SAPIENZA
UNIVERSITÀ DI ROMA

MULTI-STATE CONSTRAINT KALMAN FILTER FOR MONOCULAR VISUAL-INERTIAL NAVIGATION

Faculty of Information Engineering, Informatics, and Statistics
Department of Computer, Control, and Management Engineering Antonio Ruberti
Master's Degree in Artificial Intelligence and Robotics

Valerio Spagnoli
ID number 1887715

Advisor
Prof. Giorgio Grisetti

Co-Advisor
Dr. Luca Di Giammarino

Academic Year 2024-2025

SAPIENZA
UNIVERSITÀ DI ROMA

Multi-State Constraint Kalman Filter for Monocular Visual-Inertial Navigation

Faculty of Information Engineering, Informatics, and Statistics
Department of Computer, Control, and Management Engineering Antonio Ruberti
Master's Degree in Artificial Intelligence and Robotics

Valerio Spagnoli
ID number 1887715

Advisor
Prof. Giorgio Grisetti

Co-Advisor
Dr. Luca Di Giammarino

Academic Year 2024/2025

Thesis defended on 24 March 2025
in front of a Board of Examiners composed by:

Prof. Domenico Lembo (chairman)
Prof. Laura Astolfi
Prof. Edoardo Barba
Prof. Giorgio Grisetti
Prof. Luca Iocchi
Prof. Simone Lenti
Prof. Fabio Patrizi

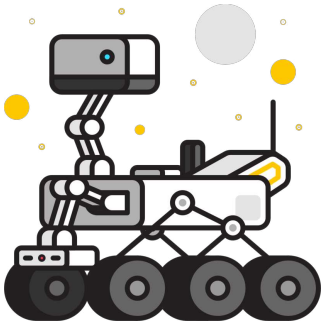
Multi-State Constraint Kalman Filter for Monocular Visual-Inertial Navigation
Master's Thesis. Sapienza University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Author's email: spagnoli.1887715@studenti.uniroma1.it

"Curiosity is such a powerful force. Without it, we wouldn't be who we are today. Curiosity is the passion that drives us through our everyday lives. We have become explorers and scientists with our need to ask questions and to wonder."

— *Clara Ma*



Acknowledgments

Ringrazio il mio relatore, Giorgio Grisetti, per avermi dato l'opportunità di approfondire questo argomento e per avermi fornito le conoscenze necessarie per affrontarlo al meglio. Un sincero grazie anche al mio correlatore, Luca Di Giammarino, per il tempo e il supporto dedicatomi durante lo sviluppo di questa tesi.

Un grande ringraziamento ai miei genitori, che mi hanno sostenuto dall'inizio alla fine. Il loro supporto e i loro consigli sono stati fondamentali per la mia crescita, così come il loro esempio, che è stato il faro da seguire durante tutti questi anni.

Un ringraziamento speciale va anche a mia sorella Lucrezia, la cui tenacia e costanza nell'affrontare un percorso tanto impegnativo mi hanno dato l'esempio e la forza di non mollare.

Infine, un enorme ringraziamento va ad Anna. Il suo supporto in questi anni è stato fondamentale, così come il suo modo di farmi sentire accettato, insegnandomi che essere se stessi è essenziale per esprimersi al meglio. La sua capacità di ascoltare, anche i silenzi, e di guidarmi con i suoi consigli è stata la chiave del mio percorso.

Abstract

Accurate and reliable localization in GNSS-denied environments remains a fundamental challenge in robotics and navigation. This thesis presents a comprehensive study and implementation of the Multi-State Constraint Kalman Filter (MSCKF) [16] for monocular Visual-Inertial Odometry (VIO), addressing the critical need for precise real-time localization with minimal computational overhead.

The MSCKF algorithm fuses visual observations with inertial measurements from an IMU within an Extended Kalman Filter (EKF) framework, enabling robust and efficient pose estimation. To enhance accuracy, the XFeat library [19] was integrated for feature extraction and matching. Leveraging a Convolutional Neural Network (CNN)-based architecture, XFeat provides a fast, precise, and hardware-independent solution. Additionally, an epipolar matching refinement step was incorporated into the MSCKF pipeline, significantly improving accuracy without a substantial increase in computational cost.

A detailed theoretical foundation covering projective geometry, IMU modeling, and Bayesian filtering techniques supports the proposed approach.

Experimental evaluations on multiple photorealistic datasets demonstrate the system's consistency and accuracy, even under challenging conditions such as reflections, motion blur, and high IMU noise levels. Despite being implemented in Python, performance profiling indicates that the system achieves real-time operation on standard hardware, highlighting its potential for further optimization through implementation in C++.

Future research directions may include developing a robust and reliable IMU initialization pipeline that does not solely depend on a V-SLAM system, which is the predominant approach in state-of-the-art methods. Further improvements could focus on enhancing data association for greater robustness in dynamic environments and integrating loop closure detection to enhance long-term reliability in real-world robotic applications.

Acronyms

AD Allan Deviation.

ARW Angular Random Walk.

ATE Absolute Trajectory Error.

AVAR Allan Variance.

BA Bundle Adjustment.

BI Bias Instability.

CNN Convolutional Neural Network.

CV Computer Vision.

EKF Extended Kalman Filter.

FoV Field of View.

GNSS Global Navigation Satellite System.

IDP Inverse Depth Parametrization.

IMU Inertial Measurement Unit.

KF Kalman Filter.

LiDAR Light Detection and Ranging.

LS Least-Squares.

MEMS Micro-ElectroMechanical Systems.

MSCKF Multi-State Constraint Kalman Filter.

PDF Probability Density Function.

PSD Power Spectral Density.

RPE Relative Pose Error.

RRW Rate Random Walk.

SLAM Simultaneous Localization and Mapping.

V-SLAM Visual SLAM.

VI-SLAM Visual-Inertial SLAM.

VIO Visual-Inertial Odometry.

VO Visual Odometry.

VRW Velocity Random Walk.

WLS Weighted Least-Squares.

Nomenclature

$\mathbf{A}^{+}$	Moore-Penrose pseudoinverse of matrix $\mathbf{A}$
${}^{\mathrm{A}}\mathbf{R}_{\mathrm{B}}$	Rotation matrix of frame B in frame A
${}^{\mathrm{A}}\mathbf{T}_{\mathrm{B}}$	Homogeneous transformation matrix of frame B in frame A
${}^{\mathrm{A}}\mathbf{x}$	Vector $\mathbf{x}$ expressed in frame A
$\mathbf{I}_{\mathrm{N}}$	Identity matrix of dimension $N \times N$
$\lfloor \mathbf{x} \times \rfloor$	Skew-symmetric matrix of a vector $\mathbf{x} \in \mathbb{R}^{3\times 1}$
$\mathbf{x}_{\mathcal{H}}$	Vector $\mathbf{x}$ expressed in homogeneous coordinates

Contents

List of Figures

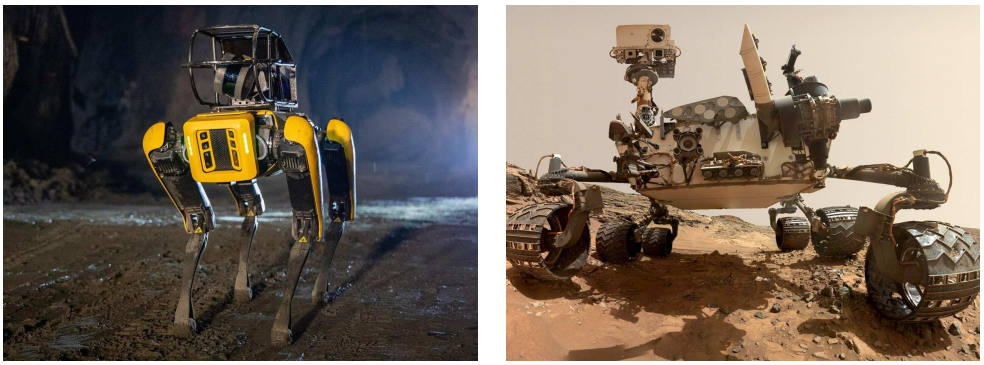
List of Tables

List of Algorithms

Chapter 1

Introduction

Accurate and reliable localization in GNSS-denied environments remains a critical challenge in robotics and autonomous systems. Numerous applications, including terrestrial and space robotic navigation, search and rescue operations, and augmented reality, demand precise real-time estimation of position and orientation.



(a) *Spot* (b) *Curiosity*

Figure 1.1. Examples of robots requiring precise localization in critical environments: (a) Boston Dynamics' *Spot*, used in search and rescue operations, and (b) NASA's *Curiosity* Mars Rover.

Over the past two decades, a variety of solutions have been developed to address this challenge, which can be broadly categorized into three main approaches:

- Proprioceptive-only systems,
- Exteroceptive-only systems, and
- Sensor fusion systems.

Proprioceptive-only systems leverage measurements from sensors such as Inertial Measurement Unit and wheel encoders to estimate the robot's position and attitude.

However, these systems are highly sensitive to noise and drift, and thus are typically employed only in specific scenarios where computational resources are limited, usually combined with high-cost sensors. An example of a proprioceptive-only pose estimation system is that implemented in NASA's Mars Curiosity Rover [21], where attitude is estimated using gyroscopes, and position is derived from wheel encoder measurements. The Mars Rovers periodically correct the drift by replacing these estimates with more accurate ones provided by a Visual Odometry system, invoked less frequently by operators from Earth.

Exteroceptive-only systems, such as Visual and LiDAR Odometry, estimate the pose of the sensor-affixed frame by utilizing environmental data captured from the surroundings. These methods offer several advantages, including greater robustness to noise compared to proprioceptive-only odometry. Additionally, they enable the construction of an environmental map, which can be leveraged for localization optimizations and loop closures through place recognition algorithms.

However, systems that rely exclusively on LiDARs and cameras also have several drawbacks. One major limitation is the high computational cost associated with odometry algorithms, which can negatively impact real-time performance. Additionally, these systems are highly sensitive to environmental conditions; cameras struggle in poor lighting, fog, and repetitive visual patterns, while LiDARs are affected by dust and adverse weather. Furthermore, both sensor types are susceptible to occlusions and challenges in dynamic environments, which can degrade localization accuracy.

Sensor fusion techniques aim to address these limitations by integrating both exteroceptive and proprioceptive sensors to improve robustness. Common configurations include sensors such as Inertial Measurement Unit (IMU) or wheel encoders combined with LiDARs or cameras.

In this context, *Monocular Visual-Inertial Odometry*, which combines a single camera with an IMU, is an attractive solution due to its low cost, sensor reliability even in challenging environments, and lightweight, compact design.

Within this family, one of the most well known and widely adopted solutions is the Multi-State Constraint Kalman Filter (MSCKF), introduced in 2007 by Mourikis and Roumeliotis [16] and further refined in numerous subsequent works.

The MSCKF was developed with the goal of design a VIO system that integrates an Inertial Measurement Unit and a monocular camera, allowing for real-time pose estimation without the need to optimize the entire trajectory or construct an explicitly optimized map. To achieve this, the framework maintains an EKF state that consists only of the current IMU state and the camera poses from which the observed features were initially detected. By structuring the state in this way, MSCKF ensures *computational efficiency* by avoiding the direct handling of map points in the filter. Instead, the map is indirectly refined through the optimization of camera poses within the EKF state, significantly reducing computational complexity

while maintaining accurate state estimation.

Furthermore, once a feature is lost from tracking, it is used to correct the state and is subsequently removed, keeping the number of processed features *bounded*. Similarly, camera poses can be removed from the state when they no longer track any features or when the maximum number of camera states has been reached, ensuring an efficient state representation.

This thesis is organized as follows:

- **Chapter 2 - Related Works**: This chapter introduces the standard nomenclature commonly used in SLAM literature and provides a survey of the most relevant SLAM techniques proposed in recent years.
- **Chapter 3 - Projective Geometry**: This chapter introduces the fundamental concepts of projective geometry relevant to monocular vision-based localization. It covers the pinhole camera model, epipolar geometry, multiple-view monocular 3D reconstruction and Inverse Depth Parametrization (IDP).
- **Chapter 4 - Inertial Measurement Unit**: This chapter describes the working principles of inertial sensors, including accelerometers and gyroscopes, their error sources, and the modeling techniques used to mitigate drift and bias. It also discusses motion integration and IMU initialization strategies.
- **Chapter 5 - Bayes Filters**: This chapter presents the mathematical principles of probabilistic state estimation, covering Bayes filters and Kalman Filter, with a focus on Extended Kalman Filter, which provides the theoretical background for MSCKF-based localization.
- **Chapter 6 - Multi-State Constraint Kalman Filter**: This chapter provides an in-depth explanation of the MSCKF algorithm, covering its state representation, propagation, update mechanisms, feature management, and update strategies.
- **Chapter 7 - Experiments and Results**: This chapter outlines the experimental setup used to evaluate the MSCKF framework, including dataset descriptions, metrics for evaluation, and performance analysis. It presents quantitative results and discusses the accuracy and efficiency of the proposed approach.
- **Chapter 8 - Conclusions**: The final chapter summarizes the key findings of the thesis, highlighting the strengths and limitations of the proposed approach.

Chapter 2

Related Works

Visual Odometry (VO) and Visual-Inertial Odometry (VIO) have been very active research topics in recent years. Since VIO systems are frequently developed as extensions of Visual Odometry systems, this section will first introduce the most important contributions in Visual Odometry, followed by Visual-Inertial Odometry methods. However, it is firstly clarified the nomenclature, with a general introduction on VO and VIO systems.

State-of-the-art localization techniques commonly rely on Simultaneous Localization and Mapping (SLAM), a fundamental approach in robotics that enables a robot to concurrently construct a map of an unknown environment and estimate its own pose within it. Generally, SLAM algorithms can be classified into two main categories:

- *Smoothing-based* (or *optimization-based*) methods, and
- *Filtering-based* methods.

Smoothing-based SLAM algorithms, considered state-of-the-art in terms of accuracy, are implemented using factor graph optimization techniques [10]. These methods represent the estimation problem through a *factor graph*, in which *nodes* denote the robot states to be estimated, and *factors* define constraints between these nodes based on relative measurements.

A significant advantage of smoothing-based SLAM is its inherent flexibility in integrating or removing sensors. Additionally, all sensors are treated uniformly within the factor graph framework: sensors like the IMU are not solely used for state propagation, nor are sensors such as cameras or LiDARs restricted to correction. Instead, each sensor introduces constraints that equally contribute to a unified and consistent representation of the robot's state.

On the other hand, filtering-based SLAM approaches have been the predominated approach for localization tasks for many years, particularly through the adoption of EKF-based pose estimation systems. EKF methods were considered the

standard choice for many applications due to their optimality under Gaussian noise assumptions, simplicity of implementation, and effective sensor fusion capabilities. Furthermore, the computational complexity of EKF-based methods remains bounded as long as the state dimension does not grow.

Strictly related to the Visual Odometry and Visual SLAM is the management of the camera measurements, which is commonly referred as front-end in SLAM applications. As clearly explained in [6], Visual Odometry (VO) methods can be classified as follows:

- *Direct* and *Indirect* methods:
  - Indirect methods extract and match image features (e.g., keypoints) across frames to obtain geometric measurements, then optimize a geometric error, such as reprojection error.
  - Direct methods skip feature extraction and directly optimize photometric errors from raw intensity measurements, leveraging information from the entire image.
- *Sparse* and *Dense* methods:
  - Sparse methods reconstruct and utilize only a small set of independent points (typically corners), without enforcing geometric constraints between these points.
  - Dense (or semi-dense) methods aim to reconstruct and utilize nearly every pixel by applying geometric priors.

It is also clarified in [6] that these classifications are independent, and all combinations exist: sparse-indirect, dense-indirect, sparse-direct, and dense-direct.

Finally, specifically regarding sensor fusion, SLAM methods can be classified as:

- *Loosely coupled*: Visual and inertial data are processed independently to produce separate odometry estimates, which are later integrated into a filtering-based or optimization-based framework.
- *Tightly coupled*: Visual features and IMU measurements are directly combined within a unified estimation framework, generating a single odometry estimate from both sources.

2.1 Visual-SLAM

Visual SLAM (V-SLAM) relies exclusively on camera inputs to estimate the motion of the robot. Several types of camera sensors have been explored in V-SLAM, such as monocular, stereo, and RGB-D cameras. Stereo and RGB-D cameras have been

adopted as the gold standard in many applications; however, monocular cameras have also been extensively studied, as they offer a compact and efficient solution, which is essential for robotics applications and embedded systems.

MonoSLAM [5], introduced by Davison et al. in 2007, was one the first real-time monocular SLAM systems. It employed an Extended Kalman Filter to maintain and update a sparse map of 3D landmarks while simultaneously estimating the camera pose. Despite limitations due to computational constraints and drift accumulation over long trajectories, MonoSLAM established the base for modern indirect V-SLAM methods.

PTAM (Parallel Tracking and Mapping) [12], introduced by Klein and Murray in 2007, separated camera tracking and mapping into two parallel threads. This approach enabled real-time operation by continuously tracking the camera pose in each frame, while asynchronously refining the 3D map through Bundle Adjustment. PTAM significantly enhanced the robustness of monocular SLAM, particularly for augmented reality applications.

LSD-SLAM (Large-Scale Direct SLAM) [7], proposed by Engel et al. in 2014, introduced a direct method that estimates camera motion and builds a semi-dense 3D map by minimizing photometric errors, rather than relying on feature correspondences. This method enabled robust, large-scale monocular SLAM with reduced drift through the use of pose-graph optimization.

ORB-SLAM [17], introduced by Mur-Artal et al. in 2015, is an indirect SLAM system incorporating robust loop closure detection, map reuse, and relocalization capabilities. Using ORB features, it performs keyframe-based tracking, local mapping, and global optimization through Bundle Adjustment. ORB-SLAM has become one of the most widely adopted Visual SLAM systems.

DSO (Direct Sparse Odometry) [6], introduced by Engel et al. in 2016, is a sparse, direct visual odometry approach that optimizes a photometric error within a sliding window of recent frames. Unlike LSD-SLAM, which constructs semi-dense maps with explicit loop closures, DSO maintains a sparse representation without enforcing geometric priors, making it particularly accurate and robust even in environments with limited texture or repetitive patterns.

2.2 Visual-Inertial SLAM

The MSCKF (Multi-State Constraint Kalman Filter) [16], introduced by Mourikis and Roumeliotis in 2007, established an efficient EKF-based Visual-Inertial Odometry framework. By marginalizing feature positions, parametrized using Inverse Depth Parametrization [4], while preserving constraints across multiple camera views, MSCKF enabled accurate real-time state estimation.

In 2012, Li and Mourikis [14] enhanced MSCKF by addressing consistency issues

arising from incorrect observability modeling. Their improved version, MSCKF 2.0, introduced observability-constrained updates and online IMU-camera calibration, significantly improving the accuracy and robustness of VIO systems.

ROVIO (Robust Visual-Inertial Odometry), proposed by Bloesch et al. in 2015 [2], is an EKF-based tightly coupled VIO framework. It introduced direct intensity-based feature tracking combined with Inverse Depth Parametrization, resulting in high robustness against motion blur and feature loss.

OKVIS (Optimization-based Keyframe VIO System) [13], developed by Leutenegger et al. in 2015, follows a keyframe-based non-linear optimization approach for VIO. It jointly optimizes camera poses, IMU states, and feature positions within a sliding window of keyframes, improving accuracy compared to filtering-based methods.

In 2017, Sun et al. [23] proposed S-MSCKF (Stereo Multi-State Constraint Kalman Filter), an extension of the MSCKF framework that incorporates stereo vision to enhance robustness in 3D feature estimation and tracking while preserving the efficient feature management of MSCKF. Unlike optimization-based approaches such as OKVIS, S-MSCKF maintains the computational efficiency of filtering-based methods while achieving comparable accuracy.

VINS-Mono (Monocular Visual-Inertial System) [20], introduced by Qin et al. in 2017, is a tightly coupled optimization-based VIO system featuring loop closure detection, relocalization, and global pose graph optimization. It achieves high accuracy by tightly fusing pre-integrated IMU measurements with visual feature observations. Similarly, VI-ORB-SLAM [18], also introduced in 2017, extends ORB-SLAM into a tightly coupled VI-SLAM framework by incorporating IMU preintegration constraints, significantly improving robustness and consistency compared to the purely visual approach.

Among the most recent contributions, ORB-SLAM3 [3] and OpenVINS [9] stand out. ORB-SLAM3, developed by Campos et al. in 2020, extends the original ORB-SLAM framework by integrating monocular, stereo, and visual-inertial modalities into a unified system. It enhances capabilities such as map reuse, relocalization, and multi-session operation, making it one of the most versatile SLAM systems available. OpenVINS, developed by Geneva et al. in 2020, is an open-source, modular visual-inertial state estimator implementing state-of-the-art VIO algorithms, including MSCKF variants and optimization-based approaches. Its modularity allows flexible adaptation across various robotic applications.

Chapter 3

Projective Geometry

A camera is an exteroceptive sensor that maps the 3D world onto a 2D plane by projecting spatial points through a perspective transformation. In robotics, cameras are extensively used to gather information about the external environment, enabling tasks such as localization, navigation, planning, and obstacle avoidance.

Although there are many types of cameras available today, this thesis focuses on monocular cameras. Their low cost and compact size make them ideal for embedded and wearable devices.

3.1 Pinhole Camera Model

The *pinhole camera model* is one of the simplest and most widely used model. It defines how a 3D point in space $\mathbf{x} \in \mathbb{R}^3$ is projected onto a $2D$ image plane, resulting in an *image point* ${}^{\text{Im}}\mathbf{x} \in \mathbb{R}^2$. This mapping is described by a projection function:

$$ {}^{\text{Im}}\mathbf{x} = proj\left(\mathbf{x}\right) \tag{3.1} $$

which maps points from the Euclidean 3-space $\mathbb{R}^3$ to Euclidean 2-space $\mathbb{R}^2$. The characteristics of this projection are determined by the *intrinsic parameters* of the camera.

3.1.1 Intrinsic Parameters and Projection Operation

The *intrinsic parameters* defines how the projection function *proj* maps a 3D point in space to a 2D image point. These parameters include:

- Focal length f: the distance between the image plane and the camera's optical centre.
- Principal point $\mathbf{p} = (p_u, p_v)$: the point on the image plane where the camera's *principal axis* intersect it.

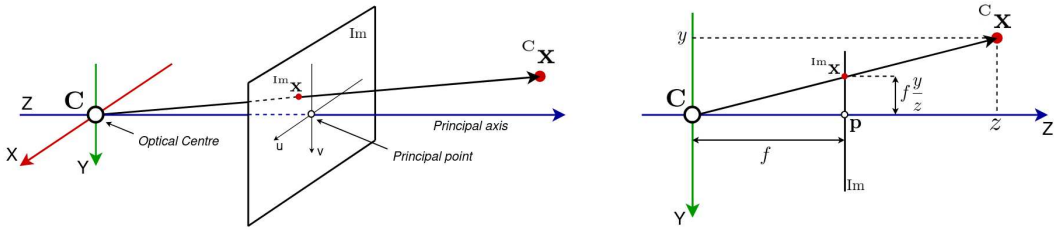


Figure 3.1. *Adapted from*: [11] - Pinhole camera geometry: C is the camera optical centre, **p** is principal point and f is the focal length.

Using these parameters, the projection function *proj* can be expressed as:

$$^{\text{Im}}\mathbf{x} = \begin{bmatrix} u \\ v \end{bmatrix} = proj\left(\mathbf{x}\right) = proj\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}\right) = \begin{bmatrix} f\frac{x}{z} + p_u \\ f\frac{y}{z} + p_v \end{bmatrix} \tag{3.2}$$

The Equation 3.2 can be expressed in homogeneous coordinates as:

$$^{\text{Im}}\mathbf{x}_{\mathcal{H}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = proj\left(\mathbf{x}_{\mathcal{H}}\right) = \frac{1}{z}\begin{bmatrix} f & 0 & p_u & 0 \\ 0 & f & p_v & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{3.3}$$

where the matrix

$$\mathbf{K} = \begin{bmatrix} f & 0 & p_u \\ 0 & f & p_v \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

is known as the *intrinsic matrix*. This matrix encapsulate the intrinsic parameters of the camera and describe how the projection function *proj* maps a 3D point onto the image plane. Using the intrinsic matrix, the Equation 3.3 can be formulated as:

$$^{\text{Im}}\mathbf{x}_{\mathcal{H}} = \frac{1}{z}\left[\mathbf{K} \mid \mathbf{0}\right]\mathbf{x}_{\mathcal{H}} \tag{3.5}$$

where:

$$\mathbf{P} = \left[\mathbf{K} \mid \mathbf{0}\right] \tag{3.6}$$

is called *projection matrix*.

3.1.2 Extrinsic Parameters

In general, points in space are expressed in the *world coordinate frame* W. However, to correctly apply the projection operation described in Section 3.1.1, the 3D point must be expressed in *camera coordinate frame* C. These to coordinated frames are related through a rotation and translation.

