REDUCING COMPUTATIONAL DEMAND OF MULTI-STATE CONSTRAINT KALMAN FILTER IN VISUAL-INERTIAL ODOMETRY APPLICATIONS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

KEREM EYICE

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

SEPTEMBER 2019

Approval of the thesis:

REDUCING COMPUTATIONAL DEMAND OF MULTI-STATE CONSTRAINT KALMAN FILTER IN VISUAL-INERTIAL ODOMETRY APPLICATIONS

submitted by **KEREM EYICE** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Prof. Dr. Tolga Çiloğlu
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Kemal Leblebicioğlu
Electrical and Electronics Engineering, METU _____

Prof. Dr. Tolga Çiloğlu
Electrical and Electronics Engineering, METU _____

Prof. Dr. Umut Orguner
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Mustafa Mert Ankaralı
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Yakup Özkazanç
Electrical and Electronics Engineering, Hacettepe University _____

Date: 02.09.2019

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Kerem Eyice

Signature :

ABSTRACT

REDUCING COMPUTATIONAL DEMAND OF MULTI-STATE CONSTRAINT KALMAN FILTER IN VISUAL-INERTIAL ODOMETRY APPLICATIONS

Eyice, Kerem
M.S., Department of Electrical and Electronics Engineering
Supervisor: Prof. Dr. Tolga Çiloğlu

September 2019, 88 pages

The aim of this study is to reduce the computational load required by Multi-State Constraint Kalman Filter in visual-inertial odometry applications while maintaining the accuracy of the localization solution. In order to accomplish this, a keyframe-based pose selection mechanism is proposed. The proposed method fuses visual measurements with inertial measurements in order to estimate the kinematics of the platform. The contribution of this study is to reduce computational demand of the filtering operations by using visual measurements obtained only from selected keyframes for navigation purposes. Experiments have been performed with visual-inertial datasets collected at different speeds of an aerial platform in order to assess the performance of the proposed method. Results showed that the proposed method can attain accurate navigation solution while decreasing the required computational load with a proper keyframe selection criteria.

Keywords: Visual-Inertial Odometry, Multi-State Constraint Kalman Filter, Vision-

Aided Inertial Navigation, Autonomous Navigation

ÖZ

GÖRSEL-ATALETSEL KONUMLAMA UYGULAMALARINDA ÇOK DURUM KISITLI KALMAN SÜZGECİNİN İŞLEM YÜKÜNÜN AZALTILMASI

Eyice, Kerem
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Prof. Dr. Tolga Çiloğlu

Eylül 2019 , 88 sayfa

Bu çalışmanın amacı; Çok Durum Kısıtlı Kalman Süzgeci tabanlı Görsel-Ataletsel Konumlama uygulamalarında konumlama çözümünün doğruluğunu korurken, gerekli olan işlem yükünü azaltmaktır. Bunu başarmak için, anahtar kareye dayalı bir poz seçim mekanizması önerilmiştir. Önerilen yöntem platform kinematiğinin kestirimi için, görsel ölçümler ile ataletsel ölçümleri birleştirir. Bu çalışmanın katkısı, sadece seçilen anahtar karelerden elde edilen görsel ölçümleri seyrüsefer amacıyla kullanarak süzgeç işlemlerinin işlem yükünü azaltmaktır. Önerilen yöntemin performansını değerlendirmek için bir hava platformunun farklı hızlarında toplanan görsel-ataletsel veri setleri ile deneyler yapılmıştır. Sonuçlar, önerilen yöntemin doğru hesaplama çözümünü elde etmenin yanı sıra, gerekli işlem yükünü uygun bir anahtar kare seçim ölçütüyle düşürdüğünü göstermiştir.

Anahtar Kelimeler: Görsel-Ataletsel Konumlama, Çok Durum Kısıtlı Kalman Süz-

geci, Görsel Destekli Ataletsel Seyrüsefer, Otonom Seyrüsefer

To my family

ACKNOWLEDGMENTS

First of all, I would like to express my deepest gratitude to my supervisor Prof. Dr. Tolga Çiloğlu for his guidance, support, and advice. I am grateful to him for letting me to carry out this study under his supervision with his encouragement and positive attitude.

I would like to extend my thanks to all the lecturers in the Department of Electrical and Electronics Engineering, who greatly helped me to store the knowledge I need to carry out this study.

I would like to thank TÜBİTAK-SAGE. In addition, I would like to thank my friends and colleagues, especially Haluk Erdem Bingöl, Görkem Kandemir, Gökhan Gökdoğan, Ali Can Arık, Ahmet Çakıroğlu, and Semih Köklücan who contributed to this study with their continuous encouragement.

I would like to thank my dear friends, Mehmet Çetinkaya, Efe Balo, and Onur Gülsem for their intimate friendship and all the precious times we spent together.

I would like to present my sincere thanks to my mom, dad, and sister for their unconditional love and support under all circumstances. Throughout my whole life, they showed unlimited patience and had always faith in me.

Last but not least, I would like to express my deepest thanks to Deniz Kaya for her support and unconditional love. Without her existence, many beautiful things in my life would not have happened.

TABLE OF CONTENTS

LIST OF TABLES

TABLES

LIST OF FIGURES

FIGURES

LIST OF ABBREVIATIONS

DCM	Direction Cosine Matrix
EKF	Extended Kalman Filter
FAST	Features from accelerated segment test
IMU	Inertial Measurement Unit
INS	Inertial Navigation System
KF MSCKF	Keyframe-Based Multi-State Constraint Kalman Filter
KLT	Kanade-Lucas-Tomasi
MSCKF	Multi-State Constraint Kalman Filter
SIFT	Scale-Invariant Feature Transform
SLAM	Simultaneous Localization and Mapping
SURF	Speeded-Up Robust Features
VIO	Visual Inertial Odometry
VO	Visual Odometry

CHAPTER 1

INTRODUCTION

1.1 Background and Motivation

Navigation, in addition to planning and following a route, is the task to determine the kinematics of the platform (position, velocity, and attitude) by sensor measurements. A navigation system consists of multiple sensors measuring the motion of the platform and/or information about the environment of the platform and a computer for processing the raw output of the sensors in order to estimate the platform's kinematics. Autonomous navigation concept has drawn intense interest in recent years due to its significance in unmanned aerial vehicles [1, 2, 3], unmanned land vehicles [4, 5] and space exploration applications [6, 7].

Autonomous navigation solution can be easily obtained where Global Navigation Satellite Systems (GNSS) signals are available and reliable [8, 9]. On the other hand, there are GNSS-denied environments such as urban canyons, forests, underground, indoors; where the quality of satellite signals degrades sharply. In addition to that, jamming and spoofing are human-made threats to the GNSS-based navigation systems which are more likely to occur than they were in the past. In these environments, obtaining accurate navigation solution becomes more difficult.

However, a wide range of GNSS-denied navigation systems have been developed for indoor environments with infrastructure [10, 11, 12]; navigation problem in unstructured and unknown environments remains challenging. Moreover; size, cost, weight, and power consumption strictly limit sensor choice in micro aerial vehicles, mobile phones, augmented reality and virtual reality devices and other applications.

Choosing appropriate sensors to design a navigation system for GNSS-denied environments is a crucial task. Available sensors for this task can be divided into two main types: proprioceptive sensors and exteroceptive sensors. The former type of sensors can provide information about the motion of the platform, while the latter type of sensors can gather information about the environment. Examples for proprioceptive sensors are wheel encoders measuring linear and rotational velocity of wheels [13, 14], Inertial Measurement Units (IMU) measuring acceleration and rotation rates of the platform [15, 16]. Examples for exteroceptive sensors include cameras [17, 18, 19], laser sensors [20, 21, 22], acoustic sensors [23, 24] and magnetometers [25, 26] which capture information about the surroundings.

Among these options, combining an IMU with a single camera offers many benefits in order to obtain navigation solution for small, lightweight, and inexpensive platforms. Firstly, both sensors are capable of gathering 3D motion information with no infrastructure requirement. Hence, using the combination of these sensors ensures operation in unknown environments. In addition to that, these sensors have complementary characteristics since the camera is exteroceptive while the IMU is a proprioceptive sensor. Thus, fusion of IMU and camera measurements can provide more robust and accurate localization solution.

Techniques that combine IMU and camera measurements for navigation purposes are termed as vision-aided inertial navigation. Images are used to extract and track distinctive features in the environment such as sparse points, edges, lines or planes. Then feature measurements are fused with IMU measurements to estimate kinematics of the platform. Vision-aided inertial navigation methods can be grouped into two main categories based on whether the navigation system has mapping ability and focuses on global optimization of the map and the whole path of the platform, or only aims to estimate kinematics of the platform. Methods that have extra mapping thread are termed as Simultaneous Localization and Mapping (SLAM) and techniques that are implemented for only localization purposes are called as Visual Odometry (VO) and Visual Inertial Odometry (VIO) [27].

Among VIO methods, A Multi-State Constraint Kalman Filter (MSCKF) technique [19] has drawn intense interest to fuse IMU and camera measurements; due to its

accuracy and consistency. MSCKF uses an error-state Kalman Filter. An error-state Kalman filter includes errors of the kinematics that are going to be observed in the state vector. Thus, an indirect correction mechanism is applied to the kinematics. IMU measurements are used to propagate the kinematics of the platform and the covariance matrix of the states according to the dynamic motion model in this technique. On the other hand, visual measurements are used to estimate the errors of the kinematics and update the covariance matrix of the states. Subsequently, estimated errors are subtracted from the propagated kinematics of the platform.

Propagation and update threads are processed at different frequencies since propagation happens at IMU measurement frequency and updates from camera measurements are obtained at camera measurement frequency. Camera measurements are the pixel positions of features extracted and tracked in consecutive frames. As the features are tracked in image sequences, pixel position measurements provide information about position and attitude change of the camera/platform between consecutive frames. Distribution of the features in the images influences observability issues of the filter states. Homogeneously distributed features in the images improve uniqueness of the information extracted from images.

In [28], MSCKF method was compared to five different recent VIO algorithms in terms of computational demand and accuracy on four different hardware platforms. These hardware platforms are a Laptop, a small desktop PC, and two embedded computers, UP Board and ODROID. Although MSCKF is the oldest algorithm, it is seen that the algorithm has close performance on accuracy with the other methods in all of the platforms. In addition to that, the algorithm was able to complete all VIO datasets used on all platforms, showing robust performance. Lastly; despite MSCKF being the oldest algorithm, it showed better performance on computational resource requirements when compared to the other algorithms.

In [29], the authors used a modified version of the MSCKF algorithm on the processor of a mobile phone (Samsung Galaxy S2). In doing so, they tested the algorithm to perform kinematics estimation when operating on a resource-constrained system. Results of the experiments showed that the MSCKF algorithm is able to obtain accurate localization solution on this platform in real-time operation.

These studies showed that MSCKF algorithm is still a valid and viable solution to fuse visual and inertial measurements on platforms having limited resources.

1.2 Contribution

MSCKF algorithm is an EKF based VIO formulation where the state of the filter is formed by two main parts: a static and a dynamic part. The static part comprises of errors of the current position, velocity, and attitude of the platform in addition to errors of biases of the accelerometers and gyroscopes. On the other hand, the dynamic part includes errors of past platform poses (A pose is the position and the attitude of the platform). States that are included in the dynamic part are the errors of poses at instants of image capture. The number of included platform pose states in the dynamic part of the state vector changes in time. In addition to that, contents of the stored poses change during the operation of the platform.

As the filter starts operation, errors of the platform poses (pose states) at the instants of image capture are appended to the dynamic part of the overall state vector. Measurements of features obtained from images are also registered to the memory.

At each image capture, filter update conditions are checked whether some of the previously observed features have to be used in filter update step or not. The first condition checked is the loss of a feature track. Previously registered measurements of a feature are used in filter update phase if the track of the feature is lost. The other update condition is that the filter reaches the maximum number of pose states allowed. In this case, at least one of the older platform pose state have to be removed from the filter state vector. However, when removing the pose state from the filter state vector, measurements of the features observed at the time that pose was recorded should be used in the filter update phase.

After the filter update phase is completed, all feature measurements used in filter update are removed from measurement storage. Subsequently, remaining feature measurements of each pose state are checked. If a pose state has no remaining feature measurement, that pose is removed from the filter's state vector.

In Figure 1.1 flow chart showing management of the state vector and feature measurements is presented.

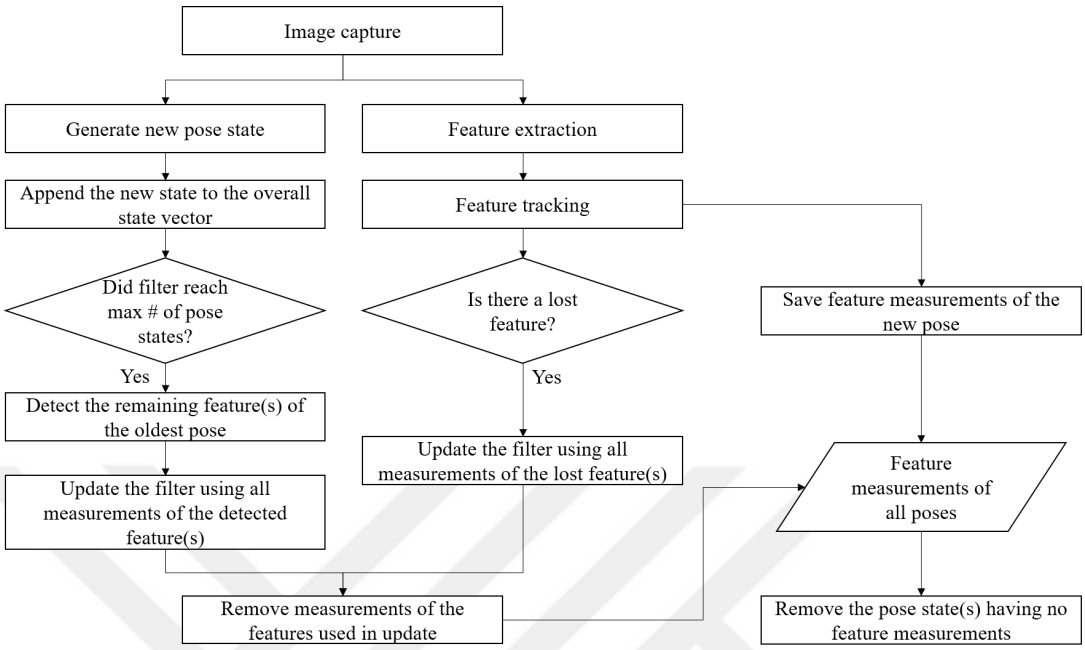

Figure 1.1: Flow chart of management of states and feature measurements.

Limit for the maximum number of pose states stored in the filter determines the size of the sliding window. This size has effects on both the accuracy and required computational resources. Errors on the estimated states are reduced as the size of the window increases. On the other hand, computational resources required for filtering operation increases if the size of the window is extended since the number of included elements in the state vector increases.

In this work, it is aimed to keep the accuracy of the localization solution while reducing the required computational resources. To this end, a keyframe-based pose selection mechanism is proposed to extend the sliding window while storing fewer elements in the state vector. The term keyframe is used in visual odometry literature to refer to a frame in which the scene observed by the camera changes more than a threshold compared to the last keyframe [30, 31, 32, 33]. A platform pose state is included in the overall state vector if the frame is a keyframe. The captured image is used for only feature tracking purposes if it is not a keyframe.

1.3 Outline of the Thesis

The outline of this thesis is as follows.

A literature survey about the studies on SLAM, VO and VIO is presented in Chapter 2. Firstly, a taxonomy of vision aided navigation systems is shared regarding various characteristics of such systems. Then, different visual inertial odometry techniques are discussed.

Chapter 3 focuses on required mathematical background for constructing MSCKF based VIO algorithm. This chapter includes underlying mechanisms of both the process and the measurement of MSCKF based VIO. Finally, details of the MSCKF algorithm are presented in this chapter.

In Chapter 4, methodology of proposed keyframe-based MSCKF for VIO algorithm is provided. After evaluating the effect of the size of the sliding window on performance of standard MSCKF with real world data, keyframe-based MSCKF for VIO is presented.

In Chapter 5, performance of the proposed keyframe-based MSCKF for VIO method is investigated. Experimental results of the proposed and standard MSCKF methods are compared.

Concluding remarks of this thesis are presented in Chapter 6.

CHAPTER 2

LITERATURE SURVEY

2.1 Categorization of Simultaneous Localization and Mapping and Visual Odometry Methods

2.1.1 Simultaneous Localization and Mapping, Visual Odometry

SLAM and VO methods have common properties in the sense that both approaches aim to estimate platform kinematics and reconstruct the structure of the scene. On the other hand, there are specific differences between these two methods regarding the size of the estimation window and the task of "loop closure".

SLAM comprises estimation of the full path of the platform and building a global map of the environment concurrently. The global map is a description of the surroundings which consists of 3D positions of all the features observed. Constructing a map is useful not only for path planning and visualization purposes but also for limiting the accumulated error in the estimation of the platform's states.

Loop closure is the term used for detection of previously observed areas upon revisits. The position errors of mapped features get higher due to the drift in platform's states estimated by SLAM systems. Loop closure can be beneficial after operating some time. Accumulated errors on the path of the platform can be reduced if the SLAM system detects a loop closure while the platform is revisiting previously observed parts of the map [27]. On the other hand, performance of loop closure depends highly on the operational type of the platform. Error removal can be performed effectively for a platform wandering in the same places or operating in small surroundings such as indoors.

On the contrary to SLAM methods, VO systems do not seek to construct a global map. These approaches aim to establish locally consistent trajectory. VO systems estimate the platform's pose in each step sequentially, and feature observations are used solely in estimation process of the platform's trajectory. Since a global map is not constructed in VO systems, there is no chance to detect loop closure and remove accumulated error with time. Although SLAM systems provide more accurate localization solution, VO methods are less complicated and are computationally cheaper. Storing the full map and estimating the whole path of the platform make SLAM systems challenging to operate in real-time applications.

2.1.2 Monocular and Stereo Vision

Number of cameras used for SLAM/VO systems is one of the distinctive characteristics of SLAM and VO systems. Stereo cameras yield an important advantage over monocular ones since scale ambiguity in the scene can be removed easily. The range between the stereo cameras and the feature can be calculated by using observations from two cameras. Hence, the 3D position of the feature can be predicted directly for further use in stereo vision-aided navigation systems. There are several examples using stereo cameras for both SLAM [34, 35] and VO [36, 37].

On the other hand, the accuracy of the depth estimate from stereo camera measurements depends strongly on the ratio of the baseline of the cameras and the scale of the scene. The baseline of the cameras has to be large enough to obtain a precise depth estimate in large scale environments. Increasing the baseline may not be possible in applications having size requirements. In such environment and application configurations, stereo cameras have less benefit over the use of monocular methods. Consequently, several monocular vision-aided navigation methods have been proposed for use in both SLAM [38, 39, 40] and VO [41, 42] systems.

2.1.3 The Amount of Pixels Used for Motion Estimation (Sparse, Semi-Dense, and Dense)

The sparsity of the used points on the images is another different characteristic of SLAM & VO systems. Most work done on vision-based navigation systems is based on using sparse features which are detected with feature extraction algorithms and tracked in consecutive images.

Dense methods use every point in the images in order to obtain better motion estimation accuracy. There are examples of dense methods in VO [43] and SLAM [44] based on RGB-D images. On the other hand, most dense SLAM works use monocular RGB cameras while estimating the depth map of the scene in temporal images [45, 46, 47]. The main drawback of dense methods is high computational cost since these methods use every pixel on the images for localization.

Semi-dense approaches use a group of high-gradient pixels in the image such as the whole pixels of edges or corners. Generally, the number of used points are more than sparse methods, but fewer than dense methods. In [48], authors proposed a SLAM technique which uses sparse high gradient features for trajectory estimation while reconstructing a semi-dense map of the environment. Monocular VO methods are proposed in [49, 50], which construct semi-dense depth map for all points in the image having non-negligible image gradient. Then, propagated information of these pixels is used to estimate motion with new image measurements.

2.1.4 Interpretation of Images: The Type of Quantitative Information

There are mainly three different ways to infer the motion of the platform from image sequences. The most commonly used method is called as "feature-based" or "indirect". Feature-based systems estimate motion by using re-projection errors of features in the images.

