Towards Robust Autonomous MAV Landing with a Gimbal Camera

by

Christopher L. Choi

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

Waterloo, Ontario, Canada, 2018

Abstract

As micro aerial vehicles (MAVs) become increasingly common as platforms for aerial inspection, monitoring and tracking, the need for robust automated landing methods increases, for both static and dynamic landing targets. Precision MAV landings are difficult, even for experienced human pilots. While semi-autonomous MAV landings have proven effective, they add additional requirements for multiple skilled operators, which in turn increase the operational costs. This is not always practical and the human in the loop prevents the possibility of more efficient robotic teams that do not require human operators. As such, an automated landing system has been a growing topic of interest to both industry and academia.

In this thesis the aim is to address three different issues. First, in order for a MAV to land autonomously onto a moving target, a complete tracking and landing system for MAVs is needed. An end-to-end system termed ATL is introduced. Results show that ATL is able to track and execute a planned trajectory onto a moving landing target at speeds of 10m/s in simulation.

Secondly, to enable autonomous MAV landings in GPS-denied environments, multiple cameras are needed for simultaneously tracking the landing target and performing state estimation. With the prevalence of gimbal cameras on commercially available MAVs for applications such as cinematography, it is advantageous to use the gimbal camera along with other cameras on-board for state estimation. An encoder-less gimbal calibration method is introduced to enable gimbal cameras to be used with state estimation algorithms. The method was validated by modifying OKVIS to jointly optimize for the gimbal joint angle.

Finally, to achieve full MAV autonomy, all software components on-board must run in real-time on a computer with limited resources. To address this issue and to take advantage of a gimbal camera the Multi-State Constraint Kalman Filter (MSCKF) algorithm is extended by incorporating a gimbal camera. The method was validated in simulation and on a KITTI raw dataset both show promising results.

Acknowledgements

I would like to first thank my supervisor, Prof. Steven Waslander, for his patience, guidance, encouragement and the opportunity to explore the field of robotics. To Stanley Brown, who helped me with the MAV landing project. To Jason Rebello, Leonid Koppel, Pranav Ganti and Arun Das for helping me publish my first academic paper titled "Encoderless Gimbal Calibration of Dynamic Camera Clusters" for ICRA 2018. Finally members of the Waterloo Autonomous Vehicle Laboratory including Melissa Mozifian, Nima Mohajerin, Michael Smart and many more, who played a crucial role in my graduate experience. I have grown personally, professionally and academically and I owe everyone a great deal for making this experience both very enjoyable and memorable.

Dedication

To my mother.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

As micro aerial vehicles (MAVs) become increasingly common as platforms for aerial inspection, monitoring and tracking, the need for robust automated landing methods increases, for both static and dynamic landing targets. Numerous scenarios have been proposed in which MAVs would partner with manned and autonomous ground vehicles and provide real-time aerial information to resolve motion planning or visibility challenges from a second perspective. Monitoring icebergs from surface vessels, finding routes through challenging terrain for ground rovers, aiding police in high speed pursuits, or package delivery where a fleet of drones might be deployed from a truck, are all examples of scenarios where the ability to dock with a moving platform could significantly extend the capabilities and use cases of today's MAVs.

Precision MAV landings are difficult, even for experienced human pilots. As an example, in 2012, a group of researchers from the University of Waterloo dropped a GPS beacon onto an iceberg with a hexacopter. While the conditions were ideal for the ship, where the waves were $1 - 2$m in height and winds were around $10 - 12$ knots. The same conditions proved to be difficult for the experienced human pilot who was controlling the hexacopter. Due to a number of sub-optimal flight conditions such as wind, waves and ship travelling at a significant speed, the hexacopter crashed onto the ship deck and fell into the sea during landing.

While semi-autonomous MAV landings have proven effective, they add additional requirements for multiple skilled operators, which in turn increases the operational costs. This is not always practical and the human in the loop prevents the possibility of more efficient robotic teams that do not require human operators. As such, an automated landing system has been a growing topic of interest to both industry and academia recently, some of which has been spurred by competitions such as the DJI 2016 SDK Developer Challenge [1] and the Mohamed Bin Zayed International Robotics Challenge (MBZIRC 2017) [2]. Both of these challenges feature a significant scoring portion of the overall challenge allocated to the task of landing on a moving target that is travelling at speeds greater than 10 km/hr. In the DJI 2017 challenge only 1 out of 10 finalists were able to land autonomously onto a moving ground target, similarly in the MBZIRC 2017 challenge out of 25 finalists only 2 teams successfully completed the landing challenge [3].

The examples discussed highlight the difficulty in precision MAV landings. In this thesis we aim to address three different issues in the problem of autonomous MAV landing onto a moving target:

- **End-to-end autonomous landing system:** In order for a MAV to land autonomously onto a moving target, a complete tracking and landing system for MAVs is needed. In Chapter 3, we introduce, and we present our end-to-end system we term ATL as well as including contributions towards robust target detection.
- **Gimbal camera calibration for state estimation:** Most existing autonomous MAV landing solutions rely on GPS for MAV state estimation. To enable autonomous MAV landings in GPS-denied environments, multiple cameras are needed for simultaneously tracking the landing target and performing state estimation. With the prevalence of gimbal cameras on commercially available MAVs for applications such as cinematography, it is advantageous to use the gimbal camera along with other cameras on-board for state estimation. We extend the work of [13] and [59] by introducing an encoderless gimbal calibration approach that eliminates the requirement of joint angle measurements from the gimbal. This method is published in ICRA 2018 [11] and is described further in Chapter 4.
- **Filter-based visual inertial odometry:** To achieve full MAV autonomy, all software components on-board must run in real-time on a computer with similar computational power to a laptop. Existing state of the art VIO algorithms are optimization based. In contrast, filter based approaches are more efficient while achieving similar accuracy to the optimization-based approaches. In Chapter 5, we extend the Multi-State Constraint Kalman Filter (MSCKF) algorithm [55] by incorporating a gimbal camera.

[1]DJI 2016 SDK Challenge: https://developer.dji.com/challenge2016/
[2]MBZIRC 2017: http://www.mbzirc.com/challenge

Chapter 2

Background

This chapter briefly covers background theory required for the different contributions presented in this thesis. The camera and distortion models is first described, followed by how features are tracked, matched and estimated. Lastly, the Error State Kalman Filter (ESKF) is covered as a precursor to the G-MSCKF in Chapter 5.

2.1 Notations

We employ the following notation throughout this work: A vector in the global frame, $\mathcal{F}_G$, can be expressed as $\mathbf{p}_G$, or more precisely if the vector describes the position of the IMU frame, $\mathcal{F}_I$, expressed in $\mathcal{F}_G$, the vector can be written as $\mathbf{p}_G^{IG}$ with G and I as start and end points, or for brevity as $\mathbf{p}_G^I$. Similarly a transformation between $\mathcal{F}_G$ to $\mathcal{F}_I$ can be represented by a homogeneous transform matrix, $\mathbf{T}_{IG}$, where its rotation matrix component can be written as $\mathbf{C}_{IG}$. A rotation matrix that is parametrized by quaternion $\mathbf{q}_{IG}$ can be represented as $\mathbf{C}\{\mathbf{q}_{IG}\}$.

2.2 Camera Model and Image Distortion Model

As cameras improve in quality, size and cost, the choice of using a camera as an odometry sensor is becoming a widely researched topic. In this section we describe the camera model and distortion model used, specifically, the pinhole camera model and the radial-tangential distortion model. We will also detail how they are used to describe the projection of 3D scene points onto the image plane.

2.2.1 Pinhole Camera Model

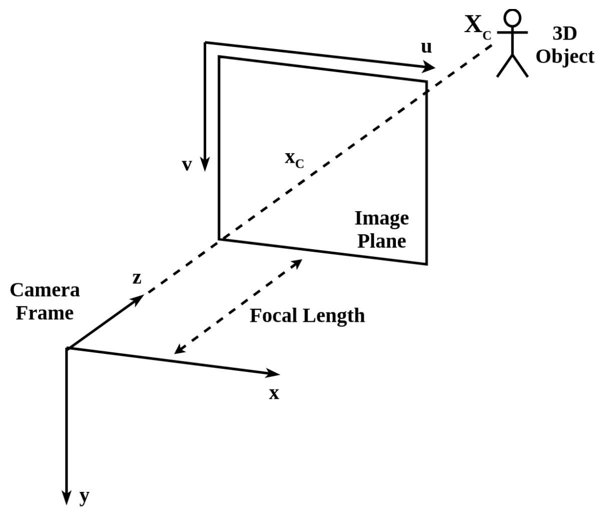


Figure 2.1: Pinhole Camera Model

The pinhole camera model describes how 3D scene points are projected onto the 2D image plane of an ideal pinhole camera. The model makes the assumption that light rays emitted from an object in the scene pass through the pinhole of the camera, and projected onto the image plane. Let us consider Fig. 2.1 where it shows a 3D point $\mathbf{X}_C = (X, Y, Z)$ expressed in the camera frame, $\mathcal{F}_C$, being projected on to the camera's 2D image plane in homogeneous coordinates $\mathbf{x}_C = (u, v, 1)$. The projection can be written as

$$u = \frac{Xf_x}{Z} \quad v = \frac{Yf_y}{Z} \tag{2.1}$$

where f_x and f_y denote the focal length in the x and y-axis. Or, in matrix form

$$\mathbf{x}_C = \mathbf{K}\mathbf{X}_C \tag{2.2}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_x & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X/Z \\ Y/Z \\ 1 \end{bmatrix} \tag{2.3}$$

where $\mathbf{K}$ represents the intrinsic matrix, c_x and c_y represents the principal point offset in the x and y direction.

In practice, the pinhole camera model only serves as an approximation to modern cameras. The assumptions made in the model are often violated with factors such as large camera apertures (pinhole size), distortion effects in camera lenses, and other factors. That is why the pinhole camera model is often used in combination with a distortion model in the hope of minimizing projection errors from 3D to 2D.

2.2.2 Projection Matrix

In Section. 2.2.1, we have described the projection of a scene point, $\mathbf{X}_C$, to the camera's image plane with coordinates, $\mathbf{x}_C$. The scene point, $\mathbf{X}_C$, was assumed to be expressed in the camera's frame of reference, this, however, is often not the case. The scene point can also be expressed in the global frame, $\mathcal{F}_G$. Therefore, a projection matrix $\mathbf{P} \in \mathbb{R}^{3\times4}$ is used to project a scene point expressed in the global frame to the camera's image plane. The projection matrix is made up of an intrinsic and extrinsic matrix.

$$\mathbf{P} = \mathbf{K}\left[\mathbf{C}_{CG} \mid \mathbf{t}_G\right] \tag{2.4}$$

The extrinsic matrix contains the camera's orientation as a rotation matrix, $\mathbf{C}_{CG} \in \mathbb{R}^{3\times3}$, and position, $\mathbf{t}_G \in \mathbb{R}^3$, expressed in the global frame.

2.2.3 Radial Tangential Distortion

Lens distortion generally exist in all camera lenses, therefore it is vital we model the distortions observed. The most common distortion model is the radial-tangential (or simply as radtan) distortion model. The two main distortion components, as the name suggests, are the radial and tangential distortion.

The radial distortion occurs due to the shape of the lens. Light passing through the center undergoes no refraction. Light passing through the edges of the lens, on the other hand, undergoes through severe bending causing the radial distortion. The effects of a positive and negative radial distortion can be seen in Fig 2.2.

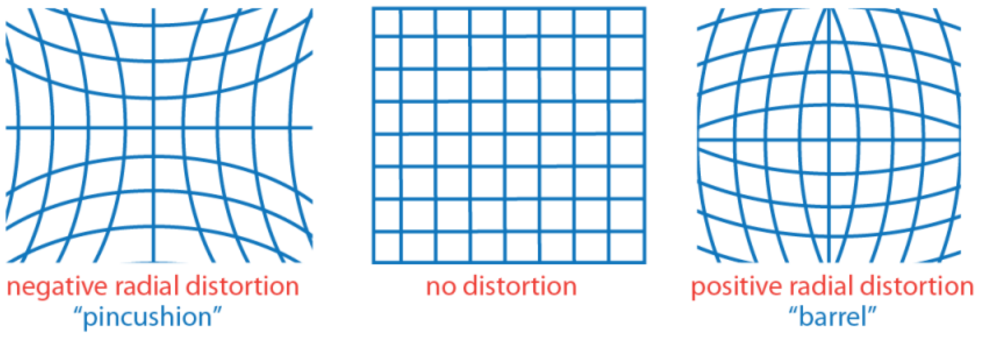


Figure 2.2: Radial Distortion[1]

The tangential distortion is due to camera sensor misalignment during the manufacturing process. It occurs when the camera sensor is not in parallel with the lens. The cause of a tangential distortion can be seen in Fig 2.3.

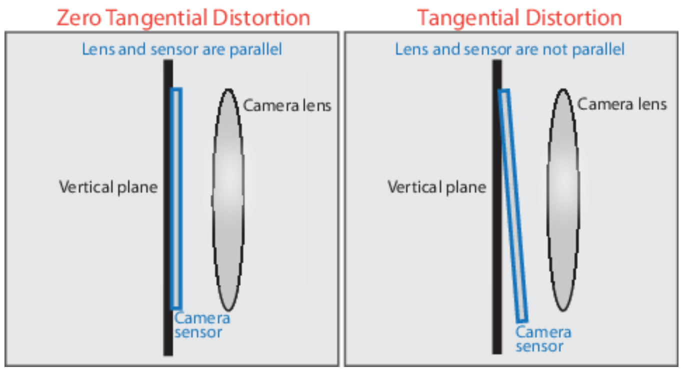


Figure 2.3: Tangential Distortion[1]

The combined radial-tangential distortion is modelled using a polynomial approximation with parameters k_1, k_2 and p_1, p_2 respectively. To apply the distortion the observed 3D point (X, Y, Z) is first projected, distorted, and finally scaled and offset in the image plane (u, v). The radial-tangential distortion model in combination with the pinhole

[1]MATLAB - Computer Vision Toolbox: https://www.mathworks.com/help/vision/ref/cameraintrinsics-class.html

camera model is given in Eq. (2.5) as

$$\begin{aligned}
x' &= X/Z \\
y' &= Y/Z \\
r^2 &= x'^2 + y'^2 \\
x'' &= x'(1 + k_1 r^2 + k_2 r^4) + 2p_1 x'y' + p_2(r^2 + 2x'^2) \\
y'' &= y'(1 + k_1 r^2 + k_2 r^4) + p_1(r^2 + 2y'^2) + 2p_2 x'y' \\
u &= f_x * x'' + c_x \\
v &= f_y * y'' + c_y
\end{aligned} \tag{2.5}$$

2.3 Feature Detection and Matching

In this section we explain how feature points are detected and matched between different camera frames. The common feature detection and matching pipeline for localization and mapping algorithms is:

1. Detect regions of interests (image feature) in the image
2. Extract image feature information using descriptors
3. Match extracted descriptors

2.3.1 FAST Feature Detection

Feature detection in computer vision is a process of gathering scene information and deciding locally whether an image feature exists. The resulting subset of image features in the image domain can in turn be used for localization and mapping algorithms to estimate the camera pose.

For our requirements corners was the chosen image feature. The most widely used corner detector is the FAST feature detector [61]. The advantage of using FAST includes its speed and high detection rate. It operates by inspecting a gray-scale image and applying a Bresenham circle or patch of configurable radius (radius of 3 for a 16 pixel circle in Fig 2.4), where each pixel value on the circle is labeled clockwise. If a set of N contiguous pixels in the circle are all brighter than the intensity of the center candidate pixel p plus a threshold value t, or are all darker compared to p minus a threshold value t, then p is considered a corner.

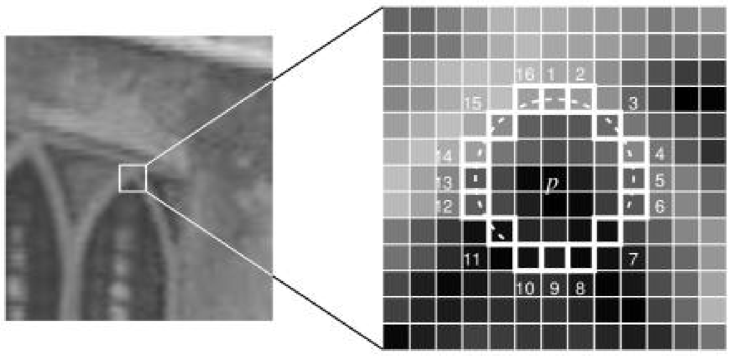


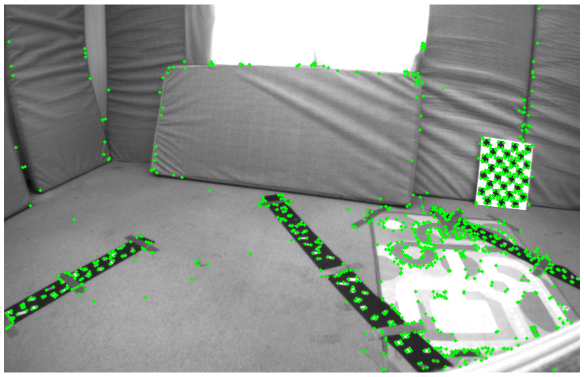
Figure 2.4: FAST Corner Detection [61]

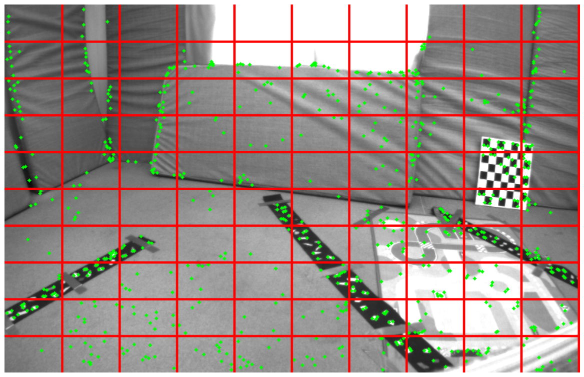
A uniform feature distribution over the image domain is known to avoid degenerate configurations for SLAM, and reduce redundant information. Further, a uniform and un-clustered corner distribution has the potential of increasing computer vision pipeline efficiency, as a lower number of features are required for the whole image. To encourage a uniform feature distribution a custom naive implementation of Grid-FAST was implemented [1]. The naive Grid-FAST was implemented as follows, given an image we divide the image into r rows and c columns with the goal of detecting a total max number of N corners. The max number of corners per grid cell n is then given as

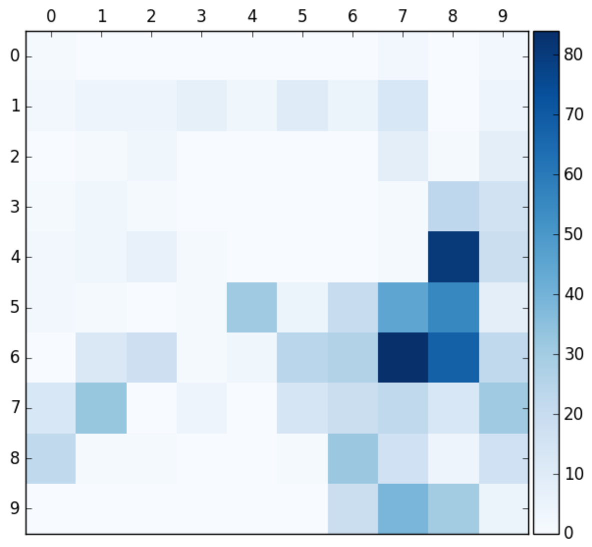
$$n = \frac{N}{r \times c}. \tag{2.6}$$

Using n we limit the corners detected in each image grid cell to naively encourage a uniform distribution.

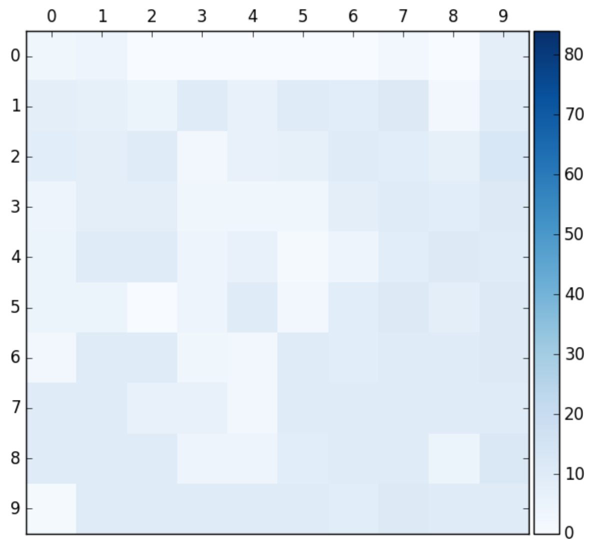
[1]At the time of writing OpenCV has removed the interface to the `GridAdaptedFeatureDetector` implementation from their code base.


(a) FAST Detection (1000 Corners)


(b) Grid-FAST Detection (714 Corners)



(c) 2D Histogram of FAST Detection



(d) 2D Histogram of Grid-FAST Detection

Figure 2.5: Comparison between FAST and Grid-FAST

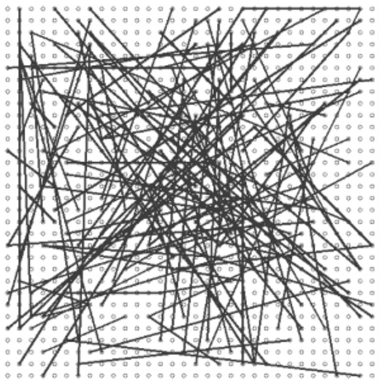
In Fig. 2.5 both FAST and Grid-FAST observe the same image scene with the same detection parameters. Grid-FAST divided the image into 10 rows and columns to encourage a uniform corner detection. While Grid-FAST detected a lower number of corners compared to FAST (714, 1000 respectively), we can observe the benefit of using Grid-FAST in Fig. 2.5c and Fig. 2.5d, where it clearly shows that FAST detection has an undesirably high detection concentration around the chessboard in this particular scene, Grid-FAST on the other hand does not exhibit the same problem. Although, Grid-FAST obtains features of lower quality in terms of repeatable detection, the threshold of corner-ness can be increased if this is an issue.

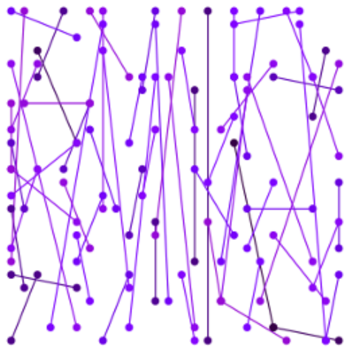
2.3.2 ORB Feature Descriptor and Matching

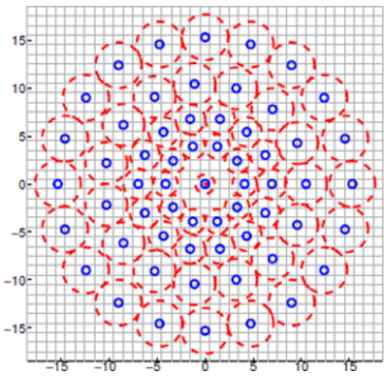
To correspond image features detected in two different image frames a feature descriptor is used. Feature descriptors are a way to describe the image feature observed for matching. There are a number of feature descriptors that extract patch information in order to create a robust and repeatable match. Feature descriptors such as SIFT [48], SURF [1], are histogram of gradients (HOG) based patch descriptors. These HOG descriptors are invariant to small rotations and lighting variations, they are however, relatively expensive to compute. The computationally expensive components are its calculation of the image gradient and large descriptor dimension. While both descriptors provide quality information of image features, the aforementioned computational factors impact the matching speed significantly.

Binary descriptors such as BRIEF [10], ORB [64] and BRISK [43] have been proposed to speed up the feature descriptor and matching process. The performance boost in binary descriptors comes in the form of using a binary sampling pattern around each image feature previously detected (see Fig 2.6), and outputting a binary vector, instead of computing image gradients and outputting a floating point vector. Each binary descriptor uses its own unique sampling pattern, and outputs a binary string to be used for matching. The matching process is cheaper compared to the HOG based descriptors, because instead of comparing two floating point vectors, comparing binary descriptors is performed by computing the Hamming distance using a XOR or bit count operation, which can be performed extremely quickly on modern CPUs [9].

Following [65], the ORB descriptor was chosen for experimentation. The ORB descriptor is considered as an improvement over BRIEF with the addition of being orientation invariant. However, it can be observed in Fig. 2.7 that the ORB detector from OpenCV strongly favours strong corners over keeping the detected points uniform over the image space. As a result the key-points are mostly clustered around the chessboard observed in the scene. This is in contrast to Grid-FAST discussed in Section. 2.3.1, where the features detected are uniformly distributed over the entire image space. Because of ORB features not being uniformly distributed, the KLT feature tracker is considered and is discussed in the following section.


(a) BRIEF Descriptor [10]


(b) ORB Descriptor [64]



(c) BRISK Descriptor [43]

Figure 2.6: Binary Descriptors

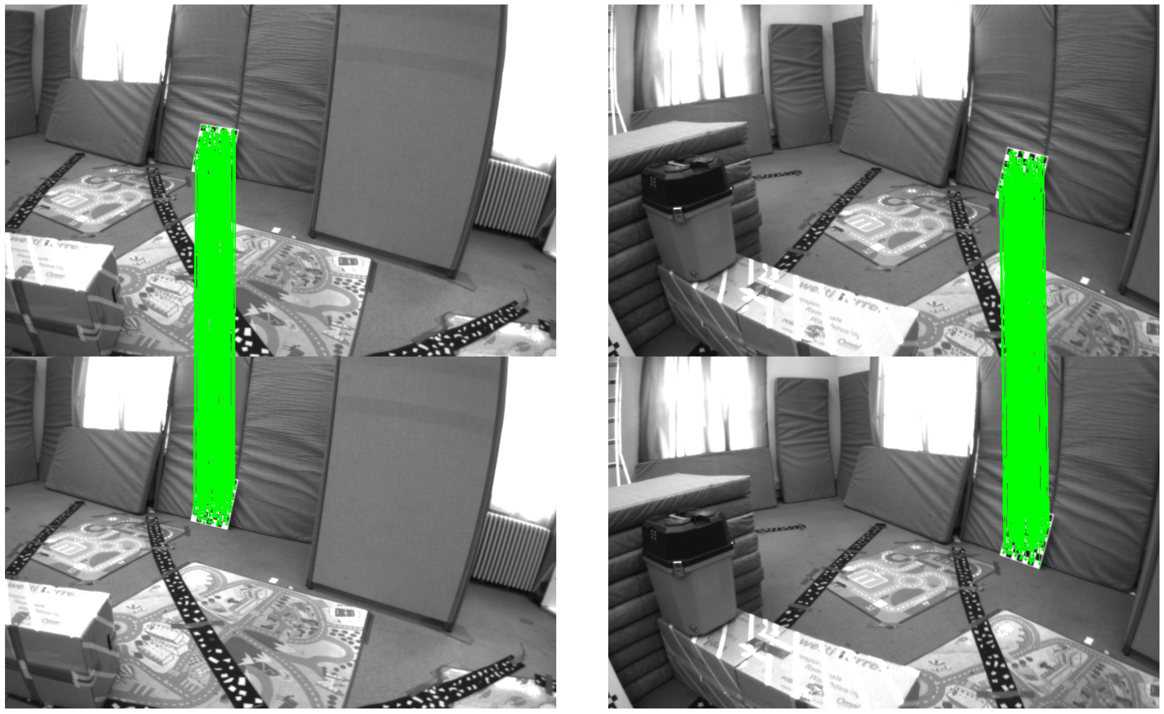


Figure 2.7: ORB descriptors detecting features in the EuRoC MAV dataset [8]

2.3.3 Kanade-Lucas-Tomasi (KLT) Feature Tracker

In the previous section, we showed how ORB descriptors and matching methods are used to correspond and match features across multiple camera frames. This process of corresponding the the same features across multiple camera frames is called feature tracking. There are many different feature tracking pipelines. Matching feature descriptors such as ORB has shown to have better temporal tracking accuracy compared to KLT-based methods [58]. However, in the work of [69], descriptor-based feature trackers were found to require more computational resources with limited gains in tracking accuracy. Making it less attractive for real-time operation. In the following we will briefly describe the KLT feature tracker, but first an understanding of optical flow is required.

Optical Flow

Optical flow estimates the velocity of each image feature in successive images of a scene. It makes the following explicit assumptions:

- Pixel intensity does not change between consecutive frames
- Displacement of features is small
- Features are within the same local neighbourhood

Let us consider a pixel, p, in the first frame which has an intensity, $I(x, y, t)$, where it is a function of the pixel location, x and y, and time, t. If we apply the aforementioned assumptions, we can say that the intensity of said pixel in the first frame to the second does not change. Additionally, if there was a small displacement, dx and dy, and small time difference, dt, between images this can be written in mathematical form as,

$$I(x, y, t) = I(x + dx, y + dy, t + dt). \tag{2.7}$$

This is known as the brightness constancy equation. To obtain the image gradient and velocity of the pixel, we can use Taylor series approximation of right-hand side of Eq. (2.7) to get,

$$I(x + dx, y + dy, t + dt) = I(x, y, t) + \frac{\partial I}{\partial x}dx + \frac{\partial I}{\partial y}dy + \frac{\partial I}{\partial t}dt + \text{H.O.T}, \tag{2.8}$$

removing common terms and dividing by dt we get,

$$I_x v_x + I_y v_y + I_t = 0 \tag{2.9}$$

or,

$$I_x v_x + I_y v_y = -I_t \tag{2.10}$$

where:

$$I_x = \frac{\partial I}{\partial x}; \quad I_y = \frac{\partial I}{\partial y}$$
$$v_x = \frac{dx}{dt}; \quad v_y = \frac{dy}{dt}.$$

The image gradients along the x and y directions are I_x and I_y, where I_t is the image gradient along time, finally, v_x and v_y are the pixel velocity in x and y directions, which is unknown. The problem with Eq. 2.10 is that it provides a single constraint with two degrees of freedom, and as such requires at least one additional constraint to identify a solution.

