MASTER THESIS

Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Engineering at the University of Applied Sciences Technikum Wien - Degree Program Mechatronics/Robotics

Pose Estimation using a Stereo-Photometric Multi-State Constraint Kalman Filter

By: Raphael Maenle, BSc

Student Number: 1710331010

Supervisors: Dr. Vinzenz Sattinger
Dr. Nicolas Thorstensen

Vienna, September 2, 2019

FH University of Applied Sciences
TECHNIKUM WIEN

Declaration

“As author and creator of this work to hand, I confirm with my signature knowledge of the relevant copyright regulations governed by higher education acts (see Urheberrechtsgesetz /Austrian copyright law as amended as well as the Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I hereby declare that I completed the present work independently and that any ideas, whether written by others or by myself, have been fully sourced and referenced. I am aware of any consequences I may face on the part of the degree program director if there should be evidence of missing autonomy and independence or evidence of any intent to fraudulently achieve a pass mark for this work (see Statute on Studies Act Provisions / Examination Regulations of the UAS Technikum Wien as amended).

I further declare that up to this date I have not published the work to hand nor have I presented it to another examination board in the same or similar form. I affirm that the version submitted matches the version in the upload tool.“

Vienna, September 2, 2019 Signature

Kurzfassung

Mobile Akteure, vom Menschen über autonome Fahrzeuge bis hin zu Augmented-Reality (AR) Geräten, wären ohne eine näherungsweise Kenntnis ihrer aktuellen Position nicht funktionsfähig. Die präzisesten Algorithmen verwenden zur Positionsabschätzung eine Vielzahl an Sensoren und benötigen eine hohe Rechenleistung. Kleine mobile Roboter und AR Geräte sind jedoch gewichtsbeschränkt, arbeiten regelmäßig in Umgebungen, in denen keine externen Sensoren verwendet werden können und bieten nur begrenzte Rechenressourcen.

'Visual-inertial odometry' (VIO) Algorithmen bieten eine Lösung für solche limitierten Geräte, indem sie nur Bewegungs*änderungen* abschätzen. Die Algorithmen fusionieren inertiale Messungen mit Kamerainformationen und sind damit weiters ausschließlich von Sensorik am Gerät abhängig. Der *Stereokamera*-basierte 'Multi-State Constraint Kalman Filter' (MSCKF) ist ein solcher odometrischer Algorithmus. Dieser wurde äußerst ressourcenschonend konzipiert und liefert gleichzeitig eine mit anderen Implementierungen vergleichbare Positionsschätzung. Der stereo MSCKF verwendet die Bewegung von Merkmalspunkten im Raum zur Korrektur der inertialen Bewegungsabschätzung. Eine Variation des MSCKF extrahiert aus Bildern einer *Monokamera* die photometrische Information mehrerer Pixelraster.

Auf Basis dieser photometrischen Implementierung des MSCKF präsentiert diese Arbeit die Herleitung und Implementierung eines stereokamera-basierten photometrischen 'Multi-State Constraint Kalman Filters'. Im ersten Schritt wird die Schätzung der Merkmalsposition im Raum durch die Messung im ersten Kamerabild - dem Ankerbild - eingeschränkt. Auf Basis dieser Formulierung wird die rasterbasierte stereo-photometrische Erweiterung konstruiert. Als Grundlage der Implementierungen wird der Open Source 'Multi-State Constraint Kalman Filter' von Kumar Robotics verwendet. Die Ankerbild- sowie die photometrische Erweiterung werden separat evaluiert. Mit je 14,5 Stunden Messaufzeichnungen werden die Implementierungen anhand eines Open Source Datensatzes umfassend bewertet. Der Ankerbild-basierte Filter weist im Vergleich zum stereo MSCKF eine Reduzierung der CPU-Last um 2,2% auf, bei einem Anstieg des relativen quadratischen Mittelwertfehlers von 0,19% auf 0,29%. Der stereo-photometrische Ansatz zeigt bei den meisten Aufnahmen eine mangelnde Robustheit, da er auf eine qualitativ hochwertige Merkmalsauswahl angewiesen ist und eine geringe Fehlertoleranz bei der Positionsabschätzung aufweist. Die Implementierung liefert einen relativen quadratischen Mittelwertfehler von 1,1% beim Betrieb innerhalb der Fehlertoleranzen.

Schlagworte: VIO, MSCKF, Odometrie, Stereo, Kalman Filter, Visual-Inertial SLAM

Abstract

Position estimation is a fundamental component of an agent's autonomy. Any mobile actor, ranging from humans, over autonomous cars to augmented reality devices, would cease to function without approximate knowledge of its current position. Therefore, extensive research has been devoted to position estimation algorithms. High precision estimators use an array of sensors and require a large amount of computational resources. But small mobile devices and robots are severely payload-limited, regularly operate in environments where external sensors are not feasible and offer only limited computing power.

Visual-inertial odometry is a class of algorithms commonly used when faced with these restrictions. Inertial information is fused with the data of one or multiple cameras making the algorithms independent of external sensors. By only estimating the *relative* change of the vehicle per time-step, the algorithms lose some of their accuracy but also greatly reduce the computational overhead. One of the most resource-friendly algorithms in this class is the *stereo*-camera based Multi-State Constraint Kalman Filter, which maintains competitive estimation results. This filter uses feature points to correct the predicted movement of the inertial sensor. A variation on a *mono*-camera based Multi-State Constraint Kalman Filter instead uses pixel-patches of photometric measurements as a basis for the cost function.

This thesis presents the derivation and implementation of a stereo-camera based photometric Multi-State Constraint Kalman Filter. Additionally, the feature position estimation of the stereo Multi-State Constraint Kalman Filter is reformulated, constraining the feature position by the first camera measurement - the anchor frame. Both implementations are based on the open-source implementation of the Multi-State Constraint Kalman Filter pipeline by Kumar Robotics. Both implementations are extensively evaluated using 14.5 hours of sensor recordings from open-source datasets. The anchor-frame based filter shows a 2.2% reduction in CPU load with a relative root mean square error (RMSE) increase from 0.19% to 0.29%. The stereo-photometric approach shows a lack of robustness in most recordings, as it is dependent on high-quality feature selection and has a small error tolerance. The implementation returns an accuracy of 1.1% relative RMSE when operating within the error tolerances, showing promise for increased robustness in future implementations.

Keywords: VIO, MSCKF, Odometry, Stereo, Kalman Filter, Visual-Inertial SLAM

Acknowledgements

This thesis would not have been possible without the work and help of a multitude of people. I would like to thank my first thesis advisor Dr. Vinzenz Sattinger, who guided me in the right direction when writing this thesis, while still allowing it to be my own work. I would further like to thank my second thesis advisor Dr. Nicolas Thorstensen, whose knowledge on the topic was a driving force in selecting the specific subject matter for this work. He accompanied me through the entire theoretical and practical sections of this thesis and I got to learn a lot from him. I would like to express my gratitude to both my parents Andrea and Kevin Maenle, who contributed a lot of time and effort proof-reading, as well as making my writing more uniform and easier to understand. Thank you to the colleagues at work and university, for their valuable feedback on the thesis and for helping me understand various ideas and concepts. Finally, I would like to thank my partner, my friends and my family, for their encouragement and support throughout the entirety of my years of study. I would not have gotten to this point without them.

Thank you.

Contents

1 Introduction

Any autonomous robot - a robotic vacuum cleaner, a self-driving car or a delivery quad-copter - benefits from knowing its current position. Location estimation in a known or unknown environment, allows a mobile robot to make a deliberate choice of movements and operations concerning its surroundings. The robotic vacuum cleaner can clean the house more efficiently and both the autonomous car and the quad-copter may navigate safely to their goal. To become a truly autonomous agent, any such drone needs answers to the three questions regarding its surroundings:

- What is the structure of the environment I am in?
- Where am I with respect to this environment?
- How do I achieve the desired position in this environment?

In a tightly controlled environment, it is possible to come close to having exact answers to all three of these questions. Amazon warehouse robots [1], for example, use global markers to constantly correct their position estimation. The unchanging warehouse surroundings can be represented in a precise map. Using this exact global position information and description of the environment, the robots can accurately transport packages through the hangar.

Contrary to this near-optimal example, many practical applications neither have a map of the environment nor is there a constant source of accurate global position information. This is true for the previous examples of the autonomous car and the vacuum cleaner, but also autonomous filming quad-copters (eg. Skydio [2]), mobile industrial robots (MiR [3], OTTO [4]) and augmented reality devices [5, 6]. In addition to this lack of information about the environment, the maximum computational capacity is severely limited. This is the case especially for smaller mobile robots such as the robotic vacuum cleaner or quad-copters, where a decreased payload capacity and power limitations are the reason for these boundaries.

Even without a preexisting map, a robot's position can be estimated. These algorithms are either based on simultaneous localization and mapping (SLAM) or pure odometry - the first of which generates a map while moving around and the second uses only the most recent information to estimate its ego-motion during the respective time-frame [7]. Examples for both algorithms are further detailed in the related literature chapter 3. Odometry makes sense for applications with limited available computing power, hard real-time constraints or where the environment is in constant flux, making any effort of mapping futile [7].

The initially presented exemplary robots must be able to operate without any *external* sensors or markers. Otherwise, it would limit the robot's ability to work in an unaccommodated environment. Even when an autonomous car has a global positioning system (GPS) signal, the received position estimation is not accurate enough to navigate a car based solely on this information [8]. Various research focuses on merging this GPS information with other sensors to increase the combined accuracy [9, 10]. Indoor operations have no access to any global information and need to rely solely on the sensors mounted on the vehicle; the Roomba 900 [11] vacuum robot, for example, uses an onboard camera without the need of any external markers. Position estimation of ground-based drones can sometimes be reduced to a two-dimensional problem [12], contrary to aerial vehicles such as quad-copters. While wheeled mobile robots such as the MiR 100 [3] or the Xiaomi Roborock [13] use 2D laser scanners, the sensors used for three-dimensional estimation must return suitable three-dimensional information. Sensors delivering such suitable information for mapping or odometry in 3D space include 3D laser scanners [14], depth cameras [15], time-of-flight sensors [16] and even event cameras [17]. The respective papers present either a corresponding SLAM or odometry implementation. Another common sensor is the regular RGB or monochrome camera (see chapter 3.3).

Vision-based algorithms generally have a computational *disadvantage* compared to other position estimation algorithms, as the camera's raw information volume tends to be large. This is a main drawback of visual position estimators. This class of algorithms base their estimation solely on onboard sensors - thus they have a significantly broader use-case compared to systems with a reliance on external information such as GPS or markers. Further, both cameras and inertial measurement units are generally cheaper than the previously mentioned sensors. Their comparatively high estimation accuracy adds another compelling argument to find a computationally efficient solution based on visual information.

Visual-inertial odometry (VIO) is a growing set of algorithms which estimate ego-motion using only visual and inertial information. At the expense of accuracy, VIOs are computationally more efficient compared to other vision-based estimators. Self-contained data sources and deployability on low-powered devices make these visual-inertial odometry algorithms generally viable for mobile robotics and explicitly viable in the context of micro-areal vehicles (MAVs), which are notoriously unfit for heavy calculations due to their payload constraint. Thus, a sizable amount of research has been invested in the field of low-powered VIO. Chapter 3 will go into further detail on vision estimators and the advantages of odometry. The Multi-State Constraint Kalman Filter (MSCKF) VIO algorithm is both among the most computationally efficient in its class and also one of the most robust VIO pipelines to date, as detailed in chapter 3.4. It uses data from an inertial measurement unit (IMU) to estimate its current position and the movement of tracked features in the past few camera frames to correct this estimation. Its derivation and expansion will be the focus of this thesis.

1.1 Motivation

The proposition of the company D-Aria [18] is to use an autonomous quad-copter to fully automate *stock-taking* in warehouses. Automating this process could save a company multiple man-hours of labor, and would allow for continuous inventory evaluation throughout the year. It, therefore, represents a desirable innovation for customers. To allow the robot to operate in already existing warehouses, this indoor application must be independent of external information. Neither external sensors or markers are available in the warehouse, nor is there access to a preexisting map.

Limited computational resources, a payload capacity constraining the possible selection of sensors and a lack of external sensors are therefore the constraints for the necessary pose estimation algorithm on the quad-copter. These constraints force the estimator to be a part of the toughest class of localization algorithms previously discussed.

VIO algorithm research aims to enhance the algorithm's accuracy, while at the same time decreasing the necessary calculation power. These conflicting interests allow for progress in both directions - always leveraging the advancements of accuracy against the sacrificed computing efficiency and vice versa. The research regarding VIO using *monocular* cameras is very expansive. Recent implementations show, that stereo-camera VIOs behave more robustly in test-scenarios. Photometry based visual odometry improves its accuracy by extracting more detailed information when abstracting the recorded images. This makes the comparison between frames more granular. There currently exists no implementation combining these approaches inside the highly efficient Multi-State Constraint Kalman Filter (MSCKF) VIO. Expanding the MSCKF to be more robust, while minimizing loss of efficiency, could ultimately allow the quad-copter of D-Aria to operate autonomously in any setting.

1.2 Project Goal

The goal of this project is to utilize both the improvements achieved by a stereo-camera formulation and the increased accuracy of the photometric approach in the Multi-State Constraint Kalman Filter VIO framework. The stereo-camera MSCKF implementation, therefore, is to be expanded by a photometric residual update-step, minimizing the erroneous movements of pixel-patches in both camera frames around the features. A complete derivation of this stereo-photometric MSCKF's description, as well as a functioning implementation is the desired output. The implementation focuses on correct algorithm design and not resource optimization.

1.3 Project Output

The stereo-photometric expansion for the stereo MSCKF is implemented in two steps. A slightly more resource-friendly anchor-frame based position estimator is designed first, which uses additional constraints to reduce calculation load. Based on this formulation, the photometric stereo-camera MSCKF is derived. The approach uses photometric patches from both cameras and minimizes the covariance based on the error between the pixel patch in the anchor image and the pixel measurement in the expected patch position.

Both formulations are evaluated separately in their absolute and relative accuracy against ground truth information using a state-of-the-art evaluation data-set. These novel update implementations are then compared to the accuracy of the stereo MSCKF implementation, with the anchor-frame based formulation registering a slightly lower CPU footprint and a reduction of accuracy of nearly 50%. The photometric implementation shows unstable behavior in most data-sets and is very dependent on a tightly controlled feature selection, as the error tolerance margin is much slimmer compared to the feature-based approaches. The stereo MSCKF remains the more robust MSCKF design and for the time being the most viable algorithm concerning the use-case of the company D-Aria.

1.4 Structure

This section concludes the introduction. With this chapter, a context for use-cases of visual-inertial odometry in the real world has been created. The project goal, the implementation of an algorithm extending an existing VIO [19] approach, has been presented as well. The project results have been summarized. Chapter 2 introduces the notation used throughout the thesis, as well as a summary of some major concepts upon which this thesis is based. This includes the Extended Kalman Filter and the Error-state Kalman Filter as well as the basics of quaternion mathematics. Refer to the appendix for more details regarding these topics in addition to inertial measurement units and camera models. Chapter 3 expands on different position estimation approaches and concepts and gives a dissection of vision-based odometry approaches. The presented algorithms are compared based on CPU usage and accuracy.

Chapter 4 presents the MSCKF framework, followed by the stereo-camera expansion. In chapter 5 the feature estimation description using the anchor-frame is derived, designing the MSCKF update steps accordingly. This description is expanded into the stereo-photometric approach in chapter 6, where the derivation and underlying assumptions are detailed. The evaluation in chapter 7 details the results of extensive tests, comparing the results of both the anchor-frame and the stereo-photometric update design to the stereo MSCKF. The following discussion, chapter 8, analyzes these results and provides some explanation to the observations. This chapter concludes with suggestions for future implementations. The final chapter 9 summarizes the work done in this thesis.

2 Notation and Fundamental Concepts

This chapter defines the general notation used within this thesis (section 2.1, 2.2 and 2.3). The Error-state Kalman Filter is described in section 2.4). For a more exhaustive explanation on the topics, please refer to the appendix.
An introduction to stochastic state estimation using the Kalman Filter and the Extended Kalman Filter can be found in appendix A. For a description of the pinhole camera model used in this thesis, as well as an overview of extrinsic camera parameters, see appendix B. Inertial measurement units (IMU), their modeling and noise behavior are discussed in appendix C. As the algorithm designs in the following chapters rely on quaternions as rotation description, see appendix D for a comprehensive description of quaternions and derivations regarding time-differentiation and the small-angle approximation.

2.1 Notation

For the sake of consistency in MSCKF related work, the nomenclature used for this thesis parallels the nomenclature of Mourikis et al. in their initial presentation of the MSCKF [20].

Scalars as lower-case regular letter	c
Vectors written as lower-case bold letter	$\mathbf{p}$
Vectors as in brackets	$[\,a\;b\,]$
Matrices as upper-case bold letter	$\mathbf{A}$
Unit quaternion with the letter q	q
Unit vector with a bar	$\bar{\mathbf{a}}$
Estimations with a circumflex	$\hat{x}$
General Errors with a tilde	$\tilde{x}$
Orientation Errors with a δ	$\delta\theta$

General operations are written as follows:

Transpose of a matrix or vector	$\mathbf{x}^\top$
Inverse of a matrix or function	$\mathbf{R}^{-1}$

2.2 Quaternions

Expressing rotations in quaternion multiplication is the most efficient description concerning the number of calculations [21]. For this reason, the quaternion rotation representation is extensively used throughout this thesis. This section defines the quaternion nomenclature. For a more precise definition of quaternion mathematics, an intuitive explanation of how they represent rotations, as well as some derivations to the concepts of small-angle rotation and the quaternion time-derivative, see appendix D.

We use quaternions in the form of

$$q \triangleq \begin{bmatrix} q_v \\ q_w \end{bmatrix} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix}$$

with q_x, q_y and q_z being the irrational components, while q_w is the real component.

Note, that we use the *ijk* quaternion, as this notation is most commonly used in VIO implementations generally and in the MSCKF implementations in particular. Although there are no conceptional changes between these styles, some calculations are structured differently. A comparison of the two can be found in the work of Sola et al. [22].

The quaternion product is defined by the $\otimes$ operator

$$p \otimes q = \begin{bmatrix} p_w q_v + q_w p_v + p_v \times q_v \\ p_w q_w - p_v^\top q_v \end{bmatrix}$$

where the non-commutative cross product is used. The quaternion multiplication can be alternatively expressed by a matrix-vector multiplication, where, as the quaternion product is non-commutative, two different matrix structures exist depending on the order.

$$q_1 \otimes q_2 = [q_1]_L q_2 \quad \text{and} \quad q_1 \otimes q_2 = [q_2]_R q_1$$

which we define as

$$[q]_L = q_w \mathbf{I} + \begin{bmatrix} [\mathbf{q}_v]_\times & q_v \\ -q_v^\top & 0 \end{bmatrix}, \quad [\mathbf{q}]_R = q_w \mathbf{I} + \begin{bmatrix} -[\mathbf{q}_v]_\times & q_v \\ -q_v^\top & 0 \end{bmatrix}. \tag{1}$$

The cross product matrix $[\mathbf{a}]_\times$ is defined as

$$[\mathbf{a}]_\times \triangleq \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}.$$

We define a function $C(\cdot)$, which returns a rotation matrix $\mathbf{R}$ from a quaternion.

$$\begin{aligned} \mathbf{Rp} &= \mathbf{q} \otimes \mathbf{p} \otimes \mathbf{q}^* \\ \mathbf{R} &= C(\mathbf{q}) = (q_w^2 - q_v^\top q_v)\mathbf{I} + 2q_v q_v^\top + 2q_w[q_v]_\times \end{aligned} \tag{2}$$

The small-angle approximation of a quaternion is calculated as

$$\begin{aligned} C(\mathbf{q})^\top &= (1-0)\mathbf{I} + 0 + [\theta]_\times \\ C(\mathbf{q})^\top &\simeq \mathbf{I} - [\theta]_\times \end{aligned} \tag{3}$$

where we use θ from the axis-angle quaternion description

$$q = \begin{bmatrix} \mathrm{u}\sin\frac{1}{2}\theta \\ \cos\frac{1}{2}\theta \end{bmatrix}$$

The time-derivative of a quaternion is

$$\dot{q} = \frac{1}{2}\mathbf{q} \otimes \begin{bmatrix} \omega \\ 0 \end{bmatrix} \tag{4}$$

with ω as the angular rate.

2.3 Extended Kalman Filter

For a comprehensive description probabilistic estimators, the Kalman Filter and the Extended Kalman Filter (EKF), refer to appendix A. The current section merely defines the specific notation used in this thesis.

Table 1: State types

true state	$\mathbf{x}$
nominal state	$\hat{\mathbf{x}}$
error	$\tilde{\mathbf{x}}$

Table 1 defines the notation of the different state types of the EKF.

We describe the relationship between the error state and the true state as

$$\text{true state} = \text{nominal state} + \text{error}$$

where the nominal state is an estimation of the true state. For completeness, here is the full EKF propagation step, followed by the notation definition:

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

and the update step:

$$\begin{aligned}\mathbf{r}_k &= \mathbf{z}_k - h(\hat{\mathbf{x}}_k) \\ \mathbf{K}_k &= \mathbf{P}_k \mathbf{H}^\top \left(\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \\ \hat{\mathbf{x}}_{k+1} &= \mathbf{K}_k \mathbf{r}_k \\ \mathbf{P}_k &= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k\end{aligned}$$

The function $f(\cdot)$ is the prediction function operating on the current state estimation and the prediction input. The linearization of this function is the Jaocobian matrix $\mathbf{F}$. The vector $\mathbf{u}$ is the prediction input. $\mathbf{Q}$ is an approximation of the process noise. During each prediction, the state estimation $\hat{\mathbf{x}}$ is changed, along with the covariance $\mathbf{P}$ around this state. The current timestep is represented by the index k. Note that these matrices can be different in each time-step.

$h(\cdot)$ remaps the current state estimation into the expected measurements. The respective linearized Jacobians are $\mathbf{F}$ and $\mathbf{H}$. During the update step, the measurement information in the vector $\mathbf{z}$ is used. The residual vector is written as $\mathbf{r}$. A note regarding error accumulation: If a state is observable, such as the rotation about the x- and y-axis of the IMU, then the error is bounded. If the error is not observable, such as the z-axis rotation, then the error can grow without bounds. Linearizing the residual to receive the Jacobian $\mathbf{H}$ is done as follows:

$$\begin{aligned}\mathbf{r} &= \mathbf{z} - \hat{\mathbf{z}} + \mathbf{n} \\ &= h(\mathbf{x}) - h(\hat{\mathbf{x}}) + \mathbf{n}\end{aligned} \tag{5}$$

using first-order Taylor approximation results to:

$$\mathbf{r} \simeq \mathbf{H}_\mathbf{x} \tilde{\mathbf{x}} + \mathbf{n} \tag{6}$$

where $\mathbf{H}_x$ is the Jacobian of the function $h()$ with respect to $\mathbf{x}$.

2.4 Error-State Kalman Filter

The true state vector in a Kalman Filter is a composition of the nominal state - the estimation of the true state - and the error between the truth and the estimation.

$$\mathbf{x}_t = \hat{\mathbf{x}} \oplus \delta\mathbf{x}$$

where $\oplus$ is a general compounding operator.

The Extended Kalman Filter propagates the estimated state through a nonlinear function $f(\cdot)$ and propagates the covariance matrix of this state's error through linearization of said function. In contrast, the indirect - or Error-state Kalman Filter (ESKF) linearizes a description of the error-state propagation and uses this linearization to transform the covariance matrix. Note, that the *estimated* error state is always zero in the prediction step, but the covariance grows with each prediction iteration. Only through a measurement update are the accumulated errors in the error state rendered observable. After calculating the error based on the received observations, these errors are added to the estimations in the state vector to correct the prediction. Through this step, the state estimation, which has been accumulating errors during each propagation in the prediction step, is rectified using the observation of these errors. The assumption of a zero-mean estimation of the error state for the prediction step is valid again until the next measurement update.

Using the error state as the point of linearization yields two major benefits: The linearization error becomes smaller and the usage of quaternions as a rotation representation can be superseded by their small-angle approximation as the following two paragraphs explain.

Linearizing non-linear functions, such as rotation propagation, intrinsically decreases the accuracy of the result. Smaller rotations have less loss of accuracy through eg. Taylor linearization, because the size of the terms following the first-order expression decreases exponentially with the size of the term. The change of the error in the estimated rotation is expected to be smaller than the change in rotation itself - therefore, the linearization of the error state results in a more accurate propagation than a linearization of the state propagation.

In the case of the MSCKF, rotational information is saved via quaternion representation. This choice makes the states rotation immune to parameter singularity - which can happen with a minimal representation. However, this redundant information may constrain the covariance matrix into becoming a singular matrix. The error state uses a minimal representation because the expected field of operation is so small, that no parameter singularity is expected to occur. This minimal representation lifts any possible constraints on the covariance matrix.

The following is a mathematical formulation of the Error-state Kalman Filter in comparison to a regular EKF. Refer to section 2.3 for the nomenclature used throughout this following section.

Extended Kalman Filter	**Error-State Kalman Filter**	
Prediction Model:	Prediction Model:	
	$\mathbf{x} = f_1(\mathbf{x}, \mathbf{u}, \mathbf{n})$	(7)
	$\hat{\mathbf{x}} = f_2(\hat{\mathbf{x}}, \mathbf{u})$	(8)
$\mathbf{x} = f(\mathbf{x}, \mathbf{u}, \mathbf{n})$	$\delta\mathbf{x} = f_2(\mathbf{x}, \delta\mathbf{x}, \mathbf{u}, \mathbf{n})$	(9)
Prediction:	Prediction:	
$\hat{\mathbf{x}} = f(\hat{\mathbf{x}}, \mathbf{u})$	$\delta\hat{\mathbf{x}} = f_2(\hat{\mathbf{x}}, \delta\hat{\mathbf{x}}, \mathbf{u}) = 0$	(10)
$\mathbf{P} \leftarrow \mathbf{F}\ \mathbf{P}\ \mathbf{F}^\top + \mathbf{Q}$	$\mathbf{P} \leftarrow \mathbf{F}_{\delta x}\ \mathbf{P}\ \mathbf{F}_{\delta x}^\top + \mathbf{F}_n\ \mathbf{Q}\ \mathbf{F}_n^\top$	(11)
Correction:	Correction:	
$\mathbf{K} = \mathbf{P}\mathbf{H}^\top(\mathbf{H}\mathbf{P}\mathbf{H}^\top + \mathbf{R})^{-1}$	$\mathbf{K} = \mathbf{P}\mathbf{H}^\top(\mathbf{H}\mathbf{P}\mathbf{H}^\top + \mathbf{R})^{-1}$	(12)
$\hat{\mathbf{x}} = \hat{\mathbf{x}} + \mathbf{K}(\mathbf{z} - h(\hat{\mathbf{x}}))$	$\delta\hat{\mathbf{x}} = \mathbf{K}(\mathbf{z} - h(\hat{\mathbf{x}}))$	(13)
	$\hat{\mathbf{x}} = \hat{\mathbf{x}} \oplus \delta\hat{\mathbf{x}}$	(14)
$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}$	$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{H})\mathbf{P}$	(15)

Eq. 7-8 split the state estimation into the error-state estimation function. The predicted error state is always 0 (as in eq. 9) as the mean estimated error is zero until a measurement occurs. Linearizing the error state process model around zero yields a Jacobian matrix $\mathbf{F}$, which is used to propagate the error-state covariance matrix (eq. 10). Through a measurement update, the Kalman gain can be calculated to minimize the covariance entries (eq. 11). Note, that the Jakobi matrix $\mathbf{H}$ in eq. 12 and 15 is the measurement function derived in concerning the *error-state* description. Eq. 14 adds the observed error to the actual state vector and eq. 15 updates the covariance matrix of the error state.

3 Related Literature on Position Estimation

There exists a vast amount of work regarding position estimation. This chapter will focus on the progress in the field of combining visual and inertial measurements. First, we will generally describe the different possible approaches, subsequently detailing the most advanced implementations and their individual advantages and shortcomings.

3.1 Odometry and Maps

In literature, ordinarily, three types of methods for position estimation are recognized [23, 24]

- 'A Priori' map
- Simultaneous Localization and Mapping (SLAM) as well as
- Dead Reckoning or Odometry.

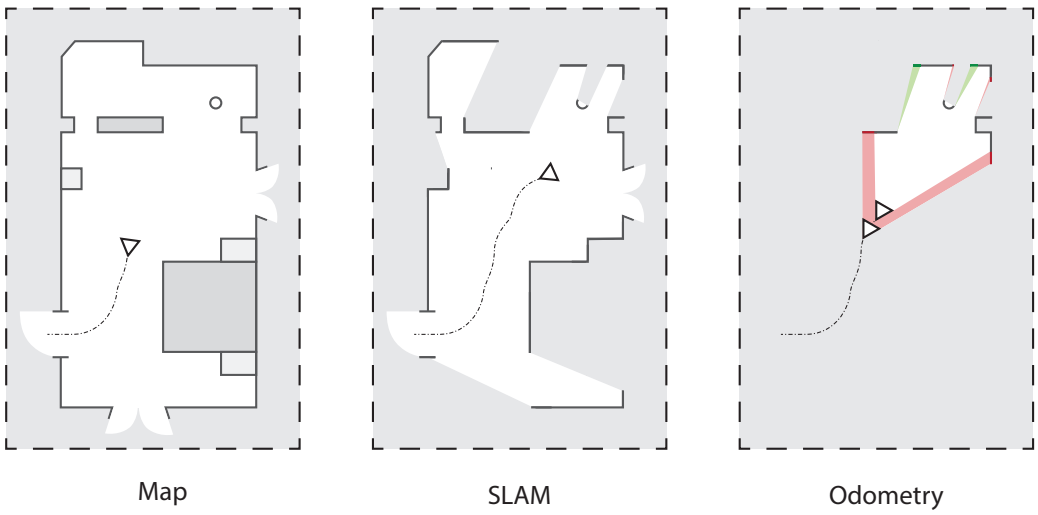

Figure 1: Visual comparison of A Priori map, SLAM and Odometry

Figure 1 shows a visual comparison between the three different strategies. The 'a priori' map has all the information used for localization beforehand. SLAM self-generates the necessary information and saves it for later re-localization. Dead reckoning only uses information from its immediate past to estimate its movements. The next paragraphs will outline these three approaches.

The 'a priori' method requires an existing map. The information in this map is used by an agent to localize itself within it. This map can be a feature map [25], visual markers, or any conceivable other static *external* information, which the agent can measure and orient itself through. Some approaches use a more high-level representation of the environment, for example, an occupancy grid [26] and blueprints of the location [27]. Common approaches for this type of localization are based on different types of the Markov Localization, using probability functions to describe the location estimation of the agent [28]. The Monte-Carlo Localization [29, 30] is one such algorithm. It is sampling-based, with an arbitrary variability between estimation accuracy vs. computational cost.

Simultaneous localization and mapping (SLAM) based methods generate a map of the surroundings, while *simultaneously* estimating the agents' position in this map. There exists a considerable body of literature regarding this topic. This section will only provide a brief description of some key concepts and a starting point for further research. It is suggested, that the hippocampus of rats use a form of SLAM, combining odometric information with landmarks [31]. This model of the hippocampus forms the basic concept of RatSLAM [32]. Autonomous driving systems are increasingly focused on using SLAM algorithms since the robot Stanley won the DARPA autonomous driving challenge in 2005 [33]. Recent approaches have shown tremendous advancements since then, partially assisted by the steady increase in computational power which makes more resource-heavy approaches feasible. Common extrinsic sensors for SLAM are laser scanners [19, 34], depth cameras, [15, 35, 36] and regular mono- or stereo-cameras [36, 37]. The company Waymo describes their usage of these sensors in their whitepaper [38] concerning an autonomous car, similar to Sun et al. in their recent paper [19]. Small mobile robots with a payload and price limit rely on visual sensors (and sometimes inertial measurements). Huletski et al. provide an overview of open-source visual SLAM implementations up until 2015 [39]. Alternative approaches to feature-based estimators are direct SLAM [37, 40] and SLAM based on other descriptors such as lines and scene geometry [41, 42]. The main challenge of these algorithms for mobile robots and AR devices is reducing the computation requirements to allow real-time information processing.

Dead reckoning forsakes the concept of a map entirely, accumulating estimated incremental changes in the position [7, 12]. The most straightforward odometry for land-based robots uses wheel encoders and an underlying motion model to estimate the position change of the robot [12]. This change is then integrated over time. The error in the position estimation grows continuously, without any global reference information the agent can return to. The increase in computational efficiency and the high degree of accuracy of these incremental estimations have allowed odometry to become a viable tool for mobile robots. Examples for dead reckoning strategies are IMU-based integration [43], flow of radial laser scans [44] and most prominently, visual (inertial) odometry, which will dominate the remainder of this chapter.

3.2 Visual and Inertial Sensor Fusion

Visual odometry (VO) uses image sequences to estimate the camera's motion. This set of algorithms can use mono- or stereo-camera setups. Mono cameras have no inherent scale information, which is compensated for, either with an additional sensor or by supplying scale information via markers of a defined size. Stereo-cameras contain intrinsic information of scale through the known distance between the two cameras. Visual odometry, regardless of the number of cameras it uses, must extract information from the image frames it receives. Such an algorithm can either operate on individual image geometries (eg. feature points) or optical flow techniques (matching intensity values between the entire images, or selected image patches)

see chapter 3.3. To restitute scale information from monocular camera-based motion, visual-*inertial* odometry (VIO) adds the information of an inertial measurement unit to the estimation process. This additional IMU data is commonly used in stereo based visual odometry systems and has been shown to greatly increase filter robustness while maintaining equivalent efficiency [45].

Recall, that odometry is different from simultaneous localization and mapping (SLAM) in the regard, that SLAM approaches save past information such as keypoints. This allows a SLAM to constantly re-evaluate the current position estimate using past experiences. It, therefore, has the advantage of loop-closure, when revisiting a saved keypoint. VIO has no such loop-closure, as its estimation is only based on the current sensor information and state, disregarding any past information. This inherent limitation of VIO systems, while reducing their long-term accuracy, allows them to run on less resource-hungry devices. Expanding on this concept of VIO, several recent implementations use the concept of a sliding window, saving a bound number of past frames or features and using the combined information, thereby increasing the accuracy of the estimation.

3.3 Visual Inertial Odometry

This section contains an executive overview of various vision-based position estimation pipelines dealing with the most prominent examples of vision-based pose estimators. We will first distinguish between some general concepts, that differentiate the different algorithms. Subsequently, notable examples of position estimation algorithms are summarized. These include OKVIS [46], ROVIO [47, 48], VINS [49], SVO [50, 51] + MSF [52], Trifo-VIO [53] as well as a description of the MSCKF [20]. The MSCKF algorithm will be concisely described as well. Chapter 4 will deal with the mathematics behind the MSCKF approach in significant detail.

First, let us distinguish between loosely and tightly coupled sensor fusion as the terminology is used in multiple papers [46, 47, 50, 52, 53, 54, 55]. Loosely coupled means, that the information from both sensors is treated independently from one another. A sensor, including its processing system, can be easily swapped out for another system with the same output. For example, one might merge the estimated pose from a GPS module with the estimated pose based on a laser-scanner. Any of the two could be swapped out for a different pose estimation system, as long as the *output* is equal. Tightly coupled fusion, on the other hand, uses the correlation between the internal states of the systems to generate a consolidated estimation of both sensors. This has the potential of generating a far more accurate estimate, depending on how interwoven the sensor outputs are [54]. Additionally, the need for sensor information abstraction is reduced - eg. feature position measurements from cameras or GPS pseudo-ranges can be used *directly* in the estimation. This, in turn, reduces the calculatory overhead. For these reasons, the following estimators, with the exception of MSF, all tightly couple sensor outputs with one another.

In the estimation procedure itself, we discriminate between optimization-based - and filter-based position estimation. Filters typically use some version of an Extended Kalman Filter to minimize the covariance, while optimization-based approaches minimize a cost function based on some calculated residual. The MSCKF is one such example of a filter based approach [20, 45], OKVIS [46] is a VIO which uses optimization.