In Figure 2.1, indirect motion estimation procedure is briefly described. $\boldsymbol{T}_{k,k-1}$ represents the motion of the platform between two consecutive steps which includes both 3×3 rotation matrix between consecutive steps, $\boldsymbol{R}_{k,k-1}$, and 3×1 translation vector,

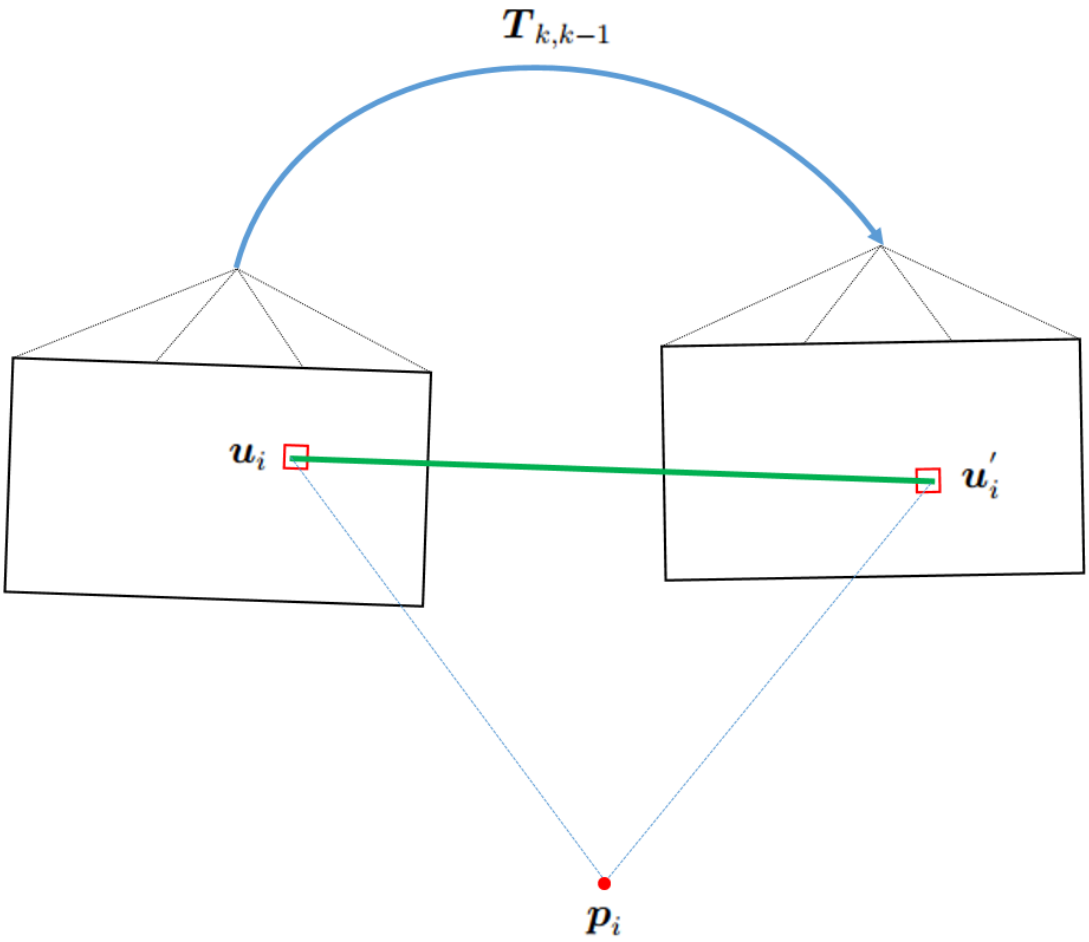

Figure 2.1: Visualization of the motion estimation in feature-based methods.

$t_{k,k-1}$.

$$T_{k,k-1} = \begin{bmatrix} R_{k,k-1} & t_{k,k-1} \end{bmatrix} \tag{2.1}$$

u_i and u_i' are the pixels of i^{th} feature in the $(k-1)^{th}$ and the k^{th} image, respectively. Lastly, p_i represents the estimated position of the feature at the reference coordinate system. Motion inferring mechanism of feature-based SLAM/VO systems is presented in Equation 2.4 where $\pi(\cdot)$ is the projection model of a feature to the image

plane.

$$\hat{\boldsymbol{u}}_i = \pi(\boldsymbol{p}_i) = \begin{bmatrix} -f\frac{M}{H} & 0 & c_x \\ 0 & f\frac{N}{W} & c_y \end{bmatrix} (\frac{1}{d_i}\boldsymbol{C}_n^c(\boldsymbol{p}_i - \boldsymbol{p}_c)) \tag{2.2}$$

In Equation 2.2, $\hat{\boldsymbol{u}}_i$ represents the projected pixel position of the feature. H, M, W, N, f, c_x, and c_y are the camera intrinsic parameters defined in Section 3.4. d_i represents a measurement or an estimation about the depth of the feature. $\boldsymbol{C}_n^c$ and $\boldsymbol{p}_c$ represent direction cosine matrix from the reference frame to the camera frame, and the position of the camera at the reference frame, respectively. The position of a feature can be projected to the image plane using the relation shared in Equation 2.2.

The position of the feature can be estimated by using its observation in the former image, $\boldsymbol{u}_i$, and a measurement or an estimation about the depth of the feature, d_i.

$$\boldsymbol{p}_i = \pi^{-1}\left(\begin{bmatrix} \boldsymbol{u}_i \\ 1 \end{bmatrix}\right) \cdot d_i \tag{2.3}$$

Subsequently, predicted observation of the same feature in the latter image can be calculated by projecting the estimated position of the feature to the image plane of the k^{th} frame. Difference between the projected pixel position of the feature in the latter image and the measurement about the pixel position of the feature is minimized in order to estimate the motion of the platform between consecutive steps.

$$\boldsymbol{T}_{k,k-1} = \min_{\boldsymbol{T}} \sum_i \text{Cost}\left(\boldsymbol{u}_i^{'} - \pi\left(\boldsymbol{T} \cdot \begin{bmatrix} \boldsymbol{p}_i \\ 1 \end{bmatrix}\right)\right) \tag{2.4}$$

Fundamental examples of feature based SLAM/VO are studied in [40, 38, 51].

In contrast, direct methods infer motion and structure of the scene from intensity values in the image. The estimation process is done by minimizing the cost of the so called photometric error which is the difference in the pixel intensities.

In Figure 2.2 and Equation 2.6, motion estimation in direct methods is represented. The feature observed in the $(k-1)^{th}$ frame is projected to the k^{th} frame. $u_i^{'}$ is the pre-

dicted pixel position of the feature in the k^{th} frame, which is projected using the pixel position of the feature in the $(k-1)^{th}$ frame and a measurement or an estimation about the depth of the point d_i.

$$\boldsymbol{u}_i^{'} = \pi \left(\boldsymbol{T} \cdot \begin{bmatrix} \pi^{-1} \left(\begin{bmatrix} \boldsymbol{u}_i \\ 1 \end{bmatrix} \right) \cdot d_i \\ 1 \end{bmatrix} \right) \tag{2.5}$$

Minimized cost is the difference between the intensity value of the point in the $(k-1)^{th}$ frame, $I_{k-1}(\boldsymbol{u}_i)$, and the intensity value of the projected pixel of the same point in the k^{th} frame, $I_k(\boldsymbol{u}_i^{'})$. There are several examples of direct methods that are using above procedure for motion estimation and structure reconstruction [52, 51, 47, 45].

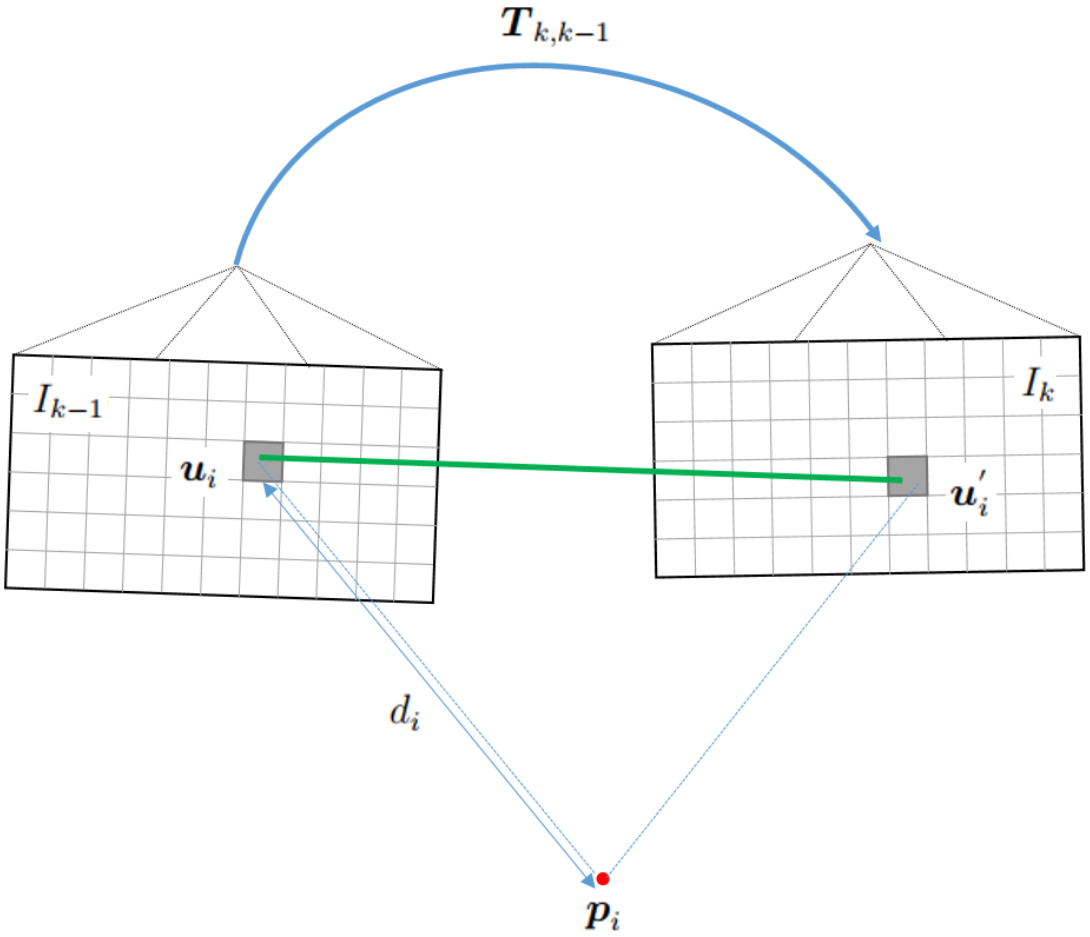

Figure 2.2: Visualization of the motion estimation in direct methods.

$$\boldsymbol{T}_{k,k-1} = \min_{\boldsymbol{T}} \sum_{i} \text{Cost}(I_k(\boldsymbol{u}_i^{'}) - I_{k-1}(\boldsymbol{u}_i)) \tag{2.6}$$

In [41], a monocular semi-direct VO approach is presented. In this method, image alignment procedure is executed in order to minimize the photometric difference between image patches corresponding to the same points when a new image is captured. In the last step, motion estimation is obtained by minimizing the reprojection error that has been established in the previous alignment step.

2.2 Visual Inertial Odometry

Complementary characteristics of inertial and visual measurements make combining these two sensor modalities an ideal choice. Thus, visual-inertial fusion based odometry techniques for state estimation problem have gained large popularity over the past decades.

Fusion techniques of these sensor modalities can be categorized in terms of the visual measurements used in the integration and integration method.

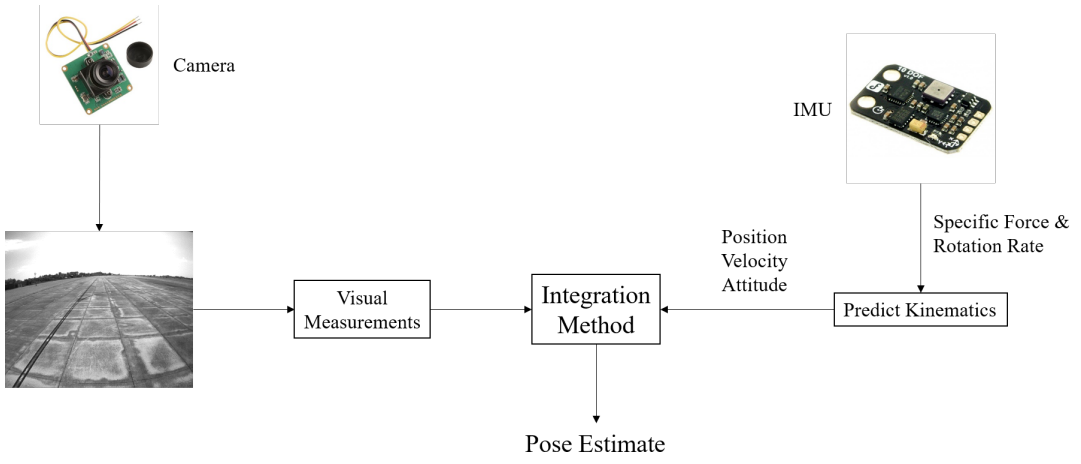

Figure 2.3: Visual Inertial Odometry flow diagram

2.2.1 Information Integration Level

VIO methods can be divided into two main types in terms of the integration level: loosely-coupled and tightly-coupled integration.

In loosely-coupled techniques, visual module is treated as a black box providing pose estimates similar to visual odometry. Subsequently, visual pose estimates are fused with predicted kinematics using IMU measurements. There are several loosely-coupled visual-inertial integration techniques [53, 54, 55, 56, 57].

In contrast, tightly-coupled methods estimate the pose by making use of raw camera measurements as visual measurements in Figure 2.3. The raw measurements used in visual-inertial integration can be the intensity values [58, 59, 60] or the feature pixel positions [61, 29, 19, 62, 33, 63] as mentioned in Section 2.1.4. Studies, which compare these two methods, have presented that loosely-coupled techniques are usually less computationally expensive, however tightly-coupled methods outperform in terms of the estimation accuracy [33].

2.2.2 Information Integration Method

VIO methods can be divided into two categories in terms of the used integration method in Figure 2.3: filtering-based and optimization based.

Filtering based methods mostly use Extended Kalman Filter (EKF) to fuse inertial and visual measurements. Filter states and covariances are propagated using inertial measurements. Filter updates are performed when visual measurements are obtained. There are several loosely-coupled filtering based methods [64, 54, 55, 57] as well as tightly-coupled approaches [19, 59, 65].

On the other hand, using nonlinear optimization as integration method results in more accurate pose estimation of the platform. However, computational demand of optimization-based methods is much higher than the filtering-based techniques. Examples of tightly-coupled optimization based VIO methods are [33, 66, 63, 32, 67]

2.2.3 Recent State of the Art Visual Inertial Odometry Techniques

Many VIO methods have been proposed in recent years in order to be used in real time navigation of small devices.

In ROVIO [59], authors proposed a sparse, filtering based, direct VIO method to be used in monocular systems. In this work, point features are parametrized by the inverse depth and bearing vector in the state of the EKF which are propagated according to the corresponding dynamics as new frames registered. Multilevel patch features of 8x8 pixels are tracked in the image sequence, and photometric errors of the patches are used directly as innovation in the EKF update state. In order to decrease the high dimensional error term and to reduce the computational cost of the filter operations, a QR-decomposition is employed on the stacked error vector.

A sparse feature based optimization method for both monocular and stereo configurations has been proposed as OKVIS [33]. Authors formulated a probabilistic cost function which fuses reprojection errors of features and IMU error terms. Since the optimization based methods require more computational resources, they limit the optimization to a set of keyframes. Keyframes are frames that are arbitrarily spaced in time which are selected according to the feature content. If the hull of the projected and matched landmarks covers less than 50 % of the image or if matched over detected keypoints is less than 20 %, the new frame is registered as a keyframe. All features that are visible in the keyframes are used in the estimation process.

Another direct VIO approach is proposed in [60], which uses stereo cameras in an optimization-based estimation. The method uses semi-dense depth maps instead of sparse points in the images. The depth map is obtained via not only using stereo configuration but also related images of the same camera taken at different poses. Authors introduced an overall energy cost comprises of image and IMU errors which is subsequently minimized via nonlinear optimization.

In [63], authors proposed a monocular optimization based VIO method which uses sparse features as the image measurements. The main contribution of this paper is preintegrating IMU measurements between keyframes in order to reduce the number of variables used in optimization. Hence, this optimization based method becomes

feasible in real-time operation. Then, the proposed approach minimizes negative log posterior probability of states given both visual and preintegrated inertial measurements.

A sparse feature based stereo MSCKF is proposed in [68]. In this method, firstly features are matched in stereo image pairs then tracked in subsequent frames. Measurements of the same feature in both cameras are used in Kalman Update step of MSCKF algorithm.

A deep-learning based approach for VIO is proposed in [69]. In this paper, the authors presented the first fully end-to-end trainable visual inertial navigation technique. The model is based on a Convolutional Neural Network - Recurrent Neural Network structure to process monocular images and IMU measurements for egomotion estimation.

CHAPTER 3

THEORETICAL BACKGROUND

In this study, a visual-inertial odometry method based on the paper [19], a Multi-State Constraint Kalman Filter is used. This method is a monocular, sparse, feature-based, tightly-coupled, and filtering-based visual-inertial odometry technique.

This chapter provides the required theoretical background to follow the problem of inertial and visual sensor fusion. The chapter begins by the definition of coordinate frames and transformations between these coordinate frames. Then, motion estimation of a platform via inertial measurements is discussed. Subsequently, the geometric relationship between the image and the scene is discussed. Next, a review of linear and nonlinear Kalman filtering is presented. Lastly, details of the MSCKF algorithm is provided.

3.1 Coordinate Frames

The concept of coordinate frames is essential for navigation applications since platform kinematics are represented using coordinate frames. In this section, the coordinate frames that are used in this study are defined.

- **Local cartesian frame (n-frame)**: Local cartesian frame is the reference frame used in this study which has a fixed origin with respect to Earth. The origin of local cartesian frame is the point on Earth where the platform starts operation. Frame's x, y and z axes point the north, east and down directions relative to the origin of the coordinate frame. In this study, it is aimed to estimate the platform's position and velocity relative to the origin of local cartesian frame

and platform's attitude relative to the axes of local cartesian frame.

- **Body frame (b-frame)**: Body frame is rigidly attached to the platform, and x, y, and z axes point the nose, right, and bottom of the platform, respectively. In most applications, axes of the body frame are coincident with the measuring axes of the IMU.

- **Camera frame (c-frame)**: Camera frame is rigidly attached to the camera, which its origin is at the optical center. Frame's x and y axes point up and to the right and are parallel to the image plane. z axis is perpendicular to the image plane and points out of the camera.

3.2 Coordinate Transformations

Coordinate Transformations describe the orientation of the axes of one coordinate frame with respect to the axes of another. This description allows one to transform a vector resolved in one coordinate frame to the another frame. In this section, three methods for Coordinate Transformations are presented: Euler Angles, Direction Cosine Matrices (DCM) and Quaternions.

- **Euler Angles**: Three angles corresponding to a specific single-axis rotation sequence are called as Euler angles. These angles are called as roll (ϕ), pitch (θ) and yaw (ψ).

- **Direction Cosine Matrices (DCM)**: Direction Cosine Matrices are 3x3 matrices which can be denoted as $\boldsymbol{C}_{\beta}^{\alpha}$ to represent the DCM from β-frame to α-frame. For example, transformation of a vector resolved in β-frame to α-frame can be done by multiplying the vector by DCM.

$$\boldsymbol{p}^{\alpha} = \boldsymbol{C}_{\beta}^{\alpha}\boldsymbol{p}^{\beta} \tag{3.1}$$

where $\boldsymbol{p}^{\alpha}$ is the vector resolved in α-frame, $\boldsymbol{p}^{\beta}$ is the vector resolved in β-frame. A DCM can be converted to and from euler angles using:

$$\boldsymbol{C}_{\beta}^{\alpha} = \begin{bmatrix} cos(\theta)cos(\psi) & cos(\theta)sin(\psi) & -sin(\theta) \\ (-cos(\phi)sin(\psi) + sin(\phi)sin(\theta)cos(\psi)) & (cos(\phi)cos(\psi) + sin(\phi)sin(\theta)sin(\psi)) & sin(\phi)cos(\theta) \\ (sin(\phi)sin(\psi) + cos(\phi)sin(\theta)cos(\psi)) & (-sin(\phi)cos(\psi) + cos(\phi)sin(\theta)sin(\psi)) & cos(\phi)cos(\theta) \end{bmatrix} \tag{3.2}$$

$$\begin{aligned}\phi &= arctan(\frac{C^{\alpha}_{\beta 2,3}}{C^{\alpha}_{\beta 3,3}}) \\ \theta &= -arcsin(C^{\alpha}_{\beta 1,3}) \\ \psi &= arctan(\frac{C^{\alpha}_{\beta 1,2}}{C^{\alpha}_{\beta 1,1}})\end{aligned} \tag{3.3}$$

- **Quaternions**: A coordinate transformation can be represented using a quaternion which is a hyper-complex number with four components, [70]:

$$\boldsymbol{q}^{\alpha}_{\beta} = (q^{\alpha}_{\beta 0}, q^{\alpha}_{\beta 1}, q^{\alpha}_{\beta 2}, q^{\alpha}_{\beta 3}) \tag{3.4}$$

The first element of a quaternion, $q^{\alpha}_{\beta 0}$, is the "scalar part" that is a function of the magnitude of the rotation. The other three components $q^{\alpha}_{\beta 1}, q^{\alpha}_{\beta 2}, q^{\alpha}_{\beta 3}$ are the "vector part" of the quaternion which are functions of both the magnitude and the axis of rotation. The vector part can be thought of as the vector around which rotation is performed. The elements of a quaternion can be represented with respect to the magnitude of the rotation, $\mu_{\beta\alpha}$, and the unit vector representing the axes of the rotation, $\boldsymbol{u}_{\beta\alpha}$ as:

$$\begin{aligned}q^{\alpha}_{\beta 0} &= cos\left(\frac{\mu_{\beta\alpha}}{2}\right) \\ q^{\alpha}_{\beta 1} &= u_{\beta\alpha 1} sin\left(\frac{\mu_{\beta\alpha}}{2}\right) \\ q^{\alpha}_{\beta 2} &= u_{\beta\alpha 2} sin\left(\frac{\mu_{\beta\alpha}}{2}\right) \\ q^{\alpha}_{\beta 3} &= u_{\beta\alpha 3} sin\left(\frac{\mu_{\beta\alpha}}{2}\right)\end{aligned} \tag{3.5}$$

In Equation 3.5, $u_{\beta\alpha 1}$, $u_{\beta\alpha 2}$, and $u_{\beta\alpha 3}$ represent components of the unit vector.