In robotics, it is common to define the world coordinate frame W such that the x-axis points forward and the z-axis points upward. In contrast, the camera

coordinate frame C is typically defined such that the z-axis points forward and the y-axis points downward.

The homogeneous transformation matrix that converts a 3D point ${}^{\mathrm{W}}\mathbf{x}_{\mathcal{H}}$ from W to C, and vice-versa, is denoted as ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}}$:

$$ {}^{\mathrm{C}}\mathbf{x}_{\mathcal{H}} = {}^{\mathrm{C}}\mathbf{T}_{\mathrm{W}}\,{}^{\mathrm{W}}\mathbf{x}_{\mathcal{H}} \tag{3.7} $$

This transformation, illustrated in Figure 3.2, is formulated as:

$$ {}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}} = \begin{bmatrix} {}^{\mathrm{W}}\mathbf{R}_{\mathrm{C}} & {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c}} \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 & {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_x}} \\ -1 & 0 & 0 & {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_y}} \\ 0 & -1 & 0 & {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_z}} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.8} $$

where ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{C}}$ represents a rotation of $-\pi/2$ around the x-axis, followed by a rotation of $-\pi/2$ around z-axis:

$$ {}^{\mathrm{W}}\mathbf{R}_{\mathrm{C}} = \mathbf{R}_{\mathrm{z}}\left(-\frac{\pi}{2}\right)\mathbf{R}_{\mathrm{x}}\left(-\frac{\pi}{2}\right) = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{3.9} $$

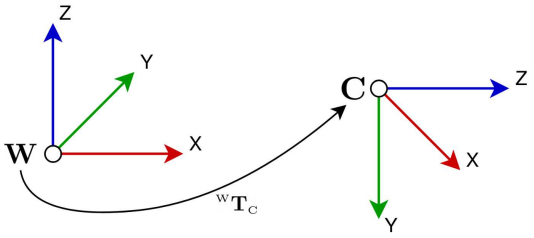


Figure 3.2. *Adapted from*: [11] - The Euclidean transformation between the world and camera coordinate frames.

By incorporating the extrinsic parameters directly into the formulation of $\mathbf{P}$, the camera projection matrix becomes:

$$ \mathbf{P} = \mathbf{K}\left[{}^{\mathrm{C}}\mathbf{R}_{\mathrm{W}} \mid {}^{\mathrm{C}}\mathbf{t}_{\mathrm{w}}\right] = \mathbf{K}\left[{}^{\mathrm{W}}\mathbf{R}_{\mathrm{C}}^{\mathrm{T}} \mid -{}^{\mathrm{W}}\mathbf{R}_{\mathrm{C}}^{\mathrm{T}}{}^{\mathrm{W}}\mathbf{t}_{\mathrm{c}}\right] \tag{3.10} $$

With this formulation the projection operation can be written as:

$$ {}^{\mathrm{Im}}\mathbf{x}_{\mathcal{H}} = \frac{1}{z}\mathbf{P}\,{}^{\mathrm{W}}\mathbf{x} = \frac{1}{z}\mathbf{K}\left[{}^{\mathrm{C}}\mathbf{R}_{\mathrm{W}} \mid {}^{\mathrm{C}}\mathbf{t}_{\mathrm{w}}\right]{}^{\mathrm{W}}\mathbf{x} \tag{3.11} $$

3.1.3 Inverse Projection

The *inverse projection* is the operation that transforms a 2D image point ${}^{\mathrm{Im}}\mathbf{x}$ into a 3D point in space, expressed in camera coordinates ${}^{\mathrm{C}}\mathbf{x}$. However, from a single image, it is not possible to directly determine the depth (z coordinate) of the point,

i.e., its distance from the camera's optical centre. Instead, the inverse projection provides a direction vector, which defines a line in space along which the 3D point must lie.

The inverse projection operation is formulated as:

$$^{\mathrm{C}}\tilde{\mathbf{x}} = proj^{-1}\left(^{\mathrm{Im}}\mathbf{x}_{\mathcal{H}}\right) = \mathbf{K}^{-1}\ ^{\mathrm{Im}}\mathbf{x}_{\mathcal{H}} \tag{3.12}$$

where $^{\mathrm{C}}\tilde{\mathbf{x}}$ represents the direction vector that passes through the camera's optical centre and the 3D point in camera coordinates $^{\mathrm{C}}\mathbf{x}$, as illustrated in Figure 3.3.

The 3D point can be expressed as a function of a *depth parameter* λ, which determines its position along the corresponding projection ray:

$$^{\mathrm{C}}\mathbf{x}(\lambda) = \lambda\,^{\mathrm{C}}\tilde{\mathbf{x}}\ ,\ \lambda \in \mathbb{R} \tag{3.13}$$

where λ represents the *depth* of the point.

Including the extrinsic calibration of the camera, the 3D point in world coordinates W can be obtained as:

$$^{\mathrm{W}}\mathbf{x}(\lambda) = {}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}}\,^{\mathrm{C}}\mathbf{x}(\lambda) = \lambda\,{}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}}\,^{\mathrm{C}}\tilde{\mathbf{x}}\ ,\ \lambda \in \mathbb{R} \tag{3.14}$$

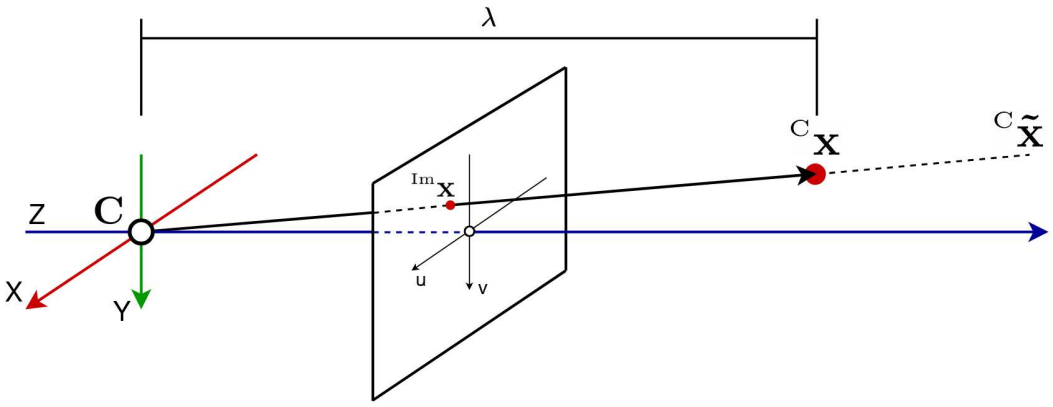


Figure 3.3. *Adapted from*: [11] - The inverse projection operation: the re-projection of the image point $^{\mathrm{Im}}\mathbf{x}$ results in a direction vector $^{\mathrm{C}}\tilde{\mathbf{x}}$ that passes through the camera's optical centre and the 3D point in camera coordinates $^{\mathrm{C}}\mathbf{x}$. However, this operation does not recover the depth λ (z-coordinate) of the 3D point $^{\mathrm{C}}\mathbf{x}$.

3.2 Epipolar Geometry

The *Epipolar Geometry* is a fundamental concept in Computer Vision (CV) that describes the geometric relationship between two camera views of a 3D scene.

Consider a 3D point $\mathbf{x}$ and its corresponding projections on two camera planes: $^{\mathrm{Im1}}\mathbf{x}$ and $^{\mathrm{Im2}}\mathbf{x}$. There exists a geometric relationship between these two image points,

such that knowing one of them, e.g. ${}^{\text{Im1}}\mathbf{x}$, along with information about the relative geometry of the two camera views, imposes a constraint on the possible location of ${}^{\text{Im2}}\mathbf{x}$ on the second image plane, significantly reducing the search space.

3.2.1 Epipolar Plane, Epipoles and Epipolar Lines

The fundamental geometric entities used in epipolar geometry include the *epipolar plane*, the *epipoles* and the *epipolar lines*.

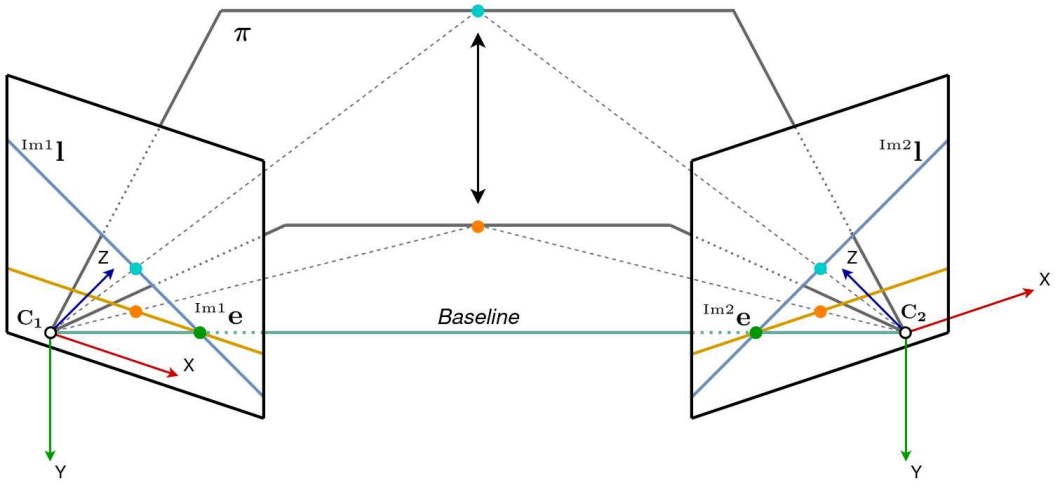


Figure 3.4. *Adapted from*: [11] - Epipolar geometry entities: The baseline intersect the two image planes, Im1 and Im2, at the epipoles ${}^{\text{Im1}}\mathbf{e}$ and ${}^{\text{Im2}}\mathbf{e}$, respectively. The intersection of the epipolar plane π with the two image planes defines the epipolar lines ${}^{\text{Im1}}\mathbf{l}$ and ${}^{\text{Im2}}\mathbf{l}$. Without considering a specific 3D point, this configuration results in a family of epipolar planes, and consequently, a family of epipolar lines on each image plane. All epipolar lines within the same image plane intersect at the corresponding epipole.

Epipoles Consider two cameras with optical centres C_1 and C_2. The epipole in the second image plane, denoted as ${}^{\text{Im2}}\mathbf{e}$, is defined as the image projection of C_1 through the second camera. Similarly, the epipole in the first image plane, denoted as ${}^{\text{Im1}}\mathbf{e}$, corresponds to the image projection of C_2 through the first camera.

The epipoles can also be defined as the *intersection points* of the line joining the camera centres with the image planes. This line, which connects the two camera optical centres, is referred to as the *baseline*.

Epipolar plane The epipolar plane π is defined as any plane containing the baseline. Without considering any specific 3D point, a family of epipolar planes can be identified, as well explained by Hartely and Zisserman [11]. However, when a 3D point and its corresponding projections onto the two images planes are taken into account, a unique epipolar plane is determined.

Epipolar lines The epipolar lines are the intersection of the epipolar plane π with the two image planes, Im1 and Im2. These lines are denoted as ${}^{\text{Im1}}\mathbf{l}$ and ${}^{\text{Im2}}\mathbf{l}$, respectively.

As stated in the previous paragraph, without considering any specific 3D point, a family of epipolar planes can be identified. Consequently, a family of epipolar lines can be identified on both image planes. All these epipolar lines intersect at their respective epipoles.

3.2.2 Fundamental Matrix

The *fundamental matrix* $\mathbf{F}$ is the algebraic representation of epipolar geometry. Given a pair of images, Im1 and Im2, an image point ${}^{\text{Im1}}\mathbf{x}$ in the first image Im1 correspond to an epipolar line ${}^{\text{Im2}}\mathbf{l}$ in the second image Im2, and vice-versa.

The epipolar line is the projection in the second image of the ray ${}^{\text{C1}}\tilde{\mathbf{x}}$ defined by the inverse projection of the image point ${}^{\text{Im1}}\mathbf{x}$ in 3D. Thus, there is a mapping:

$$ {}^{\text{Im1}}\mathbf{x} \mapsto {}^{\text{Im2}}\mathbf{l} \tag{3.15} $$

This mapping is given by the fundamental matrix:

$$ \begin{aligned} {}^{\text{Im2}}\mathbf{l} &= \mathbf{F}\ {}^{\text{Im1}}\mathbf{x}_{\mathcal{H}} \\ {}^{\text{Im1}}\mathbf{l} &= \mathbf{F}^{\text{T}}\ {}^{\text{Im2}}\mathbf{x}_{\mathcal{H}} \end{aligned} \tag{3.16} $$

Epipolar constraint

Consider a 3D point ${}^{\text{W}}\mathbf{x}$ and its corresponding camera coordinates in both views, denoted as ${}^{\text{C1}}\mathbf{x}$ and ${}^{\text{C2}}\mathbf{x}$. The projections of that points onto the image planes Im1 and Im2 are given by ${}^{\text{Im1}}\mathbf{x}$ and ${}^{\text{Im2}}\mathbf{x}$, respectively.

Thus, the 3D point ${}^{\text{W}}\mathbf{x}$ identifies a unique epipolar plane π, which contains the *baseline* and the 3D point itself.

The intersection of the epipolar plane π with the image planes Im1 and Im2 determines the epipolar lines ${}^{\text{Im1}}\mathbf{l}$ and ${}^{\text{Im2}}\mathbf{l}$, respectively. These epipolar lines contain both the corresponding epipole and image point. Consequently, each image point lies on the corresponding epipolar line, leading to the following constraint for both images:

$$ \begin{aligned} {}^{\text{Im1}}\mathbf{x}_{\mathcal{H}}^{\text{T}}\ {}^{\text{Im1}}\mathbf{l} &= 0 \\ {}^{\text{Im2}}\mathbf{x}_{\mathcal{H}}^{\text{T}}\ {}^{\text{Im2}}\mathbf{l} &= 0 \end{aligned} \tag{3.17} $$

By substituting Equation 3.16 into Equation 3.17, the *epipolar constraint* is obtained:

$$ \begin{aligned} {}^{\text{Im2}}\mathbf{x}_{\mathcal{H}}^{\text{T}}\ \mathbf{F}\ {}^{\text{Im1}}\mathbf{x}_{\mathcal{H}} &= 0 \\ {}^{\text{Im1}}\mathbf{x}_{\mathcal{H}}^{\text{T}}\ \mathbf{F}^{\text{T}}\ {}^{\text{Im2}}\mathbf{x}_{\mathcal{H}} &= 0 \end{aligned} \tag{3.18} $$

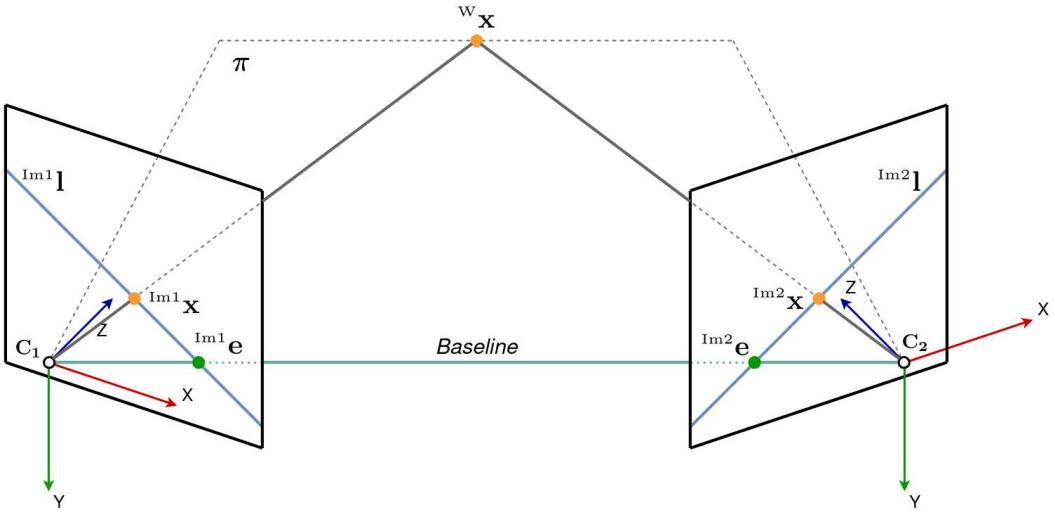


Figure 3.5. *Adapted from*: [11] - Epipolar constraint: given a 3D point $^{W}\mathbf{x}$, a unique epipolar plane π is defined, containing the baseline and the projection rays from both cameras. The epipolar lines, $^{Im1}\mathbf{l}$ and $^{Im2}\mathbf{l}$, are the intersection of π with the two image planes Im1 and Im2. This geometric configuration establish the epipolar constraint, as formulated in Equation 3.18.

3.2.3 Essential Matrix

The *essential matrix* is the specialization of the fundamental matrix to the case of calibrated cameras. It incorporates the *intrinsic camera parameters* into the epipolar geometry, allowing the fundamental matrix to be converted into the essential matrix through the intrinsic calibration matrices:

$$\mathbf{E} = \mathbf{K}_2^{\mathrm{T}} \mathbf{F} \mathbf{K}_1 \tag{3.19}$$

where $\mathbf{K}_1$ and $\mathbf{K}_2$ are the intrinsic matrix of the first and second camera.

Thus, the epipolar constraint in Equation 3.18 also holds for the essential matrix:

$$\begin{aligned} {}^{Im2}\mathbf{x}_{\mathcal{H}}^{\mathrm{T}} \; \mathbf{E} \; {}^{Im1}\mathbf{x}_{\mathcal{H}} = 0 \\ {}^{Im1}\mathbf{x}_{\mathcal{H}}^{\mathrm{T}} \; \mathbf{E}^{\mathrm{T}} \; {}^{Im2}\mathbf{x}_{\mathcal{H}} = 0 \end{aligned} \tag{3.20}$$

This equation was first introduced in 1981 by Longuet-Higgins in [15], where the essential matrix was formalized as a fundamental tool for 3D reconstruction from two views.

Recovering F from E

By rearranging the Equation 3.19, the fundamental matrix can be retrieved from the essential matrix using:

$$\mathbf{F} = \mathbf{K}_2^{-\mathrm{T}} \mathbf{E} \mathbf{K}_1^{-1} \tag{3.21}$$

This relationship allows the transition between *normalized coordinates* (essential matrix) and *pixel coordinates* (fundamental matrix).

Decomposition of E

Consider two cameras:

- the first camera with position and orientation described by ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}_1}$ with respect to the world coordinate frame W;
- the second camera with position and orientation described by ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}_2}$ with respect to the world coordinate frame W.

The relative pose between the two cameras is given by ${}^{\mathrm{C}_1}\mathbf{T}_{\mathrm{C}_2}$, which express the second camera's position and orientation relative to the first camera.

The essential matrix is then computed as:

$$\mathbf{E} = \lfloor {}^{\mathrm{C}_1}\mathbf{t}_{\mathrm{C}_2} \times \rfloor {}^{\mathrm{C}_1}\mathbf{R}_{\mathrm{C}_2} \tag{3.22}$$

where:

- ${}^{\mathrm{C}_1}\mathbf{R}_{\mathrm{C}_2}$ is the relative orientation between the two cameras;
- $\lfloor {}^{\mathrm{C}_1}\mathbf{t}_{\mathrm{C}_2} \times \rfloor$ is the skew-symmetric matrix of the translation vector between the two cameras.

3.2.4 Homography Matrix

A *homography* is a transformation that directly maps corresponding points between two images of the same *planar surface* in space. More generally, a homography describes the relationship between two views when either the camera undergoes a *pure rotation* or the scene itself is *planar*.

Specifically, a homography establishes point correspondences between two images when the scene is planar in 3D space, meaning that all relevant points lie on a single physical plane in the real world. This condition is satisfied in the following cases:

- the points actually belong to the same 3D plane in the real world, or
- the camera undergoes pure rotational motion without translation.

In the second case, all 3D world points that appear in both images have the same relative depth in each view, allowing a homography to describe their transformation.

A homography remains valid even if the physical plane is not perfectly parallel to the image plane. In this scenario, perspective distortions change, but the transformation can still be represented by a single homography matrix.

However, a homography fails when the scene contains depth variations beyond a single plane, as a single transformation can no longer accurately describe the point correspondences.

Consider two cameras with a relative pose ${}^{\mathrm{C}_1}\mathbf{T}_{\mathrm{C}_2}$ and intrinsic matrices $\mathbf{K}_1$ and $\mathbf{K}_2$. The homography matrix is defined as:

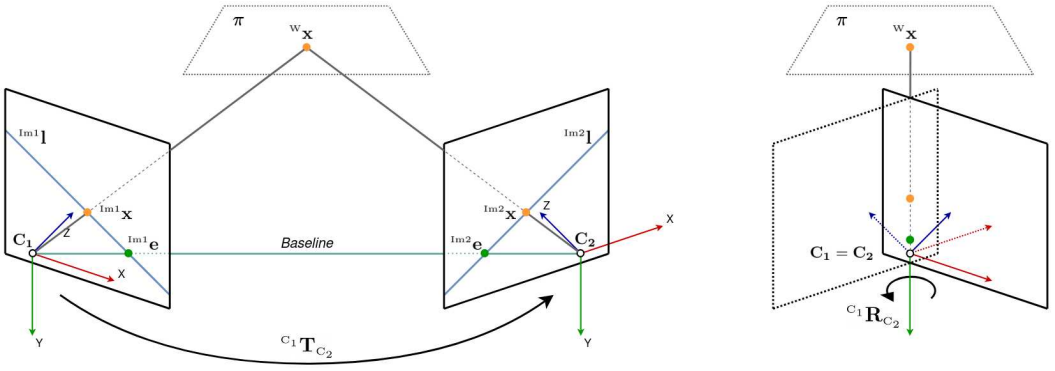


Figure 3.6. *Adapted from*: [11] - Homography: the left diagram illustrates the relationship between the image projections of points lying on the 3D plane π through a homography transformation $\mathbf{H}$. In this case the epipolar geometry is evident because the cameras undergo general motion with a non-zero baseline. The right diagram depicts the special case of pure rotation, where the baseline approaches zero ($C_1 = C_2$), making homography applicable even when the scene is not strictly planar.

$$\mathbf{H} = \mathbf{K}_2 \, {}^{C_1}\mathbf{R}_{C_2} \mathbf{K}_1^{-1} \tag{3.23}$$

This equation expresses how the homography matrix $\mathbf{H}$ transforms points from the first image to their corresponding locations in the second image, assuming the camera undergoes a rotation described by ${}^{C_1}\mathbf{R}_{C_2}$. In [11] the Homography matrix corresponding to a pure rotation is denoted with $\mathbf{H}_\infty$.

From Equation 3.23, the direct relationship between two corresponding image points, ${}^{Im1}\mathbf{x}$ and ${}^{Im2}\mathbf{x}$, which represent projections of the same 3D point ${}^{W}\mathbf{x}$ in world coordinates, is given by:

$$\begin{aligned} {}^{Im2}\mathbf{x} &= \mathbf{H} \; {}^{Im1}\mathbf{x} \\ {}^{Im1}\mathbf{x} &= \mathbf{H}^{-1} \; {}^{Im2}\mathbf{x} \end{aligned} \tag{3.24}$$

These equations highlight the one-to-one correspondence between points in the two images, as established by the homography transformation.

3.2.5 Comparison of F, E and H

The fundamental matrix $\mathbf{F}$ and the essential matrix $\mathbf{E}$ are essential for handling general camera motion, which involves both *rotation* and *translation*. These matrices define the *epipolar constraint*, meaning that a point in the first image corresponds to an *epipolar line* in the second image. The precise location of the corresponding point along this line depends on the depth of the 3D point relative to the camera.

In contrast, the homography matrix $\mathbf{H}$ provides a direct one-to-one mapping between corresponding points in two images. Given a point in the first view, the homography determines its exact location in the second view. However, this

mapping is *independent of depth* and holds only when the scene is *planar* or the camera undergoes *pure rotational motion*.

Due to these properties, homographies are particularly useful for applications such as image *rectification*, *stitching*, and *perspective transformations*. However, in general 3D scenes with depth variations, epipolar constraints and the fundamental matrix become essential for accurate point correspondence.

3.3 Multiple-view Monocular 3D Reconstruction

In Computer Vision, *3D reconstruction* refers to the process of estimating the three-dimensional structure of a scene or object from multiple two-dimensional images. One approach to solving this problem involves the use of *stereocameras*, which are exteroceptive sensors designed to exploit epipolar geometry and directly generate a *depth map* of the scene, which is an image where in each pixel is encoded the depth information. The depth map enables an explicit reconstruction of the environment.

However, for *monocular cameras* it is not possible to directly obtain a depth map. Instead, to perform 3D reconstruction, it is necessary to capture at least two images from different points of view and estimating the 3D position of corresponding points by solving a Least-Squares (LS) problem on their inverse projections (see Section 3.1.3) across multiple views.