Filter based systems accumulate inaccuracies through linearization errors. Observability consistent EKF, as proposed in [55], use the *first* estimation of a state to construct the respective Jacobians, outperforming filters, that linearize at the current state. The Unscented Kalman Filter [56] samples points around the mean which are then propagated through the underlying non-linear function. This results in a more accurate mean and covariance estimation. Nonetheless, filter-based approaches are restrained by their Markovian design and reliance on a linearization step. The optimization-based approach is not bound by this Markov assumption. Bundle Adjustment is the term used to describe the optimization over multiple layers of measurements [57]. This correction regarding a large buffer of information makes the resulting estimation more accurate than a filter, and the calculations more taxing on the computing system. Efficient optimization-based estimators use sliding windows [58, 46, 49, 59] and calculate a marginalized 'prior' factor, with condensed information from all states outside the sliding window [49, 59].

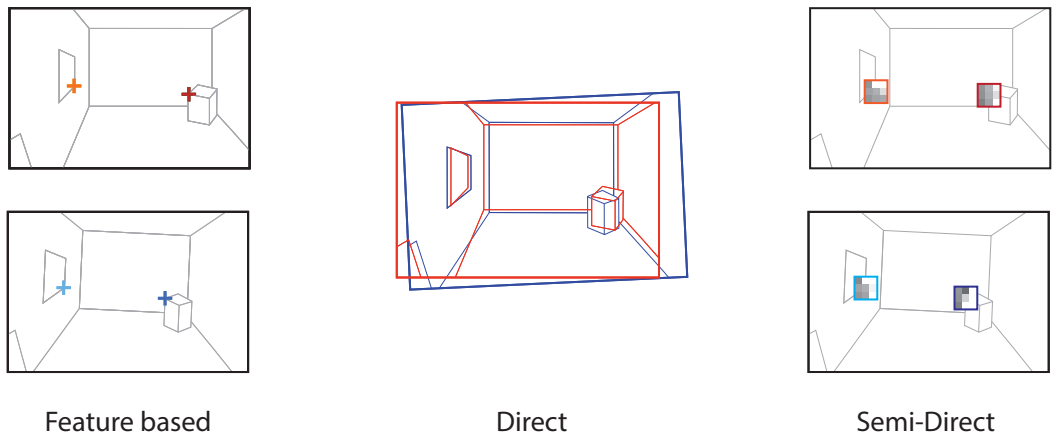

Figure 2: Difference between feature based, direct and semi-direct approach to frame comparison

A final major separation can be drawn between direct, semi-direct and indirect approaches regarding camera information. Indirect - feature-based - approaches extract some abstract feature description from the received image (eg. FAST Features [60], SIFT Features [61]). Direct approaches [37, 40] use the pixel values of two images directly, to estimate the relative movement between them. Semi-direct approaches use patches, typically based around image features, and estimate the relative change in position for these individual patches. The photo-

metric MSCKF [62] and ROVIO [47] are based on such an approach. Figure 2 visualizes these three approaches.

Figure 3 shows a summarized visualization of the main differences between the various algorithms presented in the following paragraphs. Note how algorithms utilize past information differently and extract image information in various ways.

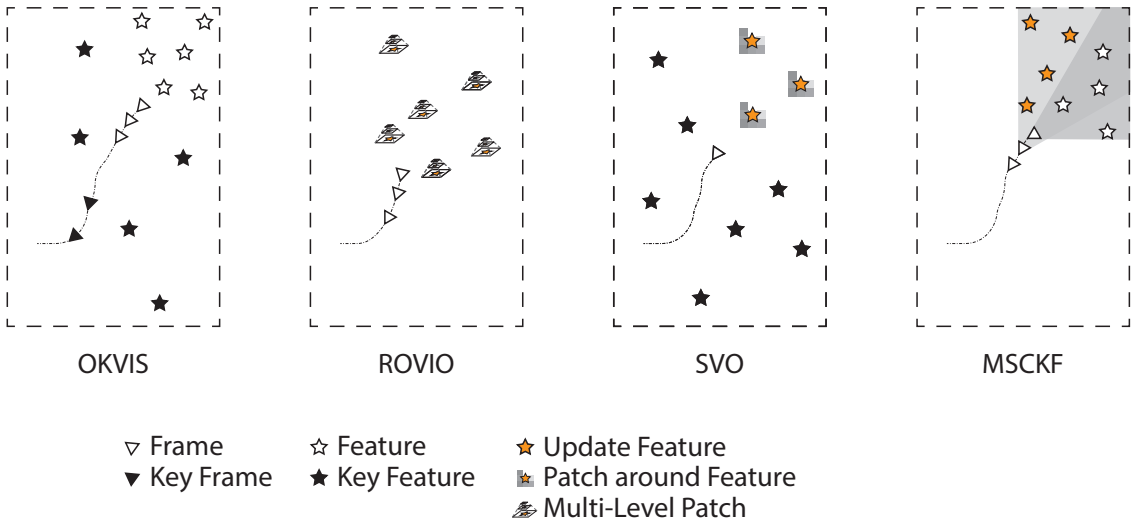

Figure 3: Visualization of four prime examples of VIO algorithms

OKVIS [46] uses an optimization approach to minimize the reprojection error of features between *keyframes* in a visual SLAM. The algorithm from the paper by Leutenegger et al. *tightly* integrates IMU data into a visual SLAM. This SLAM operates on key camera frames, which it selects based on the degree of novel information in the camera frames. If the number of frames surpasses the maximum limit of the sliding window of camera frames, non-key frames are rationalized. With every new feature measurement, a cost function containing the weighted reprojection errors concerning all feature measurements and the IMU error (based on the initial IMU estimation and the corrected IMU state) is minimized. OKVIS was conceived to operate on stereo-camera information. VINS [49] is algorithmically similar to OKVIS and advances the concept with full loop-closure and pre-integrates the inertial measurements for the cost function.

ROVIO [47, 48] is an Extended Kalman Filter, minimizing photometric multi-level patches of monocular image frames. The state includes the IMU pose information, as well as vectors to the currently tracked features. Inertial measurements are used for state propagation. The state correction proceeds any time a new image is committed by the camera. Newly extracted features are added to the state vector. For any existing feature that should be visible in the frame,

an intensity error is computed using the expected patch position in the frame and is used as innovation term to correct the accumulated errors of the propagation.

Semi-Direct Visual Odometry (SVO) [50, 51] directly uses the *pixel values* of a monocular camera to estimate its position. OKVIS and the MSCKF extract features from an image and use these features movements to estimate any position change. By contrast, the pose change between successive images in the SVO algorithm is determined by minimizing the photometric error of patches between frames - directly utilizing pixel-intensity values. The pipeline builds a map of feature points from selected keyframes, which are then used to project these patches used for the cost function. The accuracy of the features' position in space is incrementally increased with every measurement.

The Multi-State Constraint Kalman Filter (MSCKF) is an Extended Kalman Filter based position estimator, merging IMU data and the movements of detected features within a sliding window of camera images. The algorithm uses the movement of a measured feature and compares this measured movement to the expected movement based on the IMU propagation. The error between expectation and measurement is then used to correct the prediction. See chapter 4 for an in-depth explanation of this algorithm.

Trifo-VIO [53] extracts feature points *and* lines from the images of a stereo-camera setup. Additionally using line features to estimate the position increases accuracy in low-texture environments. The estimation process is based on the MSCKF algorithm using a sliding window for position estimation. In addition to line-based feature extraction, the paper presents a loop closing procedure using the Extended Kalman Filter itself. This is done by injecting the previously estimated feature position of a keypoint feature back into the update step instead of calculating a new one. This results in the update step nudging back the position estimate to the original estimated feature position.

MSF is a generic *Multi-Sensor Fusion* Extended Kalman Filter framework [52] for state estimation. The algorithm loosely couples an unlimited number of sensor information and handles losing and newly receiving sensor signals over time. The framework supports delayed updates and uses an iterated EKF [63] which recursively optimizes the linearization point to reduce linearization error. The state prediction is based on IMU measurements - the same prediction implementation is used by the MSCKF, see chapter 4 for details. On top of this buffer (timeline) of state predictions, any measurement can be applied to any state in the buffer. The subsequent states after this edited state are *repropagated* and any measurements committed to a state *after* the updated state are reused for an update of this new prediction. The MSF framework is designed to be highly modular and additional sensors may be integrated easily.

A paper by Jung et al. [64] was released during the evaluation period of this thesis, which presents their algorithm design for a photometric visual odometry pose estimator using stereo images. The Schmidt-EKF, based on the MSCKF by Geneva et al. [65] is a keyframe aided resource considerate visual-inertial SLAM and was similarly released well after the research phase of this thesis. We chose not to retroactively evaluate their implementations but wish to nevertheless take note of them here, to provide a more complete picture of the current literature.

3.4 Visual Inertial Odometry Comparison

This section compares the advantages of the different approaches and underlines these differences based on the individual papers and the in-depth comparisons of some of the estimation approaches of Delmerico and Scaramuzza [66] and Huletski et al. [39]. Estimation accuracy and computational efficiency are the two main parameters in this comparison. Note, that all algorithms in this section are among the top-performing vision-based estimators.

The paper of Delmerico and Scaramuzza [66] compares some of the listed algorithms regarding their CPU performance, memory usage, and root mean square error (RMSE) accuracy. Note, that non-inertial based algorithms such as LSD-SLAM [67] and ORB-SLAM2 [36] and stereo-camera based algorithms such as Trifo-VIO [53] and the stereo MSCKF [45] are not evaluated in the paper. We will therefore directly use the evaluation results of the respective papers to compare the algorithms to one-another.

From the listed algorithms, similarly small root mean square errors (RMSE) in the EuRoC Dataset [68] are reported by OKVIS, Trifo-VIO, the stereo MSCKF, and ROVIO. The VINS algorithm takes the lead with consistently accurate performance in all tests, although Zheng et al. [53] note, that heavy rotation is detrimental to the initialization phase of the algorithm, with OKVIS having similar problems in these situations. This was especially noticeable in the Trifo dataset [53] which uses an industrial robot to move the camera. Trifo-VIO and the stereo MSCKF seemed to produce competitive results in these rotation heavy datasets.

Trifo-VIO does not present the CPU load of their algorithm during a performance run. ROVIO takes the lead for the most resource-efficient estimator. The stereo MSCKF is also among the more resource-friendly implementations according to the tests of Sun et al. [45], where it is slightly less efficient than ROVIO and more efficient than VINS Mono and OKVIS. Another resource efficient algorithm is a combination of SVO and inertial data using MSF [69] but ranks on the lower end accuracy wise. VINS and OKVIS seem to be the most resource-hungry estimators analyzed.

Between the two most efficient algorithms: ROVIO and the stereo MSCKF, the stereo MSCKF shows slight superiority in performance, while ROVIO is less computationally demanding. Both algorithms have an open source implementation (ROVIO [70], MSCKF [71]).

4 MSCKF Analysis

The following chapter is an extensive analysis of the mechanics utilized by the MSCKF presented by Mourikis et al. [20] and the Sun et al. [45] implementation. We will first give a brief summary of the algorithm and follow up with the structure of the remaining chapter. After presenting the MSCKF pipeline in detail, the stereo-camera expansion is derived. An understanding of the mechanics of the MSCKF is necessary in order to follow the anchor-frame MSCKF and the photometric expansion in the subsequent chapters.

4.1 Overview

This section first gives a summary of how the MSCKF works conceptually. Then, the structure of the implementation is presented. The section arrangement for the rest of the MSCKF analysis chapter is derived from this structure.

4.1.1 Summary

The MSCKF is an Extended Kalman Filter based approach, which is separated into a prediction step and an update step. The prediction step utilizes the information gathered from the IMU to form an estimate of the sensor's current position. The correction step, or measurement update, uses feature movements detected by the camera in a moving window of camera frames to correct this estimated position. Figure 4 shows this process sketched out in four steps. The camera frames visualized in step number one are separated *temporally* as well as *spatially*. Every time a camera image is received, the IMU information is used to estimate the position of the camera. These camera images are then saved for a period of time, see figure 5. This process is denominated a *moving window* of images.

Features in the images are tracked through this moving window of camera frames. Once a feature is *no longer tracked* (step number 2), this feature is used to correct the movement estimation between the frames. Based on this IMU estimation, the true position of the feature is estimated in step number 3. Finally, the difference between the true measurements of the feature are compared to where the feature should have been measured based on the estimation of the feature position (step number four). This error in measurement and estimation is then used to update the prediction.

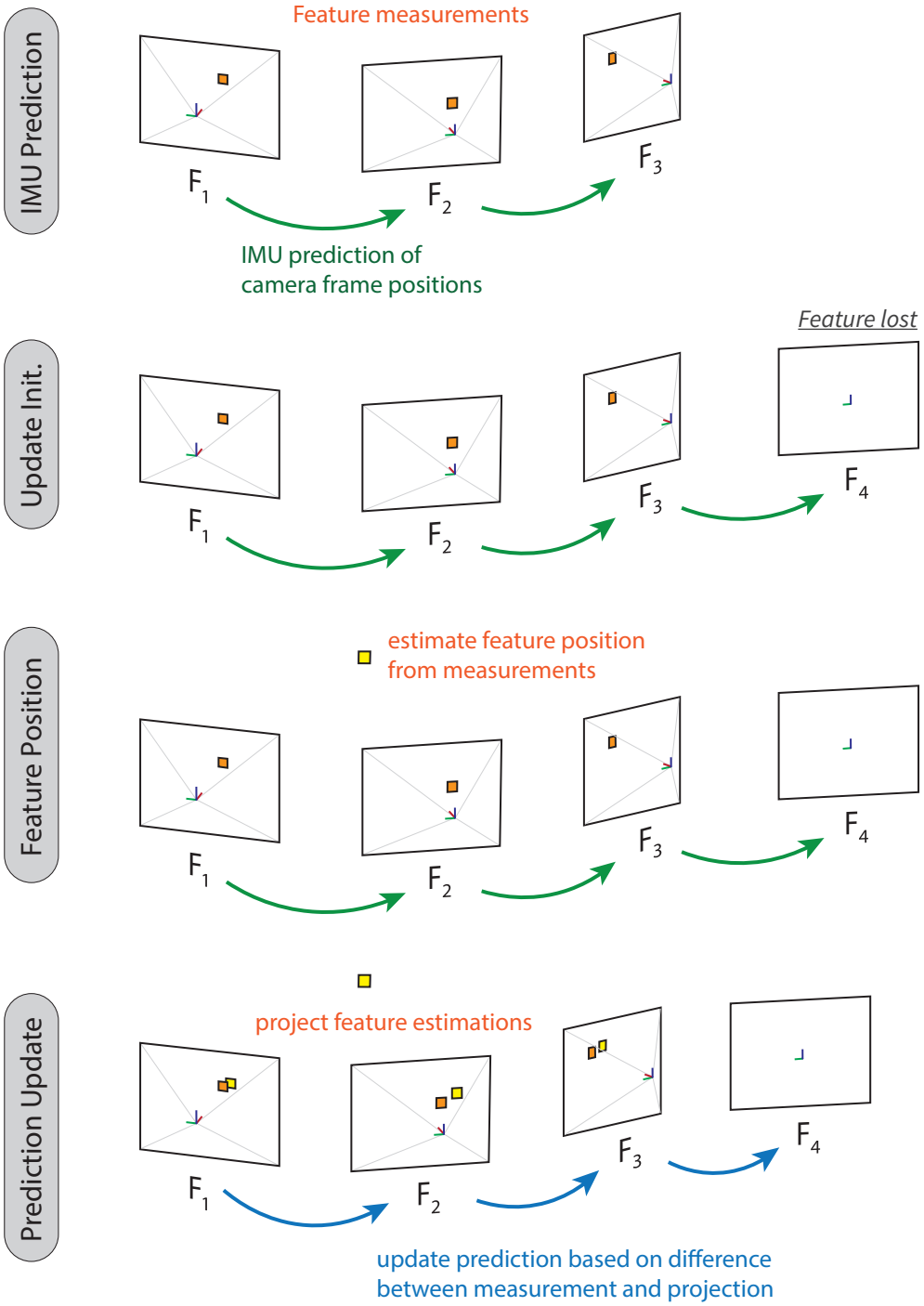

Figure 4: Prediction and update process sketch of the MSCKF split into four steps: tracking, initiating an update, estimating and updating the prediction

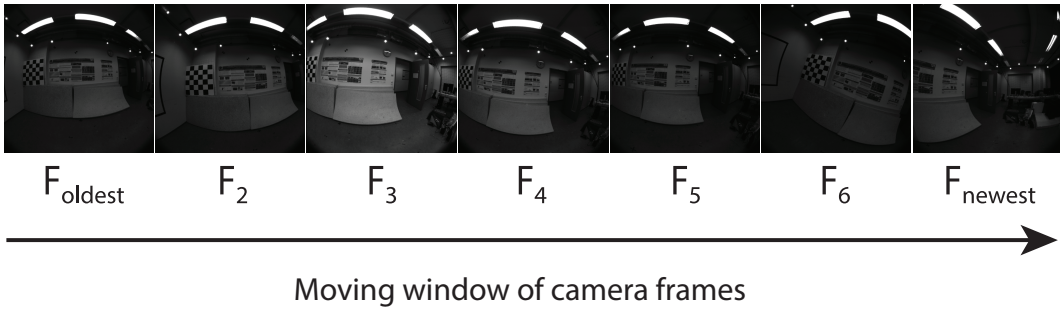

Figure 5: A moving window, saving the past few frames to be used for an update step

4.1.2 Structure

For efficiency purposes, the MSCKF implementation of Sun et al. [45] is split into two main parts, which we will call the feature extractor and the MSCKF algorithm. Figure 6 shows these two components, as well as the main individual elements within the MSCKF algorithm.

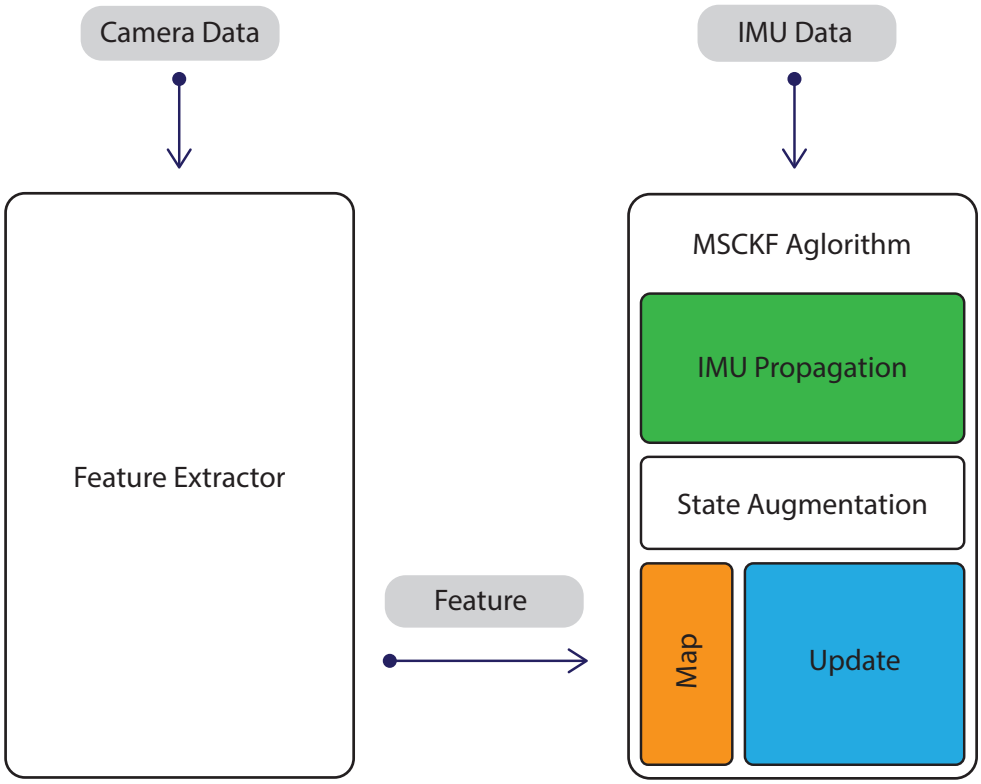

Figure 6: The building blocks of the MSCKF - feature extractor and algorithm - as well as data sources

The following sections will first present the notation used for the MSCKF. Further, the data structure within the algorithm is presented, including the state vector and the 'external' map, in

which the feature information is saved. The IMU data processing will be derived, through which we predict the movement of the camera. The camera position is saved in the state-vector - this state-vector augmentation is explained as well. The update step uses the entire information of a feature to correct the error of the IMU propagation. This correction step is the final building block needed for the MSCKF algorithm. Thereafter, we will give an overview of the feature extraction and matching process in the 'Feature Extractor' of figure 6, designed by Sun et al. [45].

4.2 Frames and Definitions

All the reference frames used in this thesis are presented in table 2.

Table 2: Frames used in this work

Ground frame	$\{G\}$
IMU frame	$\{I\}$
Camera frame	$\{C\}$

The Ground frame is considered the reference frame for the IMU frame; the camera frame is linked to the IMU frame, see figure 7 for reference.

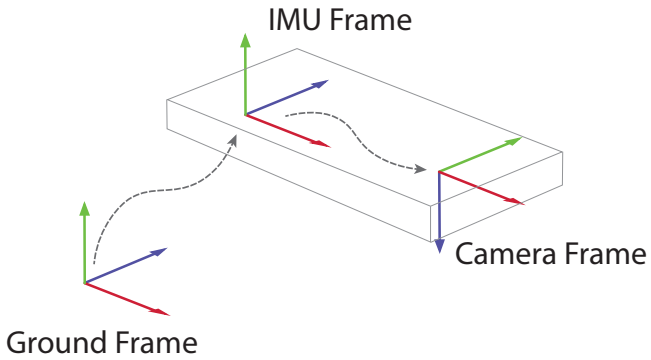

Figure 7: Relative frame position visualized through coordinate systems and arrows

The temporal arrangement of frames and variables are shown through an index character in the post-subscript position, eg. $\{I\}_k$ which is the IMU frame at time-step k. The frame associated with a variable is generally noted as a pre-superscript, for example, ${}^{G}f$ is a feature from the viewing point of the Ground frame $\{G\}$; these pre-superscripts are sometimes omitted in favor of shorthand notation to increase readability - though such a substitution will preemptively be defined in the text.

Rotations between frames are represented either in their quaternion form q, a rotation matrix R or the three-dimensional minimal representation θ vector. The forms in table 3 all represent a rotation from Ground frame orientation $\{G\}$ into the IMU frame orientation $\{I\}$. The function $C(\cdot)$ generates the equivalent rotation matrix R from a quaternion q, as defined in eq. 2.

Table 3: Rotation from Ground frame to IMU frame

quaternion formulation	${}^{I}_{G}q$
rotation matrix	${}^{I}R_{G}$
minimal representation	${}^{I}\theta_{G}$

$$C({}^{I}_{G}q) = {}^{I}R_{G} \ , \ q = \begin{bmatrix} \frac{1}{2}{}^{I}\Theta_{G} \\ 1 \end{bmatrix} \tag{16}$$

where eq. 16 is the small-angle representation of a quaternion (see eq. 3).

4.3 Data Representation

Conventional EKF implementations save all information used to estimate the robot's state inside the state vector. The MSCKF follows a unique strategy, where only a part of this information is represented in the state vector. While the pose information is saved in the state, the feature information is saved externally. This collection of feature information will be referenced to as a *map*. Details how this data dependency is handled within the Kalman Filter will follow. To compute the Error-state Kalman Filter, a description of the *error* state is necessary and will be formulated in this section.

4.3.1 State Vector and Map

The MSCKFs state vector consists of an IMU state $\mathbf{x}_{IMU}$ and multiple camera states $\mathbf{x}_{C_n}$:

$$\mathbf{x} = [\ \mathbf{x}_{IMU}^{\top}\ \mathbf{x}_{C_1}^{\top}\ \mathbf{x}_{C_2}^{\top}\ ...\ \mathbf{x}_{C_n}^{\top}\ ]^{\top} \tag{17}$$

where n represents the number of camera frames currently in the state vector.

The information in the IMU state vector $\mathbf{x}_{IMU}$ varies between MSCKF implementations. Mourikis et al. [20] use a minimum viable description consisting only of the pose, the velocity, and biases; Sun et al. [45] expand this by estimating the extrinsic parameters of the sensor setup - the relative pose of the camera regarding the IMU frames. Other implementations add

additional IMU calibration components [72].

$$\mathbf{X_{IMU}} = [\ {}^{I}_{G}q^{\top}\ \mathbf{b}_g^{\top}\ {}^{G}\mathbf{v}_I^{\top}\ {}^{I}_{C}\mathbf{b}^{\top}\ {}^{G}\mathbf{p}_I^{\top}\ {}^{I}_{C}q^{\top}\ {}^{I}\mathbf{p}_C^{\top}\ ]^{\top}$$

With the elements in the vector corresponding to

rotation from $\{G\}$ to $\{I\}$	${}^{I}_{G}q$
bias of the gyroscope	$\mathbf{b}_g$
velocity of $\{I\}$ in $\{G\}$	${}^{G}\mathbf{v}_I$
bias of the accelerometer	$\mathbf{b}_a$
position of $\{I\}$ in $\{G\}$	${}^{G}\mathbf{p}_I$
rotation from $\{I\}$ into $\{C\}$	${}^{I}_{C}q^{\top}$
position of $\{C\}$ in $\{I\}$	${}^{I}\mathbf{p}_C$

The camera state X_{C_n} consists of a quaternion and a position vector representing the orientation and position of the camera at the time the camera frame was recorded:

$$\mathbf{X_{C_i}} = [\ {}^{C}_{G}q^{\top}\ {}^{G}\mathbf{p}_C^{\top}\ ]^{\top}$$

Another MSCKF implementation [62] uses a description of the IMU state for the camera state instead of transforming the current IMU state into the camera state. Remember, that only the *pose of the camera* is recorded in the Kalman Filters state vector; no feature information is present in the state. This feature information is instead saved in a data array external to the Filter. This means, that the feature information needs no covariance assigned, but cannot be used to calculate the residual, see chapter 2.3 - this distinction will become important when formulating the update step of the filter. Figure 8 visualizes the data structure of the MSCKF. It shows the two sources of information (IMU and camera) and the data structures used to save them (map and state vector). Incoming IMU data is directly used to predict the *current* position based on the information in the state vector. This position estimation is saved in the IMU state. Every time the camera receives a frame, an estimation of the position from where this image was taken is added to the state vector in form of a camera state (using the prediction of the IMU) as shown in the state vector in figure 8 as the grey dotted frame F_7. Existing features observed in this camera image are added as observations to the existing features (meaning, the u and v of the feature in the respective image are saved). These observations are represented as grey dotted squares in the map. Feature observations that do not fit to any existing features are added as new feature observations to the *map*. These are the two grey dotted L-shapes underneath the rest of the features in the map.

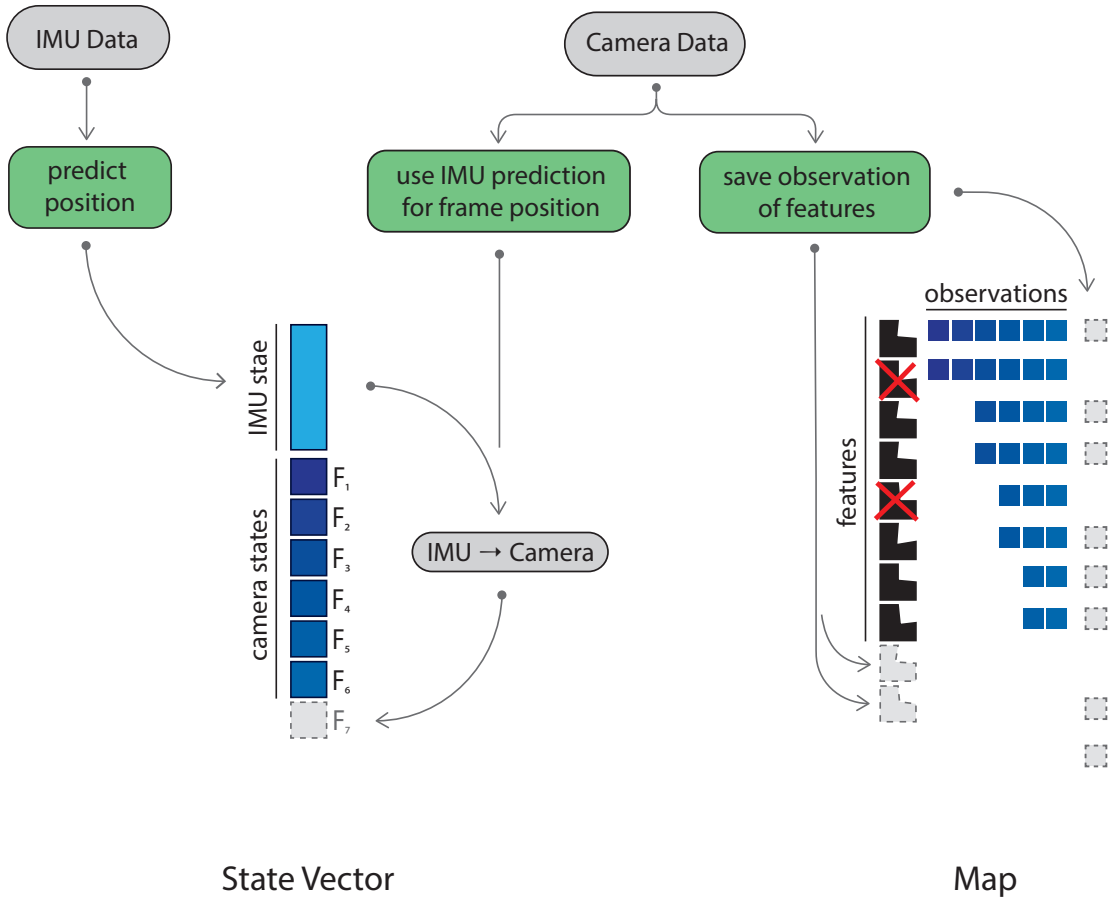

Figure 8: Data storage in the MSCKF - the IMU data is used in the prediction of the current position. This position is used for each camera frame position. Feature observations in camera frames are added to the respective feature.

The map, therefore, saves multiple features, each with a list of measurements u and v regarding a certain camera frame where the feature was seen. The measurement of a feature from one camera frame is described as z:

$$z = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}. \tag{18}$$

These observations saved in the feature, directly link to the camera states in the state vector itself. This is represented by the corresponding shades of blue in figure 8. The IMU state and the camera states in the state vector are a *best guess* as to where the agent is currently positioned and as to where the observations in the feature map were measured from.

4.3.2 Error-State Vector

Recall now, that the MSCKF is based on the Error-state Kalman Filter as described in chapter 2.4. We will, therefore, need a description of the error-state vector:

$$\tilde{\mathbf{x}} = [\ \tilde{\mathbf{x}}_{IMU}^\top\ \tilde{\mathbf{x}}_{C_1}^\top\ \tilde{\mathbf{x}}_{C_2}^\top\ ...\ \tilde{\mathbf{x}}_{C_n}^\top\ ]^\top.$$

The IMU state is the vector

$$\tilde{\mathbf{x}}_{IMU} = [\ \delta_G^I\boldsymbol{\theta}^\top\ \tilde{\mathbf{b}}_g^\top\ {}^G\tilde{\mathbf{v}}_I^\top\ {}_C^I\tilde{\mathbf{b}}^\top\ {}^G\tilde{\mathbf{p}}_I^\top\ \delta\,{}_C^I\boldsymbol{\theta}^\top\ {}^I\tilde{\mathbf{p}}_C^\top\ ]^\top. \tag{19}$$

This description of the error state is, for the most part, straightforward. The error state is equal to the difference between the true state and the nominal state. We define the general composite parameter $\oplus$ to describe the relation between the state types

$$\mathbf{x} = \hat{\mathbf{x}} \oplus \tilde{\mathbf{x}}.$$

This composite of the nominal and the error state can be exemplified by the position vector and the rotation quaternion:

$$\mathbf{p} = \hat{\mathbf{p}} + \tilde{\mathbf{p}} \tag{20}$$

$$q = \delta q \otimes \hat{q}. \tag{21}$$

For the position vector $\mathbf{p}$, the error state is simply $\tilde{\mathbf{p}} = \mathbf{p} - \hat{\mathbf{p}}$. To eliminate constraints on the covariance matrix, we want the orientation error $\tilde{q} \equiv \delta q$ to have a minimal representation, which we achieve through the small-angle approximation of the quaternion (eq. 3). This approximation can be used, as the small-angle assumption holds for the error-state estimation.

4.3.3 Control Vector

The necessary input to propagate the *estimated* state (eq. 8) through propagating the *error*-state (eq. 9) is typically called the control vector $\mathbf{u}$. The prediction in the MSCKF is based on IMU measurements, which define the vector's description

$$\mathbf{u} = \begin{bmatrix} \boldsymbol{\omega}_m \\ \boldsymbol{a}_m \end{bmatrix}$$

with $\boldsymbol{\omega}_m$ as the angular velocity measurements of the gyroscope and $\boldsymbol{a}_m$ as the acceleration measurements of the accelerometer.

4.3.4 Noise Vector

Remember, that noise in the EKF construct must always have a mean of zero - therefore, these values are generally omitted or ignored in the state *estimation* formulation. These noise values nonetheless influence the covariance matrix. The IMU noise values are labeled as $\mathbf{n} = [\mathbf{n}_g \ \mathbf{n}_{wg} \ \mathbf{n}_a \ \mathbf{n}_{wa}]$ which are the gyroscope noise, the gyroscope bias noise, the accelerometer noise and the accelerometer bias noise respectively. Refer to appendix C for more details on IMU noise.

4.4 IMU Propagation

This part of the algorithm uses the accelerometer and gyroscopic data from the IMU sensor (see appendix C) to predict the movement of the agent. We will derive the continuous-time process model based on an error-state description. This reformulation into a valid error state will take up the majority of the section. The resulting process model is then integrated into discrete-time.

4.4.1 Continuous-Time Process Model

Following the definition of the state vector and error-state vector, a continuous-time process model is derived for both, which will subsequently allow us to describe the discrete-time state transition functions eq. 8 and eq. 9. This section parallels the process model derivation in [22], where a general IMU process model is derived for an Error-state Kalman Filter. Note that this chapter neglects the influence of the earth's rotation on the IMU [20] and uses ijk quaternion nomenclature, see chapter 2.2.

We aim to describe the state dynamics as a function vector based on the current state and the received measurements, a continuous-time description of eq. 7:

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}).$$

This function vector will be stated first, derivations, definitions and proofs follow. Note, that this function assumes a simultaneous collection and receiving of the gyroscope and accelerometer information, as both data are processed conjointly. The elements in the vector correspond to

$$\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u}, \mathbf{n}) \tag{22}$$

$$ {}^{I}_{G}\dot{q} = \frac{1}{2}\,[\boldsymbol{\omega}_m - \mathbf{b}_w - \mathbf{n}_g]_L \cdot {}^{I}_{G}q \tag{a}$$

$$\dot{\mathbf{b}}_g = \mathbf{n}_{wg} \tag{b}$$

$$ {}^{G}\dot{\mathbf{v}}_I = C({}^{I}_{G}q) \cdot (\mathbf{a}_m - \mathbf{b}_a - \mathbf{n}_a) + {}^{G}\mathbf{g} \tag{c}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{wa} \tag{d}$$

$$ {}^{G}\dot{\mathbf{p}}_I = {}^{G}\mathbf{v} \tag{e}$$

$$ {}^{I}_{C}\dot{q} = \mathbf{0}_{3x1} \tag{f}$$

$$ {}^{I}\dot{\mathbf{p}}_C = \mathbf{0}_{3x1} \tag{g}$$

Eq. 22b-22g are straightforward. The function $C(\cdot)$ in eq. 22c transforms the quaternion into a rotation matrix, as described in appendix 2.2. The derivation of eq. 22a uses the quaternion rotation matrix equivalency of eq. 1 in chapter 2.2. Visually interpreted this function transforms the change in rotation ω into the current rotation ${}^{I}_{G}q$. Note that eq. 22a and eq. 22c subtract the current biases $\mathbf{b}$ - and respective noise - from the sensor measurements.