A quaternion can be converted to the corresponding DCM using:

$$\boldsymbol{C}^{\alpha}_{\beta} = \begin{bmatrix} {q^{\alpha}_{\beta 0}}^2 + {q^{\alpha}_{\beta 1}}^2 - {q^{\alpha}_{\beta 2}}^2 - {q^{\alpha}_{\beta 3}}^2 & 2(q^{\alpha}_{\beta 1}q^{\alpha}_{\beta 2} + q^{\alpha}_{\beta 0}q^{\alpha}_{\beta 3}) & 2(q^{\alpha}_{\beta 1}q^{\alpha}_{\beta 3} - q^{\alpha}_{\beta 0}q^{\alpha}_{\beta 2}) \\ 2(q^{\alpha}_{\beta 1}q^{\alpha}_{\beta 2} - q^{\alpha}_{\beta 0}q^{\alpha}_{\beta 3}) & {q^{\alpha}_{\beta 0}}^2 - {q^{\alpha}_{\beta 1}}^2 + {q^{\alpha}_{\beta 2}}^2 - {q^{\alpha}_{\beta 3}}^2 & 2(q^{\alpha}_{\beta 2}q^{\alpha}_{\beta 3} + q^{\alpha}_{\beta 0}q^{\alpha}_{\beta 1}) \\ 2(q^{\alpha}_{\beta 1}q^{\alpha}_{\beta 3} + q^{\alpha}_{\beta 0}q^{\alpha}_{\beta 2}) & 2(q^{\alpha}_{\beta 2}q^{\alpha}_{\beta 3} - q^{\alpha}_{\beta 0}q^{\alpha}_{\beta 1}) & {q^{\alpha}_{\beta 0}}^2 - {q^{\alpha}_{\beta 1}}^2 - {q^{\alpha}_{\beta 2}}^2 + {q^{\alpha}_{\beta 3}}^2 \end{bmatrix} \tag{3.6}$$

The corresponding DCM can be converted to the quaternion using:

$$
\begin{aligned}
q^{\alpha}_{\beta 0} &= \frac{1}{2}\sqrt{1 + C^{\alpha}_{\beta 1,1} + C^{\alpha}_{\beta 2,2} + C^{\alpha}_{\beta 3,3}} \\
q^{\alpha}_{\beta 1} &= \frac{C^{\alpha}_{\beta 2,3} - C^{\alpha}_{\beta 3,2}}{4q^{\alpha}_{\beta 0}} \\
q^{\alpha}_{\beta 2} &= \frac{C^{\alpha}_{\beta 3,1} - C^{\alpha}_{\beta 1,3}}{4q^{\alpha}_{\beta 0}} \\
q^{\alpha}_{\beta 3} &= \frac{C^{\alpha}_{\beta 1,2} - C^{\alpha}_{\beta 2,1}}{4q^{\alpha}_{\beta 0}}
\end{aligned}
\tag{3.7}
$$

3.3 Inertial Navigation in Local Cartesian Frame

In this section, the concept of inertial navigation in local cartesian frame is discussed.

3.3.1 Inertial Measurement Unit

Inertial navigation systems require measurements of IMU to calculate kinematics of the platform. IMU usually consists of three accelerometers, and three gyroscopes mounted orthogonally.

Accelerometers provide measurements of specific force, which is the summation of the acceleration and negative of the gravitation vector. Several error sources contaminate specific force measurements of accelerometers, such as bias, scale factor, misalignment, measurement noise, etc. The most dominant error sources for low-cost sensors are bias, which is a constant additive error, and measurement noise, which is an additive error with high bandwidth power spectral density.

Gyroscopes used in navigation systems measure rotation rate of the platform. Gyroscope measurements are subject to several error sources similar to accelerometers. The most dominant error sources for low-cost gyroscopes are bias and measurement noise.

3.3.2 Attitude Dynamics

The time derivative of the quaternion can be expressed as in Equation 3.8, [70].

$$\dot{\boldsymbol{q}}_n^b = \frac{1}{2}\boldsymbol{q}_n^b \circ \begin{pmatrix} 0 \\ \boldsymbol{\omega} \end{pmatrix} \tag{3.8}$$

In Equation 3.8, $\boldsymbol{\omega}$ represents the gyroscopic rotation rate measurements and $\circ$ is the quaternion multiplication used as:

$$\boldsymbol{p} \circ \boldsymbol{q} = \begin{pmatrix} p_1 & -p_2 & -p_3 & -p_4 \\ p_2 & p_1 & -p_4 & p_3 \\ p_3 & p_4 & p_1 & -p_2 \\ p_4 & -p_3 & p_2 & p_1 \end{pmatrix} \begin{pmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \end{pmatrix} \tag{3.9}$$

3.3.3 Velocity and Position Dynamics

Velocity dynamics of the platform in local cartesian frame can be expressed by the relation given in Equation 3.10:

$$\dot{\boldsymbol{v}}_b^n = \boldsymbol{C}_b^n \boldsymbol{f}^b + \boldsymbol{g}^n \tag{3.10}$$

In Equation 3.10, $\boldsymbol{f}^b$ represents specific force measured by accelerometers and $\boldsymbol{g}^n$ is the gravitation vector resolved in the local cartesian frame.

The dynamics of the position with respect to local cartesian frame is simply the time derivative of the velocity. The dynamics of the position is given in Equation 3.11.

$$\dot{\boldsymbol{p}}_b^n = \boldsymbol{v}_b^n \tag{3.11}$$

3.4 Imaging Geometry

Optical properties of a camera define the relationship between the image and the scene. A pinhole camera can be constructed to model this relationship for lenses having a very small aperture. Pinhole camera model is shown in Figure 3.1.

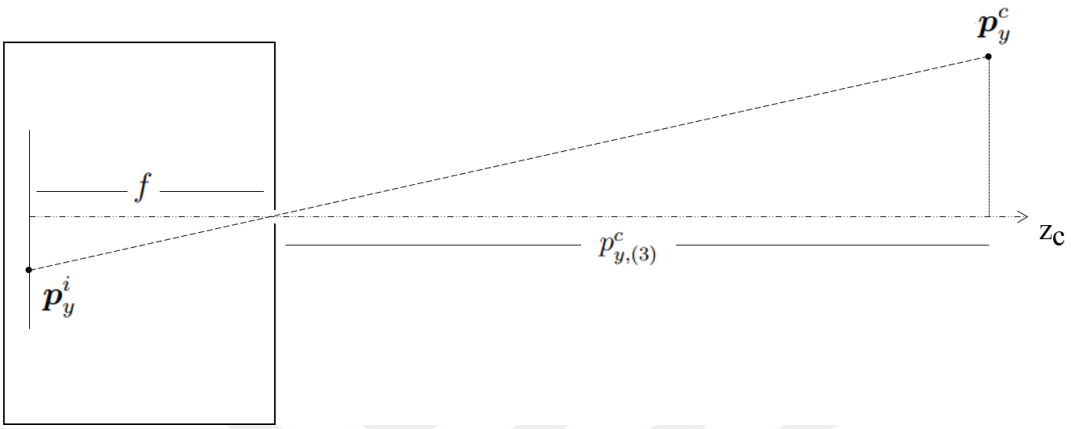

Figure 3.1: Pinhole camera projection model.

Position of a point in the image plane, $\boldsymbol{p}_y^i$, is related to the position of the point in the scene $\boldsymbol{p}_y^c$ and the focal length of the camera f.

$$\boldsymbol{p}_y^i = \frac{f}{p_{y,(3)}^c} \boldsymbol{p}_y^c \tag{3.12}$$

In Equation 3.12, $p_{y,(3)}^c$ is the distance between the point and the optical center of the camera in the z_c axis.

In order to obtain the pixel location of the point in the image, the position of the point in the physical image plane has to be converted to the pixel coordinate system. An example image plane consisting $M \times N$ pixels with height H and width W is shown in Figure 3.2.

The pixel position of the point is related to the position of the point in the image plane

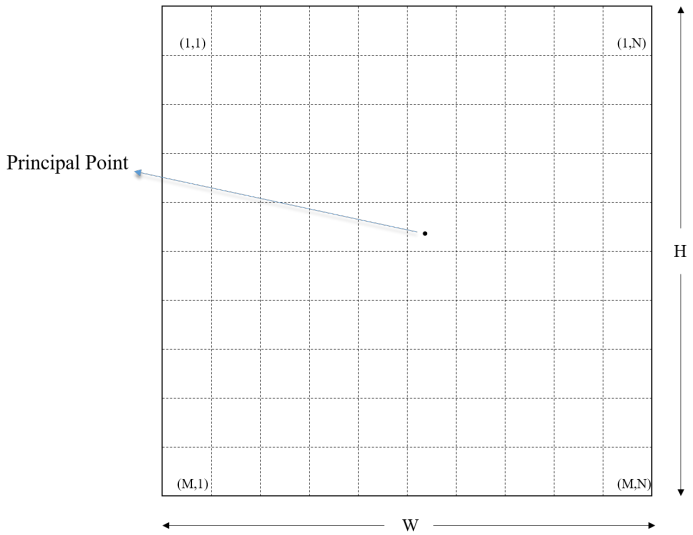

Figure 3.2: Image plane with $M \times N$ pixels and dimensions of H, W.

as:

$$\boldsymbol{p}_y^{pix} = \begin{bmatrix} -\frac{M}{H} & 0 & 0 \\ 0 & \frac{N}{W} & 0 \end{bmatrix} \boldsymbol{p}_y^i + \begin{bmatrix} c_x \\ c_y \end{bmatrix} \tag{3.13}$$

In Equation 3.13, c_x and c_y represent the pixel coordinates of the principal point. The principal point is the geometric center of the image, [71].

Normalized camera coordinate system is an auxiliary coordinate system used in image-based navigation system. A hypothetical image plane with a focal length of 1 is assumed to obtain the relationship between other coordinate systems defined previously. Thus, the position of a point in normalized camera coordinate system is related to the position in the scene as:

$$\underline{\boldsymbol{p}}_y^c = \frac{1}{p_{y,(3)}^c} \boldsymbol{p}_y^c \tag{3.14}$$

In Equation 3.14, $\underline{\boldsymbol{p}}_y^c$ represents the position of the point in the normalized camera coordinate system. Pixel position of the point is related to the position of the point in

normalized camera coordinate system as:

$$\boldsymbol{p}_y^{pix} = \begin{bmatrix} -f\frac{M}{H} & 0 & c_x \\ 0 & f\frac{N}{W} & c_y \end{bmatrix} \underline{\boldsymbol{p}}_y^c \tag{3.15}$$

3.5 Feature Detection and Tracking

Vision-based navigation systems require detection and tracking of features. These processes are essential since pixels of the features are the visual measurements used in estimation.

There are many feature detection methods in the literature. These methods can be categorized as corner detectors and blob detectors. Commonly used corner detection methods are FAST [72], Harris [73], and Shi-Tomasi [74]. Examples of blob detection methods are SIFT [75] and SURF [76]. Blob detection methods provide features that are more distinctive than the corners. However, corner detectors are less computationally complex. On the other hand, it is important to detect features that are more likely to be detected in subsequent images. Hence, using a feature detector which extracts distinctive features is advantageous in vision-based navigation applications.

SIFT method applies the difference-of-Gaussian (DoG) operator on the entire image. The lower and the upper scales of the image are convolved with a DoG operator. Then, all local minima or maxima of the output is detected across space and scales. The output of this step provides the extracted features. Since, SIFT applies the DoG operator on all scales of the image and searches for local minima and maxima in space and scales, features extracted in this method are scale invariant.

After detection of features, tracking of the features in consecutive images is employed. Kanade-Lucas-Tomasi (KLT) feature tracker [77, 74] can be used for this task. This feature tracker is robust to changes in the appearance of features during long sequences [78]. KLT feature tracker searches the best match of the feature on a specific area of the image.

In order to discuss KLT feature tracker, consider two functions; $I_1(\boldsymbol{x})$ and $I_2(\boldsymbol{x})$

which give image intensities at the pixel location $\boldsymbol{x}$ in two images. The aim is to find the displacement $\boldsymbol{d}$ which makes the difference between $I_1(\boldsymbol{x}+\boldsymbol{d})$ and $I_2(\boldsymbol{x})$ minimum in some region R of the image. In order to accomplish this, an L_2 norm error is minimized.

$$E = \sum\nolimits_{\boldsymbol{x}\in R}[I_1(\boldsymbol{x}+\boldsymbol{d}) - I_2(\boldsymbol{x})]^2 \tag{3.16}$$

A linear approximation is made to obtain $I_1(\boldsymbol{x}+\boldsymbol{d})$ as:

$$I_1(\boldsymbol{x}+\boldsymbol{d}) \approx I_1(\boldsymbol{x}) + \boldsymbol{d}\frac{\partial}{\partial \boldsymbol{x}} I_1(\boldsymbol{x}) \tag{3.17}$$

In order to minimize the error E with respect to $\boldsymbol{d}$,

$$\boldsymbol{0} = \frac{\partial}{\partial \boldsymbol{d}} E, \tag{3.18}$$

is set.

$$\boldsymbol{0} \approx \frac{\partial}{\partial \boldsymbol{d}} \sum\nolimits_{\boldsymbol{x}\in R}[I_1(\boldsymbol{x}) + \boldsymbol{d}\frac{\partial I_1}{\partial \boldsymbol{x}} - I_2(\boldsymbol{x})]^2 \tag{3.19}$$

$$\boldsymbol{0} \approx \sum\nolimits_{\boldsymbol{x}\in R} 2\frac{\partial I_1}{\boldsymbol{x}}[I_1(\boldsymbol{x}) + \boldsymbol{d}\frac{\partial I_1}{\partial \boldsymbol{x}} - I_2(\boldsymbol{x})] \tag{3.20}$$

Finally, $\boldsymbol{d}$ can be estimated as:

$$\boldsymbol{d} \approx \left[\sum\nolimits_{\boldsymbol{x}\in R}\left[\frac{\partial I_1}{\partial \boldsymbol{x}}\right]^T [I_2(\boldsymbol{x}) - I_1(\boldsymbol{x})]\right]\left[\sum\nolimits_{\boldsymbol{x}\in R}\left[\frac{\partial I_1}{\partial \boldsymbol{x}}\right]^T \left[\frac{\partial I_1}{\partial \boldsymbol{x}}\right]\right]^{-1} \tag{3.21}$$

3.6 Kalman Filtering

This part provides essentials of Kalman Filtering technique. The section begins with the development of linear Kalman filtering equations while at the end, extension of Kalman filtering for nonlinear problems is provided.

3.6.1 Linear Kalman Filter

Kalman Filter is a recursive Bayesian estimation method, [79]. Estimation process aims to find the optimal solution to a linear stochastic differential equation. The linear stochastic differential equation generally has the following form, [79]:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t) + \boldsymbol{G}(t)\boldsymbol{w}(t) \tag{3.22}$$

In Equation 3.22, $\boldsymbol{x}$, $\boldsymbol{F}$, $\boldsymbol{B}$, $\boldsymbol{u}$, $\boldsymbol{G}$ and $\boldsymbol{w}$ represent the state vector, the system dynamics matrix, the input matrix, the input vector, the noise transformation matrix and the process noise vector, respectively. The process noise is assumed to be zero mean Gaussian noise distributed with the covariance given as:

$$\begin{aligned} E\{\boldsymbol{w}(t)\boldsymbol{w}^T(t+\tau)\} &= \boldsymbol{Q}(t), \text{where } \tau = 0 \\ E\{\boldsymbol{w}(t)\boldsymbol{w}^T(t+\tau)\} &= \boldsymbol{0}, \text{where } \tau \neq 0 \end{aligned} \tag{3.23}$$

The mean of the state vector propagates as:

$$\hat{\boldsymbol{x}}(t) = \boldsymbol{\Phi}(t, t_0)\hat{\boldsymbol{x}}(t_0) + \int_{t_0}^{t} \boldsymbol{\Phi}(t, \tau)\boldsymbol{B}(\tau)\boldsymbol{u}(\tau)d\tau \tag{3.24}$$

In Equation 3.24, $\boldsymbol{\Phi}(t, t_0)$ represents the state transition matrix from time t_0 to time t.

The state vector residual, $\boldsymbol{\delta x}$ is the difference between the estimated state vector $\hat{\boldsymbol{x}}$ and the true state vector $\boldsymbol{x}$. The state vector residual is given as:

$$\boldsymbol{\delta x}(t) = \hat{\boldsymbol{x}}(t) - \boldsymbol{x}(t) \tag{3.25}$$

The error covariance matrix $\boldsymbol{P}$ is given as:

$$\boldsymbol{P}(t) = E\{\boldsymbol{\delta x}(t)\boldsymbol{\delta x}(t)^T\} \tag{3.26}$$

The error covariance matrix propagates according to the system dynamics as:

$$\boldsymbol{P}(t) = \boldsymbol{\Phi}(t, t_0)\boldsymbol{P}(t_0)\boldsymbol{\Phi}(t, t_0)^T + \int_{t_0}^{t} \boldsymbol{\Phi}(t, \tau)\boldsymbol{G}(\tau)\boldsymbol{Q}(\tau)\boldsymbol{G}(\tau)^T\boldsymbol{\Phi}(t, \tau)^T d\tau \quad (3.27)$$

Assuming the interval between two discrete propagation instances, $\Delta t = t_{k+1} - t_k$, is sufficiently small allows us to assume $\boldsymbol{F}(t_{k+1}) \approx \boldsymbol{F}(t_k)$. Thus, the state transition matrix can be approximated as:

$$\boldsymbol{\Phi}(t_k) = exp(\boldsymbol{F}(t_k)\Delta t) \quad (3.28)$$

Equation 3.28 can be solved using a series expansion. Propagation of the error covariance matrix in discrete time is given as:

$$\boldsymbol{P}(t_{k+1}) = \boldsymbol{\Phi}(t_k)\boldsymbol{P}(t_k)\boldsymbol{\Phi}(t_k)^T + \boldsymbol{Q}_d(t_k) \quad (3.29)$$

In Equation 3.29, $\boldsymbol{Q}_d(t_k)$ represents the covariance matrix of discrete time process noise. Calculation of the $\boldsymbol{Q}_d(t_k)$ matrix is discussed in Appendix A.

The measurement vector, $\boldsymbol{z}$, is a set of observations about the outputs of the system. Observations are related to the state vector linearly with additive noise:

$$\boldsymbol{z}(t_i) = \boldsymbol{H}(t_i)\boldsymbol{x}(t_i) + \boldsymbol{\nu}(t_i) \quad (3.30)$$

In Equation 3.30, $\boldsymbol{H}$ and $\boldsymbol{\nu}$ represent measurement matrix and zero-mean white Gaussian noise vector, respectively. The covariance of the measurement noise is defined by:

$$\begin{aligned} E\{\boldsymbol{\nu}(t_i)\boldsymbol{\nu}^T(t_j)\} &= \boldsymbol{R}(t_i), \text{where } i = j \\ E\{\boldsymbol{\nu}(t_i)\boldsymbol{\nu}^T(t_j)\} &= \boldsymbol{0}, \text{where } i \neq j \end{aligned} \quad (3.31)$$

Update mechanism of Kalman Filter is employed when a new measurement is obtained. Assume a new observation is received at time t_i, and the estimate of the state

vector and the covariance matrix are available prior to time t_i (t_i^-), the Kalman gain matrix, $\boldsymbol{K}(t_i)$, can be computed as:

$$\boldsymbol{K}(t_i) = \boldsymbol{P}(t_i^-)\boldsymbol{H}(t_i)^T[\boldsymbol{H}(t_i)\boldsymbol{P}(t_i^-)\boldsymbol{H}(t_i)^T + \boldsymbol{R}(t_i)]^{-1} \tag{3.32}$$

After calculating the Kalman gain matrix, the estimate of the state vector and the covariance matrix can be updated according to:

$$\begin{aligned} \hat{\boldsymbol{x}}(t_i^+) &= \hat{\boldsymbol{x}}(t_i^-) + \boldsymbol{K}(t_i)[\boldsymbol{z}(t_i) - \boldsymbol{H}(t_i)\hat{\boldsymbol{x}}(t_i^-)] \\ \boldsymbol{P}(t_i^+) &= \boldsymbol{P}(t_i^-) - \boldsymbol{K}(t_i)\boldsymbol{H}(t_i)\boldsymbol{P}(t_i^-) \end{aligned} \tag{3.33}$$

3.6.2 Extended Kalman Filter

In the previous section, the development of the linear Kalman filter is discussed. The system dynamics and the measurement model are assumed to be linear in the developed linear Kalman filter. These assumptions are not valid for most of the real applications. Thus, an extension of Kalman filter for nonlinear applications is necessary.

A nonlinear version of the Kalman filter, Extended Kalman filter, is developed in order to be used in such systems. In EKF, state propagation and update mechanisms are governed by nonlinear functions of the state vector elements. Thus, the system dynamics can be represented as, [79]:

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{f}(\boldsymbol{x}(t), \boldsymbol{u}(t), t) + \boldsymbol{G}(t)\boldsymbol{w}(t) \tag{3.34}$$

The nonlinear function, $f(\cdot)$, replaces the system matrix.

It is assumed that the error in the estimate about the state vector is much smaller than the state vector itself. Thus, a linear model can be applied to the propagation of the

error in the state vector estimate as:

$$\boldsymbol{\delta\dot{x}}(t) = \boldsymbol{F}(t)\boldsymbol{\delta x}(t) + \boldsymbol{G}(t)\boldsymbol{w}(t) \tag{3.35}$$

$$\boldsymbol{F}(t) = \left.\frac{\partial \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(t), \boldsymbol{u}=\boldsymbol{u}(t)} \tag{3.36}$$

Assuming the interval between two discrete propagation instances, $\Delta t = t_{k+1} - t_k$, is sufficiently small allows us to assume $\boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t_{k+1}) \approx \boldsymbol{f}(\boldsymbol{x}, \boldsymbol{u}, t_k)$. Thus, the state transition matrix can be approximated as:

$$\boldsymbol{\Phi}(t_k) = exp(\boldsymbol{F}(t_k)\Delta t) \tag{3.37}$$

Equation 3.37 can be solved using a series expansion. Propagation of the error covariance matrix is given as:

$$\boldsymbol{P}(t_{k+1}) = \boldsymbol{\Phi}(t_k)\boldsymbol{P}(t_k)\boldsymbol{\Phi}(t_k)^T + \boldsymbol{Q}_d(t_k) \tag{3.38}$$

In Equation 3.38, $\boldsymbol{Q}_d(t_k)$ represents the covariance matrix of discrete time process noise. Calculation of the $\boldsymbol{Q}_d(t_k)$ matrix is given in Appendix A.

