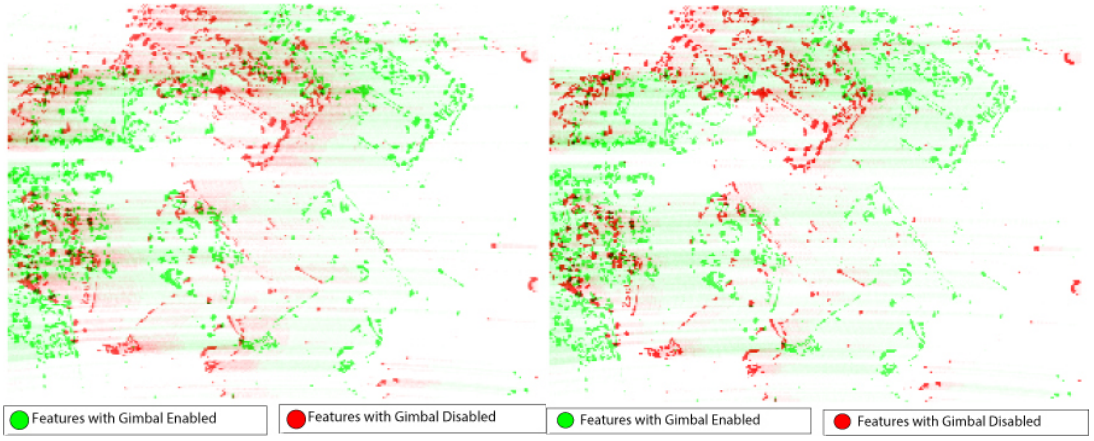
(B) Fraction of tracked features found during stationary test 2015-06-04-17-26-47

FIGURE 4.18: Fraction of tracked features for indoor stationary test. There is minimal change in fraction of points found due to lack of translational motion

The fraction of points found on each level should remain constant or increase, shown in Figure 4.18. Results are similar to those from Figure 4.17, there is no marked increase or decrease in the fraction of points found due to gimbal actuation. Once more, this implies that the actuation of the gimbal does not disturb the stability of the system when stationary providing that

the area being observed has previously been mapped.

As a visual method of examining how the gimbal affects feature density during the experiments it is possible to plot the feature locations before and after the gimbal is enabled, shown in Figure 4.19. This image represents feature density spread across the whole test. Features locations marked in red are those found with the gimbal disabled and those found when the gimbal is enabled are marked in green. Note the streaking of the features as the gimbal rotates from one orientation to another.



(A) Fraction of tracked features found during stationary test 2015-06-04-17-25-06

(B) Fraction of tracked features found during stationary test 2015-06-04-17-26-47

FIGURE 4.19: Feature propagation as gimbal is enabled during controlled indoor stationary test. Red denotes feature locations prior to the gimbal being enabled, green denotes after. Note the general trend that green features are more spread out across the frame than red features.

Figure 4.19 may be used as a method of visually verifying the operation of the gimbal control algorithm during the test. At the beginning of the test the red points tend to cluster in the left half of the image, leading to a scenario where only half of the vision frame is receiving useful data. Compare with the latter portion of the test where the points marked in green tend to be more evenly spread throughout the vision frame, yielding not only more features detected but also a lower covariance estimate and a more robust visual SLAM system overall. This feature density balancing is an inherent part of the gimbal control algorithm and allows the VSLAM filter to be presented with the largest dataset possible.

4.3.3 Indoor single axis testing

The scenario presented in Figure 4.20 is investigated. Recall with the fixed downward facing camera only half of the frame contains trackable features as the concrete floor does not provide

any suitable corners to track. The artificially textured region consists of floor tiles arranged to mimic a typical outdoor scene with randomly located trackable featured.

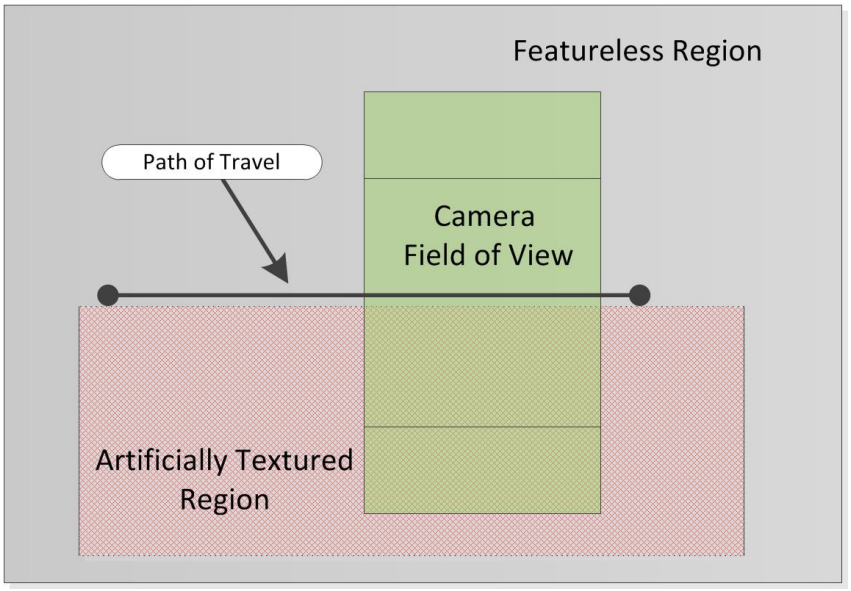


FIGURE 4.20: Indoor single axis test scenario experimental setup. Note that only half of the camera FoV will contain trackable features if looking downward

Due to indoor space limitations the path of travel is limited to two metres in length, however for proof-of-concept testing this is sufficient to demonstrate the operation of the gimbal and gimbal control algorithm. Once more, the covariance reported by the visual SLAM algorithm is examined as a metric of the estimation accuracy. It is important to note that while the covariance from PTAM can be compared during a single run it is difficult to compare across multiple runs due to the inherent scale changes. Thus, covariance is compared during the same run,with two traversals of the path prior to enabling the gimbal and two traversals after enabling the gimbal. This test is repeated test multiple times to ensure accuracy. Scale discrepancies can be alleviated by using the output of MSF however during the indoor testing over marginally trackable regions the filter suffers from instability which is clearly visible in all three test results. This instability affects the covariance propagation and ultimately leads to inaccurate state estimates. The instability also effects the abilty of MSF to obtain an accurate scale factor, thus only the PTAM outputs are examined. The cause of these instabilities is unknown however they only occur over marginally trackable regions.

The tests were performed as follows:

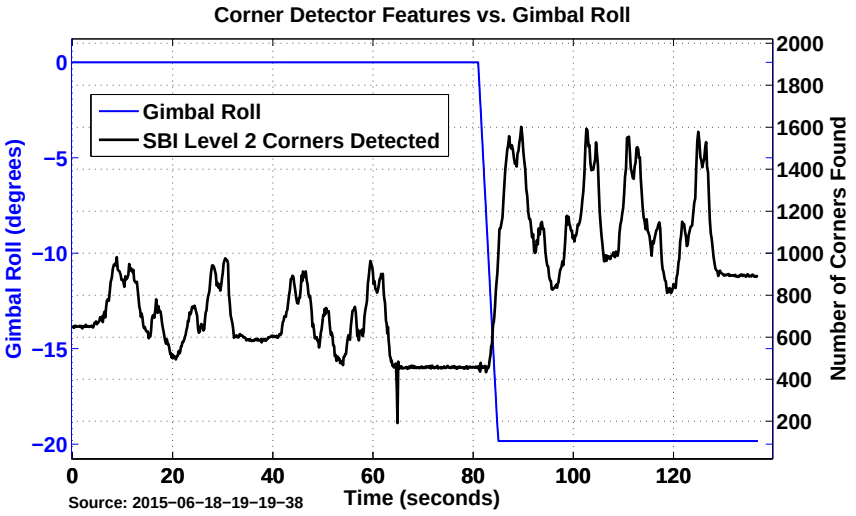
1. PTAM was initialised using a $10cm$ baseline distance with the camera facing downwards and the gimbal control algorithm disabled.

2. MSF is enabled and the gyro bias estimates are allowed to stabilise.
3. The test setup is moved forward in the Y direction at an approximate constant velocity for two metres, then returned to the origin.
4. Step 3 is repeated.
5. The gimbal control algorithm is enabled and the setup is once again translated $2m$ in the positive Y direction and returned to the origin
6. Step 5 is repeated. When the setup is returned to the origin the test is concluded.

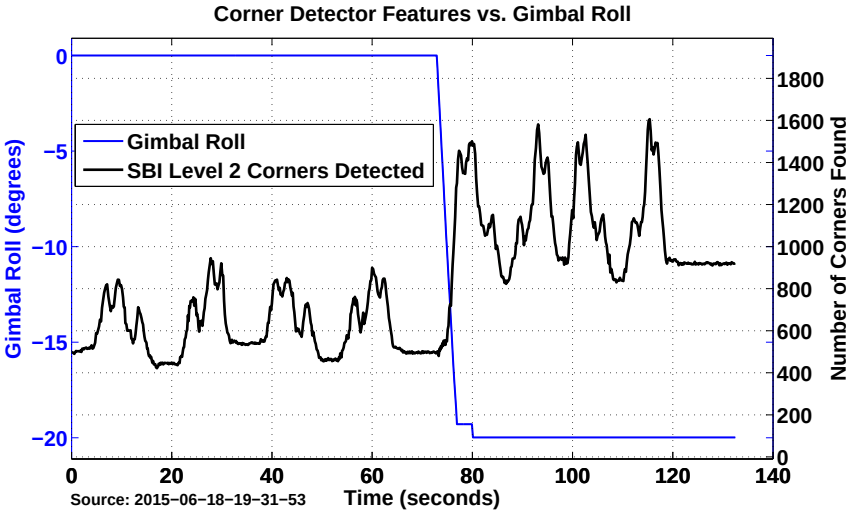
Three iterations of the above test are presented below to demonstrate repeatability. As with the stationary test we first examine the effect the gimbal roll has on the detected image corners, shown in Figure 4.21.