The Lucas-Kanade method solves the aperture problem by introducing additional conditions. This method assumes all pixels within a window centered around a pixel p will have similar motion, and that the window size is configurable. For example, a window size of 3×3 around the pixel p, the 9 points within the window should have a similar motion. Using Eq. 2.10, the intensity inside the window must therefore satisfy,

$$\begin{gathered} I_x(p_1)v_x(p_1) + I_y(p_1)v_y = -I_t(p_1) \\ I_x(p_1)v_x(p_2) + I_y(p_2)v_y = -I_t(p_2) \\ \vdots \\ I_x(p_1)v_x(p_n) + I_y(p_n)v_y = -I_t(p_n) \end{gathered}$$

where $p_1, p_2, \ldots, p_n$ are the pixels in the window. This can be re-written in matrix form $\mathbf{Ax} = \mathbf{b}$ as,

$$\mathbf{A} = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix} \quad \mathbf{x} = \begin{bmatrix} v_x \\ v_y \end{bmatrix} \quad \mathbf{b} = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{bmatrix}. \tag{2.11}$$

The linear system of equations of Eq. 2.11 is over-determined, therefore there is no exact solution. To address this issue, a least squares method can be used to approximate the solution by applying the ordinary least squares. For the system $\mathbf{Ax} = \mathbf{b}$, the least squares formula is obtained by minimizing the following,

$$\underset{\mathbf{x}}{\operatorname{argmin}} \; ||\mathbf{Ax} - \mathbf{b}||, \tag{2.12}$$

the solution of which can be obtained by using *normal equations*,

$$\mathbf{A}^T\mathbf{A}\mathbf{x} = \mathbf{A}^T\mathbf{b} \tag{2.13}$$

$$\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}. \tag{2.14}$$

Rewriting Eq 2.11 in the form of Eq. 2.14 we get,

$$\begin{bmatrix} v_x \\ v_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(p_i)^2 & \sum_i I_x(p_i)I_y(p_i) \\ \sum_i I_x(p_i)I_y(p_i) & \sum_i I_y(p_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(p_i)I_t(p_i) \\ -\sum_i I_y(p_i)I_t(p_i) \end{bmatrix} \tag{2.15}$$

which is finally used to obtain the optical flow of pixel p.

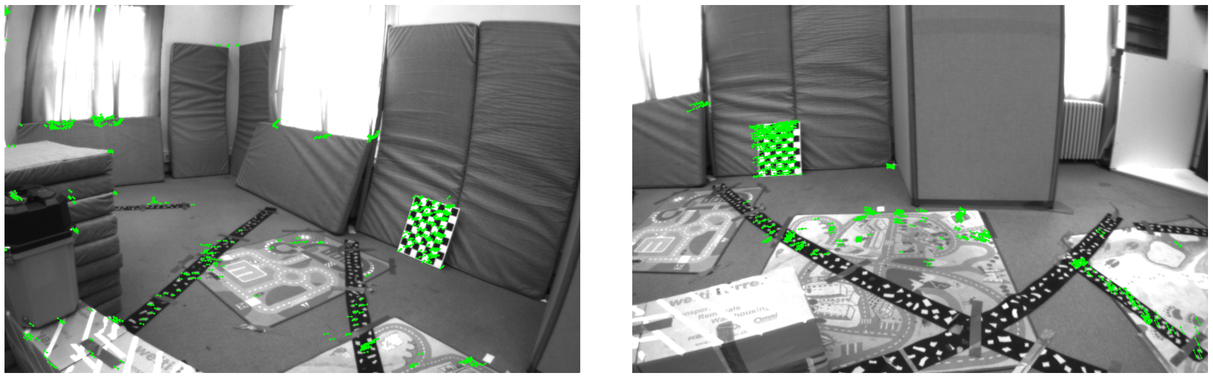


Figure 2.8: KLT Feature Tracker in Action

KLT Feature Tracker

The Lucas-Kanade method recovers feature pixel velocities from consecutive camera frames. The issue with the Lucas-Kanade method is that it assumes the features detected have small displacements between consecutive camera frames. Therefore to track features that have a large motion the Kanade-Lucas-Tomasi (KLT) feature tracker [49] uses reduced-scale versions of the input images in order to track features over multiple camera frames. The general steps of the KLT feature tracker is as follows,

1. Detect corners in the first camera frame
2. For each corners, compute the motion between consecutive camera frames using a pyramidal implementation of the Lucas-Kanade method

3. Match motion vectors between consecutive camera frames to track corners
4. Detect new corners if number of tracks currently tracking is too low
5. Repeat steps 2 to 4

2.4 Feature Estimation

In the previous section we have discussed methods to detect and match between different camera frames. In this section we introduce a method to estimate feature positions in 3D using 2D pixels measurements of the same feature in different camera frames.

2.4.1 Feature Initialization

There are various methods for initializing the feature position. The linear triangulation method [31] is frequently used. This method assumes a pair of homogeneous pixel measurements $\mathbf{z}$ and $\mathbf{z}' \in \mathbb{R}^3$ that observes the same feature, $\mathbf{X}$, in homogeneous coordinates from two different camera frames. The homogeneous projection from 3D to 2D with a known camera matrix $\mathbf{P} \in \mathbb{R}^{3\times 4}$ for each measurement is given as,

$$\begin{aligned} \mathbf{z} &= \mathbf{P}\mathbf{X} \\ \mathbf{z}' &= \mathbf{P}'\mathbf{X}. \end{aligned} \tag{2.16}$$

These equations can be combined to form a system of equations of the form $\mathbf{Ax} = \mathbf{0}$. To eliminate the homogeneous scale factor we apply a cross product to give three equations for each image point, for example $\mathbf{z} \times (\mathbf{PX}) = \mathbf{0}$ writing this out gives

$$\begin{aligned} x(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{1T}\mathbf{X}) &= 0 \\ y(\mathbf{p}^{3T}\mathbf{X}) - (\mathbf{p}^{2T}\mathbf{X}) &= 0 \\ x(\mathbf{p}^{2T}\mathbf{X}) - y(\mathbf{p}^{1T}\mathbf{X}) &= 0 \end{aligned} \tag{2.17}$$

where $\mathbf{p}^{iT}$ is the i^{th} row of $\mathbf{P}$.

From Eq. (2.17), an equation of the form $\mathbf{Ax} = \mathbf{0}$ for each image point can be formed, where $\mathbf{x}$ represents the unknown homogeneous feature location to be estimated, and $\mathbf{A}$ is given as

$$\mathbf{A} = \begin{bmatrix} x(\mathbf{p}^{3T}) - (\mathbf{p}^{1T}) \\ y(\mathbf{p}^{3T}) - (\mathbf{p}^{2T}) \\ x'(\mathbf{p}'^{3T}) - (\mathbf{p}'^{1T}) \\ y'(\mathbf{p}'^{3T}) - (\mathbf{p}'^{2T}) \end{bmatrix} \tag{2.18}$$

giving a total of four equations in four homogeneous unknowns. Solving for **A** using SVD allows us to estimate the initial feature location.

2.4.2 Feature Refinement

The linear triangulation provides an initial feature position. In an ideal world, feature positions can be solved as a system of equations. In reality, however, errors are present in the camera poses and pixel measurements. The pixel measurements observing the same 3D point are generally noisy. In addition, the pinhole camera model and radial-tangential distortion model do not perfectly model the camera projection or distortion observed. Therefore an iterative method can be used to further refine the feature position. This problem is generally formulated as a non-linear least square problem and can be solved by numerical methods, such as the Gauss-Newton algorithm.

Let us consider the case where two different cameras at known locations observe the same feature in the scene. Our goal is to optimize for the feature position. With an i^{th} feature measurement, and corresponding i^{th} camera orientation as a quaternion, $\mathbf{q}_{C_iG}$ and i^{th} camera position, $\mathbf{p}_{C_i}^{C_iG}$, we can formulate the re-projection error as,

$$\mathbf{e}_i(\boldsymbol{\theta}, \mathbf{q}_{C_iG}, \mathbf{p}_{C_i}^{C_iG}) = \mathbf{z}_i - \mathbf{h}(\boldsymbol{\theta}, \mathbf{q}_{C_iG}, \mathbf{p}_{C_i}^{C_iG}). \tag{2.19}$$

where the parameter, $\boldsymbol{\theta} \in \mathbb{R}^3$, represents the feature location to be optimized, $\mathbf{z}_i \in \mathbb{R}^2$ is the i-th pixel measurement, and $\mathbf{h} : \mathbb{R}^{10} \mapsto \mathbb{R}^2$ is the projection function that projects $\boldsymbol{\theta}$ into the measurement space using known camera extrinsics. The re-projection error is a geometric error corresponding to the Euclidean distance between measured and projected features onto the image plane, it is used for quantifying the error of the estimated feature location. With the error function, $\mathbf{e}_i$, for one measurement, a cost function, $\mathbf{f}$, for m number of pixel measurements in the form of sum of squares can be defined as,

$$\mathbf{f}(\boldsymbol{\theta}) = \sum_{i=1}^{m} \mathbf{e}_i(\boldsymbol{\theta}, \mathbf{q}_{C_iG}, \mathbf{p}_{C_i}^{C_iG})^T \mathbf{e}_i(\boldsymbol{\theta}, \mathbf{q}_{C_iG}, \mathbf{p}_{C_i}^{C_iG}). \tag{2.20}$$

Now that the cost function, $\mathbf{f}(\boldsymbol{\theta})$, is defined an unconstrained optimization can be performed using the cost function Eq. 2.20 to estimate the optimal feature position, $\boldsymbol{\theta}^*$, which minimizes the re-projection error over the set of collected measurements.

$$\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmin}}\ \mathbf{f}(\boldsymbol{\theta}). \tag{2.21}$$

2.5 Error State Kalman Filter

The Error State Kalman Filter (ESKF) is an Extended Kalman Filter (EKF) formulation that is commonly used in the presence of an IMU. Compared to the conventional Kalman filter formulation, the ESKF differs in two major ways. First, ESKF does not use a motion model of the system it is estimating the state of. Instead it uses an IMU model as a surrogate, and errors in the IMU integrated attitude, position and velocity are among the estimated variables of the ESKF. This formulation is called the error state space (indirect) formulation [53]. The rationale for using an IMU model as the process model of the filter stems from the benefit of circumventing the need to model the dynamics of the system at hand [63].

Secondly, the absence of a globally non-singular three parameter representation of rotations means a Kalman filter either has to estimate a singular (Euler angles), or a redundant attitude representation (quaternion or rotation matrix). The work of [52] was the first to explicitly introduce the idea of using a non-singular representation of attitude with a quaternion as a reference, and a three-component representation for the deviations from this reference attitude. This three-component representation is estimated in the ESKF formulation, instead of the the quaternion reference attitude. In order to avoid any singularity or discontinuity for the three dimensional parametrization, small deviations or errors away from the quaternion reference attitude are assumed.

The ESKF has the notion of true, nominal and error state values, where the true state values can be expressed as a suitable composition (be it linear sum, quaternion product or matrix product) of the nominal and error-states [68]. The nominal state does not have any noise terms or other model imperfections, the noise and errors are represented by the error state $\delta\mathbf{x}$ and estimated with the ESKF.

In the context of using an IMU as the process model for the ESKF, the high frequency IMU measurements are integrated into a nominal state $\mathbf{x}$ and the noise and errors of the IMU are estimated, which occur in parallel. The errors of the filter are corrected at the arrival of external sensor information other than the IMU (since the IMU model is the process model of the filter), such as GPS, vision, etc. In the following we will derive the true, nominal, and error kinematics and demonstrate the general form of an ESKF.

2.5.1 Inertial Measurement Unit (IMU)

The Inertial Measurement Unit (IMU) is an electronic device that measures the force, angular rate, and in some cases the magnetic field from the device's perspective. These devices usually comprises of an accelerometer and gyroscope, and in some devices a magnetometer is included as well. Modern IMUs are small, lightweight and cheap to manufacture, they are prevalent in, among other things, smartphones and aerial vehicles. In this section, only the accelerometer and gyroscope are considered. Magnetometers are not considered because they are easily influenced by nearby magnetic fields that are difficult to model.

Accelerometer

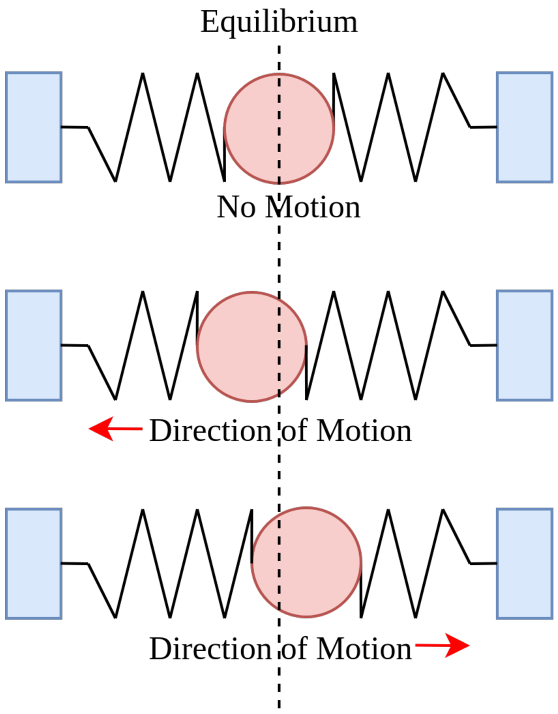


Figure 2.9: Accelerometer Mass Spring System

An accelerometer measures the force exerted on the IMU. At rest on the surface of the earth, an accelerometer measures one gravitational unit (1g), or approximately 9.81m/s^2, directly upwards. Internally, a Microelectromechanical Systems (MEMS) MEMS accelerometer operates similar to a spring-mass system where measurements are made in the direction that the mass is being pulled to (see Fig. 2.9). A 3-axis accelerometer model that contains noise and bias is given as,

$$\mathbf{a}_m = \mathbf{C}_{IG}(\mathbf{a}_G^{IG} - \mathbf{g}) + \mathbf{n}_a + \mathbf{b}_a, \tag{2.22}$$

where $\mathbf{a}_m$ is the measured acceleration, $\mathbf{C}_{IG}$ is the rotation from global to IMU frame, $\mathbf{a}_G^{IG}$ is the true acceleration of the IMU in the global frame, $\mathbf{g}$ is the true acceleration of gravity (usually $\approx 9.81\text{m/s}^2$). The accelerometer noise, $\mathbf{n}_a \sim \mathcal{N}(0, \mathbf{N}_a)$, is a zero mean normally distributed random variable. The accelerometer bias, $\mathbf{b}_a$, changes over time and is modeled as a random walk process driven by an independent noise vector, $\mathbf{n}_{wa} \sim \mathcal{N}(0, \mathbf{N}_{wa})$.

Gyroscope

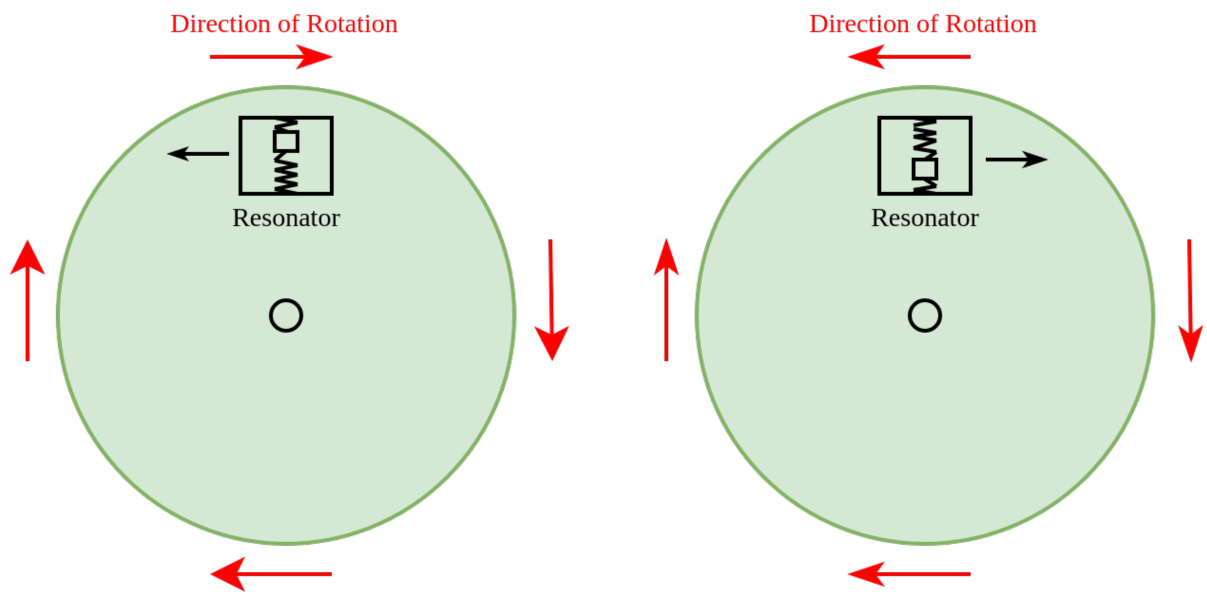


Figure 2.10: Internal Operational View of a MEMS Gyroscope Sensor

A gyroscope measures angular velocity of the IMU. It is used to sense rate of rotations around a particular axis and can be used to obtain the orientation of the IMU. Internally, a MEMS gyroscope contains a small resonating mass that is shifted as the angular velocity changes, and this movement is then converted a angular velocity measurement (see Fig. 2.10). A 3-axis gyroscope with noise and bias is given as,

$$\boldsymbol{\omega}_m = \boldsymbol{\omega}_G^{IG} + \mathbf{n}_g + \mathbf{b}_g \tag{2.23}$$

where $\boldsymbol{\omega}_m$ is the measured angular velocity, $\boldsymbol{\omega}_G^{IG}$ is the IMU angular velocity expressed in the global frame. The gyroscope noise, $\mathbf{n}_g \sim \mathcal{N}(0, \mathbf{N}_g)$ is a zero mean normally distributed random variable. The gyroscope bias, $\mathbf{b}_g$, changes over time and is modeled as a random walk process driven by an independent noise vector, $\mathbf{n}_{wg} \sim \mathcal{N}(0, \mathbf{N}_{wg})$. It is worth noting for a MEMS gyroscope the bias and white noise is greater than the effect of the Earth's rotation, and so for this work, it is neglected.

2.5.2 The True State Kinematics

The true kinematic equations for an IMU are given by [68]. They describe the time evolution of an IMU's orientation, velocity, position as well as the gyroscope bias and accelerometer bias with equations,

$$\begin{aligned} \dot{\mathbf{q}}_{IG} &= \frac{1}{2}\mathbf{q}_{IG} \otimes \boldsymbol{\omega}_G^{IG} \\ \dot{\mathbf{b}}_g &= \mathbf{n}_g \\ \dot{\mathbf{v}}_G^{IG} &= \mathbf{a}_G^{IG} \\ \dot{\mathbf{b}}_a &= \mathbf{n}_a \\ \dot{\mathbf{p}}_G^{IG} &= \mathbf{v}_G^{IG} \end{aligned} \tag{2.24}$$

where $\mathbf{q}_{IG}$ represents the true quaternion, $\boldsymbol{\omega}_G^{IG}$ and $\mathbf{b}_g$ are the true angular velocity and true angular velocity bias, $\mathbf{v}_G^{IG}$ is the true velocity, $\mathbf{a}_G^{IG}$, $\mathbf{b}_a$ and $\mathbf{n}_a$ are the true acceleration, true acceleration bias and acceleration noise. The symbol $\otimes$ in Eq. 2.24 denotes a quaternion multiplication.

By rearranging Eq. (2.22) and Eq. (2.23) the true acceleration and angular velocity expressions can be obtained,

$$\mathbf{a}_G^{IG} = \mathbf{C}\{\mathbf{q}_{IG}\}^T(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) + \mathbf{g} \tag{2.25}$$

$$\boldsymbol{\omega}_G^{IG} = \boldsymbol{\omega}_m - \mathbf{b}_g - \mathbf{n}_g \tag{2.26}$$

Substituting Eq. (2.25) and Eq. (2.26) gives the final kinematic system as,

$$\begin{aligned}\dot{\mathbf{q}}_{IG} &= \frac{1}{2}\mathbf{q}_{IG} \otimes (\boldsymbol{\omega}_m - \mathbf{b}_g - \mathbf{n}_g)\\ \dot{\mathbf{b}}_g &= \mathbf{n}_g\\ \dot{\mathbf{v}}_G^{IG} &= \mathbf{C}\{\mathbf{q}_{IG}\}(\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) + \mathbf{g}\\ \dot{\mathbf{b}}_a &= \mathbf{n}_a\\ \dot{\mathbf{p}}_G^{IG} &= \mathbf{v}_G^{IG}.\end{aligned} \tag{2.27}$$

2.5.3 The Nominal State Kinematics

The nominal state kinematics is the modelled system without any noise terms or model imperfections. By removing all noise terms in Eq. (2.27) we obtain

$$\begin{aligned}\dot{\hat{\mathbf{q}}}_{IG} &= \frac{1}{2}\hat{\mathbf{q}}_{IG} \otimes (\boldsymbol{\omega}_m - \hat{\mathbf{b}}_g)\\ \dot{\hat{\mathbf{b}}}_g &= \mathbf{0}\\ \dot{\hat{\mathbf{v}}}_G^{IG} &= \mathbf{C}\{\hat{\mathbf{q}}_{IG}\}(\mathbf{a}_m - \hat{\mathbf{b}}_a) + \mathbf{g}\\ \dot{\hat{\mathbf{b}}}_a &= \mathbf{0}\\ \dot{\hat{\mathbf{p}}}_G^{IG} &= \hat{\mathbf{v}}_G^{IG}\end{aligned} \tag{2.28}$$

Note, variables with the hat, such as $\hat{\mathbf{q}}_{IG}$, denotes a nominal state variable. The above nominal-state kinematics are used to integrate the IMU measurements as the process model in the Kalman filter.

2.5.4 The Error State Kinematics

In the previous subsection, we removed the noise terms in Eq (2.27) to obtain the nominal state kinematics Eq. (5.9) and form the process model of the Kalman filter. Here we will write the full error-state dynamic system given as

$$\delta\dot{\boldsymbol{\theta}}_I = -\lfloor \boldsymbol{\omega}_G^{IG} \times \rfloor \delta\boldsymbol{\theta}_I - \delta\mathbf{b}_g - \mathbf{n}_g \tag{2.29}$$

$$\delta\dot{\mathbf{b}}_g = \mathbf{n}_g \tag{2.30}$$

$$\delta\dot{\mathbf{v}}_G^{IG} = -\mathbf{C}\{\mathbf{q}_{IG}\}\lfloor \mathbf{a}_m - \delta\mathbf{b}_a \ \times\rfloor \delta\boldsymbol{\theta}_I - \mathbf{C}\{\mathbf{q}_{IG}\}\delta\mathbf{b}_a + \delta\mathbf{g} - \mathbf{C}\{\mathbf{q}_{IG}\}\mathbf{n}_a \tag{2.31}$$

$$\delta\dot{\mathbf{b}}_a = \mathbf{n}_a \tag{2.32}$$

$$\delta\dot{\mathbf{p}}_G^{IG} = \delta\mathbf{v}_G^{IG}. \tag{2.33}$$

To reiterate, the nominal-state kinematics in the error-state space formulation is the process model of the filter and has no errors or noise modelled. Our goal is to estimate the *errors* in the nominal-state kinematics (or errors in the IMU model) as part of the Kalman filter state vector, the error-state kinematics above will be used to form the Kalman filter transition matrix $\mathbf{F}$.

In the error-state kinematics, Eq. (2.30), Eq. (2.32) and Eq. (2.33) are the error dynamics of gyroscope bias, accelerometer bias and position respectively, they are trivial linear equations of the form $\mathbf{x}_t = \mathbf{x} + \delta\mathbf{x}$. Eq (2.29) and Eq (2.31), the error dynamics of orientation and velocity, however, are non-trivial due to their non-linearity. The derivations are presented as follows.

Derivation of Orientation Error $\delta\dot{\boldsymbol{\theta}}_I$

In the following we wish to derive the kinematics of the angular errors $\delta\dot{\boldsymbol{\theta}}_I$. We will start with the definition of the true quaternion as a quaternion multiplication of the error and nominal quaternion, and its derivative as follows

$$\mathbf{q}_{IG} = \delta\mathbf{q}_{IG} \otimes \hat{\mathbf{q}}_{IG} \tag{2.34}$$

$$\dot{\mathbf{q}}_{IG} = \delta\dot{\mathbf{q}}_{IG} \otimes \hat{\mathbf{q}}_{IG} + \delta\mathbf{q}_{IG} \otimes \dot{\hat{\mathbf{q}}}_{IG}. \tag{2.35}$$

Substituting $\dot{\mathbf{q}}_{IG}$ and $\dot{\hat{\mathbf{q}}}_{IG}$ in Eq. 2.35 with the following alternative form of quaternion derivative

$$\dot{\mathbf{q}}_{IG} = \frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_G^{IG} \\ 0\end{bmatrix} \otimes \mathbf{q}_{IG} \tag{2.36}$$

$$\dot{\hat{\mathbf{q}}}_{IG} = \frac{1}{2}\begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix} \otimes \hat{\mathbf{q}}_{IG} \tag{2.37}$$

leads to

$$\frac{1}{2}\begin{bmatrix}\boldsymbol{\omega}_G^{IG} \\ 0\end{bmatrix} \otimes \mathbf{q}_{IG} = \delta\dot{\mathbf{q}}_{IG} \otimes \hat{\mathbf{q}}_{IG} + \delta\mathbf{q}_{IG} \otimes \frac{1}{2}\begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix} \otimes \hat{\mathbf{q}}_{IG}. \tag{2.38}$$

Collecting $\frac{1}{2}$ terms by applying $-\frac{1}{2}(\delta\mathbf{q}_{IG} \otimes \begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix} \otimes \hat{\mathbf{q}}_{IG})$ to both sides yields,

$$\delta\dot{\mathbf{q}}_{IG} \otimes \hat{\mathbf{q}}_{IG} = \frac{1}{2}\left(\begin{bmatrix}\boldsymbol{\omega}_G^{IG} \\ 0\end{bmatrix} \otimes \mathbf{q}_{IG} - \delta\mathbf{q}_{IG} \otimes \begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix} \otimes \hat{\mathbf{q}}_{IG}\right). \tag{2.39}$$

Finally, removing $\hat{\mathbf{q}}_{IG}$ on both sides by $\otimes\hat{\mathbf{q}}_{IG}^{-1}$ gives us

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\left(\begin{bmatrix}\boldsymbol{\omega}_G^{IG} \\ 0\end{bmatrix} \otimes \delta\mathbf{q}_{IG} - \delta\mathbf{q}_{IG} \otimes \begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix}\right) \tag{2.40}$$

To recap, the definition of the derivative of the true quaternion defined in Eq. (2.35) is used to remove dependence on the nominal quaternion terms $\mathbf{q}_{IG}$ and $\hat{\mathbf{q}}_{IG}$. Then an expression for the derivative of the error quaternion dynamics $\dot{\mathbf{q}}_{IG}$ was found, with only quaternion error $\delta\mathbf{q}_{IG}$ and true angular velocity $\boldsymbol{\omega}_G^{IG}$ terms in the expression. This expression, however, still poses a problem, as the true angular velocity $\boldsymbol{\omega}_G^{IG}$ is unknown. To resolve this issue, the gyroscope model can be used to remove the dependence on the true angular velocity,

$$\boldsymbol{\omega}_m = \boldsymbol{\omega}_G^{IG} + \mathbf{b}_g + \mathbf{n}_g \tag{2.41}$$

and the definition for the nominal angular velocity $\hat{\boldsymbol{\omega}}_I^{IG}$,

$$\hat{\boldsymbol{\omega}}_I^{IG} = \boldsymbol{\omega}_m - \mathbf{b}_g \tag{2.42}$$

Combined to form

$$\boldsymbol{\omega}_G^{IG} = \hat{\boldsymbol{\omega}}_I^{IG} - \mathbf{b}_g - \mathbf{n}_g, \tag{2.43}$$

which is substituted Eq. (2.43) into Eq. (2.40) to get

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\left(\begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} - \mathbf{b}_g - \mathbf{n}_g \\ 0\end{bmatrix} \otimes \delta\mathbf{q}_{IG} - \delta\mathbf{q}_{IG} \otimes \begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix}\right). \tag{2.44}$$

Collecting the $\hat{\boldsymbol{\omega}}_I^{IG}$ terms together, while factoring out the gyroscope bias, $\mathbf{b}_g$, and gyroscope noise, $\mathbf{n}_g$, to get,

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\left(\begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix} \otimes \delta\mathbf{q}_{IG} - \begin{bmatrix}\mathbf{b}_g + \mathbf{n}_g \\ 0\end{bmatrix} \otimes \delta\mathbf{q}_{IG} - \delta\mathbf{q}_{IG} \otimes \begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix}\right) \tag{2.45}$$