In the EKF, the measurement model is given as:

$$\boldsymbol{z}(t_i) = \boldsymbol{h}(\boldsymbol{x}(t_i), t_i) + \boldsymbol{\nu}(t_i) \tag{3.39}$$

In Equation 3.39, $\boldsymbol{h}(\cdot)$ is a nonlinear function of the state vector. The innovation term $\boldsymbol{r}(t_i)$ can be calculated as:

$$\boldsymbol{r}(t_i) = \boldsymbol{z}(t_i) - \boldsymbol{h}(\boldsymbol{x}(t_i^-), t_i) \tag{3.40}$$

Upon calculation of the innovation, Kalman filter update is carried out:

$$\hat{\boldsymbol{x}}(t_i^+) = \hat{\boldsymbol{x}}(t_i^-) + \boldsymbol{K}(t_i)[\boldsymbol{r}(t_i)] \tag{3.41}$$

The Kalman gain matrix $\boldsymbol{K}(t_i)$ is calculated in the same way as in the linear case. However, the measurement matrix is obtained by linearizing the nonlinear measurement function:

$$\boldsymbol{H}(t_i) = \left.\frac{\partial \boldsymbol{h}(\boldsymbol{x}, t)}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(t_i)} \tag{3.42}$$

Similar to the Kalman gain matrix, the covariance matrix update mechanism is the same as in the linear case.

3.7 MSCKF for Vision-aided Inertial Navigation

MSCKF has become a state-of-the-art method for filtering based VIO approaches due to its consistency and accuracy. The most significant novelty that MSCKF introduced to VIO literature is to maintain a sliding window of past platform poses in its state vector. In contrast to MSCKF, prior studies on EKF based VIO contain a representation of currently tracked feature positions in the state vector. In [61], two different approaches to EKF based VIO are compared in terms of both the accuracy and computational efficiency. Results of Monte Carlo simulations showed that MSCKF based VIO outperforms the other EKF based VIO techniques in pose estimation accuracy while having lower computational demand.

In Figure 3.3, the flow diagram of MSCKF based VIO algorithm is presented. The expressions used to perform the computations of each block are given in the caption of Figure 3.3. INS thread processes IMU measurements in IMU readout rate in order to predict kinematics of the platform and propagate covariance matrix of the filter. On the other hand, camera observes the scene at the frame rate. A new image is processed to extract features and track existing features. Subsequently, feature pixel position measurements are recorded for further use in update thread. Upon obtaining the new

image, current pose solution of the platform is recorded and covariance matrix of the filter is augmented. After visual thread finishes operation, filter update conditions are checked. Two different update triggering mechanisms exist in MSCKF algorithm. The first one is loss of the track of a feature and the other one is reaching the maximum allowed number of pose states. As either one of the update conditions occurs, update thread of MSCKF is executed. Firstly, positions of features that are going to be used in the update step are estimated using the observations of the feature and pose states that the feature was observed. Then, estimated position of the feature is re-projected to the image plane of each observed platform poses to obtain predicted pixel position of the feature. Subsequently, innovation of the feature is calculated by subtracting predicted pixel position of the feature from measurements in each frame. Lastly Kalman update operation is carried out for all features that are processed and both kinematics of the platform and covariance matrix of the filter are updated.

In this section, a detailed explanation about a modified version of the MSCKF based integration method is presented including the structure of the state vector, propagation of the covariance matrix, augmentation of the states and update mechanism of the filter.

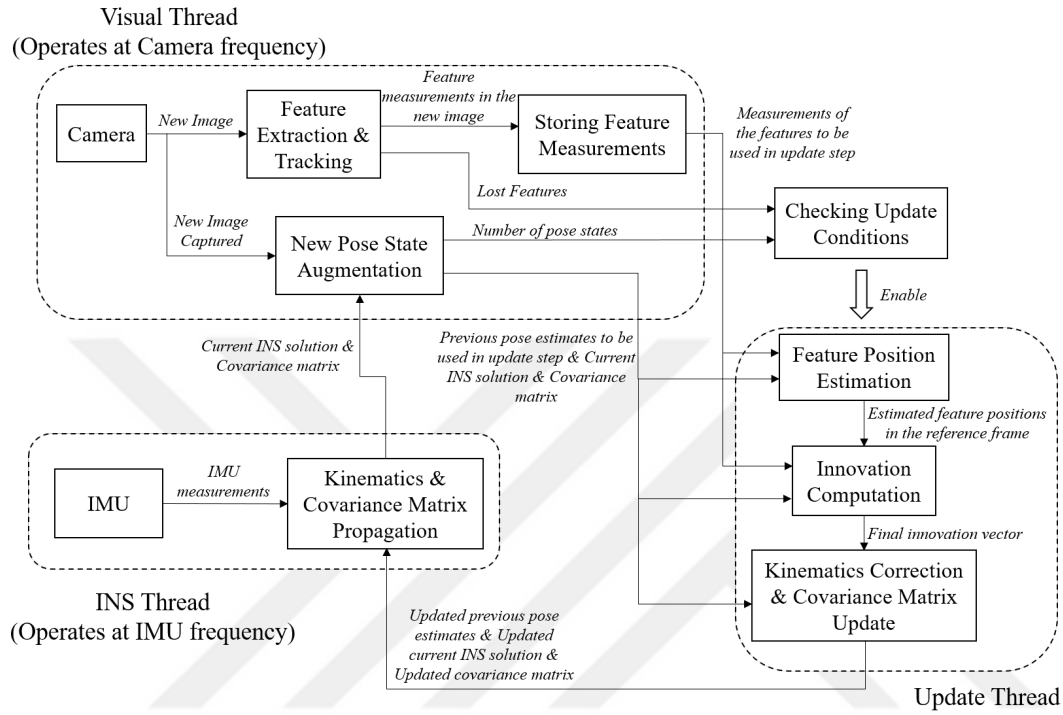

Figure 3.3: Flow diagram of MSCKF based VIO. Kinematics Propagation is applied using Equation 3.49 and Covariance Matrix Propagation is executed using Equation 3.54. New Pose State Augmentation is carried out by equations 3.59 - 3.58. Feature position estimation is achieved using all of the registered measurements of the feature and corresponding platform poses. This estimate is computed using Gauss-Newton least squares minimization. Innovation computation is implemented using equations 3.63, 3.64, 3.65, 3.70, and 3.71. Lastly, Covariance Matrix Update is given by equations 3.73 - 3.78.

3.7.1 Structure of the State Vector

The state vector of the filter comprises two main parts, INS states and a sliding window of previous poses at which images are captured and feature observations are obtained. The INS states are described as:

$$\boldsymbol{\delta x}_I = \begin{bmatrix} \boldsymbol{\delta p}_b^{nT} & \boldsymbol{\delta v}_b^{nT} & \boldsymbol{\delta \psi}^T & \boldsymbol{\delta b}_a{}^T & \boldsymbol{\delta b}_g{}^T \end{bmatrix}^T \tag{3.43}$$

In Equation 3.43, $\boldsymbol{\delta p}_b^n$ and $\boldsymbol{\delta v}_b^n$ are errors on the position and velocity estimates of the platform with respect to local cartesian frame respectively, $\boldsymbol{\delta \psi}$ is the error on the estimated attitude, $\boldsymbol{\delta b}_a$ and $\boldsymbol{\delta b}_g$ are the errors on the estimated biases of the accelerometers and the gyroscopes, respectively.

In the paper of Mourikis and Roumeliotis, [19], a pose state includes errors on the *camera* position and attitude at the time of image was captured. On the contrary, the state vector of a pose at time t_i consists of the errors on the position and attitude of the *platform* at that time instance in this study.

$$\boldsymbol{\delta x}_{C_i} = \begin{bmatrix} \boldsymbol{\delta p}_b^n(t_i)^T & \boldsymbol{\delta \psi}(t_i)^T \end{bmatrix}^T \tag{3.44}$$

Thus, the overall state vector when the last m previous pose states are included becomes as:

$$\boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{\delta x}_I{}^T & \boldsymbol{\delta x}_{C_{k-m}}{}^T & \boldsymbol{\delta x}_{C_{k-m+1}}{}^T & \cdots & \boldsymbol{\delta x}_{C_{k-2}}{}^T & \boldsymbol{\delta x}_{C_{k-1}}{}^T \end{bmatrix}^T \tag{3.45}$$

3.7.2 Propagation

Propagation of the kinematics and covariance matrix is based on the continuous time local cartesian frame inertial navigation. This subsection provides underlying mechanism of the block written as "Kinematics & Covariance Matrix Propagation" in Figure 3.3.

3.7.2.1 Continuous Time System Dynamics

Time evolutions of biases of accelerometer and gyroscopic measurements are defined as:

$$\begin{aligned} \dot{\boldsymbol{b}}_a &= \boldsymbol{n}_{wa} \\ \dot{\boldsymbol{b}}_g &= \boldsymbol{n}_{wg} \end{aligned} \tag{3.46}$$

where $\boldsymbol{n}_{wa}$ and $\boldsymbol{n}_{wg}$ are white Gaussian noise processes.

The specific force and rotation rate measurements, which are corrupted with IMU noise sources, can be expressed as:

$$\begin{aligned} \boldsymbol{f}_m &= \boldsymbol{f} + \boldsymbol{b}_a + \boldsymbol{n}_a \\ \boldsymbol{\omega}_m &= \boldsymbol{\omega} + \boldsymbol{b}_g + \boldsymbol{n}_g \end{aligned} \tag{3.47}$$

Estimated accelerometer and gyroscope biases can be subtracted from noisy measurements.

$$\begin{aligned} \hat{\boldsymbol{f}} &= \boldsymbol{f}_m - \hat{\boldsymbol{b}}_a \\ \hat{\boldsymbol{\omega}} &= \boldsymbol{\omega}_m - \hat{\boldsymbol{b}}_g \end{aligned} \tag{3.48}$$

In the paper of Mourikis and Roumeliotis, [19], continuous-time dynamics of the velocity of the platform is modeled by incorporating rotation of Earth. However, including the Earth's rotation in velocity dynamics requires information about the position of the platform with respect to Earth. Since the aim of this study is to estimate kinematics of the platform in the local cartesian frame, the position of the platform with respect to Earth is not included in the navigation system. Thus, differently from the paper of Mourikis and Roumeliotis, a simple continuous-time dynamics model for the velocity of the platform is governed in this study. Based on this simple model, local cartesian frame inertial navigation of the platform is modeled using the bias removed inertial measurements according to Equation 3.49.

$$
\begin{aligned}
\dot{\hat{\boldsymbol{p}}}_b^n &= \hat{\boldsymbol{v}}_b^n \\
\dot{\hat{\boldsymbol{v}}}_b^n &= \boldsymbol{C}_b^n \hat{\boldsymbol{f}} + \boldsymbol{g}^n \\
\dot{\hat{\boldsymbol{q}}}_n^b &= \frac{1}{2} \hat{\boldsymbol{q}}_n^b \circ \begin{pmatrix} 0 \\ \hat{\boldsymbol{\omega}} \end{pmatrix}
\end{aligned}
\tag{3.49}
$$

The Direction Cosine Matrix $\boldsymbol{C}_b^n$ can be calculated from propagated quaternion using Equation 3.6.

Continuous time propagation of the INS states is:

$$
\dot{\boldsymbol{\delta x}}_I = \boldsymbol{F}\boldsymbol{\delta x}_I + \boldsymbol{G}\boldsymbol{n}_I \tag{3.50}
$$

In Equation 3.50, $\boldsymbol{n}_I = \begin{bmatrix} \boldsymbol{n}_a{}^T & \boldsymbol{n}_g{}^T & \boldsymbol{n}_{wa}{}^T & \boldsymbol{n}_{wg}{}^T \end{bmatrix}^T$ is the process noise vector with covariance matrix of $\boldsymbol{Q}_I$. The matrices $\boldsymbol{F}$ and $\boldsymbol{G}$ for a consumer-grade IMU are given in [70] as:

$$
\boldsymbol{F} = \begin{bmatrix}
\boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\
\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & -[(\boldsymbol{C}_b^n \hat{\boldsymbol{f}})_\times] & \boldsymbol{C}_b^n & \boldsymbol{0}_{3\times3} \\
\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{C}_b^n \\
\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\
\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3}
\end{bmatrix}
\tag{3.51}
$$

$$
\boldsymbol{G} = \begin{bmatrix}
\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\
\boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\
\boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} \\
\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} \\
\boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3}
\end{bmatrix}
\tag{3.52}
$$

3.7.2.2 Discrete Time Implementation

IMU measurements are obtained with a period of Δt, and these measurements are used to calculate kinematics of the platform. The kinematics are calculated using nu-

merical integration of local cartesian frame inertial navigation equations. In addition, covariance matrix of the filter has to be propagated at these time instances.

The covariance matrix of the overall state vector consisting of m pose states can be partitioned as:

$$\boldsymbol{P}_k = \begin{bmatrix} \boldsymbol{P}_{IIk} & \boldsymbol{P}_{ICk} \\ \boldsymbol{P}_{CIk} & \boldsymbol{P}_{CCk} \end{bmatrix} \tag{3.53}$$

In Equation 3.53, $\boldsymbol{P}_{IIk}$ is the 15×15 covariance matrix of the INS states, $\boldsymbol{P}_{CCk}$ is the $6m \times 6m$ covariance matrix of the pose states and $\boldsymbol{P}_{ICk}$ is the $15 \times 6m$ correlation matrix between the INS states and the pose states.

Using this notation, propagation of the covariance matrix is given by:

$$\boldsymbol{P}_{k+1} = \begin{bmatrix} \boldsymbol{\Phi}_k \boldsymbol{P}_{IIk} {\boldsymbol{\Phi}_k}^T + \boldsymbol{Q}_k & \boldsymbol{\Phi}_k \boldsymbol{P}_{ICk} \\ \boldsymbol{P}_{CIk} {\boldsymbol{\Phi}_k}^T & \boldsymbol{P}_{CCk} \end{bmatrix} \tag{3.54}$$

In Equation 3.54, $\boldsymbol{Q}_k$ represents the covariance matrix of discrete time process noise, which can be calculated from the continuous time process noise covariance matrix, $\boldsymbol{Q}_I$, by using the methodology described in Appendix A.

3.7.3 State Augmentation

In the paper of Mourikis and Roumeliotis, [19], a pose state includes errors on the position and attitude of the *camera*. On the contrary, a pose state includes errors on the position and attitude of the *platform* in this study. Thus, differently from the paper of Mourikis and Roumeliotis, whenever a new image is captured errors on the current position and attitude of the platform is appended to the filter state vector, in this study. In addition, the *camera* pose estimate is computed from the platform position and attitude estimate upon recording a new image, in the paper [19]. On the contrary, the current position and attitude estimate of the *platform* is recorded for use in update step, in this study. Hence, computation of the camera position and attitude is not included in the state augmentation step. Instead of, in this study, camera position

and attitude are computed during update phase.

Overall filter state vector before the current pose state is appended is given as:

$$\boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{\delta x}_I{}^T & \boldsymbol{\delta x}_{C_{k-m}}{}^T & \boldsymbol{\delta x}_{C_{k-m+1}}{}^T & \cdots & \boldsymbol{\delta x}_{C_{k-2}}{}^T & \boldsymbol{\delta x}_{C_{k-1}}{}^T \end{bmatrix}^T \tag{3.55}$$

After including the current pose state to the filter state vector, state vector becomes:

$$\boldsymbol{\delta x} = \begin{bmatrix} \boldsymbol{\delta x}_I{}^T & \boldsymbol{\delta x}_{C_{k-m}}{}^T & \boldsymbol{\delta x}_{C_{k-m+1}}{}^T & \cdots & \boldsymbol{\delta x}_{C_{k-2}}{}^T & \boldsymbol{\delta x}_{C_{k-1}}{}^T & \boldsymbol{\delta x}_{C_k}{}^T \end{bmatrix}^T \tag{3.56}$$

In Equation 3.56, $\boldsymbol{\delta x}_{C_k}$ includes errors on the current pose estimate.

$$\boldsymbol{\delta x}_{C_k} = \begin{bmatrix} \boldsymbol{\delta p}_b^n(t_k)^T & \boldsymbol{\delta \psi}(t_k)^T \end{bmatrix}^T \tag{3.57}$$

Thus, the Jacobian matrix of the $\boldsymbol{\delta x}_{C_k}$ with respect to the states of the filter only includes two 3×3 identity matrices for relating the errors of the position and the attitude of the platform. Other blocks of the Jacobian matrix are zero matrices.

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times6m} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times6m} \end{bmatrix} \tag{3.58}$$

After obtaining the Jacobian matrix, the covariance matrix of the filter is augmented accordingly:

$$\boldsymbol{P}_k{}'_{(15+6(m+1))\times(15+6(m+1))} \leftarrow \begin{bmatrix} \boldsymbol{I}_{15+6m} \\ \boldsymbol{J}_{6\times(15+6m)} \end{bmatrix} \boldsymbol{P}_{k(15+6m)\times(15+6m)} \begin{bmatrix} \boldsymbol{I}_{15+6m} \\ \boldsymbol{J}_{6\times(15+6m)} \end{bmatrix}^T \tag{3.59}$$

In Equation 3.59, $\boldsymbol{P}_k$ is the covariance matrix before current pose state is included and $\boldsymbol{P}_k{}'$ is the augmented covariance matrix.

3.7.4 Measurement Model

Measurement model depends on the constraints resulted in multiple poses of the same static feature. A constraint equation can be formed from all the image measurements of the same point relating poses at which the measurements were obtained.

Position of the feature in the normalized camera coordinate system is used as the measurements of the filter. Since feature extraction and tracking phases provide pixels of the features, a transformation from pixels to the normalized camera coordinate system has to be employed.

$$\underline{\boldsymbol{p}}_f^c = \begin{bmatrix} \frac{-H}{fM} & 0 & \frac{c_x H}{fM} \\ 0 & \frac{W}{fN} & \frac{-c_y W}{fN} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_y^{pix} \\ 1 \end{bmatrix} \tag{3.60}$$

In Equation 3.60, H, M, W, N, f, c_x, and c_y are the camera intrinsic parameters defined in Section 3.4. Parameters of the transformation matrix given in Equation 3.60 is included in the Fast Flight Dataset.

Assume that measurements of a static feature f_j have been obtained from a set of M_j poses. Each of the observations of the feature is described as (where $i = 1, 2, \cdots M_j$):

$$\boldsymbol{z}_{j,i} = \frac{1}{p_{f_j,(3)}^{c_i}} \begin{bmatrix} p_{f_j,(1)}^{c_i} \\ p_{f_j,(2)}^{c_i} \end{bmatrix} + \boldsymbol{n}_{j,i} \tag{3.61}$$

In Equation 3.61, $\boldsymbol{n}_{j,i}$ is the 2×1 image noise vector, with covariance matrix $\boldsymbol{R}_{j,i} = \sigma_{im}^2 \boldsymbol{I}_{2\times 2}$.

Feature position in the camera coordinate system $\boldsymbol{p}_{f_j}^{c_i}$ can be calculated according to the below relationship.

$$\boldsymbol{p}_{f_j}^{c_i} = \boldsymbol{C}_n^{c_i}(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) \tag{3.62}$$

In Equation 3.62, $\boldsymbol{p}_{f_j}^n$ represents the 3D position of the feature in the local cartesian coordinate system. Since this position is unknown, an estimate has to be generated.

Gauss-Newton least squares minimization is applied using the measurements of the feature and corresponding platform pose estimates in order to estimate the 3D position of the feature. After obtaining a position estimate of the feature in the local cartesian frame, the feature is reprojected to each image plane, at which observations are registered. Subsequently, measurement innovations can be calculated.

$$\boldsymbol{r}_{j,i} = \hat{\boldsymbol{z}}_{j,i} - \boldsymbol{z}_{j,i} \tag{3.63}$$

In Equation 3.63, $\hat{\boldsymbol{z}}_{j,i}$ represents the reprojected measurement prediction of the feature f_j in the i^{th} frame:

$$\hat{\boldsymbol{z}}_{j,i} = \frac{1}{\hat{p}^{c_i}_{f_j,(3)}} \begin{bmatrix} \hat{p}^{c_i}_{f_j,(1)} \\ \hat{p}^{c_i}_{f_j,(2)} \end{bmatrix} \tag{3.64}$$

$$\begin{bmatrix} \hat{p}^{c_i}_{f_j,(1)} \\ \hat{p}^{c_i}_{f_j,(2)} \\ \hat{p}^{c_i}_{f_j,(3)} \end{bmatrix} = \hat{\boldsymbol{C}}^{c_i}_{n} (\hat{\boldsymbol{p}}^{n}_{f_j} - \hat{\boldsymbol{p}}^{n}_{c_i}) \tag{3.65}$$

Equation 3.63 can be linearized about the filter state vector and the errors on feature position $\boldsymbol{\delta p}^{n}_{f_j}$ as:

$$\boldsymbol{r}_{j,i} = \boldsymbol{H}_{X\,j,i} \boldsymbol{\delta x}_{C_i} + \boldsymbol{H}_{f\,j,i} \boldsymbol{\delta p}^{n}_{f_j} + \boldsymbol{n}_{j,i} \tag{3.66}$$

In Equation 3.66, $\boldsymbol{H}_{X\,j,i}$ and $\boldsymbol{H}_{f\,j,i}$ are the jacobians of the measurement $\boldsymbol{z}_{j,i}$ with respect to the state and the feature position estimate, respectively.

$$\boldsymbol{H}_{X\,j,i} = \begin{bmatrix} \boldsymbol{0}_{2\times 15} & \cdots & -\boldsymbol{J}_{j,i} \hat{\boldsymbol{C}}^{c_i}_{n} & \boldsymbol{J}_{j,i} \hat{\boldsymbol{C}}^{c_i}_{n} [(\hat{\boldsymbol{p}}^{n}_{f_j} - \hat{\boldsymbol{p}}^{n}_{c_i})]_{\times} & \cdots \end{bmatrix} \tag{3.67}$$

$$\boldsymbol{H}_{f\,j,i} = \boldsymbol{J}_{j,i} \hat{\boldsymbol{C}}^{c_i}_{n} \tag{3.68}$$

The Jacobian matrix, $\boldsymbol{J}_{j,i}$, in Equation 3.68 is:

$$\boldsymbol{J}_{j,i} = \frac{1}{\hat{p}^{c_i}_{f_j,(3)}} \begin{bmatrix} 1 & 0 & -\dfrac{\hat{p}^{c_i}_{f_j,(1)}}{\hat{p}^{c_i}_{f_j,(3)}} \\ 0 & 1 & -\dfrac{\hat{p}^{c_i}_{f_j,(2)}}{\hat{p}^{c_i}_{f_j,(3)}} \end{bmatrix} \tag{3.69}$$

Derivation of the Jacobian matrices is provided in Appendix B.