3.3.1 Intersection of Lines algorithm

The *intersection of lines* problem consists of estimating the intersection point of N lines that do not intersect at a single point. A Least-Squares solution can be derived to minimize the sum of the perpendicular distances from the estimated intersection point to all the lines.

Line Parametrization

Consider a line in thee-dimensional space. It can be fully described by a point on the line, denoted as the *base point* $\mathbf{b}$, and a direction vector $\mathbf{d}$:

$$\begin{aligned} \mathbf{b} &= \begin{bmatrix} x_b & y_b & z_b \end{bmatrix}^{\mathrm{T}} \\ \mathbf{d} &= \begin{bmatrix} x_d & y_d & z_d \end{bmatrix}^{\mathrm{T}} \quad \text{s.t.} \quad \|d\| = \mathbf{d}^{\mathrm{T}}\mathbf{d} = 1 \end{aligned} \tag{3.25}$$

Thus, the parametric representation of the line is given by:

$$l = \mathbf{b} + \lambda\mathbf{d} \quad \text{with} \quad -\infty < \lambda < \infty \tag{3.26}$$

Distance from a Point to a Line

Consider a line $l = \{\mathbf{b}, \mathbf{d}\}$ and a 3D point $\mathbf{x}$, the squared perpendicular distance from the point $\mathbf{x}$ to the line l is given by:

$$\begin{aligned} D(\mathbf{x}; \{\mathbf{b}, \mathbf{d}\}) &= \left\| (\mathbf{b} - \mathbf{x}) - ((\mathbf{b} - \mathbf{x})^{\mathrm{T}} \mathbf{d})\, \mathbf{d} \right\|^2 \\ &= \left\| (\mathbf{b} - \mathbf{x}) - \mathbf{d}\mathbf{d}^{\mathrm{T}} (\mathbf{b} - \mathbf{x}) \right\|^2 \\ &= \left\| (\mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}}) (\mathbf{b} - \mathbf{x}) \right\|^2 \\ &= (\mathbf{b} - \mathbf{x})^{\mathrm{T}} (\mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}}) (\mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}})^{\mathrm{T}} (\mathbf{b} - \mathbf{x}) \\ &= (\mathbf{b} - \mathbf{x})^{\mathrm{T}} (\mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}}) (\mathbf{b} - \mathbf{x}) \end{aligned} \tag{3.27}$$

The last line holds because the matrix $(\mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}})$ is a *projection matrix* onto the orthocomplement of $\mathbf{d}$, making it idempotent:

$$(\mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}}) (\mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}})^{\mathrm{T}} = \mathrm{I} - 2\mathbf{d}\mathbf{d}^{\mathrm{T}} + \mathbf{d}\mathbf{d}^{\mathrm{T}}\mathbf{d}\mathbf{d}^{\mathrm{T}} = \mathrm{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}} \tag{3.28}$$

This property implies that applying the projection matrix more that once does not further alter a vector, meaning that it removes its component in the direction of $\mathbf{d}$ in the first application.

It is also worth noting that $\mathbf{d}\mathbf{d}^{\mathrm{T}}$ is itself a *projection matrix*, specifically a rank-one projector that projects any vector onto the direction of $\mathbf{d}$. As a result, it satisfies the idempotency property:

$$\mathbf{d}\mathbf{d}^{\mathrm{T}}\mathbf{d}\mathbf{d}^{\mathrm{T}} = \mathbf{d}\mathbf{d}^{\mathrm{T}}. \tag{3.29}$$

Least-Squares Solution to the Intersection of Lines

Consider a set of N lines, denoted as $L = \{\mathbf{B}, \mathbf{D}\}$, which do not perfectly intersect at a single point. A unique Least-Squares solution can be determined by minimizing the sum of squared distances between the point $\mathbf{x}$ and each line $l = \{\mathbf{b}, \mathbf{d}\}$:

$$\begin{aligned} D(\mathbf{x}; \{\mathbf{B}, \mathbf{D}\}) &= \sum_{i=1}^{N} D(\mathbf{x}; \{\mathbf{b}_{\mathrm{i}}, \mathbf{d}_{\mathrm{i}}\}) \\ &= \sum_{i=1}^{N} (\mathbf{b}_{\mathrm{i}} - \mathbf{x})^{\mathrm{T}} (\mathrm{I} - \mathbf{d}_{\mathrm{i}}\mathbf{d}_{\mathrm{i}}^{\mathrm{T}}) (\mathbf{b}_{\mathrm{i}} - \mathbf{x}). \end{aligned} \tag{3.30}$$

and minimizing it with respect to $\mathbf{x}$. Thus, the objective of the Least-Squares problem is:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}}\ D(\mathbf{x}; \{\mathbf{B}, \mathbf{D}\}). \tag{3.31}$$

Taking the derivative of D with respect to $\mathbf{x}$ yields:

$$\frac{\partial D}{\partial \mathbf{x}} = \sum_{i=1}^{N} -2 (\mathrm{I} - \mathbf{d}_{\mathrm{i}}\mathbf{d}_{\mathrm{i}}^{\mathrm{T}}) (\mathbf{b}_{\mathrm{i}} - \mathbf{x}). \tag{3.32}$$

Setting the derivative to zero and rearranging, we obtain the following linear system:

$$\mathbf{H}\mathbf{x} = \mathbf{q}, \quad \mathbf{H} = \sum_{i=1}^{N} \left(\mathrm{I} - \mathbf{d}_{\mathrm{i}}\mathbf{d}_{\mathrm{i}}^{\mathrm{T}}\right), \quad \mathbf{q} = \sum_{i=1}^{N} \left(\mathrm{I} - \mathbf{d}_{\mathrm{i}}\mathbf{d}_{\mathrm{i}}^{\mathrm{T}}\right)\mathbf{b}_{\mathrm{i}}. \tag{3.33}$$

This system can be solved using the *Moore-Penrose* pseudoinverse:

$$\hat{\mathbf{x}} = \mathbf{H}^{+}\mathbf{q}. \tag{3.34}$$

where the pseudoinverse is defined as: $\mathbf{H}^{+} = \mathbf{H}^{\mathrm{T}}\left(\mathbf{H}\mathbf{H}^{\mathrm{T}}\right)^{-1}$.

Weighted Least-Squares Solution

A Weighted Least-Squares (WLS) formulation can be introduced by incorporating a confidence value c_{i} for each line, representing how strongly the solution should be biased toward that line. The cost function then becomes:

$$D\left(\mathbf{x}; \{\mathbf{B}, \mathbf{D}\}\right) = \sum_{\mathrm{i}=1}^{N} c_{\mathrm{i}} \left(\mathbf{b}_{\mathrm{i}} - \mathbf{x}\right)^{\mathrm{T}} \left(\mathrm{I} - \mathbf{d}_{\mathrm{i}}\mathbf{d}_{\mathrm{i}}^{\mathrm{T}}\right)\left(\mathbf{b}_{\mathrm{i}} - \mathbf{x}\right). \tag{3.35}$$

Minimizing this modified objective again leads to a system of linear equations, with $\mathbf{H}$ and $\mathbf{q}$ now weighted by the confidence scores:

$$\mathbf{H} = \sum_{i=\mathrm{i}}^{N} c_{\mathrm{i}} \left(\mathrm{I} - \mathbf{d}_{\mathrm{i}}\mathbf{d}_{\mathrm{i}}^{\mathrm{T}}\right), \quad \mathbf{q} = \sum_{\mathrm{i}=1}^{N} c_{\mathrm{i}} \left(\mathrm{I} - \mathbf{d}_{\mathrm{i}}\mathbf{d}_{\mathrm{i}}^{\mathrm{T}}\right)\mathbf{b}_{\mathrm{i}}. \tag{3.36}$$

The solution remains the same as in Equation (3.34). The confidence scores c_{i} determine how much the solution is affected by each line, introducing a bias toward more reliable lines.

3.3.2 Intersection of Lines for Monocular 3D Reconstruction

In the case of a *monocular camera*, the intersection of lines algorithm can be utilized to estimate the 3D position of a point observed from multiple views.

The lines are obtained through the inverse projection of the image points measured by the camera as it moves through the scene. After performing data association, all image points corresponding to the same world point ${}^{\mathrm{W}}\mathbf{x}$ can be used to compute an estimate of its position, denoted as ${}^{\mathrm{W}}\hat{\mathbf{x}}$.

The lines are parametrized using:

- the camera's position at the moment the point was observed as the base $\mathbf{b}$;
- the inverse projection of the image point, expressed in world coordinates, as the direction $\mathbf{d}$.

Algorithm 1 Intersection of Lines

Require: `lines: List[Line]`
Ensure: $\hat{\mathbf{x}}$: `vector` $(3, 1)$
1: $\mathbf{H} = \texttt{zeros}(3, 3)$
2: $\mathbf{q} = \texttt{zeros}(3, 1)$
3: **for** `line` **in** `lines` **do**
4: $\quad \mathbf{b}$ `= line.base`
5: $\quad \mathbf{d}$ `= line.direction`
6: $\quad \mathbf{c}$ `= line.confidence`
7: $\quad \mathbf{P} = \mathbf{I} - \mathbf{d}\mathbf{d}^{\mathrm{T}}$
8: $\quad \mathbf{H} \pm \mathbf{c} \cdot \mathbf{P}$
9: $\quad \mathbf{q} \pm \mathbf{c} \cdot \mathbf{P}\mathbf{b}$
10: **end for**
11: $\mathbf{x} = \mathbf{H}^{+}\mathbf{q}$

Given a camera pose at instant i, denoted as ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C_i}}$, and an image point ${}^{\mathrm{Im}}\mathbf{x}_{\mathrm{i}}$ observed from that camera, the line parameters are defined as:

$$\begin{aligned} \mathbf{b} &= {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_i}}, \\ \mathbf{d} &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{C_i}}\ proj^{-1}({}^{\mathrm{Im}}\mathbf{x}_{\mathrm{i}}). \end{aligned} \tag{3.37}$$

The confidence associated with each line is determined by the output score of the data association process, which represents the probability that the matching operation correctly assigns the most recently observed image point to the corresponding previous observations.

The estimation of a 3D world point ${}^{\mathrm{W}}\hat{\mathbf{x}}$ using the intersection of lines algorithm, requires sufficient *parallax* between the observed directions $\mathbf{d}_{\mathrm{i}}$ to ensure accurate results. Figure 3.7 illustrates the problem of 3D reconstruction from multiple views.

3.3.3 Reprojection Error

The *reprojection error* is a geometric measure that quantifies the discrepancy between a projected point and its observed counterpart in an image. It evaluates how well the reprojection of an estimated 3D point, ${}^{\mathrm{W}}\hat{\mathbf{x}}$, obtained through 3D reconstruction, aligns with the corresponding measured image point, ${}^{\mathrm{Im}}\mathbf{x}$.

Consider an estimated 3D point ${}^{\mathrm{W}}\hat{\mathbf{x}}$ and a camera characterized by intrinsic parameters $\mathbf{K}$ and extrinsic parameters ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C}}$, which together form the projection matrix $\mathbf{P}$, as defined in Equation 3.10. The image projection of ${}^{\mathrm{W}}\hat{\mathbf{x}}$ is given by:

$${}^{\mathrm{Im}}\hat{\mathbf{x}} = \mathbf{P}\,{}^{\mathrm{W}}\hat{\mathbf{x}} \tag{3.38}$$

while the corresponding measured image point is ${}^{\mathrm{Im}}\mathbf{x}$.

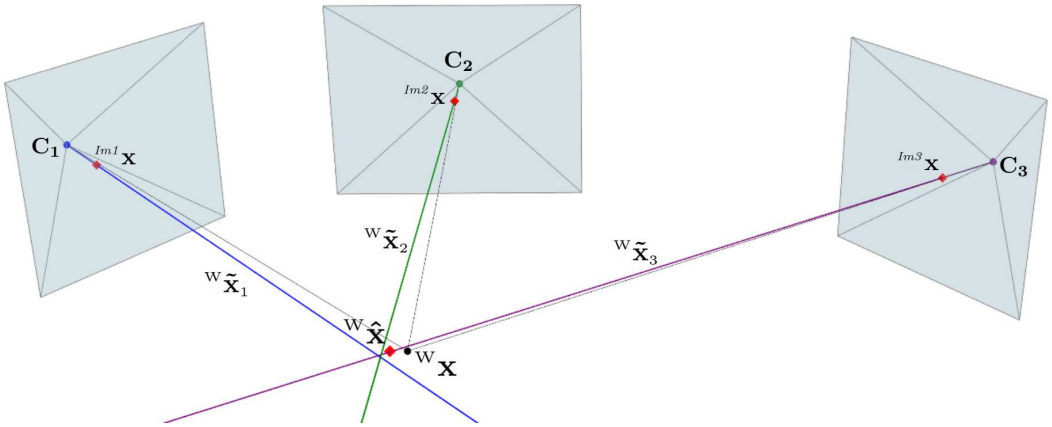


Figure 3.7. Multiple-view 3D reconstruction: the world point ${}^{\mathrm{W}}\mathbf{x}$ (black dot) is projected onto the image planes of three cameras. The projections are denoted with ${}^{\mathrm{Im1}}\mathbf{x}$, ${}^{\mathrm{Im2}}\mathbf{x}$ and ${}^{\mathrm{Im3}}\mathbf{x}$. Due to noise, the measured image points do not perfectly correspond to the actual 3D world point, introducing errors in the inverse projection process. From the lines obtained from the optical camera centres and the inverse projected image points, the 3D position of the point ${}^{\mathrm{W}}\mathbf{x}$ can be estimated using the intersection of line algorithm. The estimated world point is denoted with ${}^{\mathrm{W}}\hat{\mathbf{x}}$ (red diamond).

The reprojection error is then defined as:

$$d\left({}^{\mathrm{Im}}\mathbf{x}, {}^{\mathrm{Im}}\hat{\mathbf{x}}\right) = \left\| {}^{\mathrm{Im}}\mathbf{x} - {}^{\mathrm{Im}}\hat{\mathbf{x}} \right\| \tag{3.39}$$

which represents the Euclidean distance between the two points in the 2D image plane.

In robotics and CV, the reprojection error is widely utilized in various applications, including camera calibration, pose estimation, and bundle adjustment. Indeed, it provides a quadratic objective function that can be directly incorporated into optimization problems.

3.4 Inverse Depth Parametrization

The Inverse Depth Parametrization (IDP) is an alternative 3D point parametrization to the traditional XYZ representation. It was introduced by [4] in the context of monocular Simultaneous Localization and Mapping (SLAM) to address the challenge of estimating feature depth from a single moving camera.

The motivation for Inverse Depth Parametrization arises from the limitations of the standard XYZ parametrization, which struggles with depth uncertainty, particularly for features located far from the camera's optical centre. These distant features exhibit *low parallax*, even when the camera undergoes significant translation, making depth estimation difficult.

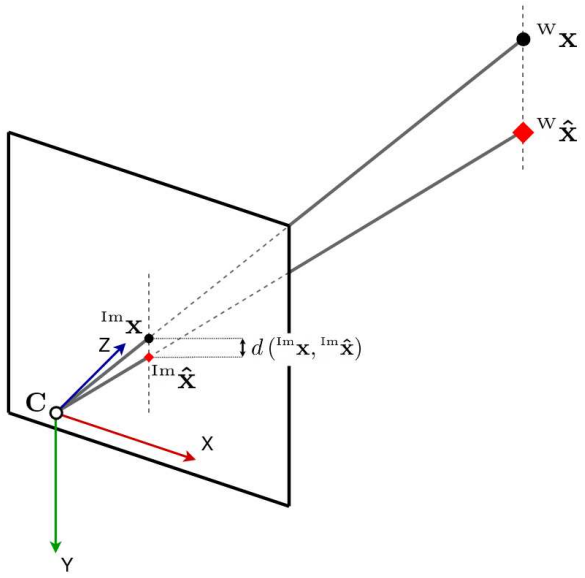


Figure 3.8. Reprojection Error: The estimated 3D point ${}^{\mathrm{W}}\hat{\mathbf{x}}$ (red diamond) is projected onto the image plane at ${}^{\mathrm{Im}}\hat{\mathbf{x}}$, while ${}^{\mathrm{Im}}\mathbf{x}$ represents the true projection of the actual 3D point ${}^{\mathrm{W}}\mathbf{x}$ (black dot). The reprojection error, defined as $d\left({}^{\mathrm{Im}}\mathbf{x}, {}^{\mathrm{Im}}\hat{\mathbf{x}}\right)$, corresponds to the Euclidean distance between ${}^{\mathrm{Im}}\mathbf{x}$ and ${}^{\mathrm{Im}}\hat{\mathbf{x}}$ in the image plane.

3.4.1 Comparison of IDP and XYZ parametrization

The standard parametrization for a 3D point in terms of Euclidean XYZ coordinates is:

$$ {}^{\mathrm{W}}\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}} \tag{3.40} $$

As presented in [4] a 3D point ${}^{\mathrm{W}}\mathbf{x}$ in IDP parametrization can be represented by the 6D state vector:

$$ {}^{\mathrm{W}}\mathbf{y}_{\mathrm{i}} = \begin{bmatrix} x_{\mathrm{b}} & y_{\mathrm{b}} & z_{\mathrm{b}} & \theta_{\mathrm{i}} & \phi_{\mathrm{i}} & \rho_{\mathrm{i}} \end{bmatrix}^{\mathrm{T}} \tag{3.41} $$

which defines a 3D point located at:

$$ {}^{\mathrm{W}}\mathbf{x} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x_{\mathrm{b}} \\ y_{\mathrm{b}} \\ z_{\mathrm{b}} \end{bmatrix} + \frac{1}{\rho_{\mathrm{i}}} \mathbf{m}\left(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}\right) \tag{3.42} $$

Here, ${}^{\mathrm{W}}\mathbf{y}_{\mathrm{i}}$ represents the ray extending from the initial camera position, $\begin{bmatrix} x_{\mathrm{b}} & y_{\mathrm{b}} & z_{\mathrm{b}} \end{bmatrix}^{\mathrm{T}}$, where the feature was first observed. The parameters θ_{i} and ϕ_{i} define the azimuth and elevation angles of the feature ${}^{\mathrm{W}}\mathbf{x}$ in the world frame relative to the i-th camera, forming the unit directional vector $\mathbf{m}\left(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}\right)$. The feature's depth λ_{i} along this ray is encoded using its inverse, $\rho_{\mathrm{i}} = 1/\lambda_{\mathrm{i}}$.

3.4.2 IDP in Monocular Navigation

In monocular navigation applications, such as Visual SLAM (V-SLAM) or Visual Odometry (VO), each observed feature imposes a constraint between the camera's location and the corresponding map feature.

Consider an estimated 3D point ${}^{\mathrm{W}}\hat{\mathbf{x}} = \begin{bmatrix}\hat{x} & \hat{y} & \hat{z}\end{bmatrix}^{\mathrm{T}}$, reconstructed from a set of N measured image points $\{{}^{\mathrm{Im1}}\mathbf{x}, ..., {}^{\mathrm{ImN}}\mathbf{x}\}$, where ${}^{\mathrm{Im1}}\mathbf{x}$ corresponds to the image measurement from the first camera that observed ${}^{\mathrm{W}}\mathbf{x}$.

For each camera $\mathrm{C_i}$ with $\mathrm{i} \in 1, ..., N$, a measurement model can be defined to describe the constraint between the camera itself and the observation of the 3D point ${}^{\mathrm{W}}\hat{\mathbf{x}}$. In XYZ parametrization, the position of the 3D point in camera coordinates is given by:

$$ {}^{\mathrm{C_i}}\hat{\mathbf{x}}_{\mathrm{XYZ}} = {}^{\mathrm{C_i}}\mathbf{R}_{\mathrm{W}} \left({}^{\mathrm{W}}\mathbf{x} - {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_i}} \right) = {}^{\mathrm{C_i}}\mathbf{R}_{\mathrm{W}} \left(\begin{bmatrix}\hat{x} \\ \hat{y} \\ \hat{z}\end{bmatrix} - {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_i}} \right) \tag{3.43} $$

Instead, when using the IDP parametrization, the position of the 3D point in camera coordinates can be expressed as:

$$ {}^{\mathrm{C_i}}\hat{\mathbf{x}}_{\mathrm{IDP}} = {}^{\mathrm{C_i}}\mathbf{R}_{\mathrm{W}} \left(\rho_{\mathrm{i}} \left(\begin{bmatrix}x_{\mathrm{b}} \\ y_{\mathrm{b}} \\ z_{\mathrm{b}}\end{bmatrix} - {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_i}} \right) + \mathbf{m}\left(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}\right) \right) \tag{3.44} $$

where $\begin{bmatrix}x_{\mathrm{b}} & y_{\mathrm{b}} & z_{\mathrm{b}}\end{bmatrix}^{\mathrm{T}} = \begin{bmatrix}x_1 & y_1 & z_1\end{bmatrix}^{\mathrm{T}}$ represents the position of the first camera from which ${}^{\mathrm{W}}\hat{\mathbf{x}}$ was observed.

In both equations, ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C_i}} = \begin{bmatrix}{}^{\mathrm{W}}\mathbf{R}_{\mathrm{C_i}} & {}^{\mathrm{W}}\mathbf{t}_{\mathrm{c_i}}\end{bmatrix}$ represents the pose of the camera employed to define the constraint with the 3D point ${}^{\mathrm{W}}\hat{\mathbf{x}}$.

The key difference between the two formulations lies in the IDP-based measurement model's sensitivity to the parallax angle between the first camera ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C_1}}$ that observed the point ${}^{\mathrm{W}}\mathbf{x}$ and the current camera ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C_i}}$, which establishes the constraint.

When a feature exhibits low parallax, 3D reconstruction becomes unreliable, leading to significant depth estimation errors. In such cases, the standard XYZ-based measurement model fails to provide accurate results in both translation and orientation.

In contrast, the IDP-based measurement model can be approximated as:

$$ {}^{\mathrm{C_i}}\hat{\mathbf{x}}_{\mathrm{IDP}} \approx {}^{\mathrm{C_i}}\mathbf{R}_{\mathrm{W}} \mathbf{m}\left(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}\right) \tag{3.45} $$

This approximation implies that under low parallax conditions, the IDP model primarily provides information about the camera's orientation and the directional vector $\mathbf{m}\left(\theta_{\mathrm{i}}, \phi_{\mathrm{i}}\right)$, while depth estimation remains uncertain.

This property arises from the fact that depth of the point ${}^{\mathrm{W}}\mathbf{x}$ with respect to the camera ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{C_i}}$ is parametrized using its inverse, $\rho_{\mathrm{i}} = 1/\lambda_{\mathrm{i}}$, allowing the model to robustly handle points at infinite depth, where $\rho_{\mathrm{i}} = 0$.

Chapter 4

Inertial Measurement Unit

Accurately measuring motion, specifically acceleration, rotation, and velocity, is essential for determining an object's orientation and position. An Inertial Measurement Unit (IMU) is an electromechanical or solid-state device equipped with various *proprioceptive* sensors, such as accelerometers, gyroscopes, and magnetometers, to capture these motion parameters.

The most widely used technology for manufacturing inertial sensors today is Micro-ElectroMechanical Systems (MEMS). These are highly compact devices that integrate both electrical and mechanical components.

4.1 IMU sensors

The accelerometer and gyroscope are the most commonly used sensors in an IMU, providing a 6 degrees of freedom ego-motion estimation of the pose of the body to which the sensor is attached.

4.1.1 Accelerometer

An accelerometer measures *specific force* (called also *proper acceleration*), the force per unit mass, relative to a free-falling body. Its measurement model is based on the principle of a damped mass-spring system. Essentially, an accelerometer can be visualized as a calibrated mass (*proof mass*) attached to a spring with a known spring constant and a damper with a calibrated damping coefficient. After a calibration, these force measurements are converted into specific-force readings. This principle explains the accelerometer's property of measuring acceleration relative to a free-falling body. A stationary mass suspended on a spring experiences zero net force, yet the force sensor registers an upward force, which is then interpreted as a specific force measurement.

This explains why a stationary accelerometer measures a local gravitational acceleration ${}^{A}\mathbf{g}$ instead of zero, while a free-falling accelerometer registers zero

acceleration.

Displacement measurements are performed using electrical capacitance. The sensor contains several differential capacitors, consisting of fixed electrodes positioned on either side of the proof mass. The proof mass features extension arms that extend into the space between these electrodes. At rest, the extension arms remain centred, generating a known capacitance that represents zero acceleration. When acceleration occurs, the proof mass shifts, causing the extension arms to move closer to one electrode. This displacement alters the capacitance, allowing acceleration to be determined. A schematic representation of the accelerometer's structure is shown in Figure 4.1.