The goal of the Error-state Kalman Filter is to have a noise-less description of the *nominal* state - every uncertainty around the nominal state must be described by the *error state*. Referencing the description of the true state in eq. 22 we can easily describe the nominal state without random disturbances from the noise vector $\mathbf{n}$:

$$\dot{\hat{\mathbf{x}}} = f(\mathbf{x}, \mathbf{u}) \tag{23}$$

$$ {}^{I}_{G}\dot{\hat{q}} = \frac{1}{2}\,[\boldsymbol{\omega}_m - \hat{\mathbf{b}}_w]_L \cdot {}^{I}_{G}\hat{q} \tag{a}$$

$$\dot{\hat{\mathbf{b}}}_g = \mathbf{0}_{3x1} \tag{b}$$

$$ {}^{G}\dot{\hat{\mathbf{v}}}_I = C({}^{I}_{G}q) \cdot (\mathbf{a}_m - \mathbf{b}_a) + {}^{G}\mathbf{g} \tag{c}$$

$$\dot{\hat{\mathbf{b}}}_a = \mathbf{0}_{3x1} \tag{d}$$

$$ {}^{G}\dot{\hat{\mathbf{p}}}_I = {}^{G}\mathbf{v} \tag{e}$$

$$ {}^{I}_{C}\dot{\hat{q}} = \mathbf{0}_{3x1} \tag{f}$$

$$ {}^{I}\dot{\hat{\mathbf{p}}}_C = \mathbf{0}_{3x1} \tag{g}$$

Note, that this model is the continuous-time description of the error-state equation (eq. 8). In the following chapter 4.4.2 we will integrate these equations into their necessary discrete-time formulation.

The change in the error state vector from eq. 19 can then be approximated by using the

inverse of the composition operator which will be described as $\oplus^{-1} \equiv \ominus$ and using eq. 22 and eq. 23:

$$\tilde{\mathbf{x}} = \mathbf{x} \ominus \hat{\mathbf{x}}.$$

This results in the following linear equations (eq. 24).

$$\dot{\tilde{\mathbf{x}}}_{IMU} = f_e(\mathbf{x}, \delta\mathbf{x}, \mathbf{u}, \mathbf{n}) \tag{24}$$

$$\delta\, {}^I_G\dot{\boldsymbol{\theta}} = -[\omega_m - \hat{\mathbf{b}}_w]_\times \cdot \delta^I_G\boldsymbol{\theta} - \tilde{\boldsymbol{\omega}}_b - \mathbf{n}_w \tag{a}$$

$$\dot{\tilde{\mathbf{b}}}_g = \mathbf{n}_{wg} \tag{b}$$

$${}^G\dot{\tilde{\mathbf{v}}}_I = -C({}^I_Gq)\ (\mathbf{a}_m - \mathbf{b}_a)\ \delta^I_G\dot{\boldsymbol{\theta}} - C({}^I_Gq)\tilde{\mathbf{b}}_w - \mathbf{n}_a \tag{c}$$

$${}^I_C\dot{\tilde{\mathbf{b}}} = \mathbf{n}_{wa} \tag{d}$$

$${}^G\dot{\tilde{\mathbf{p}}}_I = {}^G\tilde{\mathbf{v}} \tag{e}$$

$$\delta\, {}^I_C\dot{\boldsymbol{\theta}} = \mathbf{0}_{3x1} \tag{f}$$

$${}^I\dot{\tilde{\mathbf{p}}}_C = \mathbf{0}_{3x1} \tag{g}$$

Eq. 24b and eq. 24d are simply the transference of the random walk noise of the bias into the change of the bias. Eq. 24f and eq. 24g show that these rotation and translation parameters between $\{I\}$ and $\{C\}$ are considered static. In 24e, the term ${}^G\tilde{\mathbf{v}}$ is the result of the integration of the continuous changes in eq. 24c. Eq. 24a and eq. 24b are subsequently expanded on. Before presenting the derivation of these equations, the next paragraph will give a visual analysis of the resulting error terminology.

Compare first the error description of orientation and velocity to the true state description in eq. 22a and 22c to see, that the resulting terms reduce the measured values. Further, note that the noise terms $\mathbf{n}_w$, $\mathbf{n}_a$ are removed from any rotation in the formulae. As the noise covariance is considered to be spherical, i.e. uniform in all directions, changes in rotation do not change the result. This only holds true, when the gyroscope and the accelerometer noise values are equivalent in all axis.

Looking at eq. 24a, the expression $-[\omega_m - \mathbf{b}_w]_\times \cdot \delta^I_G\boldsymbol{\theta}$ describes the error in the measured rotation based on the accumulated error so far; in the next term, $\tilde{b}_w$ is the error in the bias. Note, that any errors multiplied by errors are neglected, such as the error resulting from the error in the rotation due to the error in the bias. A similar analysis can be made for eq. 24c. The first term describes the resulting error in the acceleration vector due to the estimated error in rotation, the second term is the error in the bias. Both are rotated into the current *nominal* rotation of the IMU. Again, all terms describing the coupling of errors are neglected.

Sola et al. [22] provide an elegant description of the derivation regarding the rotation and velocity error, which are outlined in the following sections.

Rotation error derivation

This derivation partitions all the elements of uncertainty into one term. It uses the definitions of the true rotation time-derivative using the angular rate and the time-derivative of the compounded estimated rotation and the error rotation, to subsequently isolate the small-angle approximation of the error rotation.

By partitioning the true change in rotation into

$$\boldsymbol{\omega} = \hat{\boldsymbol{\omega}} + \delta\boldsymbol{\omega}$$

and by grouping elements with a covariance together

$$\hat{\boldsymbol{\omega}} \triangleq \boldsymbol{\omega}_m - \hat{\mathbf{b}}_w \tag{25}$$

$$\delta\boldsymbol{\omega} \triangleq -\tilde{\mathbf{b}}_w - \mathbf{n}_w \tag{26}$$

we can write the true and estimated change in rotation as a function of these terms:

$$\frac{d}{dt}(\hat{q} \otimes \delta q) = \dot{q} = \frac{1}{2}\, q \otimes \boldsymbol{\omega}$$

where we simplify $q \equiv {}^{I}_{G}q$ to ease readability. The right-hand calculations are the time derivatives of quaternions, see eq. 4. Using the definition of the true quaternion composition from eq. 20, this can be distributed and derived into both

$$\dot{\hat{q}} \otimes \delta q + \hat{q} \otimes \dot{\delta q} = \dot{q} = \frac{1}{2}\hat{q} \otimes \delta q \otimes \boldsymbol{\omega}_t$$

$$\frac{1}{2}\hat{q} \otimes \omega \otimes \delta q + \hat{q} \otimes \dot{\delta q} = \dot{q} = \frac{1}{2}\hat{q} \otimes \delta q \otimes \boldsymbol{\omega}_t$$

isolating δq and writing it in its small-angle form, we receive

$$\begin{bmatrix} 0 \\ \delta\boldsymbol{\theta} \end{bmatrix} = \delta q \otimes \boldsymbol{\omega}_t - \boldsymbol{\omega} \otimes \delta q$$

where only the vector part $\delta\boldsymbol{\theta}$ is of interest. Using the matrix quaternion representation from eq. 1 and dispersing the parameters from eq. 25, the final error description is

$$\boxed{\dot{\delta\boldsymbol{\theta}} = -\left[\boldsymbol{\omega}_m - \boldsymbol{\omega}_b\right]_\times \delta\boldsymbol{\theta} - \delta\boldsymbol{\omega}_b - \boldsymbol{\omega}_n}\,.$$

Linear velocity error derivation

Parallel to the error in rotation derived in the previous section, Sola et al. [22] provide a similar calculation for the linear velocity error. As with the rotation error derivation, the ijk notation is used for quaternions in the following derivation.

Again we separate the elements with covariance by defining

$$\begin{aligned} \mathbf{a} &\triangleq \mathbf{a}_m - \mathbf{b}_a \\ \delta\mathbf{a} &\triangleq -\delta\mathbf{b}_a - \mathbf{n}_a \end{aligned} \tag{27}$$

which we use to describe the true acceleration

$$\mathbf{a} = \mathbf{R}\left(\mathbf{a} + \delta\mathbf{a}\right) + \mathbf{g} + \delta\mathbf{g} \tag{28}$$

where $\mathbf{R}$ can be separated into

$$\mathbf{R} = \left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right)\hat{\mathbf{R}} + O\left(\|\delta\boldsymbol{\theta}\|^2\right). \tag{29}$$

Here we can marginalize the higher order error terms in $O\left(\cdot\right)$. Parallel to the rotation error, the $\dot{\mathbf{v}}$ term is split into two descriptions

$$\begin{aligned} \dot{\mathbf{v}} + \delta\dot{\mathbf{v}} = \dot{\mathbf{v}} &= \left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right)\mathbf{R}\left(\mathbf{a} + \delta\mathbf{a}\right) + \mathbf{g} + \delta\mathbf{g} \\ \mathbf{R}\mathbf{a} + \mathbf{g} + \delta\dot{\mathbf{v}} = \dot{\mathbf{v}} &= \mathbf{R}\mathbf{a}_+\mathbf{R}\delta\mathbf{a} + \mathbf{R}[\delta\boldsymbol{\theta}]_\times\mathbf{a} + \mathbf{R}[\delta\boldsymbol{\theta}]_\times\delta\mathbf{a} + \mathbf{g} + \delta\mathbf{g} \end{aligned}$$

which can be rearranged and - through marginalization of second order terms - reduces to

$$\delta\dot{\mathbf{v}} = \mathbf{R}\left(\delta\mathbf{a} - [\mathbf{a}]_\times\,\delta\boldsymbol{\theta}\right) + \delta\mathbf{g}$$

$$\boxed{\delta\dot{\mathbf{v}} = \; -\mathbf{R}\left[\mathbf{a}_m - \mathbf{b}_a\right]_\times\delta\boldsymbol{\theta} - \mathbf{R}\delta\mathbf{b}_a + \delta\mathbf{g} - \mathbf{n}_a}\,. \tag{30}$$

We here assume, that the covariance of $\mathbf{n}_a$ is a sphere, which allows us to write $\mathbf{R}\mathbf{n}_a$ as $\mathbf{n}_a$. Eq. 30 is the resulting velocity error, which has been used in eq. 24c.

4.4.2 Discrete-Time Prediction from IMU

The resulting motion models from the previous section are a continuous-time description. We will first define the discretization for the nominal state description, and subsequently the error state transition.

To integrate the function

$$\dot{\hat{\mathbf{x}}} = f(t, \mathbf{x}, \mathbf{u})$$

over a time Δt, we write the form:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \int_{k\Delta t}^{(k+1)\Delta t} f(\tau, \mathbf{x}(\tau), \mathbf{u}(\tau))d\tau.$$

This integration of the function $f(\cdot)$ can be approximated through various approaches. For ex-

ample, Sun et al. [45] use a fourth order, Mourikis et al. [20] a fifth order Runge Kutta iterative integration. As the time-steps between IMU samplings are expected to be sub $0.01s$, these estimations are considered accurate enough [22].

Calculating the discrete form of the differential equation

$$\dot{\tilde{\mathbf{x}}} = \mathbf{F}\tilde{\mathbf{x}} + \mathbf{G}\mathbf{n}$$

is generally possible in closed-form, as Sola [22] provides. Nonetheless, to calculate the (error-) state transition matrix Φ in the MSCKF, a numerical approach is used, due to its computational superiority.

4.5 State Augmentation

The state description in eq. 17 contained multiple vectors $\mathbf{x}_{C_i}$ regarding the camera pose, where i ranges from 1 to n, where n is the number of camera poses. As a new camera frame is introduced to the MSCKF, the nominal state vector is expanded by a camera state $\mathbf{x}_{C_{n+1}}$ at the end of the current state, using the current rotation quaternion and position vector from the nominal state of the IMU, and transforming this information from $\{I\}$ into $\{C\}$ through the corresponding current estimation.

$$\mathbf{x_{C\ell}} = [\ {}^{C}_{G}q^{\top}\ {}^{G}\mathbf{p}^{\top}_{C\ell}\ ]^{\top}$$

is defined by the functions

$${}^{C}_{G}q = {}^{C}_{I}q \otimes {}^{I}_{G}q \tag{31}$$

$${}^{G}\mathbf{p}_C = {}^{G}\mathbf{p}_I + C({}^{I}_{G}q)^{\top}\,{}^{I}\mathbf{p}_C. \tag{32}$$

Additionally, the error-state covariance matrix needs to be expanded as well (with the nominal state expansion, the error state naturally expands analogously). The result of this expansion should be, that the current uncertainty of the IMU is directly linked to the uncertainty of the newly added camera state. This is achieved through defining a Jacobian, which modulates the IMU state into the current camera state

$$\mathbf{P}_k \leftarrow \mathbf{J}\ \mathbf{P}_k\ \mathbf{J}^{\top}. \tag{33}$$

Figure 9 visualizes the process. We therefore desire the Jacobi Matrix $\mathbf{J}$ to have the following structure:

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_{21\times 21} & \mathbf{0}_{6\mathrm{n}\times 6\mathrm{n}} \\ \mathbf{J}_C & \mathbf{0}_{6\mathrm{n}\times 6} \end{bmatrix} \tag{34}$$

where we write the relevant term $\mathbf{J}_C$ as

$$\mathbf{J}_C = \frac{\partial \tilde{\mathbf{X}}_{C_i}}{\partial \tilde{\mathbf{X}}_{IMU}}$$

$$\mathbf{J}_C = \begin{bmatrix} C({}^C_I\hat{q}) & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} \\ [C({}^I_G\hat{q})\ {}^I\mathbf{p}_C)]_\times & \mathbf{0}_{3\times 9} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{I}_3 \end{bmatrix}. \tag{35}$$

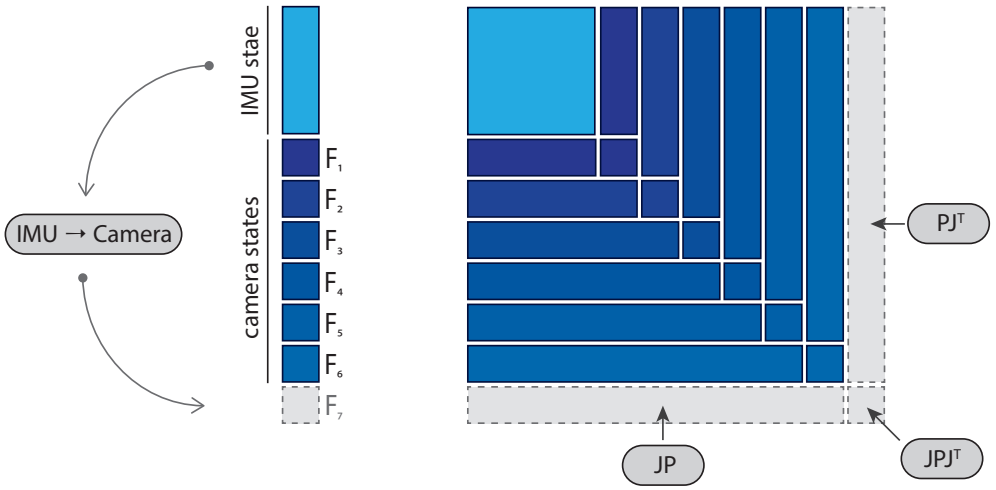

Figure 9: The state vector as well as the state covariance matrix are expanded using the current IMU state

To the best of our knowledge, no complete derivation of the state augmentation has been published. For this reason the following section, will derive the part of the $\mathbf{J}$ matrix which relates the position error in the IMU frame to the position error in C_ℓ. The derivation regarding the orientation error is shown afterwards.

We calculate the position error description by first writing the description of the true position state of the camera as a composite of the vectors in the true state:

$${}^G\mathbf{p}_C = {}^G\mathbf{p}_I + C({}^I_G q)^\top\ {}^I\mathbf{p}_C. \tag{36}$$

Additionally, we recall the definition of the true state position and rotation description as a composite of the nominal and the error state:

$$\mathbf{p} = \hat{\mathbf{p}} + \tilde{\mathbf{p}} \tag{37}$$

$$q = \delta q \otimes \hat{q}. \tag{38}$$

Using these error definitions, we can expand eq. 36

$$ {}^{G}\hat{\mathbf{p}}_C + {}^{G}\tilde{\mathbf{p}}_C = {}^{G}\hat{\mathbf{p}}_I + {}^{G}\tilde{\mathbf{p}}_I C(\delta^I_G q \otimes {}^I_G\hat{q})^\top \, ({}^I\hat{\mathbf{p}}_C + {}^I\tilde{\mathbf{p}}_C) \tag{39} $$

$$ {}^{G}\hat{\mathbf{p}}_I + C({}^I_G\hat{q})^\top \, {}^I\hat{\mathbf{p}}_C + {}^{G}\tilde{\mathbf{p}}_C = {}^{G}\hat{\mathbf{p}}_I + {}^{G}\tilde{\mathbf{p}}_I C(\delta^I_G q \otimes {}^I_G\hat{q})^\top \, ({}^I\hat{\mathbf{p}}_C + {}^I\tilde{\mathbf{p}}_C) \tag{40} $$

removing ${}^{G}\hat{\mathbf{p}}_I$ from both sides, reduces to:

$$ C({}^I_G\hat{q})^\top \, {}^I\hat{\mathbf{p}}_C + {}^{G}\tilde{\mathbf{p}}_C = {}^{G}\tilde{\mathbf{p}}_I + C(\delta^I_G q \otimes {}^I_G\hat{q})^\top \, ({}^I\hat{\mathbf{p}}_C + {}^I\tilde{\mathbf{p}}_C) $$

where we can split up the quaternion based matrix using:

$$ C(q \otimes p)^\top = C((q \otimes p)^*) = C(p^* \otimes q^*) = C(p^*)\, C(q^*) = C(p)^\top \, C(q)^\top $$

to get

$$ C({}^I_G\hat{q})^\top \, {}^I\hat{\mathbf{p}}_C + {}^{G}\tilde{\mathbf{p}}_C = {}^{G}\tilde{\mathbf{p}}_I + C({}^I_G\hat{q})^\top \, C(\delta^I_G q)^\top \, ({}^I\hat{\mathbf{p}}_C + {}^I\tilde{\mathbf{p}}_C). $$

Distributing over the sum leads to

$$ \begin{aligned} C({}^I_G\hat{q})^\top \, {}^I\hat{\mathbf{p}}_C + {}^{G}\tilde{\mathbf{p}}_C &= {}^{G}\tilde{\mathbf{p}}_I + C({}^I_G\hat{q})^\top \, C(\delta^I_G q)^\top \, {}^I\hat{\mathbf{p}}_C + C({}^I_G\hat{q})^\top \, C(\delta^I_G q)^\top \, {}^I\tilde{\mathbf{p}}_C \\ {}^{G}\tilde{\mathbf{p}}_C &= {}^{G}\tilde{\mathbf{p}}_I + C({}^I_G\hat{q})^\top \, C(\delta^I_G q)^\top \, {}^I\hat{\mathbf{p}}_C + C({}^I_G\hat{q})^\top \, C(\delta^I_G q)^\top \, {}^I\tilde{\mathbf{p}}_C - C({}^I_G\hat{q})^\top \, {}^I\hat{\mathbf{p}}_C. \end{aligned} \tag{41} $$

We can use the following relation from [22]

$$ C(q) = (q_w^2 - q_v^\top q_v)\mathbf{I} + 2q_v q_v^\top + 2q_w[q_v]_\times $$

to simplify $C(\delta^I_G q)^\top$ under the assumption of small-angle quaternions and using the definition of conjugate quaternions

$$ \begin{aligned} C(\delta^I_G q)^\top &\simeq C(\begin{bmatrix} \frac{1}{2}{}^I_G\tilde{\Theta} \\ 1 \end{bmatrix})^\top = C(\begin{bmatrix} -\frac{1}{2}{}^I_G\tilde{\Theta} \\ 1 \end{bmatrix}) = \\ &= (1^2 - \frac{1}{2}{}^I_G\tilde{\Theta}^\top \frac{1}{2}{}^I_G\tilde{\Theta})\mathbf{I} + 2\frac{1}{2}{}^I_G\tilde{\Theta}\frac{1}{2}{}^I_G\tilde{\Theta}^\top - 2 \cdot 1[\frac{1}{2}{}^I_G\tilde{\Theta}]_\times \end{aligned} $$

where we marginalize any products of errors, resulting in

$$ \begin{aligned} &= (1-0)\mathbf{I} + 0 + [{}^I_G\tilde{\Theta}]_\times \\ C(\delta^I_G q)^\top &\simeq \mathbf{I} - [{}^I_G\tilde{\Theta}]_\times. \end{aligned} $$

This we inject back into eq. 41 and distribute over the second subtraction.

$$
\begin{aligned}
{}^{G}\tilde{\mathbf{p}}_C &= {}^{G}\tilde{\mathbf{p}}_I + C({}^{I}_{G}\hat{q})^{\top}\left(\mathbf{I} - [{}^{I}_{G}\tilde{\Theta}]_{\times}\right){}^{I}\hat{\mathbf{p}}_C + C({}^{I}_{G}\hat{q})^{\top}\left(\mathbf{I} - [{}^{I}_{G}\tilde{\Theta}]_{\times}\right){}^{I}\tilde{\mathbf{p}}_C - C({}^{I}_{G}\hat{q})^{\top}\,{}^{I}\hat{\mathbf{p}}_C \\
&= {}^{G}\tilde{\mathbf{p}}_I + C({}^{I}_{G}\hat{q})^{\top}\,C(\delta{}^{I}_{G}q)^{\top}\,{}^{I}\hat{\mathbf{p}}_C + C({}^{I}_{G}\hat{q})^{\top}\,{}^{I}\tilde{\mathbf{p}}_C - C({}^{I}_{G}\hat{q})^{\top}\,[{}^{I}_{G}\tilde{\Theta}]_{\times}\,{}^{I}\tilde{\mathbf{p}}_C - C({}^{I}_{G}\hat{q})^{\top}\,{}^{I}\hat{\mathbf{p}}_C
\end{aligned}
$$

which reduces to

$$
= {}^{G}\tilde{\mathbf{p}}_I + C({}^{I}_{G}\hat{q})^{\top}\left(\mathbf{I} - [{}^{I}_{G}\tilde{\Theta}]_{\times}\right){}^{I}\hat{\mathbf{p}}_C - C({}^{I}_{G}\hat{q})^{\top}\,[{}^{I}_{G}\tilde{\Theta}]_{\times}\,{}^{I}\tilde{\mathbf{p}}_C.
$$

Another distribution and the double-error marginalization $C({}^{I}_{G}\hat{q})^{\top}\,[{}^{I}\tilde{\mathbf{p}}_C]_{\times}\,{}^{I}_{G}\tilde{\Theta} = 0$ reduces the calculation further. Using the cross-product matrix relation $[a]_x b = -[b]_x a$, we can write the position error transformation as

$$
\boxed{{}^{G}\tilde{\mathbf{p}}_C = {}^{G}\tilde{\mathbf{p}}_I + C({}^{I}_{G}\hat{q})^{\top}\,[{}^{I}\tilde{\mathbf{p}}_C]_{\times}\,{}^{I}_{G}\tilde{\Theta} + C({}^{I}_{G}\hat{q})^{\top}\,{}^{I}\tilde{\mathbf{p}}_C}\,.
$$

Now that we have a description of the position error transformation in $\mathbf{J}$, the following calculations will show how to process the orientation from IMU to C frame.

$$
{}^{C}_{G}q = {}^{C}_{I}q \otimes {}^{I}_{G}q = \delta q \otimes \hat{q}
$$

These two descriptions of the rotation state can be split up into

$$
\begin{aligned}
\delta{}^{C}_{G}q \otimes {}^{C}_{G}\hat{q} &= (\delta{}^{C}_{I}q \otimes {}^{C}_{I}\hat{q}) \otimes (\delta{}^{I}_{G}q \otimes {}^{I}_{G}\hat{q}) \\
\delta{}^{C}_{G}q \otimes ({}^{C}_{I}\hat{q} \otimes {}^{I}_{G}\hat{q}) &= (\delta{}^{C}_{I}q \otimes {}^{C}_{I}\hat{q}) \otimes (\delta{}^{I}_{G}q \otimes {}^{I}_{G}\hat{q}).
\end{aligned}
$$

Quaternion multiplying ${}^{I}_{G}\hat{q}^{*}$ and ${}^{C}_{I}\hat{q}^{*}$ from the right as well as removing brackets due to associativity of quaternion multiplication leads to

$$
\begin{aligned}
\delta{}^{C}_{G}q \otimes {}^{C}_{I}\hat{q} &= \delta{}^{C}_{I}q \otimes {}^{C}_{I}\hat{q} \otimes \delta{}^{I}_{G}q \\
\delta{}^{C}_{G}q &= \delta{}^{C}_{I}q \otimes {}^{C}_{I}\hat{q} \otimes \delta{}^{I}_{G}q \otimes {}^{C}_{I}\hat{q}^{*}.
\end{aligned}
$$

We subsequently simplify the error quaternion through a small-angle approximation:

$$
\begin{bmatrix} \frac{1}{2}{}^{C}_{G}\delta\Theta \\ 1 \end{bmatrix} = \begin{bmatrix} -\frac{1}{2}{}^{C}_{I}\delta\Theta \\ 1 \end{bmatrix} \otimes {}^{C}_{I}\hat{q} \otimes \begin{bmatrix} \frac{1}{2}{}^{I}_{G}\delta\Theta \\ 1 \end{bmatrix} \otimes {}^{C}_{I}\hat{q}^{*}.
$$

Through two alternate quaternion descriptions

$$
q \otimes p \otimes q^{*} = \begin{bmatrix} -C(q) \\ 0 \end{bmatrix}
$$

$$q = [\mathbf{v}\ w] = \mathbf{v} + w$$

we receive

$$\frac{1}{2}\delta^{C}_{G}\Theta + 1 = \frac{1}{2}{}^{C}_{I}\delta\Theta + 1 \cdot C({}^{C}_{I}\hat{q})\ \frac{1}{2}\ \delta^{I}_{G}\Theta + 1.$$

Multiplying, gives us the description

$$\frac{1}{2}{}^{C}_{G}\delta\Theta + 1 = \frac{1}{2}{}^{C}_{I}\delta\Theta\ C({}^{C}_{I}\hat{q})\ \frac{1}{2}\ {}^{I}_{G}\delta\Theta + C({}^{C}_{I}\hat{q})\ \frac{1}{2}\ {}^{I}_{G}\delta\Theta + \frac{1}{2}{}^{C}_{I}\delta\Theta + 1$$

which we reduce and marginalize the first term, which is an error multiplication

$$\boxed{{}^{C}_{G}\delta\Theta = C({}^{C}_{I}\hat{q})\ \ {}^{I}_{G}\delta\Theta + {}^{C}_{I}\delta\Theta}\ .$$

This is the error description used to calculate the previously introduced Jacobi.

4.6 Measurement Update

Propagating the IMU measurements accumulates unknown errors in the error state. The measurements of a second sensor renders these errors observable. This error measurement can then be used to correct the initial propagation estimation of the IMU, see eq. 9 - 13. The measurement update in the MSCKF filter is based on observation of features. These features are tracked over a period of time, in multiple camera frames. More specifically, the update process is triggered, either, once the feature is no longer tracked, or the moving window of camera states overflows, thereby removing a camera pose associated with features which were still being tracked. Every time a new frame arives from the camera, the following steps are taken:

1. the current IMU position estimation is saved in the state vector as a new camera frame (dotted gray elements in figure 10)
2. if one of either dependencies is fulfilled, a measurement update is triggered (orange elements in figure 10)
   - a feature is no longer tracked or
   - a camera frame is removed from the current state.

Figure 10 shows a visualization of how some features are triggered for an update step. The orange camera state is no longer inside the moving window. Therefore all features observed in this frame are used as an update. This includes the top two features in the map (which are also marked in orange). Additionally, two features are no longer observed in the newest frame in the moving window. Both these features do not have a grey dotted camera state associated with them. Both features are marked orange (one of them had already been selected through

falling out of the moving window). A total of three features are therefore used to update the IMU prediction.

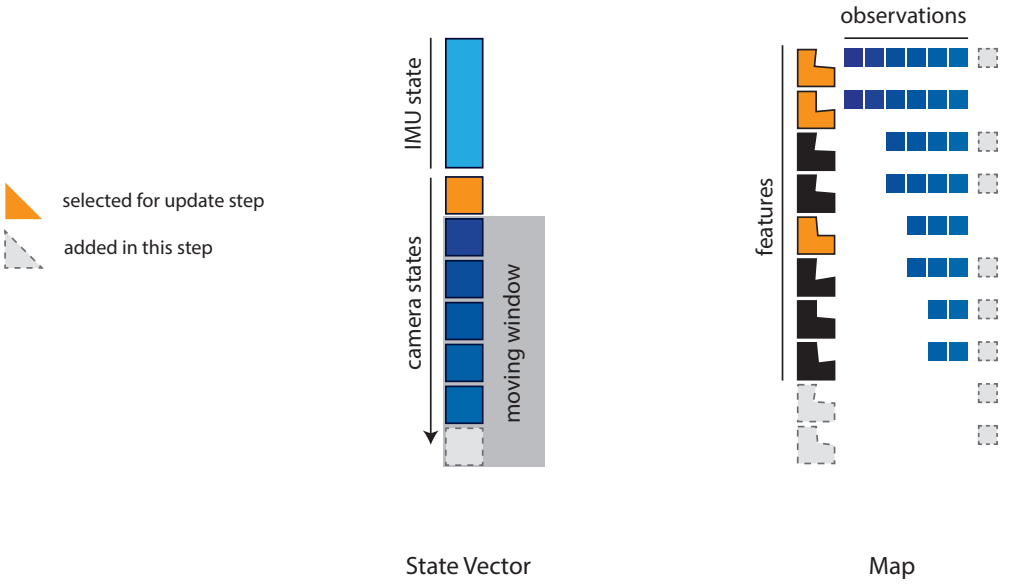

Figure 10: Update step in the MSCKF - all features observed in the camera frame removed from the moving window and all features not observed in the new camera frame are used in the update

In the measurement update, the errors accumulated during the integration of the IMU motion model are observed. This observation is done through comparing the estimated movement to the movement of the tracked features. Every feature which is either lost or has been marginalized by reducing the number of saved states, triggers the following steps in the update cycle:

1. the position of the feature is estimated based on its observations
2. based on this estimate, the expected measurements per camera frame are calculated
3. these results are compared to the true measurements and the residual is formed and
4. the residual is projected onto the nullspace of the feature-position Jakobi of the measurement model, to alleviate any dependency outside the error-stat estimate.

For a general overview of the measurement update, refer back to fig. 4, where the feature position estimation and the resulting residual through reprojection are portrayed.

4.6.1 Feature Position Estimation

The feature position in G is estimated using both the observations in the different camera frames, and the estimated relative pose change between those frames. Mourikis et al. [20] opted for a Gauss-Newton algorithm, Sun et al. [45] use the more robust Levenberg-Marquardt solver; see Gill and Murray [73] for further details on the algorithms.

The initial estimation for this depth value is calculated by using the first and the last of all observations, see figure 18 and by constructing a least square problem:

$$\mathbf{p} = \begin{bmatrix} u & v & 1 \end{bmatrix}^\top \mathrm{d}$$
$$\mathbf{p}' = \mathbf{R} \begin{bmatrix} u' & v' & 1 \end{bmatrix}^\top \mathrm{d} - \mathbf{t}$$

where $\mathbf{p}$ is the feature position in the anchor camera frame and $\mathbf{p}'$ is the feature position in the last camera frame. This can be reformulated into the feature observation in the second camera frame, by dividing by the depth of $\mathbf{p}'$, and by defining

$$\mathbf{m} = \mathbf{R} \begin{bmatrix} u & v & 1 \end{bmatrix}$$

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} m(0)\mathrm{d} - t(0) \\ m(1)\mathrm{d} - t(1) \end{bmatrix} \frac{1}{m(2)\mathrm{d} - t(2)}$$

$$\Longrightarrow \begin{bmatrix} m(0) - m(2)u' \\ m(1) - m(2)v' \end{bmatrix} \mathrm{d} = \begin{bmatrix} t(0) - t(2)u' \\ t(1) - t(2)v' \end{bmatrix}.$$

This resulting depth is then used as an initial estimation for the previously discussed algorithms. The resulting feature position is a local minimum estimation of the feature position in G.

4.6.2 Feature Reprojection

This estimated feature position from the previous section is based both on the IMU prediction and the feature position estimation. To describe the difference between the measurements and the prediction, we look at the *projections* of this estimated feature position in the individual camera frames. To achieve this, first the feature position ${}^G\mathbf{p}_f$ is transformed into the respective camera frame C_ℓ and projected into the focal plane. The steps are visualized in figure 11, the following calculations describe the process based on the pinhole camera model from chapter 2.

$${}^C\hat{\mathbf{p}}_f = C({}^G_C\hat{q})({}^G\hat{\mathbf{p}}_f - {}^G\hat{\mathbf{p}}_C) \tag{42}$$

$$\hat{\mathbf{z}} = \begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \begin{bmatrix} \frac{{}^C\hat{x}_f}{{}^C\hat{z}_f} \\ \frac{{}^C\hat{y}_f}{{}^C\hat{z}_f} \end{bmatrix} \tag{43}$$

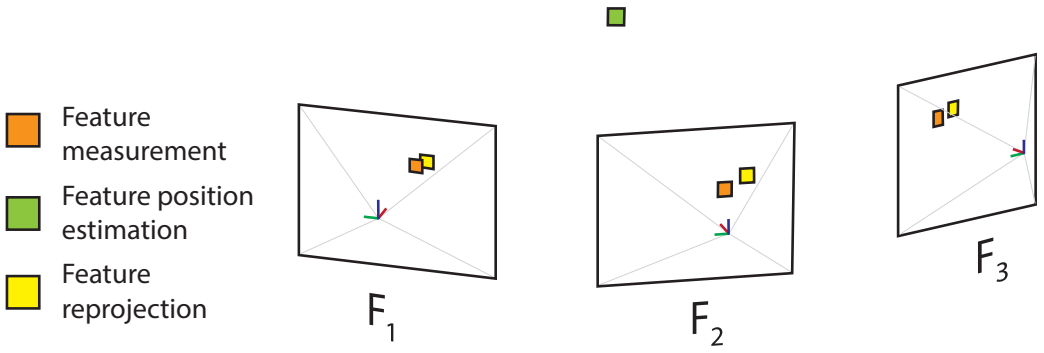

Figure 11: The feature position in the world frame is first estimated and then re-projected into the frames

4.6.3 Residual

The residual of the MSCKF is the difference between the position of the actual feature measurement and this feature projection in the camera frame. The exact formulation can be written as

$$\mathbf{r}_{i,\ell} = \mathbf{z}_{i,\ell} - \hat{\mathbf{z}}_{i,\ell} \tag{44}$$

where $\mathbf{z}$ is the original measurement of the feature in the frame, and $\hat{\mathbf{z}}$ is the reprojection of the feature into the frame, as defined in eq. 18 and eq. 42 respectively. The i is the index of the currently considered feature and the ℓ represents one of the frames, in which the feature was observed in. An example of feature measurement and projection using the TUM VIO dataset [74] can be seen in figure 12. Note, that this residual works, because the *expected* result of accumulated measurements is the actual position of the feature. This is only true, because of a Gaussian probability inherent in the IMU prediction and the measurement noise.

This residual is a function of the projection functions in eq. 42, which itself is a function of the feature position and the respective camera position. The residual of the feature i as presented here is calculated for every frame ℓ in which it was observed. Refer to figure 10, where a feature is selected as an update step (in orange) and all its observations are included in the update step. Every individual frame has a unique residual of the form of a 2×1 matrix. These *frame* residuals are stacked vertically, to form the full residual of the considered *feature*, as eq. 45 shows.

$$\mathbf{r}_i = \begin{bmatrix} \mathbf{r}_{i,0}^\top \, \mathbf{r}_{i,1}^\top \, \mathbf{r}_{i,2}^\top \, ... \, \mathbf{r}_{i,N}^\top \end{bmatrix}^\top \tag{45}$$

where N is the number of frames the residual is based on. The resulting size of the matrix is $2N \times 1$. Remember that this stacked residual is dependent on information in the state vector *and the feature position estimation*, which is not represented in the state vector.