Innovation vector of the feature f_j can be obtained by stacking the innovations of the feature f_j from all M_j measurements:

$$\boldsymbol{r}_j = \boldsymbol{H}_{X_j}\boldsymbol{\delta x} + \boldsymbol{H}_{f_j}\boldsymbol{\delta p}^n_{f_j} + \boldsymbol{n}_j \tag{3.70}$$

Since the observations of the feature in different images are independent, the measurement noise covariance matrix is $\boldsymbol{R}_j = \sigma^2_{im}\boldsymbol{I}_{(2M_j)\times(2M_j)}$.

As it is mentioned before, feature position estimate is generated using the observations of the feature and corresponding platform pose estimates. Consequently, errors on the feature position estimate are correlated to the errors on the platform pose estimates. Hence, calculated innovation in Equation 3.70 is not in the form of Equation 3.40. Thus, this innovation term cannot be applied directly in the EKF update step. In order to overcome this problem, an auxiliary innovation term ($\boldsymbol{r}_{oj}$), which is obtained by projecting $\boldsymbol{r}_j$ to the left nullspace of the matrix $\boldsymbol{H}_{f_j}$, is defined. Let $\boldsymbol{A}$ denote the unitary matrix whose columns form the basis of the left nullspace of $\boldsymbol{H}_{f_j}$, $\boldsymbol{r}_{oj}$ can be calculated as:

$$\begin{aligned} \boldsymbol{r}_{oj} = \boldsymbol{A}^T(\hat{\boldsymbol{z}}_j - \boldsymbol{z}_j) &\approx \boldsymbol{A}^T\boldsymbol{H}_{X_j}\boldsymbol{\delta x} + \boldsymbol{A}^T\boldsymbol{n}_j \\ &\approx \boldsymbol{H}_{oj}\boldsymbol{\delta x} + \boldsymbol{n}_{oj} \end{aligned} \tag{3.71}$$

Since the matrix $\boldsymbol{H}_{f_j}$ has full column rank and has dimension of $2M_j \times 3$, its left nullspace is of dimension $2M_j - 3$. Thus, $\boldsymbol{r}_{oj}$ is a $2M_j - 3$ vector. As a result, the

measurement noise covariance matrix is:

$$E\{\boldsymbol{n}_{oj}{\boldsymbol{n}_{oj}}^T\} = \sigma_{im}^2 \boldsymbol{A}^T \boldsymbol{A} = \sigma_{im}^2 \boldsymbol{I}_{(2M_j-3)\times(2M_j-3)} \tag{3.72}$$

3.7.5 Update

After obtaining innovations and measurement Jacobian matrix of features, which are going to be used in the update step, Kalman filter update can be carried out. Overall innovation vector can be obtained by stacking all innovations of the features in a single vector as:

$$\boldsymbol{r}_o = \boldsymbol{H}_o \boldsymbol{\delta x} + \boldsymbol{n}_o \tag{3.73}$$

In Equation 3.73, $\boldsymbol{r}_o$, $\boldsymbol{H}_o$, and $\boldsymbol{n}_o$ are vectors and matrix which are formed by block elements of $\boldsymbol{r}_{oj}$, $\boldsymbol{H}_{oj}$, and $\boldsymbol{n}_{oj}$, respectively.

In [19], the authors proposed employing the QR decomposition of the $\boldsymbol{H}_o$ matrix in order to reduce computational complexity.

$$\boldsymbol{H}_o = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{T}_H \\ \boldsymbol{0} \end{bmatrix} \tag{3.74}$$

In Equation 3.74, $\boldsymbol{T}_H$ is an upper triangular matrix, $\boldsymbol{Q}_1$ and $\boldsymbol{Q}_2$ are unitary matrices whose columns form bases for range and nullspace of $\boldsymbol{H}_o$ respectively. After obtaining QR decomposition of $\boldsymbol{H}_o$ matrix:

$$\boldsymbol{r}_o = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{T}_H \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\delta x} + \boldsymbol{n}_o \tag{3.75}$$

$$\begin{bmatrix} {\boldsymbol{Q}_1}^T \boldsymbol{r}_o \\ {\boldsymbol{Q}_2}^T \boldsymbol{r}_o \end{bmatrix} = \begin{bmatrix} \boldsymbol{T}_H \\ \boldsymbol{0} \end{bmatrix} \boldsymbol{\delta x} + \begin{bmatrix} {\boldsymbol{Q}_1}^T \boldsymbol{n}_o \\ {\boldsymbol{Q}_2}^T \boldsymbol{n}_o \end{bmatrix} \tag{3.76}$$

$$\boldsymbol{r}_n = \boldsymbol{Q}_1{}^T \boldsymbol{r}_o = \boldsymbol{T}_H \boldsymbol{\delta x} + \boldsymbol{n}_n \tag{3.77}$$

can be found. In Equation 3.77, $\boldsymbol{n}_n = \boldsymbol{Q}_1{}^T \boldsymbol{n}_o$ represents the measurement noise vector with covariance matrix of $\boldsymbol{R}_n = \sigma_{im}^2 \boldsymbol{I}_{r\times r}$ with r being the number of columns of $\boldsymbol{Q}_1$. Subsequently, Kalman gain matrix can be computed as:

$$\boldsymbol{K} = \boldsymbol{P}\boldsymbol{T}_H{}^T (\boldsymbol{T}_H \boldsymbol{P}\boldsymbol{T}_H{}^T + \boldsymbol{R}_n)^{-1} \tag{3.78}$$

Two events are introduced in [19] to trigger filter update mechanism. If a feature track is lost, all the corresponding measurements of the feature are processed in the Kalman update. The other triggering mechanism occurs if the number of stored pose states reaches the limit. Since including a pose state to the filter increases number of required computation in Kalman filter equations, maximum number of stored pose states has to be limited considering the computational resource of the system.

3.7.6 MSCKF Algorithm

In this subsection all the equations used in MSCKF algorithm are given once again in order.

Every measurements obtained from IMU are processed to propagate kinematics and covariance matrix. Propagation of kinematics is performed using:

$$\hat{\boldsymbol{q}}_n^b(t_{k+1}) = \hat{\boldsymbol{q}}_n^b(t_k) \circ \begin{pmatrix} 1 \\ \frac{1}{2}\boldsymbol{\omega}(t_k)\Delta t \end{pmatrix} \tag{3.79}$$

$$\hat{\boldsymbol{v}}_b^n(t_{k+1}) = \hat{\boldsymbol{v}}_b^n(t_k) + (\hat{\boldsymbol{C}}_b^n(t_k)\boldsymbol{f}^b(t_k) + \boldsymbol{g}^n(t_k))\Delta t \tag{3.80}$$

$$\hat{\boldsymbol{p}}_b^n(t_{k+1}) = \hat{\boldsymbol{p}}_b^n(t_k) + \hat{\boldsymbol{v}}_b^n(t_k)\Delta t \tag{3.81}$$

In order to propagate the covariance matrix, firstly continuous time system dynamics

matrix and the noise transformation matrix are formed.

$$\boldsymbol{F}_k = \begin{bmatrix} \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -[(\hat{\boldsymbol{C}}_b^n(t_k)\hat{\boldsymbol{f}}(t_k))_\times] & \hat{\boldsymbol{C}}_b^n(t_k) & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \hat{\boldsymbol{C}}_b^n(t_k) \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \end{bmatrix} \tag{3.82}$$

$$\boldsymbol{G}_k = \begin{bmatrix} \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{I}_{3\times3} \end{bmatrix} \tag{3.83}$$

Subsequently, state transition matrix can be calculated using second order Taylor Series expansion.

$$\begin{aligned} \boldsymbol{\Phi}_k &= exp(\boldsymbol{F}_k\Delta t) \\ \boldsymbol{\Phi}_k &\approx \boldsymbol{I} + \boldsymbol{F}_k\Delta t + \frac{\Delta t}{2!}{\boldsymbol{F}_k}^2 \end{aligned} \tag{3.84}$$

In order to calculate covariance matrix of discrete time process noise $\boldsymbol{Q}_k$, continuous time process noise matrix $\boldsymbol{Q}_I$ is formed.

$$\boldsymbol{Q}_I = \begin{bmatrix} \boldsymbol{Q}_a & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \boldsymbol{Q}_g & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{Q}_w a & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \boldsymbol{Q}_w g \end{bmatrix} \tag{3.85}$$

Then, covariance matrix of discrete time process noise can be approximated as:

$$\boldsymbol{Q}_k \approx \boldsymbol{\Phi}_k\boldsymbol{G}_k\boldsymbol{Q}_I{\boldsymbol{G}_k}^T{\boldsymbol{\Phi}_k}^T\Delta t \tag{3.86}$$

Lastly, covariance matrix of the filter can be propagated according to:

$$\boldsymbol{P}_{k+1} = \begin{bmatrix} \boldsymbol{\Phi}_k \boldsymbol{P}_{IIk} \boldsymbol{\Phi}_k{}^T + \boldsymbol{Q}_k & \boldsymbol{\Phi}_k \boldsymbol{P}_{ICk} \\ \boldsymbol{P}_{CIk} \boldsymbol{\Phi}_k{}^T & \boldsymbol{P}_{CCk} \end{bmatrix} \tag{3.87}$$

When a new image is captured, feature extraction and tracking tasks are performed and lost features are determined. Pixels of the features that are tracked in the current image and pixels of the new features are converted to the normalized camera coordinate system using:

$$\underline{\boldsymbol{p}}_f^c = \begin{bmatrix} \frac{-H}{fM} & 0 & \frac{c_x H}{fM} \\ 0 & \frac{W}{fN} & \frac{-c_y W}{fN} \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_y^{pix} \\ 1 \end{bmatrix} \tag{3.88}$$

In Equation 3.88, H, M, W, N, f, c_x, and c_y are the camera intrinsic parameters defined in Section 3.4. After obtaining measurements of the features at the normalized camera coordinate system, these measurements are recorded for further use in update phase.

Subsequently, current platform position, $\hat{\boldsymbol{p}}_b^n(t_k)$, and attitude, $\hat{\boldsymbol{q}}_n^b(t_k)$, estimates are recorded for further use in update phase. In addition to that, covariance matrix of the filter is augmented since the current pose state is appended to the overall filter state vector. Firstly, the Jacobian matrix of the current pose state with respect to the states of the filter is formed:

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times6m} \\ \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{I}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times3} & \boldsymbol{0}_{3\times6m} \end{bmatrix} \tag{3.89}$$

After obtaining the Jacobian matrix, the covariance matrix of the filter is augmented accordingly:

$$\boldsymbol{P}_{k'(15+6(m+1))\times(15+6(m+1))} \leftarrow \begin{bmatrix} \boldsymbol{I}_{15+6m} \\ \boldsymbol{J}_{6\times(15+6m)} \end{bmatrix} \boldsymbol{P}_{k(15+6m)\times(15+6m)} \begin{bmatrix} \boldsymbol{I}_{15+6m} \\ \boldsymbol{J}_{6\times(15+6m)} \end{bmatrix}^T \tag{3.90}$$

Subsequently, filter update triggering events are checked. Lost features and the remaining features that were observed at the oldest pose are determined. Gauss-Newton least squares minimization is applied using the measurements of each feature and corresponding platform pose estimates in order to estimate the 3D position of each feature in local cartesian frame separately. Then, each feature is reprojected to each image plane. This reprojection is applied using the estimated position of the feature in local cartesian frame, $\hat{\boldsymbol{p}}_{f_j}^n$, platform position, $\hat{\boldsymbol{p}}_{c_i}^n$, and attitude, $\hat{\boldsymbol{C}}_n^{c_i}$ estimates at the time when the image was captured.

$$\begin{bmatrix} \hat{p}_{f_j,(1)}^{c_i} \\ \hat{p}_{f_j,(2)}^{c_i} \\ \hat{p}_{f_j,(3)}^{c_i} \end{bmatrix} = \hat{\boldsymbol{C}}_n^{c_i}(\hat{\boldsymbol{p}}_{f_j}^n - \hat{\boldsymbol{p}}_{c_i}^n) \tag{3.91}$$

$$\hat{\boldsymbol{z}}_{j,i} = \frac{1}{\hat{p}_{f_j,(3)}^{c_i}} \begin{bmatrix} \hat{p}_{f_j,(1)}^{c_i} \\ \hat{p}_{f_j,(2)}^{c_i} \end{bmatrix} \tag{3.92}$$

Then, innovation vector of the feature f_j in the i^{th} frame can be calculated using the reprojected measurement prediction, $\hat{\boldsymbol{z}}_{j,i}$, and the measurement of the feature in the i^{th} frame, $\boldsymbol{z}_{j,i}$.

$$\boldsymbol{r}_{j,i} = \hat{\boldsymbol{z}}_{j,i} - \boldsymbol{z}_{j,i} \tag{3.93}$$

In addition to that, Jacobian matrices of the measurement with respect to the states and the feature position estimate are calculated.

$$\boldsymbol{H}_{X\,j,i} = \begin{bmatrix} \boldsymbol{0}_{2\times 15} & \cdots & -\boldsymbol{J}_{j,i}\hat{\boldsymbol{C}}_n^{c_i} & \boldsymbol{J}_{j,i}\hat{\boldsymbol{C}}_n^{c_i}[(\hat{\boldsymbol{p}}_{f_j}^n - \hat{\boldsymbol{p}}_{c_i}^n)]_\times & \cdots \end{bmatrix} \tag{3.94}$$

$$\boldsymbol{H}_{f\,j,i} = \boldsymbol{J}_{j,i}\hat{\boldsymbol{C}}_n^{c_i} \tag{3.95}$$

The Jacobian matrix, $\boldsymbol{J}_{j,i}$, in Equation 3.68 is:

$$\boldsymbol{J}_{j,i} = \frac{1}{\hat{p}^{c_i}_{f_j,(3)}} \begin{bmatrix} 1 & 0 & -\dfrac{\hat{p}^{c_i}_{f_j,(1)}}{\hat{p}^{c_i}_{f_j,(3)}} \\ 0 & 1 & -\dfrac{\hat{p}^{c_i}_{f_j,(2)}}{\hat{p}^{c_i}_{f_j,(3)}} \end{bmatrix} \tag{3.96}$$

Then, all innovation vectors, Jacobian matrices of the measurement with respect to the states and Jacobian matrices of the measurement with respect to the feature position estimate are stacked in order to obtain overall innovation vector, $\boldsymbol{r}_j$, and overall Jacobian matrices, $\boldsymbol{H}_{Xj}$, $\boldsymbol{H}_{fj}$ of the feature f_j. Following that, the unitary matrix $\boldsymbol{A}$ is obtained whose columns form the basis of the left nullspace of $\boldsymbol{H}_{fj}$. The,n an auxiliary innovation vector, $\boldsymbol{r}_{oj}$, and an auxiliary measurement matrix, $\boldsymbol{H}_{oj}$, for the feature f_j can be obtained.

$$\boldsymbol{r}_{oj} = \boldsymbol{A}^T \boldsymbol{r}_j \tag{3.97}$$

$$\boldsymbol{H}_{oj} = \boldsymbol{A}^T \boldsymbol{H}_{Xj} \tag{3.98}$$

Measurement noise covariance matrix, $\boldsymbol{R}_{oj}$, for M_j measurements of the feature f_j is formed as:

$$\boldsymbol{R}_{oj} = \sigma^2_{im} \boldsymbol{I}_{(2M_j-3)\times(2M_j-3)} \tag{3.99}$$

Then, all innovation vectors and all measurement Jacobian matrices of all the features are stacked in order to obtain overall innovation vector, $\boldsymbol{r}_o$, and overall measurement Jacobian matrix, $\boldsymbol{H}_o$. Subsequently, QR decomposition of the $\boldsymbol{H}_o$ matrix is employed.

$$\boldsymbol{H}_o = \begin{bmatrix} \boldsymbol{Q}_1 & \boldsymbol{Q}_2 \end{bmatrix} \begin{bmatrix} \boldsymbol{T}_H \\ \boldsymbol{0} \end{bmatrix} \tag{3.100}$$

After that, final innovation vector is obtained using:

$$\boldsymbol{r}_n = \boldsymbol{Q}_1{}^T \boldsymbol{r}_o \tag{3.101}$$

And measurement noise matrix is formed by:

$$\boldsymbol{R}_n = \sigma_{im}^2 \boldsymbol{I}_{r\times r} \tag{3.102}$$

In Equation 3.102, r is the number of columns of $\boldsymbol{Q}_1$.

Then, Kalman gain matrix is computed.

$$\boldsymbol{K} = \boldsymbol{P}_k^- \boldsymbol{T}_H{}^T (\boldsymbol{T}_H \boldsymbol{P}_k^- \boldsymbol{T}_H{}^T + \boldsymbol{R}_n)^{-1} \tag{3.103}$$

Covariance matrix of the filter is updated.

$$\boldsymbol{P}_k^+ = \boldsymbol{P}_k^- - \boldsymbol{K}\boldsymbol{T}_H \boldsymbol{P}_k^- \tag{3.104}$$

Finally, states of the filter are estimated and estimated errors are removed from the estimated kinematics.

$$\boldsymbol{\delta \hat{x}} = \boldsymbol{K}\boldsymbol{r}_n \tag{3.105}$$

CHAPTER 4

PROPOSED METHODOLOGY

This chapter focuses on the proposed keyframe-based MSCKF method for VIO described earlier. Firstly, VIO dataset used in offline experiments is presented. Then, experimental results related to the effect of allowed maximum number of pose states stored in the filter are shared and computational complexity of the algorithm is discussed. Lastly, keyframe-based MSCKF technique is introduced.

4.1 Fast Flight Dataset

In this work, offline experiments have been conducted using the "Fast Flight Dataset" [68, 80]. There are four flight datasets with speeds of 5 m/s, 10 m/s, 15 m/s, 17.5 m/s collected over an airport runway. The datasets consist of visual and inertial measurements of 600 meters long round-trip level flight. Two forward looking Point-Grey CM3-U3-13Y3M-CS cameras running at 40 Hz with 960×800 resolution are mounted on the platform which provide stereo images. Besides, the platform contains a VectorNav VN-100 Rugged IMU running at 200 Hz. IMU messages and stereo images are time synchronized. In addition to that, the platform contains a GPS receiver-antenna module so that GPS position and velocity measurements are provided in the dataset as reference data. The cameras are forward looking and obtaining images of the front scene of the platform. An example image from the dataset is shared in Figure 4.1. Since this study focuses on monocular visual-inertial odometry based on MSCKF, images from only one of the cameras are used in the experiments of this study.

In Table 4.1 and Table 4.2, specifications of gyroscopes and accelerometers of Vec-

Figure 4.1: An example image from Fast Flight Dataset.

torNav VN-100 Rugged IMU are given, respectively [81].

During the experiments, covariance matrix presented in Equation 4.1 is used as the continuous time process noise covariance matrix. In Equation 4.1:

- S_a is the power spectral density of the accelerometer random noise and set in the experimentation as $1e-4\frac{m^2}{s^2}$,
- S_g is the power spectral density of the gyroscope random noise and set in the experimentation as $1e-6rad^2$,
- S_{wa} is the power spectral density of the accelerometer bias variation and set in the experimentation as $1e-4\frac{m^2}{s^4}$,
- S_{wg} is the power spectral density of the gyroscope bias variation and set in the

Table 4.1: Specifications of gyroscopes of VectorNav VN-100

Range	$\pm$ 2000 °/s
In-Run Bias Stability	< 10 °/hr
Linearity	< 0.1 % FS
Noise Density	0.0035 °/s $\sqrt{\text{Hz}}$
Bandwidth	256 Hz
Alignment Error	$\pm$0.05°
Resolution	< 0.02 °/s

Table 4.2: Specifications of accelerometers of VectorNav VN-100

Range	$\pm$ 16 g
In-Run Bias Stability	< 0.04 mg
Linearity	< 0.5 % FS
Noise Density	0.14 mg/$\sqrt{\text{Hz}}$
Bandwidth	260 Hz
Alignment Error	$\pm$0.05°
Resolution	< 0.5 mg

experimentation as $1e-6\frac{rad^2}{s^2}$.

$$\boldsymbol{Q}_I = \begin{bmatrix} S_a\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & S_g\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & S_{wa}\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & S_{wg}\boldsymbol{I}_{3\times3} \end{bmatrix} \tag{4.1}$$

Measurement noise variance, σ_{im}^2, is set as $1e-4$ during the experiments. Initial state covariance matrix is given in Equation 4.2. In Equation 4.2:

- σ_p^2 is the initial variance of the error on the platform position and set in the experimentation as $25m^2$,

- σ_v^2 is the initial variance of the error on the platform velocity and set in the experimentation as $6.25e-2\frac{m^2}{s^2}$,
- σ_ψ^2 is the initial variance of the error on the platform attitude with respect to the local navigation frame and set in the experimentation as $1.6e-3rad^2$,
- σ_{ba}^2 is the initial variance of the error on the accelerometer bias and set in the experimentation as $9.6e-3\frac{m^2}{s^4}$,
- σ_{bg}^2 is the initial variance of the error on the gyroscope bias and set in the experimentation as $9.4e-5\frac{rad^2}{s^2}$,

$$\boldsymbol{P}_0 = \begin{bmatrix} \sigma_p^2\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \sigma_v^2\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \sigma_\psi^2\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \sigma_{ba}^2\boldsymbol{I}_{3\times3} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \sigma_{bg}^2\boldsymbol{I}_{3\times3} \end{bmatrix} \tag{4.2}$$

In each flights of Fast Flight Dataset, the vehicle is stationary during the first second of operation. So, in this study measurements of accelerometers during the first second are used to initialize the roll and pitch angle of the platform with respect to the local navigation frame. For this purpose coarse alignment technique is used [70].