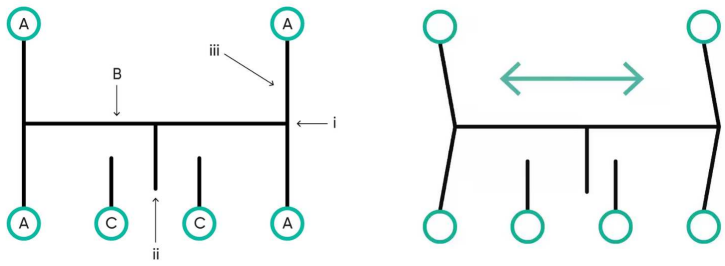


Figure 4.1. *Image source: Advanced Navigation (Accessed: 2025-02-09)* - Accelerometer scheme: (A) anchors of the proof mass, (B) proof mass, (C) fixed electrodes, (i) proof mass centre section, (ii) proof mass extension arms, (iii) link between anchors and proof mass centre section. The proof mass is anchored via flexible links (iii), which function as springs. When the extension arms (ii) move toward a fixed electrode (C), a change in capacitance occurs. In a stationary state or at a constant velocity, the extension arms remain centred.

However, in robotics applications such as localization and navigation, the goal is to measure the accelerometer's coordinate acceleration. To achieve this, the accelerometer's readings are represented using the following parametric model:

$$^{\mathrm{A}}\tilde{\mathbf{a}}(t) = {}^{\mathrm{A}}\mathbf{a}(t) + {}^{\mathrm{A}}\mathbf{b}(t) + {}^{\mathrm{A}}\mathbf{R}_{\mathrm{W}}(t)\,{}^{\mathrm{W}}\mathbf{g} + {}^{\mathrm{A}}\boldsymbol{\eta}(t) \tag{4.1}$$

where:

- $^{\mathrm{A}}\tilde{\mathbf{a}}$ represents the measured specific force expressed in the accelerometer's coordinate frame A;
- $^{\mathrm{A}}\mathbf{a}$ represents the accelerometer's acceleration expressed in the accelerometer's coordinate frame A;
- $^{\mathrm{A}}\mathbf{b}$ represents the bias of the accelerometer expressed in the accelerometer's coordinated frame A;
- $^{\mathrm{W}}\mathbf{g}$ represents the gravity vector expressed in the world's coordinate frame W;

- ${}^{\mathrm{A}}\mathbf{R}_{\mathrm{W}}$ represents the rotation of the world's coordinate frame W in the accelerometer's coordinated frame A;
- ${}^{\mathrm{A}}\boldsymbol{\eta}$ represent the noise associated with the accelerometer expressed in the accelerometer's coordinate frame A.

By modeling the measured specific force in this way, and under certain conditions with appropriate calibration procedures, it becomes possible to isolate the acceleration ${}^{\mathrm{A}}\mathbf{a}$ from the other components.

4.1.2 Gyroscope

Gyroscopes are angular rate sensors that measure an object's angular velocity. The most common method for measuring angular velocity leverages the Coriolis effect. In a typical MEMS Coriolis gyroscope, a proof mass is suspended within a reference frame by mechanical springs. This reference frame is further isolated from an outer frame using additional springs. The proof mass oscillates along a designated axis, known as the *drive axis*. When the gyroscope rotates, it induces a secondary vibration along a perpendicular axis, called the *sense axis*. The displacement of the proof mass increases with the rotation rate, generating a signal proportional to the Coriolis force and, consequently, the sensed angular velocity. A schematic representation of the gyroscope's structure is depicted in figure 4.2

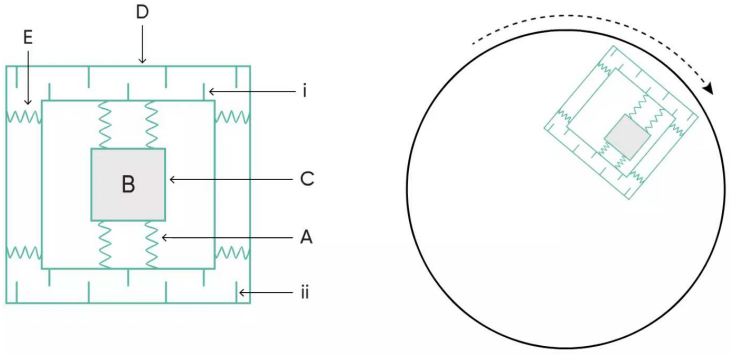


Figure 4.2. *Image source: Advanced Navigation (Accessed: 2025-02-09)* - Gyroscope scheme: (A) Springs isolating the proof mass from the inner reference frame, (B) proof mass, (C) centre of the inner reference frame, (D) outer reference frame, (E) springs isolating the inner reference frame from the outer reference frame, (i) extension arms of the inner reference frame, (ii) fixed electrodes of the outer reference frame. When the gyroscope rotates, the extension arms of the inner reference frame (i) shift toward the fixed electrodes of the outer reference frame (ii), resulting in a change in capacitance.

The angular velocity measured by a gyroscope can be modeled as:

$$ {}^{\mathrm{G}}\tilde{\boldsymbol{\omega}}(t) = {}^{\mathrm{G}}\boldsymbol{\omega}(t) + {}^{\mathrm{G}}\mathbf{b}(t) + {}^{\mathrm{G}}\boldsymbol{\eta}(t) \tag{4.2} $$

where:

- ${}^{\mathrm{G}}\tilde{\boldsymbol{\omega}}$ represents the measured angular velocity in the gyroscope's coordinate frame G;
- ${}^{\mathrm{G}}\boldsymbol{\omega}$ represents the angular velocity of the gyroscope in the gyroscope's coordinate frame G;
- ${}^{\mathrm{G}}\mathbf{b}$ represents the bias of the gyroscope in the gyroscope's coordinate frame G;
- ${}^{\mathrm{G}}\boldsymbol{\eta}$ represents the noise associated with the gyroscope expressed in the gyroscope's coordinate frame G.

4.2 Sources of error in IMUs

Although the accelerometer model in Equation 4.1 and the gyroscope model in Equation 4.2 are widely used, they relies on the following assumptions:

- specific force and acceleration have a direct one-to-one relationship;
- acceleration components do not mix to produce specific force, i.e. the specific force measured by the accelerometer in each axis (X, Y, Z) is only influenced by the acceleration along that same axis, without interferences from other axis;
- the gravity vector in the accelerometer's coordinate frame A is constant and known;
- the rotation between the world frame W and the accelerometer frame A is known.

However, these assumptions do not hold due to several errors, including:

- scale-factor errors;
- shear-factor errors;
- g-sensitivity and g^2-sensitivity.

4.2.1 Scale-factor errors

The accelerometer model in Equation 4.1 establishes a direct one-to-one relationship between units of acceleration and specific force. However, this assumption may not always hold due to sensor imperfections. To account for this discrepancy, scale factors should be introduced:

$$ {}^{\mathrm{A}}\tilde{\mathbf{a}}(t) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix} ({}^{\mathrm{A}}\mathbf{a}(t) + {}^{\mathrm{A}}\mathbf{b}(t) - {}^{\mathrm{A}}\mathbf{R}_{\mathrm{W}}{}^{\mathrm{W}}\mathbf{g} + {}^{\mathrm{A}}\boldsymbol{\eta}(t)) \tag{4.3} $$

In most IMU sensors, these scale factors are factory-calibrated and provided in the sensor's datasheet.

4.2.2 Shear-factor errors

These errors arise because the three axes of the accelerometer or gyroscope are not perfectly orthogonal. As a result, acceleration along a principal direction may be detected on multiple axes, leading to cross-axis sensitivity and measurement inaccuracies.

The accelerometer and gyroscope models can be adjusted to compensate for these errors as follows:

$$ {}^{\mathrm{A}}\tilde{\mathbf{a}}(t) = \begin{bmatrix} 1 & \sigma_{xy} & \sigma_{xz} \\ 0 & 1 & \sigma_{yz} \\ 0 & 0 & 1 \end{bmatrix} ({}^{\mathrm{A}}\mathbf{a}(t) + {}^{\mathrm{A}}\mathbf{b}(t) - {}^{\mathrm{A}}\mathbf{R}_{\mathrm{W}}{}^{\mathrm{W}}\mathbf{g} + {}^{\mathrm{A}}\boldsymbol{\eta}(t)) \tag{4.4} $$

$$ {}^{\mathrm{G}}\tilde{\boldsymbol{\omega}}(t) = \begin{bmatrix} 1 & \sigma_{xy} & \sigma_{xz} \\ 0 & 1 & \sigma_{yz} \\ 0 & 0 & 1 \end{bmatrix} ({}^{\mathrm{G}}\boldsymbol{\omega}(t) + {}^{\mathrm{G}}\mathbf{b}(t) + {}^{\mathrm{G}}\boldsymbol{\eta}(t)) \tag{4.5} $$

These models account for axis misalignment errors, where the three axes of the accelerometer and gyroscope are not perfectly orthogonal. The shear coefficients σ_{xy}, σ_{xz}, and σ_{yz} represent the degree of coupling between the axes due to misalignment. These corrections assume that the Z-axis remains unchanged between the ideal orthogonal frame and the non-orthogonal frame, simplifying the compensation model.

Similar to scale factors, shear coefficients are typically factory-calibrated and included in the sensor's datasheet for most commercial IMU sensors.

4.2.3 g-sensitivity and g^2-sensitivity

Ideally, a gyroscope should measure only angular velocity. However, in MEMS gyroscopes, the inertia of the proof mass causes specific forces (accelerations) to introduce false angular velocity readings. This happen because MEMS gyroscopes rely on the Coriolis effect, as discussed in Section 4.1.2. However, the proof mass itself has inertia, meaning it resists changes in motion when subjected to acceleration. This resistance to acceleration appears as a force on the sensor, which is mistakenly interpreted as part of the Coriolis force, leading to a false angular velocity reading.

The g^2-sensitivity represents a nonlinear (quadratic) extension of the g-sensitivity, where a change in specific force induce even greater errors in the measured angular velocity. In high-vibration environments, this effect becomes a major source of drift.

In high-quality IMUs with low *bias instability*, g-sensitivity and g^2-sensitivity are ones of the most critical sources of error. While it can be factory-calibrated, in low-cost MEMS IMUs, it is typically poorly compensated, leading to degraded performance in dynamic conditions.

4.3 IMU Error Modeling

As an electronic circuit, an Inertial Measurement Unit is inherently affected by electronic noise, which introduces stochastic errors that impact the accuracy of its readings. To effectively analyse and compensate for these errors, one of the most widely used techniques is Power Spectral Density (PSD), which describes how the power of a signal is distributed across different frequencies.

The PSD of the noise, denoted as $S_\eta(f)$, decreases significantly with frequency. To model this behaviour, it is commonly approximated using a truncated series representation:

$$S_\eta(f) = N^2 + \frac{B^2}{2\pi f} + \frac{K^2}{(2\pi f)^2}. \tag{4.6}$$

Assuming that the noise components at different frequencies originate from statistically independent sources, the total noise signal can be expressed as the sum of these individual contributions:

$$\eta(t) = \eta_N(t) + \eta_K(t) + \eta_B(t). \tag{4.7}$$

where:

- $\eta_N(t)$ represents the *white noise* component;
- $\eta_K(t)$ represents the *brown noise* (or red noise) component;
- $\eta_B(t)$ represents the *pink noise* term.

4.3.1 Noise density

The white noise term $\eta_N(t)$ is referred to as *noise density*, which quantifies how noisy a sensor is, independent of its sampling rate. By multiplying the noise density by the square root of the sampling rate, the standard deviation of the noise at that rate can be determined.

Angular Random Walk and Velocity Random Walk

When noise density is integrated over time it results in a *random walk effect*, causing small, random deviations that lead to drift in the IMU's outputs.

- For gyroscopes, this phenomenon is known as Angular Random Walk (ARW). It describes how the noise in angular velocity measurements accumulates over time, leading to increasing uncertainty in angular orientation.
- For accelerometers, this effect is referred to as Velocity Random Walk (VRW). It represents how the noise in acceleration measurements leads to progressive drift in velocity estimates over time.

4.3.2 Bias term

Both the accelerometer and the gyroscope sensors are affected by a *bias* term, which represents a systematic offset in the output readings with respect to the true input values. However, this bias is not perfectly constant and is influenced by stochastic variations over time.

Rate Random Walk

One of the primary noise sources affecting bias is *brown noise*, represented by the term $\eta_K(t)$. When integrated over time, this noise results in a *random walk* effect, meaning that small, unpredictable variations accumulate gradually. This effect is known as Rate Random Walk (RRW) for both gyroscopes and accelerometers.

RRW introduces slow, random variations in bias, leading to long-term drift in accelerometer and gyroscope measurements. The evolution of bias in an IMU can be modeled as a stochastic differential equation driven by a brown noise term:

$$\dot{\mathbf{b}}(t) = \eta_K(t) \tag{4.8}$$

where:

- $\mathbf{b}(t)$ represents the evolving bias over time.
- $\eta_K(t)$ is the brown noise term, which follows a $1/f^2$ power spectral density, causing low-frequency bias variations.

Bias Instability

The pink noise term $\eta_B(t)$, known as Bias Instability (BI), quantifies how the sensor's bias drifts over time during operation at a constant temperature. This parameter defines the lowest possible bias uncertainty, representing the fundamental noise floor of an IMU.

4.3.3 Allan Variance test

The Allan Variance (AVAR) is a time-domain analysis technique originally designed for characterizing noise and stability in clock systems [1]. To determine the characteristics of the underlying noise process, the Allan Deviation (AD) is computed as:

$$AD = \sqrt{AVAR(t)} \tag{4.9}$$

where t is the averaging time. The Allan Deviation is typically plotted as a function of t on a log-log scale, where different types of stochastic processes appear as regions with distinct slopes.

From the Allan Deviation plot, it is possible to extract the noise parameters of an IMU by identifying the following slopes:

- White noise $\eta_N(t)$ (Noise Density) appears as a slope of $-1/2$ on the AD plot. The ARW and VRW values can be obtained by fitting a line through this region and reading its value at $t = 1$.
- Brown noise $\eta_K(t)$ (RRW) appears as a slope of $+1/2$ on the AD plot. The numerical value can be obtained by evaluating the AD plot at $t = 3$.
- Pink noise $\eta_B(t)$ (BI) appears as a flat region near the minimum of the AD curve. The numerical value of Bias Instability is given by the minimum value on the Allan Deviation curve.

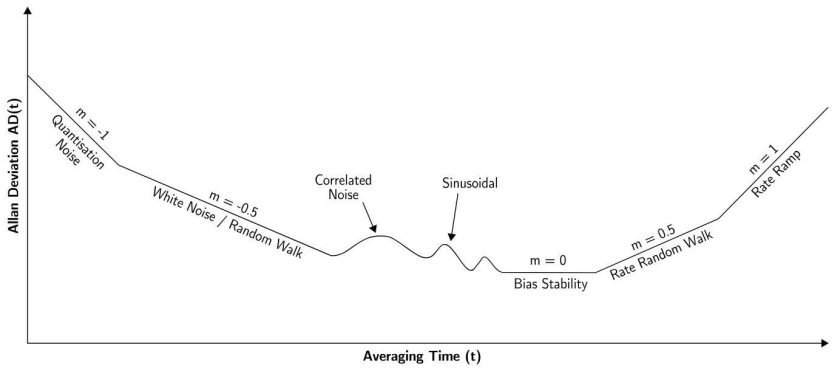


Figure 4.3. *Image source*: [26] - Example of log-log scale plot of Allan Deviation. This plot illustrates the different slope regions corresponding to various types of IMU noise.

4.4 IMU Kinematic Model and Motion Integration

For navigation applications, the objective is to recover the pose of the robot's body with respect to the world frame. This pose is represented using a homogeneous transformation matrix:

$$ {}^{\mathrm{W}}\mathbf{T}_{\mathrm{B}} = \begin{bmatrix} {}^{\mathrm{W}}\mathbf{R}_{\mathrm{B}} & {}^{\mathrm{W}}\mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{4.10} $$

where:

- ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{B}}$ is the rotation matrix, representing the orientation of the body's coordinate frame B in the world coordinate frame W.
- ${}^{\mathrm{W}}\mathbf{t}$ is the translation vector, representing the position of the body's coordinate frame B in the world coordinate frame W.

Consider a robot is equipped with a 6 degrees of freedom IMU, that includes an accelerometer and a gyroscope. The IMU frame I is assumed to be coincident with the robot's body frame B, i.e. I = B.

4.4.1 IMU Measurements

The IMU measures a specific force, given by:

$$\begin{aligned} {}^{\mathrm{I}}\tilde{\mathbf{a}}(t) &= {}^{\mathrm{I}}\mathbf{a}(t) + {}^{\mathrm{I}}\mathbf{b}_a(t) - {}^{\mathrm{I}}\mathbf{R}_{\mathrm{W}}\,{}^{\mathrm{W}}\mathbf{g} + {}^{\mathrm{I}}\boldsymbol{\eta}_a(t) \\ &= {}^{\mathrm{I}}\mathbf{R}_{\mathrm{W}}\left({}^{\mathrm{W}}\mathbf{a}(t) - {}^{\mathrm{W}}\mathbf{g}\right) + {}^{\mathrm{I}}\mathbf{b}_a(t) + {}^{\mathrm{I}}\boldsymbol{\eta}_a(t) \end{aligned} \tag{4.11}$$

where:

- ${}^{\mathrm{I}}\tilde{\mathbf{a}}$ is the measured specific force in frame I;
- ${}^{\mathrm{I}}\mathbf{a}$ is the acceleration of the IMU expressed in frame I;
- ${}^{\mathrm{W}}\mathbf{a}$ is the acceleration of the IMU expressed in frame W;
- ${}^{\mathrm{I}}\mathbf{b}_a$ is the bias of the accelerometer expressed in frame I;
- ${}^{\mathrm{I}}\mathbf{R}_{\mathrm{W}}$ is the rotation of the frame W in frame I;
- ${}^{\mathrm{W}}\mathbf{g}$ is the gravity vector vector expressed in frame W;
- ${}^{\mathrm{I}}\boldsymbol{\eta}_a$ is the noise of the accelerometer in the frame I.

and an angular velocity:

$${}^{\mathrm{I}}\tilde{\boldsymbol{\omega}}(t) = {}^{\mathrm{I}}\boldsymbol{\omega}(t) + {}^{\mathrm{I}}\mathbf{b}_g(t) + {}^{\mathrm{I}}\boldsymbol{\eta}_g(t) \tag{4.12}$$

where:

- ${}^{\mathrm{I}}\tilde{\boldsymbol{\omega}}$ is the measured angular velocity in frame I;
- ${}^{\mathrm{I}}\boldsymbol{\omega}$ is the angular velocity of the IMU expressed in frame I;
- ${}^{\mathrm{I}}\mathbf{b}_g$ is the bias of the gyroscope expressed in frame I;
- ${}^{\mathrm{I}}\boldsymbol{\eta}_g(t)$ is the noise of the gyroscope in frame I.

The pose of the IMU in the world frame W is represented by the homogeneous transformation

$${}^{\mathrm{W}}\mathbf{T}_{\mathrm{I}} = \begin{bmatrix} {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}} & {}^{\mathrm{W}}\mathbf{t} \\ 0 & 1 \end{bmatrix} \tag{4.13}$$

4.4.2 Motion Integration

The objective is to infer the motion of the IMU frame I based on the IMU measurements ${}^{\mathrm{I}}\tilde{\mathbf{a}}(t)$ and ${}^{\mathrm{I}}\tilde{\boldsymbol{\omega}}(t)$. For this purpose, the following kinematic model is adopted, following the derivation presented in [8]:

$$\begin{aligned} {}^{\mathrm{W}}\dot{\mathbf{R}}_{\mathrm{I}} &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}\lfloor {}^{\mathrm{I}}\boldsymbol{\omega}\times\rfloor \\ {}^{\mathrm{W}}\dot{\mathbf{v}} &= {}^{\mathrm{W}}\mathbf{a} \\ {}^{\mathrm{W}}\dot{\mathbf{p}} &= {}^{\mathrm{W}}\mathbf{v} \end{aligned} \tag{4.14}$$

which describe the evolution of the pose and velocity of the IMU frame I.

The state at time $t + \Delta t$ is obtained by integrating Equation 4.14:

$$\begin{aligned} {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}(t+\Delta t) &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}(t)\,\mathrm{Exp}\left(\int_t^{t+\Delta t} {}^{\mathrm{I}}\boldsymbol{\omega}(\tau)d\tau\right) \\ {}^{\mathrm{W}}\mathbf{v}(t+\Delta t) &= {}^{\mathrm{W}}\mathbf{v}(t) + \int_t^{t+\Delta t} {}^{\mathrm{W}}\mathbf{a}(\tau)d\tau \\ {}^{\mathrm{W}}\mathbf{p}(t+\Delta t) &= {}^{\mathrm{W}}\mathbf{p}(t) + \int_t^{t+\Delta t} {}^{\mathrm{W}}\mathbf{v}(\tau)d\tau + \iint_t^{t+\Delta t} {}^{\mathrm{W}}\mathbf{a}(\tau)d\tau^2 \end{aligned} \tag{4.15}$$

Assuming that ${}^{\mathrm{W}}\mathbf{a}(t)$ and ${}^{\mathrm{I}}\boldsymbol{\omega}(t)$ remain constant in the time interval $[\,t, t+\Delta t]$, is it possible to write the discretize version of Equation 4.15:

$$\begin{aligned} {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}(t+\Delta t) &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}(t)\,\mathrm{Exp}\left({}^{\mathrm{I}}\boldsymbol{\omega}(t)\Delta t\right) \\ {}^{\mathrm{W}}\mathbf{v}(t+\Delta t) &= {}^{\mathrm{W}}\mathbf{v}(t) + {}^{\mathrm{W}}\mathbf{a}(t)\Delta t \\ {}^{\mathrm{W}}\mathbf{p}(t+\Delta t) &= {}^{\mathrm{W}}\mathbf{p}(t) + {}^{\mathrm{W}}\mathbf{v}(t)\Delta t + \frac{1}{2}{}^{\mathrm{W}}\mathbf{a}(t)\Delta t^2 \end{aligned} \tag{4.16}$$

where ${}^{\mathrm{W}}\mathbf{a}(t)$ and ${}^{\mathrm{I}}\boldsymbol{\omega}(t)$ can be derived from Equation 4.11 and 4.12 respectively:

$${}^{\mathrm{W}}\mathbf{a}(t) = {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}(t)\left({}^{\mathrm{I}}\tilde{\mathbf{a}}(t) - {}^{\mathrm{I}}\mathbf{b}_a(t) - {}^{\mathrm{I}}\boldsymbol{\eta}_{ad}(t)\right) + {}^{\mathrm{W}}\mathbf{g} \tag{4.17}$$

$${}^{\mathrm{I}}\boldsymbol{\omega}(t) = {}^{\mathrm{I}}\tilde{\boldsymbol{\omega}}(t) - {}^{\mathrm{I}}\mathbf{b}_g(t) - {}^{\mathrm{I}}\boldsymbol{\eta}_{gd}(t) \tag{4.18}$$

4.5 IMU Initialization

As discussed in Section 4.4.2, correctly integrating the acceleration and angular velocity to reconstruct the IMU pose requires knowledge of the orientation of the IMU frame I with respect to the world frame W, i.e., the rotation matrix ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}$. This matrix enables the transformation of the local specific force, measured by the accelerometer, into the global specific force. This transformation allows for the removal of gravity from the IMU accelerometer measurement, ${}^{\mathrm{W}}\tilde{\mathbf{a}}$, to obtain the true acceleration of the IMU frame, ${}^{\mathrm{W}}\mathbf{a}$.

During motion, this rotation is continuously updated using angular velocity measurements, as described in Section 4.4.2. However, at startup, the initial orientation must be determined to ensure proper motion integration.

4.5.1 Rotation Initialization

The simplest method to initializing the orientation ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}$ is by leveraging the global gravity vector ${}^{\mathrm{W}}\mathbf{g} = \begin{bmatrix} 0 & 0 & -9.81 \end{bmatrix}^{\mathrm{T}}$ m/s². Assuming the IMU remains completely stationary during initialization, a set of measurements can be collected to determine the local gravity vector as measured by the accelerometer, denoted as ${}^{\mathrm{I}}\mathbf{g}$.

Consider that N accelerometer measurements are recorded in a stationary setup, the mean acceleration measurement is computed as:

$$ {}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{avg}} = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} {}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{i}} \tag{4.19} $$

Using this, the normalized local gravity vector sensed by the accelerometer is given by:

$$ {}^{\mathrm{I}}\mathbf{g}_{|||} = \frac{{}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{avg}}}{\left\| {}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{avg}} \right\|} \tag{4.20} $$

Similarly, the normalized global gravity vector is defined as:

$$ {}^{\mathrm{W}}\mathbf{g}_{|||} = \frac{{}^{\mathrm{W}}\mathbf{g}}{\| {}^{\mathrm{W}}\mathbf{g} \|} \tag{4.21} $$

From ${}^{\mathrm{I}}\mathbf{g}_{|||}$ and ${}^{\mathrm{W}}\mathbf{g}_{|||}$ the rotation matrix ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}$ can be computed as:

$$ \begin{aligned} \zeta &= \frac{{}^{\mathrm{I}}\mathbf{g}_{|||} \times {}^{\mathrm{W}}\mathbf{g}_{|||}}{\left\| {}^{\mathrm{I}}\mathbf{g}_{|||} \times {}^{\mathrm{W}}\mathbf{g}_{|||} \right\|} \\ \theta &= \arccos\left({}^{\mathrm{I}}\mathbf{g}_{|||}^{\mathrm{T}} \cdot {}^{\mathrm{W}}\mathbf{g}_{|||} \right) \\ {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}} &= \mathbf{I}_{3\times 3} + \sin(\theta) \lfloor \boldsymbol{\zeta} \times \rfloor + (1 - \cos(\theta)) \lfloor \boldsymbol{\zeta} \times \rfloor^2 \end{aligned} \tag{4.22} $$

This initialization procedure assumes that the accelerometer has no bias or that the bias is negligible. Otherwise, the mixing of gravity and bias would prevent accurate initialization with this algorithm.