The unabridged residual calculation of the MSCKF is

$$\mathbf{r}_{i,\ell} = \mathbf{z}_{i,\ell} - \begin{bmatrix} \frac{{}^{C_\ell}\hat{x}_f}{{}^{C_\ell}\hat{z}_f} \\ \frac{{}^{C_\ell}\hat{y}_f}{{}^{C_\ell}\hat{z}_f} \end{bmatrix} + \mathbf{n}_{i,\ell} \tag{46}$$

$$\begin{bmatrix} {}^{C_\ell}\hat{x}_f \\ {}^{C_\ell}\hat{y}_f \\ {}^{C_\ell}\hat{z}_f \end{bmatrix} = C({}^{G}_{C_\ell}\hat{q})({}^{G}\hat{\mathbf{p}}_f - {}^{G}\hat{\mathbf{p}}_{C_\ell}). \tag{47}$$

Figure 12: The feature measurement and projection; the distance between the two points in the image is the residual of the MSCKF

4.6.4 Update

To update the MSCKF, some prep-work is necessary. The main elements are linearizing the residual formulation, removing the feature dependent information, stacking the results and removing any remaining noise parameters.

As seen in chapter 2.3 eq. 5, the residual calculation needs a linearized description, to properly propagate the state covariance matrix. Therefore, the residual from eq. 46 is linearized in the following fashion:

$$\mathbf{r}_{i,\ell} = \mathbf{H}_{\mathbf{C}_\ell}\tilde{\mathbf{x}}_{\mathbf{C}_\ell} + \mathbf{H}_{f,\ell}{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n}_{i,\ell} \tag{48}$$

where $\tilde{\mathbf{x}}_{C_\ell}$ is the error state of the camera frame ℓ, $\mathbf{H}_{\mathbf{C}_\ell}$ is the Jacobian of the residual with respect to this state and $\mathbf{H}_{f,\ell}$ is the Jacobian of the residual of the current frame ℓ with respect to the feature estimation. These two Jacobians are the partial derivatives of eq. 46.

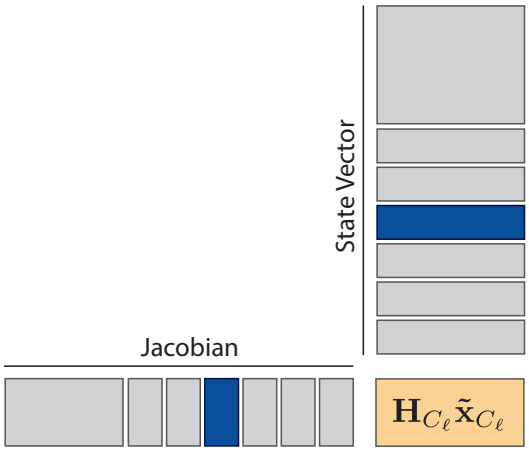

Figure 13: Expanded Jacobian matrix multiplication with state vector

Comparing this residual of eq. 46 to the description from eq. 5, we must first rewrite the term regarding $\tilde{\mathbf{x}}_{\mathbf{C}_\ell}$ into a proper state-vector dependent form:

$$\mathbf{r}_{i,\ell} = \mathbf{H}_{\mathbf{x},\ell}\tilde{\mathbf{x}} + \mathbf{H}_{f,\ell}{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n}_{i,\ell}. \tag{49}$$

As the term is a part of the state vector (see eq. 17), this can easily be done by sectioning the corresponding Jacobian $\mathbf{H}_{\mathbf{C}_\ell}$ into its part of a larger Jacobian $\mathbf{H}_{\mathbf{x}}$, as figure 13 provides.

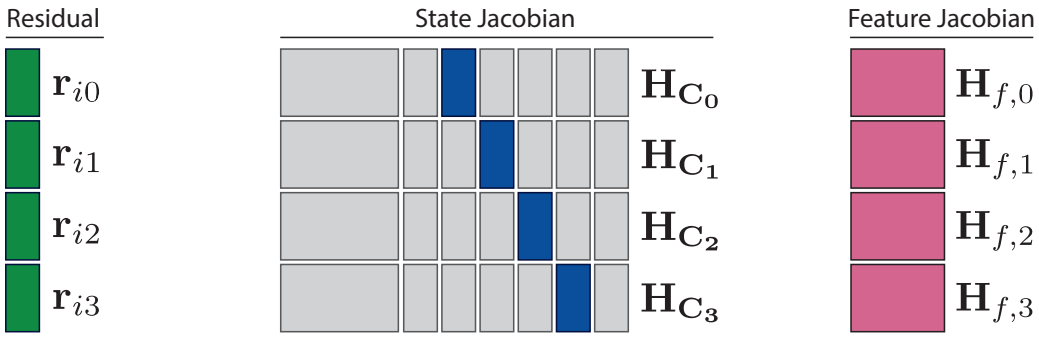

Figure 14: Vertical stacking of the Jacobians and residuals

These resulting Jacobians and residuals per frame are now stacked vertically, see figure 14. The entire information regarding this specific feature is now blocked together, represented by

$$\mathbf{r}_i = \mathbf{H}_\mathbf{x}\tilde{\mathbf{x}} + \mathbf{H}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{n}_i.$$

Another noticeable difference to the formulation in chapter 2.3 is the second parameter ${}^G\tilde{\mathbf{p}}_f$, which is not a part of the state vector. Note, that this parameter is nevertheless *correlated* with the state vector, as its parameter estimations are used to calculate the estimate ${}^G\hat{\mathbf{p}}_f$. Due to this correlation, the section $\mathbf{H}_f {}^G\hat{\mathbf{p}}_f$ cannot be partitioned into the uncorrelated Gaussian noise $\mathbf{n}$ and must be dealt with differently.

In order to still be able to use the residual and state vector, Mourikis et al. [20] project the residual $\mathbf{r}$ and Jacobian of the state-vector $\mathbf{H}_\mathbf{x}$ onto the *left nullspace* of the Jacobian for the feature position $\mathbf{H}_f$. This yields the correct residual and propagation matrix for the defined state vector. The nullspace of a matrix $\mathbf{A}$ is the set of all matrices for which

$$\begin{aligned} \mathbf{V} &= Null(\mathbf{A}) \\ \mathbf{A}\mathbf{V} &= \mathbf{0}. \end{aligned}$$

The *left* nullspace can be calculated, such that

$$\begin{aligned} \mathbf{V} &= Null(\mathbf{A}^\top) \\ \mathbf{V}^\top\mathbf{A} &= \mathbf{0}. \end{aligned}$$

This is the matrix form we need to properly project the linearized residual description of eq. 48 to remove the feature estimation from the equation. For this reason, we define $\mathbf{V}$ as

$$\mathbf{V} = Null(\mathbf{H}_f^\top)$$

which makes it the unitary matrix forming a basis of the left nullspace of $\mathbf{H}_f$. Note, that the explicit values in $\mathbf{V}$ differ depending on the actual implementation - as there are an infinite number of basis vector of the nullspace. Nevertheless, any projection onto the nullspace is adequate to create a *valid* residual description for the state vector we have defined. Using this $\mathbf{V}$ we can now calculate

$$\begin{aligned} \mathbf{r}_i &= \mathbf{H}_\mathbf{x}\tilde{\mathbf{X}} + \mathbf{H}_f {}^G\hat{\mathbf{p}}_f + \mathbf{n}_{i,\ell} \\ \mathbf{V}^\top\mathbf{r}_{i,\ell} &= \mathbf{V}^\top\mathbf{H}_\mathbf{x}\tilde{\mathbf{X}} + \mathbf{V}^\top\mathbf{H}_f {}^G\hat{\mathbf{p}}_f + \mathbf{V}^\top\mathbf{n}_i \\ \mathbf{r}_i^0 &= \mathbf{H}_\mathbf{x}^0\tilde{\mathbf{X}} + \mathbf{0} + \mathbf{n}_{i,\ell}^0 \end{aligned} \tag{50}$$

in which we have defined $\mathbf{V}^\top\mathbf{r}_i$ as $\mathbf{r}_i^0$, $\mathbf{V}^\top\mathbf{H}_\mathbf{x}$ as $\mathbf{H}_\mathbf{x}^0$ and an equivalent renaming concerning the noise parameter.

If the noise parameters in the implementation are designed to be some scalar multiple of the identity matrix,

$$\mathbf{n} = \text{scalar} \times \mathbf{I}$$

then any reprojection of this term yields the same entries. The *dimension* of the noise matrix changes nonetheless. Should the noise parameters be of elliptical form (non-identity matrix), they will need to be reprojected regardless.

The original residual $\mathbf{r}_i$ describes the stacked residual of all camera frames the feature was observed in. It is important to point out, that the nullspaced residual can no longer be interpreted as the distances between measurements and projections on the various image planes. Rather it represents the difference between the feature measurements and the movements of the camera directly. There is not necessarily an intuitive visual representation for this residual. Barring linearization inaccuracies, it is nevertheless an optimal residual and Jacobian, describing the change of the camera position in the state-vector.

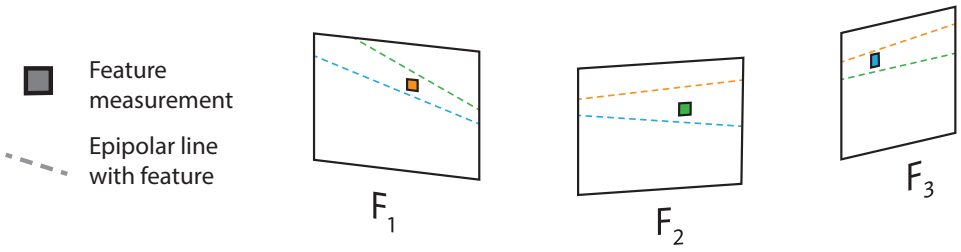

Figure 15: Alternative residual from epipolar constraints - each dotted line is generated using two camera position estimations and the feature measurement of the same color

Regarding the resulting constraints induced by $\mathbf{r}^0$ - with the number of frames considered for the feature residual being N, the size of the Jacobian $\mathbf{H}_f$ is $2N \times 3$ and (disregarding exceptions) generally has full column rank. The left nullspace V therefore has dimension $2N - 3$, which translates to r^0 being of size $2N-3 \times 1$ [75]. Mourikis et al. [20] state, that the same dimension of constraint is created, when constructing the residual based on the *epipolar* constraints between the images, as the same measurements are used multiple times, correlating the results. Figure 15 visualizes the general idea. The residual is the distance between the measured feature and the epipolar lines. According to the paper, this residual is both more cumbersome to implement and yields inferior results compared to the nullspaced residual.

Now that we have a valid residual and Jacobian, it is possible to use it as-is for the update step of the MSCKF. Analogous to chapter 2.4 on the Error-state Kalman Filter, the update to the IMU based state propagation follows. Multiple features can be selected for the update step in one update process, see figure 10. The multiple resulting residuals are stacked, parallel to how the residuals of the single measurement are stacked. The resulting residual $\mathbf{r}$ is therefore defined as

$$\mathbf{r} = \begin{bmatrix} \mathbf{r}_0^\top & \mathbf{r}_1^\top & \mathbf{r}_2^\top & ... & \mathbf{r}_m^\top \end{bmatrix}^\top$$

where r_i is shorthand notation for the *nullspaced* residual r_i^0 and m is the number of features in the update step. Equivalently, the Jacobians ($\mathbf{H}_{i,\mathbf{x}}^0$ written as $\mathbf{H}_i$) are stacked such that

$$\mathbf{H}_\mathbf{x} = \begin{bmatrix} \mathbf{H}_0^\top & \mathbf{H}_1^\top & \mathbf{H}_2^\top & ... & \mathbf{H}_m^\top \end{bmatrix}^\top \tag{51}$$

which can be collected in the equation

$$\mathbf{r} = \mathbf{H}_\mathbf{x}\tilde{\mathbf{x}} + \mathbf{n}. \tag{52}$$

With each feature and camera frame used, the size of the Jacobian and residual increases with $m \times (2N-3)$. To reduce the dimensions of the update step, QR decomposition [75] is employed on the Jacobian matrix.

$$\mathbf{H}_\mathbf{x} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T} \\ \mathbf{0} \end{bmatrix}$$

The QR decomposition separates a basis of the range $\mathbf{Q}_1$ and the nullspace $\mathbf{Q}_2$ of $\mathbf{H}_\mathbf{x}$ and the upper triangular matrix $\mathbf{T}$. Substituting this into eq. 52 yields

$$\mathbf{r} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} + \mathbf{n}$$

which can be rearanged into

$$\begin{bmatrix} Q_1^\top \mathbf{r} \\ Q_2^\top \mathbf{r} \end{bmatrix} = \begin{bmatrix} \mathbf{T} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} Q_1^\top \mathbf{n} \\ Q_2^\top \mathbf{n} \end{bmatrix}.$$

From this we can remove the trivial part, as $Q_2^\top \mathbf{r} = Q_2^\top \mathbf{n}$ and result in

$$\mathbf{Q}_1^\top \mathbf{r} = \mathbf{T}\tilde{\mathbf{x}} + \mathbf{Q}_1^\top \mathbf{n}.$$

The projection onto the *range* of the Jacobian leaves us with a smaller Jacobian and residual and leaves the parts which embody the non-noise information intact. The resulting residual and Jacobian can now be used for the update step itself.

The Kalman Gain is calculated analogously to chapter 2.3, using $\mathbf{T}$ from the new residual

$$\mathbf{K} = \mathbf{P}\mathbf{T}^T \left(\mathbf{T}\mathbf{P}\mathbf{T}^T + \mathbf{R}_n\right)^{-1}.$$

Redefining

$$\mathbf{r}_n = \mathbf{Q}_1^\top \mathbf{r}$$

we can now calculate the correction of the error-state vector, which gets compounded with the current nominal state vector, see eq. 11 and 12.

$$\delta\mathbf{x} = \mathbf{K}\mathbf{r}_n.$$

The covariance matrix is updated as well, through

$$\mathbf{P} = (\mathbf{I} - \mathbf{K}\mathbf{T})\,\mathbf{P}\,(\mathbf{I} - \mathbf{K}\mathbf{T})^\top + \mathbf{K}\mathbf{R}_n\mathbf{K}^\top.$$

The identity matrix $\mathbf{I}$ has the dimensions of the covariance matrix. The noise covariance matrix $\mathbf{R}$ has become $\mathbf{R}_n$ after projecting it onto the range of the Jacobian. We define $\mathbf{R}$ through the variance in $\mathbf{n}$, σ^2:

$$\mathbf{R} = \sigma^2\mathbf{I}_{2N-3}$$

where $\mathbf{I}$ is the size of the covariance matrix.

4.7 Monocamera Feature Extraction

We have covered all components of the MSCKF as shown in figure 6, except for the feature extractor. This element is seperate from the MSCKF algorithm. As long as the input into the feature extractor is an image and the output is a list of features on the focal plane of the camera (basically, a camera-hardware agnostic feature representation), the exact mechanics of the feature selection are not relevant.

Mourikis et al. [20] do not go into detail regarding the features used for their implementation and their implementation in general. Sun et al. [45] on the other hand present their feature extractor in open-source code. Therefore, the following section 4.8 regarding the stereo expansion will have a segment on the feature extractor used in the respective paper, as that design is used as the front-end of the implementations in this thesis.

4.8 Stereo MSCKF

This section describes the implementation by Sun et al. [45] which expands the MSCKF pipeline presented in the previous section to function on the basis of a stereo-camera and an IMU. The main components are the expansion of the residual and the corresponding Jacobian matrices as well as the design of a stereo-camera feature extractor.

4.8.1 State Vector

The main components of the state vector for the stereo-camera expanded MSCKF stay the same as in section 4.3.1. It is assumed, that the extrinsic camera calibration is known. Therefore, the *camera* states appended to the state vector remain based on the position estimation of camera 0. The transformation between camera 0 and camera 1 could be modeled into the estimation process as well, but as we will see in the measurement formulation, the stereo MSCKF is more robust to errors in the camera calibration through its definition of the residual.

4.8.2 Residual and Jacobi

One measurement of a single feature in the stereo MSCKF is defined as

$$\mathbf{z} = \begin{bmatrix} u^0 \\ v^0 \\ u^1 \\ v^1 \end{bmatrix}$$

where u and v are feature points and the index 0 and 1 represent camera 0 and camera 1 respectively. Through the extrinsic camera parameters (chapter B) the epipolar constraints between the two camera frames can be used to reduce $\mathbf{z}$ from $\mathbb{R}^4$ to $\mathbb{R}^3$. And while the epipolar constraints are used during the feature extraction to remove outliers, preserving the additional dimension of the second frame removes the reliance on stereo rectification. This increases robustness regarding errors in the camera calibration. It further makes the formulation of the measurement function easier. The measurement of the feature i in the current frame ℓ can be calculated through

$$\mathbf{z}_{i\ell} = \begin{bmatrix} u^0_{i,\ell} \\ v^0_{i,\ell} \\ u^1_{i,\ell} \\ v^1_{i,\ell} \end{bmatrix} = \begin{bmatrix} \frac{\mathbf{I}_{2\times 2}}{C^0_i Z_\ell} & \mathbf{0}_{2\times 2} \\ \mathbf{0}_{2\times 2} & \frac{\mathbf{I}_{2\times 2}}{C^1_i Z_\ell} \end{bmatrix} \begin{bmatrix} {}^{C^0_i}X_\ell \\ {}^{C^0_i}Y_\ell \\ {}^{C^1_i}X_\ell \\ {}^{C^1_i}Y_\ell \end{bmatrix}$$

with the position of the feature $\mathbf{p}_j$ in the left camera 0 frame and the right camera 1 frame being

$$
\begin{aligned}
{}^{C_i^0}\mathbf{p}_{f,i} &= \begin{bmatrix} {}^{C_i^0}X_j \\ {}^{C_i^0}Y_j \\ {}^{C_i^0}Z_j \end{bmatrix} = C({}^{C_i^0}_{G}q)({}^{G}\mathbf{p}_{f,i} - {}^{G}\mathbf{p}_{C_i^0}) \\
{}^{C_i^1}\mathbf{p}_{f,i} &= \begin{bmatrix} {}^{C_i^1}X_j \\ {}^{C_i^1}Y_j \\ {}^{C_i^1}Z_j \end{bmatrix} = C({}^{C_i^1}_{G}q)({}^{G}\mathbf{p}_{f,i} - {}^{G}\mathbf{p}_{C_i^1}) \\
&= C\left({}^{C_i^0}_{C_i^1}q\right)\left({}^{C_i^0}\mathbf{p}_{f,i} - {}^{C_i^0}\mathbf{p}_{C_i^1}\right).
\end{aligned}
$$

The feature position in the world frame ${}^{G}\mathbf{p}_{f,i}$ is determined through a least square estimation analogous to the original MSCKF algorithm. Both camera frames have feature observations, and both camera frame positions are estimated. Therefore all measurements from both camera frames are used to estimate the feature position. The measurement model leads us to a description of a cost function:

$$
\mathbf{r}_{i,\ell} = \mathbf{z}_{i,\ell} - \begin{bmatrix} \hat{u}_{i,\ell}^0 \\ \hat{v}_{i,\ell}^0 \\ \hat{u}_{i,\ell}^1 \\ \hat{v}_{i,\ell}^1 \end{bmatrix}. \tag{53}
$$

Linearizing this residual yields

$$
\mathbf{r}_{i,\ell} \simeq \mathbf{H}_{C_\ell,i}\,\tilde{\mathbf{x}}_{\mathbf{C}_\ell} + \mathbf{H}_{f,i}\,{}^{G}\tilde{\mathbf{p}}_{f,i} + \mathbf{n}_{i,\ell}
$$

parallel to the linearization in the original MSCKF. With the same procedure depicted in figure 14, the residuals and Jacobians of each frame of the feature are stacked vertically. The same nullspace projection is used. After another vertical stacking of all features into a residual and a Jacobian, we receive a general description as in eq. 52. This is subsequently projected onto the range to reduce the matrix sizes by removing the pure noise values. The Jacobians of the stereo MSCKF are simply created by use of the chain rule and partial derivations:

$$
\begin{aligned}
\mathbf{H}_{C_{i,\ell}} &= \frac{\partial \mathbf{z}_{i,\ell}}{\partial^{C_{i,\ell}^0}\mathbf{p}_{f,\ell}} \cdot \frac{\partial^{C_{i,\ell}^0}\mathbf{p}_{f,\ell}}{\partial \mathbf{x}_{C_{i,\ell}^0}} + \frac{\partial \mathbf{z}_{i,\ell}}{\partial^{C_{i,\ell}^1}\mathbf{p}_{f,\ell}} \cdot \frac{\partial^{C_{i,\ell}^1}\mathbf{p}_{f,\ell}}{\partial \mathbf{x}_{C_i^0}} \\
\mathbf{H}_{f_{i,\ell}} &= \frac{\partial \mathbf{z}_{i,\ell}}{\partial^{C_{i,\ell}^0}\mathbf{p}_{f,\ell}} \cdot \frac{\partial^{C_{i,\ell}^0}\mathbf{p}_{j,\ell}}{\partial {}^{G}\mathbf{p}_{f,\ell}} + \frac{\partial \mathbf{z}_{i,\ell}}{\partial^{C_{i,\ell}^1}\mathbf{p}_{f,\ell}} \cdot \frac{\partial^{C_{i,\ell}^1}\mathbf{p}_{f,\ell}}{\partial {}^{G}\mathbf{p}_{f,\ell}}
\end{aligned}
$$

Note, that only the camera 0 state is estimated and corrected in the state vector. The derivations regarding the state vector are thus reformulated to be derivatives toward $\mathbf{x}_{C_{i,\ell}^0}$ and ${}^{G}\mathbf{p}_{f,\ell}$.

4.8.3 Stereocamera Feature Extraction

The stereo MSCKF uses a multitude of steps to extract features from the received stereo images and track them over time in a robust and computationally efficient manner. This section gives a summary of the feature extractor designed by Sun et al. [45], as the same extractor is used for the stereo-photometric and the anchor-frame expansion. The main components of feature detection are

- tracking existing features
- finding new features in the images
- stereo matching and
- outlier detection.

Lukas-Kanade iterative optical flow [76] is used to track features through their spacial as well as temporal displacement. Features from the previous camera 0 frame are used to find features in the current camera 0 frame. The left side of figure 16 shows this approach, a step-wise tracking of features from the last recorded frame to the current one. Any feature for which no match can be found in the current frame is dropped and used in the update step (the lost features in figure 16.

Using a mask to hide all features already tracked from the previous image camera 0, the FAST feature detector is used to locate new features in the current frame $\mathrm{k}+1$.

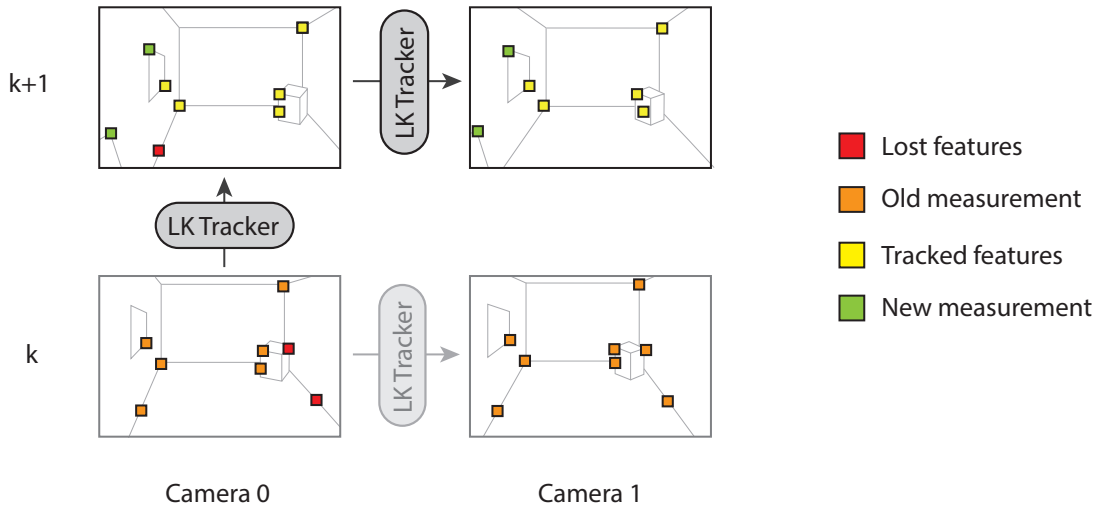

Figure 16: Tracking features through time and space

Similar to the temporal matching of features, features from camera 0 are used to find their stereo-matched feature in camera 1, as figure 16 shows (left to right). Any features matched between the Lukas-Kanade image pyramids are saved. Features that do not have an observation

in both cameras or do not have a temporal match are removed. Figure 16 gives an example. The red features in time-step k are features which have not found a twin in $\mathrm{k}+1$. The red feature in $\mathrm{k}+1$, camera 0 is a feature without a stereo-matched twin. Both features are therefore used for an update of the filter and subsequently removed.

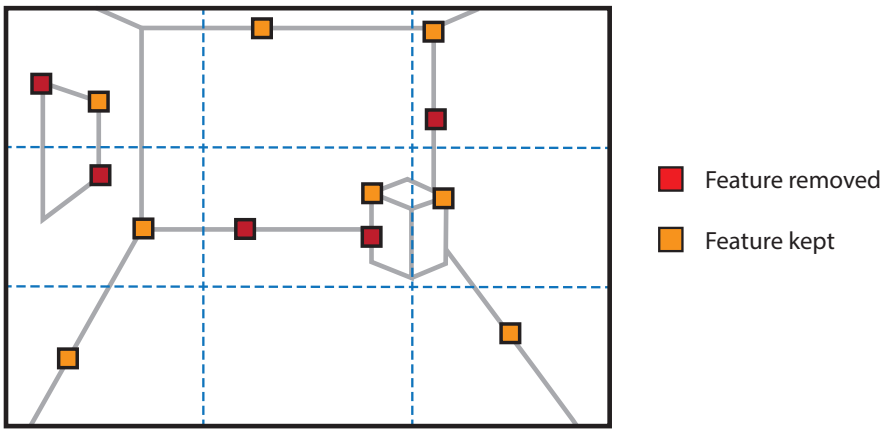

Figure 17: Grid to disperse the features more evenly across the frame

All remaining features with a spacial and temporal match are subsequently segmented into a grid overlaying the frame (see figure 17, the blue dotted lines). The features in each grid are ordered by their response quality. Each grid only keeps its top n features, where n is a set threshold value. This spreads the tracked features more evenly across the frame and makes the resulting bag of features more robust to change. The total number of features tracked is bounded by grid size and the number of features per cell. In figure 17 a 3x3 grid with one feature per parcel is shown. The number of features lost per time instant when the camera moves around is spread out more evenly.

The pair of camera 1 images of time-step k and $\mathrm{k}+1$ are used for outlier-rejection, removing any features, which movements are outside a certain expected step-size and range [77]. Additionally, a 2-point RANSAC [78] is applied between the two frames of camera 0 to remove any further outliers.

5 Anchor-Frame MSCKF

Based on the MSCKF pipeline of the previous chapter, this chapter presents a novel variation in the feature position estimation process. Instead of estimating the feature position in the world with three degrees of freedom, the first camera frame with an observation of the feature is used to constrain the feature position along a line. This not only changes the estimation process but also the residual linearization. As we will see, this residual design has a decreased accuracy compared to the stereo MSCKF but is more computationally efficient. This exact formulation is the basis for the photometric and stereo-photometric MSCKF formulation, which uses some anchor-frame measurements to decrease total computational load. The following section presents the anchor-frame based estimation process itself, section 5.2 presents the residual linearization of this formulation. The *stereo image* based anchor-frame residual is presented in the last section 5.3.

5.1 Feature Position Estimation

Figure 18 shows the differences between the MSCKF approach to estimating the feature position and the anchor-frame based approach.

The first frame with an observation of the currently considered feature - F_1 in figure 18 will be called the *anchor* frame from this point on. The frame position of the anchor frame is estimated in the state vector just like any other frame. This estimation, along with the measurement coordinates u and v of the feature returns a vector along which the position of the feature is estimated. Instead of estimating the $[x\ y\ z]^\top$ coordinates of the feature based on the collection of measurements as in chapter 4.6.1, only the *depth* of the feature with regards to the anchor frame is estimated.

Depth estimation is done through a Levenberg-Marqhart approximation. This nonlinear estimator uses the feature observations in every camera frame to optimally estimate the depth value of the feature in the frame with the initial observation - the anchor frame. The initial guess for the estimator uses the least square problem formulation based on the information in the first and last observations of the feature, as shown in the previous chapter in section 4.6.1. This reduces the estimation complexity of the Levenberg-Marquart formulation - the cost function calculation per measurement is reduced alongside with the Jacobian.

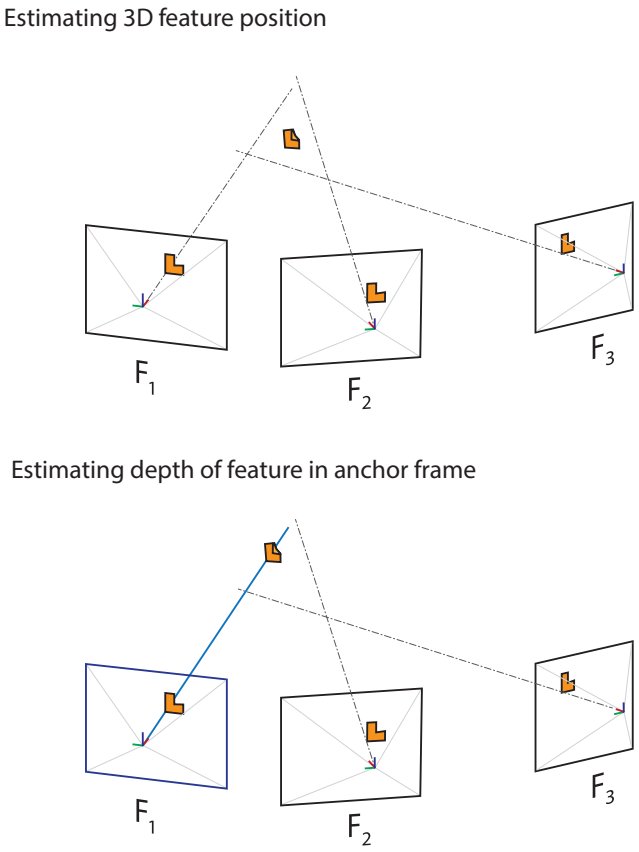

Figure 18: Feature projection comparison

5.2 Anchor-Frame Residual Linearization

The residual formulation as shown in eq. 52, is not directly edited - as the parameters of this formulation have not changed. Contrarily, the dependencies of the residual regarding the state vector and the non-state components have changed with the reformulation of ${}^{G}\mathbf{p}_f$. Specifically, the measurement function of a particular observation is now dependent on the observation camera position $\mathbf{x}_{C_{i,\ell}}$, the anchor-frame position $\mathbf{x}_{A_i}$ and the depth of the feature d_i. For numerical stability and due to a more direct estimation formulation of the Levenberg-Marquart formulation, the inverse depth ρ_i is used. The residual can therefore be written as:

$$\begin{aligned}\mathbf{r} &= \mathbf{z} - h(\hat{\mathbf{x}}_{\mathbf{A}_i}, \hat{\rho}_i, \hat{\mathbf{x}}_{C_{i,\ell}}) \\ &= \mathbf{z} - h({}^{G}\mathbf{p}_f(\hat{\mathbf{x}}_{\mathbf{A}_i}, \hat{\rho}_i), \hat{\mathbf{x}}_{C_{i,\ell}}).\end{aligned}$$

To linearize this residual, as in chapter 2.3 the Taylor expansion of the $h(\cdot)$ function is constructed after separating the true state of the variables into their nominal and error sum.

$$\begin{aligned}\mathbf{r} &= \mathbf{z} - h(\hat{\mathbf{x}}_{\mathbf{A}_i}, \hat{\rho}_i, \hat{\mathbf{x}}_{C_{i,\ell}}) \\ &\simeq \mathbf{z} - h(\mathbf{x}_{\mathbf{A}_i}, \rho_i, \mathbf{x}_{C_{i,\ell}}) + \mathbf{H}_{\rho_{i,\ell}}\tilde{\rho}_{i,\ell} + \mathbf{H}_{\mathbf{p}_{i,\ell}}\tilde{\mathbf{p}}_{i,\ell} + \mathbf{H}_{\mathbf{p}_{i,A}}\tilde{\mathbf{p}}_{i,A} \\ &\simeq \mathbf{H}_{\rho_{i,\ell}}\tilde{\rho}_{i,\ell} + \mathbf{H}_{\mathbf{p}_{i,\ell}}\tilde{\mathbf{p}}_{i,\ell} + \mathbf{H}_{\mathbf{p}_{i,A}}\tilde{\mathbf{p}}_{i,A}\end{aligned}$$

This leaves us with a linearized description of the residual. The desired Jacobi matrices can be described using the chain rule:

$$\mathbf{H}_{\rho_{i,\ell}} = \frac{\partial h_\ell}{\partial \rho_i} = \frac{\partial h}{\partial {}^C\mathbf{p}_{i,j}} \frac{\partial {}^C\mathbf{p}_{i,j}}{\partial {}^G\mathbf{p}_{i,j}} \frac{\partial {}^G\mathbf{p}_{i,j}}{\partial \tilde{\rho}_i} \tag{54}$$

$$\mathbf{H}_{\mathbf{p}_{i,\ell}} = \frac{\partial h_\ell}{\partial \mathbf{x}_{\mathbf{p}\ell}} = \frac{\partial h}{\partial {}^C\mathbf{p}_{i,j}} \frac{\partial {}^C\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}\ell}} \tag{55}$$

$$\mathbf{H}_{\mathbf{p}_{i,A}} = \frac{\partial h_\ell}{\partial \mathbf{x}_{\mathbf{p}_A}} = \frac{\partial h}{\partial {}^C\mathbf{p}_{i,j}} \frac{\partial {}^C\mathbf{p}_{i,j}}{\partial {}^G\mathbf{p}_{i,j}} \frac{\partial {}^G\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}_A}} \tag{56}$$

Note, that this description is the residuum based on the *nominal* state. The Kalman Filter operates on the *error* state. But as the following shows, on the basis that:

$$\begin{aligned}\tilde{\mathbf{x}} &= \delta\hat{\mathbf{x}} + \delta\tilde{\mathbf{x}} \\ \delta\hat{\mathbf{x}} &= 0\end{aligned}$$

the *expected* error is zero, the *expected* residual is zero as well:

$$\begin{aligned}\mathbf{r} &= \hat{r} + \delta r \\ \hat{r} &= 0.\end{aligned}$$

Therefore, the formulation of the residual derived with respect to the *error* state is the only non-zero component. The following equations are the Jacobian matrices for *one* feature in *one singular* frame. The appropriate index for this feature i and for the frame ℓ are removed from these equations to enhance readability - compare this to the previous equations, where the full Jacobians are sketched out. The derivations of the matrices are detailed afterwards.

$$\frac{\partial h}{\partial {}^{G}\mathbf{p}_j} = \frac{\partial h}{\partial {}^{C}\mathbf{p}_j} \cdot \frac{{}^{C}\mathbf{p}_j}{\partial {}^{G}\mathbf{p}_j} = \begin{bmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} \\ 0 & \frac{1}{z} & -\frac{y}{z^2} \end{bmatrix} \cdot C({}^{C}_{G}q)$$

$$\frac{\partial h}{\partial X_{Pl}} = \begin{bmatrix} \frac{\partial h}{\partial {}^{C_l}_{G}\delta\theta} & \frac{\partial h}{\partial {}^{G}\mathbf{p}_{C\ell}} \end{bmatrix} = \begin{bmatrix} \frac{\partial h}{{}^{C}\mathbf{p}_j} \cdot \frac{{}^{C}\mathbf{p}_j}{\partial {}^{C_l}_{G}q} & \frac{\partial h}{{}^{C}\mathbf{p}_j} \cdot \frac{{}^{C}\mathbf{p}_j}{\partial {}^{G}\mathbf{p}_{C\ell}} \end{bmatrix}$$

$$\frac{\partial {}^{C}\mathbf{p}_j}{\partial {}^{C_l}_{G}\delta\theta} = \lfloor {}^{C}\mathbf{p}_j \rfloor_\times$$

$$\frac{\partial {}^{C}\mathbf{p}_j}{\partial {}^{G}\mathbf{p}_{C\ell}} = -C({}^{C}_{G}q)$$

$$\frac{\partial {}^{G}\mathbf{p}_j}{\partial \rho} = C({}^{A}_{G}q)^\top \cdot \begin{bmatrix} -\frac{u}{\rho^2} & -\frac{v}{\rho^2} & -\frac{1}{\rho^2} \end{bmatrix}^\top$$

$$\frac{\partial {}^{G}\mathbf{p}_j}{\partial X_{PA}} = \begin{bmatrix} \frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{C_l}_{G}\delta\theta} & \frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{G}\mathbf{p}_A} \end{bmatrix}$$

$$\frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{C_l}_{G}\delta\theta} = -C({}^{A}_{G}q)^\top \lfloor \begin{bmatrix} \frac{u}{\rho} & \frac{v}{\rho} & \frac{1}{\rho} \end{bmatrix}^\top \rfloor_\times$$

$$\frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{G}\mathbf{p}_A} = \mathbf{I}_3$$

Recall the ranking of functions which result in the feature position

$$\mathbf{z}_{\ell,j} = \mathbf{h}\left({}^{G}\mathbf{p}\left(\rho_i, \mathbf{x}_{\mathbf{p}_A}\right), \mathbf{x}_{\mathbf{p}\ell}\right). \tag{57}$$

Note, that the linearization points for these Jacobians is the error state. To produce the correct Jacobians, the description of the true state in eq. 65 and following need to be split into a nominal and an error part. To ease readability the following substitution is made: $C({}^{C}_{G}q) = {}^{C}_{G}\mathbf{R}$. Recalling the chained Jacobians of eq. 54, the necessary error-state descriptions are of ${}^{C}\mathbf{p}_j$ and ${}^{G}\mathbf{p}_j$. The next section will derive the first Jacobian $\frac{\partial \mathbf{h}}{\partial {}^{C}\mathbf{p}_{i,j}}$ as well as the error-state descriptions and resulting Jacobians. The subsequent section summarizes the restructuring process to fit the Jacobians into a Kalman Filter framework.