4.2 Visual Front-End

Feature-based VIO algorithms require detection and tracking of the features in image sequences. Thus, a visual front-end has to be constructed in order to extract features in images and continuously track them. Since the feature detection and tracking algorithms are not in the scope of this study, public libraries are used.

During the experiments of this study, SIFT features are extracted using OpenCV library [82]. An example image showing extracted features in a sample image of Fast Flight Dataset is presented in Figure 4.2. In the experimentation, a feature selection mechanism is employed in order to reduce the required memory and processing time

while having homogeneously distributed features in the image. Image grids with an area of 80 $\times$ 80 pixels are constructed. Only two features in each image grid are selected. Two features are selected based on the responses of the features to the DoG operator. Two features having the highest responses are selected for tracking in each grid.

Figure 4.2: Extracted SIFT features in a sample image from Fast Flight Dataset.

During the experiments, features are tracked in image sequences using KLT feature tracker algorithm of MATLAB® [83]. Tracked features in a sample of consecutive image pair of Fast Flight Dataset are presented in Figure 4.3.

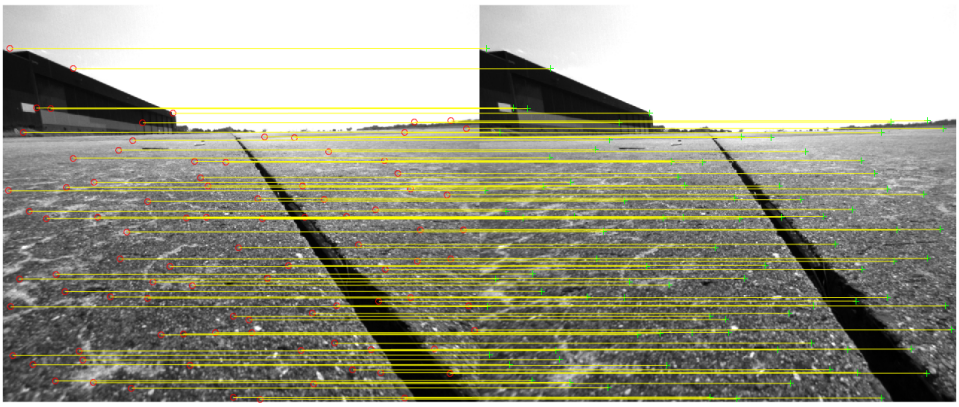

Figure 4.3: Tracked features in a sample of consecutive image pair of Fast Flight Dataset.

4.3 Effect of Maximum Number of Stored Pose States

Two events to trigger Kalman update phase of the MSCKF algorithm are mentioned in Section 3.7.5. One of the triggering condition is reaching the maximum number of pose states allowed, N_{max}, in the filter. Since computational resources of real-time systems are limited, there has to be a limit for the number of states can be stored. At least one of the old pose states has to be marginalized and measurements obtained from the corresponding state(s) have to be used in Kalman update phase when the limit is reached.

In order to observe the effect of N_{max} on the performance of MSCKF algorithm, experiments with different N_{max} values have been conducted on four datasets available in Fast Flight Dataset. Final position errors of standard MSCKF algorithm experimented with 4 different datasets are given in Figure 4.4 - Figure 4.5 and in Table 4.3 - Table 4.4. In these experiments, used N_{max} values are 10, 13, 15, 18, 20, 30, 40, 50, 60, 80, and 100. Position errors are determined by using GPS position measurements as the reference position of the platform.

Effect of increasing the maximum number of pose states stored in the filter can be observed from presented results of experiments. As the N_{max} value is increased, it is observed that the position errors generally decreased in all datasets. It can be seen

Table 4.3: Final horizontal position errors for different values of N_{max}

N_{max}	5 m/s dataset, (m)	10 m/s dataset, (m)	15 m/s dataset, (m)	17.5 m/s dataset, (m)
10	90.05	145.9	276.25	78.9
13	93.8	94.5	139	23.12
15	93.63	32.49	110	38.01
18	78.24	33.28	4.28	13.3
20	68.37	33.7	14.52	19.97
30	69.32	35.69	7.47	19.07
40	50.55	29.66	6.81	24.89
50	45.71	26.76	2.45	19.37
60	33.89	25.45	5.19	19.91
80	64.57	15.67	6.61	20.32
100	48.52	22.16	1.22	20.43

from the results that N_{max} should be selected as more than 20 states in order to ensure optimum positioning performance for all datasets. Position errors are mostly much higher when N_{max} is selected below 20 than the errors obtained when N_{max} is longer. However, increasing N_{max} beyond a moderate value for each dataset has less effect on accuracy.

In [84], the author examined the dependency of the computational complexity of the MSCKF algorithm on the number of pose states in the filter. It can be seen from the dissertation of Li [84] that overall complexity of the filter becomes cubic in the number of pose states stored. Although increasing N_{max} to a proper value results in an acceptable positioning performance, excessive increase in computational resources required to execute the filter operations is a major drawback.

Table 4.4: Final vertical position errors for different values of N_{max}

N_{max}	5 m/s dataset, (m)	10 m/s dataset, (m)	15 m/s dataset, (m)	17.5 m/s dataset, (m)
10	19.92	18.99	10.13	4.8
13	7.59	35.7	12.86	1.74
15	3.62	6.6	14.06	0.27
18	0.18	5.72	8.29	5.41
20	2.99	4.79	7.4	2.19
30	4.75	5.15	7.44	0.07
40	14.23	4.12	6.28	1.44
50	3.41	3.78	6.54	0.71
60	5.24	2.59	7.08	0.43
80	0.42	2.14	6.03	0.25
100	3.16	2.34	5.27	1.28

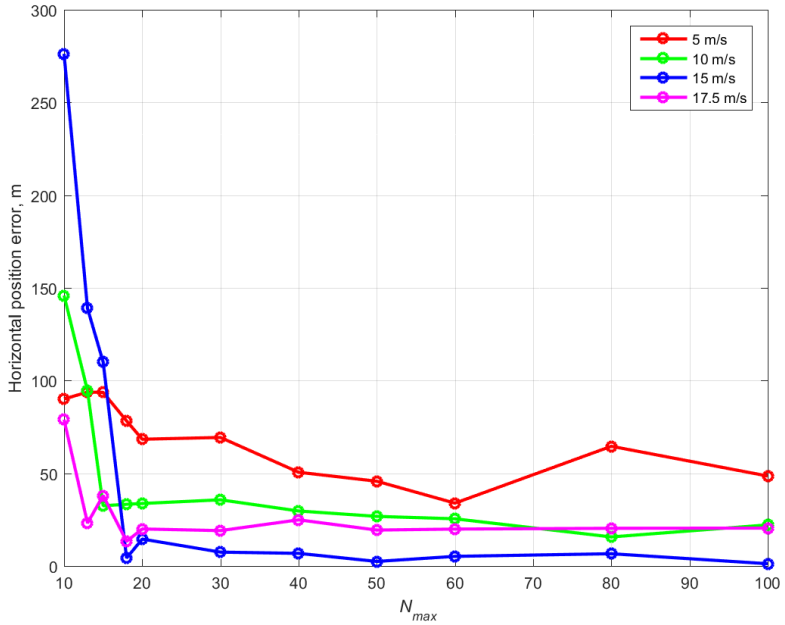

Figure 4.4: Final horizontal position errors with different N_{max} values.

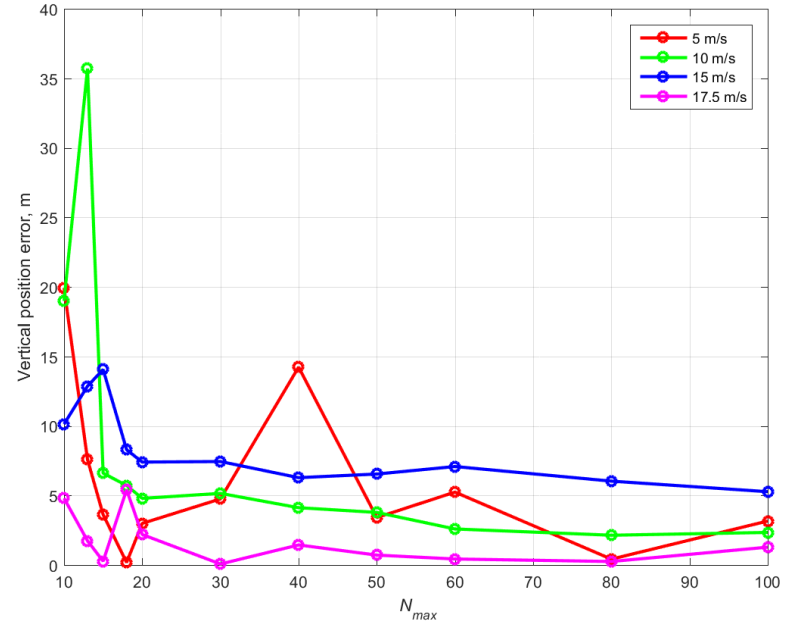

Figure 4.5: Final vertical position errors with different N_{max} values.

4.4 Keyframe-Based MSCKF For Visual Inertial Odometry

In this work, a keyframe-based pose state selection method is proposed in order to reduce the maximum number of pose states stored in the filter while keeping the positioning performance of the MSCKF algorithm. The term keyframe is used in vision aided navigation literature [30, 31, 32, 33] to denote a frame in which the scene observed by the camera is changed more than a specific extent when compared to the last keyframe.

In the proposed keyframe-based MSCKF algorithm, features are tracked in every captured image. However, a pose is added to the state vector of the filter only if the frame is a keyframe. If a frame is not a keyframe, it is used only for tracking purposes. Current frame is classified as a keyframe if and only if,

- A "translation threshold" is exceeded (The platform translated more than a predefined threshold from the position of the last keyframe),
- A "rotation threshold" is exceeded (The platform changed its attitude more than a predefined threshold relative to its attitude at the last keyframe was obtained),
- A "lost feature threshold" is exceeded (Ratio of the lost feature tracks over total active feature tracks in the current frame is more than a predefined threshold).

It is sufficient to treat the current frame as a keyframe when only one of the above criteria is satisfied, since the scene observed by the camera changes when one of the conditions occurs. Evolution of the state vector in standard MSCKF algorithm and proposed keyframe-based MSCKF algorithm on an example are illustrated in Figure 4.6 and Figure 4.7, respectively.

The aim of selecting only keyframes to use in MSCKF algorithm is to infer almost the same amount of information from fewer observations by discarding redundant observations. Required computational resource can be highly reduced while keeping estimation accuracy at the same level.

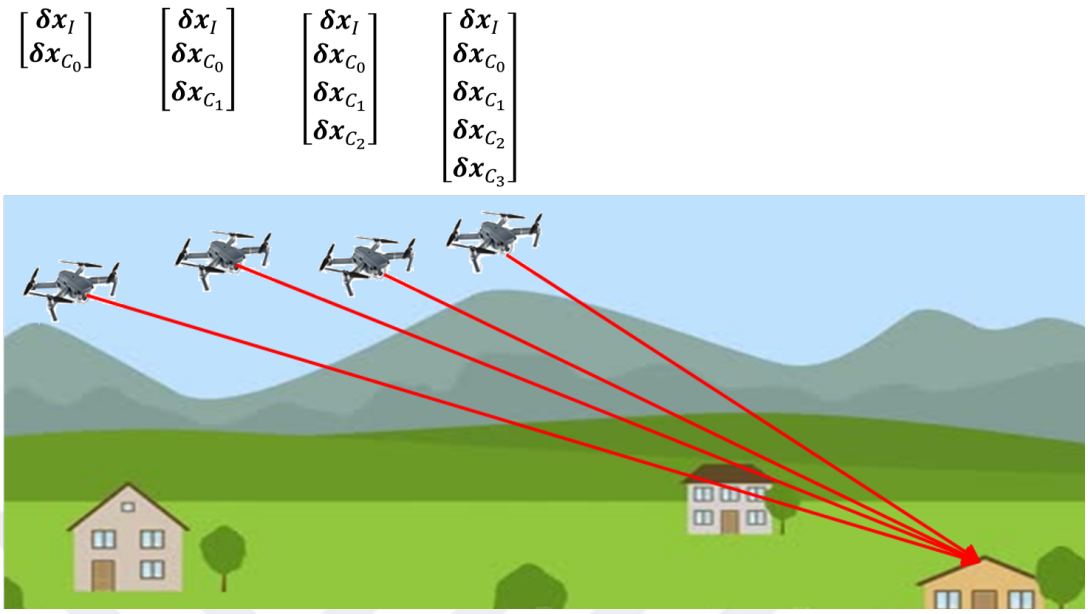

Figure 4.6: Evolution of the overall filter state vector in standard MSCKF algorithm.

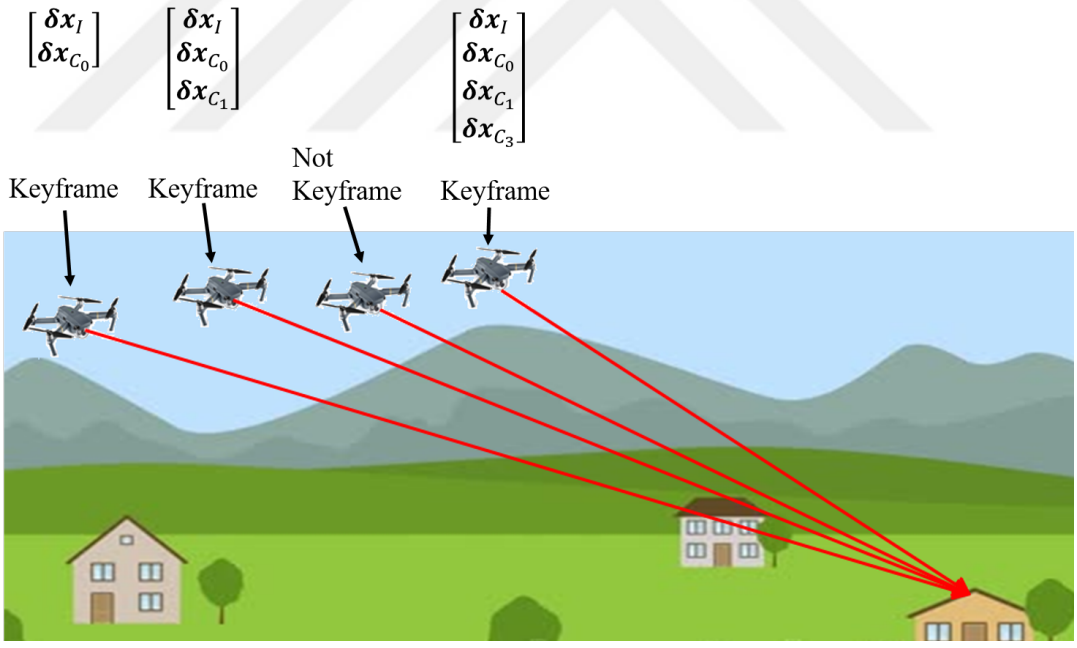

Figure 4.7: Evolution of the overall filter state vector in proposed keyframe-based MSCKF algorithm.

CHAPTER 5

EXPERIMENTAL RESULTS

Experiments have been conducted with the proposed keyframe-based MSCKF algorithm in order to investigate estimation performance. It is observed during experimental work with Fast Flight Dataset that the level of predefined translation threshold has the most influence on positioning accuracy among the criteria discussed in Section 4.4. Thus, it is aimed to observe the effect of varying translation threshold on accuracy of final position estimate. In Figure 5.1 - Figure 5.8 and Table 5.1 - Table 5.2, final position accuracy of the proposed keyframe-based MSCKF algorithm is shared with different translation threshold values for 4 datasets available in Fast Flight Dataset. In these experiments, used keyframe selection translation threshold values are 0.1, 0.3, 0.5, 0.8, 1, 1.2 and 1.5 m. During these experiments, the maximum number of allowed pose states is limited to 10. In order to compare the performance of the proposed algorithm with standard MSCKF algorithm, position errors of standard MSCKF where N_{max} is 10 and 60 are indicated in figures as flat lines.

Table 5.1: Final horizontal position errors of keyframe-based MSCKF for different translation threshold values. N_{max} is set as 10

Translation threshold (m)	5 m/s dataset, (m)	10 m/s dataset, (m)	15 m/s dataset, (m)	17.5 m/s dataset, (m)
0.1	170	57.6	77.33	152
0.3	88.52	32.58	21.46	7.36
0.5	65.12	20.62	10.85	12.46
0.8	71.85	20.64	14.09	10.22
1	47.71	10.22	9.89	12.99
1.2	48.44	18.61	8.78	11.57
1.5	29.05	29.96	6.24	11.36

Table 5.2: Final vertical position errors of keyframe-based MSCKF for different translation threshold values. N_{max} is set as 10

Translation threshold (m)	5 m/s dataset, (m)	10 m/s dataset, (m)	15 m/s dataset, (m)	17.5 m/s dataset, (m)
0.1	15.14	14.9	8.94	16.46
0.3	16.07	6.62	14.58	0.25
0.5	8.24	6.92	7.16	0.44
0.8	10.93	5.49	7.36	2.55
1	8.2	5.3	7.22	2.01
1.2	7.57	5.01	7.78	2.41
1.5	9.4	6.78	7.73	2.58

Results of the experiments showed the effect of proposed keyframe-based MSCKF algorithm when selecting N_{max} as a small value such as 10. Horizontal position errors are 90.05, 145.9, 276.25, and 78.9 meters for 5 m/s, 10 m/s, 15 m/s, and 17.5 m/s datasets respectively when N_{max} is selected as 10 in standard MSCKF configuration. These errors are reduced to 33.89, 25.45, 5.19, and 19.91 meters when N_{max} is selected as 60 in standard MSCKF. Vertical position errors are 19.92, 18.99, 10.13, and 4.8 when N_{max} is 10 and 5.24, 2.59, 7.08, and 0.43 when N_{max} is 60 in standard MSCKF.

Accuracy of horizontal position solution of proposed keyframe-based MSCKF with N_{max} as 10 is similar to the standard MSCKF with N_{max} as 10 when the predefined translation threshold for keyframe decision is selected at lower values. On the other hand, increasing the translation threshold for keyframe decision enhances the performance of keyframe-based MSCKF. Horizontal position errors of keyframe-based MSCKF with N_{max} as 10 become much better than the standard MSCKF when N_{max} is 10 as the translation threshold is selected as more than 1 meters. Horizontal positioning performance of proposed method gets to a similar level of standard MSCKF with N_{max} as 60.

Vertical positioning performance of the proposed method has a similar characteristic with horizontal channel. Vertical position errors of keyframe-based MSCKF with an N_{max} value of 10 are at similar levels of standard MSCKF with an N_{max} value of 10 when the translation threshold is selected lower. Selecting the translation threshold

beyond 1-meter results in better performance than the standard MSCKF. On the other hand, experimental results showed that vertical positioning performance of keyframe-based MSCKF can not reach the performance of standard MSCKF with N_{max} value of 60. Vertical position errors of standard MSCKF when N_{max} is selected as 60 are 5.24, 2.59, 7.08, and 0.43 meters for datasets 5 m/s, 10 m/s, 15 m/s, and 17.5 m/s, respectively. Performance of keyframe-based MSCKF with N_{max} value of 10 is about 7-10, 5-7, 7-8, and 2-3 meters in vertical channel when the best performance is obtained at horizontal positioning solution. These results showed that the proposed method can attain better performance in horizontal channel with fewer N_{max} value while the accuracy of vertical positioning solution degrades slightly.

During experiments, it is observed that choosing a translation threshold above 1.5 meters resulted in inconsistent positioning solutions. In most of the test cases, estimated kinematics of the filter diverged. In addition to that, a consistent solution were not able to be obtained with keyframe-based MSCKF when the N_{max} value is set below 10. In most of these experiments, filter estimate diverged.

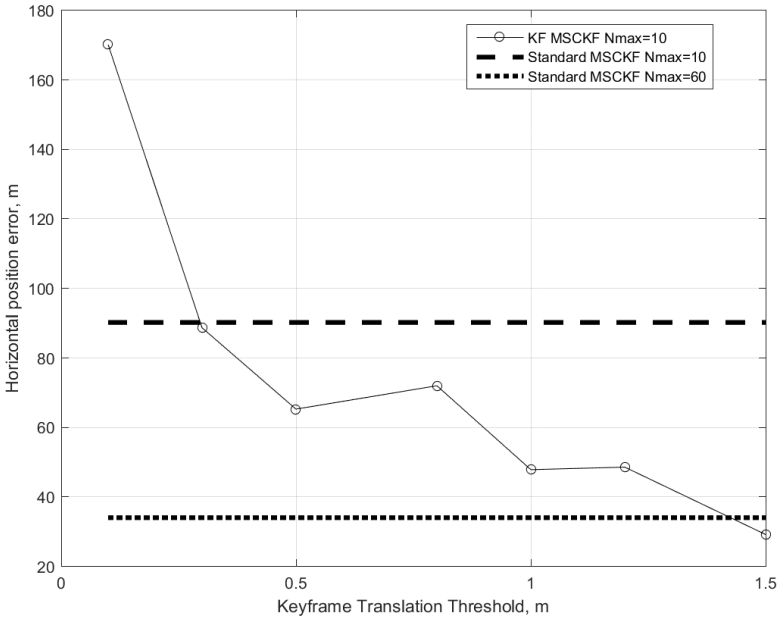

Figure 5.1: Final horizontal position error of keyframe-based MSCKF, 5 m/s dataset.