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\left(\begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix} \otimes \delta\mathbf{q}_{IG} - \delta\mathbf{q}_{IG} \otimes \begin{bmatrix}\hat{\boldsymbol{\omega}}_I^{IG} \\ 0\end{bmatrix}\right) - \frac{1}{2}\begin{bmatrix}\mathbf{b}_g + \mathbf{n}_g \\ 0\end{bmatrix} \otimes \delta\mathbf{q}_{IG} \tag{2.46}$$

Using the left and right quaternion composite identities, Eq. 2.46 becomes,

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\left(\begin{bmatrix} -\lfloor\hat{\boldsymbol{\omega}}_I^{IG} \times\rfloor & \hat{\boldsymbol{\omega}}_I^{IG} \\ -\hat{\boldsymbol{\omega}}_I^{IGT} & 0 \end{bmatrix} \cdot \delta\mathbf{q}_{IG} - \begin{bmatrix} +\lfloor\hat{\boldsymbol{\omega}}_I^{IG} \times\rfloor & \hat{\boldsymbol{\omega}}_I^{IG} \\ -\hat{\boldsymbol{\omega}}_I^{IGT} & 0 \end{bmatrix} \cdot \delta\mathbf{q}_{IG}\right) - \frac{1}{2}\begin{bmatrix} \mathbf{b}_g + \mathbf{n}_g \\ 0 \end{bmatrix} \otimes \delta\mathbf{q}_{IG}, \tag{2.47}$$

where $\lfloor\boldsymbol{\omega}\ \times\rfloor$ is the skew symmetric operator and defined as,

$$\lfloor\boldsymbol{\omega}\ \times\rfloor = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & \omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{2.48}$$

Simplifying Eq. (2.47),

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\begin{bmatrix} -2\lfloor\hat{\boldsymbol{\omega}}_I^{IG}\ \times\rfloor & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1}^T & 0 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} \mathbf{b}_g + \mathbf{n}_g \\ 0 \end{bmatrix} \otimes \delta\mathbf{q}_{IG} \tag{2.49}$$

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\begin{bmatrix} -2\lfloor\hat{\boldsymbol{\omega}}_I^{IG}\ \times\rfloor & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1}^T & 0 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} -\lfloor(\mathbf{b}_g + \mathbf{n}_g)\ \times\rfloor & (\mathbf{b}_g + \mathbf{n}_g) \\ -(\mathbf{b}_g + \mathbf{n}_g)^T & 0 \end{bmatrix}\begin{bmatrix} \delta\mathbf{q}_{IG} \\ 1 \end{bmatrix} \tag{2.50}$$

$$\delta\dot{\mathbf{q}}_{IG} = \frac{1}{2}\begin{bmatrix} -2\lfloor\hat{\boldsymbol{\omega}}_I^{IG}\ \times\rfloor & \mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1}^T & 0 \end{bmatrix} - \frac{1}{2}\begin{bmatrix} \mathbf{b}_g + \mathbf{n}_g \\ 0 \end{bmatrix} + \text{H.O.T} \tag{2.51}$$

Ignoring second order terms, we can write

$$\delta\dot{\mathbf{q}}_{IG} \approx \begin{bmatrix} \delta\dot{\mathbf{q}}_{IG_r} \\ \delta\dot{q}_w \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_I \\ \dot{1} \end{bmatrix} = \begin{bmatrix} -\lfloor\hat{\boldsymbol{\omega}}_I^{IG}\ \times\rfloor\delta\mathbf{q}_{IG_r} - \frac{1}{2}(\mathbf{b}_g + \mathbf{n}_g) \\ 0 \end{bmatrix}, \tag{2.52}$$

or finally,

$$\delta\dot{\boldsymbol{\theta}}_I \approx -\lfloor\boldsymbol{\omega}_G^{IG}\ \times\rfloor\delta\boldsymbol{\theta}_I - \mathbf{b}_g + \mathbf{n}_g. \tag{2.53}$$

Derivation of Linear Velocity Error $\delta\dot{\mathbf{v}}_G^{IG}$

To derive the linear velocity error $\delta\dot{\mathbf{v}}_G^{IG}$ we begin with the nominal velocity $\hat{\mathbf{v}}_G^{IG}$

$$\dot{\hat{\mathbf{v}}}_G^{IG} = \mathbf{C}\{\mathbf{q}_{IG}\}(\mathbf{a}_m - \mathbf{b}_a) + \mathbf{g} \tag{2.54}$$

and introduce the nominal and error body accelerations, $\hat{\mathbf{a}}_B$ and $\delta\mathbf{a}_B$ respectively defined as

$$\hat{\mathbf{a}}_B \triangleq \mathbf{a}_m - \mathbf{b}_a \tag{2.55}$$

$$\delta \mathbf{a}_B \triangleq -\delta \mathbf{b}_a - \mathbf{n}_a \tag{2.56}$$

into Eq. (2.54) to give

$$\dot{\hat{\mathbf{v}}}_G^{IG} = \mathbf{C}\{\mathbf{q}_{IG}\}\hat{\mathbf{a}}_B + \mathbf{g}. \tag{2.57}$$

The expression for the true acceleration $\mathbf{a}_G^{IG}$ in the inertial frame can be written as a composition of the true rotation $\mathbf{C}_{IG}$, nominal and error body accelerations $\hat{\mathbf{a}}_B$ and $\delta\mathbf{a}_B$, and nominal and error gravitational acceleration terms $\hat{\mathbf{g}}$ and $\delta\mathbf{g}$ as,

$$\mathbf{a}_G^{IG} = \mathbf{C}_{IG}(\hat{\mathbf{a}}_B + \delta\mathbf{a}_B) + \hat{\mathbf{g}} + \delta\mathbf{g}. \tag{2.58}$$

We continue by writing an expression for $\dot{\hat{\mathbf{v}}}_G^{IG}$ into two different forms (left and right developments)

$$\dot{\hat{\mathbf{v}}}_G^{IG} + \delta\dot{\mathbf{v}}_G^{IG} = \mathbf{C}_{IG}(\hat{\mathbf{a}}_B + \delta\mathbf{a}_B) + \hat{\mathbf{g}} + \delta\mathbf{g} \tag{2.59}$$

Using the small angle approximation, the true rotation $\mathbf{C}_{IG} \approx \mathbf{C}_{IG}(\mathbf{I}_{3\times3} + \lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor)$, Eq. (2.59) becomes,

$$\dot{\hat{\mathbf{v}}}_G^{IG} + \delta\dot{\mathbf{v}}_G^{IG} = \mathbf{C}_{IG}(\mathbf{I}_{3\times3} + \lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor)(\hat{\mathbf{a}}_B + \delta\mathbf{a}_B) + \hat{\mathbf{g}} + \delta\mathbf{g} \tag{2.60}$$

$$\dot{\hat{\mathbf{v}}}_G^{IG} + \delta\dot{\mathbf{v}}_G^{IG} = \left(\mathbf{C}_{IG} + \mathbf{C}_{IG}\lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\right)\left(\hat{\mathbf{a}}_B + \delta\mathbf{a}_B\right) + \hat{\mathbf{g}} + \delta\mathbf{g} \tag{2.61}$$

$$\mathbf{C}_{IG}\hat{\mathbf{a}}_B + \hat{\mathbf{g}} + \delta\dot{\mathbf{v}}_G^{IG} = \mathbf{C}_{IG}\hat{\mathbf{a}}_B + \mathbf{C}_{IG}\delta\hat{\mathbf{a}}_B + \mathbf{C}_{IG}\lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\hat{\mathbf{a}}_B + \mathbf{C}_{IG}\lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\delta\mathbf{a}_B + \hat{\mathbf{g}} + \delta\mathbf{g}. \tag{2.62}$$

After removing $\mathbf{C}_{IG}\hat{\mathbf{a}}_B + \hat{\mathbf{g}}$ from both sides we are left with

$$\delta\dot{\mathbf{v}}_G^{IG} = \mathbf{C}_{IG}\delta\mathbf{a}_B + \mathbf{C}_{IG}\lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\hat{\mathbf{a}}_B + \mathbf{C}_{IG}\lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\delta\mathbf{a}_B + \delta\mathbf{g} \tag{2.63}$$

$$\delta\dot{\mathbf{v}}_G^{IG} = \mathbf{C}_{IG}(\delta\mathbf{a}_B + \lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\hat{\mathbf{a}}_B) + \mathbf{C}_{IG}\lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\delta\mathbf{a}_B + \delta\mathbf{g}. \tag{2.64}$$

Eliminating $\mathbf{C}_{IG}\lfloor\delta\boldsymbol{\theta}_I \ \times\rfloor\delta\mathbf{a}_B$ in Eq (2.64) and reorganizing the cross products (with $\lfloor\mathbf{a} \ \times\rfloor\mathbf{b} = -\lfloor\mathbf{b} \ \times\rfloor\mathbf{a}$) gives

$$\delta\dot{\mathbf{v}}_G^{IG} = \mathbf{C}_{IG}(\delta\mathbf{a}_B - \lfloor\hat{\mathbf{a}}_B \ \times\rfloor\delta\boldsymbol{\theta}_I) + \delta\mathbf{g}, \tag{2.65}$$

then we substitute Eq. (2.55) and Eq. (2.56) into Eq. (2.65),

$$\delta\dot{\mathbf{v}}_G^{IG} = \mathbf{C}_{IG}(-\delta\mathbf{b}_a - \mathbf{n}_a - \lfloor\mathbf{a}_m - \mathbf{b}_a \ \times\rfloor\delta\boldsymbol{\theta}_I) + \delta\mathbf{g}, \tag{2.66}$$

or equivalently,

$$\delta\dot{\mathbf{v}}_G^{IG} = -\mathbf{C}_{IG}\lfloor\mathbf{a}_m - \mathbf{b}_a \ \times\rfloor\delta\boldsymbol{\theta}_I - \mathbf{C}_{IG}\delta\mathbf{b}_a + \delta\mathbf{g} - \mathbf{C}_{IG}\mathbf{n}_a. \tag{2.67}$$

We can further simplify the acceleration noise term $\mathbf{C}_{IG}\mathbf{n}_a$ in Eq. (2.67), if we assume the acceleration noise is white, uncorrelated and isotropic, then we can redefine the noise with no consequence according to,

$$\mathbf{n}_a = \mathbf{C}_{IG}\mathbf{n}_a, \tag{2.68}$$

and end up with,

$$\delta\dot{\mathbf{v}}_G^{IG} = -\mathbf{C}_{IG}\lfloor \mathbf{a}_m - \mathbf{b}_a \ \times\rfloor \delta\boldsymbol{\theta}_I - \mathbf{C}_{IG}\delta\mathbf{b}_a + \delta\mathbf{g} - \mathbf{n}_a. \tag{2.69}$$

2.5.5 ESKF Prediction Equations

In the previous sub-sections, the nominal and error states were introduced. In following we derive the error state Jacobian matrices to perform EKF prediction. Let the nominal state vector $\hat{\mathbf{x}}$, error state vector $\delta\mathbf{x}$, input vector $\mathbf{u}$, and the noise vector $\mathbf{n}_I$ be

$$\hat{\mathbf{x}} = \begin{bmatrix} \hat{\mathbf{q}}_{IG} \\ \hat{\mathbf{b}}_g \\ \hat{\mathbf{v}}_G^{IG} \\ \hat{\mathbf{b}}_a \\ \hat{\mathbf{p}}_G^{IG} \end{bmatrix}, \quad \delta\mathbf{x} = \begin{bmatrix} \delta\mathbf{q}_{IG} \\ \delta\mathbf{b}_g \\ \delta\mathbf{v}_G^{IG} \\ \delta\mathbf{b}_a \\ \delta\mathbf{p}_G^{IG} \end{bmatrix}, \quad \mathbf{u} = \begin{bmatrix} \mathbf{a}_m \\ \boldsymbol{\omega}_m \end{bmatrix}, \quad \mathbf{n}_I = \begin{bmatrix} \mathbf{n}_g \\ \mathbf{n}_{wg} \\ \mathbf{n}_a \\ \mathbf{n}_{wa} \end{bmatrix}. \tag{2.70}$$

Using Eq. (5.8) we can define the linearized continuous-time model for the IMU error-state as

$$\delta\dot{\mathbf{x}} = \mathbf{F}\delta\mathbf{x} + \mathbf{G}\mathbf{n}_I \tag{2.71}$$

By simple inspection of Eq. (2.29) to Eq. (2.33) the transition matrix $\mathbf{F}$ is,

$$\mathbf{F} = \begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}}_I^{IG} \ \times\rfloor & -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -\mathbf{C}\{\hat{\mathbf{q}}_{IG}\}^T\lfloor \hat{\mathbf{a}} \ \times\rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{C}\{\hat{\mathbf{q}}_{IG}\}^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}_{15\times15} \tag{2.72}$$

the Jacobian of the nominal-state kinematics with respect to the noise vector is,

$$\mathbf{G} = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{C}\{\hat{\mathbf{q}}_{IG}\}^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}_{15\times12}$$

The IMU measurements $\boldsymbol{\omega}_m$ and $\mathbf{a}_m$ are sampled with a time period, ΔT, and these measurements are used for state propagation in the EKF. A number of numerical techniques can be used to discretize the continuous-time transition matrix, $\mathbf{F}$. From the simplest integration method such as the Euler method, to the Runge-Kutta method of the 4th or 5th order. For our requirements, we found the 4th order Runge-Kutta numerical integration of the nominal-state kinematics of Eq. (5.9) to be adequate for our applications.

To propagate the uncertainty of the state, the discrete time state transition matrix and discrete time noise covariance matrix needs to be computed first,

$$\mathbf{\Phi}_k = \mathbf{\Phi}(t_{k+1}, t_k) = \exp\left(\int_{t_k}^{t_{k+1}} \mathbf{F}(\tau) d\tau\right) \tag{2.73}$$

$$\mathbf{Q}_k = \int_{t_k}^{t_{k+1}} \mathbf{\Phi}(t_{k+1}, \tau)\mathbf{G}\mathbf{Q}\mathbf{G}\mathbf{\Phi}(t_{k+1}, \tau)^T d\tau \tag{2.74}$$

where $\mathbf{Q} = \mathbf{E}[\mathbf{n}_I\mathbf{n}_I^T]$ is the noise covariance matrix of the system in continuous time. The propagated covariance of the IMU state is then,

$$\mathbf{P}_{k+1|k} = \mathbf{\Phi}_k\mathbf{P}_{k|k}\mathbf{\Phi}_k^T + \mathbf{Q}_k. \tag{2.75}$$

2.5.6 ESKF Measurement Update

As mentioned at the start of this section, the ESKF circumvents the need for a dynamic motion model of the system by using an IMU model as a surrogate. Since the IMU model is used as the process model of the EKF filter, a measurement from a different sensor (secondary sensor), such as GPS or vision is required to perform the measurement update step of the EKF.

The secondary sensor measurements are assume to depend on the state, such as,

$$\mathbf{y} = h(\mathbf{x}) + \mathbf{n}_r \tag{2.76}$$

where $h(\cdot)$ is a non-linear function of the system true-state, and $\mathbf{n}_r$ is assumed to be a zero-mean white Gaussian noise with covariance $\mathbf{R}$,

$$\mathbf{n}_r \sim \mathcal{N}(0, \mathbf{R}). \tag{2.77}$$

Finally, in order to apply the usual EKF filter correction equations,

$$\mathbf{K} = \mathbf{P}\mathbf{H}^T(\mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{C})^{-1} \tag{2.78}$$

$$\begin{aligned} \delta\mathbf{x} &= \mathbf{K}(\mathbf{y} - h(\hat{\mathbf{x}})) && (2.79) \\ \mathbf{P}_{k+1|k+1} &= (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}_{k+1|k}. && (2.80) \end{aligned}$$

We need to define the measurement Jacobian matrix, $\mathbf{H}$, with respect to the error state $\delta\mathbf{x}$, evaluated using the best true-state estimate $\mathbf{x} = \hat{\mathbf{x}} + \delta\mathbf{x}$.

$$\mathbf{H} = \left.\frac{\partial h}{\partial \delta\mathbf{x}}\right|_{\mathbf{x}} \tag{2.81}$$

Chapter 3

Autonomous Landing of a MAV onto a Moving Ground Target

One of the first autonomous landings of a helicopter onto a moving platform is Boeing's Unmanned Little Bird [29], a full sized helicopter that demonstrated consistent landing on a trailer towed behind a truck and landing on a ship at sea. Their system relies heavily on accurate RTK GPS system with integrated IMU attached to both the landing pad and the helicopter to provide accurate relative and absolute state estimation.

The heavy reliance on high-cost GPS for autonomous landing has led to a search for alternate approaches. Visual fiducial systems such as AprilTags [57] or ArUco [60], [79] have been employed, which allow a camera to extract pose information from a known target for relative positioning[47], [38], [41], [25]. Other work proposes the use of a laser array mounted on the bottom of a MAV [16] to determine the pose of a landing platform. A pair of IR cameras configured in a stereo configuration [39] have also been used, which allow for low light, night time landings. There has also been a more recent effort to use a SLAM based solution to allow a MAV to automatically select a safe landing zone [19]. While applicable to static landing methods outlined in [39], [19] and [16] have yet to be applied to landing on a moving platform, and in the case of [39], the instrumentation used to assist in the landing would need to move with the landing platform.

When landing on a moving platform, not only is the relative pose information required, but for accurate control and planning the landing target's inertial pose estimation is also required. Kalman filters have been used to fuse both GPS and visual measurements to obtain the landing target's inertial pose estimate, such as Kim et al (2014) [38] where an Unscented Kalman Filter was used, Square Root Unscented Kalman Filter(SRUKF) [79], or a standard Extended Kalman Filter (EKF) [25]. In some cases, measurements are used directly as inputs to a PID controller minimize the horizontal separation distance between the MAV and landing platform to zero before landing onto the platform [47, 41].

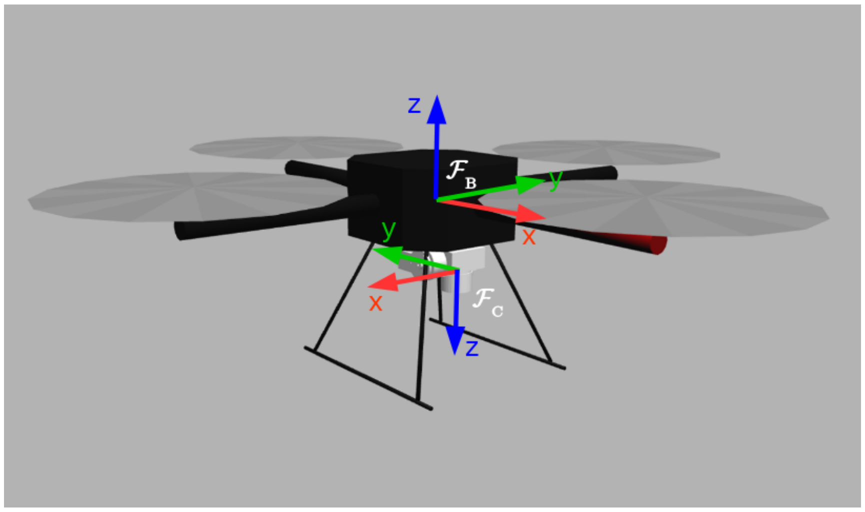
Further, if the landing target is moving at speeds beyond near-hover velocity (i.e. where tilt angle $> 10°$), or if the landing platform is inclined or rotated, a landing trajectory must be calculated to ensure an accurate and smooth final touchdown. One example of such an approach [16] presents a landing trajectory planner that performs a perching manoeuvre onto stationary platforms inclined at up to 30 deg.

In this chapter we describe a complete end-to-end Autonomous Tracking and Landing system which we call ATL. This system makes use of the AprilTag fiducial system [57] to estimate the relative pose of the landing target with respect to the pursuing MAV. We also extend previous results of this approach [29, 78, 41], by incorporating a gimbal camera, illumination invariant target detection and trajectory planner that aims to address the aforementioned issues. Stanley J. Brown and I contributed equally in this work, and we made contributions towards a faster and more robust AprilTag detection by using an illumination invariant technique [50]. The system is validated through a high fidelity simulation of a MAV and moving ground target using ROS and Gazebo.

3.1 System Overview

In this section we describe the elements of our problem: landing target, MAV, and ATL system. This work assumes the MAV is hovering at a high altitude on top of the landing target at the start of the autonomous landing procedure. The landing target which measures 1m on the width and length is represented by the AprilTag in Fig. 3.1, and is assumed to have dynamics of a two-wheel ground robot. The pursing MAV has an array of sensors in order to perform autonomous landing, namely a flight control unit (FCU) with GPU and IMU to estimate the MAV pose, and a 2-axis roll-pitch gimbal camera to detect and measure the relative pose of the landing target.

Four main coordinate frames are defined and illustrated in Fig. 3.1. The global frame, $\mathcal{F}_G$. The MAV body frame, $\mathcal{F}_B$, is at the center of the MAV body. The camera frame, $\mathcal{F}_C$, attached to the gimbal camera is downward facing. The coordinate frame for the gimbal mechanism is not depicted in Fig. 3.1, because it is assumed to be small or has near zero linkage length. This means that the camera center is at the pivot of the gimbal mechanism, and the orientation of the camera is equal to the gimbal configuration. Finally, the landing target frame, $\mathcal{F}_L$, is assumed to be always on the ground, with its x-axis pointing forwards.



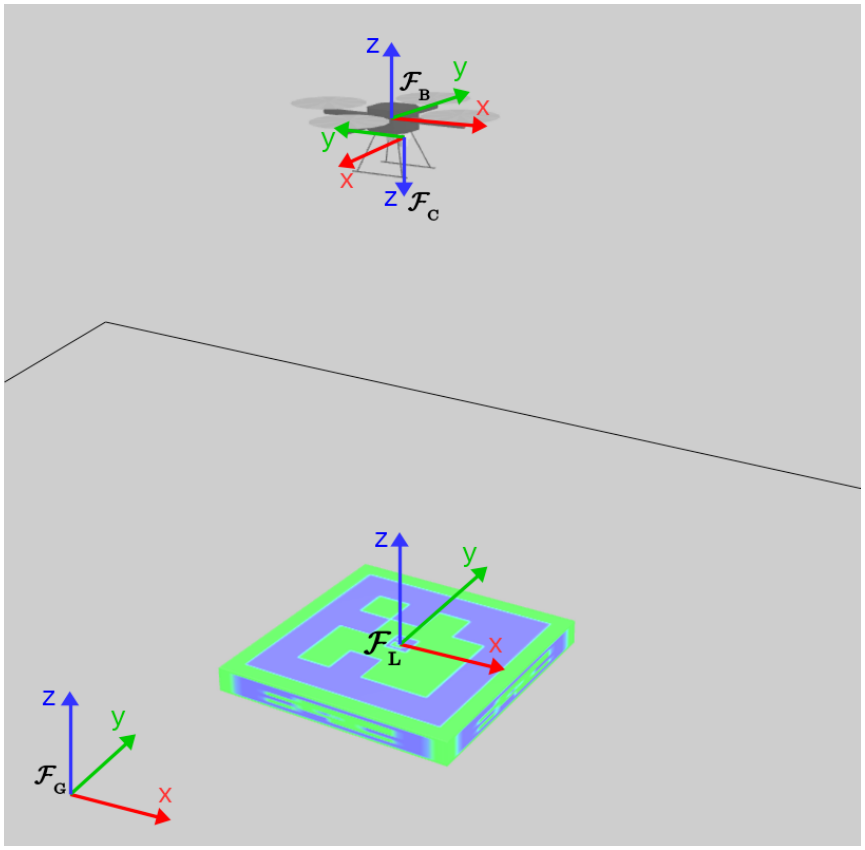


Figure 3.1: Coordinate frames used for autonomous MAV landing

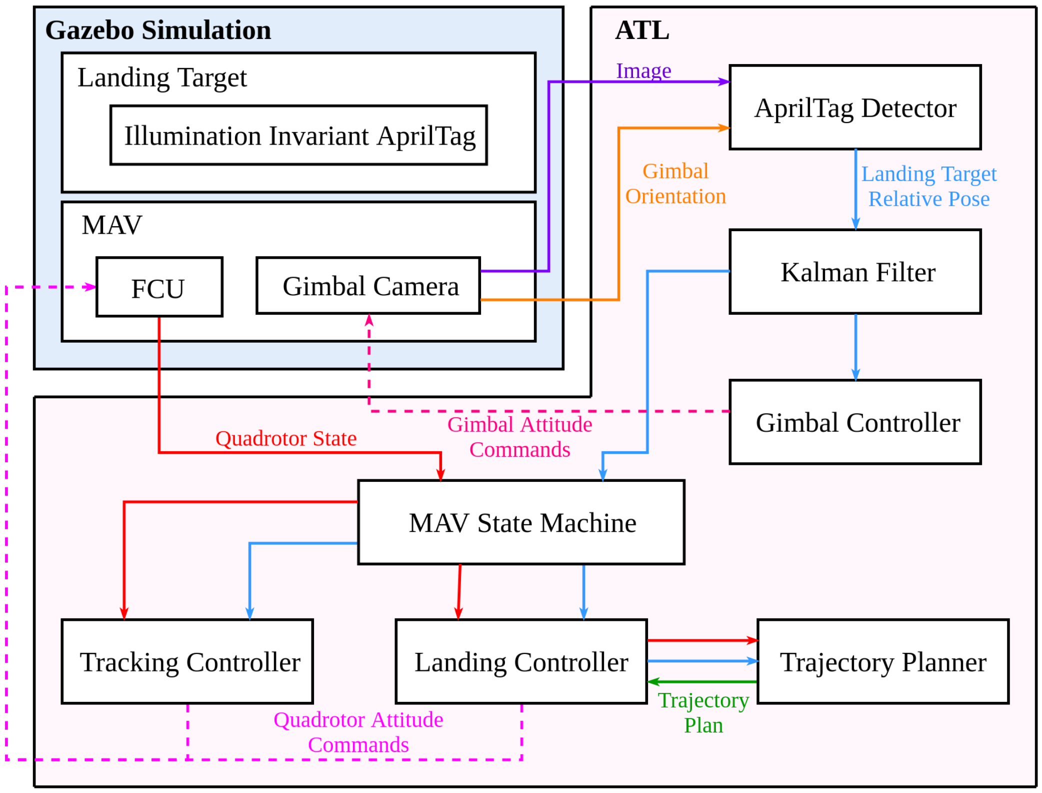


Figure 3.2: ATL Simulation and System Diagram

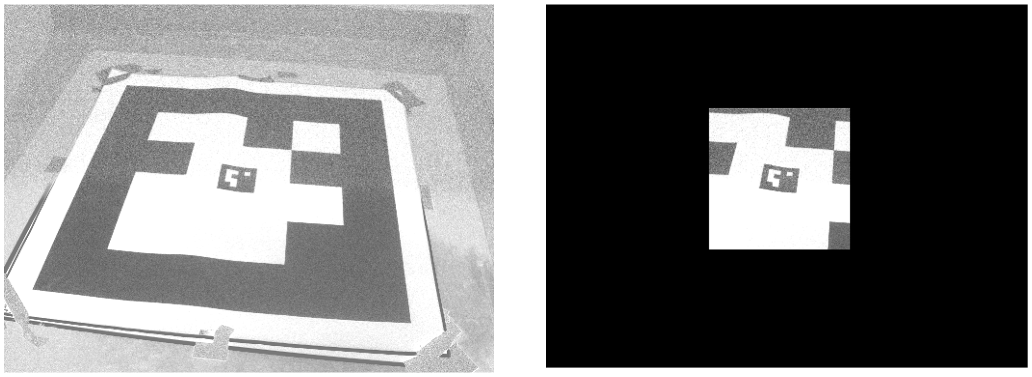
The system diagram in Fig. 3.2 shows our Gazebo simulation of a MAV consisting of an FCU and gimbal, as well as an AprilTag as the landing target modelled as a two wheel robot. The simulation provides inputs to the ATL system, such as the gimbal camera image, gimbal configuration and MAV state. The camera image and gimbal orientation are processed by the AprilTag detector to obtain the relative pose estimation of the landing target relative to the camera, the measurement is then transformed to the MAV body frame and filtered with a Kalman filter as inputs to the MAV state machine. At the same time the gimbal controller is used to keep the AprilTag in the camera's field of view constantly. The MAV state machine uses the tracking controller to track the landing target for a specific amount of time, once the tracking condition has been met the MAV state machine switches to landing mode by using the landing controller, and performs autonomous landing by following a pre-computed trajectory from the trajectory planner. Both the MAV tracking and landing controller outputs low-level attitude commands to the on-board flight controller unit (FCU).

In the following sections the perception system of ATL is first described, this includes the landing target detection, how the relative pose is filtered with a Kalman Filter and simultaneously tracked with the gimbal controller. Secondly, we describe the offline trajectory planner used to pre-compute MAV landing trajectories. Finally, the tracking and landing controllers used to control the MAV for autonomous landing are discussed.

3.2 Landing Target Detection

In order to land on a moving platform, the platform must be uniquely identified, isolated from the surrounding area, and its relative pose to the MAV must be estimated. To achieve this goal the AprilTag visual fiducial system [57] was used, which provides a pose measurement of the landing platforms relative to the camera. However, as noted by Ling (2014) [47], running the AprilTags library on an embedded computer system requires significant computational resources. For this reason several modifications to the AprilTag detection system was introduced:

1. **Adaptive Windowing**: expanding on the concept first introduced by Ling (2014) [47], the input image is was masked to areas where the AprilTag was last seen, we call this method adaptive windowing. This has the effect of reducing the AprilTag detector's search space.