4.5.2 Bias Initialization

Although the IMU will be used within a Kalman Filter (KF) framework, where the bias is estimated online, it can be beneficial to provide an initial estimate of the bias, if possible, to improve accuracy.

Accelerometer Bias

As discussed in Section 4.5.1, the mixing of the local gravity vector with accelerometer bias prevents the accurate estimation of the IMU orientation ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}$. For the same reason, it is not possible to estimate the accelerometer bias if the orientation ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}$ also needs to be initialized. Therefore, to estimate the accelerometer bias, it must be assumed that the IMU orientation ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}$ is already known and does not require initial estimation.

With these assumptions the accelerometer bias can be estimated as follow:

$$ \begin{aligned} {}^{\mathrm{I}}\mathbf{g} &= {}^{\mathrm{I}}\mathbf{R}_{\mathrm{W}} {}^{\mathrm{W}}\mathbf{g} \\ {}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{avg}} &= \frac{1}{N}\sum_{\mathrm{i}=1}^{N} {}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{i}} \\ {}^{\mathrm{I}}\mathbf{b}_{a} &= {}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{avg}} - {}^{\mathrm{I}}\mathbf{g} \end{aligned} \tag{4.23} $$

where ${}^{\mathrm{I}}\tilde{\mathbf{a}}_{\mathrm{avg}}$ is computed from N accelerometer measurements recorded in a stationary setup.

Gyroscope Bias

The gyroscope bias can be initialized by simply averaging N angular velocity measurements recorded in a stationary setup:

$$ {}^{\mathrm{I}}\mathbf{b}_g = {}^{\mathrm{I}}\tilde{\boldsymbol{\omega}}_{\mathrm{avg}} = \frac{1}{N}\sum_{\mathrm{i}=1}^{N} {}^{\mathrm{I}}\tilde{\boldsymbol{\omega}}_{\mathrm{i}} \tag{4.24}$$

Chapter 5

Bayes Filters

In probability theory, *recursive Bayesian estimation*, also known as *Bayes filter*, is a fundamental probabilistic method for estimating an unknown Probability Density Function (PDF) over time. This approach relies on continuously refining the probability distribution using sequential measurements and a mathematical process model.

In robotics, Bayes filters are used as a framework for state estimation, particularly in sensor fusion localization. They allow a robot to determine its position and orientation by maintaining a set of probabilistic beliefs that are dynamically updated as new sensor data becomes available. The estimation process follows two key stages:

- *prediction*: the system estimates its future state by propagating the current state through a motion model, leveraging prior knowledge of the system's dynamics;
- *correction*: the predicted state is updated based on incoming sensor measurements, incorporating environmental information and constraints.

Through its recursive prediction-correction cycle, the Bayes filter enables state estimation by integrating data from multiple sensors, a process known as *sensor fusion*.

The prediction phase usually relies on *proprioceptive* sensors, which provide information about the robot's egomotion. Commonly used sensors in this category include IMUs and wheel encoders, as they offer informations of the robot's internal state and movement dynamics.

Vice-versa, the correction phase usually depends on *exteroceptive* sensors, which capture information about the surrounding environment. By leveraging this data, the robot can establish constraints and refine its estimated pose. Among the most widely used exteroceptive sensors are monocular cameras, stereocameras, and LiDARs.

5.1 Mathematical formulation of Bayes Filter

Consider a sequence of time-indexed sensor observations $\mathbf{z}_1, ..., \mathbf{z}_t$, obtained from an exteroceptive sensor, and control commands $\mathbf{u}_1, ..., \mathbf{u}_t$, provided by a proprioceptive sensor. The *belief* at time t is defined as the posterior probability distribution of the random variable $\mathbf{x}_t$, given all available sensor measurements and control inputs up to time t:

$$bel\left(\mathbf{x}_t\right) = p\left(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) \tag{5.1}$$

By applying Bayes' theorem, the belief can be formulated as:

$$bel\left(\mathbf{x}_t\right) = p\left(\mathbf{x}_t | \mathbf{z}_{1:t}, \mathbf{u}_{1:t}\right) = \eta\ p\left(\mathbf{z}_t | \mathbf{x}_t, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right)\ p\left(\mathbf{x}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) \tag{5.2}$$

In general, computing this posterior distribution becomes increasingly complex over time due to the growing number of sensor measurements. To make the computation feasible, Bayes filters assume the *Markov property*, which states that the sensor measurements at time t depend only on the robot's current state, and that the state $\mathbf{x}_t$ is conditioned exclusively on the previous state $\mathbf{x}_{t-1}$. Consequently, past states before $\mathbf{x}_{t-1}$ do not contribute with additional information.

Under this assumption, Equation 5.2 simplifies to:

$$bel\left(\mathbf{x}_t\right) = \eta\ p\left(\mathbf{z}_t | \mathbf{x}_t\right)\ p\left(\mathbf{x}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right) \tag{5.3}$$

Applying the law of total probability to $p\left(\mathbf{x}_t | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right)$, Equation 5.3 can be expressed as:

$$bel\left(\mathbf{x}_t\right) = \eta\ p\left(\mathbf{z}_t | \mathbf{x}_t\right) \int p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right)\ p\left(\mathbf{x}_{t-1} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right)\ d\mathbf{x}_{t-1} \tag{5.4}$$

By applying the Markov assumption to $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t})$ and assuming that control inputs affect only the transition between consecutive states (independence assumption on $p\left(\mathbf{x}_{t-1} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t}\right)$), the belief can be reformulated as:

$$bel\left(\mathbf{x}_t\right) = \eta\ p\left(\mathbf{z}_t | \mathbf{x}_t\right) \int p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) p\left(\mathbf{x}_{t-1} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}\right)\ d\mathbf{x}_{t-1} \tag{5.5}$$

Noting that, in Equation 5.5:

$$p\left(\mathbf{x}_{t-1} | \mathbf{z}_{1:t-1}, \mathbf{u}_{1:t-1}\right) = bel\left(\mathbf{x}_{t-1}\right) \tag{5.6}$$

can be introduced the recursive term of the Bayes filter, obtaining the following formulation of the belief:

$$bel\left(\mathbf{x}_t\right) = \eta\ p\left(\mathbf{z}_t | \mathbf{x}_t\right) \int p\left(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t\right) bel\left(\mathbf{x}_{t-1}\right)\ d\mathbf{x}_{t-1} \tag{5.7}$$

The Equation 5.7, can be decomposed into the two fundamental steps of the recursive Bayes filter: *prediction* and *correction*:

- The prediction step estimates the current belief $\overline{bel}(\mathbf{x}_t)$ based exclusively on the previous state $\mathbf{x}_{t-1}$ and the control input $\mathbf{u}_t$:

$$\overline{bel}\left(\mathbf{x}_t\right) = \int p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t\right) bel\left(\mathbf{x}_{t-1}\right) \; d\mathbf{x}_{t-1} \tag{5.8}$$

- The correction step adjust the predicted belief $\overline{bel}(\mathbf{x}_t)$ by incorporating the latest measurement $\mathbf{z}_t$ obtained at the current state $\mathbf{x}_t$:

$$bel\left(\mathbf{x}_t\right) = \eta \; p\left(\mathbf{z}_t|\mathbf{x}_t\right) \overline{bel}\left(\mathbf{x}_t\right) \tag{5.9}$$

The prediction step is defined by the *motion model*, while the correction step corresponds to the *measurement model*. Both models are dependent on the sensors used to obtain proprioceptive and exteroceptive observations. A more detailed derivation of recursive Bayes filter is provided in [24].

The recursive Bayes filter serves as a foundational framework for various practical implementations, including the Kalman Filter (KF) and Extended Kalman Filter (EKF).

5.2 Kalman Filter

The Kalman Filter (KF) is an optimal state estimator for systems that satisfy the *linear Gaussian* assumption. At each time step t, it represents the belief using a mean vector $\boldsymbol{\mu}_t$ and a covariance matrix $\boldsymbol{\Sigma}_t$. As well explained in [24], the posterior distribution remains Gaussian if the following conditions hold in addition to the Markov assumption of the Bayes filter:

1. the state transition model is linear;
2. the measurement model is linear;
3. the initial belief follows a normal distribution.

1. Linearity of the State Transition Model

The state transition probability $p\left(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t\right)$ must be a *linear* function of the previous state $\mathbf{x}_{t-1}$ and control input $\mathbf{u}_t$, with added Gaussian noise. This is mathematically expressed as:

$$\mathbf{x}_t = \mathbf{A}_t\mathbf{x}_{t-1} + \mathbf{B}_t\mathbf{u}_t + \boldsymbol{\epsilon}_t \tag{5.10}$$

where the variables are defined as follows:

- $\mathbf{x}_t \in \mathbb{R}^{n\times 1}$: state vector at time t,
- $\mathbf{u}_t \in \mathbb{R}^{m\times 1}$: control input vector at time t,

- $\boldsymbol{\epsilon}_t \in \mathbb{R}^{n \times 1}$: Gaussian noise term modeling the uncertainty in state transitions,
- $\mathbf{A}_t \in \mathbb{R}^{n \times n}$: state transition matrix,
- $\mathbf{B}_t \in \mathbb{R}^{n \times m}$: control input matrix.

The noise term $\boldsymbol{\epsilon}_t$ is assumed to follow a Gaussian distribution with zero mean and covariance $\mathbf{R}_t \in \mathbb{R}^{n \times n}$.

Equation 5.10 defines the state transition probability $p(\mathbf{x}_t | \mathbf{x}_{t-1}, \mathbf{u}_t)$. This probability distribution can be explicitly derived by substituting Equation 5.10 into the multivariate normal distribution formula:

$$p(\mathbf{x_t}) = \det(2\pi\boldsymbol{\Sigma}_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}_t - \boldsymbol{\mu}_t)^T \boldsymbol{\Sigma}_t^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_t)\right) \tag{5.11}$$

where:

- the mean is given by $\boldsymbol{\mu}_t = \mathbf{A}_t \mathbf{x}_{t-1} + \mathbf{B}_t \mathbf{u}_t$;
- the covariance matrix is $\boldsymbol{\Sigma}_t = \mathbf{R}_t$.

2. Linearity of the Measurement Model

The measurement probability $p(\mathbf{z}_t | \mathbf{x}_t)$ must be a *linear* function of the state $\mathbf{x}_t$, with added Gaussian noise. This relationship is expressed as:

$$\mathbf{z}_t = \mathbf{C}_t \mathbf{x}_t + \boldsymbol{\delta}_t \tag{5.12}$$

where the variables are defined as follows:

- $\mathbf{z}_t \in \mathbb{R}^{k \times 1}$: measurement vector at time t,
- $\mathbf{x}_t \in \mathbb{R}^{n \times 1}$: state vector at time t,
- $\boldsymbol{\delta}_t \in \mathbb{R}^{k \times 1}$: Gaussian noise term modeling measurement uncertainty,
- $\mathbf{C}_t \in \mathbb{R}^{k \times n}$: measurement matrix.

The noise term $\boldsymbol{\delta}_t$ is assumed to follow a Gaussian distribution with zero mean and covariance $\mathbf{Q}_t$. Substituting Equation 5.12 into the multivariate normal distribution formula, the measurement probability can be written as:

$$p(\mathbf{z}_t | \mathbf{x}_t) = \det(2\pi\boldsymbol{\Sigma}_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{z}_t - \boldsymbol{\mu}_t)^T \boldsymbol{\Sigma}_t^{-1}(\mathbf{z}_t - \boldsymbol{\mu}_t)\right) \tag{5.13}$$

where:

- the mean is given by $\boldsymbol{\mu}_t = \mathbf{C}_t \mathbf{x}_t$;
- the covariance matrix is $\boldsymbol{\Sigma}_t = \mathbf{Q}_t$.

3. Gaussianity of the Initial Belief

The initial belief $bel(\mathbf{x}_0)$ must follow a normal distribution. Letting $\boldsymbol{\mu}_0$ and $\boldsymbol{\Sigma}_0$ represent the mean and covariance of $bel(\mathbf{x}_0)$, the initial belief can be expressed as a multivariate normal distribution:

$$p(\mathbf{x}_0) = \det(2\pi\boldsymbol{\Sigma}_0)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}_0 - \boldsymbol{\mu}_0)^{\mathrm{T}} \boldsymbol{\Sigma}_0^{-1}(\mathbf{x}_0 - \boldsymbol{\mu}_0)\right) \tag{5.14}$$

These three assumptions guarantee that the posterior belief $bel(\mathbf{x}_t)$ remains Gaussian at all time steps t.

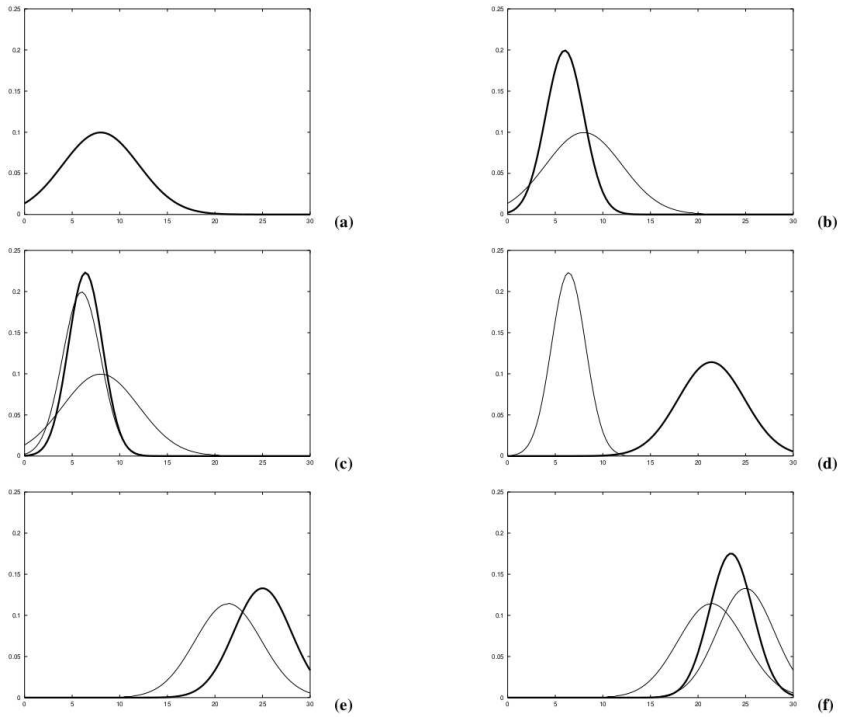


Figure 5.1. *Image source*: [24] - Visualization of the Kalman Filter's belief propagation and update process. (a) The initial belief, represented as a Gaussian distribution. (b) The measurement probability distribution $p(\mathbf{z}_t|\mathbf{x}_t)$ (bold) representing the likelihood of the observed measurement given the state. (c) The updated belief (bold) after incorporating the measurement, resulting in a refined Gaussian distribution. (d) The belief after the motion step (prediction), shifting the distribution to the right and increasing uncertainty. (e) The new measurement probability distribution. (f) The corrected belief (bold) after integrating the new measurement, reducing uncertainty and refining the state estimate.

5.2.1 Kalman Filter algorithm

The Kalman Filter algorithm is outlined in Algorithm 2. This represents a single iteration of the recursive estimation process, which takes as input the previous

belief $bel\left(\mathbf{x}_{t-1}\right)$, characterized by the mean $\boldsymbol{\mu}_{t-1}$ and covariance $\boldsymbol{\Sigma}_{t-1}$, along with the current control input $\mathbf{u}_t$ and the latest measurement $\mathbf{z}_t$.

The prediction step, performed in lines 1 and 2, computes the predicted belief $\overline{bel}(\mathbf{x}_t)$ based on the system dynamics and control input. The correction step, executed in lines 3 to 5, updates this belief by incorporating the measurement $\mathbf{z}_t$. A key component of this step is the computation of the *Kalman Gain* $\mathbf{K}_t \in \mathbb{R}^{n \times k}$, which determines the extent to which the measurement influences the updated state estimate.

Algorithm 2 Kalman Filter Algorithm

Require: $\boldsymbol{\mu}_{t-1}$, $\boldsymbol{\Sigma}_{t-1}$, $\mathbf{u}_t$, $\mathbf{z}_t$
Ensure: $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

1: $\bar{\boldsymbol{\mu}}_t = \mathbf{A}_t \boldsymbol{\mu}_{t-1} + \mathbf{B}_t \mathbf{u}_t$
2: $\bar{\boldsymbol{\Sigma}}_t = \mathbf{A}_t \boldsymbol{\Sigma}_{t-1} \mathbf{A}_t^{\mathrm{T}} + \mathbf{R}_t$
3: $\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^{\mathrm{T}} \left(\mathbf{C}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{C}_t^{\mathrm{T}} + \mathbf{Q}_t \right)^{-1}$
4: $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t \left(\mathbf{z}_t - \mathbf{C}_t \bar{\boldsymbol{\mu}}_t \right)$
5: $\boldsymbol{\Sigma}_t = \left(\mathbf{I} - \mathbf{K}_t \mathbf{C}_t \right) \bar{\boldsymbol{\Sigma}}_t$

5.3 Extended Kalman Filter

The Kalman Filter relies on the assumption that both the state transition and measurement models are linear functions of the state. A key property enabling its derivation is that any linear transformation of a Gaussian random variable results in another Gaussian distribution. This property allows the Kalman Filter to efficiently compute the parameters of the resulting Gaussian in closed form.

Figure 5.2(a) illustrates the effect of a *linear transformation* on a Gaussian random variable. Given a random variable $\mathbf{X} \sim \mathcal{N}\left(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2\right)$ (bottom right), applying a linear function $\mathbf{y} = a\mathbf{x} + b$ (top right) results in a transformed random variable $\mathbf{Y} \sim \mathcal{N}\left(\mathbf{y}; a\boldsymbol{\mu} + b, a^2\boldsymbol{\sigma}^2\right)$ (top left). This transformation preserves the Gaussian nature of the distribution, which is a fundamental characteristic exploited in the update step of the Kalman Filter.

The Extended Kalman Filter (EKF) overcomes the *linearity assumption* of the Kalman Filter, which is rarely satisfied in real-world applications. The EKF extends the applicability of the KF to systems with nonlinear state transitions and measurement models. As a result, these processes are described by the nonlinear functions g and h, respectively:

$$\begin{aligned} \mathbf{x}_t &= g\left(\mathbf{x}_{t-1}, \mathbf{u}_t\right) + \boldsymbol{\epsilon}_t \\ \mathbf{z}_t &= h\left(\mathbf{x}_t\right) + \boldsymbol{\delta}_t \end{aligned} \tag{5.15}$$

This formulation generalizes the linear Gaussian model of the Kalman Filter. Specifically:

- the function g replaces the state transition matrices $\mathbf{A}_t$ and $\mathbf{B}_t$ in Equation 5.10,
- the function h replaces the measurement matrix $\mathbf{C}_t$ in Equation 5.12.

However, the introduction of nonlinear functions g and h means that the posterior belief is no longer guaranteed to be Gaussian, as illustrated in Figure 5.2(b). To address this, the Extended Kalman Filter approximates the true belief using a Gaussian distribution. As in the Kalman Filter, the belief $bel(\mathbf{x}_t)$ at time t is represented by a mean $\boldsymbol{\mu}_t$ and a covariance $\boldsymbol{\Sigma}_t$. While the EKF retains the same belief representation as the standard Kalman Filter, it provides only an approximate solution rather than an exact one.

5.3.1 Linearization via Taylor Expansion

The key idea behind the EKF is to *linearize* the nonlinear functions g and h at the Gaussian's mean. By replacing g and h with their tangent linearizations around this mean, projecting the Gaussian through those linear functions again yields a Gaussian distribution, although with an approximation error. Figure 5.3 illustrates this concept: the Gaussian variable $\mathbf{X} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ (bottom-right) is mapped by the linearization of g at μ (dashed line in the top-right), producing a Gaussian variable $\mathbf{Y} \sim \mathcal{N}(\mathbf{y}; a\boldsymbol{\mu} + b, a^2\boldsymbol{\sigma}^2)$ (dashed line in the top-left). The difference between this approximate Gaussian (dashed line) and the true Gaussian (solid line) is the *linearization error*.

The linearization process is carried out using the *first-order Taylor expansion*, which provides a local linear approximation of the nonlinear functions g and h.

The slope of the function g is determined by its partial derivative:

$$g'(\mathbf{x}_{t-1}, \mathbf{u}_t) \coloneqq \frac{\partial g(\mathbf{x}_{t-1}, \mathbf{u}_t)}{\partial \mathbf{x}_{t-1}} = \mathbf{G}_t \in \mathbb{R}^{n \times n} \tag{5.16}$$

which corresponds to the *Jacobian* matrix of the state transition model. Similarly, the slope of the function h is given by:

$$h'(\mathbf{x}_t) \coloneqq \frac{\partial h(\mathbf{x}_t)}{\partial \mathbf{x}_t} = \mathbf{H}_t \in \mathbb{R}^{n \times k} \tag{5.17}$$

which defines the *Jacobian* matrix of the measurement model.

Since both functions depend on their evaluation points, a natural choice is to linearize around the most probable state at that time. For Gaussian distributions, this corresponds to the mean of the posterior: $\boldsymbol{\mu}_{t-1}$ for g and $\bar{\boldsymbol{\mu}}_t$ for h.

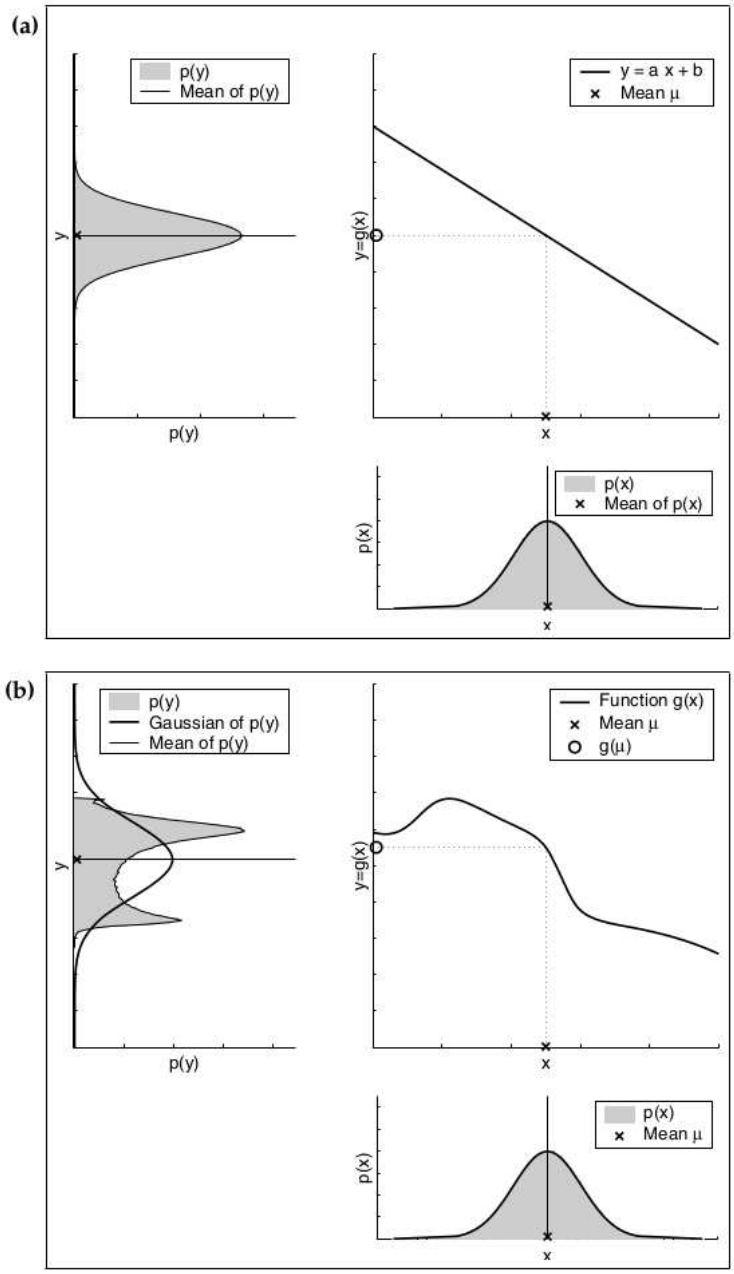


Figure 5.2. *Image source*: [24] - Illustration of the projections of a random variable $\mathbf{X} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ through (a) a linear function $\mathbf{y} = a\mathbf{x} + b$ and (b) a nonlinear function. In (a) the linear mapping yields a Gaussian random variable $\mathbf{Y} \sim \mathcal{N}(\mathbf{y}; a\boldsymbol{\mu} + b, a^2\boldsymbol{\sigma}^2)$. In (b), the nonlinear mapping no longer produces a Gaussian distribution.