5.2.1 Error-State Jacobian Derivation

The derivation of $\frac{\partial \mathbf{h}}{\partial {}^{C}\mathbf{p}_j}$ is the derivation of eq. 65 regarding ${}^{C}\mathbf{p}_j$, which is

$$\frac{\partial h}{\partial {}^{C}\mathbf{p}_j} = \begin{bmatrix} \frac{1}{z} & 0 & -\frac{x}{z^2} \\ 0 & \frac{1}{z} & -\frac{y}{z^2} \end{bmatrix}.$$

To derive ${}^{C}\tilde{\mathbf{p}}_j$ and subsequently calculate its derivative with respect to the *current* camera

state, we must first extract the error-state formulation from ${}^{C}\mathbf{p}_j$

$$\begin{aligned}
{}^{C}\mathbf{p}_j = {}^{C}\hat{\mathbf{p}}_j + {}^{C}\tilde{\mathbf{p}}_j &= {}^{C}_{G}\mathbf{R}\,({}^{G}\mathbf{p}_j - {}^{G}\mathbf{p}_l) \\
&= {}^{C}_{G}\delta\mathbf{R}\cdot {}^{C}_{G}\hat{\mathbf{R}}\,({}^{G}\hat{\mathbf{p}}_j + {}^{G}\tilde{\mathbf{p}}_j - {}^{G}\hat{\mathbf{p}}_l - {}^{G}\tilde{\mathbf{p}}_l) \\
&\simeq (\mathbf{I}_3 - \lfloor\delta\Theta\rfloor_\times)\cdot {}^{C}_{G}\hat{\mathbf{R}}\,({}^{G}\hat{\mathbf{p}}_j + {}^{G}\tilde{\mathbf{p}}_j - {}^{G}\hat{\mathbf{p}}_l - {}^{G}\tilde{\mathbf{p}}_l)
\end{aligned}$$

where according to Trawny et. al. [79] a small-angle quaternion rotation can be described as a rotation matrix

$$\delta q \simeq \begin{bmatrix} \frac{1}{2}\delta\theta \\ 1 \end{bmatrix} = (\mathbf{I}_3 - \lfloor\delta\Theta\rfloor_\times).$$

We can now cross multiply to receive

$${}^{C}\hat{\mathbf{p}}_j + {}^{C}\tilde{\mathbf{p}}_j \simeq {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\hat{\mathbf{p}}_j - {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\hat{\mathbf{p}}_l + {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_j - {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_l - \lfloor\delta\Theta\rfloor_\times\cdot {}^{C}_{G}\hat{\mathbf{R}})({}^{G}\hat{\mathbf{p}}_j + {}^{G}\tilde{\mathbf{p}}_j - {}^{G}\hat{\mathbf{p}}_l - {}^{G}\tilde{\mathbf{p}}_l)$$

where we can remove

$${}^{C}\hat{\mathbf{p}}_j = {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\hat{\mathbf{p}}_j - {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\hat{\mathbf{p}}_l$$

as we are interested in the error term representation only. Further cross-multiplying, and removing any terms dependent on errors of errors, we receive:

$$\begin{aligned}
{}^{C}\tilde{\mathbf{p}}_j &\simeq {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_j - {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_l - (\lfloor\delta\Theta\rfloor_\times\cdot {}^{C}_{G}\hat{\mathbf{R}})({}^{G}\hat{\mathbf{p}}_j + {}^{G}\tilde{\mathbf{p}}_j - {}^{G}\hat{\mathbf{p}}_l - {}^{G}\tilde{\mathbf{p}}_l) \\
&\simeq {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_j - {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_l - \lfloor\delta\Theta\rfloor_\times {}^{C}_{G}\hat{\mathbf{R}}\,{}^{G}\hat{\mathbf{p}}_j - \cancel{\lfloor\delta\Theta\rfloor_\times {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_j} \\
&\quad + \lfloor\delta\Theta\rfloor_\times {}^{C}_{G}\hat{\mathbf{R}}\,{}^{G}\hat{\mathbf{p}}_l + \cancel{\lfloor\delta\Theta\rfloor_\times {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_l} \\
&\simeq {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_j - {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_l - \lfloor\delta\Theta\rfloor_\times {}^{C}_{G}\hat{\mathbf{R}}\,{}^{G}\hat{\mathbf{p}}_j + \lfloor\delta\Theta\rfloor_\times {}^{C}_{G}\hat{\mathbf{R}}\,{}^{G}\hat{\mathbf{p}}_l.
\end{aligned}$$

Using the following equivalency from [79]

$$\lfloor\mathbf{a}\rfloor_\times\cdot\mathbf{b} = -\lfloor\mathbf{b}\rfloor_\times\cdot\mathbf{a}$$

we receive

$$\boxed{\simeq {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_j - {}^{C}_{G}\hat{\mathbf{R}}{}^{G}\tilde{\mathbf{p}}_l + \lfloor {}^{C}_{G}\hat{\mathbf{R}}\,{}^{G}\hat{\mathbf{p}}_j\rfloor_\times\,\delta\Theta - \lfloor {}^{C}_{G}\hat{\mathbf{R}}\,{}^{G}\hat{\mathbf{p}}_l\rfloor_\times\,\delta\Theta}\,.$$

Using this equivalency, we can now refer back to the Jacobian descriptions in eq. 54. The desired partial derivatives are

$$\frac{\partial {}^{C}\mathbf{p}_j}{\partial {}^{G}\tilde{\mathbf{p}}_j} \quad , \quad \frac{\partial {}^{C}\mathbf{p}_j}{\partial {}^{C}_{G}\delta\theta} \quad , \quad \frac{\partial {}^{C}\mathbf{p}_j}{\partial {}^{G}\tilde{\mathbf{p}}_l}.$$

The first derivation regarding the error in the position of the feature is trivially:

$$\boxed{\frac{\partial^C \mathbf{p}_j}{\partial^G \tilde{\mathbf{p}}_j} = {}^C_G\hat{\mathbf{R}}}$$

and similar the derivation regarding the error in the position of the camera

$$\boxed{\frac{\partial^C \mathbf{p}_j}{\partial^G \tilde{\mathbf{p}}_l} = -{}^C_G\hat{\mathbf{R}}}.$$

Finally, regarding the error in the cameras orientation:

$$\begin{aligned}\frac{\partial^C \mathbf{p}_j}{\partial^C_G \delta\theta} &= \lfloor {}^C_G\hat{\mathbf{R}}\, {}^G\hat{\mathbf{p}}_j \rfloor_\times - \lfloor {}^C_G\hat{\mathbf{R}}\, {}^G\hat{\mathbf{p}}_l \rfloor_\times \\ &= \lfloor {}^C_G\hat{\mathbf{R}}\, {}^G\hat{\mathbf{p}}_j - {}^C_G\hat{\mathbf{R}}\, {}^G\hat{\mathbf{p}}_l \rfloor_\times \\ &= \lfloor {}^C_G\hat{\mathbf{R}}({}^G\hat{\mathbf{p}}_j - {}^G\hat{\mathbf{p}}_l) \rfloor_\times\end{aligned}$$

$$\boxed{\frac{\partial^C \mathbf{p}_j}{\partial^C_G \delta\theta} = \lfloor {}^C\hat{\mathbf{p}}_j \rfloor_\times}.$$

The second set of derivations regards the position of the feature in the world frame ${}^G\mathbf{p}_j$. Substituting $\mathbf{u} = \begin{bmatrix} u & v & 1 \end{bmatrix}^\top$, it can be formulated as

$$\begin{aligned}{}^G\mathbf{p}_j &= {}^G\mathbf{p}_A + {}^A_G\mathbf{R}^\top \frac{1}{\rho}\mathbf{u} \\ {}^G\hat{\mathbf{p}}_j + {}^G\tilde{\mathbf{p}}_j &\simeq {}^G\hat{\mathbf{p}}_A + {}^G\tilde{\mathbf{p}}_A + ((\mathbf{I}_3 - \lfloor {}^A_G\delta\theta \rfloor_\times)\, {}^A_G\hat{\mathbf{R}})^\top \frac{1}{\rho}\mathbf{u}\end{aligned}$$

multiplying the small-angle approximation from the left. With some basic matrix calculations, we receive

$$\begin{aligned}{}^G\hat{\mathbf{p}}_j + {}^G\tilde{\mathbf{p}}_j &\simeq {}^G\hat{\mathbf{p}}_A + {}^G\tilde{\mathbf{p}}_A + ((\mathbf{I}_3 - \lfloor {}^A_G\delta\theta \rfloor_\times)\, {}^A_G\hat{\mathbf{R}})^\top \frac{1}{\rho}\mathbf{u} \\ &\simeq {}^G\hat{\mathbf{p}}_A + {}^G\tilde{\mathbf{p}}_A + {}^A_G\hat{\mathbf{R}}^\top (\mathbf{I}_3 - \lfloor \delta {}^A_G\theta \rfloor_\times)^\top \frac{1}{\rho}\mathbf{u} \\ &\simeq {}^G\hat{\mathbf{p}}_A + {}^G\tilde{\mathbf{p}}_A + {}^A_G\hat{\mathbf{R}}^\top (\mathbf{I}_3^\top - \lfloor {}^A_G\delta\theta \rfloor_\times^\top) \frac{1}{\rho}\mathbf{u} \\ &\simeq {}^G\hat{\mathbf{p}}_A + {}^G\tilde{\mathbf{p}}_A + {}^A_G\hat{\mathbf{R}}^\top (\mathbf{I}_3 + \lfloor {}^A_G\delta\theta \rfloor_\times) \frac{1}{\rho}\mathbf{u}\end{aligned}$$

subtracting ${}^G\hat{\mathbf{p}}_j$ from both sides in the last line, using its initial description:

$${}^G\tilde{\mathbf{p}}_j \simeq {}^G\tilde{\mathbf{p}}_A + {}^A_G\hat{\mathbf{R}}^\top (\mathbf{I}_3 + \lfloor {}^A_G\delta\theta \rfloor_\times) \frac{1}{\rho}\mathbf{u} - {}^A_G\hat{\mathbf{R}}^\top \frac{1}{\hat{\rho}}\mathbf{u}.$$

Cross-multiplying the small-angle approximation, results in the following error state description of the feature position in the world coordinate frame:

$$ {}^{G}\tilde{\mathbf{p}}_j \simeq {}^{G}\tilde{\mathbf{p}}_A + {}^{A}_{G}\hat{\mathbf{R}}^{\top} \frac{1}{\rho}\,\mathbf{u} + {}^{A}_{G}\hat{\mathbf{R}}^{\top} \lfloor {}^{A}_{G}\delta\theta \rfloor_{\times} \frac{1}{\rho}\,\mathbf{u} - {}^{A}_{G}\hat{\mathbf{R}}^{\top} \frac{1}{\hat{\rho}}\,\mathbf{u} \tag{58} $$

from which we can derive the three partial derivatives

$$ \frac{\partial {}^{G}\mathbf{p}_j}{\partial \tilde{\rho}} \quad , \quad \frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{A}_{G}\delta\theta} \quad , \quad \frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{G}\mathbf{p}_A}. $$

First, the derivation regarding the inverse depth error $\tilde{\rho}$

$$ \frac{\partial {}^{G}\mathbf{p}_j}{\partial \tilde{\rho}} = -{}^{A}_{G}\hat{\mathbf{R}}^{\top} \frac{1}{(\hat{\rho}+\tilde{\rho})^2}\,\mathbf{u} + {}^{A}_{G}\hat{\mathbf{R}}^{\top} \lfloor {}^{A}_{G}\delta\theta \rfloor_{\times} \frac{1}{(\hat{\rho}+\tilde{\rho})^2}\,\mathbf{u} $$

where the mean of $\tilde{\rho}$ and $\delta\theta$ is 0. For that reason, the derivation reduces to

$$ \boxed{\frac{\partial {}^{G}\mathbf{p}_j}{\partial \tilde{\rho}} = -{}^{G}_{A}\hat{\mathbf{R}}\,\frac{1}{\hat{\rho}^2}\,\mathbf{u}}\,. $$

The next derivation is the small-angle rotation error between the anchor frame and the Ground frame. Again, we use eq. 58 and use the small-angle equivalency (eq. 3 in appendix 2.2), focusing only on the relevant part for the derivation:

$$ \begin{aligned} {}^{G}\tilde{\mathbf{p}}_j &\simeq \ldots + {}^{A}_{G}\hat{\mathbf{R}}^{\top} \lfloor {}^{A}_{G}\delta\theta \rfloor_{\times} \frac{1}{\rho}\,\mathbf{u} - \ldots \\ &\simeq \ldots - {}^{A}_{G}\hat{\mathbf{R}}^{\top} \lfloor \frac{1}{\rho}\,\mathbf{u} \rfloor_{\times} {}^{A}_{G}\delta\theta - \ldots \end{aligned} $$

from which point forward we can simply derive by the small angle ${}^{A}_{G}\delta\theta$ which is now a vector in the last step of a matrix multiplication.

$$ \boxed{\frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{A}_{G}\delta\theta} = -{}^{A}_{G}\hat{\mathbf{R}}^{\top} \lfloor \frac{1}{\rho}\,\mathbf{u} \rfloor_{\times}}\,. $$

The final Jacobian, the derivation of eq. 58 towards the anchor position in the world frame is simply

$$ \boxed{\frac{\partial {}^{G}\mathbf{p}_j}{\partial {}^{G}\mathbf{p}_A} = \mathbf{I}_{3\times 3}}\,. $$

5.2.2 Restructuring of the Feature Jacobian

This last section leaves us with a complete description of the Jacobians for the anchor-based residual for one frame of one feature.

$$\mathbf{r} \simeq \mathbf{H}_{\rho_{i,\ell}} \tilde{\rho}_{i,\ell} + \mathbf{H}_{\mathbf{p}_{i,\ell}} \tilde{\mathbf{p}}_{i,\ell} + \mathbf{H}_{\mathbf{p}_{i,A}} \tilde{\mathbf{p}}_{i,A}$$

Similar to the original MSCKF, these Jacobians must be brought into a state-vector affable form. We write the residual of one frame of one feature in the form

$$\mathbf{r}_{i,\ell} = \mathbf{H}_{\mathbf{x}_{i,\ell}} \tilde{\mathbf{x}} + \mathbf{H}_{\rho_{i,\ell}} \rho_i. \tag{59}$$

Note, that that ρ_i is the same for every frame of the feature. $\mathbf{H}_{\mathbf{x}_{i,\ell}}$ is constructed as

$$\mathbf{H}_{\mathbf{x}_{i,\ell}} = \begin{bmatrix} \mathbf{0} \ \dots \ \mathbf{0} \ \mathbf{H}_{\mathbf{p}_{i,A}} \ \mathbf{0} \ \dots \ \mathbf{0} \ \mathbf{H}_{\mathbf{p}_{i,\ell}} \ \mathbf{0} \dots \mathbf{0} \end{bmatrix}$$

and the resulting Jacobians are stacked into a single feature Jacobian:

$$\mathbf{H}_{\mathbf{x}_i} = \begin{bmatrix} \mathbf{H}_{\mathbf{x}_{i,1}}^\top \ \mathbf{H}_{\mathbf{x}_{i,2}}^\top \ \dots \ \mathbf{H}_{\mathbf{x}_{i,n}}^\top \end{bmatrix}^\top$$

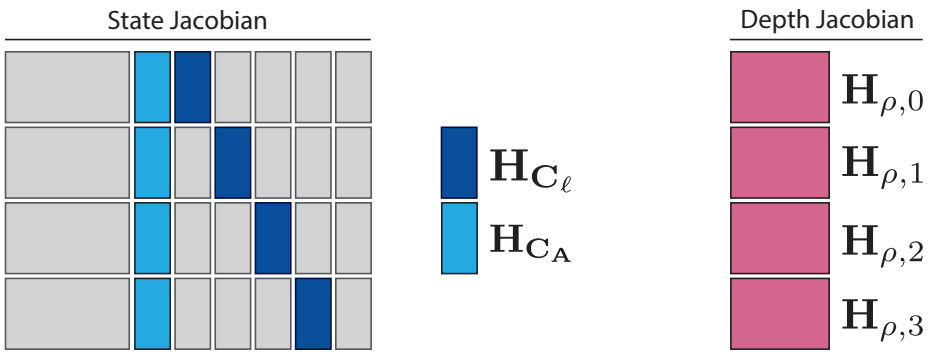

Figure 19: The Jacobians of multiple measurements for one feature

and analogously for $\mathbf{H}_{\rho_i}$. Figure 19 visualizes the stacking structure of the anchor Jacobian and the frame Jacobian for *all frames* in which the considered feature has an observation. The Jacobian regarding the inverse depth ρ is also stacked. ρ is not estimated in the state vector though. Just as in the previous chapter 4.6.4, the linearized residual in eq. 59 is projected onto the nullspace of $\mathbf{H}_{\rho_i}$.

5.3 Stereo Anchor-Frame Residual

We now have a complete formulation of the MSCKF using the anchor-frame based feature estimation. Using a *stereo* camera setup for feature estimation increases the robustness and accuracy of the process, as the known stereo baseline between the two cameras allows for an exact triangulation of points. Therefore, we try to integrate the information of a second camera into the residual formulation of this anchor-based MSCKF. The following section will describe this expansion in detail. Note that the stereo expansion of the MSCKF - which was conceptually explained in section 4.8 - works similar to the calculations presented in this chapter.

The stereo residual is reformulated exactly as in eq. 53. Again, to increase robustness regarding errors in the external camera calibration, the residual is not reduced through epipolar constraints but remains in $\mathbb{R}^4$. Linearizing this residual using the anchor-based measurement function eq. 57 yields the following anchor Jacobians. Compare to eq. 54, in which only one camera was used.

The Jacobian structure, using partial derivatives towards camera 0 and camera 1, is

$$\mathbf{H}_{\rho_{i,\ell}} = \frac{\partial h_\ell}{\partial \rho_i} = \frac{\partial h}{\partial {}^{C^0}\mathbf{p}_{i,j}} \frac{\partial {}^{C^0}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \tilde{\rho}_i} + \frac{\partial h}{\partial {}^{C^1}\mathbf{p}_{i,j}} \frac{\partial {}^{C^1}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \tilde{\rho}_i} \tag{60}$$

$$\mathbf{H}_{\mathbf{p}_{i,\ell}} = \frac{\partial h_\ell}{\partial \mathbf{x}_{\mathbf{p}\ell}} = \frac{\partial h}{\partial {}^{C^0}\mathbf{p}_{i,j}} \frac{\partial {}^{C^0}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}\ell}} + \frac{\partial h}{\partial {}^{C^1}\mathbf{p}_{i,j}} \frac{\partial {}^{C^1}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}\ell}} \tag{61}$$

$$\mathbf{H}_{\mathbf{p}_{i,A}} = \frac{\partial h_\ell}{\partial \mathbf{x}_{\mathbf{p}_A}} = \frac{\partial h}{\partial {}^{C^0}\mathbf{p}_{i,j}} \frac{\partial {}^{C^0}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}_A}} + \frac{\partial h}{\partial {}^{C^1}\mathbf{p}_{i,j}} \frac{\partial {}^{C^1}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}_A}} \tag{62}$$

where

$$\frac{\partial h}{\partial {}^{C^0}\mathbf{p}_f} = \begin{bmatrix} \frac{1}{{}^{C^0}z} & 0 & -\frac{{}^{C^0}x}{{}^{C^0}z^2} \\ 0 & \frac{1}{{}^{C^0}z} & -\frac{{}^{C^0}y}{{}^{C^0}z^2} \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\frac{\partial h}{\partial {}^{C^1}\mathbf{p}_f} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ \frac{1}{{}^{C^0}z} & 0 & -\frac{x}{{}^{C^0}z^2} \\ 0 & \frac{1}{{}^{C^0}z} & -\frac{y}{{}^{C^0}z^2} \end{bmatrix}$$

and from the previous section we already know

$$\frac{\partial {}^{C^1}\mathbf{p}_f}{\partial \mathbf{x}_{C^0}} = \begin{pmatrix} \left\lfloor {}^{C^0}\hat{\mathbf{p}}_f \right\rfloor_\times & -C\left({}^{C^0}_{G}\hat{\mathbf{q}}\right) \end{pmatrix}$$

from which we can compute

$$\frac{\partial^{C^2}\mathbf{p}_f}{\partial \mathbf{x}_{C^1}} = C\left({}^{C^1}_{C^0}\mathbf{q}\right)^\top \frac{\partial^{C^1}\mathbf{p}_f}{\partial \mathbf{x}_{C^0}}$$
$$\frac{\partial^{C^2}\mathbf{p}_f}{\partial \mathbf{x}_{C^1}} = C\left({}^{C^1}_{C^0}\mathbf{q}\right)^\top \left(\left\lfloor {}^{C^0}\hat{\mathbf{p}}_f \right\rfloor_\times \quad - C\left({}^{C^0}_{G}\hat{\mathbf{q}}\right)\right).$$

This expansion merges the information of both cameras into a cost function for the Kalman Filter.

With this chapter, we have changed the estimation process of the feature to an anchor-frame based formulation. This has the advantage of reducing the size of the nullspace matrix and simplifying the estimation process of the feature position to a one-dimensional problem. The next chapter will alter this formulation of the MSCKF by using the pixel values around the feature directly, instead of using the feature position as the measurement.

6 Photometric Expansion

Zheng et al. [62] present a photometric patch-based visual inertial odometry algorithm based on the monocamera MSCKF pipeline of chapter 4, changing the update step from a feature-based to a semi-dense formulation. The following chapter presents detailed derivations for this photometric expansion of the monocular MSCKF and building on that, expands this concept to stereo images. The anchor-frame based measurement formulation is used as a basis for this photometric expansion. The first section will summarize the photometric approach - the subsequent section will expand on the update step and the Jacobians used for the calculation, including the addition of the stereo camera in the residual. The indirect approach of the irradiance formulation, which directly uses the anchor frame in the residual formulation, will be explained thereafter. It is used to reduce the computational cost of the algorithm.

6.1 Overview

The difference between the standard MSCKF and the photometric MSCKF is the measurement function. The standard MSCKF uses the distance between the estimated and the measured position of a feature - and proceeds to use it for the Kalman Filter covariance optimization. The photometric MSCKF instead uses the pixel intensity values around this feature and corrects the

error between the initial intensity patch and the measured patch at the estimated position of the feature, see figure 20. The estimation step regarding the IMU is the same as in the standard MSCKF. Differences are found in the calculation of the residual and its linearization for the state propagation. The goal of the photometric expansion in [62] was to use the *same* feature points as the standard MSCKF uses, to show only the error reduction through a change in the cost function.

Refer to figure 4 for a summary of the entire MSCKF. The following figure 20 visualizes the general concept of the photometric residual. The feature position in the world coordinate frame is estimated with the process presented in the anchor-frame based MSCKF. The pixel points *around* the considered feature are projected into the world frame as well, using the same inverse depth estimation ρ. Similar to the reprojection of figure 11, all the points of this patch around the feature are reprojected into each considered frame.

In step one of figure 20 the patch around the measured feature position is extracted. The *reprojection* of the anchor patch into this currently considered frame is extracted as well. Note, that the reprojected patch is subject to errors based on the errors from the IMU prediction, just as the feature position in the MSCKF is. This is the error we want to reduce. Step two shows the actual relation between the two patches and the pixel version of the patches individually. The residual is constructed in step three, showing the uncorrelated parts of the patch. The patch derivative in step four represents the *direction* the reprojected patch has to move, in order to minimize the discrepancy. Note, that there are two distinct gradients in the x and the y direction of the image. The correction based on the residual and the gradients in x and y direction are visualized as direction and magnitude arrows in every pixel in the final step in figure 20. In essence, this is what the photometric cost function of this chapter uses to update the IMU prediction, represented through arrows in every pixel in the final step.

6.2 Update Step

Initially, only the feature observations in each frame are saved for each feature. Alongside a moving window of camera positions saved in the state variable, the image from the camera frames themselves are temporarily saved - these images are later used in the measurement update. Recall, that the update procedure is called, when a feature is no longer tracked (because it has left the frame) or when a feature observation is removed when removing frames from the moving window of camera positions.

In this update step, all observations of a feature are used to estimate the true 3D position of the feature in the world frame, which gives us an estimate of ρ, the inverse depth of the feature in relation to the anchor frame. Using the estimated position of the anchor frame from the state vector, the observation of the feature and the inverse depth information regarding the feature in the anchor frame ρ, the position of the pixel patch around the feature is calculated.

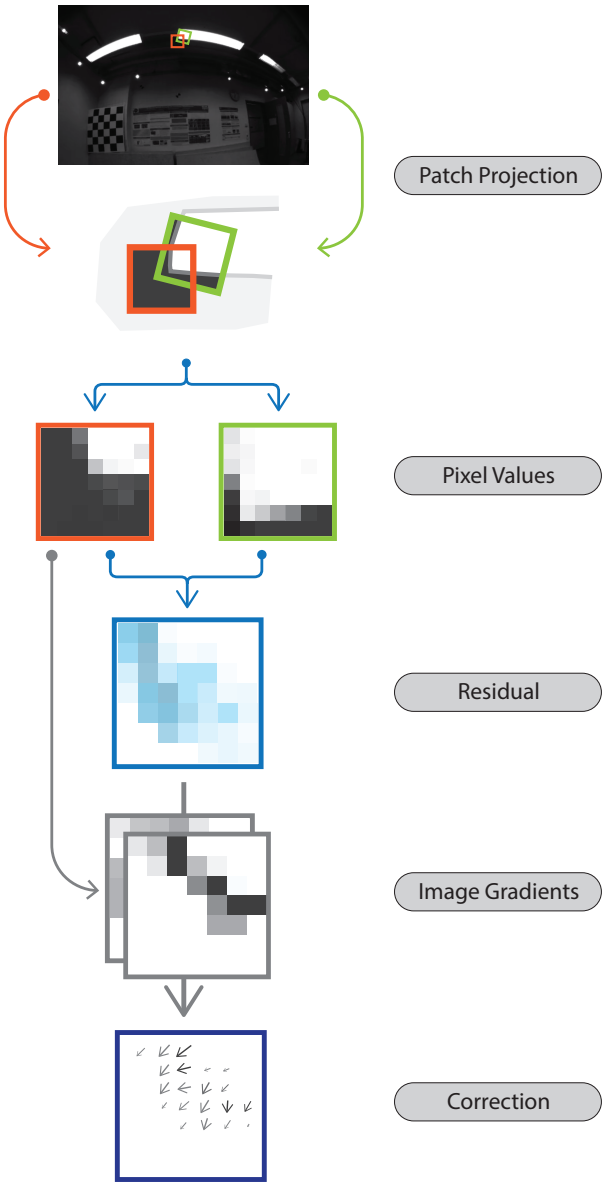

Figure 20: The residual formulation of the patch-based photometric MSCKF visualized

Note, that to calculate the points in this pixel patch, the *true pixel positions* in the camera image are used, calling them u' and v'. These points in the image plane are projected into space using the pinhole model.

$$ {}^{G}\mathbf{p}_j = {}^{G}\mathbf{p}_A + C({}^{A}_{G}q)^{\top} \cdot \begin{bmatrix} \frac{u}{\rho} \\ \frac{v}{\rho} \\ \frac{1}{\rho} \end{bmatrix} \tag{63} $$

The values of u' and v' are calculated deterministically around the *observed* feature position

in the current frame. Therefore, the only estimations used in the previous calculation are ρ and the anchor camera state $\mathbf{x}_{C_l} = [{}^G_A q \ {}^G\mathbf{p}_A]$.

These j patch points are then transformed into the new camera frame and projected onto the new camera frames image plane:

$$ {}^C\mathbf{p}_j = \begin{bmatrix} {}^{Cl}x \\ {}^{Cl}y \\ {}^{Cl}z \end{bmatrix} = C({}^C_G q) \cdot ({}^G\mathbf{p}_j - {}^G\mathbf{p}_C l) \tag{64} $$

$$ \hat{z} = \begin{bmatrix} \frac{{}^{Cl}x}{{}^{Cl}z} \\ \frac{{}^{Cl}y}{{}^{Cl}z} \end{bmatrix} \tag{65} $$

where the backward projection of eq. 65 again uses the pinhole model. The resulting pixel value at this point is considered the measurement of the irradiance. Note, that the image frames in the algorithms are pre-undistorted using the appropriate camera distortion model, see figure 21. This allows the usage of the simple pinhole-model projection (appendix B). This back-projection of the anchor patch onto the new image is the reason the anchor-frame description is used as the precursor to the photometric approach. Using the feature position from the anchor-frames point of view creates a defined pose for the patch in space, without having to add additional constraints regarding the patch orientation.

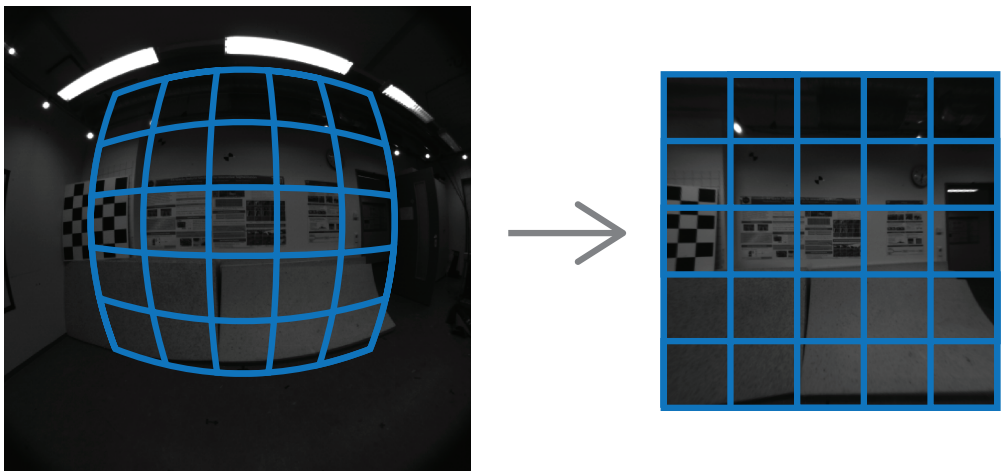

Figure 21: Undistorting an image using a fish-eye model for the camera

6.3 Photometric Residual

With these two types of data:

- the measurement of the pixel values in certain points based on the reprojection of points around a feature and
- the true pixel values based on the measurement around the actual feature

the residual can be constructed and linearized, parallel to eq. 59.

$$r_{i,\ell,j} = z_{i,\ell,j} - \begin{bmatrix} \frac{Cl_x}{Cl_z} \\ \frac{Cl_y}{Cl_z} \end{bmatrix}$$

where j defines the pixel in the patch and $N \times N$ is used to describe the size of the pixel patch. For a patch-size of 3×3, j would range from 0 to 8. These different residual values for each point in the patch are stacked, forming the residual of the measurement

$$\mathbf{r}_{i,\ell} = \begin{bmatrix} r_{i,\ell,0}^\top & r_{i,\ell,1}^\top & \dots & r_{i,\ell,N\times N}^\top \end{bmatrix}^\top .$$

This residual formulation can be linearized in the same manner as in chapter 5.2. Recall, the ranking of the functions which result in the irradiance at a specific point:

$$I_{\ell,j} = I_\ell \left(\mathbf{h} \left({}^G\mathbf{p} \left(\rho_i, \mathbf{x}_{\mathbf{p}_A}, j \right), \mathbf{x}_{\mathbf{p}\ell} \right) \right).$$

Note that the only difference in the photometric *measurement* function compared to the anchor measurement function in eq. 57 is the added irradiance function block. The *residual* formulation is different compared to the anchor formulation, but the individual Jacobians remain mostly untouched, as they are only dependent on the measurement description. The desired Jacobi matrices can be described using chain rule:

$$\mathbf{H}_{\rho_{i\ell,j}} = \frac{\partial I_\ell}{\partial \rho_i} = \frac{\partial I_\ell}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial {}^C\mathbf{p}_{i,j}} \frac{\partial {}^C\mathbf{p}_{i,j}}{\partial {}^G\mathbf{p}_{i,j}} \frac{\partial {}^G\mathbf{p}_{i,j}}{\partial \tilde{\rho}_i} \tag{66}$$

$$\mathbf{H}_{\mathbf{p}_{i,\ell,j}} = \frac{\partial I_\ell}{\partial \mathbf{x}_{\mathbf{p}\ell}} = \frac{\partial I_\ell}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial {}^C\mathbf{p}_{i,j}} \frac{\partial {}^C\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}\ell}} \tag{67}$$

$$\mathbf{H}_{\mathbf{p}_{iA,j}} = \frac{\partial I_\ell}{\partial \mathbf{x}_{\mathbf{p}_A}} = \frac{\partial I_\ell}{\partial \mathbf{h}} \frac{\partial \mathbf{h}}{\partial {}^C\mathbf{p}_{i,j}} \frac{\partial {}^C\mathbf{p}_{i,j}}{\partial {}^G\mathbf{p}_{i,j}} \frac{\partial {}^G\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}_A}} \tag{68}$$

The change in irradiance ∂I_ℓ in x and y direction is extracted from the pixel values. As figure 22 shows, the surrounding horizontal and vertical pixels of the pixel in question convoluted with an image-gradient kernel return an estimate to this change in irradiance.

For the irradiance change in x direction ΔI_x the left pixel values in row $p_{x-1,y}$ are subtracted from the right pixel values of row $p_{x+1,y}$. The equivalent calculation is done for the y direction.

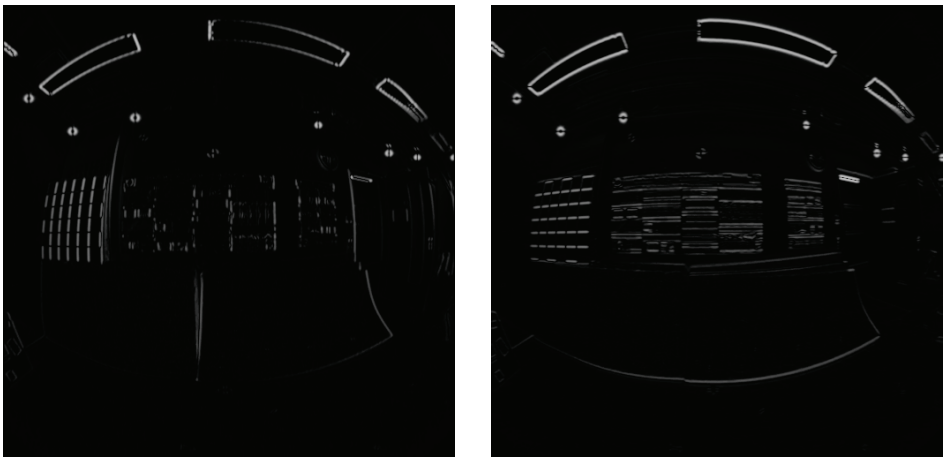

Figure 22: Image gradient example for the x and y direction of an image

This 1D Sobel kernel is a standard procedure in image processing [80]. Other implementations of the Sobel kernel use a Gaussian smoothing kernel in addition to the 1D Sobel kernel, to reduce noise errors. This has been shown to greatly increase the quality of the gradient image [80]. The following are three examples of Sobel kernels with various levels of Gaussian smoothing on the *x*-axis. Y-axis Sobel kernels are the transpose of the x-axis Sobel kernel.

$$\mathbf{S}_1 = \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{S}_3 = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

$$\mathbf{S}_5 = \begin{bmatrix} 1 & 2 & 0 & -2 & -1 \\ 4 & 8 & 0 & -4 & -8 \\ 6 & 12 & 0 & -12 & -6 \\ 4 & 8 & 0 & -4 & -8 \\ 1 & 2 & 0 & -2 & -1 \end{bmatrix}$$

with each line being convoluted with the Gaussian smoothing operator $[1\ 2\ 1]$ an additional time. The convolution of this Sobel kernel with an image returns the *change* in irradiance.