Similar to the stationary test shown in Figure 4.15 there is a substantial improvement in the number of corners detected by the FAST16 algorithm. It is noted that in this test the gimbal control algorithm was done using points on Level 2 of PTAM's SBI model in comparison to the tests conducted in Section 4.3.2 which accounts for the large discrepancy in detected number of points. In each test there is an increase from an average of approximately 700 points to roughly 1200 points, an increase of 71%. The standard deviation is also increased; however, the minimum detected number of points with the gimbal enabled tends to be above the maximum points found with the gimbal disabled. The covariance measurement from PTAM is also examined to verify that the increase in points fed to the filter do not corrupt the estimate and see how the accuracy is affected. Figures 4.22 to 4.24 show results for the completed tests.

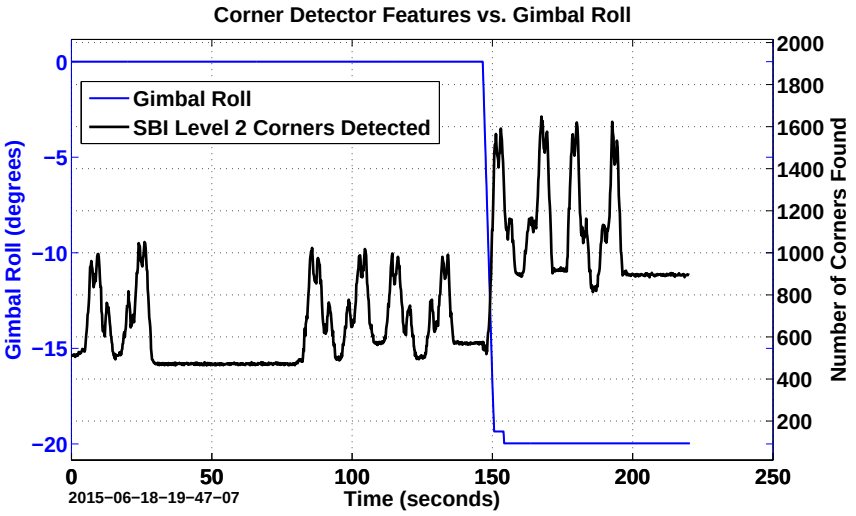
In every test conducted there is a substantial decrease in the covariance reported by PTAM as a result of the gimbal movement. Additionally, note the standard deviation of the covariance also decreases dramatically as a result of the increased number of visible features. The aforementioned instability of MSF can clearly be seen, particularly in Figure 4.22 prior to the gimbal being enabled. Instabilities such as these render the filter unusable for flight and would result in a loss of mission or vehicle, so it is important that they be mitigated. In all cases the overall map size was also increased, yielding a more stable position estimate and correspondingly lower covariance. There is particular increase in features located on SBI Level 1. These lower level features are used in the fine tracking stage and having more low level features allows a more precise pose estimate to be measured. There is also an improvement in Level 3 map points



(A) Gimbal Roll vs. Detected Features for Test 2015-06-18-19-19-38



(B) Gimbal Roll vs. Detected Features for Test 2015-06-18-19-31-53



(C) Gimbal Roll vs. Detected Features for Test 2015-06-18-19-47-07

FIGURE 4.21: Single axis movement test showing effect of gimbal roll on detected features. Enabling the gimbal provides a notable increase in the number of visible features.

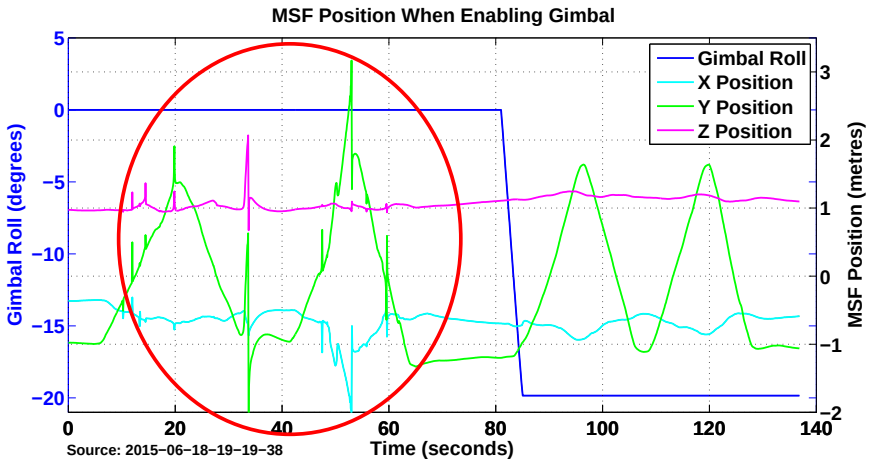


FIGURE 4.22: Position estimates from PTAM during test 2015-06-18-19-19-38. Tracking discontinuities are present inside the red encircled area and represent instabilities in the position estimate from MSF. These large jumps would render the filter unusable for navigation. Note these discontinuities are no longer present when the gimbal is enabled.

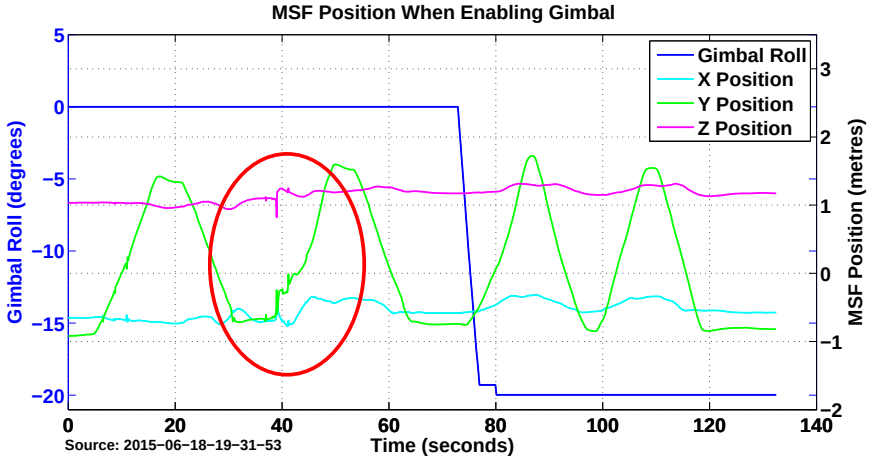


FIGURE 4.23: Position estimates from PTAM during test 2015-06-18-19-31-53. There are position discontinuities of approximately $0.4m$ in the red encircled area, which are alleviated when the gimbal is enabled.

which are vital to the coarse tracking stage and provide more robust position estimates during rapid manoeuvres. High level map points are critical to maintaining a robust map in PTAM. While the algorithm will work satisfactorily with sparse low level map points, lack of high level map points means the coarse tracking stage is inaccurate and no lower level points will be found, resulting in lost tracking.

Data from the tests are analysed and presented below to provide a quantitative measurement of gimbal performance.

From the data presented in Tables 4.1 to 4.3 there is clearly evidence that allowing the camera to gimbal in this scenario provides a substantial decrease in the reported covariance of the system.

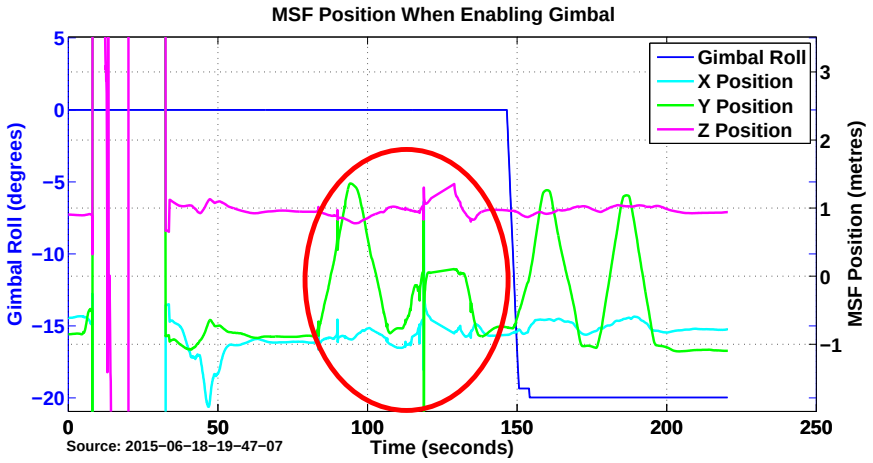


FIGURE 4.24: Position estimates from PTAM during test 2015-06-18-19-47-07. The red encircled area shows discontinuities in the position estimate and also a period of tracking loss, where there is only 1m translation in Y instead of 2m. Once again note these discontinuities are no longer present when the gimbal is enabled.