Thus, the function g is approximated at $\boldsymbol{\mu}_{t-1}$ and $\mathbf{u}_t$ as:

$$\begin{aligned} g(\mathbf{x}_{t-1}, \mathbf{u}_t) &\approx g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + g'(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t)(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \\ &= g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + \mathbf{G}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1}) \end{aligned} \tag{5.18}$$

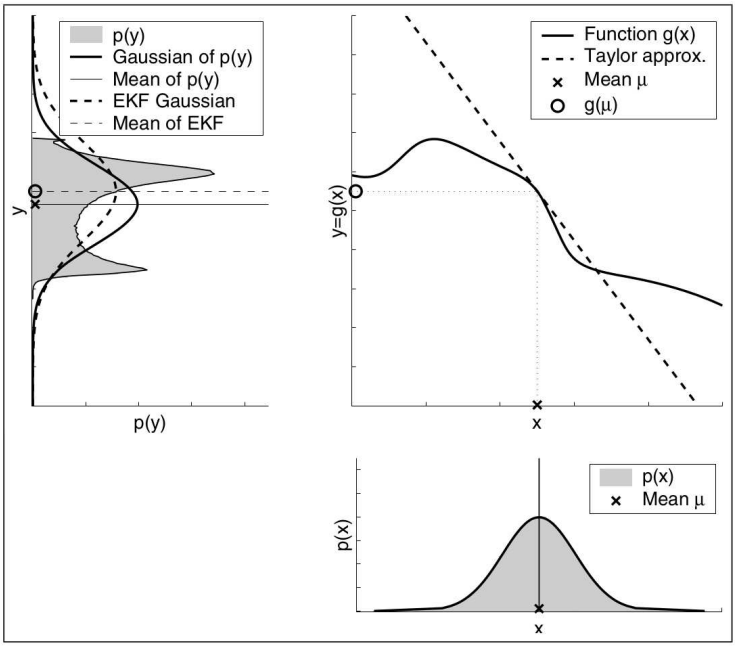


Figure 5.3. *Image source*: [24] - Illustration of the projections of a random variable $\mathbf{X} \sim \mathcal{N}(\mathbf{x}; \boldsymbol{\mu}, \boldsymbol{\sigma}^2)$ (bottom-right) through a linear approximation of a nonlinear function g (dashed line in the top-right). The result is a Gaussian variable $\mathbf{Y} \sim \mathcal{N}(\mathbf{y}; a\boldsymbol{\mu} + b, a^2\boldsymbol{\sigma}^2)$ (dashed line in the top-left). The solid line in top-left represents the true Gaussian of $\mathbf{Y}$.

which can be expressed as a multivariate normal distribution:

$$p(\mathbf{x_t}) = \det(2\pi\boldsymbol{\Sigma}_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}_t - \boldsymbol{\mu}_t)^{\mathrm{T}}\boldsymbol{\Sigma}_t^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_t)\right) \tag{5.19}$$

where:

- the mean is given by $\boldsymbol{\mu}_t = g(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t) + \mathbf{G}_t(\mathbf{x}_{t-1} - \boldsymbol{\mu}_{t-1})$;
- the covariance matrix is $\boldsymbol{\Sigma}_t = \mathbf{R}_t$.

Similarly, the function h is approximated at $\bar{\boldsymbol{\mu}}_t$ as:

$$h(\mathbf{x}_t) \approx h(\bar{\boldsymbol{\mu}}_t) + h'(\bar{\boldsymbol{\mu}}_t)(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t) = h(\bar{\boldsymbol{\mu}}_t) + \mathbf{H}_t(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t) \tag{5.20}$$

which leads to the following multivariate normal distribution:

$$p(\mathbf{z}_t|\mathbf{x}_t) = \det(2\pi\boldsymbol{\Sigma}_t)^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{z}_t - \boldsymbol{\mu}_t)^{\mathrm{T}}\boldsymbol{\Sigma}_t^{-1}(\mathbf{z}_t - \boldsymbol{\mu}_t)\right) \tag{5.21}$$

where:

- the mean is given by $\boldsymbol{\mu}_t = h(\bar{\boldsymbol{\mu}}_t) + \mathbf{H}_t(\mathbf{x}_t - \bar{\boldsymbol{\mu}}_t)$;
- the covariance matrix is $\boldsymbol{\Sigma}_t = \mathbf{Q}_t$.

5.3.2 Extended Kalman Filter algorithm

The EKF algorithm closely follows the structure of the KF algorithm. The key differences lie in the use of the nonlinear functions g and h, along with their respective Jacobian matrices, $\mathbf{G}_t$ and $\mathbf{H}_t$, for the state transition and measurement updates.

Algorithm 3 Extended Kalman Filter Algorithm

Require: $\boldsymbol{\mu}_{t-1}$, $\boldsymbol{\Sigma}_{t-1}$, $\mathbf{u}_t$, $\mathbf{z}_t$

Ensure: $\boldsymbol{\mu}_t$, $\boldsymbol{\Sigma}_t$

1: $\bar{\boldsymbol{\mu}}_t = g\left(\boldsymbol{\mu}_{t-1}, \mathbf{u}_t\right)$

2: $\bar{\boldsymbol{\Sigma}}_t = \mathbf{G}_t \boldsymbol{\Sigma}_{t-1} \mathbf{G}_t^{\mathrm{T}} + \mathbf{R}_t$

3: $\mathbf{K}_t = \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^{\mathrm{T}} \left(\mathbf{H}_t \bar{\boldsymbol{\Sigma}}_t \mathbf{H}_t^{\mathrm{T}} + \mathbf{Q}_t\right)^{-1}$

4: $\boldsymbol{\mu}_t = \bar{\boldsymbol{\mu}}_t + \mathbf{K}_t \left(\mathbf{z}_t - h\left(\bar{\boldsymbol{\mu}}_t\right)\right)$

5: $\boldsymbol{\Sigma}_t = \left(\mathbf{I} - \mathbf{K}_t \mathbf{H}_t\right) \bar{\boldsymbol{\Sigma}}_t$

Chapter 6

Multi-State Constraint Kalman Filter

The Multi-State Constraint Kalman Filter (MSCKF) is an EKF-based algorithm developed for real-time vision-aided inertial navigation. Originally introduced by [16] in 2007, it has since been refined through multiple studies.

The MSCKF efficiently leverages localization information by incorporating multiple visual feature observations. The key insight behind this approach is that when a static feature is observed from multiple camera poses, a geometric constraint can be formulated that relates all these poses. By expressing these constraints without explicitly including the 3D feature positions in the filter state vector, the MSCKF achieves a computational complexity that remains *linear* in the number of features.

6.1 State vector

The objective of the MSCKF is to track the 3D pose of the IMU-affixed frame I relative to the world frame W. To simplify the effects of Earth's rotation on IMU measurements, the world frame W was defined as the Earth-Centred, Earth-Fixed (ECEF) frame in the original paper [16]. IMU measurements are processed in real time as they become available, propagating both the EKF state and covariance. In contrast, each time an image is captured, the current camera pose estimate, expressed as a static transformation from the current IMU pose estimate, is appended to the state vector, augmenting the state. This augmentation is essential for processing feature measurements since, during EKF updates, observations of each tracked feature impose constraints between all camera poses from which the feature was observed.

Thus, at any given moment, the EKF state vector consists of:

- the evolving IMU state,
- a history of up to M_{max} past camera poses.

6.1.1 IMU state vector

The evolving IMU state is represented by the vector:

$$\hat{\mathbf{x}}_{\text{IMU}} = \begin{bmatrix} {}^{\text{W}}\hat{\mathbf{R}}_{\text{I}} & {}^{\text{I}}\hat{\mathbf{b}}_{\text{g}}^{\text{T}} & {}^{\text{W}}\hat{\mathbf{v}}_{\text{I}}^{\text{T}} & {}^{\text{I}}\hat{\mathbf{b}}_{\text{a}}^{\text{T}} & {}^{\text{W}}\hat{\mathbf{t}}_{\text{I}}^{\text{T}} \end{bmatrix}^{\text{T}} \tag{6.1}$$

where:

- ${}^{\text{W}}\hat{\mathbf{R}}_{\text{I}}$ is the rotation matrix defining the IMU orientation in the world frame W;
- ${}^{\text{I}}\hat{\mathbf{b}}_{\text{g}}^{\text{T}}$ represents the gyroscope bias, expressed in the IMU frame I;
- ${}^{\text{W}}\hat{\mathbf{v}}_{\text{I}}^{\text{T}}$ denotes the velocity of the IMU in the world frame W;
- ${}^{\text{I}}\hat{\mathbf{b}}_{\text{a}}^{\text{T}}$ corresponds to the accelerometer bias, expressed in frame I;
- ${}^{\text{W}}\hat{\mathbf{t}}_{\text{I}}^{\text{T}}$ represents the position of the IMU in the world frame W.

Following this structure, the IMU error state is defined as:

$$\tilde{\mathbf{x}}_{\text{IMU}} = \begin{bmatrix} {}^{\text{W}}\boldsymbol{\delta\theta}_{\text{I}}^{\text{T}} & {}^{\text{I}}\tilde{\mathbf{b}}_{\text{g}}^{\text{T}} & {}^{\text{W}}\tilde{\mathbf{v}}_{\text{I}}^{\text{T}} & {}^{\text{I}}\tilde{\mathbf{b}}_{\text{a}}^{\text{T}} & {}^{\text{W}}\tilde{\mathbf{t}}_{\text{I}}^{\text{T}} \end{bmatrix}^{\text{T}} \tag{6.2}$$

where:

- ${}^{\text{W}}\boldsymbol{\delta\theta}_{\text{I}}^{\text{T}}$ represents the orientation error of the IMU about the three axes in the world frame W;
- ${}^{\text{I}}\tilde{\mathbf{b}}_{\text{g}}^{\text{T}}$ is the error in the estimated gyroscope bias;
- ${}^{\text{W}}\tilde{\mathbf{v}}_{\text{I}}^{\text{T}}$ denotes the error in the estimated velocity of the IMU;
- ${}^{\text{I}}\tilde{\mathbf{b}}_{\text{a}}^{\text{T}}$ is the error in the estimated accelerometer bias;
- ${}^{\text{W}}\tilde{\mathbf{t}}_{\text{I}}^{\text{T}}$ represents the error in the estimated position of the IMU.

It is important to note that the choice of orientation error parametrization follows the analysis by Li and Mourikis in [14], which demonstrated that defining orientation error as the difference between the true and estimated world frames is preferable to defining it based on the difference between the true and estimated IMU frames. The latter approach, originally used in the MSCKF formulation in [16], introduces undesirable terms in the observability matrix of the EKF's linearized system model, negatively affecting the filter's performance.

6.1.2 Full state vector

Assuming that M camera poses are included in the EKF state vector at time-step k, this vector has the following form:

$$\hat{\mathbf{x}}_{\text{k}} = \begin{bmatrix} \hat{\mathbf{x}}_{\text{IMU,k}}^{\text{T}} & {}^{\text{W}}\hat{\mathbf{R}}_{\text{C}_1} & {}^{\text{W}}\hat{\mathbf{t}}_{\text{C}_1}^{\text{T}} & \cdots & {}^{\text{W}}\hat{\mathbf{R}}_{\text{C}_\text{M}} & {}^{\text{W}}\hat{\mathbf{t}}_{\text{C}_\text{M}}^{\text{T}} \end{bmatrix} \tag{6.3}$$

where:

- $\hat{\mathbf{x}}^{\mathrm{T}}_{\mathrm{IMU,k}}$ is the IMU state vector at time-step k;
- ${}^{\mathrm{W}}\hat{\mathbf{R}}_{\mathrm{C_i}}$ and ${}^{\mathrm{W}}\hat{\mathbf{t}}_{\mathrm{C_i}}$ denote the rotation matrix and translation vector estimates of the i-th camera, with $\mathrm{i} = 1, ..., \mathrm{M}$.

Accordingly, the EKF error state vector is defined as:

$$\tilde{\mathbf{x}}_{\mathrm{k}} = \begin{bmatrix} \tilde{\mathbf{x}}^{\mathrm{T}}_{\mathrm{IMU,k}} & {}^{\mathrm{W}}\boldsymbol{\delta\theta}^{\mathrm{T}}_{\mathrm{C_1}} & {}^{\mathrm{W}}\tilde{\mathbf{t}}^{\mathrm{T}}_{\mathrm{C_1}} & \cdots & {}^{\mathrm{W}}\boldsymbol{\delta\theta}^{\mathrm{T}}_{\mathrm{C_M}} & {}^{\mathrm{W}}\tilde{\mathbf{t}}^{\mathrm{T}}_{\mathrm{C_M}} \end{bmatrix} \tag{6.4}$$

6.1.3 MSCKF error state dimensionality

Consider that at time-step k there are M camera poses into the EKF state vector. The dimension of the error state is composed by:

- 15 scalar elements for $\hat{\mathbf{x}}^{\mathrm{T}}_{\mathrm{IMU,k}}$, 3 for each component of the vector;
- 6 scalar elements for each camera pose $\begin{bmatrix} {}^{\mathrm{W}}\boldsymbol{\delta\theta}^{\mathrm{T}}_{\mathrm{C_i}} & {}^{\mathrm{W}}\tilde{\mathbf{t}}^{\mathrm{T}}_{\mathrm{C_i}} \end{bmatrix}^{\mathrm{T}}$

for a total of $\mathrm{N} = 15 + 6\mathrm{M}$ elements. This is the EKF filter dimensionality at time-step k.

6.2 EKF Propagation

The MSCKF propagation equations are obtained by discretizing the continuous-time IMU system model.

6.2.1 IMU continuous-time system model

As introduced in Equation 4.14, the continuous-time evolution of the estimated IMU state is governed by:

$$\begin{aligned} {}^{\mathrm{W}}\dot{\mathbf{R}}_{\mathrm{I}}(t) &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}(t)\lfloor {}^{\mathrm{I}}\boldsymbol{\omega}\times\rfloor(t) \\ {}^{\mathrm{W}}\dot{\mathbf{v}}_{\mathrm{I}}(t) &= {}^{\mathrm{W}}\mathbf{a}(t) \\ {}^{\mathrm{W}}\dot{\mathbf{t}}_{\mathrm{I}}(t) &= {}^{\mathrm{W}}\mathbf{v}_{\mathrm{I}}(t) \\ {}^{\mathrm{I}}\dot{\mathbf{b}}_{\mathrm{a}}(t) &= {}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{wg}}(t) \\ {}^{\mathrm{I}}\dot{\mathbf{b}}_{\mathrm{g}}(t) &= {}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{wa}}(t) \end{aligned} \tag{6.5}$$

where ${}^{\mathrm{W}}\mathbf{a}(t)$ represents the linear acceleration in the world frame W, and ${}^{\mathrm{I}}\boldsymbol{\omega}(t)$ denotes the angular velocity in the IMU frame I. The terms ${}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{wg}}(t)$ and ${}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{wa}}(t)$ represent zero-mean, white Gaussian noise processes driving the random-walk evolution of the accelerometer and gyroscope biases, respectively.

The accelerometer and gyroscope measurements, as described in Section 4.4.1, are given by:

$$\begin{aligned} {}^{\mathrm{I}}\mathbf{a}_{\mathrm{m}}(t) &= {}^{\mathrm{I}}\mathbf{R}_{\mathrm{W}}\big({}^{\mathrm{W}}\mathbf{a}(t) - {}^{\mathrm{W}}\mathbf{g}\big) + {}^{\mathrm{I}}\mathbf{b}_{\mathrm{a}}(t) + {}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{a}}(t) \\ {}^{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{m}}(t) &= {}^{\mathrm{I}}\boldsymbol{\omega}(t) + {}^{\mathrm{I}}\mathbf{b}_{\mathrm{g}}(t) + {}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{g}}(t) \end{aligned} \tag{6.6}$$

where ${}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{g}}(t)$ and ${}^{\mathrm{I}}\boldsymbol{\eta}_{\mathrm{a}}(t)$ represent zero-mean, white Gaussian noise processes that model the intrinsic sensor noise affecting the gyroscope and accelerometer measurements, respectively. The PSD of these noise terms, commonly referred to as *noise density*, characterizes the measurement uncertainty per unit of bandwidth, as described in Section 4.3.1.

Taking the expected value of the IMU state propagation equations (6.5), the evolution of the estimated IMU state can be expressed as:

$$\begin{aligned}
{}^{\mathrm{W}}\dot{\hat{\mathbf{R}}}_{\mathrm{I}} &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}} \lfloor {}^{\mathrm{I}}\hat{\boldsymbol{\omega}} \times \rfloor \\
{}^{\mathrm{W}}\dot{\hat{\mathbf{v}}}_{\mathrm{I}} &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}} {}^{\mathrm{I}}\hat{\mathbf{a}} + {}^{\mathrm{W}}\mathbf{g} \\
{}^{\mathrm{W}}\dot{\hat{\mathbf{t}}}_{\mathrm{I}} &= {}^{\mathrm{W}}\hat{\mathbf{v}}_{\mathrm{I}} \\
{}^{\mathrm{I}}\dot{\hat{\mathbf{b}}}_{\mathrm{a}} &= \mathbf{0}_{3\times 1} \\
{}^{\mathrm{I}}\dot{\hat{\mathbf{b}}}_{\mathrm{g}} &= \mathbf{0}_{3\times 1}
\end{aligned} \tag{6.7}$$

where ${}^{\mathrm{I}}\hat{\boldsymbol{\omega}} = {}^{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{m}} - {}^{\mathrm{I}}\hat{\mathbf{b}}_{\mathrm{g}}$ and ${}^{\mathrm{I}}\hat{\mathbf{a}} = {}^{\mathrm{I}}\mathbf{a}_{\mathrm{m}} - {}^{\mathrm{I}}\hat{\mathbf{b}}_{\mathrm{a}}$. For brevity, the explicit time dependency has been omitted.

The linearized continuous-time model for the IMU error state is:

$$\dot{\tilde{\mathbf{x}}}_{\mathrm{IMU}} = \mathbf{F}\,\tilde{\mathbf{x}}_{\mathrm{IMU}} + \mathbf{G}\,\boldsymbol{\eta}_{\mathrm{IMU}} \tag{6.8}$$

where: $\boldsymbol{\eta}_{\mathrm{IMU}} = \begin{bmatrix} \boldsymbol{\eta}_{\mathrm{g}}^{\mathrm{T}} & \boldsymbol{\eta}_{\mathrm{wg}}^{\mathrm{T}} & \boldsymbol{\eta}_{\mathrm{a}}^{\mathrm{T}} & \boldsymbol{\eta}_{\mathrm{wa}}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}}$ is the system noise with covariance matrix $\mathbf{Q}_{\mathrm{IMU}}$, which depends on the IMU noise characteristics and is computed off-line during sensor calibration, as depicted in Section 4.3.

The matrices $\mathbf{F}$ and $\mathbf{G}$ describing the linearized system model are given by:

$$\mathbf{F} = \begin{bmatrix}
-\lfloor {}^{\mathrm{I}}\hat{\boldsymbol{\omega}} \times \rfloor & -\mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\
\mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\
-{}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}} \lfloor {}^{\mathrm{I}}\hat{\mathbf{a}} \times \rfloor & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & -{}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}} & \mathbf{0}_{3\times 3} \\
\mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\
\mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3}
\end{bmatrix} \tag{6.9}$$

and

$$\mathbf{G} = \begin{bmatrix}
-\mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\
\mathbf{0}_{3\times 3} & \mathbf{I}_3 & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \\
\mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & -{}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}} & \mathbf{0}_{3\times 3} \\
\mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{I}_3 \\
\mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3}
\end{bmatrix} \tag{6.10}$$

6.2.2 IMU discrete-time modeling

An IMU operates at a fixed sampling rate, meaning that the signals ${}^{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{m}}$ and ${}^{\mathrm{I}}\mathbf{a}_{\mathrm{m}}$ are discretely sampled at intervals of T, typically ranging between 10^{-2} and 10^{-3} seconds.

Each time new IMU measurements are received, they are used to perform a propagation step in the EKF. This is achieved by integrating the measurements using Euler Integration, as extensively detailed in Section 4.4.2.

In addition, the EKF covariance matrix must be propagated over time. To achieve this, the discrete-time state transition matrix and the discrete-time process noise covariance matrix must first be computed.

The discrete-time state transition matrix is obtained by numerically integrating the differential equation:

$$\dot{\mathbf{\Phi}}\left(t_{k+1}, t_k\right) = \mathbf{F}\,\mathbf{\Phi}\left(t_{k+1}, t_k\right) \tag{6.11}$$

which leads to the approximation:

$$\mathbf{\Phi}_k = \mathbf{\Phi}\left(t_{k+1}, t_k\right) \approx \mathbf{I}_{15} + \mathbf{F}\Delta t + \frac{1}{2}\mathbf{F}^2\Delta t^2 + \frac{1}{6}\mathbf{F}^3\Delta t^3 \tag{6.12}$$

while, the discrete-time process noise covariance matrix is computed as:

$$\mathbf{Q}_k = \mathbf{\Phi}_k\mathbf{G}\,\mathbf{Q}_{\text{IMU}}\,\mathbf{G}^{\text{T}}\mathbf{\Phi}_k^{\text{T}}\Delta t \tag{6.13}$$

Next, consider the following partitioning of the state covariance matrix:

$$\mathbf{P}_{k|k} = \begin{bmatrix} {}^{\text{II}}\mathbf{P}_{k|k} & {}^{\text{IC}}\mathbf{P}_{k|k} \\ {}^{\text{IC}}\mathbf{P}_{k|k}^{\text{T}} & {}^{\text{CC}}\mathbf{P}_{k|k} \end{bmatrix} \tag{6.14}$$

where:

- ${}^{\text{II}}\mathbf{P}_{k|k}$ is the 15×15 covariance matrix associated with the evolving IMU state;
- ${}^{\text{CC}}\mathbf{P}_{k|k}$ is the $6\text{M} \times 6\text{M}$ covariance matrix corresponding to the camera pose estimates;
- ${}^{\text{IC}}\mathbf{P}_{k|k}$ is the $15 \times 6\text{M}$ cross-covariance matrix, describing the correlation between the errors in the IMU state and the camera pose estimates.

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

$$\mathbf{P}_{k+1|k} = \begin{bmatrix} {}^{\text{II}}\mathbf{P}_{k+1|k} & \mathbf{\Phi}_k\,{}^{\text{IC}}\mathbf{P}_{k|k} \\ {}^{\text{IC}}\mathbf{P}_{k|k}^{\text{T}}\mathbf{\Phi}_k^{\text{T}} & {}^{\text{CC}}\mathbf{P}_{k|k} \end{bmatrix} \tag{6.15}$$

where the propagated IMU covariance matrix, ${}^{\text{II}}\mathbf{P}_{k+1|k}$, is computed as:

$${}^{\text{II}}\mathbf{P}_{k+1|k} = \mathbf{\Phi}_k\,{}^{\text{II}}\mathbf{P}_{k|k}\mathbf{\Phi}_k^{\text{T}} + \mathbf{Q}_k \tag{6.16}$$

6.3 State Augmentation and Data Association

When a new image is recorded, the EKF state vector and covariance matrix must be augmented with the corresponding camera pose, while the camera measurements are processed and stored.

6.3.1 State Augmentation

Upon capturing a new image, the camera pose estimate is obtained from the IMU pose estimate as:

$$^{\mathrm{W}}\mathbf{T}_{\mathrm{C_i}} = {}^{\mathrm{W}}\mathbf{T}_{\mathrm{I_i}}\,{}^{\mathrm{I}}\mathbf{T}_{\mathrm{C}} \tag{6.17}$$

where:

- $^{\mathrm{W}}\mathbf{T}_{\mathrm{I_i}} = \left[{}^{\mathrm{W}}\mathbf{R}_{\mathrm{I_i}} \mid {}^{\mathrm{W}}\mathbf{t}_{\mathrm{I_i}}\right]$ is the homogeneous transformation matrix representing the pose of the IMU with respect to the world frame W;
- $^{\mathrm{I}}\mathbf{T}_{\mathrm{C}} = [{}^{\mathrm{I}}\mathbf{R}_{\mathrm{C}} \mid {}^{\mathrm{I}}\mathbf{t}_{\mathrm{C}}]$ is the static homogeneous transformation matrix that defines the known rotation and translation offset between the camera and the IMU;
- $^{\mathrm{W}}\mathbf{T}_{\mathrm{C_i}} = \left[{}^{\mathrm{W}}\mathbf{R}_{\mathrm{C_i}} \mid {}^{\mathrm{W}}\mathbf{t}_{\mathrm{C_i}}\right]$ is the computed homogeneous transformation matrix describing the camera pose with respect to the world frame W.