This result is the delta value in *pixel*-space. As the measurement of feature position is measured in the focal plane, we need to transform this change per pixel into a camera agnostic form.

This transformation can be achieved using the focal length $\mathbf{f}$ in the intrinsic camera parameters (refer to appendix B).

$$\frac{\partial \mathbf{I}_\ell}{\partial \mathbf{h}} = \frac{\partial \mathbf{I}_\ell}{\partial \text{pixel}} \frac{\partial \text{pixel}}{\partial \mathbf{h}} = [\text{Sobel}_x \ \text{Sobel}_y] \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix}$$

This is the only change necessary, when moving from the anchor Jacobians to the photometric measurement Jacobians.

6.4 Residual Calculation

As previously noted, eq. 6 and eq. 73 do not have the same form. To remove these parameters, Mourikis et al. [20] project the according Jacobian matrix, the residual and the noise onto its nullspace.

To use this method of removal, the linearized residual in eq. 73 is to be brought into the form of eq. 50. This is done equal to how the anchor-frame MSCKF residual linearization was reformulated. The stacking of the measurement Jacobians is done in the same way as shown in eq. 51 and figure 19 and will not be reiterated here.

6.5 Stereo-Photometry

Using the Jacobian formulation of the photometric expansion eq. 66 and the Jacobians of the anchor-frame stereo expansion eq. 60, we can combine both concepts by adding the image gradient derivative to the respective camera frame blocks:

$$\mathbf{H}_{\rho_{i,\ell}} = \frac{\partial h_\ell}{\partial \rho_i} = \frac{\partial \mathbf{I}_\ell^0}{\partial h} \frac{\partial h}{\partial {}^{C^0}\mathbf{p}_{i,j}} \frac{\partial {}^{C^0}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \tilde{\rho}_i} + \frac{\partial \mathbf{I}_\ell^1}{\partial h} \frac{\partial h}{\partial {}^{C^1}\mathbf{p}_{i,j}} \frac{\partial {}^{C^1}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \tilde{\rho}_i} \tag{69}$$

$$\mathbf{H}_{\mathbf{p}_{i,\ell}} = \frac{\partial h_\ell}{\partial \mathbf{x}_{\mathbf{p}\ell}} = \frac{\partial \mathbf{I}_\ell^0}{\partial h} \frac{\partial h}{\partial {}^{C^0}\mathbf{p}_{i,j}} \frac{\partial {}^{C^0}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}\ell}} + \frac{\partial \mathbf{I}_\ell^1}{\partial h} \frac{\partial h}{\partial {}^{C^1}\mathbf{p}_{i,j}} \frac{\partial {}^{C^1}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}\ell}} \tag{70}$$

$$\mathbf{H}_{\mathbf{p}_{i,A}} = \frac{\partial h_\ell}{\partial \mathbf{x}_{\mathbf{p}_A}} = \frac{\partial \mathbf{I}_\ell^0}{\partial h} \frac{\partial h}{\partial {}^{C^0}\mathbf{p}_{i,j}} \frac{\partial {}^{C^0}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}_A}} + \frac{\partial \mathbf{I}_\ell^1}{\partial h} \frac{\partial h}{\partial {}^{C^1}\mathbf{p}_{i,j}} \frac{\partial {}^{C^1}\mathbf{p}_{i,j}}{\partial {}^{G}\mathbf{p}_{i,j}} \frac{\partial {}^{G}\mathbf{p}_{i,j}}{\partial \delta\mathbf{x}_{\mathbf{p}_A}}. \tag{71}$$

This combines the past results of the anchor-frame formulation and the photometric formulation into a complete stereo-photometric Jacobian description.

6.6 Anchor-Image Approximated Stereo-Photometry

Instead of using the measurements - and the image gradient - of the current image as a reference value for the residual projection, we can use the measurements and gradient of the anchor

frame. This is the photometric formulation used by Zheng et al. [62].

The anchor-frame projection is equal to the pixel patch around the feature, as the feature position was estimated using the anchor-frame constraint. Using this data for every one of the measurements greatly reduces the cost of each measurement estimation but adds additional uncertainty to the residual. The irradiance vector of the anchor image $\boldsymbol{\xi}_i$ is the same for the entire feature. Due to changes in the exposure time $a_{i,\ell}$ of the different frames, the difference between two irradiance vectors of two different images might be skewed. To counteract this, the irradiance vectors of both images are normalized by their respective exposure time and an estimated exposure bias $b_{i,\ell}$. This models the change of irradiance through a linear function. Note, that we can split both the a and b parameter into a frame respective (a_ℓ, b_ℓ) and feature respective part (a_i, b_i). The exposure-normalized irradiance measurement in a frame is therefore

$$\frac{\xi_{i,\ell}}{a_{i,\ell}} - b_{i,\ell} = \frac{\xi_{i,\mathbf{A}}}{a_{i,\mathbf{A}}} - b_{i,\mathbf{A}}$$

from which we can set the irradiance estimation of the feature: ξ_i as

$$\xi_i = \frac{\xi_{i,\mathbf{A}}}{a_{i,\mathbf{A}}} - b_{i,\mathbf{A}}.$$

With this we can now form the measurement residual

$$\mathbf{r}_{i,\ell} \doteq \mathbf{I}_\ell \left(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p}_A}, \hat{\mathbf{x}}_{\mathbf{p}_\ell}\right) - \hat{a}_{i\ell}\hat{\boldsymbol{\xi}}_i - \hat{b}_{i\ell}\mathbf{1} \ . \tag{72}$$

It is simply the difference between the *expected* measurement, which is based on the prediction of the IMU, and the true measurement of the feature in the anchor image. The irradiance estimation assumes, that all camera measurements of the j points in 3D space are perfect - therefore, if the prediction is perfect, the change between the anchor frame and the considered frame should only be a function of the exposure time.

Parallel to the linearization of eq. 5, the photometric residual of eq. 72 is Taylor approximated.

$$\begin{aligned}
\mathbf{r}_{i,\ell} &\doteq \mathbf{I}_\ell \left(\hat{\rho}_i, \hat{\mathbf{x}}_{\mathbf{p_A}}, \hat{\mathbf{x}}_{\mathbf{p}_\ell}\right) - \hat{a}_{i,\ell}\hat{\boldsymbol{\xi}}_i - \hat{b}_{i,\ell}\mathbf{1} \\
&= \mathbf{I}_\ell \left(\rho_i - \tilde{\rho}_i, \mathbf{x}_{\mathbf{p_A}} - \tilde{\mathbf{x}}_{\mathbf{p_A}}, \mathbf{x}_{\mathbf{p}_\ell} - \tilde{\mathbf{x}}_{\mathbf{p}_\ell}\right) - \hat{a}_{i,\ell}\hat{\boldsymbol{\xi}}_i - \hat{b}_{i,\ell}\mathbf{1} \\
&\simeq \mathbf{I}_\ell \left(\rho_i, \mathbf{x}_{\mathbf{p}_A}, \mathbf{x}_{\mathbf{p}_\ell}\right) - \mathbf{H}_{\rho_\ell}\tilde{\rho}_{i,\ell} - \mathbf{H}_{\mathbf{p}_\ell}\tilde{\mathbf{p}}_{i,\ell} - \mathbf{H}_{\mathbf{p_A}}\tilde{\mathbf{p}}_{i,\mathbf{A}} - \hat{a}_{i,\ell}\hat{\boldsymbol{\xi}}_i - \hat{b}_{i,\ell}\mathbf{1} \\
&\simeq \mathbf{I}_\ell \left(\rho_i, \mathbf{x}_{\mathbf{p_A}}, \mathbf{x}_{\mathbf{p}_\ell}\right) - \mathbf{H}_{\rho_\ell}\tilde{\rho}_{i,\ell} - \mathbf{H}_{\mathbf{p}_\ell}\tilde{\mathbf{p}}_{i,\ell} - \mathbf{H}_{\mathbf{p_A}}\tilde{\mathbf{p}}_{i,\mathbf{A}} - (a_{i,\ell} - \tilde{a}_{i,\ell})(\boldsymbol{\xi}_i - \tilde{\boldsymbol{\xi}}_i) - (b_{i,\ell} - \tilde{b}_{i,\ell})\mathbf{1} \\
&\simeq \mathbf{I}_\ell \left(\rho_i, \mathbf{x}_{\mathbf{p_A}}, \mathbf{x}_{\mathbf{p}_\ell}\right) - \mathbf{H}_{\rho_\ell}\tilde{\rho}_{i,\ell} - \mathbf{H}_{\mathbf{p}_\ell}\tilde{\mathbf{p}}_{i,\ell} - \mathbf{H}_{\mathbf{p_A}}\tilde{\mathbf{p}}_{i,\mathbf{A}} - a_{i,\ell}\boldsymbol{\xi}_i + \tilde{a}_{i,\ell}\boldsymbol{\xi}_i + a_{i,\ell}\tilde{\boldsymbol{\xi}}_i - \tilde{a}_{i,\ell}\tilde{\boldsymbol{\xi}}_i - b_{i,\ell}\mathbf{1} + \tilde{b}_{i,\ell}\mathbf{1}
\end{aligned}$$

removing higher order error terms and injecting the true irradiance description, we receive:

$$\mathbf{r}_{i,\ell} \simeq \ -\mathbf{H}_{\rho_{i\ell}}\tilde{\rho}_{i,\ell} - \mathbf{H}_{\mathbf{p}_{i,\ell}}\tilde{\mathbf{p}}_{i,\ell} - \mathbf{H}_{\mathbf{p}_{i,\mathbf{A}}}\tilde{\mathbf{p}}_{i,\mathbf{A}} + \tilde{a}_{i\ell}\hat{\boldsymbol{\xi}}_i + \hat{a}_{i\ell}\tilde{\boldsymbol{\xi}}_i + \tilde{b}_{i,\ell}\mathbf{1} + n_{i,\ell}. \tag{73}$$

The description of the residual (eq. 72) and more importantly its function h - which returns the irradiance - is a function of the estimated distance to the anchor frame ρ, the estimation of the anchor pose $\hat{\mathbf{x}}_{\mathbf{p}_A}$ and the estimation of the considered frame pose $\hat{\mathbf{x}}_{\mathbf{p}_\ell}$. Note, that the individual Jacobians therefore stay the same as in eq. 66 but the resulting Jacobians for the nullspacing are stacked differently. Notably so, because we have an array of different terms in the residual which are not tracked in the state vector, such as ξ, a and b. For this reason, we must construct the Jacobians as shown in the following part. As a reminder, the resulting structure we want to achieve to correctly be able to nullspace the feature position information is

$$r_{i,\ell} \simeq \mathbf{H}_x \tilde{x} + \mathbf{H}_y \tilde{y} + \mathbf{n}.$$

Note here, that as in the work of Zheng et al. [62], it is possible to estimate any of the photometric parameters a_ℓ, $a_{i,\ell}$, b_ℓ or $b_{i,\ell}$ by adding them to the state-vector in the camera state description. The general vector χ represents a collector for any photometric parameters in the state vector, making a single camera-state

$$\mathbf{x}_{C_\ell} = \begin{bmatrix} {}^{C_\ell}_{G}q \; {}^{G}\mathbf{p}_{C_\ell} \; \chi \end{bmatrix}^\top.$$

With this modification, the error state of a camera is now multiplied by a Jacobian structured as follows:

$$\mathbf{H}_\mathbf{X} = [\mathbf{0} \dots [-\mathbf{H}_\mathbf{A} \; \mathbf{0}] \dots \mathbf{0} \dots [-\mathbf{H}_{C_\ell} \; \mathbf{1}] \dots \mathbf{0}]^\top$$

with the $\mathbf{0}$ and $\mathbf{1}$ in the sub-segments of the segments being the parts, which are multiplied with χ all the photometric parameters corresponding to the respective camera frame. Note, that these are the *estimated* values which we included in the state vector. The structure of $\tilde{y}$ is straightforward, accumulating all the parameters, which are used but not tracked inside the state vector. Note that the structure differs depending on which of the four photometric parameters are tracked and which are not. All four parameters are added here, to demonstrate the process. Any variables which one wishes to estimate in χ are of course removed from $\tilde{y}$ and from $\mathbf{H}_y$ correspondingly.

$$\tilde{y} = \begin{bmatrix} \tilde{\xi}_i^\top \; \tilde{a}_{i,\ell} \; \tilde{b}_{i,\ell} \end{bmatrix}^\top$$

The description of $\mathbf{H}_y$ is therefore

$$\mathbf{H}_y = \begin{bmatrix} \hat{a}_{i,\ell}\mathbf{I}_{N\times N} \; \begin{bmatrix} \mathbf{0} \dots \hat{\xi}_i \dots \mathbf{0} \end{bmatrix} \; \mathbf{1} \; -\mathbf{H}_{\rho_{i,\ell}} \end{bmatrix}^\top.$$

The inner block regarding the irradiance estimation $\hat{\xi}_i$ is constructed depending on the current considered frame. Compare to eq. 73, where it is multiplied with $a_{i,\ell}$. Figure 23 visualizes this. Note, that for the nullspacing to work, the size of a patch must be larger than 1 $\times$ 1, as without it the matrix has 0 degrees of freedom.

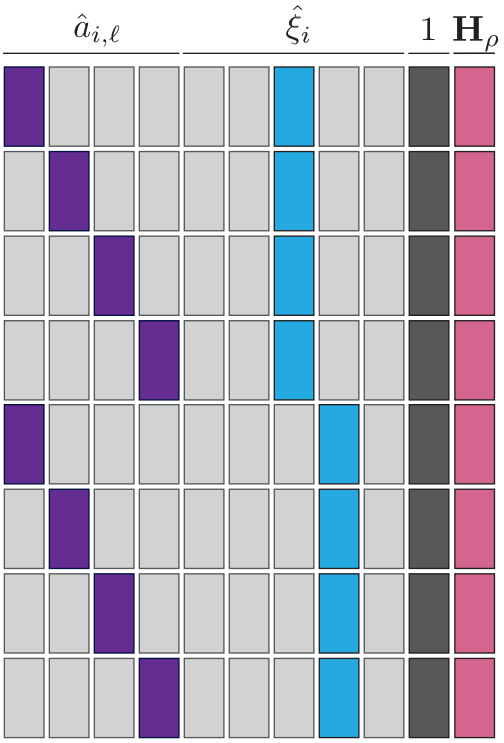

Figure 23: Creating the Jacobian used to nullspace the residual - two frames stacked on top of each other, each with a 2 $\times$ 2 patch size

6.7 Outlier Detection

Not all features are equivalent in their produced update step. Patch-gradients can be higher or lower than the desired true update step would be. Some patch information is only highly local to the patch, while other patch update directions are valid for larger segments of the picture, see figure 24. The figure shows two different patches. One, where the area in which the reprojected patch produces the correct update direction is large, and one where the region is not much larger than the patch itself. Note, that while the update direction may be correct, the update magnitude is still prone to error, unless the image gradient is perfectly even.

Further, features can be matched incorrectly by the feature extractor front-end or a features inverse depth ρ can be off. Both of these sources of error lead to highly incorrect patch reprojection in the images. To remove such update outliers, the Mahalanobis gating test is used. A Mahalanobis gating test calculates the weighted distance of the residual to the mean of the current estimation [81]. The Kalman Filter lends itself to this form of outlier detection, as the covariance of the current state is optimally estimated alongside the mean state-vector prediction.

Figure 24: Two patches with different local behavior in the same image frame

As the error-state formulation of the Kalman Filter has a mean prediction of zero, the formulation of the resulting mean square error γ is

$$\gamma = \mathbf{r}^\top (\mathbf{H}\mathbf{P}\mathbf{H}^\top + \mathrm{n}\,\mathbf{I})^{-1}\mathbf{r}$$

with the state transition Jacobian $\mathbf{H}$, the state covariance $\mathbf{P}$, the noise parameter n and the residual $\mathbf{r}$. The resulting γ is compared to chi-squared lookup table, with the desired quantil parameter. The degrees of freedom for the chi-squared gating test is the dimensions of the residual. See Ugoni & Walker [82] for more details on the topic of the Chi square gating test.

The Mahalanobis gating test inherently tends to eliminate any low patch-gradient, as it is a part of the covariance of the state description. Large changes in the residual lead to a large update, which get eliminated as well, unless the covariance is already large.

7 Implementation and Evaluation

The formulation and derivation of the three position estimators - the stereo MSCKF, the stereo anchor-based MSCKF and the stereo-photometric MSCKF have been presented in the last few chapters. When referring to the photometric MSCKF, the stereo-photometric MSCKF is meant. The implementation of the estimators is based on the open-source MSCKF of Sun et al. [45]. The existing feature extractor front-end and the IMU prediction step are used directly, the novel update procedures are implemented separately, compare to figure 6. The implementations will

be evaluated in this chapter using a state of the art open-source VIO dataset.

After a brief look at the necessary parameters used by these algorithms and a section regarding the implementation of the moving window, we will present the evaluation techniques used to compare our approaches of the MSCKF implementations. Subsequently, we will detail the results of the evaluation.

7.1 Necessary Parameters

With the complete MSCKF formulation in place, we have a collection of parameters, which must be set and tuned. This section will give a comprehensive list of these variables and a brief explanation of why the respective values were chosen to optimize the stereo-photometric MSCKF output.

The feature extractor parameters are presented in table 4. The grid parameters control the size of the grid and the number of features per cell, as in figure 17. The Lukas-Kanade feature tracker is parameterized by its pyramids' depth and the size of the patches used for feature tracking in the pyramid. This iterative tracker is bound by a maximum iteration and a minimum change in parameters - defined by the track precision. The tracked FAST features have a threshold constraining the minimum illumination change around the point of interest.

Table 4: Feature extraction parameters

grid row	3-5
grid column	3-5
grid minimum feature number	2-4
Grid maximum feature number	4-6
Pyramid Levels	3
Patch size	15
max iteration	30
Track precision	0.01
Fast threshold	10
Gating quantile	0.05

All these parameters depend on the camera resolution and the scene recorded. The values in table 4 are values that have worked well for Sun et al. [45] and shown robust results in our experiments using the different recordings of the TUM dataset. A minimum pyramid level of 3 is typically necessary to allow useful feature measurements to pass for the MSCKF algorithm. The values regarding grid size and amount of features are different for the various scenes and datasets. Any grid size below three shows too much clustering of the features in the recordings tested to allow the algorithm to function properly. Grid sizes larger than five need a large number

of minimum features per grid to not disregard vital features. This would put a heavy toll on the resources used by the algorithm.

Table 5: MSCKF algorithm parameters

moving window size	12-20
patch size	3
measurement noise values	0.01
initial covariance	0.01

Table 5 shows the parameters set for the MSCKF algorithm. It is parameterized mainly through the size of the moving window, the initial covariance values, and the noise values. The Mahalanobis gating test is further controlled through the desired quantile. The stereo-*photometric* approach is additionally controlled through the measurement patch-size and the image scale. While the most accurate result inside the error tolerance of the stereo-photometric implementation was achieved using a 5 $\times$ 5 patch, the more robust approach (with more features passing the gating test) was a residual using a 3 $\times$ 3 patch.

Tests showed, that using a moving window size below 12 frames reduced the accuracy in the feature estimation too much, to allow for successful tracking for any of the implementations. Values over 20 frames lead to a reduced number of features passing the gating test, which decreased the value of the update step. The measurement noise values are set to 0.01, which according to empirical evaluation maximizes the number of features passing over to the update step. The values used by Sun et al. regarding the IMU noise parameters are retained, as are the covariance matrix initialization values of 0.01 along the trace.

7.2 Moving Window Management and Patch Size

The proposed moving window frame removal steps from Sun et al. [45] estimate the lowest information content frames based on the predicted change in position between the frames. The frames with the smallest change to its neighbors are removed from the moving window. Zheng et al. [62] on the other hand simplify this process by removing the oldest frame in the buffer. As this latter approach has been shown to have worked for photometric implementations and feature-based implementations by Mourikis et al. [20], the strategy was used for this stereo-photometric implementation, as well as for the anchor-frame based MSCKF.

Zheng et al. evaluate different patch sizes, ranging from 4 $\times$ 4 to 7 $\times$ 7. Our implementation had the least update step rejection when using 3 $\times$ 3 pixel patches, which is why the following evaluation will use this patch size. Note, that the size of the patch in the real world is different depending on the image resolution and the distance to the camera. The larger a feature patch is, the greater the noise induced into the residual by any irradiance changes around the, but not

connected to, the actual feature. Therefore these factors may contribute to the different results in optimal accuracy between the two implementations. To reduce the computational stress on the algorithm, the size of the image was down-scaled. This reduces the optimal value of the patch size by the same factor and makes the image gradients more pronounced.

7.3 Methods of Evaluation

The hardware used for evaluation was a 2.8 GHz Intel Core i5 5675C quad-core processor with 8Gb of RAM. The TUM VIO dataset [74] was used to evaluate the performance of the presented algorithms for a total of 14.5 hours of evaluation per algorithm.

The MSCKF pipeline is non-deterministic, as the feature extractor uses RANSAC. Therefore when using the entire MSCKF pipeline for evaluation, the results would be skewed based on the individual results of the feature extractor. As the feature extraction itself is *independent* of the state estimation, the same random seed can be used for individual runs. This assures the same features and feature rankings in every frame for every run with this seed. With this, only the change in the update-step itself is evaluated. As the performance of the estimators varies depending on the respective quality of the features, five differently seeded feature extractions were used per dataset.

We compare the algorithms based on their relative change to ground truth per time-step, their root mean square error (RMSE) when completing a run and their time until divergence if they did not. The divergence rates per algorithm - the number of successful runs - per dataset are measured as well. The CPU load of the algorithms on one kernel using ten frames per second performance is measured to compare computational advantages. The playback speed of ten frames per second was chosen, to ensure that none of the algorithms experienced any frame-drops during testing. The error in the relative steps is a particularly interesting result for the stereo-photometric implementation. It allows us to evaluate the mean accuracy of the algorithm when operating within the limits of its error threshold. Any data past a point of divergence from the ground truth is discarded for this evaluation. A filter is considered divergent, if any update step change is larger than 0.5 meters, no update step has been generated for over one second or if the relative error of the estimation step is larger than 100% for longer than one second.
As VIO has not global optimization, the relative pose error [83] is calculated using the change in position as follows:

$$r_{rpe} = \sqrt{\sum_i ||trans(\mathbf{E}_i)||^2}$$
$$\mathbf{E}_i = \left(\hat{\mathbf{T}}_i^{-1}\hat{\mathbf{T}}_{i+\Delta}\right)^{-1}\left(\mathbf{T}_i^{-1}\mathbf{T}_{i+\Delta}\right)$$

using only the translation ($trans(\cdot)$) component of the pose error for the error calculation.

$\mathbf{T}_i$ is the true pose, $\hat{\mathbf{T}}_i$ is the estimated pose. i and $i + \Delta$ are the last and current measurement time-step.

The selected evaluation methods are a combination of the evaluation steps used by various other VIO research papers. This paragraph gives an overview of how theses papers assessed their position estimation algorithms.

Mourikis et al. [20] evaluate their algorithm by measuring the final position error of the estimated trajectory in a moving car. Sun et al. [45] evaluate their implementation using the EuRoC dataset [68] and compare the root mean square error of the final position, as well as the CPU load to various other state estimators (OKVIS, ROVIO, and VINS). Zheng et al. [62] evaluate their photometric approach by creating their own dataset, using ground truth information from a motion capture system. In their paper the results are compared to their implementation of the feature-based MSCKF. The root mean square error for the entire trajectory is measured, to show the expected performance. The 90th percentile error is computed throughout the run of all recordings. The average represents the expected minimum performance of the algorithm. OKVIS [46] is similarly evaluated by calculating the error over the course of the entire trajectory. The datasets used are custom created and include 8 km long trajectories from car rides, walking in small circles and walking long indoor loops to test their loop-closure process. Trifo-VIO [53] is evaluated by a custom made dataset, moving the sensors with a 6-axis robot, showing a high amount of rotation. Additionally, the EuRoC dataset is used, to compare the design to the stereo MSCKF and OKVIS. VINS Mono [49] also uses the EuRoC visual-inertial dataset to compare the total trajectory error. The resulting accuracy is compared to OKVIS.

Unlike the EuRoC [68] and the KITTI dataset from Geiger et al. [84], the TUM dataset [74] includes the camera's exposure time in each frame. This information is used for the stereo-photometric residual. The Trifo-VIO dataset [53] shows recordings of heavy rotation with little translation and change of scenery compared to TUM and EuRoC recordings. For this reason, the more sensor-information complete and scenery-expansive TUM VIO dataset [74] was used for the following evaluation.

This dataset includes a multitude of different recordings with partial ground truth coverage measured by a Motion Capturing system. The measurements include the shutter-time of a stereo camera setup as well as IMU measurements and they supply an ample amount of visual-inertial calibration data for noise measurement and camera and IMU calibration. For these evaluations, the TUM supplied Kalibr calibration results were used. The cameras are calibrated using the equidistant model for distortion parameters. This TUM VIO dataset contains a total of 28 different recordings. Five of these recordings are movement in a corridor with severe lighting changes and heavy rotation. Six recordings move the camera through a large open indoor space with heavy sunlight and shadow movements. Eight recordings of the outdoors show spacious movement of the agent over the course of around 20 minutes per recording.

Five recordings are supplied in which the camera is moved around solely inside a room with constant ground truth information. These recordings contain a lot of rotation and twisting of the camera setup. The final three recordings show the camera moving indoor and going down a slide. Again, heavy lighting changes and long stretches of few features are present.

7.4 Evaluation Results

The following figures 25, 26, 27 and 28 as well as the table 6 give an array of overview-charts regarding the five different *settings* of the recordings in the TUM VIO dataset (Corridor, Magistrale, Outdoor, Room, Slide). The averages over all recordings in the same setting are generated for each algorithm and then compared. E.g. all six Room recording results are used to construct an average measurement result for each of the three algorithms. Note, that from the various results, only completed runs are used to construct the average.

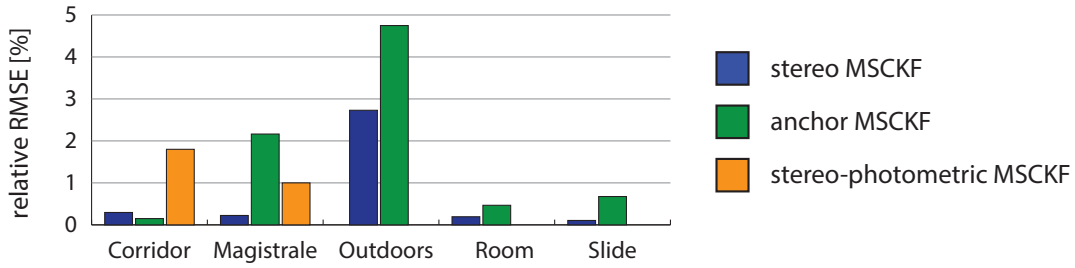

Figure 25: The average RMSE position error based on the change in movement compared to the ground truth change, corrected for rotation and using all estimates before divergence detection

The most relevant measurement in VIO comparison is the relative error per timestep. This type of evaluation can only be done with available ground truth. All recordings in the TUM dataset have ground truth measurement in the room in which the recordings begin and end. Note, that for this evaluation, the pose estimations are used up until the algorithm diverges - any subsequent estimations are discarded. We will present and discuss the divergence rate of the different filters separately.

Figure 25 shows the error in the delta change of the various approaches. On average over all datasets, the error of the stereo MSCKF is, at 0.82%, half the size of the error produced by the anchor-frame based estimator at 1,68%. The average for the stereo-photometric based algorithm lies around 1.28%. As the Outdoors recordings show a significant increase in relative RMSE for the other two algorithms - with the stereo-photometric approach diverging within the first five seconds, these recordings distort the resulting comparison.

If we only evaluate the timeframes during which all three algorithms converge, the resulting relative RMSE are 0.19%, 0.29% and 1.1% for the stereo, the anchor and the stereo-photometric MSCKF respectively.

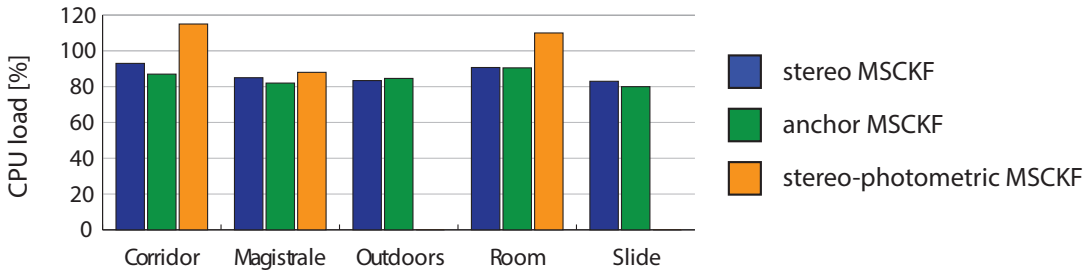

Figure 26: The average CPU load during the non-divergent sections of the algorithm

CPU performance of the algorithm itself is shown in figure 26. Note, that as the feature measurements were pre-recorded for these evaluations, the CPU measurements represent the actual MSCKF algorithm, without the CPU heavy feature extractor front-end. The stereo-photometric approach uses the most CPU power with a total average of 104.3% kernel usage. The performance of the anchor-based feature estimator at on average 84.8% kernel usage is marginally better than the power consumption of the stereo MSCKF with 87%.

Table 6: Nr. of completed runs per recording scene

Recording	**Nr. of runs**	**Completed runs**		
		stereo	anchor-frame	stereo-photometric
Corridor	15	12	12	2
Magistrale	18	8	0	0
Outdoors	24	3	1	0
Room	18	16	14	0
Slide	9	5	0	0

The number of successful runs and the average time before the filter diverges demonstrate the overall robustness of the algorithm. While all of the MSCKF based implementations struggle to complete the TUM datasets, the stereo-photometric design shows an especially high divergence rate, with very specific points of failure in the different sets. We will discuss these points of failure in the coming section. Table 6 shows the overall success rate of the algorithm split into the different types of courses recorded (Corridor, Magistrale, Outdoor, Room, Slide). The divergence rate over all datasets is 42% for the stereo MSCKF, 67% for the anchor-frame MSCKF and 97% for the stereo-photometric MSCKF.

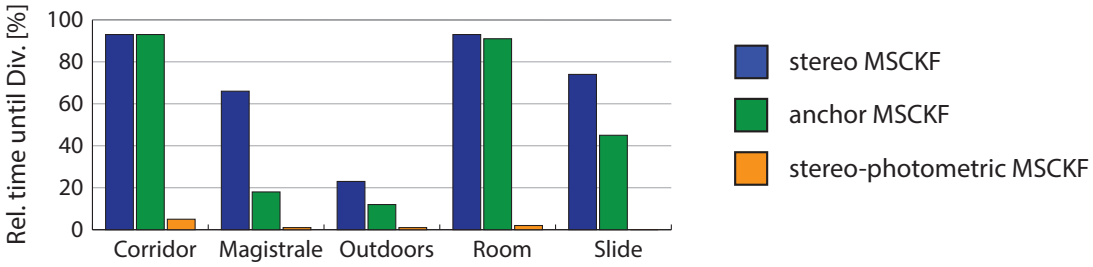

Figure 27: The average percent of divergence over the entire time of the scenes

Figure 27 shows the average amount of seconds the VIO algorithm remains stable in the respective course type. Note, that for the previous evaluations, only the time in which the algorithm showed convergent behavior was used. These results are relative to our definition of divergence, as explained in the previous section. The total percentage of convergent time over all datasets is 70% for the stereo MSCKF, 52% for the anchor-frame MSCKF and 2% for the stereo-photometric MSCKF.

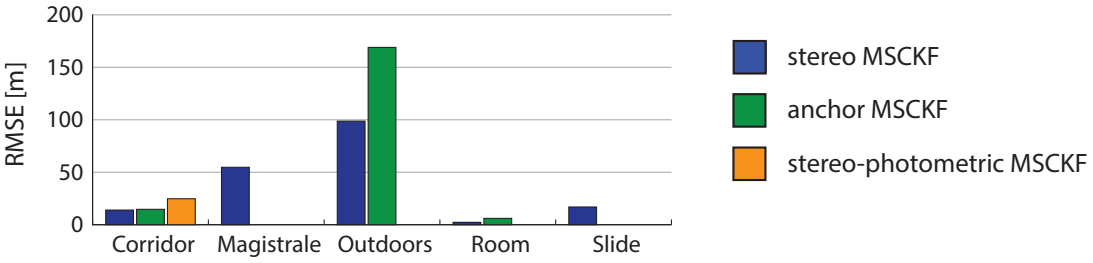

Figure 28: The average RMS error when comparing the starting position to the final position

Figure 28 compares the resulting final position inside the room with the available ground truth data information. Compare this to table 6, which shows how often the respective algorithms succeeded in tracking through the entire course of the algorithm.

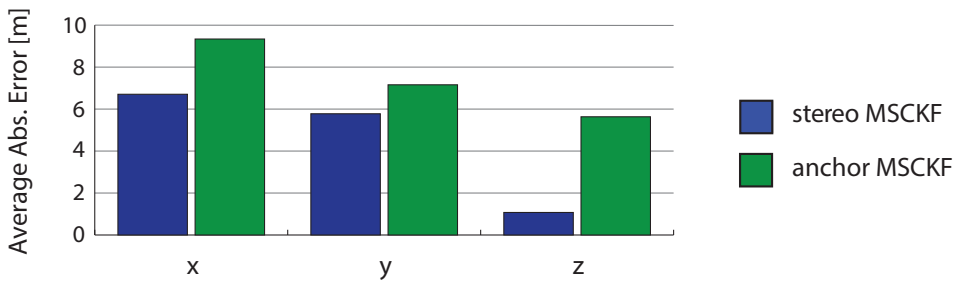

Figure 29: Comparison of the final position error split up to show the error in each axis - for the two datasets with the most complete runs

As visualized in figure 28 the total accumulated error for the different implementations tends to be quite large. The stereo-photometric implementation has very few completed runs, as it tends to diverge at specific points in the dataset. This comparison therefore mainly shows the stereo based approach and the anchor-frame based approach. The stereo based approach tends to have a reduced RMSE. Splitting the resulting error up per axis, as shown in figure 29, we can see that generally, the z error is smaller than the x and y error. While between the two different types of update steps the x and y error components remain largely the same, the z error in the anchor-frame estimation is over five times larger than the original design. When visualizing exemplary recordings of Room 4 in figure 30, the z-height drift becomes visible.