Run 06-18-19-19			
	Gimbal Disabled	Gimbal Enabled	Improvement
Avg. Cov. X	0.00414	0.00174	57.8%
Avg. Cov. Y	0.00436	0.00174	50.0%
Avg. Cov. Z	0.00215	0.00174	34.9%
Std. Dev. X	0.00118	0.00066	43.9%
Std. Dev. Y	0.00152	0.00083	45.2%
Std. Dev. Z	0.00073	0.00027	63.2%

TABLE 4.1: Average and Standard Deviation for Covariance During Run 06-18-19-19

Run 06-18-19-31			
	Gimbal Disabled	Gimbal Enabled	Improvement
Avg. Cov. X	0.00917	0.00349	62.0%
Avg. Cov. Y	0.01054	0.00425	59.7%
Avg. Cov. Z	0.00528	0.00296	43.8%
Std. Dev. X	0.00276	0.00092	66.5%
Std. Dev. Y	0.00400	0.00112	71.8%
Std. Dev. Z	0.00179	0.00065	63.5%

TABLE 4.2: Average and Standard Deviation for Covariance During Run 06-18-19-31

Run 06-18-19-47			
	Gimbal Disabled	Gimbal Enabled	Improvement
Avg. Cov. X	0.00388	0.00182	53.0%
Avg. Cov. Y	0.00400	0.00174	49.3%
Avg. Cov. Z	0.00203	0.00130	36.0%
Std. Dev. X	0.00091	0.00058	36.2%
Std. Dev. Y	0.00129	0.00056	56.5%
Std. Dev. Z	0.00055	0.00041	24.2%

TABLE 4.3: Average and Standard Deviation for Covariance During Run 06-18-19-47

This in turn allows the EKF to provide a more confident state estimate of the overall system. On a practical note it is clear from Figures 4.22 to 4.24 that the spiking instabilities of MSF disappear when the gimbal is enabled, allowing the filter to function in correctly in a scenario where it previously was unable to provide correct state estimates. While in this particular setting the system would be able to function with a fixed camera mounted in a different orientation, the ability to transition to a more accurate downward looking camera when flying at altitude may negate any gained benefit. It can also be seen from Figure 4.21 that enabling the gimbal results in more map points being detected at the SBI level. There is also a corresponding increase in total map points that is strongly correlated with activating the gimbal.

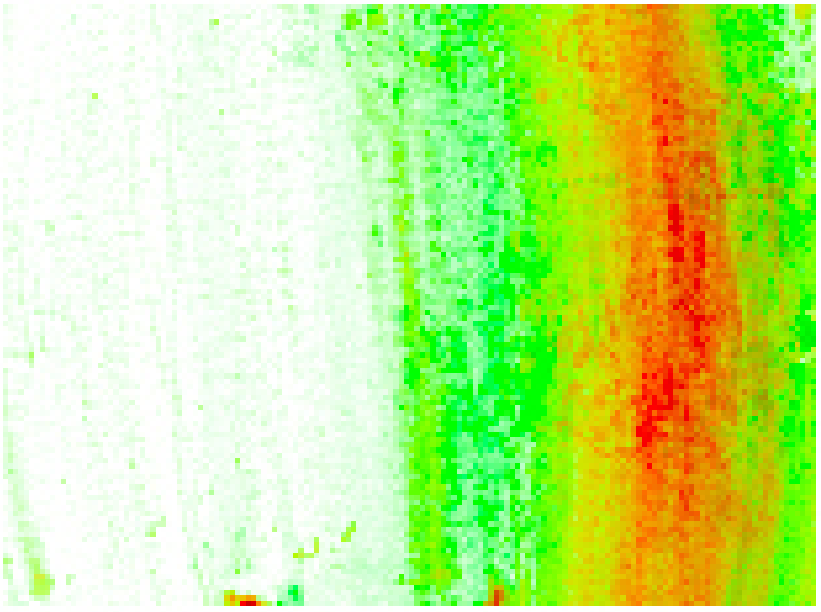


FIGURE 4.25: Feature density map of indoor ground test. Note the trend towards a more evenly distributed feature set (green) with the gimbal enabled compared to the beginning (red)

The feature density map of the resultant camera frames is shown in Figure 4.25. In contrast with the stationary feature density map shown in Figure 4.19 there are no distinct feature regions. This is due to the fact there is motion during this test, and instead information is gathered by examining vertical trends in the image as they represent regions of high feature density during the experiment. Once again, red denotes the feature densities prior to enabling the gimbal while green represents the feature densities after enabling the gimbal. During the initial portion of the test the features are all concentrated in the right third of the image, resulting in a higher covariance estimate and less features visible overall. After enabling the gimbal there is a marked increase in the spread of tracked features. The larger distribution of features allows

the VSLAM algorithm to have a more confident state estimate which explains the decrease in covariance presented in earlier figures.

FIGURE 4.26: Test rig for outdoor runway testing.

4.3.4 Outdoor Testing

After verification of gimbal operation in an controlled indoor scenario, outdoor experiments were conducted to study algorithm operation under real-world conditions. The outdoor test setup is shown in Figure 4.26. The cart traverses along the runway edge, yielding a camera view that is approximately half runway and half grass texture. Similar to indoor testing, PTAM is not able to find and track points on the repetitively textured runway. Additionally outdoor testing allows for comparison with GNSS and also with the estimates provided by MicroPilot's LCKF. The experimental layout is similar to the indoor test and is shown in Figure 4.27.

The experimental procedure was as follows:

1. The autopilot is initialised and begins recording information to the datalog.
2. PTAM is initialised using a $10cm$ baseline distance with the camera facing downwards and the gimbal control algorithm disabled.
3. MSF is initialised and the bias estimates are allowed to stabilise.
4. The test setup is translated 10 metres in the Y direction, pausing for 3 seconds at the 10 metre mark prior to returning to the origin.

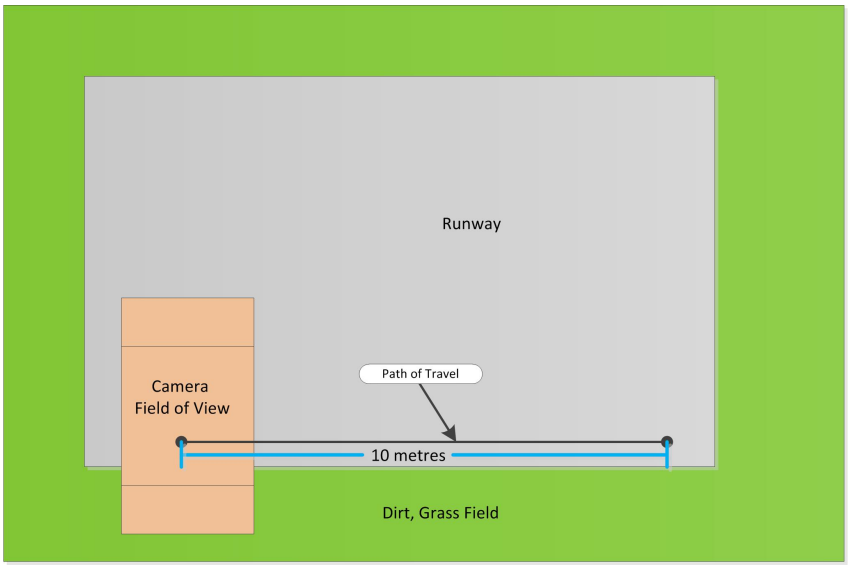


FIGURE 4.27: Test track for outdoor ground test

5. Step 4 is repeated a second time.
6. The gimbal is then enabled and steps 4 and 5 are repeated.
7. Datalogs from autopilot and SBC are downloaded for later analysis.

The data from MSF is to be compared to the data from the LCKF additional data are logged to allow synchronisation. GPS position, pressure altitude and magnetic compass heading are logged to the computer running MSF to allow synchronisation. Additionally, recall from Section 2.3.1 that yaw is not observable with standard vision based measurements alone. As such, for comparison purposes the yaw reported by MSF will be rotated to match that reported by the autopilot at the start of the test via a rotation matrix $R_z(\delta\psi)$, where $\delta\gamma = \psi_{autopilot} - \psi_{msf}$ at the time of initialisation. Figure 4.28 shows the yaw measured by MSF and the autopilot's LCKF during one test run. Note at the start of the test MSF is showing a yaw of 358° while the autopilot is showing a yaw of 261.1°. This implies a rotation between the autopilot and MSF of $\delta\gamma = 261.1° - 358° = -96.9°$.

Thus any MSF data used for comparative study with the autopilot will be rotated by $R_z(-96.9°)$. Additionally, data from MSF are transformed from the East North Up (ENU) co-ordinate frame to use the North East Down (NED) frame. Figure 4.29 shows how MSF position estimates change as a result of the heading correction.

Results from three experimental trials are presented in Figure 4.30 and summarised in Table 4.4. Note that once again the gimbal controller is able to provide a substantial increase in the number

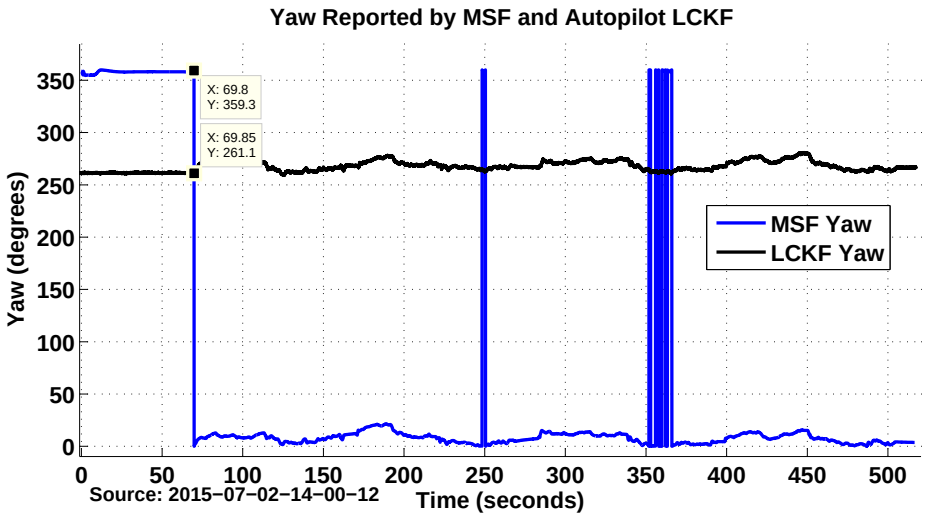


FIGURE 4.28: Yaw measured by MSF and autopilot LCKF

of points found by the corner tracking algorithm, with an average increase in the number of detected features of 43.4% when the gimbal is enabled. It is important to note that this metric does not convey information about how the VSLAM algorithm will perform but does demonstrate that the gimbal control algorithm works as expected and produces notable results.