(a) Original Image (b) Adaptive Windowing

Figure 3.3: Adaptive Windowing in Action

2. **Adaptive Image Re-sizing**: the image dimensions is varied when the tag is located various distances, for example if the tag distance is greater than 5 meters, an image

size of 640 by 480 pixels is used, however if target is below 4.5 meters a 320 by 240 image size is used. Adaptive image re-sizing has no impact on the relative pose extracted, but this approach sacrifices detection rate for the ability to detect small AprilTags in the image space thus a balance needs to be achieved. This method combined with active windowing allows the AprilTag detection process to achieve consistently higher detection rates throughout the detection range, a requirement for good target tracking and estimation.

3. **AprilTag Inception**: as the MAV approaches the landing target, the detection fails when portions of the AprilTag falls out of the camera's field of view. To resolve this issue a smaller AprilTag is nested inside of a larger AprilTag (see Fig. 3.4). Such that during MAV landing the detection would switch to the smaller one when detection of the larger AprilTag is lost. It is important to note that the corners of the smaller AprilTag does not cover or overlap the corners of the larger one. The large AprilTag pattern was specifically chosen to allow for a smaller AprilTag to be placed in the middle without disrupting the detection.

4. **Illumination Invariant**: our most impactful contribution towards robust fast AprilTag detection emerged from experience with the standard black and white AprilTag during outdoor experiments, where the detection becomes unreliable in certain lighting conditions. In particular, detection fails when strong shadows cover tag features fully or partially. The cause of failure is due to how the detection process relies on image gradients to detect the edges and lines of the tag in order to extract the relative tag pose. Depending on the time of day or weather conditions, this can have a significant impact on reliable AprilTag detection. This sensitivity to illumination was addressed by using the illumination invariant transform by Maddern et al. (2014) [50].

The illumination invariant transform takes three input channels from the image, and returns a single illumination adjusted channel, I, as follows,

$$I = \log(R_2) - \alpha \log(R_1) - (1 - \alpha) \log(R_3) \tag{3.1}$$

where R_1, R_2, R_3 are sensor responses (or image channels) corresponding to peak sensitivities at ordered wavelengths $\lambda_1 < \lambda_2 < \lambda_3$, and α is determined by Eq. (3.2).

$$\begin{aligned} \frac{1}{\lambda_2} &= \frac{\alpha}{\lambda_1} + \frac{(1 - \alpha)}{\lambda_3} \\ \alpha &= \frac{\lambda_1(\lambda_2 - \lambda_3)}{\lambda_2(\lambda_1 - \lambda_3)} \end{aligned} \tag{3.2}$$

This transform, however, has a non-intuitive effect on black and white targets, as the three channels tend to be equally over and under exposed in RGB images. As a result, the transform leads to similar values for white and black pixels, eliminating the ability for the AprilTag library to detect edges. To resolve this issue, we designed a new AprilTag so that the single channel image produced by using Eq. (3.1) produces a grey scale like image that is robust to shadows and changes in illumination. Examining Eq. (3.1), it can be observed the resulting pixel intensities are maximized when the camera observes green (R_2) and minimized when viewing a mixture of red and blue, (R_1 and R_3 respectively). The proposed illumination invariant AprilTag shown in Fig. 3.4 is created by replacing the white and black portions of a typical AprilTag with green and magenta. This modification was tested under various lighting conditions. Fig. 3.5 shows the tag's appearance after performing the illumination invariant transform, creating a single channel image that replaces the typical single channel, grey scale image that is typically used by the AprilTag library. The images shown in Fig. 3.5 are taken using a PointGrey Chameleon3 (CM3-U3-28S4C-CS) with a Sony ICX818 image sensor. The corresponding values of $\lambda_1, \lambda_2, \lambda_3$ and α are 480 nm, 510 nm, 640 nm and 0.56 respectively as noted in the sensor data sheets.

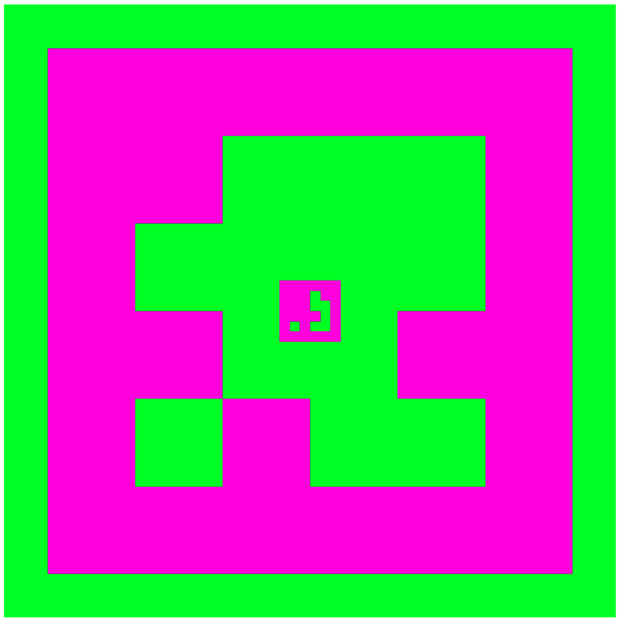
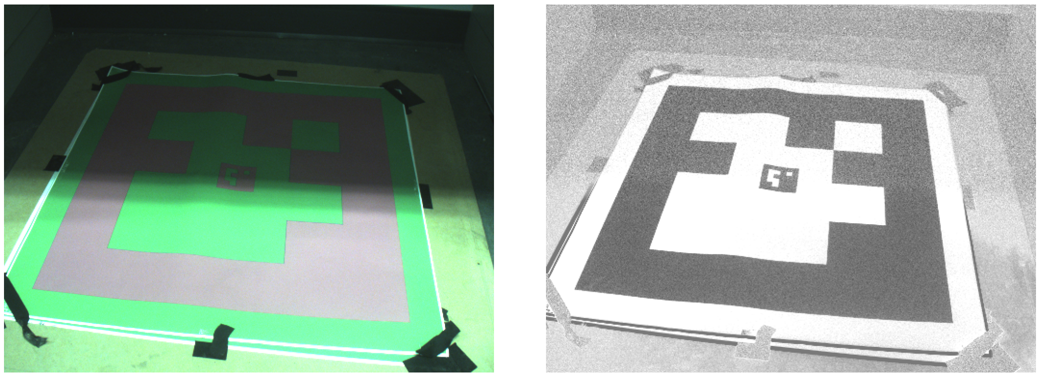


Figure 3.4: Proposed Illumination Invariant AprilTag with tag id 0 from the 16h5 family embedded into the center of tag id 5 from the 16h5 family. The green color appear white after applying Eq. (3.1) while the magenta color appears black.



(a) Original Image (b) Illumination Invariant Transform

Figure 3.5: Illumination Invariant AprilTag in Action

3.3 Landing Target Estimation

The relative target position, linear velocity, angular velocity and heading are estimated with an Extended Kalman Filter which is then used by the gimbal controller for continuous landing target tracking, and the tracking and landing controllers for tracking and landing onto the moving ground target (see Fig. 3.2).

Process Model: A two-wheel robot motion model is used to approximate the forward kinematics of the ground target. Let the relative target state in the *body frame*, $\mathbf{x}$, be defined as

$$\mathbf{x} = \begin{bmatrix} x & y & z & \theta & v & v_z & \omega & a & a_z \end{bmatrix}^T \tag{3.3}$$

where x, y and z is the landing target position relative to the pursuing MAV, heading θ, wheel velocity v, steering angular velocity ω and linear velocity v_z in the z direction. And let the target inputs be

$$\mathbf{u}(t) = [\mathbf{n}_\omega \quad \mathbf{n}_a \quad \mathbf{n}_{a_z}]^T \tag{3.4}$$

here, $\mathbf{n}_\omega$, $\mathbf{n}_a$ and $\mathbf{n}_{a_z}$ are all driven by Gaussian process noise of known mean and covariance.

The target model dynamics are then given by,

$$\begin{aligned}
\dot{x} &= v\cos(\theta) \\
\dot{y} &= v\sin(\theta) \\
\dot{z} &= v_z \\
\dot{\theta} &= \omega \\
\dot{v} &= a \\
\dot{v}_z &= a_z \\
\dot{\omega} &= \mathbf{n}_\omega \\
\dot{a} &= \mathbf{n}_a \\
\dot{a}_z &= \mathbf{n}_{a_z}
\end{aligned} \tag{3.5}$$

Measurement Model: The measurement model is the landing target position, (x, y, z), and heading, θ, in the body frame, $\mathcal{F}_B$. But because the measurement of the AprilTag, $\mathbf{z}_C$, is in the camera frame, $\mathcal{F}_C$, a transformation is required as follows,

$$\mathbf{x}_B = \mathbf{T}_{B:C}\mathbf{z}_C. \tag{3.6}$$

An EKF is employed to propagate forward the dynamic model of the target between measurements, while the measurement update is performed upon when the AprilTag is detected and transformed.

3.4 Gimbal Controller

To continuously track the ground target, a 2-axis gimbal is used. The gimbal can be actuated in roll and pitch to change the camera view. The following assumptions are made. First, the gimbal joint angles are independent. Secondly, the gimbal is small or has near zero linkage length, and that the camera center is at the pivot point of the gimbal mechanism. Third, the camera orientation is equal to the gimbal configuration. Lastly, the landing target position, $\mathbf{x}_B$, relative to the body frame is known. With the assumptions, the gimbal joint angle in roll, ϕ_d, and pitch, θ_d, can be calculated via:

$$d = ||\mathbf{x}_B|| \tag{3.7}$$

$$\phi_d = \sin^{-1}(y/d) \tag{3.8}$$

$$\theta_d = \sin^{-1}(x/d). \tag{3.9}$$

Using the calculated desired roll, ϕ_d, and pitch, θ_d, the gimbal can be controlled with two independent PID controllers as follows,

$$e_\phi = \phi_d - \phi \tag{3.10}$$

$$e_\theta = \theta_d - \theta \tag{3.11}$$

$$u_\phi = k_p^\phi e_\phi + k_i^\phi \int_0^t e_\phi dt + k_d^\phi \dot{e}_\phi \tag{3.12}$$

$$u_\theta = k_p^\theta e_\theta + k_i^\theta \int_0^t e_\theta dt + k_d^\theta \dot{e}_\theta \tag{3.13}$$

where, e_ϕ and e_θ are the errors of roll and pitch calculated from the difference between desired and actual roll ϕ and pitch θ, and k_p, k_i, k_d are the gains of the PID controller. The control inputs u_ϕ and u_θ for roll and pitch respectively can be used directly for controlling the gimbal.

3.5 Trajectory Planning

In addition to the relative state estimation problem, when attempting to land on a target moving at high speeds there is a possibility that the MAV's blades may impact the landing pad's surface if large pitch or roll motions are produced by the landing controller. At speed, MAV aerodynamics dictate that significant pitch angles be maintained to match target speed, and yet landing is most reliably achieved by matching the landing deck's attitude. As such, there exists a clear need for motion planning in order to provide a smooth, safe and repeatable landing that balances end state constraints with tracking performance.

In the following section we develop a versatile, light weight trajectory planner that defines an optimal trajectory to execute a safe landing on the target. This is achieved by incorporating important dynamic model components such as ground effect and non-linear drag force models. The trajectory is defined with 2 degrees of freedom, while relative heading tracking is used to mimic target trajectories that vary the direction of forward velocity. The plan is then executed using a combined feed forward and feedback control.

3.5.1 2D Quadrotor Model

To simplify the trajectory planning, a three degree of freedom (DOF) MAV model in the vertical x-z plane is used. As aerodynamic drag forces occur mainly in opposition to the direction of travel, it is possible to neglect the lateral tracking of the target in the planning phase of the problem. Instead, we align the vehicle motion with the target motion and heading through lateral roll and yaw control, and compute an independent descent and landing plan for longitudinal control, as defined in Section 3.5.2. This assumption allows the planning process to avoid full 6 DOF rigid body dynamics, and is notably less computationally expensive as a result.

The 2D MAV state $\mathbf{x}$ includes both position and velocity in the x and z direction.

$$\mathbf{x} = \begin{bmatrix} x & \dot{x} & z & \dot{z} \end{bmatrix}^T \in \mathbb{R}^4 \tag{3.14}$$

The inputs to the MAV model are the acceleration, a_z, in the z direction in the body frame, which can be directly related to total thrust, and the pitch angle, θ, in the inertial frame.

$$\mathbf{u}(t) = \begin{bmatrix} a_z \\ \theta \end{bmatrix} \in \mathbb{R}^2 \tag{3.15}$$

The dynamics in this reduced coordinate system include both ground effect and non-linear drag terms. The ground effect, S_{ge}, scales the requested vertical acceleration, a_z, when operating within a height, h_{ge}, of the ground and target, as follows.

$$S_{ge} = \left(k_{ge} \frac{h_{ge} - \min(h_{ge}, z)}{h_g e} \right)^2 \tag{3.16}$$

The coefficient, k_{ge}, is MAV dependent, and can be identified in hover flight near the ground. A linear drag model is employed, as it has been demonstrated to be sufficiently accurate over a moderate forward speed range [42], but could easily be extended to a quadratic model. The drag is defined as resisting the velocity of the MAV through the free stream in the x-z MAV plane. The complete 2D MAV model is then,

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}) = \begin{bmatrix} x_2 \\ a_z \sin(\theta) - k_{dx}\dot{x} \\ x_4 \\ a_z \cos(\theta)(1 + S_{ge}) - k_{dz}\dot{z} - g \end{bmatrix} \tag{3.17}$$

where g is the gravitational constant, k_{dx} and k_{dz} are the linear drag coefficients. The trajectory optimization problem can proceed in both continuous or discrete time, and we restrict our focus to discrete time. In this work, the dynamics defined in Eq. (3.17) are converted to discrete time through zero-order hold.

$$\mathbf{x}(t+1) = \mathbf{x}(t) + f(\mathbf{x}(t), \mathbf{u}(t))dt \tag{3.18}$$

3.5.2 Trajectory Problem Formulation

The objective of the trajectory is to achieve landing of the MAV onto a non-stationary target by following a pre-defined desired descent trajectory. Further, the optimal trajectory aims to balance a number of conflicting costs, which include minimal deviation from the desired trajectory, minimal total input and minimal input control difference, which penalizes rapid changes in the control input.

The desired trajectory, $x_{des} \in \mathbb{R}^{4 \times t_f + 1}$, is a nominally defined as a constant velocity descent trajectory from initial position and velocity, $\mathbf{x}_0$, to final position and velocity, $\mathbf{x}_f$. Both boundaries are fixed for the optimization, in that the MAV initial condition is known, and the final condition must match the position and velocity of the target at the final time t_f.

$$\begin{aligned} \mathbf{x}_0 &= \begin{bmatrix} x_0 & \dot{x_0} & z_0 & \dot{z_0} \end{bmatrix}^T \\ \mathbf{x}_f &= \begin{bmatrix} x_f & \dot{x_f} & z_f & \dot{z_f} \end{bmatrix}^T \end{aligned} \tag{3.19}$$

In addition, the final inputs for thrust and pitch are also fixed, in that the pitch must match the landing platform orientation, and the thrust must be significantly below hover thrust.

The deviation from the desired path, p_{diff}, is defined as, $p_{\text{diff}} = ||\mathbf{x} - \mathbf{x}_{\text{des}}||^2$ the total input cost, u_{tot}, is, $u_{\text{tot}} = ||\mathbf{u}||^2$ and the input difference cost, u_{diff}, is defined as,

$$u_{\text{diff}} = \sum_{i=0}^{t_f - 1} \left\| \mathbf{u}(i+1) - \mathbf{u}(i) \right\|^2 . \tag{3.20}$$

The complete cost function $J : \mathbb{R}^{6 \times t_f + 1} \to \mathbb{R}$ is then a weighted summation of the deviation and input costs,

$$J(\mathbf{x}, \mathbf{u}) = w_1 \cdot p_{\text{diff}} + w_2 \cdot u_{\text{total}} + w_3 \cdot u_{\text{diff}} \tag{3.21}$$

the optimal control problem can now be written as,

$$\begin{aligned} \min_{\mathbf{x(t)} \in \mathbf{X},\ \mathbf{u(t)} \in \mathbf{U}} \quad & J(\mathbf{x}, \mathbf{u}) \\ \text{s.t.} \quad & \mathbf{x}(t+1) = f(\mathbf{x}(t), \mathbf{u}(t))dt \quad \forall t \in [0, t_f] \\ & \mathbf{x}(t = 0) = \mathbf{x}_0 \\ & \mathbf{x}(t = t_f) = \mathbf{x}_d \end{aligned}$$

The result of the optimization is then used by the landing controller described in Section. 3.7 to execute the landing trajectory.

3.6 Tracking Controller

Using the landing target position $\mathbf{x}_B$ in body frame estimated from the Kalman filter described in Section. 3.3, a tracking controller can be developed to minimize the target position error in the $x - y$ (horizontal) plane, this was achieved by implementing a PD controller

$$\mathbf{e} = \mathbf{x}_B \tag{3.22}$$

$$u_\phi = -1\left(k_p^\phi e_y + k_d^\phi \dot{e}_y\right) \tag{3.23}$$

$$u_\theta = k_p^\theta e_x + k_d^\theta \dot{e}_x \tag{3.24}$$

where $\mathbf{e}$ is the landing target position error in body frame, k_p, k_d are proportional and derivative gains of the PD controller. The attitude commands u_ϕ and u_θ for roll and pitch respectively is used directly by the MAV's FCU for control.

3.7 Landing Controller

For the final landing phase a planned trajectory path from Section. 3.5 is executed, to track the planned trajectory path a PID feedback-feed-forward control was developed where the feed-forward reference signals are the planned pitch u_{ff}^θ and thrust u_{ff}^T, and the feedback reference signal is the planned velocity $\mathbf{v}_p$,

$$\mathbf{e}_v = \mathbf{v}_p - \mathbf{u}_{\text{fb}} \tag{3.25}$$

$$u_\phi = -1\left(k_p^\phi e_y + k_i^\phi \int_0^t e_y dt + k_d^\phi \dot{e}_y\right) \tag{3.26}$$

$$u_\theta = k_p^\theta e_x + k_i^\theta \int_0^t e_x dt + k_d^\theta \dot{e}_x + u_{\text{ff}}^\theta \tag{3.27}$$

$$u_T = k_p^T e_z + k_i^T \int_0^t e_z dt + k_d^T \dot{e}_z + u_{\text{ff}}^T \tag{3.28}$$

where $\mathbf{e}_v$ is the velocity error between planned velocity $\mathbf{v}_p$ and feedback velocity $\mathbf{u}_{\text{fb}}$, and k_p, k_i, k_d are the gains of the PID controller. Similar to the tracking controller, the attitude commands u_ϕ, u_θ, and u_T for roll, pitch and thrust respectively are used directly by the MAV's FCU for control.

3.8 Experiments and Results

We demonstrate the performance of our autonomous MAV landing system, ATL, in simulation using Gazebo, a high fidelity simulation environment. The simulation includes complete models of the landing target and a MAV containing a gimbal camera. The ATL system is demonstrated on both constant and variable target heading trajectories, and relative pose estimation accuracy is demonstrated throughout the landing mission, confirming the feasibility of employing our method for fully automated landings in a range of conditions.

The proposed gimbal tracking algorithm, along with the target estimation and landing trajectory planner are all validated in the Gazebo simulation environment using an expanded 3D version of the MAV dynamics outlined in Section. 3.5, and following similar models defined in the literature [35]. The camera is mounted at the end of a two axis gimbal camera connected to the bottom of the MAV body, and images are acquired at 60Hz. The estimation, planning and control systems are all implemented in ROS, and hence realistic processing and communication delays affect the results of the simulation. The final landing position is offset by 0.2 meters in the z direction to represent the landing gear of the simulated MAV.

Fig. 3.6 shows the landing target travelling in a straight line from $(0, 0, 0)$ towards $(200, 0, 0)$ at approximately 10 m/s. The pursuing MAV is positioned at $(0, 0, 5)$ at the start of the simulation, and is assumed to be detecting and hovering above the landing target at the start of the simulation. As the landing target moves, the MAV estimates the relative position of the landing target, and uses the gimbal controller to track the landing target in order to keep it within the camera's field of view. The tracking controller using the relative position estimates and attempts to minimize the horizontal distance between the MAV and landing target. Once the tracking controller has tracked the landing target for a preset number of seconds, the MAV executes a pre-computed landing trajectory on to the landing target.

The estimated relative position of the landing target during simulation is shown in Fig. 3.7. The RMSE between the estimation and ground truth in the body x, y and z are 0.0131m, 0.0792m and 0.1085m respectively. The combination of under estimating the relative position of the landing target, and under-reactive controls due sub-optimal controller gains led to oscillatory controls, which in turn led to an oscillatory relative estimation. The noise in ground truth is due to transformation errors from MAV and landing target position. The direct measurements contain noise generated by pixel discretization and detection misalignment, whose effect on control performance is mitigated by the EKF.

Once the landing target has been tracked for a preset number of seconds, the MAV switches to landing mode. In landing mode the MAV executes a pre-computed trajectory. The actual and planned MAV position, velocity, attitude and thrust are compared. Fig. 3.8 shows the MAV's ability in executing the planned optimal descent trajectory for landing. The trajectory planned includes a rapid initial pitch and decreased thrust excursion to match the target velocity and initiate a descent, and a final feathering manoeuvre to match the platform orientation and limit landing velocity. The landing controller performs the manoeuvre with limited error, and is sufficiently accurate to land on the platform reliably. The whole simulation is presented in Fig. 3.6 for a straight line motion of the ground target.

The same system was tested against a curved landing target motion travelling at 1 m/s, and Fig. 3.9 shows that the MAV was still able to track and land on a moving target with some significant lateral motion. Both simulation trials demonstrate the validity of the decoupling the assumption for tracking longitudinal and lateral errors independently.

3.9 Conclusions

In this chapter we have presented a set of novel methods to make the detection of AprilTags more robust on strong shadows and in various lighting conditions. Further, we lowered the computational cost of extracting the AprilTag from an image. We also present a complete end-to-end solution for autonomous landing of a MAV onto a moving platform, complete with a tracking controller using a gimballed camera and a trajectory planner that takes into account the attitude constraints of the MAV. The landing planner is validated in a Gazebo simulation which demonstrates the proposed planner landing at speeds of 10m/s.

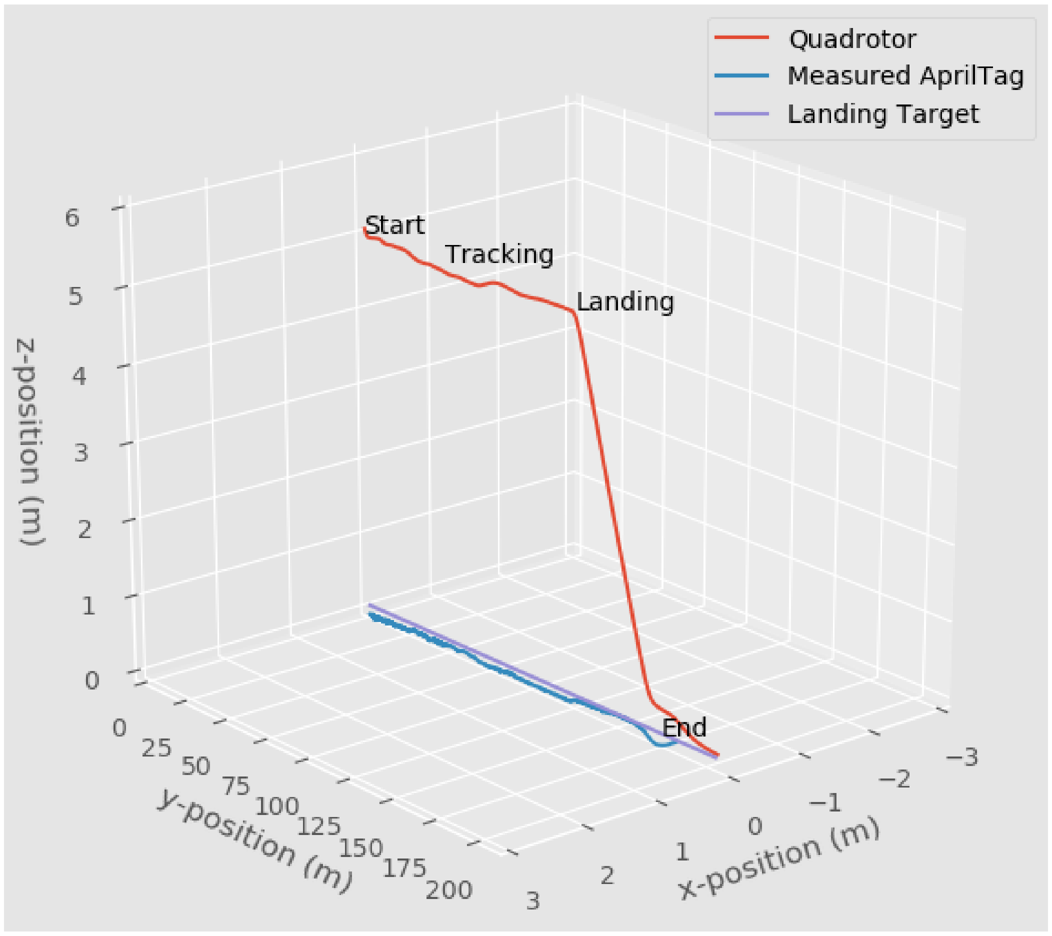


Figure 3.6: 3D plot of a straight line tracking and landing at 10m/s

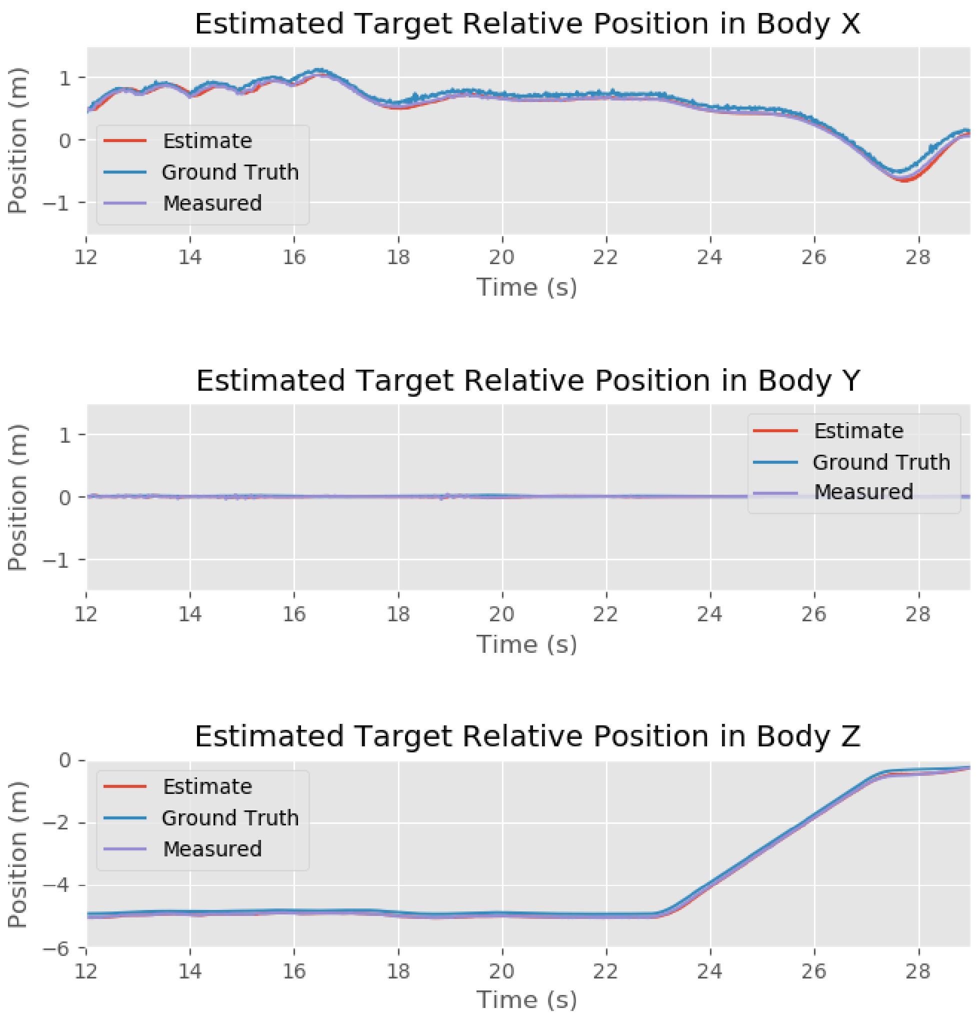


Figure 3.7: EKF estimation while tracking and landing on the target travelling in a straight line at 10m/s