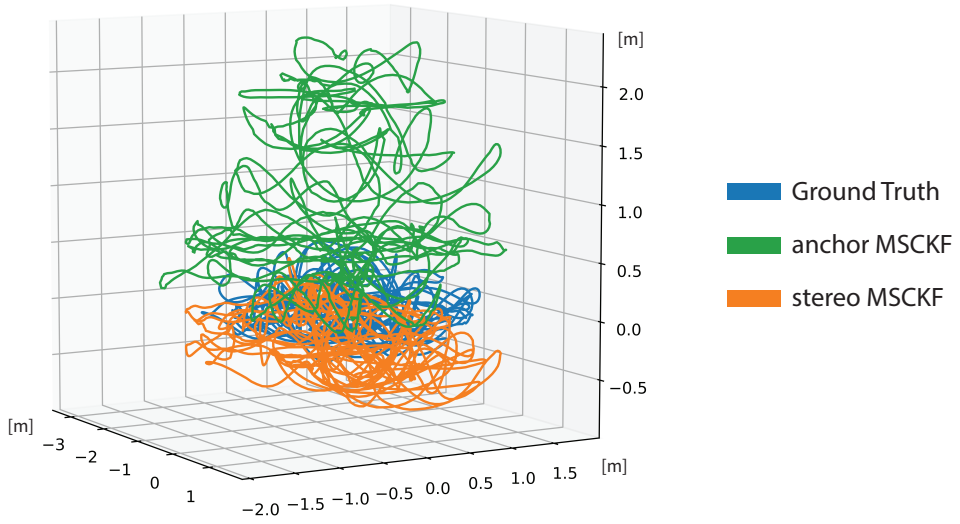

Figure 30: Error in z-axis estimation in Room 4

8 Discussion and Next Steps

The preceding chapter clearly shows a lack of robust behavior in the stereo-photometric implementation. This chapter will analyze the different reasons as to why this is the case and present possible solutions. The first section 8.1 will look at error tolerances in the feature projections with specific examples. In the subsequent section 8.2, we will present the difference between low-quality and high-quality features and the algorithms reaction to various input. In section 8.3, the number of features passing through the outlier-rejection are compared and correlated with the rate of divergence. Section 8.4 discusses the results of the anchor-based CPU improvements and leverages them against the decreased accuracy shown in our test.

8.1 Error Tolerance

As shown in the image summary of the photometric residual in figure 20, the algorithm is designed to minimize errors in patch misalignment. This approach works well for patch movements close to the original patch, as the direction of the gradient is correct in the local region of the feature. The farther the patch moves outside the original zone, the less likely it is, that the irradiance of the reprojected patch correlates with the original irradiance. Therefore, the greater the error in the prediction of the patch position, the poorer the performance of the correction step in the stereo-photometric residual. Compare this to the feature-based correction step, where the feature position itself is used, and the inaccuracy in the residual is only a function of the feature tracking itself. With this implementation, the feature estimation error is essentially correctable over the entire size of the image.

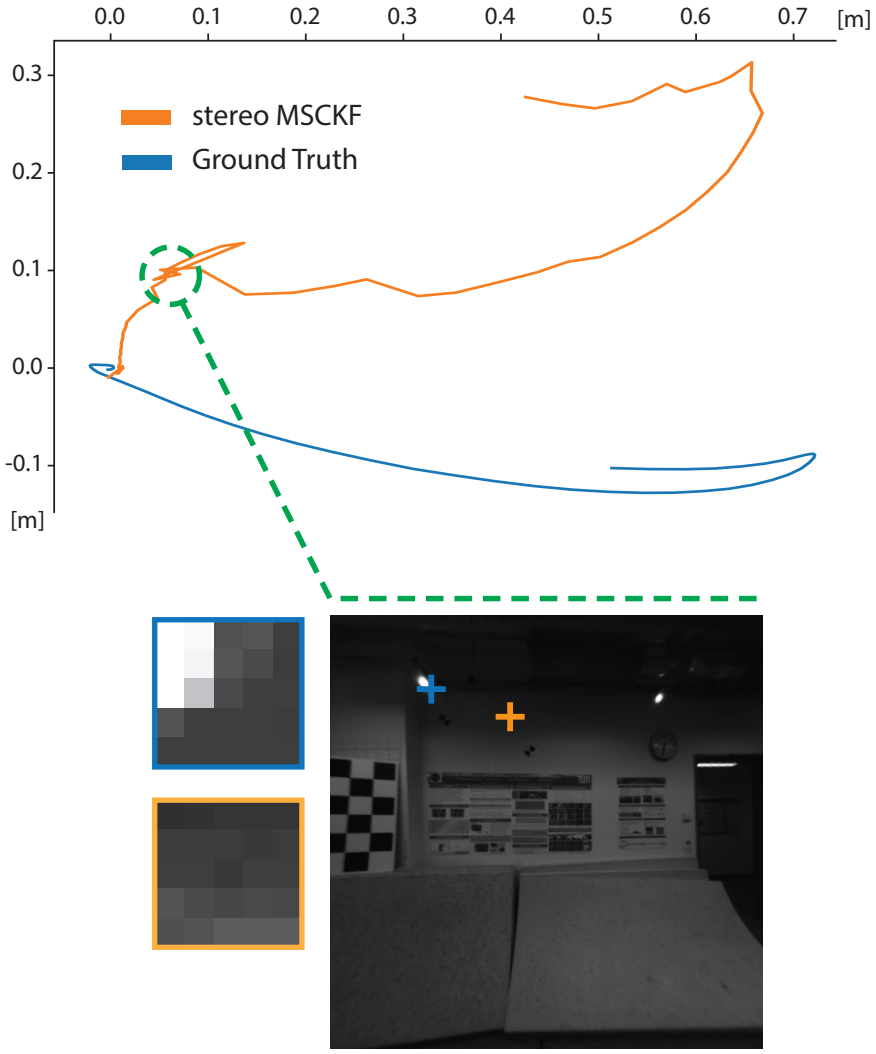

Figure 31: Error in prediction of the first two seconds of Room 3 compared to ground truth and the resulting feature reprojection errors versus the resulting pixel patches

The error tolerance in the stereo-photometric implementation is far smaller than in the feature-based implementation. Therefore, both the IMU-based prediction and the individual correction steps must operate within the error margins allowed by the patch size. For this reason, the error tolerance is a function of the patch size and the accuracy of the prediction. Figure 31 shows the error in the prediction of the Room 3 recording. The initial error of the IMU prediction - and the resulting estimated feature position - is far larger than the patch size would allow for. This is shown by the resulting patch projections in the frames. Note, that the feature position based estimators are easily able to correct this shift, while the stereo-photometric implementation diverges within the first second.

The same phenomenon of high errors in the prediction leading to the divergence of the stereo-photometric filter is observed in multiple instances and is a prime reason for the lack of robust behavior. Further examples include the prediction in Magistrale 1, Outdoor 5 and Slide 2 in figure 32.

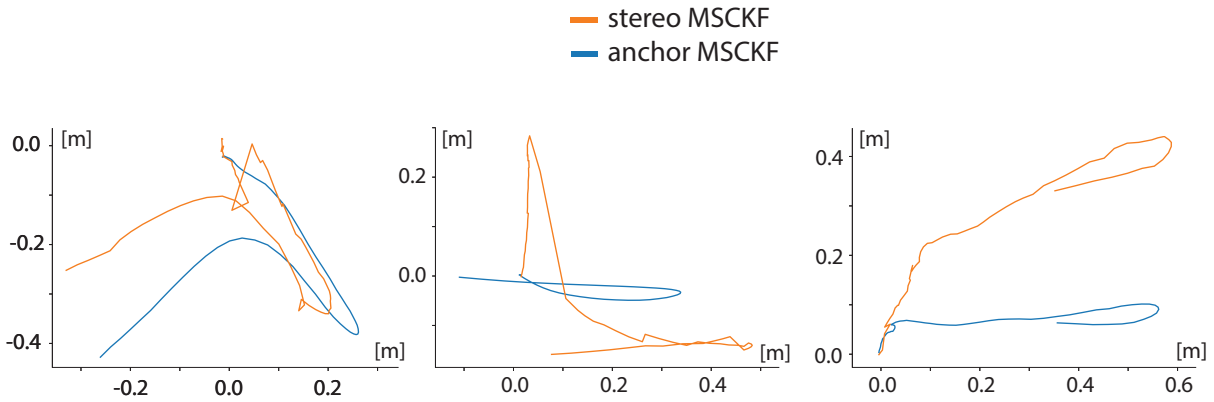

Figure 32: Various examples of erroneous prediction steps going beyond the maximum tolerance of the photometric implementation

Another factor in the update step calculation is the feature position estimation in space, which is intrinsically inferior in the anchor-based approach (refer back to figure 4), adding to the inaccuracy of the stereo-photometric implementation.

8.2 Feature Patch Quality

The feature patches quality is determined by how pronounced the gradient is and how well the gradient change in the surrounding of the patch correlates to the gradient of the patch itself. Different patch examples can be seen in figure 33. Patches are more likely to result in a correct update step if the image gradient is more pronounced and well defined. The more defined a feature surrounding is, the more accurate its tracking over the course of multiple images can be. A weaker patch gradient forces larger residuals to move the resulting correction

farther as the necessary change in u, v direction is large. This is especially problematic when the surroundings of the patch - into which the projection might fall - show a higher change in irradiance compared to the gradient itself.

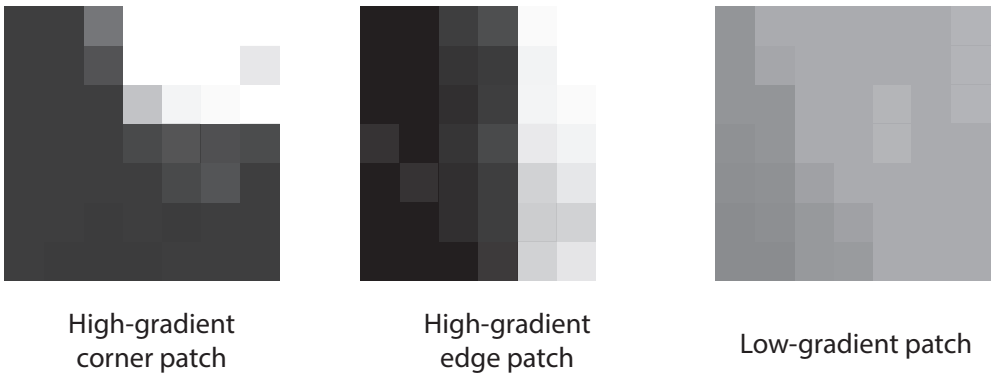

Figure 33: Three common types of patches output from the feature extracting process

High patch gradients tend to behave more robust regarding these phenomenons, as high erroneously associated irradiance changes only produce small changes in the correction. Therefore, high-gradient patches are generally desirable to use in the update step. Note, that this is exactly the filter result of the Mahalanobis gating test: low-patch gradients are removed due to them lowering the values of the propagated covariance matrix - decreasing the chance of a residual passing (see chapter 6.7).

The patch size controls how much of the feature surroundings are included in the gradient and the residual. The larger the patch is, the greater the probability, that fragments of non-feature elements skew the gradient direction and therefore the resulting update. A smaller patch size is preferable, so long as the correlated residual patch is large enough to encompass the actual facets, which make the feature locally unique and create a well-defined update step.

8.3 Amount of Accepted Features

Figure 34 shows an example run of a Corridor recording, comparing the number of features and the relative accuracy. The number of features used per update is plotted in orange. The blue plot visualizes the relative RMSE accuracy corresponding with the update step. Note, that the plot only covers the section of the recording with available ground truth, as we can only measure relative RMSE in these sections. In timestamp one, at half a second, a strong relative error is measured. This error is similar to the one visualized in figure 31 and a common phenomenon when the agent is not moving. The red marking at timestamp two shows the time of divergence for the stereo-photometric MSCKF, which correlates with the peak relative error in the anchor-

frame MSCKF implementation. This peak is not nearly as pronounced in the stereo MSCKF. The same phenomenon can be seen at timestamp four. Timestamp three shows a peak in the number of features detected by the stereo MSCKF which is missing from the anchor-frame MSCKF. This increased number of features might explain why the error remains lower in the following second. Comparing the number of features measured in the time between timestamp three and four shows, that the stereo MSCKF consistently measures more features. Note, that during the time between timestamp one and two, the stereo-photometric MSCKF uses far fewer features than both feature-based implementations.

We see, that the number of features in the stereo-photometric approach is 89% lower than in the anchor-frame MSCKF and experiences a substantial dip moments before diverging. The low number of features is explained by the strict outlier removal process, which is chosen to minimize erroneous update steps created by patch projections outside the immediate surroundings of the original patch. In our experiments, the divergence rate dropped further, the more lenient the outlier detection was tuned (refer to section 4 for tuning parameters).

8.4 Anchor-Frame Modification

The evaluation in chapter 7 shows a decrease in performance accuracy when switching from the anchor-frame MSCKF (constraining the estimated feature position via the first observation) to the stereo MSCKF (estimating the feature position constrained only by measurements). The time until divergence decreases by 30% from 97% to 67% and the relative RMSE more than doubles from 0.82% to 1.68%. This decrease in accuracy is especially noticeable in the Magistrale and the Slide recordings, where the reduced estimation precision results in the majority of the evaluations diverging. The sections in which divergence rates peak, are after a few moments of standing still or during reduced lateral movement. Both of these scenarios show a decrease in features passing the outlier-rejection phase. This is true for both the stereo MSCKF and the anchor-frame MSCKF, but the anchor-frame MSCKF's drop in features is far pronounced, see figure 34.

The marginal increase in CPU performance of the anchor-frame based MSCKF is due to the reduction in complexity for the feature estimation, as well as the smaller nullspace matrix needed in the calculations of the update step of eq. 50. When leveraging this 2.2% decrease in resource necessity against the substantial reduction in estimation accuracy, the stereo MSCKF formulation remains the more desirable update procedure.

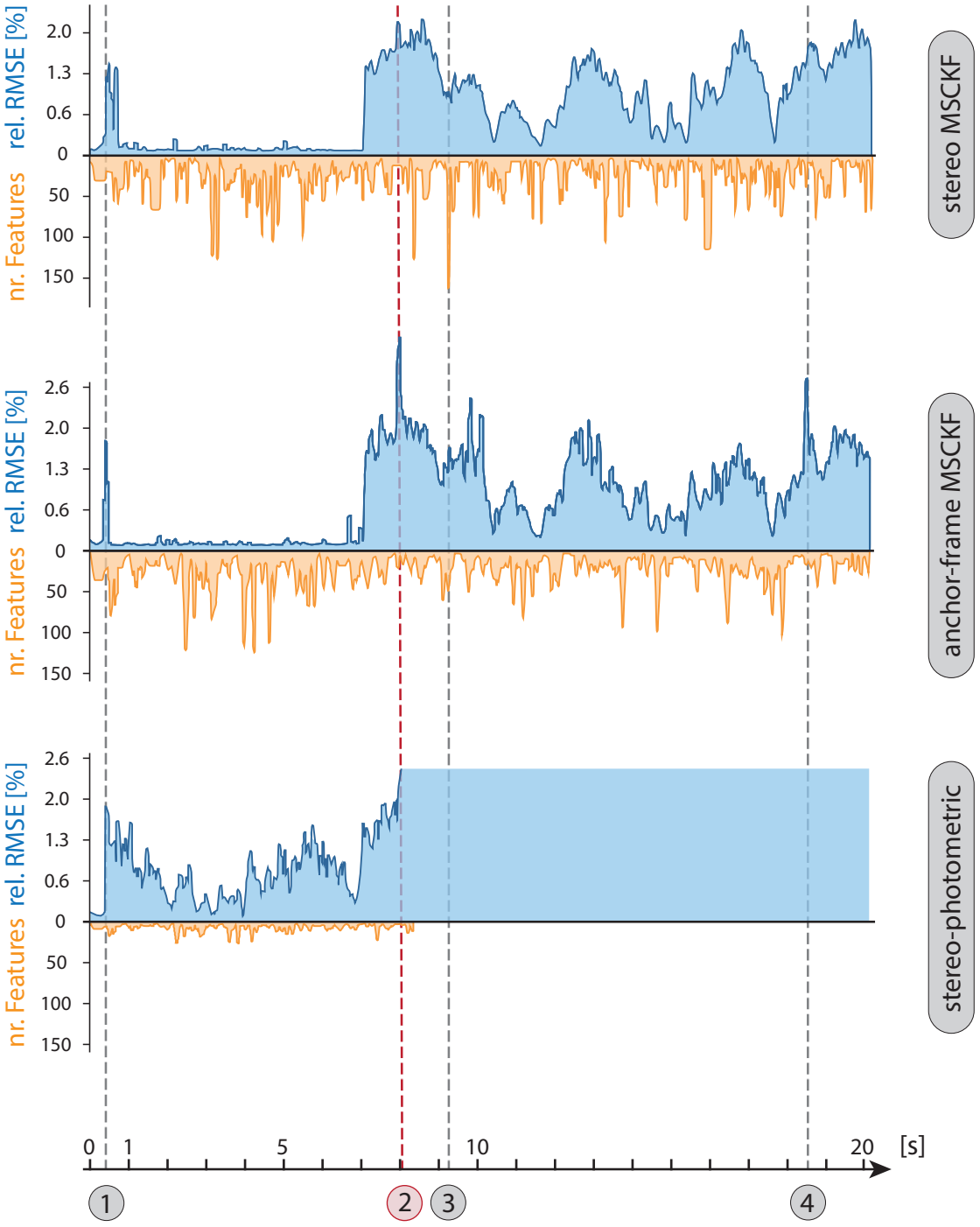

Figure 34: Comparing the number of features used in the update step for each algorithm to the relative RMSE in each time step using the first 20 seconds of the Corridor 1 recording

8.5 Improvements

As the evaluation chapter has shown, the stereo-photometric approach in its current form is in no way fit to be used as an alternative to the stereo MSCKF. While we can see stretches of convergence during ideal segments, the lack of robustness when diverging past the error tolerance of the patch makes the algorithm no match for a feature-based estimator. This section will propose some adaptions of the algorithm, which might increase stability and accuracy.

Note, that the work of Zheng et al. [62] proposes a photometric adaption to the *mono*-MSCKF and achieves more preferable results than the estimator presented in this thesis. Comparing both approaches, the stereo-photometric design is similar. We chose to pre-undistort the images into a pinhole model, before collecting the irradiance information from the image. Zheng et al. propose using the fish-eye model directly in the residual description (compare to eq. 65, where the pinhole model back-projection onto the image plane is used). Additionally, they correct for the camera vignetting in the measurement function itself, by modeling the lens attenuation through a gamma correction model and rectifying the image irradiance. Designing the filter to be consistent means using the first available estimates of the IMU prediction of the frame position in the update step. This has been shown to increase the filters estimation accuracy by Li et al. [85] and is used in the implementation of Zheng et al. [62] and Sun et al. [45]. As shown by Li et al. this change shows effectiveness primarily on long recordings.

In addition to these incremental improvements to the algorithm proposed by Zheng et al. an improvement might be a hybrid design using the feature position and the semi-direct approach, to combine the long-range feature correction capabilities with the accuracy of the patch-based implementation in proximity the original feature. The residual would be a construction of both the feature position u, v and the patch irradiance ξ. A different hybrid implementation fusing the semi-direct approach with the direct approach could allow for a reduction in CPU load - which direct implementations tend to suffer from - while maintaining their accuracy improvements. This design would increase the size of the patch points in the image the farther away the patch is from the feature origin - this is similar to the image pyramid design utilized by ROVIO [48].

Broadening the range of feature descriptors, such as Trifo-VIO [53] does (matching both feature points as well as lines) might allow for increasingly flexible filters. Being able to select the information extracted from the images based on the current scene would allow for an on-air switch between more robust versus more accurate descriptors.

9 Summary

The inventory-tracking quad-copter by the company D-Aria [18] relies on accurate position estimation to navigate in warehouses. For this reason, a highly efficient, accurate, and robust state estimator is a necessity. The goal of this thesis was to implement a stereo-photometric update calculation into the MSCKF pipeline by Sun et al. [45]. Based on the results of Zheng et al. [62] the design was presumed to increase filter-estimation accuracy.

The design of the MSCKF was explained and derived in significant detail in chapter 4. Alongside the survey of position estimators in chapter 3, the chapter not only presents an entry-point for the reader into VIO and the MSCKF, but also allows for a full picture of both algorithm expansions demonstrated in this thesis. The novelty factor of this thesis is the derivation and implementation of a patch-based stereo-photometric position estimator in chapter 6. Focus has also been put on a novel anchor-frame based feature modification in chapter 5, a precursor to the stereo-photometric MSCKF.

The MSCKF itself uses a sliding window of frames, tracking features over the course of this window. The feature movements inside this sliding window are used to correct the pose estimation of the IMU. The stereo-photometric MSCKF extracts pixel patches around the feature in both cameras and compares the original patch around the feature to a projection of the patch, based on the IMU estimation. The anchor-based approach is a precursor to the stereo-photometric algorithm and reduces the dimensionality of the feature position estimation, thereby reducing CPU load but also estimation accuracy. The anchor-based MSCKF shows a 2.2% reduction in CPU load but a surprising loss of accuracy (from 0.8% to 1.7% relative RMSE) and robustness (from 42% to 67% divergence rate).

We do not believe, that the slight reduction in CPU usage by the anchor-frame MSCKF is worth the substantial deterioration in pose estimation. The stereo-photometric MSCKF is not robust enough to be used in any practical setting. While the estimation accuracy for the convergent stretches of time is within range of the stereo and anchor-based MSCKF, the high divergence rate shows, that further improvements and tuning are vital for the stereo-photometric approach, to become a viable alternative for the stereo MSCKF. For the time being, due to its well-balanced CPU load vs. pose accuracy, the most promising VIO pose estimator for the company D-Aria remains the stereo-MSCKF.

Bibliography

[1] K. Systems, “Kiva Systems - Distribution Consultant Review,” [Online] Available: http://www.mwpvl.com/html/kiva_systems.html (Accessed 01.08.2019).

[2] Skydio, “Skydio, Inc.” [Online] Available: https://www.skydio.com (Accessed 01.08.2019).

[3] MiR, “MiR100 | Mobile Industrial Robots,” [Online] Available: https://www.mobile-industrial-robots.com/en/products/mir100 (Accessed 01.08.2019).

[4] OTTO, “OTTO Self Driving Vehicles for Automated Material Transport,” [Online] Available: https://ottomotors.com (Accessed 01.08.2019).

[5] Google, “ARCore,” [Online] Available: https://developers.google.com/ar/ (Accessed 01.08.2019).

[6] Microsoft, “Hololens,” [Online] Available: https://www.microsoft.com/en-us/hololens (Accessed 01.08.2019).

[7] S. Thrun, W. Burgard, and D. Fox, *Probabilistic robotics*. MIT Press, 2005.

[8] J. Marek and L. Stepanek, “Accuracy and availability of the satellite navigation system GPS,” in *15th Conference on Microwave Techniques COMITE 2010*. IEEE, 4 2010, pp. 121–124.

[9] J. Farrell, *Aided navigation : GPS with high rate sensors*. McGraw-Hill Education, 2008.

[10] S. Zhao, Y. Chen, H. Zhang, and J. A. Farrell, “Differential GPS aided Inertial Navigation: a Contemplative Realtime Approach,” *IFAC Proceedings Volumes*, vol. 47, 1 2014.

[11] iRobot, “iRobot,” [Online] Available: https://www.irobot.at (Accessed 01.08.2019).

[12] M. Ben-Ari and F. Mondada, “Robotic Motion and Odometry,” in *Elements of Robotics*. Cham: Springer International Publishing, 2018, pp. 63–93.

[13] Xiaomi, “Xiaomi Roborock Intelligent Robot Vacuum Cleaner,” [Online] Available: https://en.roborock.com (Accessed 01.08.2019).

[14] J. Choi, “Hybrid map-based SLAM using a velodyne laser scanner,” in *17th International IEEE Conference on Intelligent Transportation Systems, ITSC 2014, Qingdao, China, October 8-11, 2014*, 2014, pp. 3082–3087.

[15] C. Kerl, J. Sturm, and D. Cremers, “Dense visual SLAM for RGB-D cameras,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, 2013, pp. 2100–2106.

[16] X. Hai-Xia, Z. Wei, and Z. Jiang, “3d visual SLAM with a time-of-flight camera,” in *2015 IEEE Workshop on Signal Processing Systems, SiPS 2015, Hangzhou, China, October 14-16, 2015*, 2015, pp. 1–6.

[17] A. R. Vidal, H. Rebecq, T. Horstschaefer, and D. Scaramuzza, “Ultimate SLAM? Combining Events, Images, and IMU for Robust Visual SLAM in HDR and High-Speed Scenarios,” *IEEE Robotics and Automation Letters*, vol. 3, no. 2, 2018.

[18] D-Aria, “D-Aria Logistics,” [Online] Available: http://d-aria.at (Accessed 01.08.2019).

[19] X. Sun, F. Sun, B. Wang, J. Yin, X. Sheng, and Q. Xiao, “Robotic autonomous exploration SLAM using combination of Kinect and laser scanner,” in *2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*. IEEE, 11 2017, pp. 632–637.

[20] A. I. Mourikis and S. I. Roumeliotis, “A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation,” University of California, Riverside, Tech. Rep., 2007.

[21] M. D. Shuster, “A Survey of Attitude Representations,” *The Journal of the Astronautical Sciences*, vol. Vol. 41, no. No. 4, pp. 439–510, 1993.

[22] J. Solà, “Quaternion kinematics for the error-state Kalman filter,” Institute of Robotics and Industrial Informatics, Barcelona, Catallunya, Spain, Tech. Rep., 11 2017.

[23] M. Li, A. Mourikis, J. Farrell, and W. Ren, “Visual-Inertial Odometry on Resource-Constrained Systems,” University of California Riverside, Riverside, Tech. Rep., 2014.

[24] K. Yousif, A. Bab-Hadiashar, and R. Hoseinnezhad, “An Overview to Visual Odometry and Visual SLAM: Applications to Mobile Robotics,” *Intelligent Industrial Systems*, vol. 1, pp. 289–311, 2015.

[25] N. Trawny, A. I. Mourikis, S. I. Roumeliotis, A. E. Johnson, and J. Montgomery, “Vision-Aided Inertial Navigation for Pin-Point Landing using Observations of Mapped Landmarks,” University of Minnesota, Minnesota, Tech. Rep., 2007.

[26] A. W. Alhashimi, G. Nikolakopoulos, and T. Gustafsson, “Observation model for Monte Carlo Localization,” Luleå University of Technology, Luleå, Tech. Rep., 2014.

[27] S. Klaudt, A. Zlocki, and L. Eckstein, “A-priori map information and path planning for automated valet-parking,” in *IEEE Intelligent Vehicles Symposium, IV 2017, Los Angeles, CA, USA, June 11-14, 2017*, 2017, pp. 1770–1775.

[28] W. Burgard, A. Derr, D. Fox, and A. B. Cremers, “Integrating Global Position Estimation and Position Tracking for Mobile Robots: The Dynamic Markov Localization Approach,” Institute of Computer Science III, University of Bonn, Tech. Rep., 1998.

[29] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, “Monte Carlo Localization for Mobile Robots,” Carnegie Mellon University, Pittsburgh, Tech. Rep., 1999.

[30] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, “Monte Carlo Localization: Efficient Position Estimation for Mobile Robots,” School of Computer Science, Carnegie Mellon University, Pittsburgh, PA and Computer Science Department III, University of Bonn, Germany, Tech. Rep., 1999.

[31] A. Arleo, “Spatial learning and navigation in neuro mimetic systems : modeling the rat hippocampus / Aging, Vision and Navigation View project Convis: A convolutional vision modelling toolbox View project,” Universita degli Studi di Milano, Milano, Tech. Rep., 2000.

[32] M. Milford, G. Wyeth, and D. Prasser, “RatSLAM: a hippocampal model for simultaneous localization and mapping,” in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA ’04. 2004.* IEEE, 2004, pp. 403–408.

[33] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann, K. Lau, C. Oakley, M. Palatucci, V. Pratt, P. Stang, S. Strohband, C. Dupont, L.-E. Jendrossek, C. Koelen, C. Markey, C. Rummel, J. Van Niekerk, E. Jensen, P. Alessandrini, G. Bradski, B. Davies, S. Ettinger, A. Kaehler, A. Nefian, and P. Mahoney, “Stanley: The Robot that Won the DARPA Grand Challenge,” *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.

[34] G. Grisetti, C. Stachniss, and W. Burgard, “Improved techniques for grid mapping with rao-blackwellized particle filters,” *IEEE Trans. Robotics*, vol. 23, no. 1, pp. 34–46, 2007.

[35] R. Jamiruddin, A. O. Sari, J. Shabbir, and T. Anwer, “RGB-Depth SLAM review,” *CoRR*, vol. abs/1805.07696, 2018.

[36] R. Mur-Artal and J. D. Tardos, “ORB-SLAM2: an Open-Source SLAM System for Monocular, Stereo and RGB-D Cameras,” *CoRR*, vol. abs/1610.06475, 10 2016.

[37] J. Engel, J. Stückler, and D. Cremers, “Large-scale direct SLAM with stereo cameras,” in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, 2015, pp. 1935–1942.

[38] Waymo, “Waymo Safety Report,” Waymo LLC, Tech. Rep., 2017, [Online] Available: https://storage.googleapis.com/sdc-prod/v1/safety-report/waymo-safety-report-2017-10.pdf (Accessed 01.08.2019).

[39] A. Huletski, D. Kartashov, and K. Krinkin, “Evaluation of the modern visual SLAM methods,” in *2015 Artificial Intelligence and Natural Language and Information Extraction, Social*

Media and Web Search FRUCT Conference (AINL-ISMW FRUCT). IEEE, 11 2015, pp. 19–25.

[40] V. C. Usenko, J. Engel, J. Stückler, and D. Cremers, "Direct visual-inertial odometry with stereo cameras," in *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, 2016, pp. 1885–1892.

[41] M. Bloesch, J. Czarnowski, R. Clark, S. Leutenegger, and A. J. Davison, "Codeslam - learning a compact, optimisable representation for dense visual SLAM," in *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, 2018, pp. 2560–2568.

[42] A. Vakhitov and V. S. Lempitsky, "Learnable line segment descriptor for visual SLAM," *IEEE Access*, vol. 7, pp. 39 923–39 934, 2019.

[43] M. Brossard, A. Barrau, and S. Bonnabel, "AI-IMU dead-reckoning," *CoRR*, vol. abs/1904.06064, 2019.

[44] M. Jaimez, J. G. Monroy, and J. Gonzalez-Jimenez, "Planar odometry from a radial laser scanner. A range flow-based approach," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2016, pp. 4479–4485.

[45] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar, "Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight," *IEEE Robotics and Automation Letters*, vol. 3, pp. 965–972, 4 2018.

[46] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[47] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback," *I. J. Robotics Res.*, vol. 36, no. 10, pp. 1053–1072, 2017.

[48] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct EKF-based approach," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 9 2015, pp. 298–304.

[49] T. Qin, P. Li, and S. Shen, "VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 8 2018.

[50] C. Forster, M. Pizzoli, and D. Scaramuzza, "SVO: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5 2014, pp. 15–22.

[51] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, “SVO: Semidirect Visual Odometry for Monocular and Multicamera Systems,” *IEEE Transactions on Robotics*, vol. 33, no. 2, pp. 249–265, 4 2017.

[52] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, “A robust and modular multi-sensor fusion approach applied to MAV navigation,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, Tokyo, Japan, November 3-7, 2013*, 2013, pp. 3923–3929.

[53] F. Zheng, G. Tsai, Z. Zhang, S. Liu, C.-C. Chu, and H. Hu, “Trifo-VIO: Robust and Efficient Stereo Visual Inertial Odometry Using Points and Lines,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 10 2018, pp. 3686–3693.

[54] G. Falco, M. Pini, and G. Marucco, “Loose and Tight GNSS/INS Integrations: Comparison of Performance Assessed in Real Urban Scenarios,” *Sensors (Basel, Switzerland)*, vol. 17, no. 2, 1 2017.

[55] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, “Observability-based Rules for Designing Consistent EKF SLAM Estimators,” *The International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, 4 2010.

[56] E. A. Wan and R. v. d. Merwe, “The Unscented Kalman Filter,” in *Kalman Filtering and Neural Networks*, S. Haykin, Ed. Beaverton: Oregon Graduate Institute of Science and Technology, 2001, ch. 7.

[57] W. Förstner and B. P. Wrobel, “Bundle Adjustment,” in *Photogrammetric Computer Vision*, 11th ed. Springer, Cham, 2016, ch. 15, pp. 643–725.

[58] P. Bernal-Polo and H. M. Barberá, “Kalman filtering for attitude estimation with quaternions and concepts from manifold theory,” *Sensors*, vol. 19, no. 1, p. 149, 2019.

[59] T. Qin, J. Pan, S. Cao, and S. Shen, “A general optimization-based framework for local odometry estimation with multiple sensors,” *CoRR*, vol. abs/1901.03638, 2019.

[60] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part I*, 2006, pp. 430–443.

[61] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[62] X. Zheng, Z. Moratto, M. Li, and A. I. Mourikis, “Photometric patch-based visual-inertial odometry,” in *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, 2017, pp. 3264–3271.

[63] A. H. Jazwinski, *Stochastic processes and filtering theory*, 1st ed. Dover Publications, 1970.

[64] J. H. Jung, S. Heo, and C. G. Park, “Patch-based stereo direct visual odometry robust to illumination changes,” *International Journal of Control, Automation and Systems*, vol. 17, no. 3, pp. 743–751, Mar 2019.

[65] P. Geneva, J. Maley, and G. Huang, “An efficient schmidt-ekf for 3d visual-inertial SLAM,” *CoRR*, vol. abs/1903.08636, 2019.

[66] J. A. Delmerico and D. Scaramuzza, “A benchmark comparison of monocular visual-inertial odometry algorithms for flying robots,” in *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*, 2018, pp. 2502–2509.

[67] M. Höll and V. Lepetit, “Monocular LSD-SLAM Integration within AR System,” TU Graz and Université Bordeaux, Tech. Rep., 2 2017.

[68] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The EuRoC micro aerial vehicle datasets,” *The International Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 9 2016.

[69] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, “Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle,” *J. Field Robotics*, vol. 33, no. 4, pp. 431–450, 2016.

[70] M. Hutter and R. Siegwart, “Robust Visual Inertial Odometry framework,” [Online] Available: https://github.com/ethz-asl/rovio (Accessed 01.08.2019).

[71] K. Robotics, “KumarRobotics/msckf_vio: Robust Stereo Visual Inertial Odometry for Fast Autonomous Flight,” [Online] Available: https://github.com/KumarRobotics/msckf_vio (Accessed 01.08.2019).

[72] M. Shelley, “Monocular Visual Inertial Odometry on a Mobile Device,” Master’s thesis, TU München, 2014.

[73] P. E. Gill and W. Murray, “Algorithms for the Solution of the Nonlinear Least-Squares Problem,” *SIAM Journal on Numerical Analysis*, vol. 15, no. 5, pp. 977–992, 1978.

[74] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers, “The TUM VI Benchmark for Evaluating Visual-Inertial Odometry,” in *iros*, 10 2018.

[75] D. Poole, *Linear algebra : a modern introduction*, 4th ed. Peterborough: Cengage Learning, 2014.

[76] B. D. Lucas and T. Kanade, “An iterative image registration technique with an application to stereo vision (ijcai),” in *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI ’81)*, April 1981, pp. 674–679.

[77] B. Kitt, A. Geiger, and H. Lategahn, “Visual odometry based on stereo image sequences with RANSAC-based outlier rejection scheme,” in *2010 IEEE Intelligent Vehicles Symposium*. IEEE, 6 2010, pp. 486–492.

[78] M. A. Fischler and R. C. Bolles, “Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography,” *Commun. ACM*, vol. 24, no. 6, pp. 381–395, Jun. 1981.

[79] N. Trawny and S. I. Roumeliotis, “Indirect Kalman Filter for 3D Attitude Estimation,” University of Minnesota, Minnesota, Tech. Rep., 2005.

[80] J. A. M. Saif, M. H. Hammad, and I. A. A. Alqubati, “Gradient Based Image Edge Detection,” *IACSIT International Journal of Engineering and Technology*, vol. 8, no. 3, pp. 153–156, 2016.

[81] G. Chang, “Robust Kalman filtering based on Mahalanobis distance as outlier judging criterion,” *Journal of Geodesy*, vol. 88, no. 4, pp. 391–401, 4 2014.

[82] A. Ugoni and B. Walker, “The Chi square test: an introduction,” *COMSIG review / COMSIG, Chiropractors and Osteopaths Musculo-Skeletal Interest Group*, vol. 4, pp. 61–64, 1995.

[83] J. Sturm, N. Engelhard, F. Endres, W. Burgard, and D. Cremers, “A benchmark for the evaluation of RGB-D SLAM systems,” in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2012, Vilamoura, Algarve, Portugal, October 7-12, 2012*, 2012, pp. 573–580.

[84] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, “Vision meets robotics: The KITTI dataset,” *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 9 2013.

[85] M. Li and A. I. Mourikis, “High-precision, consistent ekf-based visual-inertial odometry,” *I. J. Robotics Res.*, vol. 32, no. 6, pp. 690–711, 2013.