Detected Features Vs. Gimbal Roll			
Test Run Date Code	Gimbal Disabled	Gimbal Enabled	Improvement
2015-07-02-14-00-12	1584.7	2330.3	47.0%
2015-07-02-14-20-36	1910.8	2773.4	45.1%
2015-07-02-14-41-37	1973.9	2728.3	38.2%

TABLE 4.4: Mean number of features detected during outdoor tests

The covariance estimate from PTAM also showed a marked improvement after the gimbal is enabled, as shown in Figure 4.31. After enabling the gimbal there is a clear decrease in both the mean and standard deviation of the covariance. Test results are summarised in Table 4.5.

PTAM Covariance Vs. Gimbal Roll			
Test Run Date Code	Gimbal Disabled	Gimbal Enabled	Improvement
2015-07-02-14-00-12	0.00463	0.00344	25.7%
2015-07-02-14-20-36	0.00460	0.00359	22.0%
2015-07-02-14-41-37	0.00237	0.00170	28.3%

TABLE 4.5: PTAM covariance estimate during outdoor tests

The decrease in covariance is in line with prior experiments and shows an increase in the quality of the estimate provided by VSLAM. Of particular note are times $t = 345s$ and $t = 560s$ for fig. 4.31a, $t = 420s$ and $t = 540s$ for fig. 4.31b and $t = 410s$ and $t = 530s$ for fig. 4.31c. At this point the camera overlooks the corner of the runway and begins to track features in the

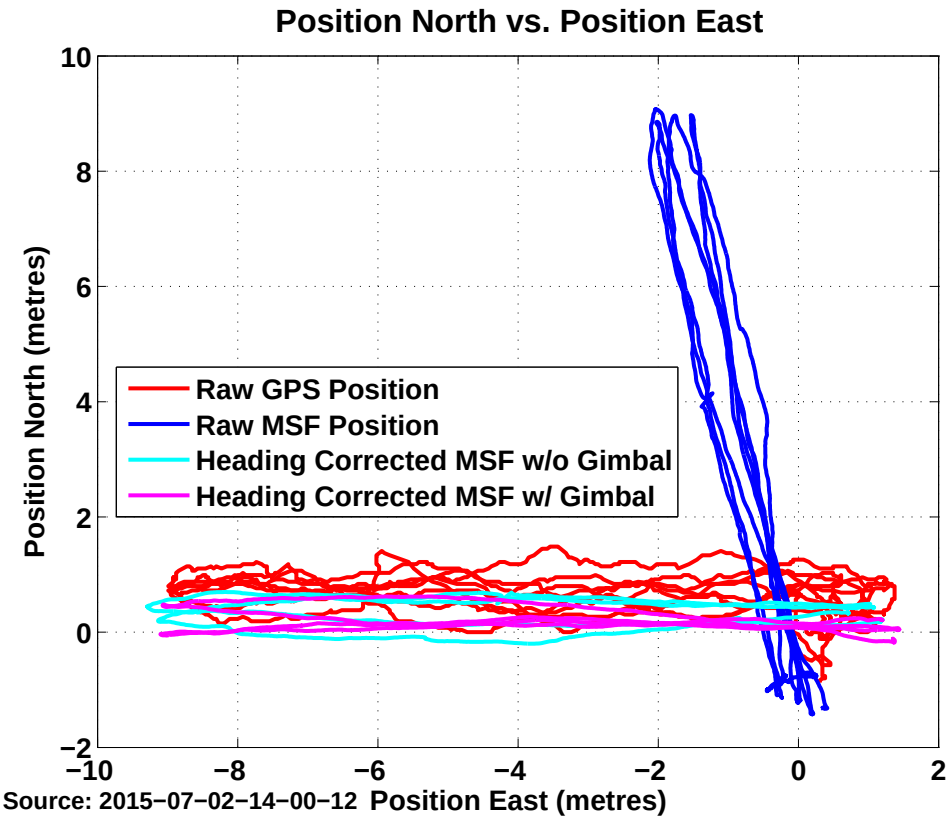
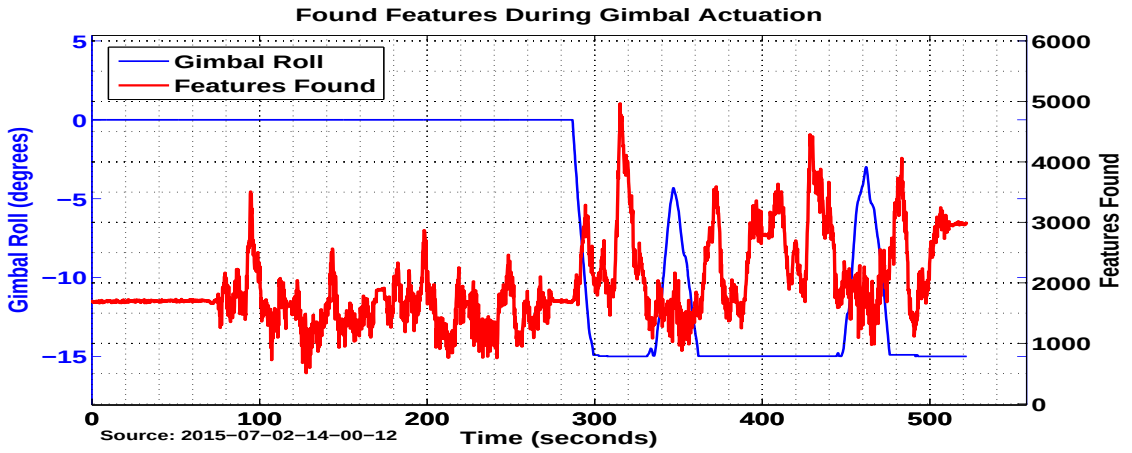


FIGURE 4.29: MSF position after heading correction is applied. Heading correction is only applied at the start of the test and it is expected that this will drift over the duration of the experiment.

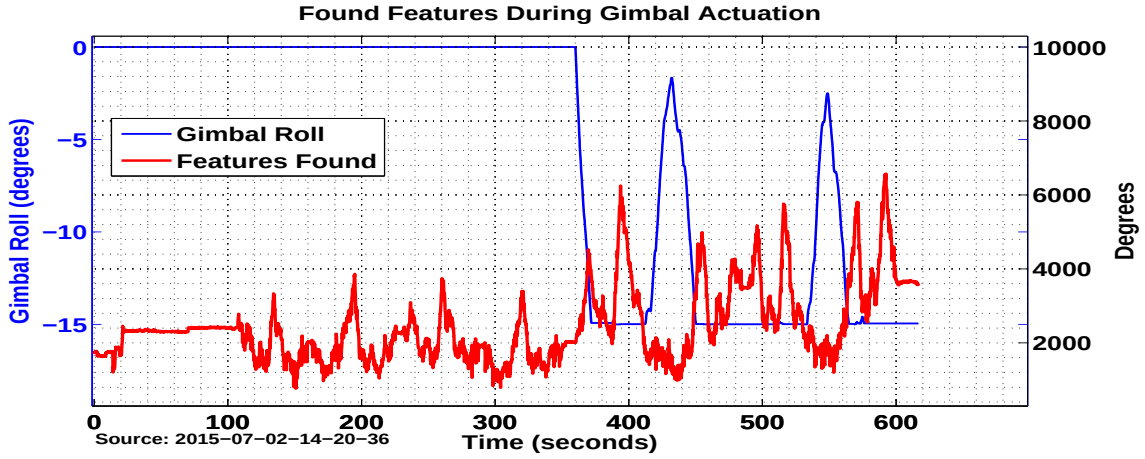
left portion of the image. As a result the gimbal begins to rotate the camera back to the left. During this period there are fewer features found and an increased covariance. This increased covariance is approximately the average covariance when the gimbal is disabled and thus is not a cause for concern. Additionally if the 'zero velocity deadband' of the gimbal controller was enabled this behaviour would not be seen as the UAS is stationary during this period. Immediately after beginning to travel back to the origin the covariance drops and number of tracked points increases.

Perhaps the most important metric - position - is shown in Figures 4.32 to 4.34. Ground truth in this scenario would be a line running from $-10m$ to $0m$ in the east-west direction. It is expected in this scenario that there will be an initial offset, however this offset should be DC and constant throughout the test. Since there was no movement in the north axis it is assumed that any motion reported in this direction is error.

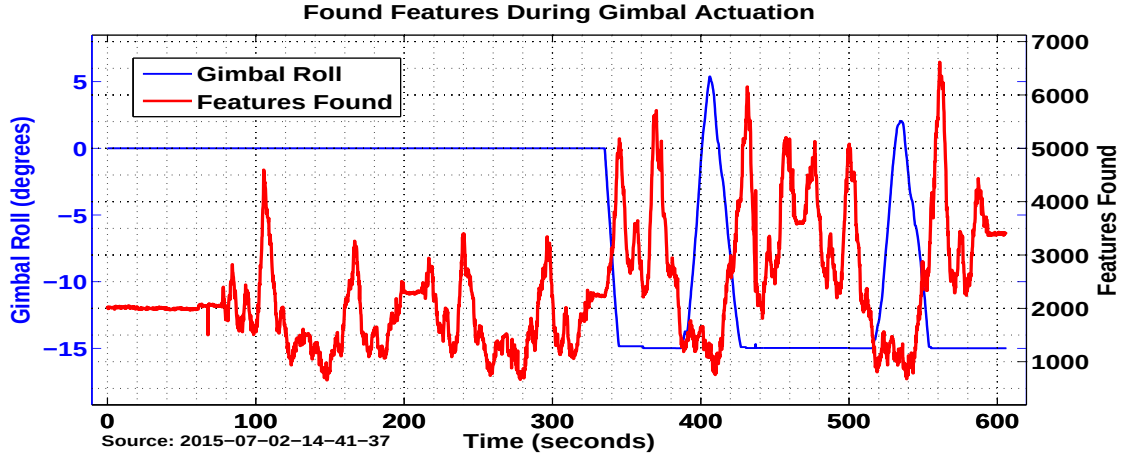
In general it is noted that enabling the gimbal results in less standard deviation in the north direction when comparing with the gimbal disabled. The notable exception is the test shown