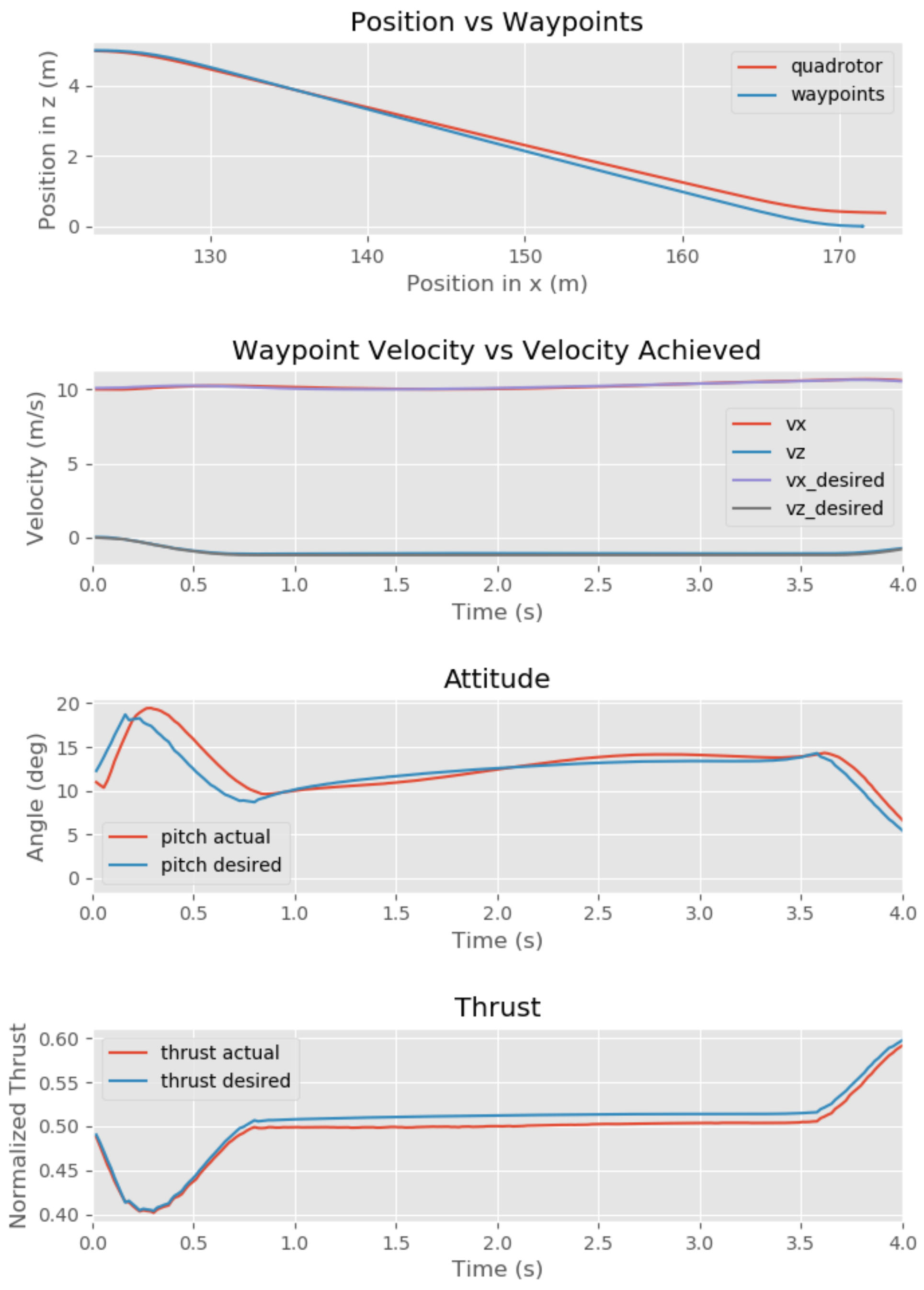


Figure 3.8: Landing Trajectory

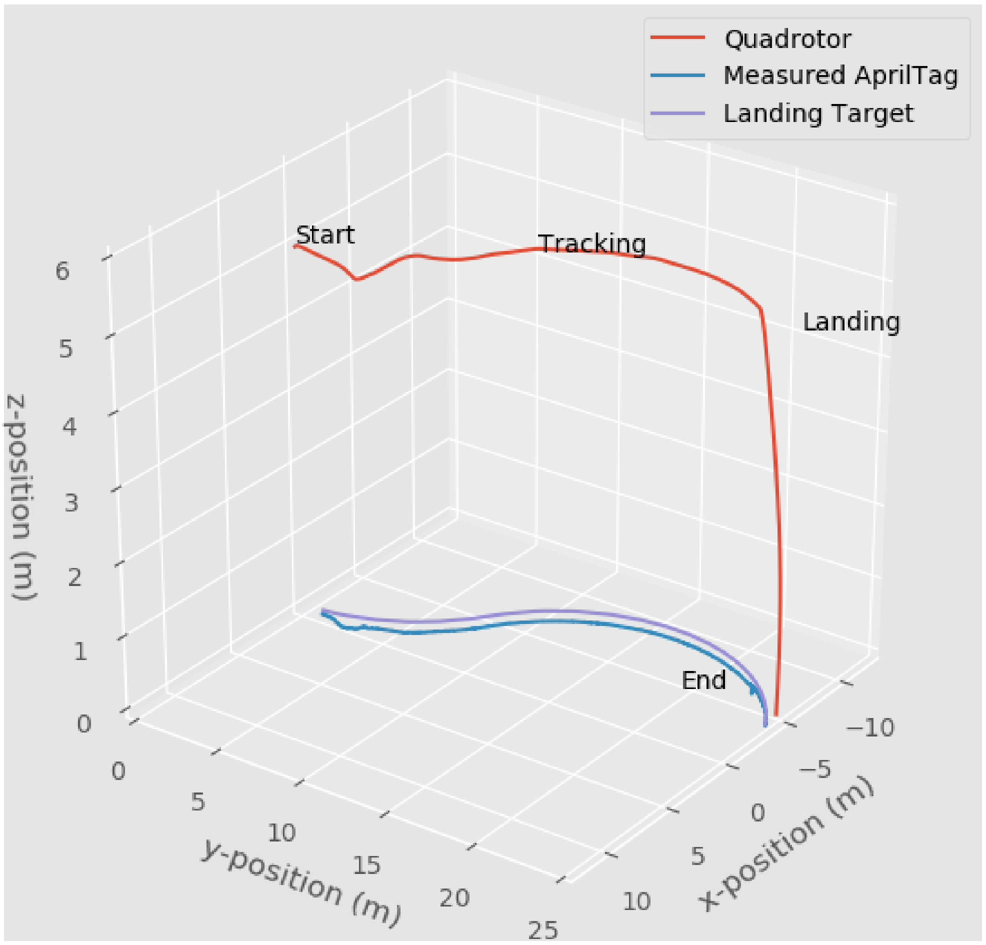


Figure 3.9: 3D plot of a curved line tracking and landing at 1m/s

Chapter 4

Encoderless Gimbal Calibration of a Dynamic Camera Cluster

In GPS-denied environments, the state information of the MAV will need to be acquired from alternative sensors. Lidar is bulky, heavy, and expensive therefore it is not well suited for light weight MAVs. Cameras on the other hand are cheap, small and light weight, therefore they are an attractive sensor for state estimation for a MAV.

Vision based localization and mapping algorithms have mostly considered either a monocular or a stereo camera configuration [21, 17, 76]. Multi-camera clusters (MCCs) on the other hand are becoming an effective camera configuration for robotic platforms, such as self-driving cars and drones [33, 71]. The advantage of MCCs comes in the form of increased field of view compared to a monocular or stereo camera configuration, thereby enabling features to be tracked over a longer period of time. MCCs have two possible configurations: static camera clusters (SCCs) or dynamic camera clusters (DCCs). SCCs have rigidly mounted camera configurations with static extrinsic parameters, while DCCs incorporate one or more cameras that are mounted on an actuated mechanism, such as a gimbal, allowing the camera to change viewpoint independent of robot motion. The ability to dynamically change viewpoint enhances the tracking ability of interesting and informative features, which could in turn be used to improve localization estimates. The drawback of active DCC viewpoint selection, however, is that the time-varying camera extrinsics must be resolved with sufficient accuracy to perform visual odometry or SLAM without negatively impacting the estimation performance.

There is extensive literature on camera-to-camera calibration approaches that use fiducial markers to generate overlapping observations between cameras [71, 40, 45]. But the work of [13] was the first to introduce the calibration of time-varying extrinsic parameters for various DCC configurations, such as one static and one actuated camera as well as two actuated cameras. This work was extended to determine optimal gimbal configurations which locally minimize parameter uncertainty using next-best-view [59]. Although the proposed approaches are effective in performing DCC calibration, both methods rely on having encoder feedback to extract the position of the gimbal motor, and also require knowledge of the joint angles to determine the kinematic chain between the static and actuated cameras.

In practice, commercially available gimbals do not always have the joint angle information from encoder motors, and in some cases, the integration of an encoder may not be desirable due to weight or cost limitations. Therefore the lack of joint angle feedback also introduces difficulties while performing VIO or SLAM algorithms, as accurate joint angle information is beneficial to resolving the time-varying extrinsics during robot motion.

In this work, we present two main contributions. First, we develop an encoderless DCC calibration approach which eliminates the requirement of joint angle measurements from the actuated mechanism. Our approach estimates the DCC calibration parameters in conjunction with the joint angles of the actuated mechanism for each configuration of the collected measurement set. By eliminating the need for joint angle information, this calibration can be performed on any actuated mechanism, allowing for visual SLAM algorithms to be run on any off-the shelf DCC. Second, we extend the Visual Inertial Odometry (VIO) algorithm developed in [44] and incorporate the calibrated DCC in order to simultaneously estimate the gimbal joint angles and the VIO localization state. We obtain experimental results using a gimbal based DCC mounted on a custom MAV platform, and evaluate the relative performance of SCCs and DCCs in a large outdoor environment with a variety of static and gimballed camera configurations. We show that our overall pose estimate using a DCC is comparable to those obtained using an SCC configuration without significant degradation in performance. This was a collaborative effort with Jason Rebello, Leonid Koppel, Pranav Ganti and Arun Das. The encoderless dynamic camera cluster calibration method was formulated by Arun Das. Simulation and results verification was Jason Rebello and Pranav Ganti's contribution. Leonid Koppel with the help of Arun Das was responsible for modifying OKVIS to incorporate a dynamic camera cluster for online camera extrinsics estimation. My contribution includes generating all experimental results, and construction of a custom MAV research platform to achieve aerial demonstrations of encoderless gimbal SLAM. This chapter presents the complete work of the team so as to enable proper understanding of the results generated through my efforts.

4.1 Notation

General Rigid Body Transformation: Let a 3D point in coordinate frame $\mathcal{F}_x$ be denoted as $\mathbf{p}^x \in \mathbb{R}^3$. The rigid body transformation from frame $\mathcal{F}_a$ to $\mathcal{F}_b$ is expressed as $\mathbf{T}^{\tau}_{b:a} \in \mathbb{SE}(3)$, where $\mathbf{T}^{\tau}_{b:a} : \mathbb{R}^3 \mapsto \mathbb{R}^3$, and $\tau = [r_x, r_y, r_z, t_x, t_y, t_z]$ is the parameter vector used to construct the transformation. The rotation is represented using 3-2-1 Euler angles $r_x, r_y, r_z \in [0, 2\pi)$, while the translation values along the respective axes are $t_x, t_y, t_z \in \mathbb{R}^3$. Although this work uses Euler angles, the rotations can be represented using other conventions, such as quaternions or $\mathbb{SO}(3)$ rotation matrices.

Image Projections of 3D Points: A point $\mathbf{p}^c_i$ in camera frame $\mathcal{F}_c$ can be projected into a pixel location on the 2D image plane with projection function $\psi(\mathbf{p}^c_i) : \mathbb{R}^3 \mapsto \mathbb{P}^2$, defined as $\psi(\mathbf{p}^c_i) = [u^c_i \quad v^c_i]^T$. Here u_i and v_i are the pixel coordinates of the projected point along the u and v dimensions respectively.

Denavit-Hartenberg Parameterization: The actuated mechanism of the gimbal is modeled as a serial manipulator with rotational joints, represented by the Denavit-Hartenberg (DH) convention. The DH parameters $[\theta_l, d_l, a_l, \alpha_l]^T$, where $\theta_l, \alpha_l \in [0, 2\pi)$ and $d_l, a_l \in \mathbb{R}$ is used to describe the homogeneous rigid body transformation from one frame to another, $\mathcal{F}_{l-1}$ to $\mathcal{F}_l$. We distinguish between θ_l and the rest of the DH parameters by defining $\omega_l = [\, d_l, a_l, \alpha_l]^T$. For a detailed summary of the DH convention, the reader is referred to [30].

4.2 Problem Formulation

In the following section we will summarize a novel DCC calibration approach, which performs the calibration of the DCC without the use of encoder feedback.

The camera extrinsics between the static and dynamic camera in a DCC has the form

$$\mathbf{T}^{\Theta,\beta}_{d:s} = \mathbf{T}^{\tau_d}_{d:e}\mathbf{T}^{\omega,\beta}_{e:b}\mathbf{T}^{\tau_s}_{b:s} \tag{4.1}$$

where $\mathbf{T}^{\tau_s}_{b:s}$ defines the transformation from the static camera to the mechanism base frame, $\mathbf{T}^{\omega,\lambda}_{e:b}$ defines the transformation from the base frame of the mechanism to the end effector frame, and $\mathbf{T}^{\tau_d}_{d:e}$ defines the transformation from the end-effector frame to the dynamic camera frame. Note that $\mathbf{T}^{\omega,\beta}_{e:b}$ is a chain of transforms through the mechanism's links computed using its forward kinematics, and it is a function of its DH parameters and control inputs.

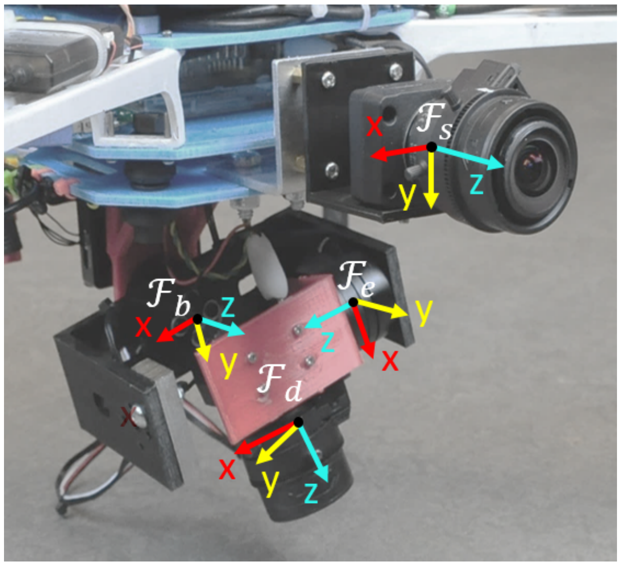


Figure 4.1: Frame diagram of a 2-DOF gimbal based DCC. Frames $\mathcal{F}_s$, $\mathcal{F}_d$, $\mathcal{F}_b$, $\mathcal{F}_e$ represent the static camera, dynamic camera, mechanism base, and mechanism end-effector frames respectively. An IMU is mounted to the back of the dynamic camera which is used for image stabilization of the gimbal and for estimating approximate joint angles to initialize the encoder-less calibration.

The goal of the calibration process is to obtain the rigid body transformation $\mathbf{T}_{d:s}^{\Theta,\beta}$ from the static camera frame $\mathcal{F}_s$, to the dynamic camera frame $\mathcal{F}_d$, for an L joint mechanism, where Θ is the set of estimated kinematic parameters used to build the rigid body transform and $\beta \in \mathbb{R}^L$, defined as $\beta_i = [\theta_0^i \cdots \theta_L^i]$, is the estimated joint angles for the i^{th} measurement set. In order to calibrate the DCC without the use of encoder feedback, both the kinematic parameters, Θ, the joint angles β for each measurement set are simultaneously estimated.

To calibrate the DCC, we first obtain a set of measurements from both the static and dynamic camera observing the same known fiducial target. Each i-th *measurement set* is defined as $Z_i = \{P_i^s, P_i^d, Q_i^s, Q_i^d, \beta_i\}$, where P_i^s, $P_i^d \in \mathbb{R}^3$ are the set of corresponding marker point positions defined in the static and dynamic camera frames respectively, which are easily computed by passing the known target points through the target to camera transformation $\mathbf{T}_{c:t}$. The pose of each camera with respect to the marker frame, $\mathbf{T}_{c:t}$, $c \in \{s, d\}$ is obtained by solving the Perspective-n-Point (PnP) problem [20]. Q_i^s and Q_i^d $\in \mathbb{R}^2$ represent the set of pixel measurements for the target points, as observed by the static and dynamic cameras, respectively, and β_i is the set of joint angles for the mechanism at the i-th measurement set.

The re-projection error between the measured marker point j in the static camera frame and the corresponding measured point in the dynamic camera frame, using the i-th measurement set and the transformation between camera frames, is defined as:

$$e_j^d(\Theta, \beta_i) = z_j^d - \Psi^d(\mathbf{T}_{d:s}^{\Theta,\beta}\mathbf{p}_j^s) \tag{4.2}$$

where $z_j^d \in Q_i^d$ is the measurement of point j, observed in the dynamic camera, and $\mathbf{p}_j^s \in P_i^s$ is the 3D position of point j as observed from the static camera. Since both the actuated and static camera observe the same marker at each measurement set, we can similarly compute the error for points observed in the actuated frame and projected into the static frame as

$$e_j^s(\Theta, \beta_i) = z_j^s - \Psi^s((\mathbf{T}_{d:s}^{\Theta,\beta})^{-1}\mathbf{p}_j^d) \tag{4.3}$$

where $z_j^s \in Q_i^s$ is the measurement of point j observed in the static camera, and $\mathbf{p}_j^d \in P_i^d$ is the 3D position of point j as observed from the dynamic camera.

Given a total of K measurement sets, let us define a the complete set of estimated joint angles as $\zeta = [\beta_1 \ \cdots \ \beta_K]$. The total squared re-projection error as a function of the estimation parameters, $\Lambda(\Theta, \zeta) : \mathbb{R}^n \mapsto \mathbb{R}$ over all of the collected measurement sets, $\Gamma = \{Z_1, Z_2, \ldots, Z_k\}$, is defined as

$$\begin{aligned}\Lambda(\Theta, \zeta) = \sum_{Z_i \in \Gamma} \sum_{j=1}^{|P_i^s|} & e_j^d(\Theta, \beta_i)^T e_j^d(\Theta, \beta_i) \\ & + e_j^s(\Theta, \beta_i)^T e_j^s(\Theta, \beta_i).\end{aligned} \tag{4.4}$$

Finally, an unconstrained non-linear optimization of Eq. (4.4) is performed in order to find the optimal calibration parameters, Θ^*, and optimal joint angle values, ζ^*, which minimizes the total re-projection error over the set of collected measurements,

$$\Theta^*, \zeta^* = \underset{\Theta,\zeta}{\operatorname{argmin}} \, \Lambda(\Theta, \zeta) \tag{4.5}$$

In practice, we have found that the kinematic parameters Θ can be sufficiently initialized using approximate, hand-measured values, and the joint angles ζ can be sufficiently initialized using approximate angles obtained from an IMU.

4.3 Visual Inertial Odometry using a DCC

To validate our claim that a DCC can be used for localization and mapping algorithms, we extended OKVIS [44], a VIO algorithm, to accommodate a DCC. OKVIS is an indirect, non-linear optimization-based algorithm which is capable of online estimation of time-varying camera extrinsics by minimizing re-projection error. The original implementation, however, models the camera extrinsics as a general transformation, and considers only small changes modeled by a Gaussian process, such as thermal expansion. We extended OKVIS by incorporating the gimballed extrinsics model of Eq. (4.1), and is capable of estimating joint angles with quick and un-modeled mechanism motion.

In the following, we briefly explain the approach presented in [44]. Let us define the IMU and global frames as $\mathcal{F}_I$ and $\mathcal{F}_G$, respectively. The VIO state vector for the robot, $\mathbf{x}$, is estimated at every time-step k of the algorithm, and is given by

$$\mathbf{x}_k = [\mathbf{p}_I^{IG} \; \mathbf{q}_{IG} \; \mathbf{v}_I^{IG} \; \mathbf{b}_g \; \mathbf{b}_a]^T \tag{4.6}$$

where $\mathbf{p}_I^{IG}$ denotes the position vector of the IMU frame of the cluster with respect to the global frame, $\mathbf{q}_{IG}$ is the quaternion which defines the rotation between the IMU and global frame, $\mathbf{v}_I^{IG}$ is the velocity of the IMU expressed in the IMU frame, and $\mathbf{b}_g$ and $\mathbf{b}_a$ are the bias states of the IMU's gyroscope and accelerometer, respectively. OKVIS formulates the visual-inertial localization and mapping problem as a joint optimization of a cost function containing both the weighed re-projection error and temporal error term from the IMU. Here, for brevity we will only summarize the re-projection error, since it will be modified to include our DCC. Let $\mathbf{T}_{I:G} \in \mathbb{SE}(3)$ be the rigid body transformation matrix composed using $\mathbf{p}_I^{IG}$ and $\mathbf{q}_{IG}$, and $\mathbf{p}_{f_j}^{f_j G}$ be a 3D landmark point, estimated through key-frame triangulation. Then, the point re-projection error term for the i^{th} cluster camera is defined as

$$e_j^i(\mathbf{T}_{I:G}) = z_j^i - \Psi^i(\mathbf{T}_{C_i:I}\mathbf{T}_{I:G}\mathbf{p}_{f_j}^{f_j G}) \tag{4.7}$$

where z_j^i is the pixel measurement in camera i of landmark $\mathbf{p}_{f_j}^{f_j G}$, and $\mathbf{T}_{C_i:I}$ is the extrinsic calibration between the IMU sensor and camera C_i. The re-projection error term from Eq. (4.7) is combined with additional IMU and key-frame error terms, in order to optimize both the robot and landmark states.

In order to perform estimation of the mechanism joint angles, the joint angles to be estimated at time k is augmented to the localization state vector

$$\mathbf{x}_k^{\beta_k} = [\mathbf{x}_k \; \beta_k]^T \tag{4.8}$$

where $\beta_k = [\theta_1^k \; \cdots \; \theta_L^k]$ are the L joint angles of the mechanism to be optimized. When using a DCC, the extrinsics between the *dynamic* camera and IMU, $\mathbf{T}_{d:I}$, can be decomposed as

$$\mathbf{T}_{d:I}^{\beta_k} = \mathbf{T}_{d:e}^{\tau_d}\mathbf{T}_{e:b}^{\omega,\beta_k}\mathbf{T}_{b:s}^{\tau_s}\mathbf{T}_{s:I} \tag{4.9}$$

where $\mathbf{T}_{s:I}$ is the transformation from IMU to static camera, which can be computed offline [26]. Using Eq. (4.9), the re-projection error Eq. (4.7) is modified to include the dynamic camera as

$$e_j^i(\mathbf{T}_{I:G}, \beta_k) = z_j^i - \Psi^i(\mathbf{T}_{d:I}^{\beta_k} \mathbf{T}_{I:G} \mathbf{p}_{f_j}^{f_j G}) \tag{4.10}$$

Our extended OKVIS uses the modified re-projection error term from Eq. (4.10), along with the original IMU and key-frame error terms from the original implementation to estimate the augmented robot state vector $\mathbf{x}_{\beta_k}^k$, which includes the mechanism joint angles. To perform the optimization, we analytically compute the Jacobian

$$\frac{\partial e_j^i(\mathbf{T}_{I:G}, \beta_k)}{\partial \beta_k} \tag{4.11}$$

which describes how the modified re-projection error from (4.10) is affected by small changes in the joint angles, β_k.

4.4 Experimental Validation

To validate our proposed approach, two sets of experiments were performed. First, we demonstrate the successful encoder-less calibration of a 2-DOF gimbal on physical hardware. Secondly, we perform visual inertial odometry using the calibrated gimbal, and show that our gimballed DCC VIO configuration performs comparably to a monocular or standard SCC setup.

4.4.1 Encoder-less Gimbal DCC Calibration

The method discussed in Sec. 4.2 was used to calibrate a DCC consisting of a 2-DOF gimbal and two PointGrey Firefly MV cameras, which were software triggered at 15 fps and have a resolution of 640x480. The gimbal is controlled by an Alexmos SimpleBGC 32-bit gimbal motor controller, and the DCC is mounted onto a custom built MAV. The SCC and DCC configurations both have a baseline of approximately 10 cm. The 2-DOF gimbal, along with the labelled frames, is depicted in Fig. 4.1, and the MAV is depicted in Fig. 4.2.

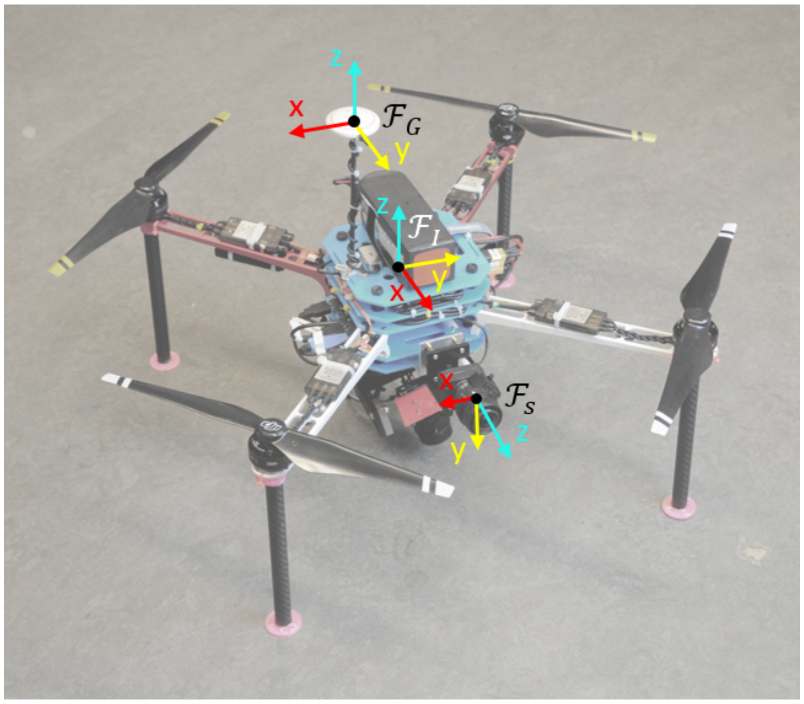


Figure 4.2: Custom MAV hardware with 2-DOF gimbal based DCC mounted on bottom. Frames $\mathcal{F}_s$, $\mathcal{F}_I$, $\mathcal{F}_G$ denote the static camera, IMU, and GPS frames, respectively.

The calibration was performed by collecting a set of 83 independent image pairs with different gimbal configurations. The gimbal configurations were sampled uniformly over the 2-DOF joint angle space. An AprilGrid was used as the fiducial target for calibration, in contrast to a chessboard target, the AprilGrid detection is more robust to large viewpoint changes and partial target observations [26]. For validation of the gimbal calibration, an independent verification set of 70 image pairs was collected, also using the same AprilGrid fiducial target. In both calibration and validation set, we ensured sufficient observation overlap of the fiducial target between the static and gimbal camera. The initial joint angle values for estimation were obtained from the gimbal IMU, often used for camera stablization.

Table 4.1: Re-projection error statistics of the physical gimbal

Dataset	Calibration	Validation
Number of images	83	70
Average re-projection error (pixels)	1.5234	1.7933
Standard deviation (pixels)	0.8487	1.1934

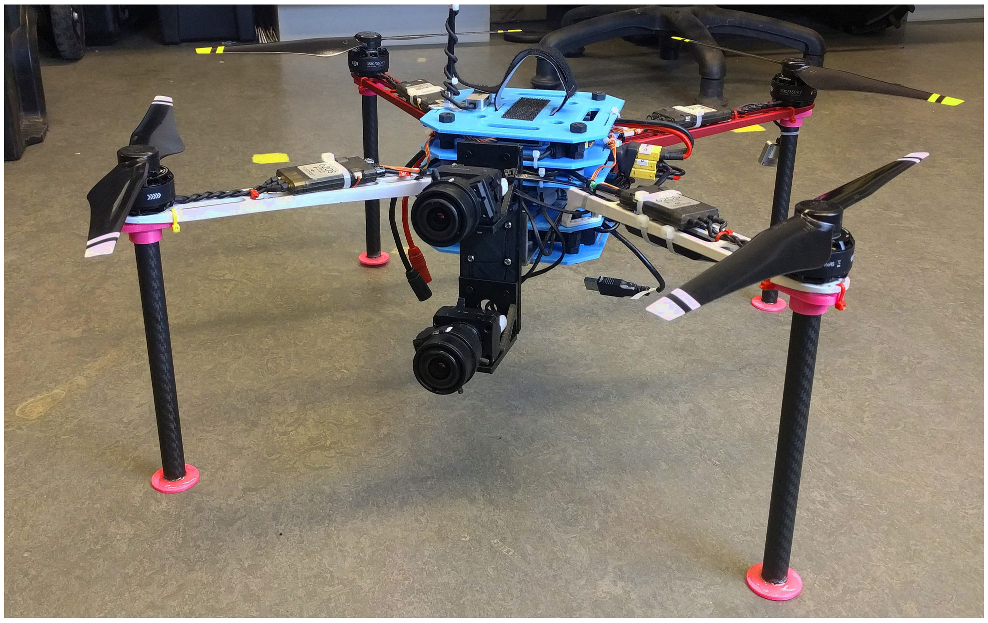
The results for the calibration and verification sets are summarized in Table. 4.1. We see that the average re-projection error for the calibration set is approximately 1.5 pixels, which indicates a good quality calibration was achieved. The average error of the validation set is comparable to the result from the calibration set, at approximately 1.8 pixels, which corroborates the accuracy of the calibrated kinematic parameters. Note that this re-projection error is highly dependent on the quality of the intrinsic calibration, which can be improved through the use of higher quality lenses.