The camera pose estimate from Equation 6.17 is then appended to the state vector of the EKF, and the state covariance matrix is updated as:

$$\mathbf{P}_{\mathrm{k|k}} \leftarrow \begin{bmatrix} \mathbf{I}_{6\mathrm{M}+15} \\ \mathbf{J} \end{bmatrix} \mathbf{P}_{\mathrm{k|k}} \begin{bmatrix} \mathbf{I}_{6\mathrm{M}+15} \\ \mathbf{J} \end{bmatrix}^{\mathrm{T}} \tag{6.18}$$

where the Jacobian matrix $\mathbf{J}$ is obtained by differentiating Equation 6.17 with respect to the IMU state vector $\hat{\mathbf{x}}_{\mathrm{IMU}}$. The transformation in Equation 6.17 can be rewritten as:

$$\begin{aligned} {}^{\mathrm{W}}\mathbf{R}_{\mathrm{C_i}} &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I_i}}\,{}^{\mathrm{I}}\mathbf{R}_{\mathrm{C}} \\ {}^{\mathrm{W}}\mathbf{t}_{\mathrm{C_i}} &= {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I_i}}\,{}^{\mathrm{I}}\mathbf{t}_{\mathrm{C}} + {}^{\mathrm{W}}\mathbf{t}_{\mathrm{I_i}} \end{aligned} \tag{6.19}$$

Differentiating these expressions with respect to the IMU state vector yields:

$$\mathbf{J} = \begin{bmatrix} {}^{\mathrm{I}}\mathbf{R}_{\mathrm{C}}^{\mathrm{T}} & \mathbf{0}_{3\times 9} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 6\mathrm{M}} \\ \lfloor {}^{\mathrm{W}}\mathbf{R}_{\mathrm{I}}\,{}^{\mathrm{I}}\mathbf{t}_{\mathrm{C}} \times \rfloor & \mathbf{0}_{3\times 9} & \mathbf{I}_3 & \mathbf{0}_{3\times 6\mathrm{M}} \end{bmatrix} \tag{6.20}$$

6.3.2 Data Association

For each captured image, features are extracted and matched against the currently stored features.

Feature extraction and matching are performed using *XFeat* [19], a lightweight and efficient Convolutional Neural Network (CNN) designed for local feature detection and matching. XFeat offers a fast, accurate, and hardware-independent solution, making it ideal for resource-constrained devices such as mobile robots, augmented reality systems, and embedded platforms.

Feature storage format

In the MSCKF framework, *features* are not directly incorporated into the EKF state but are instead stored separately in a list. Each element in this list is a `feature` and contains the following information:

- `keypoints`: Image coordinates of the same 3D world point, reprojected into the image plane at different time-steps;
- `descriptors`: A vector of 64 `float` elements representing the feature descriptor associated with the corresponding keypoint.
- `scores`: A `float` scalar indicating the confidence level of the XFeat feature extractor for the corresponding keypoint.
- `camera_indices`: A vector of `int` elements, where each entry represents the index of the camera that captured the keypoint.
- `lines`: A vector of rays obtained from the inverse projection of the corresponding keypoint (see Section 3.1.3). Each line is parametrized as described in Section 3.3.1.
- `inverse_depth_point`: The inverse depth parametrization of the 3D point that generated the keypoints (see Section 3.4).
- `tracked_for_n_frames`: A counter indicating the number of frames in which the 3D point was successfully tracked (not necessarily consecutive).
- `lost_for_n_frames`: A counter indicating the number of consecutive frames in which the 3D point was not tracked, following at least one successful tracking frame.

Feature Matching

The feature matching process consists of two main steps:

- descriptor-based matching;
- epipolar matching refinement.

Descriptor-Based Matching Descriptor-based matching is performed using the XFeat matcher, which compares feature descriptors and accepts a match only if the *cosine similarity* between descriptors exceeds a predefined threshold. The matching process involves two sets:

- *Current keypoint set*: A set of tuples containing n keypoints, descriptors, and scores extracted from the most recently recorded image:

$$\mathbf{P} = [(\mathbf{k}_1, \mathbf{d}_1, s_1), ..., (\mathbf{k}_n, \mathbf{d}_n, s_n)] \tag{6.21}$$

  where each tuple consists of a keypoint $\mathbf{k}_i$, its descriptor $\mathbf{d}_i$ and score s_i.

- *Stored feature set*: A list containing m *descriptor-index pairs* from previously tracked features:

$$\mathbf{Q} = [(\mathbf{d}_1, \mathbf{f}_1^{idx}), ..., (\mathbf{d}_m, \mathbf{f}_m^{idx})] \tag{6.22}$$

  where each descriptor $\mathbf{d}_j$ represents a feature and is associated with an index $\mathbf{f}_j^{idx}$.

The set $\mathbf{Q}$ is constructed as follows:

1. for each stored feature $\mathbf{f}_j$, compute a *representative descriptor* $\mathbf{d}_j$ as the weighted mean of all its associated descriptors, using their scores as weights;
2. store each pair $(\mathbf{d}_j, \mathbf{f}_j^{idx})$ in a list for use in the next time-step.

Matching is then performed between the *current descriptors* from $\mathbf{P}$, $[\mathbf{d}_1, ..., \mathbf{d}_n]$, and the *stored descriptors* in $\mathbf{Q}$ from the previous time-step, $[\mathbf{d}_1, ..., \mathbf{d}_m]$, using the XFeat matcher.

This approach ensures that *matching descriptors implicitly determine feature correspondences* while maintaining a low computational cost, as each matched descriptor in the stored list is directly associated with a *specific feature index*. Consequently, newly detected keypoints can be accurately linked to their corresponding features.

If an element in the current keypoint set, $(\mathbf{k}_i, \mathbf{d}_i, s_i)$, does not match any element in the stored feature set, it is considered a newly tracked feature and is added to the stored features. On the other hand, if a stored feature does not match any newly detected keypoint, it is considered lost for that frame, and the `lost_for_n_frames` counter is incremented.

Epipolar Matching Refinement Since descriptor-based matching is an approximation, where a newly detected keypoint is matched against a *representative descriptor* of a feature that aggregates multiple keypoints and their descriptors, an *epipolar refinement procedure* is applied to improve the accuracy of the data association step. This refinement enhances the reliability of feature matching and, consequently, improves the performance of the MSCKF filter.

The epipolar matching refinement process is carried out as follows:

1. For each tuple $(\mathbf{k}_i, \mathbf{d}_i, s_i)$ in the new camera measurement, retrieve the matched *feature index* $\mathbf{f}_j^{idx}$, which corresponds to the stored feature $\mathbf{f}_j$.

2. Iterate over the keypoints associated with the feature $\mathbf{f}_j$, extracting at each step:
   - a previously observed keypoint $\mathbf{k}_j$;
   - the camera index idx_j^C, from which the corresponding camera pose in the EKF state, ${}^W\mathbf{T}_{C_j}$, can be retrieved.
3. Compute the relative homogeneous transformation between the current camera pose estimate ${}^W\mathbf{T}_{C_i}$ and ${}^W\mathbf{T}_{C_j}$, i.e., ${}^{C_j}\mathbf{T}_{C_i}$.
4. If the relative translation between C_j and C_i is below a small threshold, the motion is assumed to be *purely rotational*, which can be described by a *Homography matrix* (see Section 3.2.4). In this case:
   (a) Compute the Homography matrix $\mathbf{H}$ using Equation 3.23.
   (b) Using the formulation in Equation 3.24, predict the position of $\mathbf{k}_j$ in the image plane of camera C_i, obtaining $\hat{\mathbf{k}}_j$, and the position of $\mathbf{k}_i$ in the image plane of camera C_j, obtaining $\hat{\mathbf{k}}_i$.
   (c) Compute the *Homography score* as the mean of the norm between the predicted and actual keypoints:
   $$\mathbf{H}_{\text{score}} = \frac{1}{2}\left(\left\|\mathbf{k}_j - \hat{\mathbf{k}}_j\right\| + \left\|\mathbf{k}_i - \hat{\mathbf{k}}_i\right\|\right) \tag{6.23}$$
   (d) Reject the match if $\mathbf{H}_{\text{score}}$ exceeds a predefined threshold.
5. Otherwise, if the motion includes both *rotation and translation*, it is assumed to follow a *Fundamental matrix* model. In this case:
   (a) Compute the *Essential matrix* $\mathbf{E}$ from ${}^{C_j}\mathbf{R}_{C_i}$ and ${}^{C_j}\mathbf{t}_{C_i}$ using Equation 3.22.
   (b) Derive the *Fundamental matrix* $\mathbf{F}$ using Equation 3.21.
   (c) Compute the *epipolar constraint score* from Equation 3.18 as:
   $$\mathbf{F}_{\text{score}} = \mathbf{k}_i^T \mathbf{F} \mathbf{k}_j \tag{6.24}$$
   (d) Reject the match if $\mathbf{F}_{\text{score}}$ exceeds a predefined threshold.

If the descriptor-based match passes the epipolar test, the feature $\mathbf{f}_j$ is updated by incorporating the newly detected keypoint $\mathbf{k}_i$, its corresponding descriptor $\mathbf{d}_i$, the score s_i, the line obtained from the inverse projection of $\mathbf{k}_i$, and the associated camera index. Finally, the `tracked_for_n_frames` variable is incremented, while `lost_for_n_frames` is reset to zero.

Conversely, if the descriptor-based match fails the epipolar test, the feature $\mathbf{f}_j$ is considered lost, and the `lost_for_n_frames` counter is incremented.

6.4 Measurement model and Correction

The *measurement model* proposed by Mourikis and Roumeliotis in [16] is based on the principle that observing a static feature from multiple camera poses imposes constraints involving all these poses. Specifically, all measurements corresponding to the same 3D point contribute to defining a constraint equation that links the camera poses from which the observations were made.

To leverage these constraints effectively, camera observations are grouped based on *tracked features*, as detailed in Section 6.3.2.

6.4.1 Measurement model

Since the EKF is employed for state estimation, constructing a measurement model requires defining a residual $\mathbf{r}$ that depends linearly on the state errors $\tilde{\mathbf{x}}$, following the general formulation:

$$\mathbf{r} = \mathbf{H}\tilde{\mathbf{x}} + \boldsymbol{\eta} \tag{6.25}$$

where $\mathbf{H}$ is the measurement matrix, and $\boldsymbol{\eta}$ represents the measurement noise, which is modeled as zero-mean Gaussian white noise.

Consider a single feature $\mathbf{f}_{\mathrm{j}}$ observed from a set of M_{j} camera poses ${}^{\mathrm{W}}\hat{\mathbf{T}}_{\mathrm{C_i}}$, where $\mathrm{i} \in \mathcal{S}_{\mathrm{j}} = \{1, ..., \mathrm{M}_{\mathrm{j}}\}$. Each of these M_{j} observations is modeled as:

$$\mathbf{z}_{\mathrm{i}}^{\mathrm{j}} = \frac{1}{{}^{\mathrm{C_i}}z_{\mathrm{j}}} \begin{bmatrix} {}^{\mathrm{C_i}}x_{\mathrm{j}} \\ {}^{\mathrm{C_i}}y_{\mathrm{j}} \end{bmatrix} + \boldsymbol{\eta}_{\mathrm{i}}^{\mathrm{j}} \quad , \quad \begin{bmatrix} {}^{\mathrm{C_i}}x_{\mathrm{j}} \\ {}^{\mathrm{C_i}}y_{\mathrm{j}} \\ {}^{\mathrm{C_i}}z_{\mathrm{j}} \end{bmatrix} = \mathbf{K}^{-1\ {}^{\mathrm{Im_i}}}\mathbf{x}_{\mathrm{j},\mathcal{H}} \quad , \quad \mathrm{i} \in \mathcal{S}_{\mathrm{j}} \tag{6.26}$$

where $\boldsymbol{\eta}_{\mathrm{i}}^{\mathrm{j}}$ is a 2×1 image noise vector with covariance matrix $\mathbf{R}_{\mathrm{i}}^{\mathrm{j}} = \sigma_{\mathrm{Im}}^2 \mathbf{I}_2$.

The position of the feature expressed in the camera frame $\mathrm{C_i}$ can be written as:

$${}^{\mathrm{C_i}}\hat{\mathbf{p}}_{\mathrm{f_j}} = {}^{\mathrm{C_i}}\hat{\mathbf{R}}_{\mathrm{W}} \left({}^{\mathrm{W}}\hat{\mathbf{p}}_{\mathrm{f_j}} - {}^{\mathrm{W}}\hat{\mathbf{t}}_{\mathrm{C_i}} \right) \tag{6.27}$$

However, to better handle low-parallax cases, the Inverse Depth Parametrization parametrization is adopted (see Section 3.4), leading to the alternative formulation given in Equation 3.44:

$${}^{\mathrm{C_i}}\hat{\mathbf{p}}_{\mathrm{f_j}} = {}^{\mathrm{C_i}}\hat{\mathbf{R}}_{\mathrm{W}} \left(\hat{\rho}_{\mathrm{i}}^{\mathrm{j}} \left({}^{\mathrm{W}}\hat{\mathbf{t}}_{\mathrm{C_1}} - {}^{\mathrm{W}}\hat{\mathbf{t}}_{\mathrm{C_i}} \right) + \mathbf{m} \left(\hat{\theta}_{\mathrm{i}}^{\mathrm{j}}, \hat{\phi}_{\mathrm{i}}^{\mathrm{j}} \right) \right) \tag{6.28}$$

where ${}^{\mathrm{W}}\hat{\mathbf{t}}_{\mathrm{C_1}}$ represents the position of the first camera that observed the feature.

The depth $\hat{\rho}_{\mathrm{i}}^{\mathrm{j}}$ is estimated by computing ${}^{\mathrm{W}}\hat{\mathbf{p}}_{\mathrm{f_j}}$ through *3D reconstruction via intersection of lines* (see Section 3.3.1). This estimation leverages all lines obtained from the inverse projection of keypoints associated with the feature $\mathbf{f}_{\mathrm{j}}$ (see Section 3.1.3).

Once the estimate of the feature position is obtained, the *measurement residual* can be computed as:

$$\mathbf{r}_{\mathrm{i}}^{\mathrm{j}} = \mathbf{z}_{\mathrm{i}}^{\mathrm{j}} - \hat{\mathbf{z}}_{\mathrm{i}}^{\mathrm{j}} \tag{6.29}$$

where:

$$\hat{\mathbf{z}}_{\mathrm{i}}^{\mathrm{j}} = \frac{1}{{}^{\mathrm{C_i}}\hat{z}_{\mathrm{j}}} \begin{bmatrix} {}^{\mathrm{C_i}}\hat{x}_{\mathrm{j}} \\ {}^{\mathrm{C_i}}\hat{y}_{\mathrm{j}} \end{bmatrix} \quad , \quad {}^{\mathrm{C_i}}\hat{\mathbf{p}}_{\mathrm{f_j}} = \begin{bmatrix} {}^{\mathrm{C_i}}\hat{x}_{\mathrm{j}} \\ {}^{\mathrm{C_i}}\hat{y}_{\mathrm{j}} \\ {}^{\mathrm{C_i}}\hat{z}_{\mathrm{j}} \end{bmatrix} \tag{6.30}$$

By linearizing around the estimated camera pose and feature position, the residual in Equation 6.29 can be approximated as:

$$\mathbf{r}_{\mathrm{i}}^{\mathrm{j}} \approx {}^{\mathbf{x}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}}\tilde{\mathbf{x}} + {}^{\mathbf{f}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}}{}^{\mathrm{W}}\tilde{\mathbf{p}}_{\mathrm{f_j}} + \boldsymbol{\eta}_{\mathrm{i}}^{\mathrm{j}} \tag{6.31}$$

where ${}^{\mathrm{W}}\tilde{\mathbf{p}}_{\mathrm{f_j}}$ represents the error in the estimated position of the feature $\mathbf{f}_{\mathrm{j}}$. The matrices ${}^{\mathbf{x}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}}$ and ${}^{\mathbf{f}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}}$ correspond to the Jacobians of the measurement $\mathbf{z}_{\mathrm{i}}^{\mathrm{j}}$ with respect to the state and the feature position, respectively, and are given by:

$$\begin{aligned} {}^{\mathbf{x}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}} &= \begin{bmatrix} \mathbf{0}_{2\times 15} & \mathbf{0}_{2\times 6} & \dots & \mathbf{J}_{\mathrm{i}}^{\mathrm{j}} \lfloor {}^{\mathrm{C_i}}\hat{\mathbf{p}}_{\mathrm{f_j}} \times \rfloor & -\mathbf{J}_{\mathrm{i}}^{\mathrm{j}}{}^{\mathrm{C_i}}\mathbf{R}_{\mathrm{W}} & \dots & \mathbf{0}_{2\times 6} \end{bmatrix} \\ {}^{\mathbf{f}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}} &= \mathbf{J}_{\mathrm{i}}^{\mathrm{j}}{}^{\mathrm{C_i}}\mathbf{R}_{\mathrm{W}} \\ \mathbf{J}_{\mathrm{i}}^{\mathrm{j}} &= \frac{1}{{}^{\mathrm{C_i}}\hat{z}_{\mathrm{j}}} \begin{bmatrix} 1 & 0 & -\frac{{}^{\mathrm{C_i}}\hat{x}_{\mathrm{j}}}{{}^{\mathrm{C_i}}\hat{z}_{\mathrm{j}}} \\ 0 & 1 & -\frac{{}^{\mathrm{C_i}}\hat{y}_{\mathrm{j}}}{{}^{\mathrm{C_i}}\hat{z}_{\mathrm{j}}} \end{bmatrix} \end{aligned} \tag{6.32}$$

By stacking the residuals from all $\mathrm{M_j}$ observations of this feature, the full residual of the feature $\mathbf{f}_{\mathrm{j}}$ can be expressed as:

$$\mathbf{r}^{\mathrm{j}} = \mathbf{H}_{\mathbf{x}}^{\mathrm{j}}\tilde{\mathbf{x}} + \mathbf{H}_{\mathbf{f}}^{\mathrm{j}}{}^{\mathrm{W}}\tilde{\mathbf{p}}_{\mathrm{f_j}} + \boldsymbol{\eta}^{\mathrm{j}} \tag{6.33}$$

where:

- $\mathbf{r}^{\mathrm{j}}$ and $\boldsymbol{\eta}^{\mathrm{j}}$ are vectors containing block elements $\mathbf{r}_{\mathrm{i}}^{\mathrm{j}}$ and $\boldsymbol{\eta}_{\mathrm{i}}^{\mathrm{j}}$, respectively, for $\mathrm{i} \in \mathcal{S}_{\mathrm{j}}$;
- $\mathbf{H}_{\mathbf{x}}^{\mathrm{j}}$ and $\mathbf{H}_{\mathbf{f}}^{\mathrm{j}}$ are matrices whose block elements correspond to ${}^{\mathbf{x}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}}$ and ${}^{\mathbf{f}}\mathbf{H}_{\mathrm{i}}^{\mathrm{j}}$, respectively, for $\mathrm{i} \in \mathcal{S}_{\mathrm{j}}$.

Since the feature observations in different images are independent, the covariance matrix of $\boldsymbol{\eta}^{\mathrm{j}}$ is given by:

$$\mathbf{R}^{\mathrm{j}} = \sigma_{\mathrm{Im}}^2 \mathbf{I}_{2\mathrm{M_j}} \tag{6.34}$$

As the feature position estimate, ${}^{\mathrm{W}}\hat{\mathbf{p}}_{\mathrm{f_j}}$, is derived using the Intersection of Lines algorithm, which relies on the estimated camera poses within the state estimate, $\hat{\mathbf{x}}$, the resulting position error, ${}^{\mathrm{W}}\tilde{\mathbf{p}}_{\mathrm{f_j}}$, in Equation 6.4.1 becomes inherently correlated with the state errors, $\tilde{\mathbf{x}}$.

Consequently, the residual $\mathbf{r}^{\mathrm{j}}$ does not follow to the structure of Equation 6.25 and cannot be directly utilized for measurement updates in the EKF.

To address this issue, [16] proposes an alternative residual, $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$, obtained by projecting the residual $\mathbf{r}^{\mathrm{j}}$ from Equation 6.4.1 onto the left nullspace of the matrix $\mathbf{H}_{\mathbf{f}}^{\mathrm{j}}$.

Let $\mathbf{A}$ be the unitary matrix whose columns form a basis for the left nullspace of $\mathbf{H}_{\mathbf{f}}^{\mathrm{j}}$. The projected residual is then given by:

$$\mathbf{r}_{\mathrm{o}}^{\mathrm{j}} = \mathbf{A}^{\mathrm{T}}\left(\mathbf{z}^{\mathrm{j}} - \hat{\mathbf{z}}^{\mathrm{j}}\right) \approx \mathbf{A}^{\mathrm{T}}\mathbf{H}_{\mathbf{x}}^{\mathrm{j}}\tilde{\mathbf{x}} + \mathbf{A}^{\mathrm{T}}\boldsymbol{\eta}^{\mathrm{j}} = \mathbf{H}_{\mathrm{o}}^{\mathrm{j}}\tilde{\mathbf{x}} + \boldsymbol{\eta}_{\mathrm{o}}^{\mathrm{j}} \tag{6.35}$$

Since the matrix $\mathbf{H}_{\mathbf{f}}^{\mathrm{j}}$ of size $2\mathrm{M}_{\mathrm{j}} \times 3$ has full column rank, its left nullspace has a dimension of $2\mathrm{M}_{\mathrm{j}} - 3$. As a result, the projected residual $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$ is a $(2\mathrm{M}_{\mathrm{j}} - 3) \times 1$ vector. As this residual is *independent* of the errors in the feature coordinates, it can be directly used for EKF updates.

The Equation 6.35 establishes a *linearized* constraint that links all camera poses from which the feature $\mathbf{f}_{\mathrm{j}}$ was observed. This formulation contains all the information provided by the measurements $\mathbf{z}_{\mathrm{i}}^{\mathrm{j}}$ for the M_{j} camera states, ensuring that the resulting EKF update is optimal, except for the inaccuracies introduced by linearization.

6.4.2 EKF Correction

At a given time step, the constraints corresponding to L features must be processed. For each feature, the respective residual vector $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$ and measurement matrix $\mathbf{H}_{\mathrm{o}}^{\mathrm{j}}$ are computed, where $\mathrm{j} \in \{1, ..., \mathrm{L}\}$. The overall residual, incorporating all features, is then obtained by stacking the individual residuals $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$ into a single vector:

$$\mathbf{r}_{\mathrm{o}} = \mathbf{H}_{\mathbf{x}}\tilde{\mathbf{x}} + \boldsymbol{\eta}_{\mathrm{o}} \tag{6.36}$$

where:

- $\mathbf{r}_{\mathrm{o}}$ and $\boldsymbol{\eta}_{\mathrm{o}}$ are vectors composed of block elements $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$ and $\boldsymbol{\eta}_{\mathrm{o}}^{\mathrm{j}}$, respectively, for each $\mathrm{j} \in \{1, ..., \mathrm{L}\}$;
- $\mathbf{H}_{\mathbf{x}}$ is a matrix constructed from block elements $\mathbf{H}_{\mathbf{x}}^{\mathrm{j}}$, for each $\mathrm{j} \in \{1, ..., \mathrm{L}\}$.

Since feature measurements are statistically independent, the noise vectors $\boldsymbol{\eta}_{\mathrm{o}}^{\mathrm{j}}$ are uncorrelated. Consequently, the covariance matrix of the noise vector $\boldsymbol{\eta}_{\mathrm{o}}$ is given by:

$$\mathbf{R}_{\mathrm{o}} = \sigma_{\mathrm{Im}}^2\mathbf{I}_{\mathrm{d}} \quad , \quad d = \sum_{\mathrm{j}=1}^{\mathrm{L}}\left(2\mathrm{M}_{\mathrm{j}} - 3\right) \tag{6.37}$$

where d represents the dimension of the residual vector $\mathbf{r}_{\mathrm{o}}$.