(A) Map Points vs. Gimbal Roll, test 2015-07-02-14-00-12

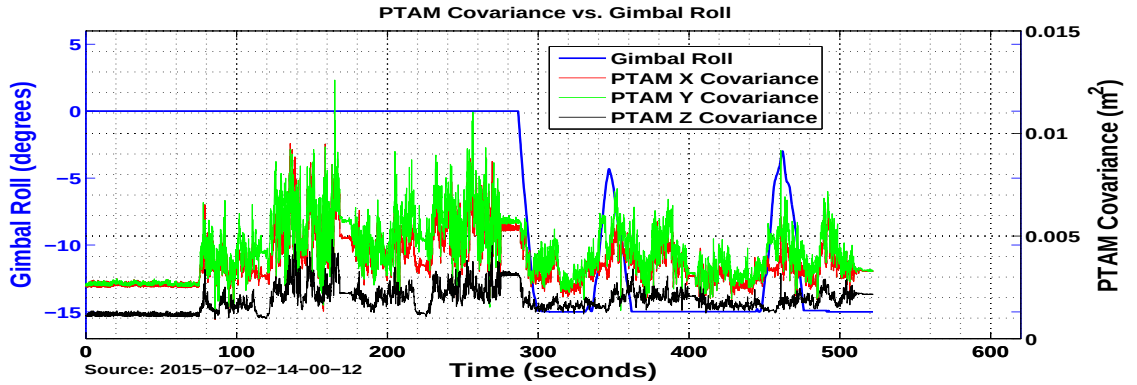


(B) Map Points vs. Gimbal Roll, test 2015-07-02-14-20-36

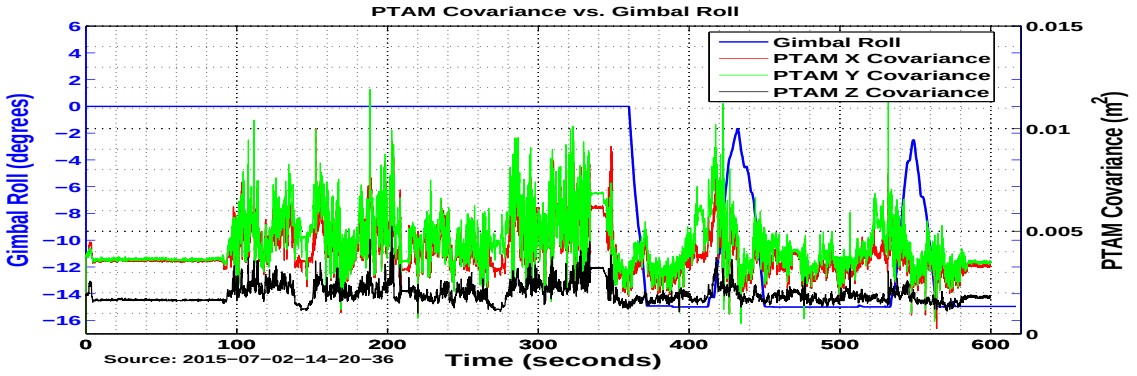


(C) Map Points vs. Gimbal Roll, test 2015-07-02-14-41-37

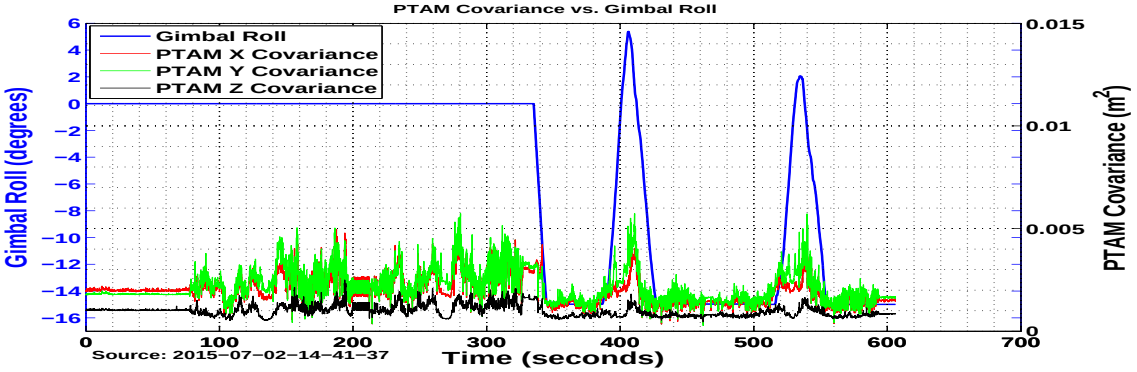
FIGURE 4.30: Detected features with gimbal roll for outdoor single axis test. The gimbal is able to provide an average 43% increase in found points.



(A) PTAM Covariance vs. Gimbal Roll, test 2015-07-02-14-00-12



(B) PTAM Covariance vs. Gimbal Roll, test 2015-07-02-14-20-36



(C) PTAM Covariance vs. Gimbal Roll, test 2015-07-02-14-41-37

FIGURE 4.31: Detected features with gimbal roll for outdoor single axis test

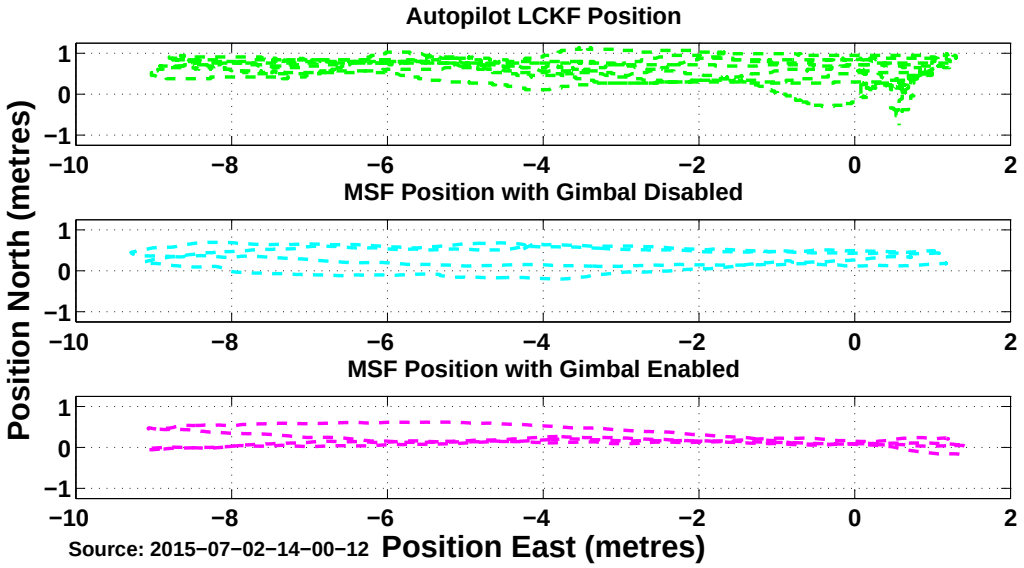


FIGURE 4.32: Autopilot position compared with MSF position with and without the gimbal enabled

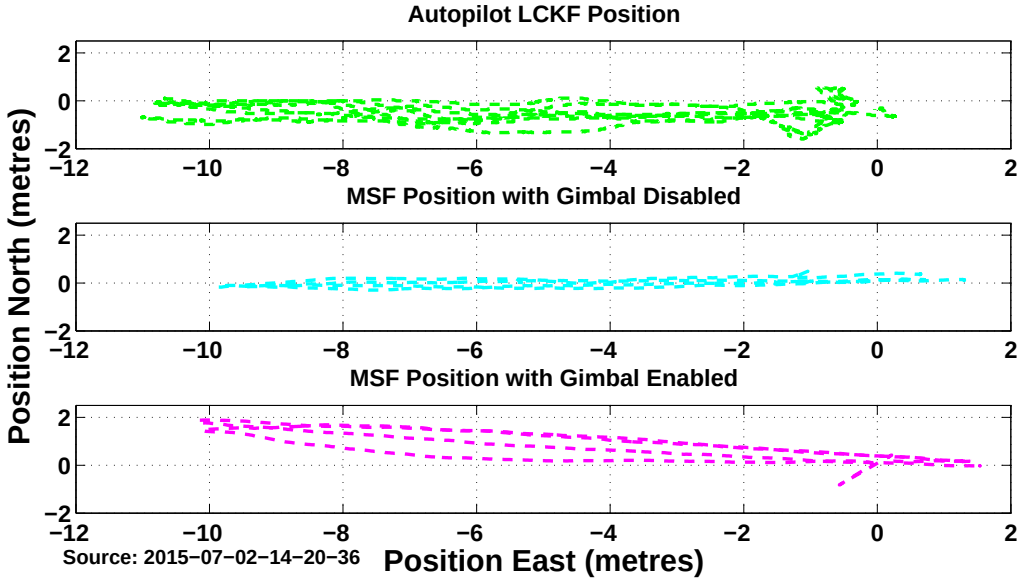


FIGURE 4.33: Autopilot position compared with MSF position with and without the gimbal enabled

in Figure 4.33 where enabling the gimbal results in a 2m north deviation in 10 metres. This appears to be a result of several compounding factors. First, observe that the initial scale is not estimated correctly with the gimbal disabled – the distance travelled should only be $10m$, not $11m$. Additionally, the standard deviation increase appears negligible, instead it appears that the error may be caused by a constant heading offset, indicating the potential that the MSF world frame may have rotated with respect to the initialisation point. This is the large