4.4.2 VIO using Gimbal DCC

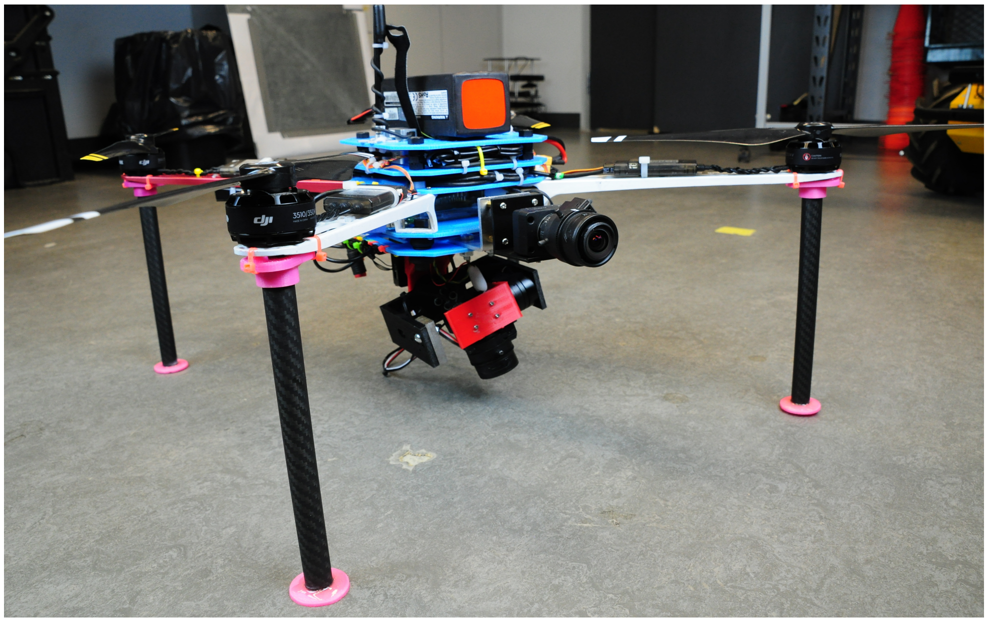
To demonstrate the calibration result, we collected two independent sets of MAV flight data, traversing in the same rectangular trajectory loop depicted in Fig. 4.4, which is approximately 180m in distance and ran it offline against our modified OKVIS. The first flight data used a standard SCC configuration (see Fig. 4.3a), the static transformation between the static camera frame, $\mathcal{F}_s$, and MAV IMU frame, $\mathcal{F}_I$, were obtained using Kalibr [26]. The second flight data, on the other hand, used a DCC configuration (see Fig. 4.3b). During the second flight, the gimbal DCC was pre-programmed to point towards features in the environment we deemed would increase the accuracy of the VIO estimation. The effectiveness of the gimbal viewpoint selection, however, was not evaluated and is left as future work.

Using two independent flight data, three camera configurations over the same flight path, and sensor suite were compared: monocular, SCC and DCC. Sample images from the two datasets are presented in Fig. 4.5. The ground truth of the trajectory is collected using the DJI N3 Autopilot GPS, which has an accuracy of approximately 2m standard deviation.

Trajectory drift was observed in all three tested camera configurations when compared to GPS ground truth. The cause of observed drift could be due to experimental factors, such as the high altitude of the flown trajectory and the lack of IMU excitation. The MAV's flight altitude of 10m in conjunction with the small camera baseline results in inaccurate

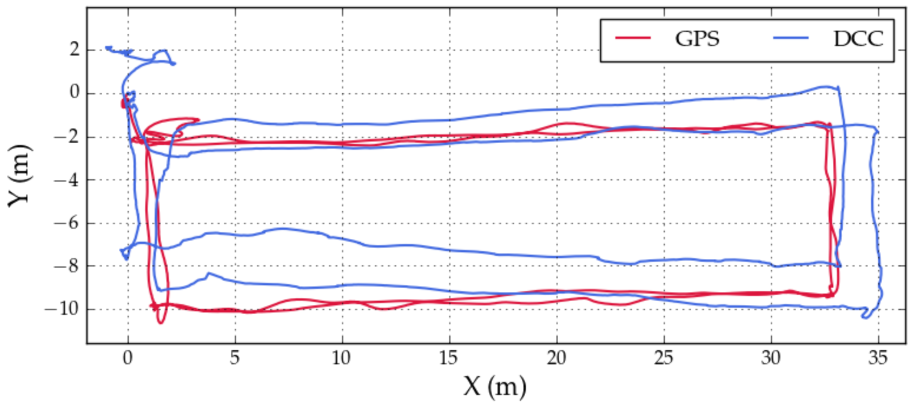


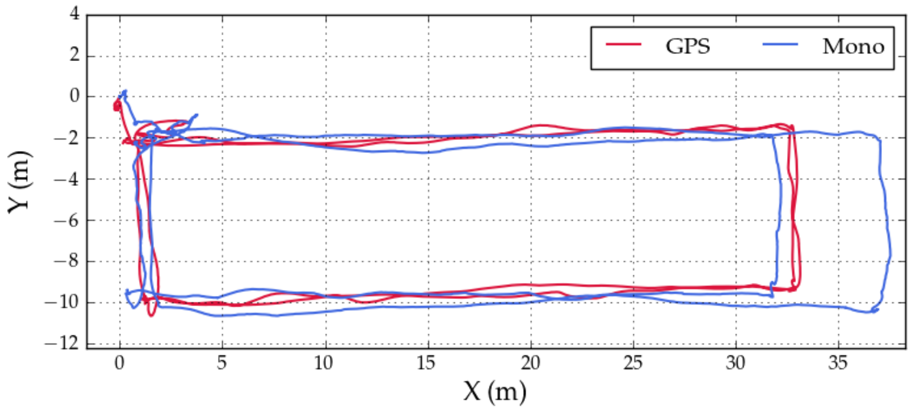
(a) Static Camera Cluster (SCC) Configuration



(b) Dynamic Camera Cluster (DCC) Configuration

Figure 4.3: Camera configurations used for experiments





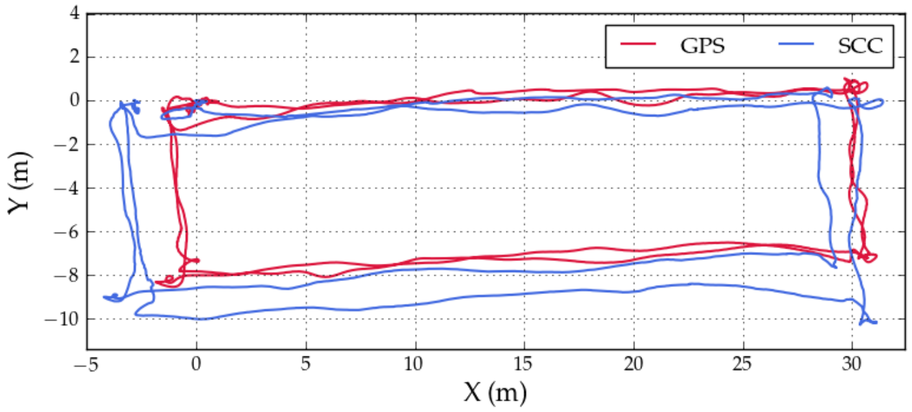
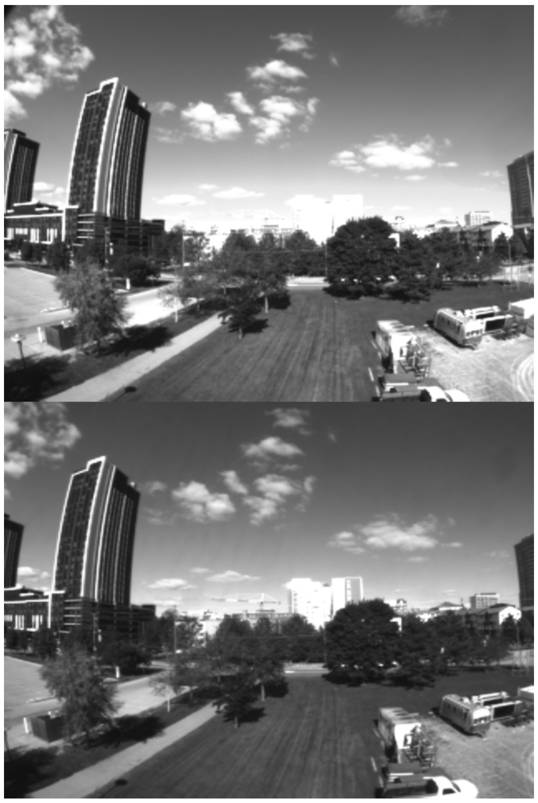
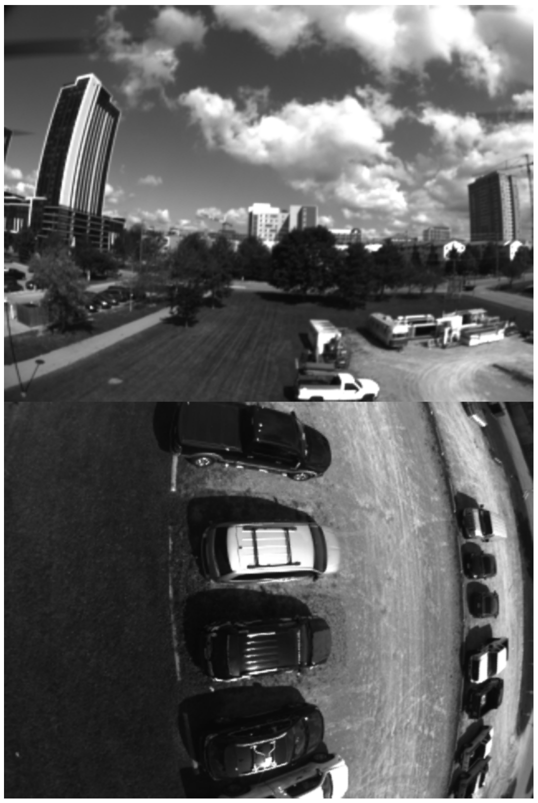


Figure 4.4: Estimated trajectories with (a) Dynamic Camera Cluster, (b) Monocular, and (c) Static Camera Cluster configurations. For all flight scenarios, the MAV starts at (0,0) and then flies two rectangular loops.



(a)



(b)

Figure 4.5: Example frames from the two datasets. a The static configuration, with one camera above the other. b The dynamic configuration, with the gimbal pointed down.

feature depth initialization and a high average feature distance, which in turn causes poor scale observability. In addition to this, the constant-velocity trajectory executed by the MAV inhibits IMU excitation, which could be another factor contributing to the poor scale recovery. Table. 4.2, which reports the RMSE of the trajectory normalized by the total distance flow, shows that the relative performance of each camera configuration is consistent.

Table 4.2: Translation and rotation error of the three camera configurations, normalized by total distance travelled

	DCC	Mono	SCC
Normalized translation RMSE (%)	1.58×10^{-2}	1.24×10^{-2}	1.12×10^{-2}
Normalized rotation RMSE (rad/m)	4.3×10^{-4}	2×10^{-4}	9.4×10^{-4}

From the results, we observe DCC exhibits slightly larger normalized translation error compared the mono and SCC results. This can be attributed to OKVIS's 3D-2D RANSAC step, which selects landmark measurements based on the predicted motion, this step relies on the camera extrinsics. In the case of the DCC, however, we disabled the 3D-2D RANSAC step, because the measurements filtered out by RANSAC are precisely those required to estimate the joint angles for the time-varying extrinsics, thereby potentially introducing outliers to the VIO algorithm for motion estimation. A possible future refinement would be a two-stage approach: first optimize without outlier rejection to obtain the dynamic camera extrinsics, then re-optimize with the updated extrinsics for an improved motion estimate. Despite larger error for the DCC experiment, it is evident that all configurations performed comparably, which verifies that the integration of the DCC did not significantly change the localization performance of the algorithm.

The estimated gimbal joint angles, when compared to the values obtained using encoders mounted on the gimbal for ground truth, shown in Fig. 4.6, shows that our extended OKVIS was able to accurately estimate the joint angles. It is important to state that no motion model or input from the gimbal IMU was used for the estimation, and only visual data from the camera images were used. Table. 4.3 shows the RMSE of the roll and pitch joint angles. A component for the joint angle errors stems from a noticeable time lag in the joint angle estimation, which can be improved through hardware synchronization in the DCC.

Table 4.3: Roll and Pitch RMSE for the DCC configuration.

	RMSE (rad)
Roll angle	4.5×10^{-2}
Pitch angle	9.2×10^{-2}

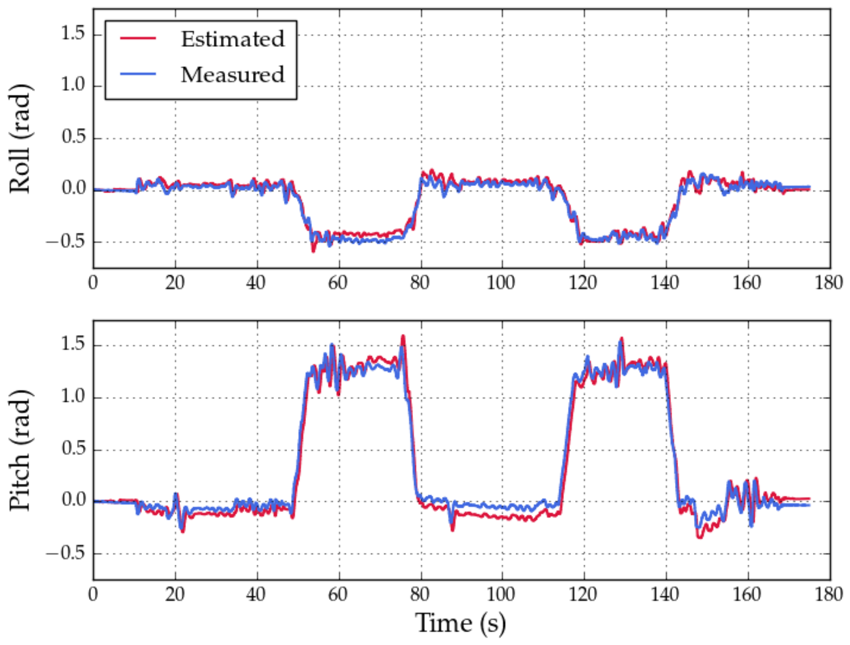


Figure 4.6: Estimated gimbal joint angles compared to ground truth provided by the gimbal encoders.

4.5 Conclusion

This chapter presented a method to perform encoderless calibration of a DCC. We achieved this by jointly estimating the kinematic parameters of the DCC and gimbal joint angles of the mechanism for each measurement of the fiducial target. The calibration result was then validated by modifying OKVIS to perform online estimation of the gimbal joint angles in a DCC configuration. We then demonstrated that the extended VIO was able to successfully estimate the joint angles, and that it performs comparably to a VIO solution using a monocular or SCC camera configuration.

Chapter 5

Gimbal Multi State Constraint Kalman Filter (G-MSCKF)

In Chapter 3, an end-to-end autonomous tracking and landing system (ATL) was introduced. It was demonstrated that a MAV with robust and accurate state estimates could track and execute a planned trajectory to land on top of a moving ground target at speeds of 10 m/s in simulation. In practice, however, the heavy reliance on GPS for precise MAV pose estimation is problematic. First precise GPS sensors for MAV state estimation are expensive. Second, the environment where the MAV performs the landing can affect the GPS signal quality.

To remove the requirement for precise GPS sensors on MAVs, a camera sensor for state estimation was chosen due to its weight, size, and cost. Rigidly mounted cameras on MAVs, however, frequently observe motion blur cause by sudden attitude changes due to wind or aggressive controls. Which is why an encoder-less gimbal calibration method was introduced in Chapter 4. The method enabled state estimation algorithms such as OKVIS to take advantage of the gimbal camera for state estimation. However, an important aspect of full MAV autonomy is the ability for all software components, from low-level sensor drivers to high-level algorithms to run real-time all on-board a MAV. The computational power on-board a MAV is often limited due to the available payload. This adds additional pressure on the optimization based VIO algorithm such as OKVIS to be as efficient as possible while maintaining an accurate pose estimate of the MAV.

In this chapter we propose a filter based VIO algorithm to address the aforementioned issues, because they are computationally more efficient, and they can achieve comparable accuracy compared to optimization based methods. The state of the art filter based VIO algorithm, MSCKF [55], was chosen for its accuracy and efficiency. As for the sensor configuration, a dynamic camera cluster(DCC) is used in this work to take advantage of the image stabilization offered by the gimbal camera. Our main contributions is to incorporate a gimbal camera with MSCKF which we call G-MSCKF. The author is the sole contributor of this work.

5.1 Related Work

Visual-inertial navigation systems have been deployed in a wide range of robotic applications [55, 67, 44, 6], and there exists a large number of estimation algorithms for the camera-IMU sensor combination. These algorithms can be characterized as either *loosely* [77, 66] or *tightly* [55, 44] coupled systems. In *loosely* coupled systems, image and IMU measurements are processed independently before fusing into a single estimate, in contrast, *tightly* coupled systems process both image and IMU measurements together. The advantage of decoupling the sensor measurements in loosely coupled systems is to limit the computational complexity [44], but this comes at the cost of lower accuracy state estimation and biases [46]. For this reason we are focused on tightly coupled systems for the remainder of this chapter.

Existing tightly coupled VIO algorithms can generally be categorized into two groups: optimization based or filter-based approaches. While optimization based approaches such as [21, 44, 73] obtain optimal state estimation by jointly minimizing the residual using measurements from sensors, they are also computationally more expensive compared to the filter-based approaches. An alternative to optimization based approach is the filter-based approach, such as Extended Kalman Filter (EKF) [55], or Unscented Kalman Filter [37], which achieve higher computational efficiency with comparable accuracy.

The most computationally efficient filter-based approaches utilize feature measurements to derive constraints between consecutive pairs of camera frames [6]. However, in order to attain higher estimation accuracy, constraints between multiple camera poses must be made where the same feature is observed in multiple camera frames. Methods such as [27] and [18] implement a sliding window of robot or camera poses in the filter state. Both algorithms form constraints between pairs of camera poses. The drawback of both approaches is that information is lost when a feature is tracked in multiple images, since constraints are not formed across multiple camera poses. To address this drawback, methods such as the Variable State Dimension Filter (VSDF) [54] have been introduced, which uses a delayed linearization to increase robustness against linearization inaccuracies. The VSDF also exploits the sparsity of the information matrix that arises when no dynamic motion model is used. The disadvantage of VSDF is the computational complexity is at best quadratic in the number of features when a dynamic motion model is available (such as in visual-inertial navigation systems) [14].

An algorithm that addresses the issues discussed is the Multi-State Constraint Kalman Filter (MSCKF) [55]. In contrast to the VSDF, the MSCKF exploits the benefit of delayed linearization while maintaining a *linear* computational complexity. MSCKF uses an ESKF formulation with extensions to incorporate vision information over a longer history. In particular, the MSCKF maintains a window of camera poses and simultaneously update them using batch-optimized estimates of features observed across all camera poses in the window. This update step is typically performed when a feature goes out of view of the camera, but it may also be triggered if a feature track length has exceeded a preset threshold. As a result of these properties, the MSCKF was chosen for our tightly coupled filter-based VIO algorithm.

To the best of my knowledge no VIO filter-based solution incorporate a gimbal camera or DCC sensor configuration. Only a few VIO solutions are designed for stereo or multi-camera system [71, 44, 73, 58], due perhaps to the cost associated with additional image processing required.

5.2 State Vector

MSCKF uses an ESKF formulation with extensions to incorporate vision information over a longer history. Instead of augmenting the state vector with feature positions, a window of camera poses are kept. The following sections will focus on MSCKF's extensions over ESKF.

5.2.1 Estimate-state vector

The full estimate-state vector $\hat{\mathbf{x}}_k$ at time k consists of the current IMU state estimate $\hat{\mathbf{x}}_{I,k}$, and N camera poses $\hat{\mathbf{x}}_C$, where N is the latest camera pose. This leads to a state vector of the form

$$\hat{\mathbf{x}}_k = \begin{bmatrix} \hat{\mathbf{x}}_{I,k}^T & \hat{\mathbf{x}}_{C_1}^T & \dots & \hat{\mathbf{x}}_{C_N}^T \end{bmatrix}^T . \tag{5.1}$$

The IMU state $\mathbf{x}_I \in \mathbb{R}^{15}$ is defined as,

$$\hat{\mathbf{x}}_I = \begin{bmatrix} \mathbf{q}_{IG}^T & \mathbf{b}_g^T & \mathbf{v}_G^{IG^T} & \mathbf{b}_a^T & \mathbf{p}_G^{IG^T} \end{bmatrix}^T \tag{5.2}$$

where the unit quaternion $\mathbf{q}_{IG}$ represents the rotation from the global frame $\mathcal{F}_G$ to the IMU frame $\mathcal{F}_I$, $\mathbf{b}_g \in \mathbb{R}^3$ and $\mathbf{b}_a \in \mathbb{R}^3$ are gyroscope and accelerometer biases. Both IMU bias terms are modeled as random walk processes, driven by the white Gaussian noise vectors $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$. The vectors $\mathbf{v}_G^{IG} \in \mathbb{R}^3$ and $\mathbf{p}_G^{IG} \in \mathbb{R}^3$ represents the velocity and position of the IMU from the origin of $\mathcal{F}_G$ to $\mathcal{F}_I$ expressed in $\mathcal{F}_G$. The i-th camera state vector $\mathbf{x}_{C_i}$ is defined as,

$$\hat{\mathbf{x}}_{C_i} = \begin{bmatrix} \mathbf{q}_{C_i G}^T & \mathbf{p}_G^{C_i G T} \end{bmatrix}^T \tag{5.3}$$

which is comprised of a unit quaternion $\mathbf{q}_{C_i G}$ and position $\mathbf{p}_G^{C_i G} \in \mathbb{R}^3$ that describes the i-th camera pose in the global frame.

Note, the camera pose $\hat{\mathbf{x}}_{C_i}$ denotes the static camera in a DCC camera configuration. The pose of the dynamic, or gimbal camera, can be calculated from a transform, $\mathbf{T}_{d:s}$, which can found using the encoder-less gimbal calibration described in Chapter 4.

5.2.2 Error-state vector

Using the true state vector defined in Eq. (5.1) the would cause singularities in the covariance matrices, due to the unit constraint on the quaternions in the true-state vector. Instead the MSCKF estimates the errors of the IMU. The *IMU error-state* vector is defined as,

$$\delta\mathbf{x}_I = \begin{bmatrix} \delta\boldsymbol{\theta}^T & \delta\mathbf{b}_g^T & \delta\mathbf{v}_G^{IGT} & \delta\mathbf{b}_a^T & \delta\mathbf{p}_G^{IGT} \end{bmatrix}^T \tag{5.4}$$

where the standard additive error definition is used for the position, velocity and bias error terms (i.e., the error in the estimate $\hat{x}$ of a quantity x is defined as $\tilde{x} = x - \hat{x}$). The orientation error, on the other hand, is described by the error quaternion $\delta\mathbf{q}$, which is defined by the relation $\mathbf{q} = \delta\mathbf{q} \otimes \hat{\mathbf{q}}$. The symbol $\otimes$ in the previous expression denotes a quaternion multiplication. The error quaternion is related to the error state as,

$$\delta\mathbf{q} \simeq \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}^T & 1 \end{bmatrix}^T \tag{5.5}$$

Here, the error quaternion $\delta\mathbf{q}$ is assumed to describe a small 3 DOF rotation that causes the true and estimated attitude to coincide, thus following [55] the minimal representation $\delta\boldsymbol{\theta}$ was used to describe the attitude errors. Similarly, the i-th *camera error-state* is defined as

$$\delta\mathbf{x}_{C_i} = \begin{bmatrix} \delta\mathbf{q}_{C_i G}^T & \delta\mathbf{p}_G^{C_i G T} \end{bmatrix}^T. \tag{5.6}$$

The complete *error-state* vector is thus,

$$\delta \mathbf{x}_k = \begin{bmatrix} \delta \mathbf{x}_{I,k}^T & \delta \mathbf{x}_{C_1}^T & \dots & \delta \mathbf{x}_{C_N}^T \end{bmatrix}^T . \tag{5.7}$$

To maintain a bounded computational complexity, the camera states are marginalized when the number of camera states exceeds a preset limit.

5.3 Prediction Update

The MSCKF uses a similar ESKF prediction step derived in Section. 2.5, which is summarized here for convenience. Let the nominal state vector $\hat{\mathbf{x}}$, error state vector $\delta\mathbf{x}$, input vector $\mathbf{u}$, and the noise vector $\mathbf{n}_I$ be

$$\hat{\mathbf{x}}_I = \begin{bmatrix} \hat{\mathbf{q}}_{IG} \\ \hat{\mathbf{b}}_g \\ \hat{\mathbf{v}}_G^{IG} \\ \hat{\mathbf{b}}_a \\ \hat{\mathbf{p}}_G^{IG} \end{bmatrix}, \quad \delta\mathbf{x}_I = \begin{bmatrix} \delta\mathbf{q}_{IG} \\ \delta\mathbf{b}_g \\ \delta\mathbf{v}_G^{IG} \\ \delta\mathbf{b}_a \\ \delta\mathbf{p}_G^{IG} \end{bmatrix}, \quad \mathbf{u}_I = \begin{bmatrix} \mathbf{a}_m \\ \boldsymbol{\omega}_m \end{bmatrix}, \quad \mathbf{n}_I = \begin{bmatrix} \mathbf{n}_g \\ \mathbf{n}_{wg} \\ \mathbf{n}_a \\ \mathbf{n}_{wa} \end{bmatrix}. \tag{5.8}$$

where the process noise of the IMU, $\mathbf{n}_I = \begin{bmatrix} \mathbf{n}_g^T & \mathbf{n}_{wg}^T & \mathbf{n}_a^T & \mathbf{n}_{wa}^T \end{bmatrix}$, comprises of the Gaussian noise of the gyroscope and accelerometer measurement with vectors $\mathbf{n}_g$ and $\mathbf{n}_a$, and $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ represents the random walk rate of the gyroscope and accelerometer measurement biases. The nominal state kinematics are,

$$\begin{aligned} \dot{\hat{\mathbf{q}}}_{IG} &= \frac{1}{2}\hat{\mathbf{q}}_{IG} \otimes (\boldsymbol{\omega}_m - \hat{\mathbf{b}}_g) \\ \dot{\hat{\mathbf{b}}}_g &= \mathbf{0} \\ \dot{\hat{\mathbf{v}}}_G^{IG} &= \mathbf{C}\{\hat{\mathbf{q}}_{IG}\}(\mathbf{a}_m - \hat{\mathbf{b}}_a) + \mathbf{g} \\ \dot{\hat{\mathbf{b}}}_a &= \mathbf{0} \\ \dot{\hat{\mathbf{p}}}_G^{IG} &= \hat{\mathbf{v}}_G^{IG} \end{aligned} \tag{5.9}$$

where $\hat{\mathbf{q}}_{IG}$ represents the nominal quaternion, $\boldsymbol{\omega}_m$ is the measured angular velocity, $\hat{\mathbf{b}}_g$ is the nominal gyroscope bias, $\hat{\mathbf{v}}_G^{IG}$ is the nominal velocity, $\mathbf{a}_m$ is the measured acceleration, $\hat{\mathbf{b}}_a$ and $\hat{\mathbf{n}}_a$ are nominal acceleration bias and acceleration noise. The linearized continuous time model for the IMU error state is,

$$\delta\dot{\mathbf{x}}_I = \mathbf{F}_I \delta\mathbf{x}_I + \mathbf{G}_I \mathbf{n}_I \tag{5.10}$$

The matrices $\mathbf{F}_I$ and $\mathbf{G}_I$ in Eq. (5.10) are,

$$\mathbf{F}_I = \begin{bmatrix} -\lfloor \hat{\boldsymbol{\omega}}_I^{IG} \times \rfloor & -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ -\mathbf{C}(\hat{\mathbf{q}}_{IG})^T \lfloor \hat{\mathbf{a}} \times \rfloor & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{C}(\hat{\mathbf{q}}_{IG})^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}_{15\times15}$$

$$\mathbf{G}_I = \begin{bmatrix} -\mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{I}_3 & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{C}(\hat{\mathbf{q}}_{IG})^T & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_3 \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix}_{15\times12}$$

Every time an IMU measurement is received with a time period, ΔT, the 4th order Runge-Kutta numerical integration is used to propagate the IMU nominal state. Before propagating the covariance matrix, it is important to stress that the MSCKF covariance matrix is different from the ESKF's. Specifically the MSCKF covariance matrix includes the covariances of the camera poses from the state vector, partitioned as,

$$\mathbf{P}_{k+1|k} = \begin{bmatrix} \mathbf{P}_{II_{k|k}} & \mathbf{P}_{IC_{k|k}} \\ \mathbf{P}_{IC_{k|k}}^T & \mathbf{P}_{CC_{k|k}} \end{bmatrix} \tag{5.11}$$

where $\mathbf{P}_{II_{k|k}}$ is the 15×15 covariance matrix of the IMU state, $\mathbf{P}_{CC_{k|k}}$ is the $6N \times 6N$ covariance matrix of the camera pose estimates, and finally $\mathbf{P}_{IC_{k|k}}$ is the correlation between the errors in the IMU-gimbal state and the camera pose estimates. With this notation, the covariance matrix of the propagated state can be written as

$$\mathbf{P}_{k+1|k} = \begin{bmatrix} \mathbf{P}_{II_{k|k}} & \Phi(t_k + \Delta T, t_k)\mathbf{P}_{IC_{k|k}} \\ \mathbf{P}_{IC_{k|k}}^T \Phi(t_k + \Delta T, t_k)^T & \mathbf{P}_{CC_{k|k}} \end{bmatrix} \tag{5.12}$$

where the state transition matrix $\Phi(t_k + \Delta T, t_k)$ and the propagated IMU state covariance $\mathbf{P}_{II_{k+1|k}}$ are computed with,

$$\Phi(t_k + \Delta T, t_k) = \mathbf{I}_{15} + \mathbf{F}_I \Delta T \tag{5.13}$$

$$\begin{aligned} \mathbf{P}_{II_{k+1|k}} &= \Phi(t_k + \Delta T, t_k)\mathbf{P}_{II_{k|k}} \Phi^T(t_k + \Delta T, t_k) \\ &\quad + \mathbf{G}_I \mathbf{Q}_I \mathbf{G}_I^T \Delta T. \end{aligned} \tag{5.14}$$

5.4 State Augmentation

Upon new captured images, the MSCKF state is augmented with the new camera state. The pose of the new camera state $N+1$ is computed from the latest IMU state at time k as:

$$\hat{\mathbf{q}}_{C_{N+1}G} = \hat{\mathbf{q}}_{CI} \otimes \hat{\mathbf{q}}_{IG,k} \tag{5.15}$$

$$\hat{\mathbf{p}}_G^{C_{N+1}G} = \hat{\mathbf{p}}_G^{IG} + \mathbf{C}(\hat{\mathbf{q}}_{IG,k})^T \hat{\mathbf{p}}_C^{CI} \tag{5.16}$$

where $\hat{\mathbf{q}}_{CI}$ and $\hat{\mathbf{p}}_C^{CI}$ are the camera-IMU extrinsics, both are known and can be obtained through an offline extrinsics calibration, such as Kalibr [26]. This new camera pose estimate is augmented to the full state vector, and the covariance matrix is augmented by:

$$\mathbf{P}_{k|k} = \begin{bmatrix} \mathbf{I}_{15+6N} \\ \mathbf{J}_k \end{bmatrix} \mathbf{P}_{k|k} \begin{bmatrix} \mathbf{I}_{15+6N} \\ \mathbf{J}_k \end{bmatrix}^T \tag{5.17}$$

where the Jacobian $\mathbf{J}$ is derived from Eq. (5.15) and (5.16) as:

$$\mathbf{J}_k = \begin{bmatrix} \mathbf{C}(\hat{\mathbf{q}}_{CG,k}) & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{0}_{3\times 6N} \\ \lfloor \mathbf{C}(\hat{\mathbf{q}}_{IG,k})^T \hat{\mathbf{p}}_C^{CI} \times \rfloor & \mathbf{0}_{3\times 9} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6N} \end{bmatrix}_{6\times 15+6N} \tag{5.18}$$

5.5 Measurement Model

In the following section, the measurement model used in MSCKF will be presented. The MSCKF algorithm uses the EKF for state estimation, it formulates a measurement model that produces a residual, $\mathbf{r}$, that depends linearly on the *error-state*, $\delta\mathbf{x}$, with the following general form:

$$\mathbf{r} = \mathbf{H}\delta\mathbf{x} + \text{noise} \tag{5.19}$$

where $\mathbf{H}$ is the measurement Jacobian matrix, and the noise is assumed to be a zero-mean, white noise that is uncorrelated to the error state. The contribution of MSCKF compared to other filter based VIO algorithm, lies in the formulation of the measurement model. MSCKF groups camera observations *per tracked feature*, rather than *per camera pose*. Measurements of tracked features are used to constrain camera poses at which the feature was observed, *without* including the feature position in the filter state vector.