[86] C. Grinstead and J. L. Snell, *Introduction to Probability*, 2nd ed. Hanover, VM: Dartmouth College, 1999.

[87] J. Rice, *Mathematical Statistics and Data Analysis*, 1st ed. Belmont, CA: Brooks/Cole Cenage Learning, 2007.

[88] P. Zarchan and H. Musoff, *Fundamentals of Kalman filtering : a practical approach*, 1st ed. American Institute of Aeronautics and Astronautics, 2000.

[89] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*, 2nd ed. New York: Cambridge University Press, 2004.

[90] D. A. Forsyth and J. Ponce, *Computer Vision - A Modern Approach, Second Edition*. Pitman, 2012.

[91] P. Furgale, J. Rehder, and R. Siegwart, “Unified temporal and spatial calibration for multi-sensor systems,” in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 11 2013, pp. 1280–1286.

[92] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, “Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes,” in *2016 IEEE International Conference on Robotics and Automation, ICRA 2016, Stockholm, Sweden, May 16-21, 2016*, 2016, pp. 4304–4311.

[93] F. Devernay and O. Faugeras, “Straight lines have to be straight,” *Machine Vision and Applications*, vol. 13, no. 1, pp. 14–24, 8 2001.

[94] G. K. Batchelor, *An introduction to fluid dynamics*. Cambridge University Press, 1999.

[95] W. S. F. Iv, J. H. Wall, and D. M. Bevly, “Characterization of Various IMU Error Sources and the Effect on Navigation Performance,” in *dump*, Auburn, 2005.

[96] L. Landau and E. Lifshitz, *Mechanics*, 1st ed. Pergamon Press, 1980.

[97] John Voight, “Quaternion algebras,” in *Quaternion algebras*, 2nd ed. Hannover: Dartmouth College, 2019, ch. 1, pp. 3–6.

[98] B. Eater, “Visualizing quaternions, an explorable video series,” [Online] Available: https://eater.net/quaternions (Accessed 01.08.2019).

List of Figures

List of Tables

List of Abbreviations

1D one dimensional

2D two dimensional

3D three dimensional

BRIEF Binary Robust Independent Elementary Features

CPU Central processing unit

e.g. for example

EKF Extended Kalman Filter

ESKF Error-State Kalman Filter

EuRoC European Robitics Challenges

eq equation

FAST Features from Accelerated Segment Test

GPS Global Positioning System

IMU Inertial Measurement Unit

KF Kalman Filter

LSD Large-Scale Direct

m meter

MAV Micro-Aerial Vehicle

MiR Mobile industrial Robot

MSF Modular sensor fusion

MSCKF Multi-State Constraint Kalman Filter

nr. number

OKVIS Open Keyframe-based Visual-Inertial SLAM

ORB Oriented FAST and Rotated BRIEF

RAM Random Access Memory

RANSAC Random sample consensus

rel. relative

RGB Red - Green - Blue

RMSE Root Mean Square Error

ROVIO Robust Visual Inertial Odometry

s second

SIFT Scale-invariant feature transform

SLAM Simultaneous Localization and Mapping

SVO Semi-direct Visual Odometry

TUM Technische Universität München

VINS Visual-Inertial Systems

VIO Visual Inertial Odometry

VO Visual Odometry

vs. versus

A Extended Kalman Filter

This chapter describes how Kalman Filters (KF) generally work - without going into detail on the mathematical derivation of the concept. We also define the Kalman Filter notation as used in this thesis.

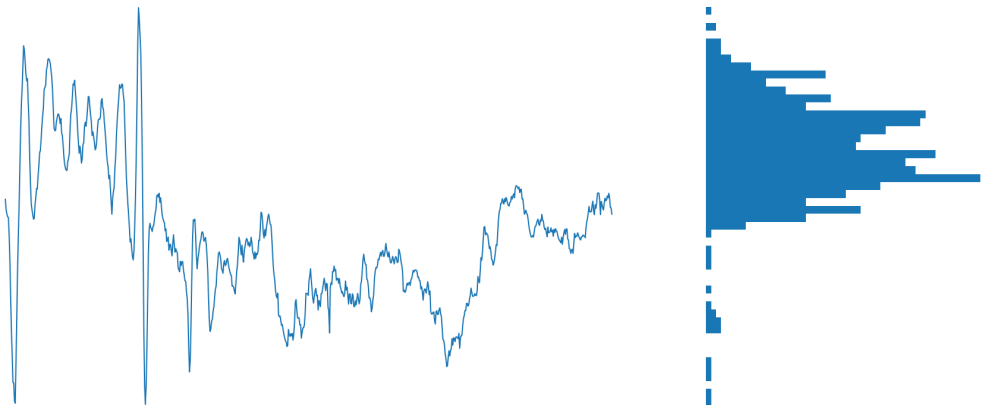

Figure 35: IMU measurements condensed into a histogram resembling a Gaussian function

A.1 Probabilistic Estimation

This section is based on the books regarding probability theory by Grinstead and Snell [86] as well as Rice [87]. An explanation of the Kalman Filter relies heavily on the concept of Gaussian distribution, which is defined through a mean μ and a variance σ^2. The lower the value of the variance, the higher the probability is, that a randomly sampled variable x from this distribution is near μ. The noise measurement of a still IMU - as discussed appendix C - resembles a Gaussian distribution, see figure 35.

This concept can be expanded to multiple dimensions. Figure 36 shows two Gaussian curves combined into a two-dimensional Gaussian. In multiple dimensions, the mean scalar μ is instead a vector $\boldsymbol{\mu}$ and the variance turns into the covariance matrix $\boldsymbol{\Sigma}$. The diagonal elements of this covariance matrix are the variances of the elements σ_x^2, σ_y^2. The off-center values represent the *joint variability* of the respective variables - how much the variable x influences the variable y.

$$\mathbf{\Sigma} = \begin{bmatrix} \sigma_x^2 & \sigma_x y \sigma_y \\ \sigma_y x & \sigma_y^2 \end{bmatrix}$$

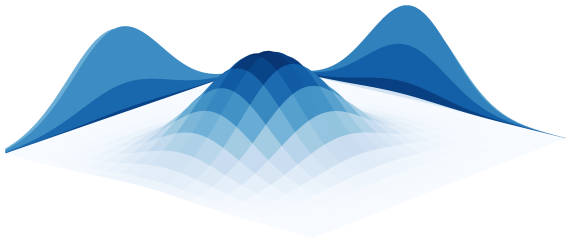

Figure 36: two-dimensional Gaussian function

A.2 Kalman Filter

The following section summarizes the work of Thrun et al. [7] and Zarchan et al. [88]. The Kalman Filter is used regularly for position estimation in mobile robotics [7, 88]. It uses one input to predict its change of state (typically through a dynamic model of the system) and then corrects this prediction with some other measurement. The filter predicts both the mean and the covariance of the state. A simple example is a robot moving in one dimension, estimating its current pose through an IMU and wheel-odometry. A Kalman Filter is designed to make a prediction *independent* of any information other than the one in its current *state* and using only the current *prediction input* and its current *measurements* (note that the prediction input is commonly called the *control* input). This independence of any previous information is called a Markovian assumption. In the robot example, the state could be a vector constructed of the position- and the velocity estimation. The filter not only estimates the next state vector (the $\boldsymbol{\mu}$) from its inputs but also estimates the covariance matrix $\mathbf{\Sigma}$ - the certainty of its estimation.

Generally, we describe the relationship between the error state and the true state as

$$\text{true state} = \text{nominal state} + \text{error}$$

where the nominal state describes the estimate of the state. The notation is defined in tables 7 and 8.

The process noise represents the uncertainty of the underlying model in the prediction, the sensor noise is the intrinsic uncertainty in the correction step. We will now look at these two steps, the prediction and the correction including their mathematical formulation. The steps will then be expanded into the Extended Kalman Filter.

Table 7: State types

true state	$\mathbf{x}$
nominal state	$\hat{\mathbf{x}}$
error	$\tilde{\mathbf{x}}$

Table 8: Covariance and noise

covariance matrix	$\mathbf{P}$
process noise	$\mathbf{Q}$
sensor noise	$\mathbf{R}$

Using a dynamic model of the underlying system (eg. the model of the one-dimensional robot), the prediction input is used to predict the robot's next state. Note that this input can be from any source, be it a sensor input or a control input. In the robot example, we choose the IMU measurements to be used in the prediction, as it is desirable to have a higher rate of predictions than corrections. The IMU acceleration information contains noise and biases, so integrating the data adds errors to the current state. An approximation of this error is expressed in the process noise $\mathbf{Q}$. During each prediction, the state estimation $\hat{\mathbf{x}}$ is changed, along with the covariance around this state. The matrix $\mathbf{F}$ is a linear predictor of the next state based on the current state, the vector $\mathbf{u}$ is the prediction input, which is mapped to a change in the state prediction through the matrix $\mathbf{B}$. The current timestep is represented by the index k. Note that all of these matrices can be different in each time-step.

$$\begin{aligned}\hat{x}_{k+1} &= \mathbf{F}_k\hat{\mathbf{x}}_k + \mathbf{B}_k\mathbf{u} \\ \mathbf{P}_{k+1} &= \mathbf{F}_k\mathbf{P_k}\mathbf{F}_k^\top + \mathbf{Q}_k\end{aligned}$$

During the update step, the measurement information in the vector $\mathbf{z}$ is used. The true measurement is compared with the expected measurements based on the predicted state $\hat{\mathbf{x}}$ using the linear sensor model matrix $\mathbf{H}$. This results in what we call the residual. Using an updated covariance matrix and the two noise parameters $\mathbf{Q}$ and $\mathbf{R}$, the variance terms in the covariance matrix are minimized through the Kalman Gain. Using this Gain and the residual, the state estimation is updated alongside the covariance matrix.

$$\begin{aligned}\mathbf{r}_k &= \mathbf{z}_k - \mathbf{H}_k\hat{\mathbf{x}}_k \\ \mathbf{K}_k &= \mathbf{P}_k\mathbf{H}^\top\left(\mathbf{H}_k\mathbf{P}_k\mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \\ \hat{\mathbf{x}}_{k+1} &= \mathbf{K}_k\mathbf{r}_k \\ \mathbf{P}_k &= \left(\mathbf{I} - \mathbf{K}_k\mathbf{H}_k\right)\mathbf{P}_k\end{aligned}$$

A.3 Nonlinear Estimation Function

The Kalman Filter model is limited by its inherently linear approach to prediction and correction through the $\mathbf{F}$, $\mathbf{B}$ and $\mathbf{H}$ matrices. The Extended Kalman Filter [7] allows for *nonlinear* propagation of the state and linearizes these functions to properly propagate the covariance matrices, which need to be linearly transformed to preserve their Gaussian properties. This is done through a first-order Taylor approximation of the functions yielding a Jacobian matrix. For completeness, here is the full EKF propagation step:

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

and the update step:

$$\begin{aligned}\mathbf{r}_k &= \mathbf{z}_k - h(\hat{\mathbf{x}}_k) \\ \mathbf{K}_k &= \mathbf{P}_k \mathbf{H}^\top \left(\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\top + \mathbf{R}_k\right)^{-1} \\ \hat{\mathbf{x}}_{k+1} &= \mathbf{K}_k \mathbf{r}_k \\ \mathbf{P}_k &= \left(\mathbf{I} - \mathbf{K}_k \mathbf{H}_k\right) \mathbf{P}_k\end{aligned}$$

The function $f(\cdot)$ is the prediction function operating on the current state estimation and the prediction input. $h(\cdot)$ remaps the current state estimation into the expected measurements. The respective linearized Jacobians are $\mathbf{F}$ and $\mathbf{H}$. A note regarding error accumulation: If a state is observable, such as the rotation about the x- and y-axis of the IMU, then the error is bounded. If the error is not observable, such as the z-axis rotation, then the error can grow without bounds.

Linearizing the residual to receive the Jacobian $\mathbf{H}$ is done as follows:

$$\mathbf{r} = \mathbf{z} - \hat{\mathbf{z}} + \mathbf{n}$$

using first-order Taylor approximation results to:

$$\mathbf{r} = h(\mathbf{x}) - h(\hat{\mathbf{x}}) + \mathbf{n} \tag{74}$$

$$\mathbf{r} = h(\hat{\mathbf{x}} + \tilde{\mathbf{x}}) - h(\hat{x}) + \mathbf{n}$$

$$\mathbf{r} \simeq h(\hat{\mathbf{x}}) + \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} - h(\hat{\mathbf{x}}) + n$$

$$\mathbf{r} \simeq \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \mathbf{n} \tag{75}$$

where H_x is the Jacobian of the function $h()$ with respect to $\mathbf{x}$.

While the Kalman Filter operates optimally, as the propagation functions are linear, the Extended Kalman Filter loses some accuracy through its linearization. Various implementations try to minimize the linearization error. This includes the iterated EKF and the Unscented Kalman Filter. The strategy commonly used in VIO systems is the Error-state Kalman Filter, which will be explained in the following chapter.

B Camera Models

This section will first introduce a common camera description, the pinhole model. The cameras used in this thesis will be modeled following this approach. Further, this section will give a summary on stereo-camera theory, as they are a common tool in VIOs.

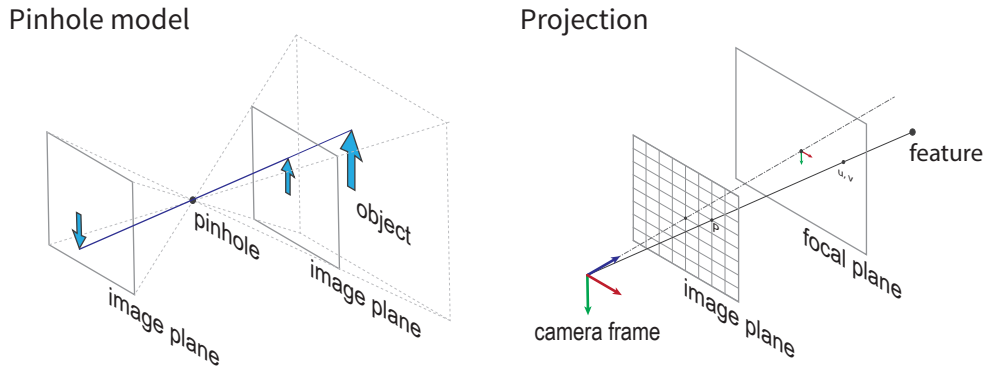

Figure 37: The pinhole model and a schematic of the projection from the camera frame to the image plane

The pinhole camera model derives its name from the pinhole camera, which passes the light reflecting from an object through a small hole to create an inverted image on the other side [89], see figure 37. As can be seen from this figure, the image plane can be assumed to be in front *or* behind the camera center, which is the point where the image light passes through. Assuming the cameras focal length $f = 1$, we can project a point represented in the camera coordinate frame by $\begin{bmatrix} X & Y & Z \end{bmatrix}^\top$ in $\mathbb{R}^3$ onto this image plane at $f = 1$ by calculating

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{X}{Z} \\ \frac{Y}{Z} \\ \frac{Z}{Z} \end{bmatrix}$$

with $[u\ v]^\top$ the coordinates of the point in $\mathbb{R}^2$.

To u, v representation is independent of the cameras focal length and principal point. To calculate the *pixel* coordinates of the point $\mathbf{P}$, we use the true focal length of the camera f as well as the position of the principal point p.

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{P} = \mathbf{K} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$$

where $\mathbf{K}$ is described as the *intrinsic* camera parameter matrix [90] - it maps pixel coordinates to points in the camera frame.

A stereo-camera setup needs additional *extrinsic* camera parameters to express the change in pose between the first and the second camera, see figure 38.

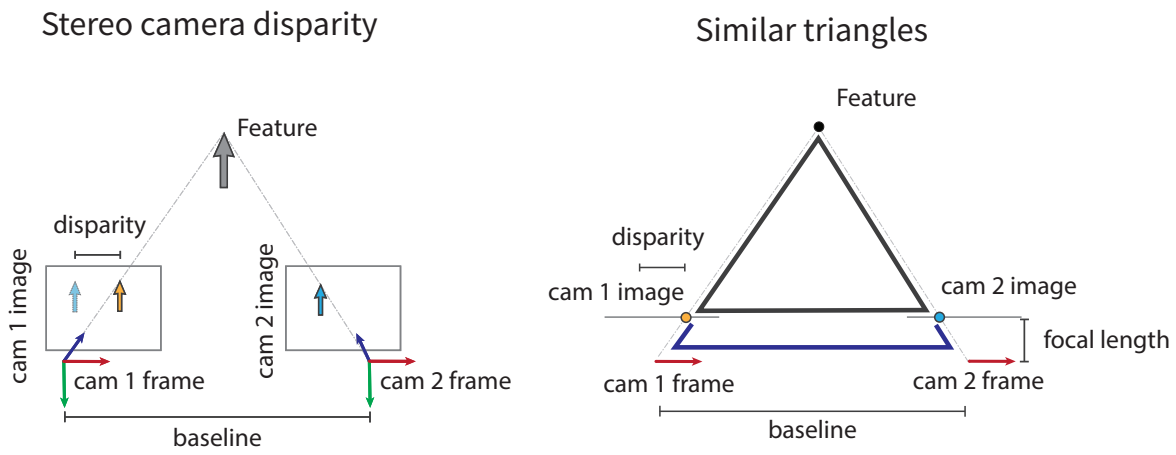

Figure 38: Stereo-cameras and the disparity between them with a visualization of the triangles used to calculate the depth of the feature in space

An advantage that stereo setups yield is the availability of true geometric information regarding the camera measurements. The determined relation between the two cameras - often described as the stereo baseline - allows for accurate triangulation of features with *accurate scale* information. This information is calculated using the *disparity* of the projection of the same feature point in both images of the stereo-camera setup. This disparity is inversely proportional to the depth of the feature point in the projected image, see figure 38.

Both the intrinsic and extrinsic parameters can be extracted either directly from the camera setup (measured) or by using calibration tools such as Kalibr [91, 92] which conveniently calibrates additional parameters such as camera distortion parameters (depending on the distortion model, see [93]) and the relative pose to an IMU.

C Inertial Measurement Unit

The inertial measurement unit (IMU) sensor returns the acceleration and angular rate of the body it is attached to. Measurement values are subject to noise and biases; this section will summarize IMU modeling and implementation.

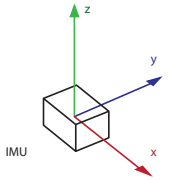

Figure 39: Axis of an Inertial Measurement Unit

To achieve a pose estimation from an IMU alone, the accelerometer values must be integrated twice. This inevitably increases any noise or biases, making this process unstable for longer stretches of time. To minimize these effects, an accurate estimation of the biases and the noise is necessary. The accelerometer in the IMU senses the gravitational pull, making rotations around the x- and y-axis observable (see axis in figure 39). Global rotation about the z-axis is not observable. The gyroscope senses the earth's rotation and if this component is not modeled in correctly, integration may be subject to increased drift [22]. Noise in recorded measurements of both the accelerometer and the gyroscope are mainly of mechanical or thermal nature, as are the experienced biases [94]. A simple model of the IMU gyroscope from Flenniken et al. [95] is,

$$g = r + c_g + b_g + n_g$$

where r is the true rotation rate, c_g is some constant offset, b_g is a walking bias and n_g is the sensor noise of the gyroscope. A similar model exists for the accelerometer

$$a = \ddot{x} + c_a + b_a + n_a.$$

The noise parameters n_g, n_a are assumed to be normally distributed with zero mean, and the bias parameters b_g, b_a are modeled through a first-order Markov process. The bias and the constant offset are merged into a single parameter in most VIO specific works.

D Quaternions

This chapter presents the concept of quaternions and their general notation. After explaining the advantages, which imply the reasons why quaternions are used throughout this work, this chapter will form a more intuitive understanding of quaternions for the reader. In parallel, quaternion mathematics and their relation to rotation matrices as well as their small-angle approximation are shown. The last concept is crucial to significant simplifications in the coming chapters.

There are multiple concepts for three-dimensional rotation representation. Among the most common ones are euler angles, rotation matrices, axis-angle representations and quaternions [96]. Expressing rotations in quaternion multiplication is the most efficient description concerning the number of calculations [21]. Further, this expression is minimal when wanting to avoid gimbal-lock and also is more numerically stable than other approaches, as Shuster et al. [21] show.

D.1 Intuitive Quaternions

Rotations in a plane can generally be expressed by a single angle, assuming a predefined orientation - a counter-clockwise a.k.a. positive rotation. Besides the common x-y coordinate frames for 2d representation, complex numbers provide another, with the imaginary axis Im being normal to the real axis Re.

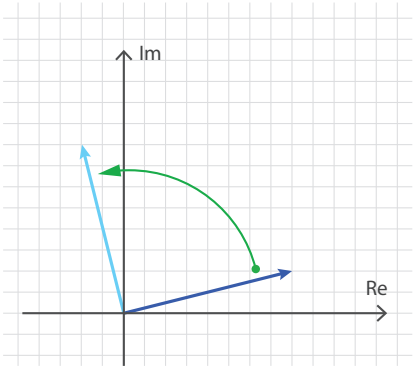

Figure 40: Multiplying a complex number with *i*

When multiplying any complex number p, eg. $p = 4 + 1i$, by the complex number i, we can observe a rotation of the resulting point in the complex plane around the origin, see figure 40. An arbitrary rotation around the origin can be achieved analogously to how they are commonly known through rotation matrices. Rotating the example vector $[x\ y]^\top$ around the angle θ for example can be written as:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos\theta & -sin\theta \\ sin\theta & cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}.$$

For the complex number plane, instead of using this matrix representation, the multiplication of two complex numbers is enough to result in an unambiguous rotation.

$$pq = (a + bi)(\cos\theta + i\sin\theta) \tag{76}$$

$$a' + b'i = a\cos\theta - b\sin\theta + (a\sin\theta + b\cos\theta)i \tag{77}$$

Both representations are equivalent in their resulting rotation.

A similar concept of rotation in the two-dimensional plane can be applied into four dimensions and has been famously done by Rodrigues in 1840 and Hamilton in 1843 [97]. A quaternion is built up of four components: one real part and three imaginary parts called i, j and k. A quaternion q can be expressed as:

$$q = \mathrm{a} + \mathrm{b}i + \mathrm{c}j + \mathrm{d}k \tag{78}$$

where a, b, c and d are real numbers scaling the respective components. As the following shows, it is not possible to construct a *three* dimensional quaternion to represent rotations in three dimensions.

If the two-dimensional Complex numbers are extended by a second irrational number j, the following number is a valid general description:

$$t = a + \mathrm{b}i + \mathrm{c}j$$

multiplying this t by a different complex number, it shows that

$$ti = (a + \mathrm{b}i + \mathrm{c}j)i = -\mathrm{b} + ai + \mathrm{c}ij$$

where ij has no valid representation in this three-dimensional space. Therefore, the additional dimension for $k = ij$ is included to receive the four-dimensional quaternion description of eq. 78.

If the same rotation operation as in eq. 76 is applied to this new quaternion - to achieve a desired rotation in the plane spanned by the real axis and i (as this operation does in the case of the Complex numbers):

$$
\begin{aligned}
i\mathbf{p} &= i(\mathrm{a} + \mathrm{b}i + \mathrm{c}j + \mathrm{d}k) \\
i\mathbf{p} &= \mathrm{a}i - \mathrm{b} + \mathrm{c}ij + \mathrm{d}ik \\
i\mathbf{p} &= \mathrm{a}i - \mathrm{b} + \mathrm{c}k - \mathrm{d}j
\end{aligned}
$$

this operation rotates the *real* - i plane by the expected 90° but also the j - k plane by 90°. This rotation is not preventable - but by defining the **anti-commutativity property**,

$$ij = -ji$$

the right-multiplication of i now *inverts* the second rotation. Using this property, it is possible to negate the 'unwanted' rotation in the j - k plane through right-multiplying i.

$$i\mathbf{p}i = i(\mathrm{a} + \mathrm{b}i + \mathrm{c}j + \mathrm{d}k)i \tag{79}$$

$$i\mathbf{p}i = -\mathrm{a} - \mathrm{b}i + \mathrm{c}j + \mathrm{d}k \tag{80}$$

This shows, that the operation always preserves - and doubles - the rotation around the real axis and negates the unavoidable second rotation. To uphold the rotations around the *other* imaginary plane and leave the real axis untouched, the quaternion $\mathbf{p}$ is multiplied by the negated rotation quaternion, $-i$ in this case:

$$i\mathbf{p}(-i) = \mathrm{a} + \mathrm{b}i - \mathrm{c}j - \mathrm{d}k.$$

Again, this operation, the rotation of $\mathbf{p}$ now doubles the rotation along the affected imaginary axis i and leaves the orientation of the real axis, untouched.

This concludes in the general quaternion rotation description:

$$\mathbf{q} \otimes \mathbf{p} \otimes \mathbf{q}^{-1} \tag{81}$$

where the operator $\otimes$ is the general case of the quaternion multiplication. It is the multidimensional case of the simple multiplication by i done in the previous calculations and will be detailed in the following chapter. From eq. 76 we can see, that the angle of rotation encoded in the rotation quaternion $\mathbf{q}$ in eq. 81 must be *half* the desired angle, as this rotation is committed *twice* to the quaternion $\mathbf{p}$.

As seen in figure 41, an easy way to visualize the rotation a quaternion describes is by using the imaginary values $b{\cdot}i$, $c{\cdot}j$ and $d{\cdot}k$ as a description of a vector in 3D space (cyan colored line). The real part a of the quaternion then describes the rotation *around this vector*, as visualized by the black arrow. The remaining part of this chapter will explain why this visual representation is valid.

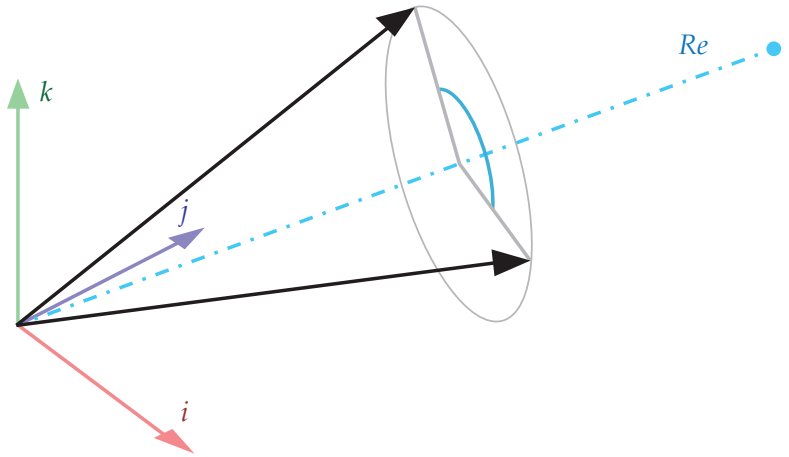

Figure 41: Quaternion rotation represented in axis-angle form

As this chapter has layed out, a quaternion describes the general rotation of a rigid body in four dimensions. The projection of this rotation around the axis i, j and k into three dimensions allows for a description of rotation in three dimensions. This projection is key to understand the visually accessible description of quaternion rotations of figure 41. To achieve a pure three-dimensional rotation, the real axis dimension is constrained by the anti-commutativity of the quaternion. This is used to rotate the hypercube back into place regarding all rotations along the real axis through multiplication of $\mathbf{q}^{-1}$. The remaining rotations around the ij, jk and ki planes are *not* inverted but doubled (compare to eq. 79). This projection from four to three dimensions is best visualized by the quaternion visualization tool by Grand Sanderson [98], in which stereographic projections are used as an aid to understand the concept.

D.2 Quaternion Mathematics

After some brief discussion regarding quaternion notation and structure, this section will present the mathematics used in the context of quaternions used in this thesis.

This previous section has introduced

$$i^2 = j^2 = k^2 = ijk = -1$$

and the general quaternion description

$$q \triangleq \mathrm{a} + \mathrm{b}i + \mathrm{c}j + \mathrm{d}k$$

which can also be seen as

$$\mathbf{q} \triangleq \begin{bmatrix} \mathbf{q}_v \\ q_w \end{bmatrix} = \begin{bmatrix} q_x \\ q_y \\ q_z \\ q_w \end{bmatrix}.$$

Here the real element q_w is located underneath the vector of the imaginary numbers $\mathbf{q}_v$. Note, that we use one of two quaternion types called *ijk* compared to the *hamilton* notation. A comparison of the two styles can be found in the work of Sola et al. [22]. Although there are no conceptional changes between these styles, some calculations are structured differently. The reason for choosing the ijk formulation is that it is commonly used in VIO implementations generally and in the MSCKF implementations in particular.

With quaternion notation defined, the main quaternion properties are now briefly presented. For more details regarding quaternion mathematics, again we refer to Sola et al. [22].

The quaternion product is defined by the $\otimes$ operator and results in

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w q_x + p_x q_w + p_y q_z - p_z q_y \\ p_w q_y - p_x q_z + p_y q_w + p_z q_x \\ p_w q_z + p_x q_y - p_y q_x + p_z q_w \\ p_w q_w - p_x q_x - p_y q_y - p_z q_z \end{bmatrix}$$

and is generally **non-commutative**, as is visible from the equivalent quaternion product definition

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w \mathbf{q}_v + q_w \mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \\ p_w q_w - \mathbf{p}_v^\top \mathbf{q}_v \end{bmatrix}$$

where the non-commutative cross product is used. The product is **associative**

$$(\mathbf{p} \otimes \mathbf{q}) \otimes \mathbf{r} = \mathbf{p} \otimes (\mathbf{q} \otimes \mathbf{r}).$$

The quaternion multiplication can be alternatively expressed by a matrix-vector multiplication, where, as the product is non-commutative, two different matrix structures exist.

$$\mathrm{q}_1 \otimes \mathrm{q}_2 = \ [\mathrm{q}_1]_L \mathrm{q}_2 \quad \text{and} \quad \mathrm{q}_1 \otimes \mathrm{q}_2 = [\mathrm{q}_2]_R \, \mathrm{q}_1 \ .$$

The left multiplication matrix $[\mathrm{q}_2]_L$ and the right multiplication matrix $[\mathrm{q}_2]_R$ are constructed as follows

$$[\mathbf{q}]_L = \begin{bmatrix} q_w & -q_z & q_y & q_x \\ q_z & q_w & -q_x & q_y \\ -q_y & q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{bmatrix}, \quad [\mathbf{q}]_R = \begin{bmatrix} q_w & q_z & -q_y & q_x \\ -q_z & q_w & q_x & q_y \\ q_y & -q_x & q_w & q_z \\ -q_x & -q_y & -q_z & q_w \end{bmatrix} \tag{82}$$

which may be written as

$$[\mathbf{q}]_L = q_w \mathbf{I} + \begin{bmatrix} [\mathbf{q}_v]_\times & \mathbf{q}_v \\ -\mathbf{q}_v^\top & 0 \end{bmatrix}, \quad [\mathbf{q}]_R = q_w \mathbf{I} + \begin{bmatrix} -[\mathbf{q}_v]_\times & \mathbf{q}_v \\ -\mathbf{q}_v^\top & 0 \end{bmatrix}.$$

Here, the cross product matrix $[\mathbf{a}]_\times$ is defined as

$$[\mathbf{a}]_\times \triangleq \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}$$

The identity quaternion is

$$\mathbf{q_1} = 1 = \begin{bmatrix} \mathbf{0}_v \\ 1 \end{bmatrix}$$

The conjugate quaternion is defined as

$$\mathbf{q}^* \triangleq \begin{bmatrix} -\mathbf{q}_v \\ q_w \end{bmatrix}$$

and the inverse quaternion as

$$\mathrm{q}^{-1} = \mathrm{q}^* / \|\mathrm{q}\|^2$$

where

$$\mathbf{q} \otimes \mathbf{q}^{-1} = \mathbf{q}^{-1} \otimes \mathbf{q} = \mathbf{q_1}$$

If the quaternion is a *unit* quaternion

$$\|\mathrm{q}\|^2 = \mathbf{q_1}$$

then

$$q^{-1} = q^*.$$

As rotation quaternions are unit quaternions, this property results in the previously mentioned

rotation eq. 81, simplified to using the conjugate of the rotation quaternion $\mathbf{q}$

$$\mathbf{q} \otimes \mathbf{p} \otimes \mathbf{q}^{-1} = \mathbf{q} \otimes \mathbf{p} \otimes \mathbf{q}^{*}$$

where the vector $\mathbf{p}$ is a pure quaternion, with no real entry. The resulting vector in 3D space is described by the i, j and k components.

To define a rotation matrix $C(\cdot)$ resulting from any rotation quaternion we can finally use the following relation from [22]

$$\begin{aligned} \mathbf{R}\mathbf{p} &= \mathbf{q} \otimes \mathbf{p} \otimes \mathbf{q}^{*} \\ \mathbf{R} &= C(\mathbf{q}) = (q_w^2 - q_v^\top q_v)\mathbf{I} + 2q_v q_v^\top + 2q_w[q_v]_\times \end{aligned}$$

D.2.1 Small-Angle Rotation

As very often, the considered rotations are very small, it is feasible to approximate these small-angle quaternions through linearization. This further allows the number of components needed for its representation to be reduced to three.

Any unit quaternion can be described by the relationship

$$\mathbf{q} = \begin{bmatrix} \mathrm{u} \sin \frac{1}{2}\theta \\ \cos \frac{1}{2}\theta \end{bmatrix}$$

Should the angle of q be small, we can use the relation of eq. 2 to simplify $C(q)$ using a first-order taylor approximation, we can calculate

$$\begin{aligned} C(\mathbf{q}) \simeq C(\begin{bmatrix} \frac{1}{2}\theta \\ 1 \end{bmatrix}) = \\ = (1^2 + \frac{1}{2}\theta\frac{1}{2}\theta)\mathbf{I} + 2\frac{1}{2}\theta\frac{1}{2}\theta + 2 \cdot 1[\frac{1}{2}\theta]_\times \end{aligned}$$

Where we marginalize any products of errors, resulting in

$$\begin{aligned} &= (1-0)\mathbf{I} + 0 + [\theta]_\times \\ C(\mathbf{q}) &\simeq \mathbf{I} + [\theta]_\times \end{aligned}$$

The equivalent calculation can be done for $C(q)^\top$ under the assumption of small-angle quaternions and using the definition of conjugate quaternions

$$C(\mathbf{q})^\top \simeq C(\begin{bmatrix}\frac{1}{2}\theta\\1\end{bmatrix})^\top = C(\begin{bmatrix}-\frac{1}{2}\theta\\1\end{bmatrix}) =$$
$$= (1^2 - \frac{1}{2}\theta^\top\frac{1}{2}\theta)\mathbf{I} + 2\frac{1}{2}\theta\frac{1}{2}\theta^\top - 2\cdot 1[\frac{1}{2}\theta]_\times$$

Where we marginalize any products of errors, resulting in

$$= (1-0)\mathbf{I} + 0 + [\theta]_\times$$
$$C(\mathbf{q})^\top \simeq \mathbf{I} - [\theta]_\times$$

D.2.2 Quaternion Derivative

The change in rotation over time expressed in a quaternion can be calculated using the change from $q(t)$ to $q(t+\Delta t)$ and using these descriptions in the derivation

$$\frac{dq(t)}{dt} \triangleq \lim_{\Delta t\to 0}\frac{q(t+\Delta t)-q(t)}{\Delta t}$$

We can set $q(t+\Delta t) = q\otimes\Delta q$ to receive

$$\dot{q} = \lim_{\Delta l\to 0}\frac{q\otimes\Delta q - q}{\Delta t}$$
$$= \lim_{\Delta t\to 0}\frac{q\otimes\left(\begin{bmatrix}\Delta\frac{1}{2}\phi\\1\end{bmatrix} - \begin{bmatrix}\mathbf{0}\\1\end{bmatrix}\right)}{\Delta t}$$
$$= \lim_{\Delta t\to 0}\frac{\mathbf{q}\otimes\begin{bmatrix}\Delta\frac{1}{2}\phi\\0\end{bmatrix}}{\Delta t}$$

We can now define the angular rate as

$$\omega(t) \triangleq \frac{d\phi(t)}{dt} \triangleq \lim_{\Delta t\to 0}\frac{\Delta\phi}{\Delta t}$$

which is simply the change of perturbation per timestep.

Using this, we can now simplify to

$$\dot{q} = \frac{1}{2}\mathbf{q}\otimes\begin{bmatrix}\omega\\0\end{bmatrix}$$