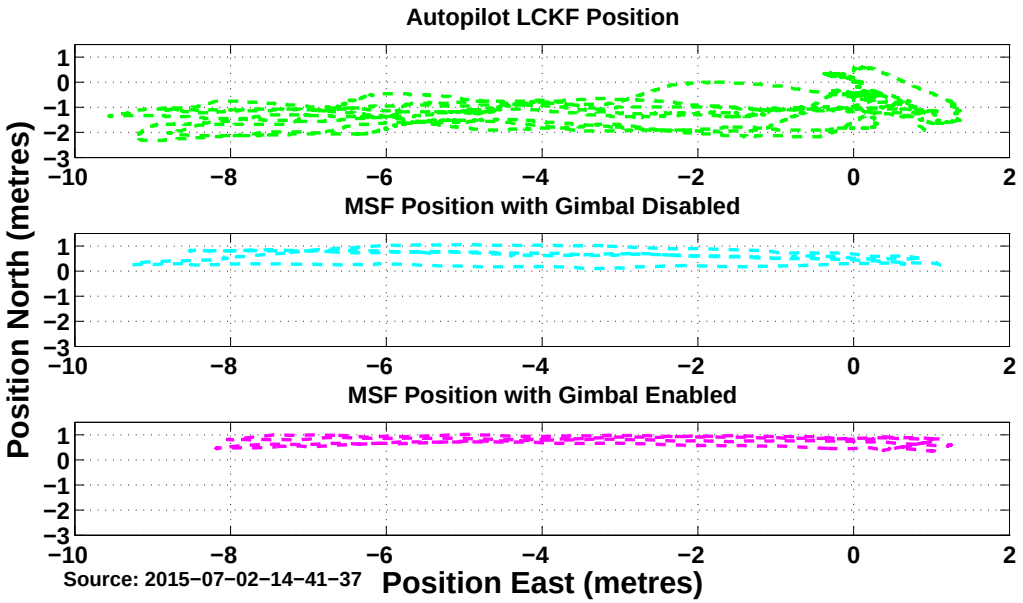


FIGURE 4.34: Autopilot position compared with MSF position with and without the gimbal enabled

disadvantage of relying on a system where only relative measurements are available - it is susceptible to accumulating error with respect to world axes that are not observable without additional sensors (such as a compass or GNSS).

In general there is a small $(1m - 2m)$ offset in reported GPS position. This can be accounted for by the fact that the GPS position accuracy is $2.5m$ CEP and there is typically a drift as the GPS receiver turns on and acquires more satellites and increases the accuracy of its lock. This drift will typically be small and bounded. By comparison, PTAM drifts spatially (specifically as a function of distance travelled due to the global optical flow behaviour). As a result, if the UAS remains stationary PTAM is highly accurate, however if the UAS begins to move the drift errors from the VSLAM subsystem begin accumulating.

It is also important to note the effect of scale on the overall system accuracy. In Figure 4.32 the scale is accurately estimated and the east position varies from $-9m \rightarrow 1m$, a $10m$ total travelled distance. In contrast, in Figure 4.34 the scale is not accurately estimated and thus the travelled distance dramatically changes when the gimbal is enabled. A global position reference can be used to remove this drift and scale error (as done by Lynen [21]) providing the covariance of PTAM is correctly scaled to the distance travelled.

In all the presented cases we see that the deviation of the MSF position is less than that reported by the LCKF from the autopilot. While this is notable it should also be mentioned that low

deviation is only particularly useful while hovering. In general, accurate estimation of current position with respect to a fixed origin is more desirable, in which case fusion of the VSLAM system with a global reference (e.g. GNSS) is a necessity.

4.4 Flight Testing Results

After thorough ground testing it is desirable to verify results through flight testing under real-world conditions. The aircraft was equipped with all sensors necessary for flight and is flown along the north edge of runway 09/27 at MicroPilot's flight test facility. It is expected that the gimbal will rotate to maintain the highest feature density possible. During actual flight testing it was found that PTAM is not able to maintain reliable estimates during the takeoff sequence and thus the pose estimate is corrupted at the beginning of the flight test procedure. Due to limitations in the experimental setup it is not possible to initialise PTAM in flight and thus no data was captured from PTAM and MSF during the test flights. Thus, the only data available as a result of these tests are the corners found by the FAST16 corner detector since they do not rely on the stability of PTAM or MSF. While the lack of reliable flight test data is a setback it is possible to estimate if the gimbal would have an effect by examining how the visible corners change due to gimbal behaviour.

4.4.1 Test Aircraft

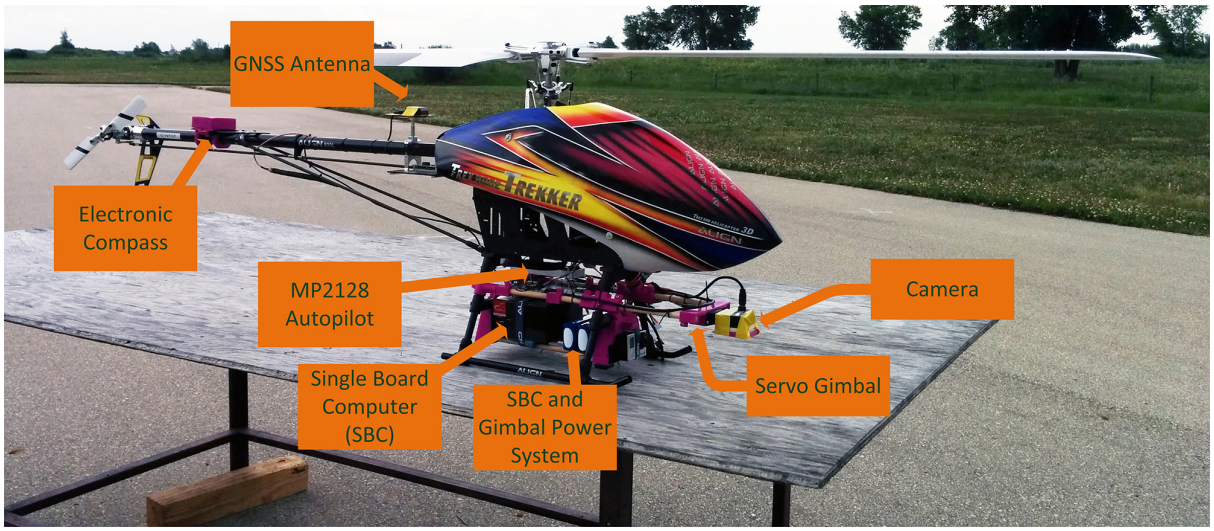


FIGURE 4.35: T-REX 800E Trekker with Autopilot, Compass, GNSS, SBC and Camera for recording flight test data

Figure 4.35 shows an Align T-REX 800E Trekker that was equipped with a Gigabyte Brix computer running PTAM and ROS according to the block diagram shown in Figure 3.10. A MicroPilot 2128^{HELI2} autopilot was installed underneath the helicopter's main shaft to minimise the effects of centripetal acceleration in measured accelerations. The camera gimbal and computer mount were designed in CAD and then 3D printed to custom fit to the airframe. The gimbal is servo driven and receives PWM position commands from the autopilot, which receives angle position commands from the Single Board Computer via UART.

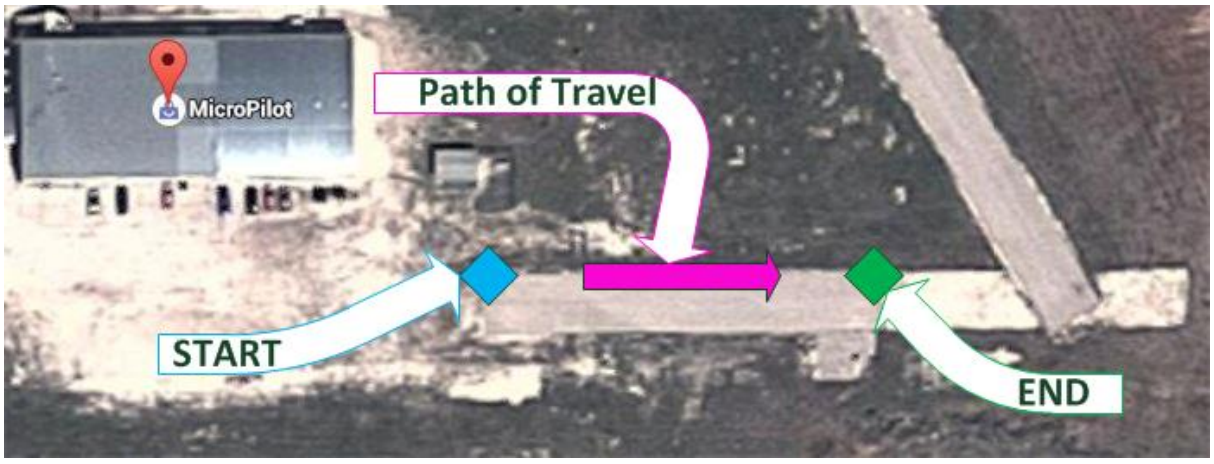


FIGURE 4.36: Flight path taken during test

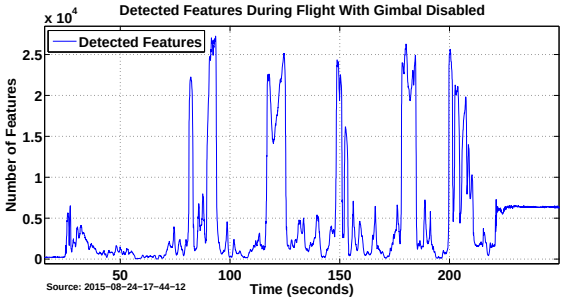
4.4.2 Flight Test Procedure

The helicopter took off from a table placed at the west edge of runway 09 and flew east towards the far edge with a heading of approximately 90°. The helicopter maintained approximately 1m in height for the duration of the test. After reaching the east side of the runway it performed a 180° turn and flew along runway 27 with a heading of approximately 270° and landed on the table. The test was performed with and without the gimbal enabled.