The measurement model considers the case of a single feature, f_j, observed by the DCC, but only the static camera pose, $(\mathbf{q}_{C_iG} \quad \mathbf{p}_G^{C_iG})$, will be part of the state vector. The measurement of the j-th feature observed by both cameras in the i-th camera pose, $\mathbf{z}_i^{f_j}$, is represented as,

$$\mathbf{z}_i^{f_j} = \begin{pmatrix} u_{f_j}^{C_{s,i}} \\ v_{f_j}^{C_{s,i}} \end{pmatrix} = 1/Z_j^{C_{s,i}} \begin{pmatrix} X_{f_j}^{C_{s,i}} \\ Y_{f_j}^{C_{s,i}} \end{pmatrix} \tag{5.20}$$

where $(u_{f_j}^{C_{s,i}}, v_{f_j}^{C_{s,i}})$ represents the feature point observed in the image plane by the static camera, and $(X_{f_j}^{s,i} \; Y_{f_j}^{s,i} \; Z_{f_j}^{s,i})^T$, is the position of the feature in the static camera frame. The feature position expressed in the static camera frame is given by,

$$\mathbf{p}_{f_j}^{f_j\,C_{s,i}} = \begin{pmatrix} X_{f_j}^{C_{s,i}} \\ Y_{f_j}^{C_{s,i}} \\ Z_{f_j}^{C_{s,i}} \end{pmatrix} = \mathbf{C}(\mathbf{q}_{C_{s,i},G})(\mathbf{p}_{f_j}^{f_j\,G} - \mathbf{p}_{C_{s,i}}^{C_{s,i}\,G}) \tag{5.21}$$

where $\mathbf{p}_{f_j}^{f_j\,G}$ is the 3D feature position in the global frame, and it is estimated using least-squares minimization discussed in Section. 2.4 to obtain an estimate based on the current estimated camera poses. The initial 3D feature position is initialized using the two views of the same feature from the static and dynamic camera frames of the DCC, and this is the author's contribution towards incorporating a gimbal camera to MSCKF. The transform between the static to dynamic camera transform, $\mathbf{T}_{d:s}$, is assumed to be known and it can be found using the encoderless gimbal calibration method discussed in Chapter 4.

Once the estimate of the feature position in the global frame is found, the measurement residual can be obtained with,

$$\mathbf{r}_i^{f_j} = \mathbf{z}_i^{f_j} - \hat{\mathbf{z}}_i^{f_j}, \tag{5.22}$$

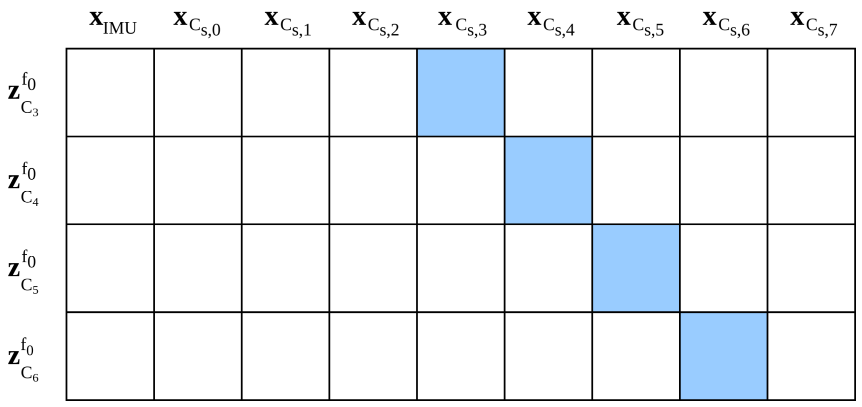
where $\hat{\mathbf{z}}_i^{f_j}$ is the estimated measurement projected from the estimated feature position. Linearizing about the estimates for the camera pose and for the feature position, the residual $\mathbf{r}_i^{f_j}$ in Eq. (5.22) can be approximated as,

$$\mathbf{r}_i^{f_j} \simeq \mathbf{H}_{\mathbf{x}_i}^{f_j} \delta\mathbf{x} + \mathbf{H}_{f_j,C_{s,i}}^{f_j} \delta\mathbf{p}_{f_j}^{f_j\,G} + \mathbf{n}_i^{f_j} \tag{5.23}$$

where $\mathbf{H}_{\mathbf{x}_i}^{f_j}$ and $\mathbf{H}_{f_j,C_{s,i}}^{f_j}$ represent the Jacobians of the measurement $\mathbf{z}_i^{f_j}$ with respect to the state $\mathbf{x}_i$ and feature f_j at the i-th static camera pose $C_{s,i}$, and $\delta\mathbf{p}_{f_j}^{f_j\,G}$ is the error in the feature estimate. To obtain residuals over all camera poses, the residuals are stacked over all measurements (see Fig. 5.1), and so Eq. (5.23) can be rewritten as:

$$\mathbf{r}^{f_j} \simeq \mathbf{H}_{\mathbf{x}}^{f_j} \delta\mathbf{x} + \mathbf{H}_{f_j,C_s}^{f_j} \delta\mathbf{p}_{f_j}^{f_j\,G} + \mathbf{n}^{f_j} \tag{5.24}$$

where $\mathbf{r}^{f_j}$, $\mathbf{H}_{\mathbf{x}}^{f_j}$, $\mathbf{H}_{f_j,C_s}^{f_j}$ and $\mathbf{n}^{f_j}$ are block vectors or matrices.



$\mathbf{x}_{\text{IMU}}$
$\mathbf{x}_{\text{C}_{\text{s},0}}$
$\mathbf{x}_{\text{C}_{\text{s},1}}$
$\mathbf{x}_{\text{C}_{\text{s},2}}$
$\mathbf{x}_{\text{C}_{\text{s},3}}$
$\mathbf{x}_{\text{C}_{\text{s},4}}$
$\mathbf{x}_{\text{C}_{\text{s},5}}$
$\mathbf{x}_{\text{C}_{\text{s},6}}$
$\mathbf{x}_{\text{C}_{\text{s},7}}$
$\mathbf{z}_{\text{C}_3}^{\text{f}_0}$
$\mathbf{z}_{\text{C}_4}^{\text{f}_0}$
$\mathbf{z}_{\text{C}_5}^{\text{f}_0}$
$\mathbf{z}_{\text{C}_6}^{\text{f}_0}$

Figure 5.1: Stacked Jacobian $\mathbf{H}_{\mathbf{x}}$ for a single feature, f_0, that was tracked from the third to sixth camera pose. The nonzero entries are marked in blue in which the feature was detected and tracked. Note that all features should have a non-zero entry in the second to last camera pose in the sliding window, since they are tracked up until the most recent image.

The issue with (5.24) is that the state estimate, $\hat{\mathbf{x}}$, is used to compute the feature position estimate, therefore the feature position error, $\delta\mathbf{p}_{f_j}^{f_j\,G}$, correlates with the error state, $\delta\mathbf{x}$, and thus Eq. (5.24) cannot be used directly. To remedy this problem, following [55], null space marginalization can be used to remove the correlation. This is achieved by creating the left null space, $\mathbf{A}^T$, a unitary matrix whose columns form the basis of the left null space of the matrix, $\mathbf{H}_{f_j,C_{s,i}}^{f_j}$. Once the left null space, $\mathbf{A}^T$, is found it can be used to project $\mathbf{r}^{f_j}$ on the left null space of $\mathbf{H}_{f_j,C_{s,i}}^{f_j}$ to form a new residual $\mathbf{r}_o^{f_j}$.

$$\mathbf{r}_o^{f_j} = \mathbf{A}^T \underbrace{\left(\mathbf{z}^{f_j} - \hat{\mathbf{z}}^{f_j}\right)}_{\mathbf{r}^{f_j}} \simeq \mathbf{A}^T\mathbf{H}_{\mathbf{x}}^{f_j}\delta\mathbf{x} + \underbrace{\mathbf{A}^T\mathbf{H}_{f_j,C_s}^{f_j}}_{=\mathbf{0}}\delta\mathbf{p}_{f_j}^{f_j\,G} + \mathbf{A}^T\mathbf{n}^{(j)} \tag{5.25}$$

$$\mathbf{r}_o^{f_j} = \mathbf{H}_o^{f_j}\delta\mathbf{x} + \mathbf{n}_o^{f_j} \tag{5.26}$$

Using Eq. (5.26), the update step of the EKF can be performed.

5.6 Measurement Update

In the previous section, the measurement model was presented, and it describes the geometric constraints imposed by observing a static feature from multiple camera poses. This section details how the EKF measurement update in the MSCKF is triggered. There are two possible measurement update scenarios:

1. **A tracked feature becomes lost**: when a feature that has been tracked in a number of camera frames is lost, the measurements of this feature (a.k.a feature track) along with the camera poses are used to estimate the feature position in the global frame. This, in turn is used to form measurement residuals (see Sec. 5.5) for the measurement update phase of the EKF.

2. **Max number of camera poses is reached**: upon every new camera frame, the filter state vector is augmented with a new camera pose (see Sec. 5.4). Once the preset max number of camera poses is reached, the tracked features that correspond to the camera states about to be removed are marginalized, similar to when features are lost.

Continuing from the previous section with Eq. (5.26)

$$\mathbf{r}_o^{f_j} = \mathbf{H}_o^{f_j}\delta\mathbf{x} + \mathbf{n}_o^{f_j}$$

an issue that arises with the residual vector, $\mathbf{r}_o^{f_j}$, is that it can be quite large in practice. Therefore the QR-decomposition of $\mathbf{H}_o^{f_j}$ is used to reduce the computational complexity of the EKF update,

$$\mathbf{H}_o^{f_j} = \begin{bmatrix}\mathbf{Q}_1 & \mathbf{Q}_2\end{bmatrix}\begin{bmatrix}\mathbf{T}_H \\ \mathbf{0}\end{bmatrix}, \tag{5.27}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are unitary matrices and $\mathbf{T}_H$ is an upper triangular matrix. Substituting Eq. (5.27) into Eq. (5.26) yields,

$$\mathbf{r}_o^{f_j} = [\mathbf{Q}_1 \;\; \mathbf{Q}_2]\begin{bmatrix}\mathbf{T}_H \\ \mathbf{0}\end{bmatrix}\delta\mathbf{x} + \mathbf{n}_o \tag{5.28}$$

$$\begin{bmatrix}\mathbf{Q}_1^T & \mathbf{r}_o^{f_j} \\ \mathbf{Q}_2^T & \mathbf{r}_o^{f_j}\end{bmatrix} = \begin{bmatrix}\mathbf{T}_H \\ \mathbf{0}\end{bmatrix}\delta\mathbf{x} + \begin{bmatrix}\mathbf{Q}_1^T\mathbf{n}_o \\ \mathbf{Q}_2^T\mathbf{n}_o\end{bmatrix}. \tag{5.29}$$

The final steps of the measurement update is to compute the Kalman gain, compute corrections to the state and finally update the state covariance matrix with the following equations:

$$\mathbf{K} = \mathbf{P}\mathbf{T}_H^T(\mathbf{T}_H\mathbf{P}\mathbf{T}_H^T + \mathbf{R}_n)^{-1} \tag{5.30}$$

$$\Delta\mathbf{x} = \mathbf{K}\mathbf{r}_n \tag{5.31}$$

$$\mathbf{P}_{k+1|k+1} = (\mathbf{I}_{15+6N} - \mathbf{K}\mathbf{T}_H)\mathbf{P}_{k+1|k}(\mathbf{I}_{15+6N} - \mathbf{K}\mathbf{T}_H)^T + \mathbf{K}\mathbf{R}_n\mathbf{K}^T \tag{5.32}$$

After the measurement update, if the sliding window exceeds the preset maximum length, the oldest camera pose along with the corresponding rows and columns in covariance matrix are removed.

5.7 Experiments and Results

To validate the proposed G-MSCKF described, offline visual inertial odometry was performed on three different datasets. The first dataset is in simulation, second dataset is from the KITTI raw dataset, and finally a real-world dataset is collected using the Triclops Sensor Module (TSM).

5.7.1 Simulation Results

A simulation experiment was carried out to verify the performance of the proposed algorithm. We used a Bezier curve method to simulate a smooth camera trajectory by interpolating between position setpoints in the simulation. The first and second derivative can be calculated to obtain the velocity and acceleration of the camera. One Bezier curve describes the position, velocity and acceleration while the second Bezier curve describes the rotation and rotational velocity.

Features observed in the simulation were randomly generated at the start. In each camera frame each feature was checked to see if they are within the camera's field of view. If they are within the camera's field of view the feature is tracked, and if not, they are used for measurement update in G-MSCKF.

Fig. 5.2 shows the estimated (blue) and ground truth (red) camera trajectory travelling from $(0, 0, 0)$ to $(20, 10, 5)$ over 10 seconds. For this particular camera trajectory, $10,000$ visual features were randomly placed on the walls of a 60 m $\times$ 60 m $\times$ 30 m simulated room centered at $(0, 0, 0)$. The camera captured images at 10 Hz and traversed the trajectory

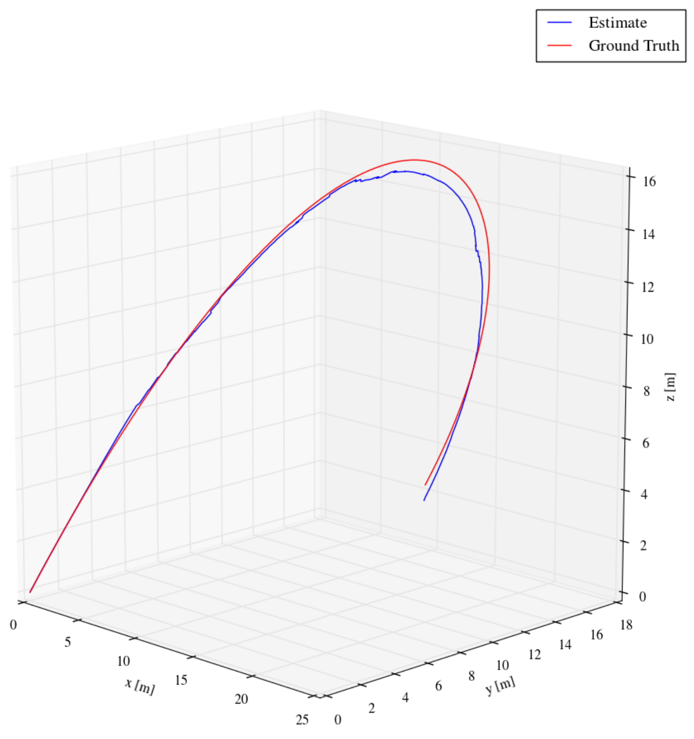


Figure 5.2: Estimate and Ground Truth Trajectory

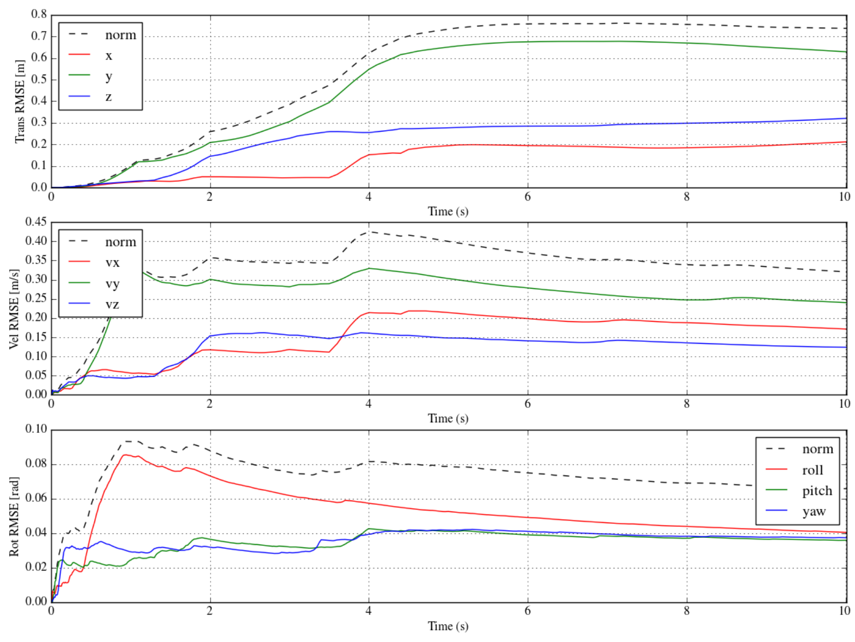


Figure 5.3: RMSE Errors in Simulation

Table 5.1: RMSE translation and rotation error of the simulated camera trajectory

RMSE Trans [m]	x	0.2123
	y	0.6285
	z	0.3211
RMSE Rot [rad]	roll	0.0404
	pitch	0.0358
	yaw	0.0373

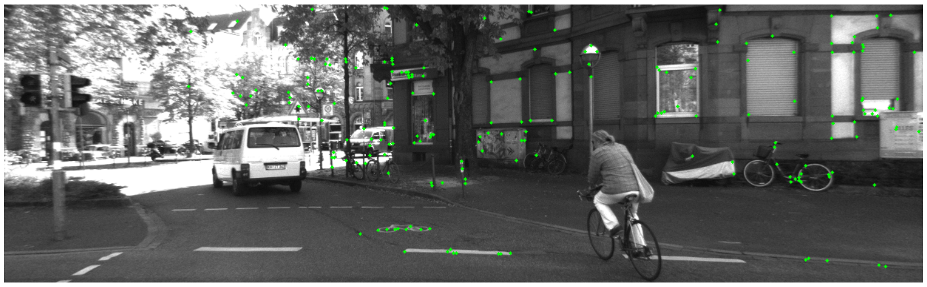
at a velocity of 7.11 m/sec. Each camera frame was accompanied with the gimbal joint angles, which were varied between -0.03 and 0.03 radians over the whole trajectory. The IMU measurements were calculated from the derivatives of the Bezier curves at a rate of 100Hz.

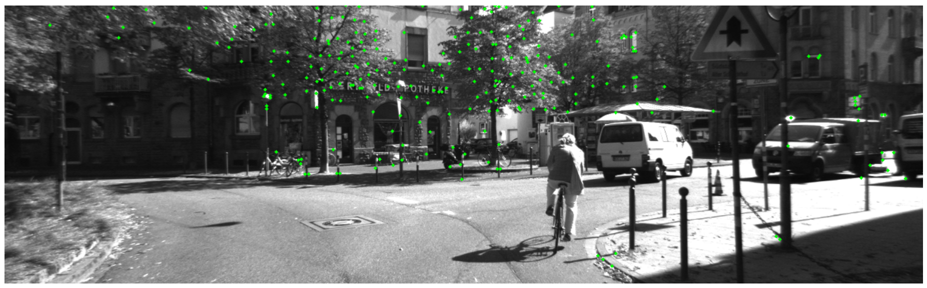
The resulting RMSE position error in x, y and z are 0.2123m, 0.6285 m and 0.3211 m respectively. The RMSE orientation error in roll, pitch and yaw are 0.0404 rad, 0.0358 rad and 0.0358 rad respectively. The final position error over a path length of approximately 71.22 m was 0.71 m, or approximately 1% of the total distance travelled. While there is a large roll, position and velocity error in the y direction relative to other components, these errors occurred early and is maintained to the end. The root cause could be due to the rapid rotations and translations at the start of the simulation. Several similar experiments were performed and show consistency across all trials, demonstrating the consistency of the G-MSCKF formulation in simulation.

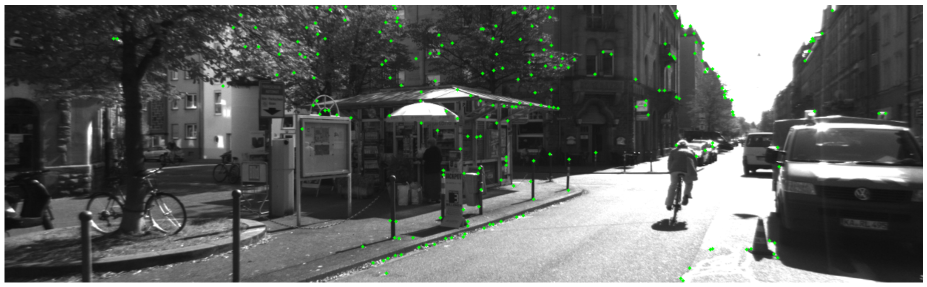
5.7.2 KITTI Data Results

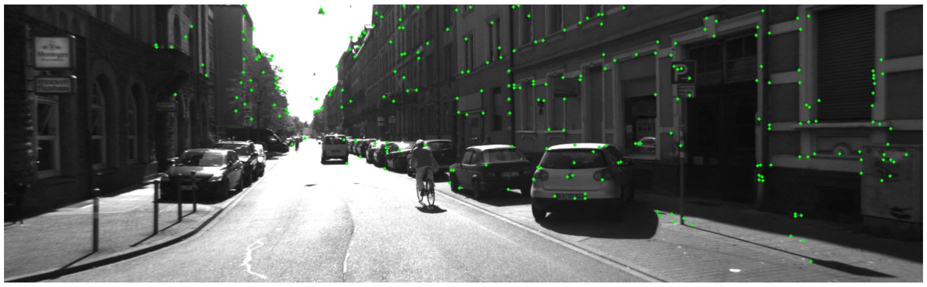
To test the proposed G-MSCKF on real data, the algorithm was evaluated on a KITTI raw dataset sequence [28], specifically sequence 0005. The dataset sequence, however, was not captured using a dynamic camera cluster (DCC) or gimbal camera configuration, therefore the gimbal camera joint angles were fixed throughout the whole trajectory, essentially emulating a stereo camera configuration.

For visual inertial odometry, only data collected from two PointGrey Flea2 grayscale cameras (FL2-14S3MC) capturing images at 10Hz with a resolution of 1392×512, and an OXTS RT3003 inertial and GPS navigation system running at 100Hz were used. The camera images were synchronized with the 3D Velodyne Lidar onboard, where as the GPS/IMU data was collected at 100Hz and aligned to the closest Lidar timestamps, resulting in a worst-case time difference of 5ms between the camera frame and IMU measurement. Fig. 5.4 shows features tracked by the front-end of G-MSCKF.


(a) Time: 1.6s


(b) Time: 5.8s


(c) Time: 9.5s



(d) Time: 13.3s

Figure 5.4: KITTI Raw Dataset - 2011 09 26 - Sequence 0005

Table 5.2: RMSE translation and rotation error of the camera trajectory over sequence 0005 of the KITTI raw dataset

RMSE Trans [m]	x	2.7672
	y	1.0893
	z	2.2182
RMSE Rot [rad]	roll	0.0427
	pitch	0.0275
	yaw	0.1985

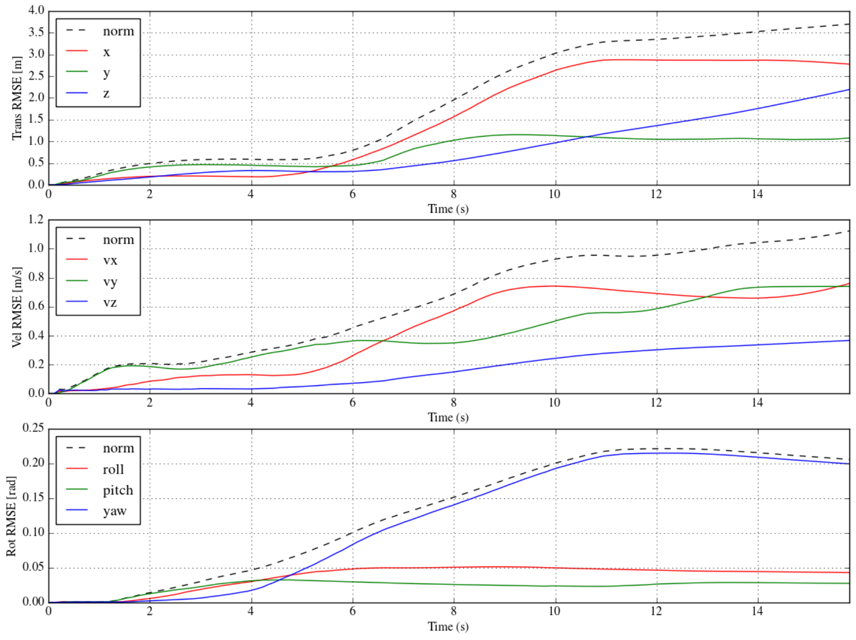
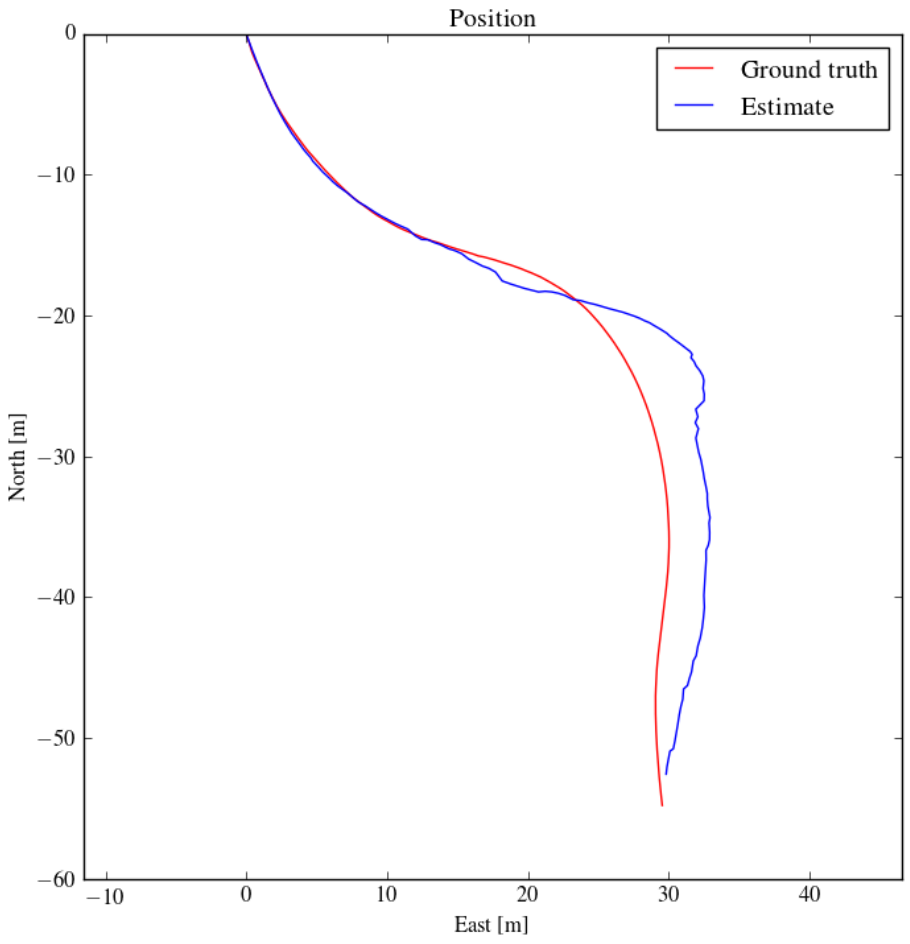


Figure 5.6: RMSE Errors on KITTI Raw Dataset Sequence



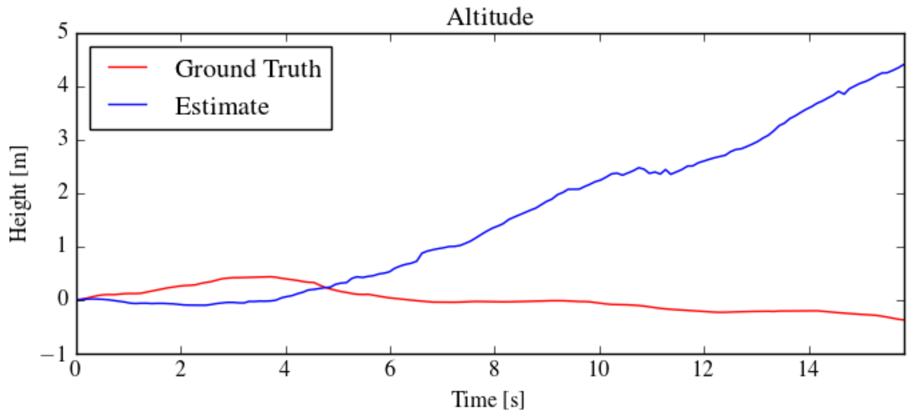


Figure 5.5: Estimated and Ground Truth Position and Altitude

Due to the moving objects in the scene (cyclist and van; see Fig. 5.4) throughout the whole dataset sequence and rapid camera motions, the estimated position and attitude had large RMSE errors. In particular the altitude and roll, leading to a final position error of 5.27 m over a path length of $\approx$ 87 m, or 6% of the total distance travelled. The cause for large errors began at around 5.8 seconds into the sequence, specifically when the camera rotates horizontally to the right rapidly causing difficulty for the feature tracker. This can be observed in Fig. 5.4b, where most of the detected features are above the horizontal ground plane and not uniformly distributed, thus causing the estimated height to start diverging. Inspecting the RMSE errors in Fig 5.6 supports the observation that the errors started to take affect from that moment onwards. Additionally, similar to [12] during experimentation G-MSCKF was found to be sensitive to tuning parameters. Varying general parameters such as sliding-window size or minimum track length could cause the estimation to diverge wildly.