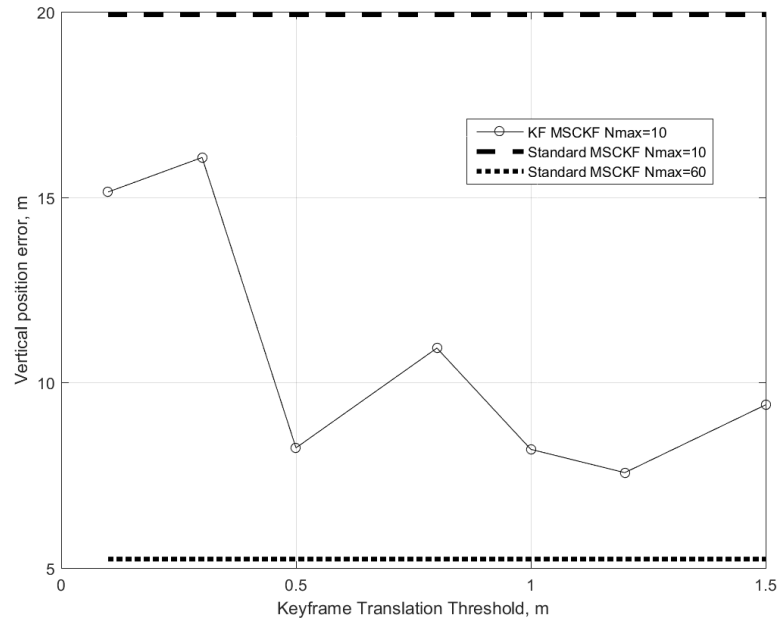

Figure 5.2: Final vertical position error of keyframe-based MSCKF, 5 m/s dataset.

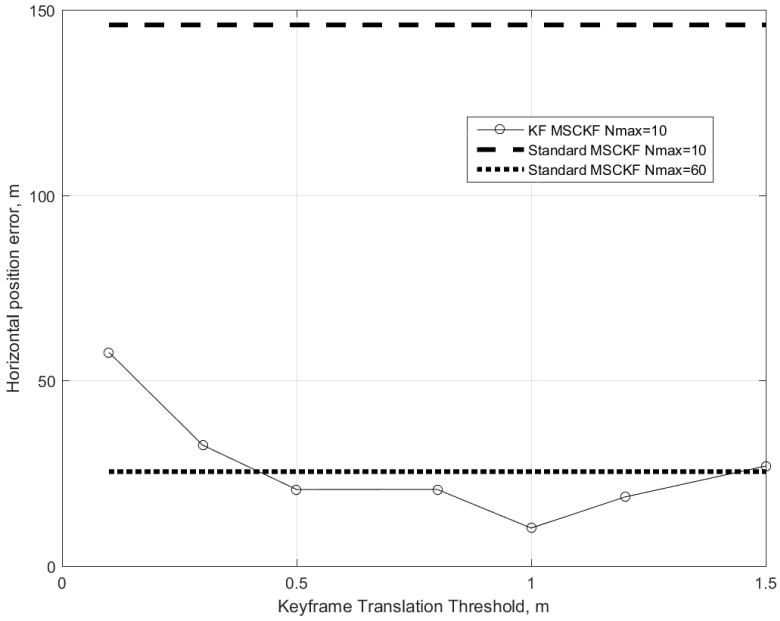

Figure 5.3: Final horizontal position error of keyframe-based MSCKF, 10 m/s dataset.

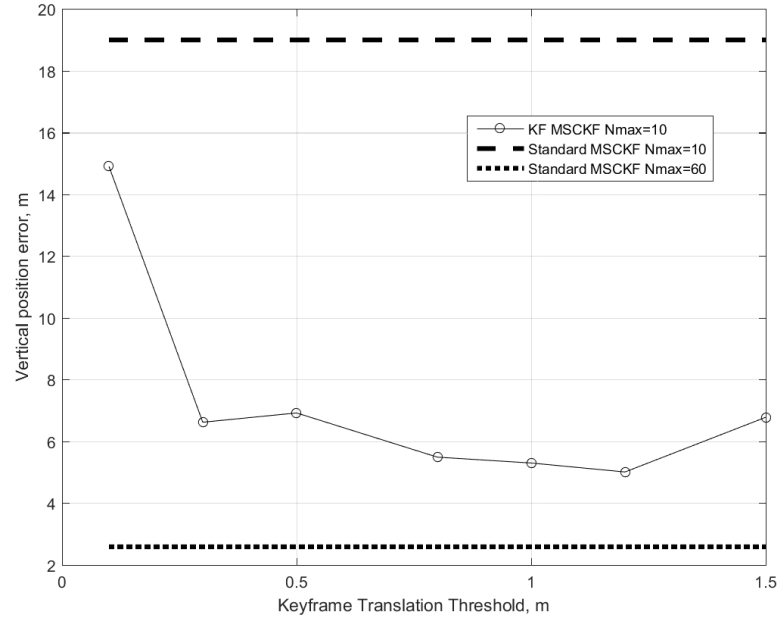

Figure 5.4: Final vertical position error of keyframe-based MSCKF, 10 m/s dataset.

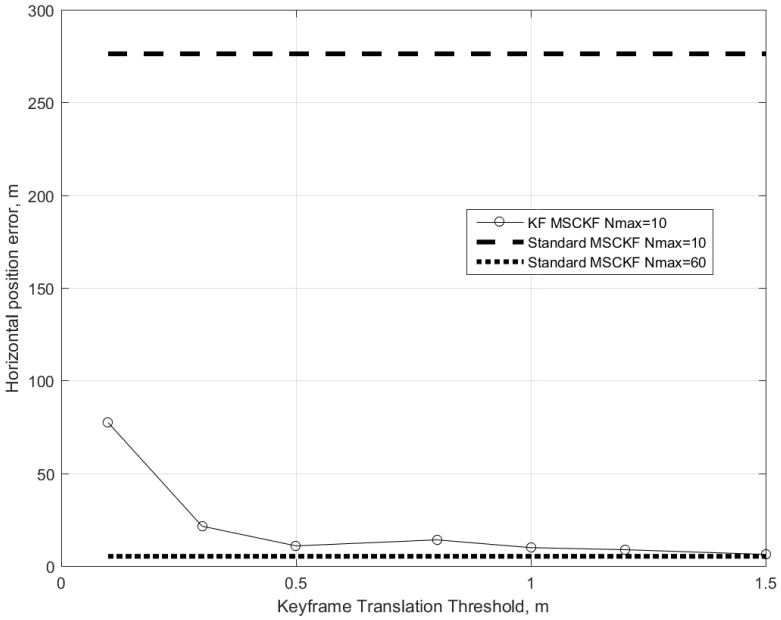

Figure 5.5: Final horizontal position error of keyframe-based MSCKF, 15 m/s dataset.

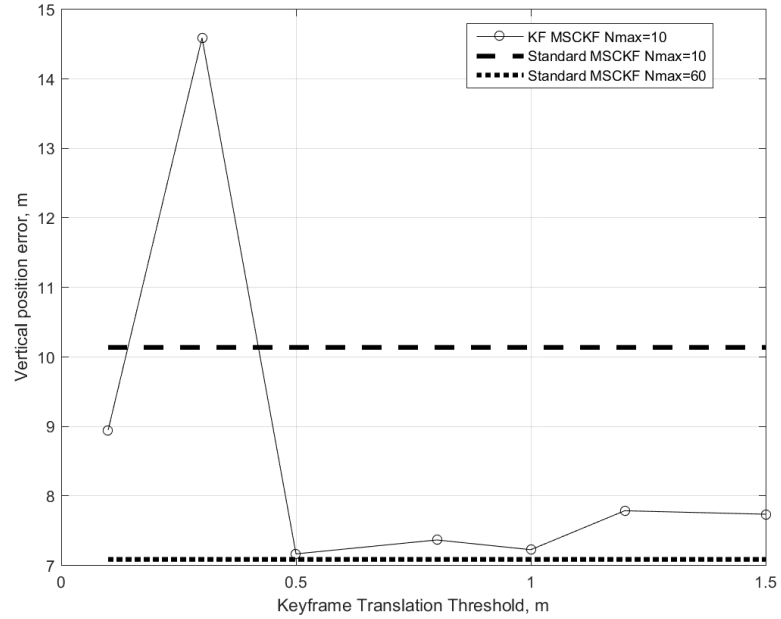

Figure 5.6: Final vertical position error of keyframe-based MSCKF, 15 m/s dataset.

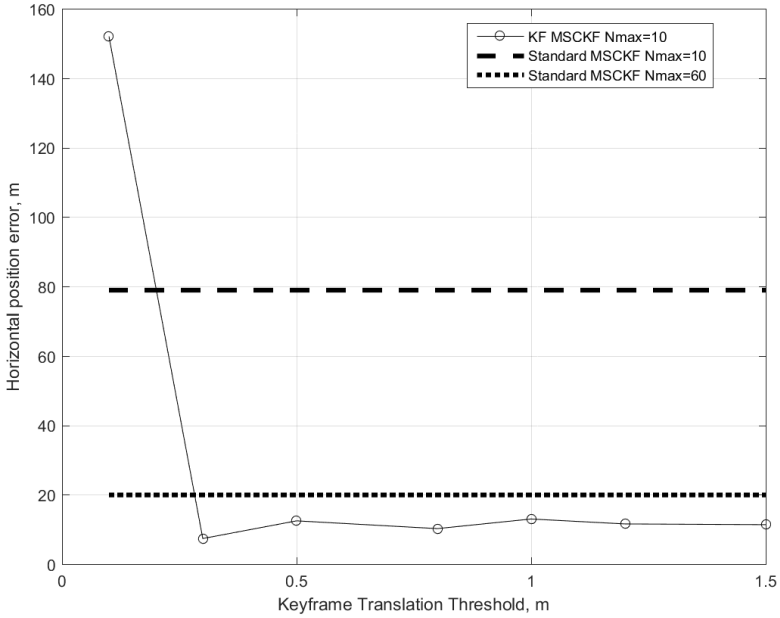

Figure 5.7: Final horizontal position error of keyframe-based MSCKF, 17.5 m/s dataset.

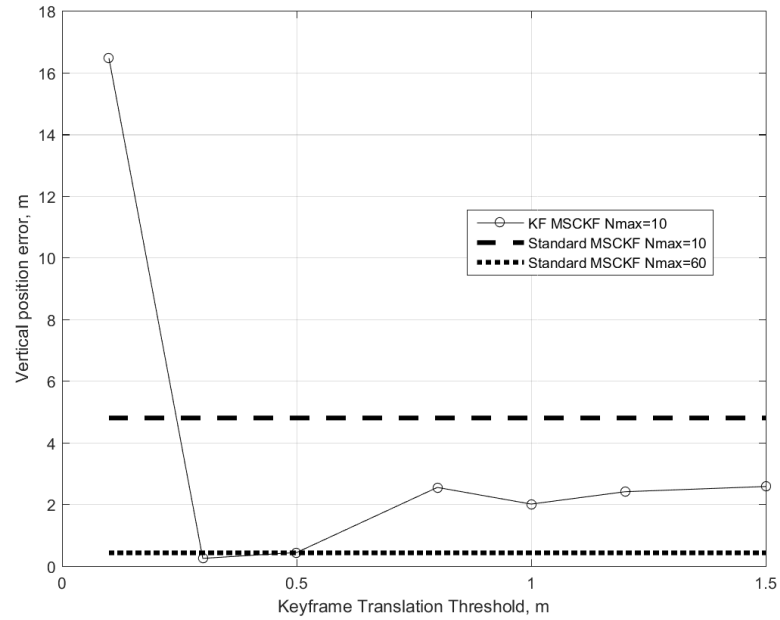

Figure 5.8: Final vertical position error of keyframe-based MSCKF, 17.5 m/s dataset.

CHAPTER 6

CONCLUSION

Autonomous navigation of small and inexpensive platforms is a crucial task for various civilian and military applications. Reducing the required computational resources is a challenging problem in the design of VIO applications of such platforms. In this thesis, we proposed a keyframe-based MSCKF methodology in order to obtain accurate positioning solution with lower computational demand.

First of all, effect of the maximum number of pose states stored at MSCKF is investigated with Fast Flight Dataset. Results of this analysis showed that the accuracy of the estimated position improves as the number of pose states in the state vector increases. However increasing the maximum number of pose states stored may cause problems in real-time operation due to the increase in size of the state vector. In addition to that increased computational demand may cause reduction in the operation time of the platform. Thus, a keyframe-based MSCKF is proposed in this study in order to reduce computational demand while keeping accuracy almost at the same level. This mechanism provides keeping only the pose states that are captured where the scene is changed more than a specific level. Lastly, performance of the proposed methodology is compared against standard MSCKF algorithm with a higher value of allowed maximum number of pose states. This comparison showed that using a keyframe-based pose selection mechanism enables the MSCKF to obtain compatible positioning accuracy with fewer pose states augmented in the overall state vector. These results demonstrate that keyframe-based approach provides accurate and computationally efficient VIO algorithms to real-time autonomous navigation applications on platforms with limited computational resources.

REFERENCES

[1] Stephan Weiss, Markus W Achtelik, Simon Lynen, Michael C Achtelik, Laurent Kneip, Margarita Chli, and Roland Siegwart. Monocular vision for long-term micro aerial vehicle state estimation: A compendium. *Journal of Field Robotics*, 30(5):803–831, 2013.

[2] Daniel Paul Magree. *Monocular vision-aided inertial navigation for unmanned aerial vehicles*. PhD thesis, Georgia Institute of Technology, 2015.

[3] Michael Blösch, Stephan Weiss, Davide Scaramuzza, and Roland Siegwart. Vision based mav navigation in unknown and unstructured environments. In *2010 IEEE International Conference on Robotics and Automation*, pages 21–28. IEEE, 2010.

[4] Bismaya Sahoo. Direct visual-inertial odometry using epipolar constraints for land vehicles. Master's thesis, University of Waterloo, 2018.

[5] Christopher Brunner, Thierry Peynot, and Teresa Vidal-Calleja. Combining multiple sensor modalities for a localisation robust to smoke. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2489–2496. IEEE, 2011.

[6] U Schmidt, T Fiksel, A Kwiatkowski, I Steinbach, B Pradarutti, K Michel, and E Benzi. Autonomous star sensor astro aps: flight experience on alphasat. *CEAS Space Journal*, 7(2):237–246, 2015.

[7] Vittorio Franzese, Pierluigi Di Lizia, and Francesco Topputo. Autonomous optical navigation for lumio mission. In *2018 Space Flight Mechanics Meeting*, page 1977, 2018.

[8] Aboelmagd Noureldin, Tashfeen B Karamat, Mark D Eberts, and Ahmed El-Shafie. Performance enhancement of mems-based ins/gps integration for low-

cost navigation applications. *IEEE Transactions on vehicular technology*, 58(3):1077–1096, 2008.

[9] George T Schmidt and Richard E Phillips. Ins/gps integration architectures. Technical report, MASSACHUSETTS INST OF TECH LEXINGTON MA, 2010.

[10] Yuan Zhuang, Jun Yang, You Li, Longning Qi, and Naser El-Sheimy. Smartphone-based indoor localization with bluetooth low energy beacons. *Sensors*, 16(5):596, 2016.

[11] Antonio Ramón Jiménez Ruiz, Fernando Seco Granja, José Carlos Prieto Honorato, and Jorge I Guevara Rosas. Accurate pedestrian indoor navigation by tightly coupling foot-mounted imu and rfid measurements. *IEEE Transactions on Instrumentation and measurement*, 61(1):178–189, 2011.

[12] Lisa Ran, Sumi Helal, and Steve Moore. Drishti: an integrated indoor/outdoor blind navigation system and service. In *Second IEEE Annual Conference on Pervasive Computing and Communications, 2004. Proceedings of the*, pages 23–30. IEEE, 2004.

[13] Jiraphan Inthiam and Chirdpong Deelertpaiboon. Self-localization and navigation of holonomic mobile robot using omni-directional wheel odometry. In *TENCON 2014-2014 IEEE Region 10 Conference*, pages 1–5. IEEE, 2014.

[14] Frédéric Chenavier and James L Crowley. Position estimation for a mobile robot using vision and odometry. In *Proceedings 1992 IEEE International Conference on Robotics and Automation*, pages 2588–2593. IEEE, 1992.

[15] Jong-Hyuk Kim, Salah Sukkarieh, and Stuart Wishart. Real-time navigation, guidance, and control of a uav using low-cost sensors. In *Field and Service Robotics*, pages 299–309. Springer, 2003.

[16] Haitao Xiang and Lei Tian. Development of a low-cost agricultural remote sensing system based on an autonomous unmanned aerial vehicle (uav). *Biosystems engineering*, 108(2):174–190, 2011.

[17] Jong-Hyuk Kim and Salah Sukkarieh. Airborne simultaneous localisation and map building. In *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, volume 1, pages 406–411. IEEE, 2003.

[18] David Ball, Ben Upcroft, Gordon Wyeth, Peter Corke, Andrew English, Patrick Ross, Tim Patten, Robert Fitch, Salah Sukkarieh, and Andrew Bate. Vision-based obstacle detection and navigation for an agricultural robot. *Journal of field robotics*, 33(8):1107–1130, 2016.

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

[20] Chaomin Luo, Mohan Krishnan, Mark Paulik, Bo Cui, and Xingzhong Zhang. A novel lidar-driven two-level approach for real-time unmanned ground vehicle navigation and map building. In *Intelligent Robots and Computer Vision XXXI: Algorithms and Techniques*, volume 9025, page 902503. International Society for Optics and Photonics, 2014.

[21] Duy Pham and Young Suh. Pedestrian navigation using foot-mounted inertial sensor and lidar. *Sensors*, 16(1):120, 2016.

[22] Kerem Eyice and Onur Çulha. Lrf assisted slam for airborne platforms. In *2018 IEEE/ION Position, Location and Navigation Symposium (PLANS)*, pages 1131–1134. IEEE, 2018.

[23] Miranda Kreković, Ivan Dokmanić, and Martin Vetterli. Echoslam: Simultaneous localization and mapping with acoustic echoes. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 11–15. Ieee, 2016.

[24] Christine Evers, Alastair H Moore, and Patrick A Naylor. Acoustic simultaneous localization and mapping (a-slam) of a moving microphone array and its surrounding speakers. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10. IEEE, 2016.

[25] David Caruso. *Improving Visual-Inertial Navigation Using Stationary Environmental Magnetic Disturbances*. PhD thesis, Université Paris-Saclay, 2018.

[26] Mehmet Erçin Özgeneci and Tolga Çiloğlu. Rotation and acceleration based adaptation for attitude and heading reference system. In *2013 21st Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2013.

[27] Cesar Cadena, Luca Carlone, Henry Carrillo, Yasir Latif, Davide Scaramuzza, José Neira, Ian Reid, and John J Leonard. Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age. *IEEE Transactions on robotics*, 32(6):1309–1332, 2016.

[28] Jeffrey Delmerico and Davide Scaramuzza. A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2502–2509. IEEE, 2018.

[29] Mingyang Li and Anastasios I Mourikis. Vision-aided inertial navigation for resource-constrained systems. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1057–1063. IEEE, 2012.

[30] Rui Wang, Martin Schworer, and Daniel Cremers. Stereo dso: Large-scale direct sparse visual odometry with stereo cameras. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3903–3911, 2017.

[31] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization. In *BMVC*, 2017.

[32] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[33] 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*, 34(3):314–334, 2015.

[34] Raul Mur-Artal and Juan D Tardós. Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE Transactions on Robotics*, 33(5):1255–1262, 2017.

[35] Pompilio Araújo, Rodolfo Miranda, Diedre Carmo, Raul Alves, and Luciano Oliveira. Air-sslam: A visual stereo indoor slam for aerial quadrotors. *IEEE Geoscience and Remote Sensing Letters*, 14(9):1643–1647, 2017.

[36] Andrew Howard. Real-time stereo visual odometry for autonomous ground vehicles. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3946–3952. IEEE, 2008.

[37] Bernd Kitt, Andreas Geiger, and Henning Lategahn. Visual odometry based on stereo image sequences with ransac-based outlier rejection scheme. In *2010 ieee intelligent vehicles symposium*, pages 486–492. IEEE, 2010.

[38] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. Orb-slam: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[39] Koray Çelik and Arun K Somani. Monocular vision slam for indoor aerial vehicles. *Journal of electrical and computer engineering*, 2013:4–1573, 2013.

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

[41] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 15–22. IEEE, 2014.

[42] Vikram Mohanty, Shubh Agrawal, Shaswat Datta, Arna Ghosh, Vishnu Dutt Sharma, and Debashish Chakravarty. Deepvo: A deep learning approach for monocular visual odometry. *arXiv preprint arXiv:1611.06069*, 2016.

[43] Frank Steinbrücker, Jürgen Sturm, and Daniel Cremers. Real-time visual odometry from dense rgb-d images. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 719–722. IEEE, 2011.

[44] Christian Kerl, Jürgen Sturm, and Daniel Cremers. Dense visual slam for rgb-d cameras. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2100–2106. IEEE, 2013.