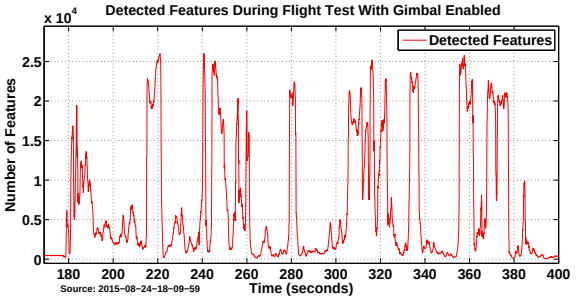
4.4.3 Flight Test Results

Due to the fact that PTAM was not stable during flight tests there are no data available for pose from PTAM or MSF. As a result, the only available data are detected features from the FAST16 corner detector. In this mode it is also not possible to synchronise the data with autopilot datalogs and as such no GPS ground truth is available. Figure 4.37a shows the detected points

during the flight test with the gimbal disabled and Figure 4.37b shows the detected features during the flight with the gimbal enabled.



(A) View from SLAM camera with gimbal disabled. Only 1000 features are detected due to looking at featureless asphalt.



(B) View from SLAM camera with gimbal enabled. Approximately 5000 features are detected.

FIGURE 4.37: View from VSLAM camera during flight testing. Many more features can be seen over grass than over the asphalt.

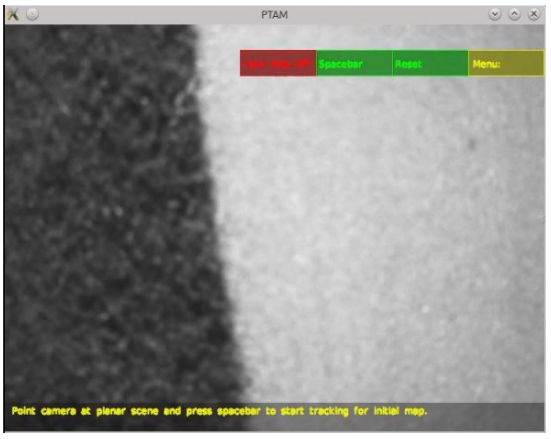
During the test without the gimbal enabled the mean number of features detected was 4557, and with the gimbal enabled this increased to 6495. However, there is a very large standard deviation associated with each dataset, with the gimbal disabled dataset having a standard deviation of $6253pts^2$ and the gimbal enabled dataset showing a standard deviation of $7769pts^2$. The large standard deviation can be attributed to the low flight altitude, since a translation of $1m$ places the helicopter completely over the runway with no trackable points. The large deviation makes it impossible to determine any conclusive results from the data.

Through video verification we can clearly see that the gimbal is rotating towards more feature dense areas. This has the largest effect when the helicopter is near the runway edge where a large portion of a fixed downward looking camera's view is occupied by featureless asphalt. Clearly from both on board video and actual SLAM camera video there are more textured surfaces visible with the gimbal enabled.

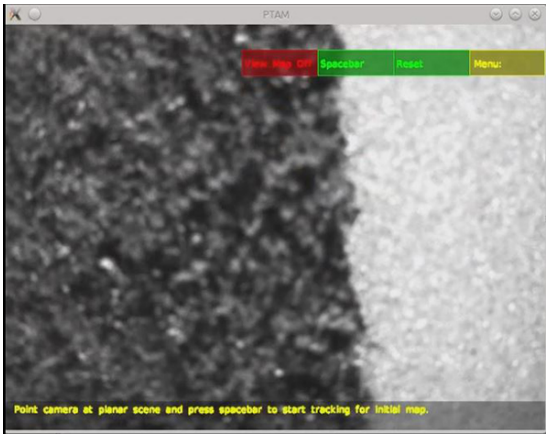
(A) On-board view with gimbal disabled. Note the directly downward looking camera largely sees featureless runway instead of textured grass.

(B) On-board view with gimbal enabled. The camera is able to tilt towards the feature-rich grass in an attempt to maintain a stable visual SLAM estimate.

FIGURE 4.38: On-board views with gimbal disabled and enabled. The gimbal control algorithm works as expected, pointing the camera towards more feature rich areas.



(A) View from SLAM camera with gimbal disabled. Only 1000 features are detected due to looking at featureless asphalt.



(B) View from SLAM camera with gimbal enabled. Approximately 5000 features are detected.

FIGURE 4.39: View from VSLAM camera during flight testing. Many more features can be seen over grass than over the asphalt.

4.5 Discussion of Results

4.5.1 Gimbal Control Algorithm

From the data presented it is clear that the gimbal control algorithm is able to increase the number of features visible to the VSLAM algorithm. By using the simple methodology of sub-image segregation and feature density weighting a controller can be developed which is able to

ensure that the camera can be rotated to view reliably tracked features. This technique allows a monocular system to acquire some of the flexibility and robustness from multiple camera stereo vision SLAM systems while retaining the weight and complexity advantages inherent with monocular systems. The decoupled nature of the algorithm presented allow it to be easily transportable to other VSLAM algorithms such as LSD SLAM and MonoSLAM, while the rotation inverser reduces the required complexity of associated Kalman filter algorithms.

The current control algorithm is somewhat limited in its usefulness due to its simplistic implementation. While it is certainly capable of increasing the number of features found by the corner tracking algorithm this does not necessarily result in increased robustness of the VSLAM algorithm. Since there is no feedback from the VSLAM algorithm it is entirely possible that the commanded movement of the camera may cause the SLAM algorithm to become unstable. Careful control algorithm design is used to mitigate this through slew rate limiting and the zero velocity deadband, but without true feedback from the SLAM algorithm instabilities will continue to be a concern.

4.5.2 Rotation Inverser

Since the implementation of the rotation inverser is largely open loop it is likely the largest source of error present in these tests. As noted in Figure 4.9 the roll channel does experience notable error but overall the system remains stable. Additionally the X position error is quite large at $8cm$. While this is likely in part due to the obscured features during the test it is important to remember that any errors due to the gimbal movement are not accurately represented in the covariance estimate from PTAM. As such these errors may directly affect position accuracy. This is not typically an issue as $8cm$ is negligible when compared with the current GNSS standard deviations which are on the order of metres.

4.5.3 Reliability of PTAM

The goal of both the gimbal control algorithm and the rotation inverser is to increase the reliability of PTAM. Using the covariance estimate from PTAM's bundle adjustment algorithm it is clear that the use of the gimbal control algorithm is capable of decreasing the covariance reported by PTAM. In addition, there is an increased number of features tracked on each SBI level along with a corresponding increase in the total points in the world map.

During indoor testing in Section 4.3.3 the gimbal was able to make estimates from MSF stable while traversing sparsely featured regions. This is particularly notable as a large target area visual SLAM is reliable indoor operation. The use of a gimballed camera would allow for a downward facing camera for use in outdoor scenes which could be automatically rotated to forward or side looking when transitioning from outdoor to indoor setting.

In Section 4.3.4 the gimbal algorithm was able to provide a decreased position deviation in two out of three tests. In the third test there appeared to be a heading change mid-way through the experiment which causes the increased error. In all test cases the performance of MSF was on par with the autopilot's LCKF however the inherent scale drift and scale estimation error ultimately lead to position error as motion occurs. Since scale information is not directly observable from any monocular SLAM system a gimballed camera will not be able to address this error, however by using the gimballed camera to track the largest amount of features possible scale drift can be largely mitigated.

PTAM was particularly unreliable during flight testing. This is likely due to the fact the experiments are conducted close to the ground where high apparent feature velocities cause instability. In work conducted by Weiss in [41] it is noted that PTAM is not initialised until the aircraft is at high altitude ($> 13m$) and allowed to stabilise at that altitude. In prior work the speed was also limited to $2m/s$ compared to between $2m/s - 5m/s$ in tests conducted during the course of this research. This is largely due to the fact the aircraft in these experiments is under manual control and not able to be piloted as precisely as autonomous control. Experimental setup limitations prevented the system from being initialised while in flight meaning the exact steps presented in [41] could not be replicated.

Approximately 30 attempts at flight tests were conducted with 28 resulting in tracking loss immediately after takeoff. Two flights maintained marginal tracking during a brief hover prior to forward flight but lost tracking during the forward flight transition. While the gimbal controller was able to improve the reliability of PTAM it did not increase the reliability to a level where it could be trusted as the sole position measurement on a UAS. GNSS losses are typically temporary and once lock is regained a global position reference is once again available. In contrast, when VSLAM tracking is lost the most useful approach is typically to re-initialise the SLAM algorithm from the current location, resulting in a large position drift. As such, where available VSLAM algorithms should be assisted with GNSS measurements to provide the most stable state estimate possible.

Chapter 5

Conclusion and Future Work

5.1 Conclusions

This thesis has presented a novel method of modularly coupling a gimballed camera into an existing VSLAM algorithm for the purposes of increasing estimation reliability. The objective was to show that incorporating a movable camera and intelligent pointing algorithm it is possible to increase the robustness of visual SLAM algorithms. Additionally, a rotation inverser was demonstrated which allows this camera pointing algorithm to be integrated with existing SLAM and state estimation algorithms without major modification or performance penalties. From both ground based and flight testing this thesis concludes that a gimballed camera can be successfully integrated with existing VSLAM algorithms and provide a notable performance increase in terms of the covariance reported by the SLAM algorithm.

The static weighted feature density control algorithm for the gimbal performed adequately for the scenarios presented during the experimental validation. The basic 1D control algorithm correctly rotates the camera towards the most feature dense regions, allowing the VSLAM algorithm to track more corners and ultimately create a more accurate visual SLAM estimate. This has trickle down effects which in turn allow for a more accurate state estimate overall.

The rotation inverser algorithm is able to successfully remove gimbal rotation based on a open loop architecture, thereby negating the need for expensive and bulky position sensors. The success of this algorithm implies that the gimballed SLAM technique would work successfully on other SLAM algorithms, such as Large Scale Direct (LSD) SLAM, and opens up future research possibilities in the field of non-rigidly attached visual SLAM.