5.7.3 Real World Data Results

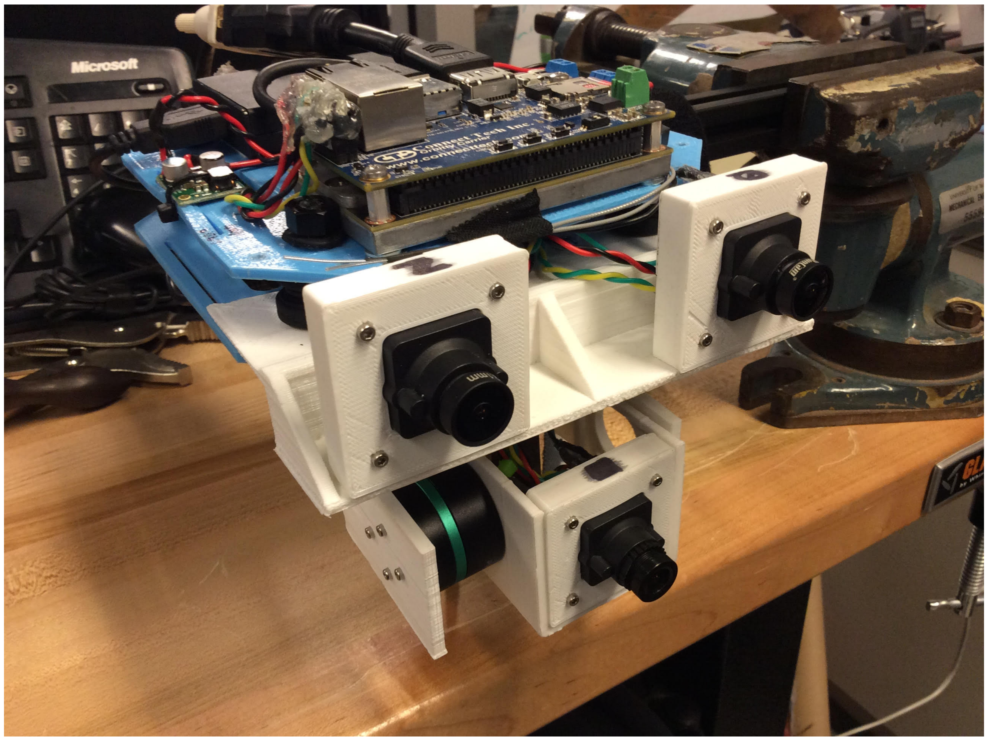


Figure 5.7: Triclops Sensing Module (TSM)

In addition to the KITTI raw datasets, the performance of G-MSCKF on data collected by the Triclops Sensing Module (TSM) shown in Fig. 5.7 was compared. The Triclops sensing unit consists of three IDS UI-1221LE monochrome camera capturing images at 20Hz with a resolution of 752×480, and a XSens XTi-300 IMU running at 100Hz. The aim with the TSM is form a VIO benchmark platform where monocular, stereo and DCC camera configurations can be compared within the same dataset. To the best of our knowledge no VIO dataset contains a gimbal camera as part of the sensor suite.

In photogrammetry, it is a known fact camera frames should be timestamped at mid-exposure [26]. The IDS UI-1221LE cameras, however, timestamps the camera frame at end of the camera exposure. To resolve this problem the method by [56] was used to negatively offset the trigger signal by half of the camera's exposure time, and timestamped the camera frame with the IMU's timestamp instead. For our purpose, the XSens XTi-300 IMU was used as the camera trigger source. The camera intrinsics, extrinsics of the rigidly mounted stereo pair, and extrinsics between the rigidly mounted stereo pair and IMU were calibrated using Kalibr [26]. The extrinsics between the first static camera and dynamic camera (gimbal camera) or DCC, on the other hand, were calibrated using the encoder-less gimbal calibration method introduced in Chapter 4.

A problem occurred after the encoderless gimbal calibration, when validating the gimbal extrinsics it was found that the optimized transform from the static to dynamic camera, $\mathbf{T}_{d:s}$, does not model the TSM gimbal well. In constrast to the stereo camera configuration on board the TSM (see Fig. 5.9) which has an RMSE reprojection error of 1.09 pixels, the gimbal camera RMSE reprojection error is significantly higher when the joint angles vary significantly (see Fig. 5.10, where the error varies from 1.5 to 2.38 pixels). The limitations in the calibration approach meant that G-MSCKF performance was very poor, and more effort is needed in future work to reduce those errors due to the extreme sensitivity of G-MSCKF to calibration and noise parameters. As such it was not possible to validate the G-MSCKF approach appropriately. In future, this could be addressed by jointly optimizing the DCC transform, $\mathbf{T}_{d:s}$, as part of the VIO formulation, however, due to time constraints, I was unable to incorporate them into the G-MSCKF formulation at this time.

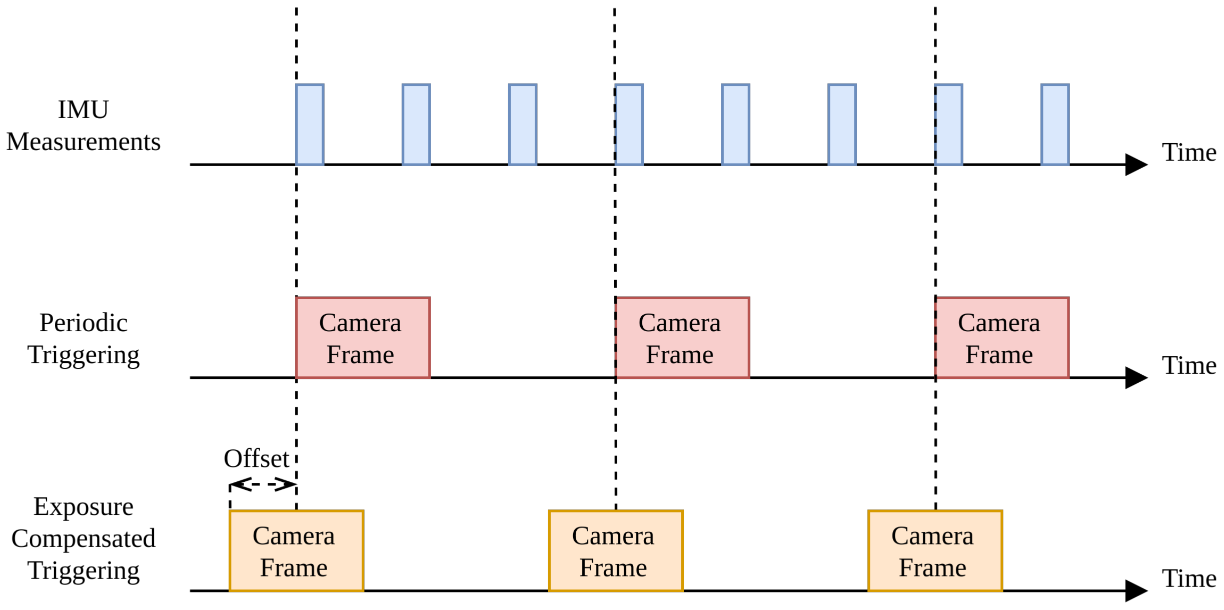
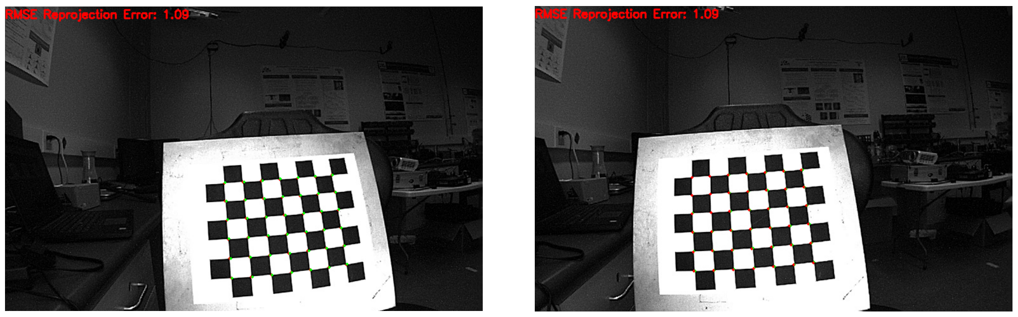
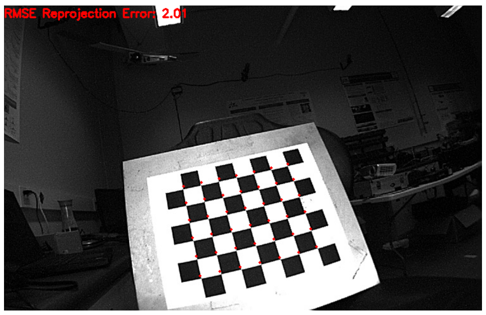


Figure 5.8: Difference between synchronizing the cameras using periodic triggering and exposure compensated triggering. Assuming the camera frames are timestamped at the end of exposure, periodic triggering introduces observable time-differences between IMU measurements and camera frame. Exposure compensated triggering aims to mitigate this problem.

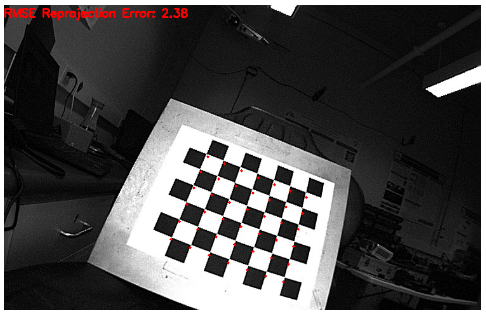


(a) Left camera frame (b) Right camera frame

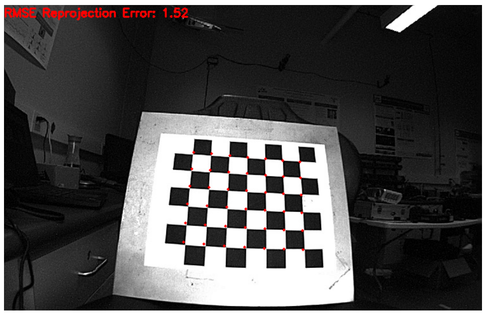
Figure 5.9: Stereo Extrinsics Validation: The green dots in the left camera frame, denotes the detected chessboard corners on the right frame projected onto the left frame, and vice versa with the red dots. A reprojection error of 1.09 pixels indicates a good calibration.



(a)



(b)



(c)

Figure 5.10: Gimbal Extrinsics Validation: The red dots are the detected chessboard corners from the static camera projected onto the image of the gimbal camera. It can be observed that the reprojection errors are larger in (a) and (b) when gimbal joint angles are large, compared to the gimbal configuration near $(0, 0)$ in (c). A reprojection error of higher than 1.5 pixels is deemed a bad calibration.

Chapter 6

Conclusion

In this thesis, three different issues related to autonomous MAV landings were investigated. First, an end-to-end system to perform autonomous autonomous landing of a MAV onto a moving ground target was demonstrated in Chapter 3. The result was an end-to-end landing system called ATL. The system showed the possibility of using low-cost sensors for autonomous MAV landings, and it was tested and demonstrated in simulation using ROS and Gazebo. Noteworthy contributions were made on robust landing target detection at a higher rate.

Secondly, an encoder-less gimbal calibration to enable active vision for vision based localization and mapping algorithms was demonstrated in Chapter 4. Where contributions were made towards an encoderless gimbal calibration to enable state estimation algorithms to make use of a gimbal camera. The method was validated by modifying OKVIS to jointly optimize for the weighed re-projection error, temporal error term from the IMU, and gimbal joint angles.

Lastly, a gimbal camera was incorporated with a filter-based VIO for efficient and robust MAV state estimation in Chapter 5. Although it was only demonstrated in simulation, it is the author's belief that jointly estimating the gimbal transform as part of the filter state vector will help resolve issues met during real-world experiments.

6.1 Future Work

The work presented in this thesis has significant potential on improving autonomous MAV landing. For future work validating the ATL system on a real physical MAV, and perform landing in night time conditions would demonstrate the robustness of the proposed illumination invariant AprilTag.

Secondly, gimbal cameras are increasingly being used for image stabilization on MAVs, and with gimbal cameras on MAVs becoming as ubiquitous as MAVs itself today, it would be advantageous to utilize the gimbal camera for state estimation. However, before it can be used as part of a state estimation algorithm, it is important to obtain a good gimbal calibration. An encoder-less gimbal calibration was introduced in Chapter 4, however, the calibration method does not model the configurations of a custom gimbal well, leading to large reprojection errors while varying the gimbal joint angles. Future work would include a more thorough degeneracy analysis of the gimbal calibration method, and methods for a more precise gimbal calibration for custom gimbals.

Lastly, the work on G-MSCKF showed both the promise of a light-weight filter for state estimation and the extreme sensitivity of the MSCKF approach to outliers, tuning parameters and measurement noise on a real world dataset. Due to its sensitivity to tuning parameters, a simpler formulation of a filter-based VIO should be considered for robust MAV flight, such as ROVIO [6]. In contrast to MSCKF, ROVIO does not form multi-camera pose constraints with the environment for localization. Instead it forms constraints between pairs of camera poses. The computational cost of ROVIO is slightly higher than MSCKF, but results in a simpler implementation as the book-keeping to maintain camera poses in the filter state, and feature tracks currently tracking can be a burden from an implementation standpoint.

References

[1] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision (ECCV)*, pages 404–417. Springer, 2006.

[2] D. S. Bayard and P. Brugarolas. An estimation algorithm for vision-based exploration of small bodies in space. *American Control Conference*, pages 4589–4595 vol. 7, 2005.

[3] Marius Beul, Sebastian Houben, Matthias Nieuwenhuisen, and Sven Behnke. Fast autonomous landing on a moving target at mbzirc. In *Mobile Robots European Conference on Mobile Robots (EMCR)*, pages 1–6. IEEE, 2017.

[4] Yingcai Bi and Haibin Duan. Implementation of autonomous visual tracking and landing for a low-cost quadrotor. *Optik-International Journal for Light and Electron Optics*, 124(18):3296–3300, 2013.

[5] M. Bloesch, H. Sommer, T. Laidlow, M. Burri, G. Nuetzi, P. Fankhauser, D. Bellicoso, C. Gehring, S. Leutenegger, M. Hutter, and R. Siegwart. A Primer on the Differential Calculus of 3D Orientations. *ArXiv e-prints*, June 2016.

[6] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct EKF-based approach. In *Intelligent Robots and Systems (IROS)*, pages 298–304. IEEE, 2015.

[7] Alexandre Borowczyk, Duc-Tien Nguyen, André Phu-Van Nguyen, Dang Quang Nguyen, David Saussié, and Jerome Le Ny. Autonomous landing of a quadcopter on a high-speed ground vehicle. *Journal of Guidance, Control, and Dynamics*, 40(9):2378–2385, 2017.

[8] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research (IJRR)*, 35(10):1157–1163, 2016.

[9] Michael Calonder, Vincent Lepetit, Mustafa Ozuysal, Tomasz Trzcinski, Christoph Strecha, and Pascal Fua. Brief: Computing a local binary descriptor very fast. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 34(7):1281–1298, 2012.

[10] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European Conference on Computer Vision (ECCV)*, pages 778–792. Springer, 2010.

[11] Christopher L. Choi, Jason Rebello, Leonid Koppel, Pranav Ganti, Arun Das, and Steven L. Waslander. Encoderless gimbal calibration of dynamic multi-camera clusters. In *IEEE International Conference on Robotics and Automation (ICRA)*, May 2018.

[12] Lee E Clement, Valentin Peretroukhin, Jacob Lambert, and Jonathan Kelly. The battle for filter supremacy: a comparative study of the multi-state constraint kalman filter and the sliding window filter. In *Computer and Robot Vision (CRV)*, pages 23–30. IEEE, 2015.

[13] Arun Das and Steven L Waslander. Calibration of a dynamic camera cluster for multi-camera visual SLAM. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4637–4642, 2016.

[14] Matthew C Deans. Maximally informative statistics for localization and mapping. In *International Conference on Robotics and Automation (ICRA)*, pages 1824–1829. IEEE, 2002.

[15] David D Diel. *Stochastic constraints for vision-aided inertial navigation*. PhD thesis, Massachusetts Institute of Technology, 2005.

[16] John A Dougherty and Taeyoung Lee. Monocular estimation of ground orientation for autonomous landing of a quadrotor. *Journal of Guidance, Control, and Dynamics*, 39(6):1407–1416, 2016.

[17] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 40:611–625, 2018.

[18] Ryan Eustice, Hanumant Singh, John J Leonard, Matthew R Walter, and Robert Ballard. Visually navigating the RMS titanic with SLAM information filters. In *Robotics: Science and Systems*, pages 57–64, 2005.

[19] Matthias Faessler, Flavio Fontana, Christian Forster, Elias Mueggler, Matia Pizzoli, and Davide Scaramuzza. Autonomous, vision-based flight and live dense 3D mapping with a quadrotor micro aerial vehicle. *Journal of Field Robotics*, 33(4):431–450, 2016.

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

[21] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 15–22. IEEE, 2014.

[22] Alessandro Freddi, Alexander Lanzon, and Sauro Longhi. A feedback linearization approach to fault tolerance in quadrotor vehicles. *IFAC Proceedings Volumes*, 44(1):5413–5418, 2011.

[23] Johannes Friis, Ebbe Nielsen, Rasmus Foldager Andersen, Jesper Boending, Anders Jochumsen, and A Friis. Autonomous landing on a moving platform. *Control Engineering, 8th Semester Project, Aalborg University, Denmark*, 2009.

[24] Simone Frintrop and Patric Jensfelt. Attentional landmarks and active gaze control for visual SLAM. *IEEE Transactions on Robotics*, 24(5):1054–1065, 2008.

[25] Mengyin Fu, Kuan Zhang, Yang Yi, and Chao Shi. Autonomous landing of a quadrotor on an UGV. In *IEEE International Conference on Mechatronics and Automation (ICMA)*, pages 988–993, 2016.

[26] Paul Furgale, Joern Rehder, and Roland Siegwart. Unified temporal and spatial calibration for multi-sensor systems. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1280–1286, 2013.

[27] R Garcia, J Puig, P Ridao, and X Cufi. Augmented state kalman filtering for AUV navigation. pages 4010–4015 vol.4, May 2002.

[28] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The KITTI dataset. *The International Journal of Robotics Research (IJRR)*, 32(11):1231–1237, 2013.

[29] Mark Hardesty, Sandy Kennedy, Sheena Dixon, Travis Berka, Jason Graham, and Don Caldwell. Development of navigation and automated flight control system solutions for maritime VTOL UAS operations. *USNA12*, pages 1–20, 2012.

[30] Richard Scheunemann Hartenberg and Jacques Denavit. *Kinematic synthesis of linkages*. McGraw-Hill, 1964.

[31] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[32] Richard I Hartley and Peter Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[33] Lionel Heng, Gim Hee Lee, and Marc Pollefeys. Self-calibration and visual SLAM with a multi-camera system on a micro aerial vehicle. *Autonomous Robots*, 39(3):259–277, 2015.

[34] B. Herisse, T. Hamel, R. Mahony, and F.-X. Russotto. Landing a VTOL unmanned aerial vehicle on a moving platform using optical flow. *IEEE Transactions on Robotics*, 28(1):77–89, Feb 2012.

[35] Gabriel M Hoffmann, Haomiao Huang, Steven L Waslander, and Claire J Tomlin. Precision flight control for a multi-vehicle quadrotor helicopter testbed. *Control Engineering Practice*, 19(9):1023–1036, 2011.

[36] Botao Hu and Sandipan Mishra. A time-optimal trajectory generation algorithm for quadrotor landing onto a moving platform. In *American Control Conference (ACC)*, pages 4183–4188, 2017.

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

[38] JeongWoon Kim, Yeondeuk Jung, D. Lee, and D.H. Shim. Outdoor autonomous landing on a moving platform for quadrotors using an omnidirectional camera. In *International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1243–1252, May 2014.

[39] Weiwei Kong, Daibing Zhang, Xun Wang, Zhiwen Xian, and Jianwei Zhang. Autonomous landing of a UAV with a ground-based actuated infrared stereo vision system. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2963–2970, 2013.

[40] Pierre Lébraly, Eric Royer, Omar Ait-Aider, Clément Deymier, and Michel Dhome. Fast calibration of embedded non-overlapping cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 221–227, 2011.

[41] Daewon Lee, T. Ryan, and H.J. Kim. Autonomous landing of a VTOL UAV on a moving platform using image-based visual servoing. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 971–976, May 2012.

[42] Robert C Leishman, John C Macdonald, Randal W Beard, and Timothy W McLain. Quadrotors and accelerometers: State estimation with an improved dynamic model. *IEEE Control Systems*, 34(1):28–41, 2014.

[43] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2548–2555, 2011.

[44] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research (IJRR)*, 34(3):314–334, 2015.

[45] Bo Li, Lionel Heng, Kevin Koser, and Marc Pollefeys. A multiple-camera system calibration toolbox using a feature descriptor-based calibration pattern. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1301–1307, 2013.

[46] Mingyang Li and Anastasios I Mourikis. High-precision, consistent EKF-based visual-inertial odometry. *The International Journal of Robotics Research (IJRR)*, 32(6):690–711, 2013.

[47] K. Ling, University of Waterloo. Department of Mechanical, and Mechatronics Engineering. *Precision Landing of a Quadrotor UAV on a Moving Target Using Low-cost Sensors*. University of Waterloo, 2014.

[48] David G Lowe. Object recognition from local scale-invariant features. In *IEEE International Conference on Computer Vision (ICCV)*, pages 1150–1157, 1999.

[49] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[50] Will Maddern, Alex Stewart, Colin McManus, Ben Upcroft, Winston Churchill, and Paul Newman. Illumination invariant imaging: Applications in robust vision-based localisation, mapping and classification for autonomous vehicles. In *Proceedings of the*

Visual Place Recognition in Changing Environments Workshop, IEEE International Conference on Robotics and Automation (ICRA), Hong Kong, China, May 2014.

[51] Travis Manderson, Florian Shkurti, and Gregory Dudek. Texture-aware SLAM using stereo imagery and inertial information. In *Conference on Computer and Robot Vision (CRV)*, pages 456–463, 2016.

[52] F Landis Markley. Multiplicative vs. additive filtering for spacecraft attitude determination. *Dynamics and Control of Systems and Structures in Space*, (467-474), 2004.

[53] Peter S Maybeck. *Stochastic models, estimation, and control*, volume 1. Academic press, 1982.

[54] Philip F McLauchlan and David W Murray. Active camera calibration for a head-eye platform using the variable state-dimension filter. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 18(1):15–22, 1996.

[55] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3565–3572, 2007.

[56] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T Furgale, and Roland Siegwart. A synchronized visual-inertial sensor system with FPGA pre-processing for accurate real-time SLAM. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 431–437, 2014.

[57] Edwin Olson. Apriltag: A robust and flexible visual fiducial system. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407, 2011.

[58] Mrinal K Paul, Kejian Wu, Joel A Hesch, Esha D Nerurkar, and Stergios I Roumeliotis. A comparative analysis of tightly-coupled monocular, binocular, and stereo VINS. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 165–172, 2017.

[59] Jason Rebello, Arun Das, and Steven L Waslander. Autonomous active calibration of a dynamic camera cluster using next-best-view. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[60] Francisco J. Romero-Ramirez, Rafael Muoz-Salinas, and Rafael Medina-Carnicer. Speeded up detection of squared fiducial markers. *Image and Vision Computing*, 76:38 – 47, 2018.

[61] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision (ECCV)*, pages 430–443. Springer, 2006.

[62] Stergios I Roumeliotis, Andrew E Johnson, and James F Montgomery. Augmenting inertial navigation with image-based motion estimation. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4326–4333, 2002.

[63] Stergios I Roumeliotis, Gaurav S Sukhatme, and George A Bekey. Circumventing dynamic modeling: Evaluation of the error-state kalman filter applied to mobile robot localization. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1656–1663, 1999.

[64] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. ORB: An efficient alternative to sift or surf. In *IEEE International Conference on Computer Vision (ICCV)*, pages 2564–2571, 2011.

[65] MA Shelley. Monocular visual inertial odometry on a mobile device. *Master's thesis, Institut für Informatik, TU München, Germany*, 2014.

[66] Shaojie Shen, Yash Mulgaonkar, Nathan Michael, and Vijay Kumar. Multi-sensor fusion for robust autonomous flight in indoor and outdoor environments with a rotorcraft MAV. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 4974–4981, 2014.

[67] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. *Journal of Field Robotics*, 27(5):587–608, 2010.

[68] Joan Sola. Quaternion kinematics for the error-state kalman filter. *arXiv preprint arXiv:1711.02508*, 2017.

[69] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters (RA-L)*, 3(2):965–972, 2018.

[70] Chun Kiat Tan, Jianliang Wang, Yew Chai Paw, and Fang Liao. Autonomous ship deck landing of a quadrotor using invariant ellipsoid method. *IEEE Transactions on Aerospace and Electronic Systems*, 52(2):891–903, 2016.

[71] Michael J Tribou, Adam Harmat, David WL Wang, Inna Sharf, and Steven L Waslander. Multi-camera parallel tracking and mapping with non-overlapping fields of view. *The International Journal of Robotics Research*, 34(12):1480–1500, 2015.

[72] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustmenta modern synthesis. In *International Workshop on Vision Algorithms*, pages 298–372. Springer, 1999.

[73] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. Direct visual-inertial odometry with stereo cameras. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 1885–1892, 2016.

[74] Rudolph Van Der Merwe and Eric A Wan. The square-root unscented kalman filter for state and parameter-estimation. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 3461–3464. IEEE, 2001.

[75] H. Voos and H. Bou-Ammar. Nonlinear tracking and landing controller for quadrotor aerial robots. In *IEEE International Conference on Control Applications (CCA)*, pages 2136–2141, Sept 2010.

[76] Rui Wang, Martin Schwörer, and Daniel Cremers. Stereo DSO: Large-scale direct sparse visual odometry with stereo cameras. In *International Conference on Computer Vision (ICCV)*, 2017.

[77] 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 *IEEE International Conference on Robotics and Automation (ICRA)*, pages 957–964, 2012.

[78] Karl Engelbert Wenzel, Andreas Masselli, and Andreas Zell. Automatic take off, tracking and landing of a miniature UAV on a moving carrier vehicle. *Journal of Intelligent and Robotic Systems*, 61(1-4):221–238, 2011.

[79] Shuo Yang, Jiahang Ying, Yang Lu, and Zexiang Li. Precise quadrotor autonomous landing with SRUKF vision perception. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2196–2201, 2015.