[45] Richard A Newcombe, Steven J Lovegrove, and Andrew J Davison. Dtam: Dense tracking and mapping in real-time. In *2011 international conference on computer vision*, pages 2320–2327. IEEE, 2011.

[46] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. Remode: Probabilistic, monocular dense reconstruction in real time. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2609–2616. IEEE, 2014.

[47] Alejo Concha and Javier Civera. Dpptam: Dense piecewise planar tracking and mapping from a monocular sequence. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5686–5693. IEEE, 2015.

[48] Raúl Mur-Artal and Juan D Tardós. Probabilistic semi-dense mapping from highly accurate feature-based monocular slam. In *Robotics: Science and Systems*, volume 2015. Rome, 2015.

[49] Jakob Engel, Jurgen Sturm, and Daniel Cremers. Semi-dense visual odometry for a monocular camera. In *Proceedings of the IEEE international conference on computer vision*, pages 1449–1456, 2013.

[50] Thomas Schöps, Jakob Engel, and Daniel Cremers. Semi-dense visual odometry for ar on a smartphone. In *2014 IEEE international symposium on mixed and augmented reality (ISMAR)*, pages 145–150. IEEE, 2014.

[51] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 1–10. IEEE Computer Society, 2007.

[52] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1935–1942. IEEE, 2015.

[53] Salim Sirtkaya, Burak Seymen, and A Aydin Alatan. Loosely coupled kalman filtering for fusion of visual odometry and inertial navigation. In *Proceedings of the 16th International Conference on Information Fusion*, pages 219–226. IEEE, 2013.

[54] Jean-Philippe Tardif, Michael George, Michel Laverne, Alonzo Kelly, and Anthony Stentz. A new approach to vision-aided inertial navigation. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4161–4168. IEEE, 2010.

[55] Markus Achtelik, Michael Achtelik, Stephan Weiss, and Roland Siegwart. Onboard imu and monocular vision based control for mavs in unknown in-and outdoor environments. In *2011 IEEE International Conference on Robotics and Automation*, pages 3056–3063. IEEE, 2011.

[56] Inkyu Sa, Hu He, Van Huynh, and Peter Corke. Monocular vision based autonomous navigation for a cost-effective mav in gps-denied environments. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pages 1355–1360. IEEE, 2013.

[57] 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 *2012 IEEE international conference on robotics and automation*, pages 957–964. IEEE, 2012.

[58] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017.

[59] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE, 2015.

[60] Vladyslav Usenko, Jakob Engel, Jörg Stückler, and Daniel Cremers. Direct

visual-inertial odometry with stereo cameras. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1885–1892. IEEE, 2016.

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

[62] Michael J Veth. *Fusion of imaging and inertial sensors for navigation*. PhD thesis, Graduate School of Engineering, Air Force Institute of Technology, 2006.

[63] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33(1):1–21, 2016.

[64] Simon Lynen, Markus W Achtelik, Stephan Weiss, Margarita Chli, and Roland Siegwart. A robust and modular multi-sensor fusion approach applied to mav navigation. In *2013 IEEE/RSJ international conference on intelligent robots and systems*, pages 3923–3929. IEEE, 2013.

[65] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. *IEEE Transactions on Robotics*, 28(1):61–76, 2011.

[66] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation. Georgia Institute of Technology, 2015.

[67] Yi Lin, Fei Gao, Tong Qin, Wenliang Gao, Tianbo Liu, William Wu, Zhenfei Yang, and Shaojie Shen. Autonomous aerial navigation using monocular visual-inertial fusion. *Journal of Field Robotics*, 35(1):23–51, 2018.

[68] 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*, 3(2):965–972, 2018.

[69] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

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

[72] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[73] Christopher G Harris and JM Pike. 3d positional integration from image sequences. *Image and Vision Computing*, 6(2):87–90, 1988.

[74] Jianbo Shi and Carlo Tomasi. Good features to track. Technical report, Cornell University, 1993.

[75] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[76] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

[77] Bruce D Lucas, Takeo Kanade, et al. An iterative image registration technique with an application to stereo vision. 1981.

[78] Friedrich Fraundorfer and Davide Scaramuzza. Visual odometry: Part ii: Matching, robustness, optimization, and applications. *IEEE Robotics & Automation Magazine*, 19(2):78–90, 2012.

[79] Robert Grover Brown, Patrick YC Hwang, et al. *Introduction to random signals and applied Kalman filtering*, volume 3. Wiley New York, 1992.

[80] Dataset · KumarRobotics/msckf_vio Wiki · GitHub. `https://github.com/KumarRobotics/msckf_vio/wiki/dataset#fast-flight-dataset`. Accessed: 2019-01-19.

[81] VectorNav VN-100 IMU specifications. `https://www.vectornav.com/products/vn-100/specifications`. Accessed: 2019-02-21.

[82] OpenCV SIFT feature detection. https://docs.opencv.org/2.4/modules/nonfree/doc/feature_detection.html. Accessed: 2019-02-12.

[83] KLT feature tracker of MATLAB®. https://www.mathworks.com/help/vision/ref/vision.pointtracker-system-object.html. Accessed: 2019-02-15.

[84] Mingyang Li. *Visual-inertial odometry on resource-constrained systems*. PhD thesis, UC Riverside, 2014.

APPENDIX A

CALCULATION OF DISCRETE PROCESS NOISE COVARIANCE MATRIX

By integrating Equation 3.35, propagation of the state vector in discrete time can be obtained as:

$$\boldsymbol{\delta x}(t_{k+1}) = \boldsymbol{\Phi}(t_k)\boldsymbol{\delta x}(t_k) + \boldsymbol{w}_d(t_k) \tag{A.1}$$

where $\boldsymbol{w}_d(t_k)$ is the discrete process noise vector:

$$\boldsymbol{w}_d(t_k) = \int_{t_k}^{t_{k+1}} exp(\boldsymbol{F}(t_k - \tau)\boldsymbol{G}(\tau)\boldsymbol{w}(\tau)d\tau \tag{A.2}$$

By defining discrete time process noise covariance matrix as:

$$\begin{aligned}\boldsymbol{Q}_d(t_k) &= E[\boldsymbol{w}_d(t_k)\boldsymbol{w}_d(t_k)^T] \\ &= E[\int_{t_k}^{t_{k+1}}\int_{t_k}^{t_{k+1}} exp(\boldsymbol{F}(t_k - \tau)\boldsymbol{G}(\tau)\boldsymbol{w}(\tau)\boldsymbol{w}(\chi)^T\boldsymbol{G}(\chi)^T exp(\boldsymbol{F}(t_k - \chi)^T]\end{aligned} \tag{A.3}$$

By assuming process noise, $\boldsymbol{w}$, is white,

$$E[\boldsymbol{w}(\tau)\boldsymbol{w}(\chi)^T] = \boldsymbol{Q}\delta(\tau - \chi) \tag{A.4}$$

where $\delta(\cdot)$ is Kronecker delta function. Discrete time process noise covariance matrix

becomes:

$$\boldsymbol{Q}_d(t_k) = \int_{t_k}^{t_{k+1}} exp(\boldsymbol{F}(t_k - \tau)\boldsymbol{G}(\tau)\boldsymbol{Q}\boldsymbol{G}(\tau)^T exp(\boldsymbol{F}(t_k - \tau)^T] \tag{A.5}$$

An approximation to the above relation can be obtained by neglecting the time propagation of the system noise over an iteration:

$$\boldsymbol{Q}_d(t_k) \approx \boldsymbol{\Phi}(t_k)\boldsymbol{G}(t_k)\boldsymbol{Q}\boldsymbol{G}(t_k)^T\boldsymbol{\Phi}(t_k)^T\Delta t \tag{A.6}$$

APPENDIX B

DERIVATION OF MEASUREMENT JACOBIAN MATRICES

The relation between measurement of the feature f_j with pose states of C_i and the position of the feature is given in Equations 3.64 - 3.65. Following the chain rule, Jacobian of the measurement with respect to a vector can be calculated:

$$\frac{\partial \boldsymbol{\delta z}_{j,i}}{\partial *} = \frac{\partial \boldsymbol{\delta z}_{j,i}}{\partial \boldsymbol{\delta p}_{f_j}^{c_i}} \frac{\partial \boldsymbol{\delta p}_{f_j}^{c_i}}{\partial *} \tag{B.1}$$

where $\boldsymbol{\delta z}_{j,i} = \tilde{\boldsymbol{z}}_{j,i} - \boldsymbol{z}_{j,i}$, and $\dfrac{\partial \boldsymbol{\delta z}_{j,i}}{\partial \boldsymbol{\delta p}_{f_j}^{c_i}}$ previously used as $\boldsymbol{J}_{j,i}$ in Equations 3.67 - 3.68.

$$\boldsymbol{z}_{j,i} = \frac{1}{p_{f_j,(3)}^{c_i}} \begin{bmatrix} p_{f_j,(1)}^{c_i} \\ p_{f_j,(2)}^{c_i} \end{bmatrix} = \begin{bmatrix} \dfrac{p_{f_j,(1)}^{c_i}}{p_{f_j,(3)}^{c_i}} \\ \\ \dfrac{p_{f_j,(2)}^{c_i}}{p_{f_j,(3)}^{c_i}} \end{bmatrix} \tag{B.2}$$

and

$$\tilde{\boldsymbol{z}}_{j,i} = \begin{bmatrix} \dfrac{\tilde{p}_{f_j,(1)}^{c_i}}{\tilde{p}_{f_j,(3)}^{c_i}} \\ \\ \dfrac{\tilde{p}_{f_j,(2)}^{c_i}}{\tilde{p}_{f_j,(3)}^{c_i}} \end{bmatrix} = \begin{bmatrix} \dfrac{p_{f_j,(1)}^{c_i} + \delta p_{f_j,(1)}^{c_i}}{p_{f_j,(3)}^{c_i} + \delta p_{f_j,(3)}^{c_i}} \\ \\ \dfrac{p_{f_j,(2)}^{c_i} + \delta p_{f_j,(2)}^{c_i}}{p_{f_j,(3)}^{c_i} + \delta p_{f_j,(3)}^{c_i}} \end{bmatrix} \tag{B.3}$$

$$\boldsymbol{\delta z}_{j,i} = \begin{bmatrix} \dfrac{p^{c_i}_{f_j,(1)} + \delta p^{c_i}_{f_j,(1)}}{p^{c_i}_{f_j,(3)} + \delta p^{c_i}_{f_j,(3)}} - \dfrac{p^{c_i}_{f_j,(1)}}{p^{c_i}_{f_j,(3)}} \\ \\ \dfrac{p^{c_i}_{f_j,(2)} + \delta p^{c_i}_{f_j,(2)}}{p^{c_i}_{f_j,(3)} + \delta p^{c_i}_{f_j,(3)}} - \dfrac{p^{c_i}_{f_j,(2)}}{p^{c_i}_{f_j,(3)}} \end{bmatrix}$$

$$= \begin{bmatrix} \dfrac{p^{c_i}_{f_j,(1)} p^{c_i}_{f_j,(3)} + \delta p^{c_i}_{f_j,(1)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(1)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(1)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \\ \\ \dfrac{p^{c_i}_{f_j,(2)} p^{c_i}_{f_j,(3)} + \delta p^{c_i}_{f_j,(2)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(2)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(2)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \end{bmatrix} \tag{B.4}$$

$$= \begin{bmatrix} \dfrac{\delta p^{c_i}_{f_j,(1)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(1)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \\ \\ \dfrac{\delta p^{c_i}_{f_j,(2)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(2)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \end{bmatrix}$$

$$\frac{\partial \boldsymbol{\delta z}_{j,i}}{\partial \delta p^{c_i}_{f_j,(1)}} = \begin{bmatrix} \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(1)}} \dfrac{\delta p^{c_i}_{f_j,(1)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(1)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \\ \\ \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(1)}} \dfrac{\delta p^{c_i}_{f_j,(2)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(2)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \end{bmatrix} \approx \begin{bmatrix} \dfrac{1}{p^{c_i}_{f_j,(3)}} \\ 0 \end{bmatrix} \tag{B.5}$$

$$\frac{\partial \boldsymbol{\delta z}_{j,i}}{\partial \delta p^{c_i}_{f_j,(2)}} = \begin{bmatrix} \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(2)}} \dfrac{\delta p^{c_i}_{f_j,(1)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(1)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \\ \\ \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(2)}} \dfrac{\delta p^{c_i}_{f_j,(2)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(2)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \end{bmatrix} \approx \begin{bmatrix} 0 \\ \dfrac{1}{p^{c_i}_{f_j,(3)}} \end{bmatrix} \tag{B.6}$$

$$
\begin{aligned}
\frac{\partial \boldsymbol{\delta z}_{j,i}}{\partial \delta p^{c_i}_{f_j,(3)}} &= \begin{bmatrix} \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(3)}} \dfrac{\delta p^{c_i}_{f_j,(1)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(1)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \\ \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(3)}} \dfrac{\delta p^{c_i}_{f_j,(2)} p^{c_i}_{f_j,(3)} - p^{c_i}_{f_j,(2)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \end{bmatrix} \\
&= \begin{bmatrix} \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(3)}} \dfrac{-p^{c_i}_{f_j,(1)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \\ \dfrac{\partial}{\partial \delta p^{c_i}_{f_j,(3)}} \dfrac{-p^{c_i}_{f_j,(2)} \delta p^{c_i}_{f_j,(3)}}{{p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}} \end{bmatrix} \\
&= \begin{bmatrix} \dfrac{-p^{c_i}_{f_j,(1)} ({p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}) + p^{c_i}_{f_j,(1)} \delta p^{c_i}_{f_j,(3)}}{({p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)})^2} \\ \dfrac{-p^{c_i}_{f_j,(2)} ({p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)}) + p^{c_i}_{f_j,(2)} \delta p^{c_i}_{f_j,(3)}}{({p^{c_i}_{f_j,(3)}}^2 + \delta p^{c_i}_{f_j,(3)})^2} \end{bmatrix} \\
&\approx \begin{bmatrix} \dfrac{-p^{c_i}_{f_j,(1)} {p^{c_i}_{f_j,(3)}}^2}{{p^{c_i}_{f_j,(3)}}^4} \\ \dfrac{-p^{c_i}_{f_j,(2)} {p^{c_i}_{f_j,(3)}}^2}{{p^{c_i}_{f_j,(3)}}^4} \end{bmatrix} \\
&\approx \begin{bmatrix} \dfrac{-p^{c_i}_{f_j,(1)}}{{p^{c_i}_{f_j,(3)}}^2} \\ \dfrac{-p^{c_i}_{f_j,(2)}}{{p^{c_i}_{f_j,(3)}}^2} \end{bmatrix}
\end{aligned}
\tag{B.7}
$$

Thus,

$$\boldsymbol{J}_{j,i} = \begin{bmatrix} \frac{1}{p^{c_i}_{f_j,(3)}} & 0 & -\frac{p^{c_i}_{f_j,(1)}}{{p^{c_i}_{f_j,(3)}}^2} \\ 0 & \frac{1}{p^{c_i}_{f_j,(3)}} & -\frac{p^{c_i}_{f_j,(2)}}{{p^{c_i}_{f_j,(3)}}^2} \end{bmatrix} = \frac{1}{p^{c_i}_{f_j,(3)}} \begin{bmatrix} 1 & 0 & -\frac{p^{c_i}_{f_j,(1)}}{p^{c_i}_{f_j,(3)}} \\ 0 & 1 & -\frac{p^{c_i}_{f_j,(2)}}{p^{c_i}_{f_j,(3)}} \end{bmatrix} \tag{B.8}$$

is obtained.

After obtaining $\boldsymbol{J}_{j,i}$, $\frac{\partial \boldsymbol{\delta p}^{c_i}_{f_j}}{\partial *}$ terms have to be calculated for every element in the state vector. It can be seen from the Equation 3.65 that the error on the position of feature f_j resolved in the i^{th} camera coordinate systems is only related to the errors on the i^{th} pose state, $\boldsymbol{\delta x}_{C_i}$, and the error on the feature position estimate resolved in the local cartesian frame, $\boldsymbol{\delta p}^n_{f_j}$. Thus, all blocks except i^{th} pose state in the measurement Jacobian matrix with respect to the state vector are zero matrices.

Jacobian with respect to the error on the position of i^{th} pose state can be calculated as:

$$\begin{aligned} \boldsymbol{\delta p}^{c_i}_{f_j} &= \tilde{\boldsymbol{p}}^{c_i}_{f_j} - \boldsymbol{p}^{c_i}_{f_j} \\ &= \boldsymbol{C}^{c_i}_n(\boldsymbol{p}^n_{f_j} - \tilde{\boldsymbol{p}}^n_{c_i}) - \boldsymbol{C}^{c_i}_n(\boldsymbol{p}^n_{f_j} - \boldsymbol{p}^n_{c_i}) \\ &= \boldsymbol{C}^{c_i}_n(\boldsymbol{p}^n_{f_j} - \boldsymbol{p}^n_{c_i} - \boldsymbol{\delta p}^n_{c_i}) \\ &= -\boldsymbol{C}^{c_i}_n \boldsymbol{\delta p}^n_{c_i} \end{aligned} \tag{B.9}$$

Using above relation:

$$\frac{\partial \boldsymbol{\delta p}^{c_i}_{f_j}}{\partial \boldsymbol{\delta p}^n_{c_i}} = -\boldsymbol{C}^{c_i}_n \tag{B.10}$$

can be obtained.

Equation 3.65 has to be rewritten in order to obtain Jacobian with respect to the attitude error:

$$\boldsymbol{p}_{f_j}^{c_i} = \boldsymbol{C}_n^{c_i}(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) = \boldsymbol{C}_b^{c_i}\boldsymbol{C}_n^b(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) \tag{B.11}$$

The relation between attitude error and the erroneous direction cosine matrix is discussed in [70] and used in derivation of system dynamics matrix given in Equation 3.51 as:

$$\tilde{\boldsymbol{C}}_n^b = \boldsymbol{C}_n^b(\boldsymbol{I} - [\boldsymbol{\delta\psi}]_\times) \tag{B.12}$$

where $[\boldsymbol{\delta\psi}]_\times$ represents skew-symmetric matrix of the attitude error vector. For an attitude error vector of $\boldsymbol{\psi} = \begin{bmatrix}\delta\psi_{(1)} & \delta\psi_{(2)} & \delta\psi_{(3)}\end{bmatrix}^T$, skew-symmetric matrix becomes:

$$[\boldsymbol{\delta\psi}]_\times = \begin{bmatrix} 0 & -\delta\psi_{(3)} & \delta\psi_{(2)} \\ \delta\psi_{(3)} & 0 & -\delta\psi_{(1)} \\ -\delta\psi_{(2)} & \delta\psi_{(1)} & 0 \end{bmatrix} \tag{B.13}$$

Thus, the measurement Jacobian with respect to the attitude error of i^{th} pose state can be derived as:

$$\begin{aligned}
\boldsymbol{\delta p}_{f_j}^{c_i} &= \tilde{\boldsymbol{p}}_{f_j}^{c_i} - \boldsymbol{p}_{f_j}^{c_i} \\
&= \boldsymbol{C}_b^{c_i}\tilde{\boldsymbol{C}}_n^b(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) - \boldsymbol{C}_b^{c_i}\boldsymbol{C}_n^b(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) \\
&= \boldsymbol{C}_b^{c_i}\boldsymbol{C}_n^b(\boldsymbol{I} - [\boldsymbol{\delta\psi}]_\times)(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) - \boldsymbol{C}_b^{c_i}\boldsymbol{C}_n^b(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) \\
&= \boldsymbol{C}_n^{c_i}(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) - \boldsymbol{C}_n^{c_i}[\boldsymbol{\delta\psi}]_\times(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) - \boldsymbol{C}_n^{c_i}(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) \\
&= -\boldsymbol{C}_n^{c_i}[\boldsymbol{\delta\psi}]_\times(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n)
\end{aligned} \tag{B.14}$$

Using the above relation for two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$

$$[\boldsymbol{a}]_\times\boldsymbol{b} = -[\boldsymbol{b}]_\times\boldsymbol{a} \tag{B.15}$$

Equation B.14 becomes:

$$\boldsymbol{\delta p}_{f_j}^{c_i} = \boldsymbol{C}_n^{c_i}[(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n)]_\times \boldsymbol{\delta \psi} \tag{B.16}$$

Hence, the measurement Jacobian with respect to the attitude error can be obtained as:

$$\frac{\partial \boldsymbol{\delta p}_{f_j}^{c_i}}{\partial \boldsymbol{\delta \psi}} = \boldsymbol{C}_n^{c_i}[(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n)]_\times \tag{B.17}$$

Lastly, the Jacobian of the measurement with respect to the feature position error given in 3.68 can be derived as:

$$\begin{aligned} \boldsymbol{\delta p}_{f_j}^{c_i} &= \tilde{\boldsymbol{p}}_{f_j}^{c_i} - \boldsymbol{p}_{f_j}^{c_i} \\ &= \boldsymbol{C}_n^{c_i}(\tilde{\boldsymbol{p}}_{f_j}^n - \boldsymbol{p}_{c_i}^n) - \boldsymbol{C}_n^{c_i}(\boldsymbol{p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) \\ &= \boldsymbol{C}_n^{c_i}(\boldsymbol{p}_{f_j}^n + \boldsymbol{\delta p}_{f_j}^n - \boldsymbol{p}_{c_i}^n) \\ &= \boldsymbol{C}_n^{c_i} \boldsymbol{\delta p}_{f_j}^n \end{aligned} \tag{B.18}$$

As a result, the Jacobian matrix can be obtained as:

$$\frac{\partial \boldsymbol{\delta p}_{f_j}^{c_i}}{\partial \boldsymbol{\delta p}_{f_j}^n} = \boldsymbol{C}_n^{c_i} \tag{B.19}$$