While PTAM was reliable on the ground and data were able to be gathered it was not functional in flight at all. Some possibilities for this behaviour were outlined in Section 4.5.3, including issues with manual control and being initialised too close to the ground. Since it was not possible to gather reliable flight test data it is not possible to draw any definitive conclusions about the function of a gimbal on a visual SLAM system.

5.2 Future Work

This thesis presents a foundation on which numerous possible future research topics may be based. Obviously the visual SLAM algorithm should be made stable on the aircraft in flight prior to any future work being pursued. The next logical step would be to add the ability to have fully autonomous control and incorporate the initialisation-in-flight feature used by Weiss in [41]. This would allow the ability to test the aircraft under stable conditions with controlled movements being more accurate than a human pilot can command. Additionally, initialising PTAM at a higher altitude will inherently make it less vulnerable to velocity or attitude changes disrupting the VSLAM estimate. The next desirable phase would be to improve reliability at lower altitudes. While this was accomplished by Harmat et. al. in [14] by utilising a forward facing camera, it should be possible to modify the gimbal control algorithm to work with MSF to tilt the camera to forward or side looking at lower altitudes, allowing more robust horizon tracking rather than ground tracking.

While a 1D gimbal acting in the roll axis was shown to increase the stability of PTAM, a next logical step would be to investigate 2D gimbals, particularly on hovering vehicles in sparsely featured environments where a single rotation may not yield enough trackable features to maintain a stable position estimate. Additionally, the gimbal control algorithm presented is loosely coupled in the sense that it only observes corners fed to the SLAM algorithm. A dynamically updated weighting matrix would be able to adapt to changes in aircraft velocity better than a static weight matrix, and more advanced algorithms could use Hough or Fourier transforms to measure the randomness of the observed features, preventing confusion from self-similar textures. Moreover, while more features should logically increase the robustness of the algorithm this is not necessarily the case and it may be possible to obtain better performance by more tightly coupling the gimbal control algorithm with the VSLAM portion and incorporating covariance estimates to be used as inputs to the gimbal control algorithm.

Alternatively, rather than using a gimballed camera as a means of increasing VSLAM robustness, the opposite approach may be taken, where VSLAM is applied to a user pointable camera and the rotation inverser is used to maintain tracking consistency. This has the obvious advantage of reducing the total number of cameras carried on the UAS, as now the mission payload can also be used as a visual SLAM input camera as well. There is a strong potential for the user to force a tracking loss through camera motion which would need to be addressed, either through tracking loss mitigation measures or by limiting the user's inputs to the camera.

Bibliography

[1] Honeywell Aerospace. Honeywell embedded GPS/INS (EGI). https://aerospace.honeywell.com/~/media/Brochures/N61-0411-000-002%20-%20Embedded%20GPS-INS%20EGI.ashx, https://aerospace.honeywell.com/~/media/Brochures/N61-0411-000-002%20-%20Embedded%20GPS-INS%20EGI.ashx. Published June 2015. Last accessed 17 July 2015.

[2] Omead Amidi. *An autonomous vision-guided helicopter*. PhD thesis, Carnegie Mellon University Pittsburgh, PA, 1996.

[3] Nicholas Ayache and Olivier D. Faugeras. Building, Registrating, and Fusing Noisy Visual Maps. *International Journal of Robotic Research*, 7:45–65, 1988.

[4] Vadims Bistrovs and A Kluga. MEMS INS/GPS data fusion using particle filter. *Elektronika ir Elektrotechnika*, 112(6):77–80, 2011.

[5] R. O. Castle and D. W. Murray. Object recognition and localization while tracking and mapping. In *Proc 8th IEEE/ACM International Symposium on Mixed and Augmented Reality, Orlando, Florida, Oct 19-22*, 2009.

[6] Andrew J Davison. Real-time simultaneous localisation and mapping with a single camera. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pages 1403–1410. IEEE, 2003.

[7] Andrew J Davison, Ian D Reid, Nicholas D Molton, and Olivier Stasse. Monoslam: Real-time single camera slam. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(6):1052–1067, 2007.

[8] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. *Machine vision and applications*, 13(1):14–24, 2001.

[9] H. Durrant-Whyte. Uncertain geometry in robotics. *IEEE Journal of Robotics and Automation*, (4):23–31, 1988.

[10] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: Part i. *IEEE Robotics & Automation Magazine*, 13:99–108, 2006.

[11] Pantelis Elinas and James J Little. Stereo vision SLAM: Near real-time learning of 3D point-landmark and 2D occupancy-grid maps using particle filters. In *Visual SLAM Workshop*. Citeseer, 2007.

[12] Paul D Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech House, 2013.

[13] Adam Harmat, Inna Sharf, and Michael Trentini. Parallel tracking and mapping with multiple cameras on an unmanned aerial vehicle. In *Proceedings of the International Conference on Intelligent Robotics and Applications*, volume 1, pages 421–432, 2012.

[14] Adam Harmat, Michael Trentini, and Inna Sharf. Multi-camera tracking and mapping for unmanned aerial vehicles in unstructured environments. *Journal of Intelligent & Robotic Systems*, 78(2):291–317, 2014.

[15] Stephan Hrabar. *Vision–Based 3D Navigation for an Autonomous Helicopter*. PhD thesis, University of Southern California, 2006.

[16] Sungsik Huh, David Hyunchul Shim, and Jonghyuk Kim. Integrated navigation system using camera and gimbaled laser scanner for indoor and outdoor autonomous flight of UAVs. In *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, pages 3158–3163. IEEE, 2013.

[17] Michal Jama and Dale Schinstock. Parallel tracking and mapping for controlling VTOL airframe. *Journal of Control Science and Engineering*, 2011:1–10, January 2011.

[18] Jonathan Kelly and Gaurav S Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. *The International Journal of Robotics Research*, 30(1):56–79, 2011.

[19] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Mixed and Augmented Reality (ISMAR)*, pages 225–234. Curran Associates, Nov 2007.

[20] Thomas Lemaire, Cyrille Berger, Il-Kyun Jung, and Simon Lacroix. Vision-based SLAM: Stereo and monocular approaches. *International Journal of Computer Vision*, 74(3):343–364, 2007.

[21] S Lynen, M Achtelik, S Weiss, M Chli, and R Siegwart. A robust and modular multi-sensor fusion approach applied to MAV navigation. In *Proc. of the IEEE/RSJ Conference on Intelligent Robots and Systems (IROS)*, 2013.

[22] Robert Mahony, Sung-Han Cha, Tarek Hamel, and France Antipolis. A coupled estimation and control analysis for attitude stabilisation of mini aerial vehicles. In *Australiasian Conference on Robotics and Automation, Auckland, New Zealand*, pages 1–10, 2006.

[23] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM: A factored solution to the simultaneous localization and mapping problem. In *Proceedings of the AAAI National Conference on Artificial Intelligence*, Edmonton, Canada, 2002. AAAI.

[24] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. FastSLAM 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, Acapulco, Mexico, 2003.

[25] Fritz K Mueller. A history of inertial guidance. Technical report, DTIC Document, 1963.

[26] NOAA. NOAA pressure altitude calculator. `http://www.srh.noaa.gov/epz/?n=wxcalc_pressurealtitude`. Published October 1 2009. Last accessed January 26, 2015.

[27] Gabriel Nutzi and Stephan Weiss. Fusion of imu and vision data for absolute scale estimation in monocular SLAM. *Journal of Intelligent & Robotic Systems*, 61:287–299, January 2011.

[28] Point Grey Research. PGR stereo accuracy chart. `http://www.ptgrey.com/KB/10022`, `http://www.ptgrey.com/Content/Images/uploaded/modifiedstereoaccuracy.xls`. Published June 6 2014. Last accessed January 26, 2015.

[29] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 32:105–119, 2010.

[30] Stergios Roumeliotis, Gaurav Sukhatme, George A Bekey, et al. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *Robotics and Automation, 1999. Proceedings. 1999 IEEE International Conference on*, volume 2, pages 1656–1663. IEEE, 1999.

[31] Davide Scaramuzza, Agostino Martinelli, and Roland Siegwart. A flexible technique for accurate omnidirectional camera calibration and structure from motion. In *Computer Vision Systems, 2006 ICVS'06. IEEE International Conference on*, pages 45–45. IEEE, 2006.

[32] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Vision-based state estimation and trajectory control towards high-speed flight with a quadrotor. In *Robotics: Science and Systems*. Citeseer, 2013.

[33] Robert Sim, Matt Griffin, Alex Shyr, and James J Little. Scalable real-time vision-based slam for planetary rovers. *IEEE IROS Workshop on Robot Vision for Space Applications*, 2005.

[34] R. Smith, M. Self, and P. Cheeseman. Autonomous robot vehicles. chapter Estimating Uncertain Spatial Relationships in Robotics, pages 167–193. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[35] Randall C. Smith and Peter Cheeseman. On the representation and estimation of spatial uncertainty. *The International Journal of Robotics Research*, 5(4), 1986.

[36] Hauke Strasdat, José MM Montiel, and Andrew J Davison. Visual SLAM: why filter? *Image and Vision Computing*, 30(2):65–77, 2012.

[37] Alexander A Trusov. Overview of MEMS gyroscopes: History, principles of operations, types of measurements. *University of California, Irvine, USA, maj*, 2011.

[38] S Weiss and R Siegwart. Real-time metric state estimation for modular vision-inertial systems. In *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, 2011.

[39] Stephan Weiss, Markus W Achtelik, Simon Lynen, Margarita Chli, and Roland Siegwart. Real-time onboard visual-inertial state estimation and self-calibration of mavs in unknown environments. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 957–964. IEEE, 2012.

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

[41] Stephen Weiss. *Vision Based Navigation for Micro Helicopters.* PhD thesis, ETH Zurich, August 2012.

[42] Oliver J Woodman. An introduction to inertial navigation. *University of Cambridge, Computer Laboratory, Tech. Rep. UCAMCL-TR-696*, 14:15, 2007.