A feature residual $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$, along with its corresponding measurement matrix $\mathbf{H}_{\mathrm{o}}^{\mathrm{j}}$, is included in the overall residual vector $\mathbf{r}_{\mathrm{o}}$ and the overall measurement matrix $\mathbf{H}_{\mathbf{x}}$ only if it satisfies the *Mahalanobis gating test*. Specifically, it is computed:

$$\gamma_{\mathrm{j}} = \left(\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}\right)^{\mathrm{T}}\left(\mathbf{H}_{\mathrm{o}}^{\mathrm{j}}\mathbf{P}_{\mathrm{k+1|k}}\left(\mathbf{H}_{\mathrm{o}}^{\mathrm{j}}\right)^{\mathrm{T}} + \sigma_{\mathrm{Im}}^2\mathbf{I}_{2\mathrm{M}_{\mathrm{j}}-3}\right)^{-1}\mathbf{r}_{\mathrm{o}}^{\mathrm{j}} \tag{6.38}$$

which is then compared against a threshold determined by the 95th percentile of the chi-squared (χ^2) distribution with $2\mathrm{M}_{\mathrm{j}} - 3$ degrees of freedom. The residual is accepted if:

$$\gamma_{\mathrm{j}} \leq \chi^2\left(0.95, 2\mathrm{M}_{\mathrm{j}} - 3\right) \tag{6.39}$$

As the dimension d of the residual can be large, the computational complexity of the EKF update may significantly increase. To mitigate this, the QR decomposition of the matrix $\mathbf{H}_{\mathbf{x}}$ is utilized:

$$\mathbf{H}_{\mathbf{x}} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T}_{\mathbf{H}} \\ \mathbf{0} \end{bmatrix} \tag{6.40}$$

where $\mathbf{Q}_1$ and $\mathbf{Q}_2$ are unitary matrices whose columns span the range and nullspace of $\mathbf{H}_{\mathbf{x}}$, respectively, while $\mathbf{T}_{\mathbf{H}}$ is an upper triangular matrix. Using this definition of the matrix $\mathbf{H}_{\mathbf{x}}$, the Equation 6.36 become:

$$\mathbf{r}_{\mathrm{o}} = \begin{bmatrix} \mathbf{Q}_1 & \mathbf{Q}_2 \end{bmatrix} \begin{bmatrix} \mathbf{T}_{\mathbf{H}} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} + \boldsymbol{\eta}_{\mathrm{o}} \tag{6.41}$$

which can be rewritten as:

$$\begin{bmatrix} \mathbf{Q}_1^{\mathrm{T}} \mathbf{r}_{\mathrm{o}} \\ \mathbf{Q}_2^{\mathrm{T}} \mathbf{r}_{\mathrm{o}} \end{bmatrix} = \begin{bmatrix} \mathbf{T}_{\mathbf{H}} \\ \mathbf{0} \end{bmatrix} \tilde{\mathbf{x}} + \begin{bmatrix} \mathbf{Q}_1^{\mathrm{T}} \boldsymbol{\eta}_{\mathrm{o}} \\ \mathbf{Q}_2^{\mathrm{T}} \boldsymbol{\eta}_{\mathrm{o}} \end{bmatrix} \tag{6.42}$$

Since the term $\mathbf{Q}_2^{\mathrm{T}} \mathbf{r}_{\mathrm{o}}$ consists only of noise, it can be discarded. Consequently, Equation 6.42 simplifies to:

$$\mathbf{r}_{\mathrm{n}} = \mathbf{Q}_1^{\mathrm{T}} \mathbf{r}_{\mathrm{o}} = \mathbf{T}_{\mathbf{H}} \tilde{\mathbf{x}} + \boldsymbol{\eta}_{\mathrm{n}} \tag{6.43}$$

where $\boldsymbol{\eta}_{\mathrm{n}} = \mathbf{Q}_1^{\mathrm{T}} \boldsymbol{\eta}_{\mathrm{o}}$ is a noise vector with covariance $\mathbf{R}_{\mathrm{n}} = \mathbf{Q}_1^{\mathrm{T}} \mathbf{R}_{\mathrm{o}} \mathbf{Q}_1 = \sigma_{\mathrm{Im}}^2 \mathbf{I}_{\mathrm{r}}$, where r is the number of columns in $\mathbf{Q}_1$.

The EKF update is then performed by computing the Kalman Gain:

$$\mathbf{K} = \mathbf{P}_{\mathrm{k+1|k}} \mathbf{T}_{\mathbf{H}}^{\mathrm{T}} \left(\mathbf{T}_{\mathbf{H}} \mathbf{P}_{\mathrm{k+1|k}} \mathbf{T}_{\mathbf{H}}^{\mathrm{T}} + \mathbf{R}_{\mathrm{n}} \right)^{-1} \tag{6.44}$$

The state correction vector is given by:

$$\Delta \mathbf{x} = \mathbf{K} \mathbf{r}_{\mathrm{n}} \tag{6.45}$$

while the state covariance matrix is updated as:

$$\mathbf{P}_{\mathrm{k+1|k+1}} = \left(\mathbf{I}_{15+6\mathrm{M}} - \mathbf{K} \mathbf{T}_{\mathbf{H}} \right) \mathbf{P}_{\mathrm{k+1|k}} \left(\mathbf{I}_{15+6\mathrm{M}} - \mathbf{K} \mathbf{T}_{\mathbf{H}} \right)^{T} + \mathbf{K} \mathbf{R}_{\mathrm{n}} \mathbf{K}^{\mathrm{T}} \tag{6.46}$$

6.4.3 Update strategies

The EKF updates are triggered only under specific conditions. In [16], an update is performed when::

1. a feature that has been tracked over multiple images is no longer detected, meaning it has moved out of the camera's FoV;
2. the maximum allowable number of camera poses, $\mathrm{M}_{\mathrm{max}}$, has been reached.

In the first case, all measurements associated with features that have moved out of the camera's FoV are processed as described in Sections 6.4.1 and 6.4.2. Once processed, these features are removed from storage. After the feature removal step, a check is performed to determine if there are any camera states in the EKF that are no longer associated with any tracked features. Such camera states are then removed from the EKF state vector, and the covariance matrix is updated accordingly.

The second update strategy is employed to manage the computational complexity of the EKF update as the state vector and covariance matrix grow in size. When the maximum number of allowed camera poses is reached, at least one pose must be removed from the EKF state vector, to reduce the size of both the state vector and the covariance matrix. Before a camera pose is removed, all feature observations associated with that pose are utilized. Specifically, features observed by any camera pose chosen to be discarded are used for an EKF update, after which all instances of the removed camera poses are deleted from the stored feature data. Unlike the first update strategy, features used for an EKF update in this case are not deleted from storage, unless they are no longer observed by any camera pose.

In [16], it was proposed to remove $M_{max}/3$ camera poses, evenly spaced in time, starting from the second-oldest pose. However, as noted in [23], this approach can lead to sudden jumps in computational load in real-time implementations. A more desirable strategy is to remove a single camera state at each time step to distribute the computational cost more evenly. However, removing only one camera state per step is impractical in the MSCKF framework, as it would result in a trivial measurement model based on Equation 6.31.

To address this issue, in [23] was proposed to remove two camera states every other EKF update step. In this project, was adopted a similar strategy, removing two camera states whenever the maximum number of camera states in the EKF state vector is reached. The two camera states selected for removal are those with the fewest tracked features, ensuring minimal impact on the estimation process.

In addition to the first two criteria, this project introduces a third update strategy that utilizes features currently within the camera's FoV but with sufficient parallax to ensure an accurate estimation of their 3D positions. The parallax is computed between the first and last tracked keypoints associated with each feature. Unlike the first two strategies, features selected under this criterion are not removed after the update, meaning that feature removal occurs only under the first two update strategies.

6.4.4 MSCKF algorithm

Both Algorithm 4 and Figure 6.1 highlight the key steps of the MSCKF. The notation used is clarified as follows:

- γ represents the Mahalanobis distance;

- $\chi^2\left(0.95, 2\mathrm{M_j} - 3\right)$ denotes the Mahalanobis gating test threshold;
- M is the current number of camera states in the EKF state vector;
- $\mathrm{M_j}$ represents the number of cameras that have observed the feature $\mathbf{f_j}$;
- $\mathrm{M_{max}}$ is the maximum allowable number of camera states in the EKF state vector;
- In the MSCKF algorithm, the $\leftarrow$ symbol denotes the stacking operator.

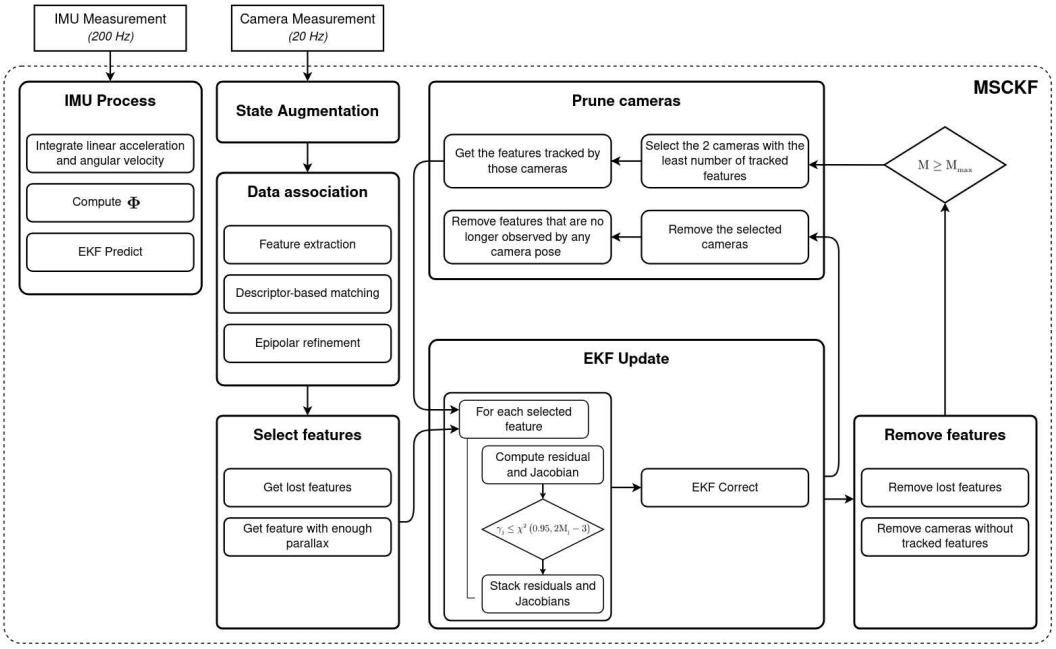


Figure 6.1. MSCKF Workflow: The MSCKF receives both IMU and camera measurements as inputs. The IMU measurement is utilized to execute an EKF prediction step. The camera measurement firstly undergoes *state augmentation*, wherein the EKF state covariance matrix is resized accordingly, then the *data association* step extracts features from the current image and matches them with those in the stored image using descriptors. Following this, an epipolar refinement procedure eliminates false positive matches. Next, a *feature selection* step is conducted, involving the first and third types of update strategies. The selected features contribute to an EKF update before being removed from storage. If the maximum allowable number of camera states is reached, the two cameras tracking the least number of features are chosen for removal. However, prior to their deletion, an EKF update is performed using the features tracked by these cameras. It is important to underlying that the last step is executed regardless of whether an EKF update has already been conducted using the selected features from the first and third update strategies (as illustrated in the diagram). For clarity, this latter scenario is not explicitly depicted in the diagram to enhance readability.

Algorithm 4 MSCKF Algorithm

Require: IMU: ${}^{\mathrm{I}}\mathbf{a}_{\mathrm{m}}$, ${}^{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{m}}$

Require: Camera: $\mathbf{P} = [(\mathbf{k}_1, \mathbf{d}_1, s_1), ..., (\mathbf{k}_{\mathrm{n}}, \mathbf{d}_{\mathrm{n}}, s_{\mathrm{n}})]$

IMU_process$({}^{\mathrm{I}}\mathbf{a}_{\mathrm{m}}, {}^{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{m}})$
state_augmentation$\left({}^{\mathrm{W}}\mathbf{T}_{\mathrm{I_i}}, {}^{\mathrm{I}}\mathbf{T}_{\mathrm{C}}\right)$
data_association$(\mathbf{P})$
valid_features = [lost_features, enough_parallax_features] = select_features()
if len(valid_features) $>$ 0 **then**
    for $\mathbf{f}_{\mathrm{j}}$ in valid_features **do**
        $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}, \mathbf{H}_{\mathrm{o}}^{\mathrm{j}}$ = compute_residual_and_jacobian()
        if $\gamma_{\mathrm{j}} \leq \chi^2\left(0.95, 2\mathrm{M}_{\mathrm{j}} - 3\right)$ **then**
            $\mathbf{H}_{\mathrm{o}} \leftarrow \mathbf{H}_{\mathrm{o}}^{\mathrm{j}}$
            $\mathbf{r}_{\mathrm{o}} \leftarrow \mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$
        end if
    end for
    EKF_correct$(\mathbf{H}_{\mathrm{o}}, \mathbf{r}_{\mathrm{o}})$
    remove_features(lost_features)
    not_tracking_cameras = get_cameras_without_features()
    remove_cameras(not_tracking_cameras)
end if
if $\mathrm{M} \geq \mathrm{M}_{\mathrm{max}}$ **then**
    poorest_cameras = select_poorest_cameras()
    extracted_features = extract_features(poorest_cameras)
    if len(extracted_features) $>$ 0 **then**
        for $\mathbf{f}_{\mathrm{j}}$ in extracted_features **do**
            $\mathbf{r}_{\mathrm{o}}^{\mathrm{j}}, \mathbf{H}_{\mathrm{o}}^{\mathrm{j}}$ = compute_residual_and_jacobian()
            if $\gamma_{\mathrm{j}} \leq \chi^2\left(0.95, 2\mathrm{M}_{\mathrm{j}} - 3\right)$ **then**
                $\mathbf{H}_{\mathrm{o}} \leftarrow \mathbf{H}_{\mathrm{o}}^{\mathrm{j}}$
                $\mathbf{r}_{\mathrm{o}} \leftarrow \mathbf{r}_{\mathrm{o}}^{\mathrm{j}}$
            end if
        end for
        EKF_correct$(\mathbf{H}_{\mathrm{o}}, \mathbf{r}_{\mathrm{o}})$
    end if
    remove_cameras(poorest_cameras)
    not_tracked_features = get_features_without_any_camera()
    remove_features(not_tracked_features)
end if

Chapter 7

Experiments and Results

To validate the MSCKF algorithm, experiments were conducted on photo-realistic datasets. The IMU data was generated using ground truth poses.

7.1 IMU Data Generation

The decision to use synthetic IMU data was primarily driven by a fundamental limitation: as discussed in Section 4.5, IMU initialization using the gravity vector suffers from the inability to decouple the initial accelerometer bias from gravity measurements. This limitation prevents the system from accurately estimating both orientation and accelerometer biases during the initialization step.

Some approaches, such as the one presented in [18], leverage the presence of a camera to correctly initialize the IMU orientation, thereby enabling proper estimation of accelerometer biases, as discussed in Section 4.5.2. However, employing a similar technique requires a robust Visual SLAM system, which must run for some seconds on the initial camera measurements to accurately estimate the IMU-camera system orientation.

This issue makes the use of real IMU measurements impractical, thereby necessitating the use of synthetic IMU data.

Furthermore, using synthetic IMU measurements provides the flexibility to manually select the noise density and random walk parameters for both the accelerometer and gyroscope.

7.1.1 IMU Measurements from positions and orientations

Consider a set of homogeneous transformation matrices, ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{I}_1}, ..., {}^{\mathrm{W}}\mathbf{T}_{\mathrm{I}_n}$, describing the ground truth trajectory of the IMU. The initial velocity is set to zero, ${}^{\mathrm{W}}\mathbf{v}_{\mathrm{I}_0} = \mathbf{0}_{3\times1}$. Starting from the second pose ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{I}_2}$, the IMU body acceleration and angular velocity are computed as:

$$
\begin{aligned}
{}^{\mathrm{W}}\mathbf{v}_{\mathrm{I_i}} &= \frac{{}^{\mathrm{W}}\mathbf{t}_{\mathrm{I_i}} - {}^{\mathrm{W}}\mathbf{t}_{\mathrm{I_{i-1}}}}{\Delta t} \\
{}^{\mathrm{I}}\mathbf{a}_{\mathrm{i}} &= {}^{\mathrm{W}}\mathbf{R}^{\mathrm{T}}_{\mathrm{I_{i-1}}} \left(\frac{{}^{\mathrm{W}}\mathbf{v}_{\mathrm{I_i}} - {}^{\mathrm{W}}\mathbf{v}_{\mathrm{I_{i-1}}}}{\Delta t} + {}^{\mathrm{W}}\mathbf{g} \right) \\
{}^{\mathrm{I}}\boldsymbol{\omega}_{\mathrm{i}} &= \frac{2}{\Delta t} \begin{bmatrix} w_{\mathrm{i-1}} x_{\mathrm{i}} - x_{\mathrm{i-1}} w_{\mathrm{i}} - y_{\mathrm{i-1}} z_{\mathrm{i}} + z_{\mathrm{i-1}} y_{\mathrm{i}} \\ w_{\mathrm{i-1}} y_{\mathrm{i}} + x_{\mathrm{i-1}} z_{\mathrm{i}} - y_{\mathrm{i-1}} w_{\mathrm{i}} - z_{\mathrm{i-1}} x_{\mathrm{i}} \\ w_{\mathrm{i-1}} z_{\mathrm{i}} - x_{\mathrm{i-1}} y_{\mathrm{i}} + y_{\mathrm{i-1}} x_{\mathrm{i}} - z_{\mathrm{i-1}} w_{\mathrm{i}} \end{bmatrix}
\end{aligned}
\tag{7.1}
$$

where:

- ${}^{\mathrm{W}}\mathbf{q}_{\mathrm{I_i}} = \begin{bmatrix} x_{\mathrm{i}} & y_{\mathrm{i}} & z_{\mathrm{i}} & w_{\mathrm{i}} \end{bmatrix}$ is the quaternion representation of ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I_i}}$;
- ${}^{\mathrm{W}}\mathbf{q}_{\mathrm{I_{i-1}}} = \begin{bmatrix} x_{\mathrm{i-1}} & y_{\mathrm{i-1}} & z_{\mathrm{i-1}} & w_{\mathrm{i-1}} \end{bmatrix}$ is the quaternion representation of ${}^{\mathrm{W}}\mathbf{R}_{\mathrm{I_{i-1}}}$;
- Δt corresponds to the time interval determined by the sensor frequency.

White Gaussian noise, characterized by zero mean and a predefined noise density, is applied to both acceleration and angular velocity. Additionally, biases are initialized to zero and incorporated into both the noisy acceleration and angular velocity, updating them at each time step according to a random walk model with a specified standard deviation.

7.2 Evaluation Metrics

The system has been evaluated based on two key aspects:

- **Estimation Error**: The accuracy of the MSCKF is assessed using the Relative Pose Error (RPE), computed for individual trials on the dataset sequences. Since the main objective is to estimate the final pose of the IMU-camera system rather than reconstruct the entire trajectory, analysing displacement errors between consecutive poses provides a more relevant accuracy measure. This evaluation method is particularly relevant for Visual-Inertial Odometry systems without loop-closure mechanisms, as is the case of the presented system. Consequently, the ATE is reported exclusively to verify whether the system's estimates remain within the 3σ confidence bounds.

  The evaluation is conducted under varying levels of IMU noise:

  - Accelerometer noise density: 0.001, 0.005, 0.01;
  - Gyroscope noise density: 0.0001, 0.0005, 0.001;
  - Accelerometer bias random walk: 0.0001, 0.0005, 0.001;
  - Gyroscope bias random walk: 0.00001, 0.00005, 0.0001.

which represent the average standard deviations of noise for typical IMU sensors used in robotic applications.

- **Real-time Performance**: The average loop frequency has been reported to evaluate the system's real-time performances.

7.2.1 Relative Pose Error (RPE)

The Relative Pose Error quantifies the difference between the estimated and true relative transformation between two consecutive poses. Let

- ${}^{\mathrm{W}}\mathbf{T}^{\mathrm{est}}_{\mathrm{I}_{\mathrm{i}-1}}$ and ${}^{\mathrm{W}}\mathbf{T}^{\mathrm{est}}_{\mathrm{I}_{\mathrm{i}}}$ denote two consecutive homogeneous transformations estimated by the MSCKF, and let
- ${}^{\mathrm{W}}\mathbf{T}^{\mathrm{gt}}_{\mathrm{I}_{\mathrm{i}-1}}$ and ${}^{\mathrm{W}}\mathbf{T}^{\mathrm{gt}}_{\mathrm{I}_{\mathrm{i}}}$ denote the corresponding ground truth transformations.

The RPE is computed as:

$$\begin{aligned}
{}^{\mathrm{I}_{\mathrm{i}-1}}\mathbf{T}^{\mathrm{est}}_{\mathrm{I}_{\mathrm{i}}} &= \left({}^{\mathrm{W}}\mathbf{T}^{\mathrm{est}}_{\mathrm{I}_{\mathrm{i}-1}}\right)^{-1} {}^{\mathrm{W}}\mathbf{T}^{\mathrm{est}}_{\mathrm{I}_{\mathrm{i}}} \\
{}^{\mathrm{I}_{\mathrm{i}-1}}\mathbf{T}^{\mathrm{gt}}_{\mathrm{I}_{\mathrm{i}}} &= \left({}^{\mathrm{W}}\mathbf{T}^{\mathrm{gt}}_{\mathrm{I}_{\mathrm{i}-1}}\right)^{-1} {}^{\mathrm{W}}\mathbf{T}^{\mathrm{gt}}_{\mathrm{I}_{\mathrm{i}}} \\
\mathbf{T}^{\mathrm{rel}}_{\mathrm{error}} &= \left({}^{\mathrm{I}_{\mathrm{i}-1}}\mathbf{T}^{\mathrm{gt}}_{\mathrm{I}_{\mathrm{i}}}\right)^{-1} {}^{\mathrm{I}_{\mathrm{i}-1}}\mathbf{T}^{\mathrm{est}}_{\mathrm{I}_{\mathrm{i}}} \\
e^{\mathrm{rel}}_{\mathbf{t},\mathrm{i}} &= \left\| \mathbf{t}^{\mathrm{rel}}_{\mathrm{error}} \right\| \\
e^{\mathrm{rel}}_{\mathbf{r},\mathrm{i}} &= \arccos\left(\frac{\mathrm{trace}\left(\mathbf{R}^{\mathrm{rel}}_{\mathrm{error}}\right) - 1}{2} \right)
\end{aligned} \tag{7.2}$$

where $\mathbf{T}^{\mathrm{rel}}_{\mathrm{error}}$ represents the transformation error between the estimated and true relative poses, with $\mathbf{t}^{\mathrm{rel}}_{\mathrm{error}}$ and $\mathbf{R}^{\mathrm{rel}}_{\mathrm{error}}$ denoting its translation and rotation components, respectively.

To ensure consistency, the relative pose error is reported as a percentage (where 1.0 corresponds to 100%) with respect to the relative ground truth translation and orientation between the considered frames:

$$\begin{aligned}
e^{\mathrm{rel}\%}_{\mathbf{t},\mathrm{i}} &= \frac{e^{\mathrm{rel}}_{\mathbf{t},\mathrm{i}}}{\left\| {}^{\mathrm{i}-1}\mathbf{t}^{\mathrm{gt}}_{\mathrm{i}} \right\|} \\
e^{\mathrm{rel}\%}_{\mathbf{r},\mathrm{i}} &= \frac{e^{\mathrm{rel}}_{\mathbf{r},\mathrm{i}}}{\arccos\left(\frac{\mathrm{trace}\left({}^{\mathrm{i}-1}\mathbf{R}^{\mathrm{gt}}_{\mathrm{i}}\right) - 1}{2} \right)}
\end{aligned} \tag{7.3}$$

where ${}^{\mathrm{i}-1}\mathbf{t}^{\mathrm{gt}}_{\mathrm{i}}$ and ${}^{\mathrm{i}-1}\mathbf{R}^{\mathrm{gt}}_{\mathrm{i}}$ are the translational and rotational components of ${}^{\mathrm{I}_{\mathrm{i}-1}}\mathbf{T}^{\mathrm{gt}}_{\mathrm{I}_{\mathrm{i}}}$, respectively.

7.2.2 Absolute Trajectory Error (ATE)

The Absolute Trajectory Error quantifies the difference between the estimated and true homogeneous transformation for each time step i. Let

- ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{I_i}}^{\mathrm{est}}$ denote the homogeneous transformation estimated by the MSCKF at time step i, and
- ${}^{\mathrm{W}}\mathbf{T}_{\mathrm{I_i}}^{\mathrm{gt}}$ denote the corresponding ground truth transformation at time step i.

The ATE is computed as:

$$\begin{aligned}
\mathbf{T}_{\mathrm{error}}^{\mathrm{abs}} &= \left({}^{\mathrm{gt}}\mathbf{T}_{\mathrm{I_i}}\right)^{-1} {}^{\mathrm{W}}\mathbf{T}_{\mathrm{I_i}}^{\mathrm{est}} \\
\mathbf{e}_{\mathbf{t},\mathrm{i}}^{\mathrm{abs}} &= \mathbf{t}_{\mathrm{error}}^{\mathrm{abs}} = \begin{bmatrix} x_{\mathrm{i}}^{\mathrm{abs}} & y_{\mathrm{i}}^{\mathrm{abs}} & z_{\mathrm{i}}^{\mathrm{abs}} \end{bmatrix}^{\mathrm{T}} \\
\mathbf{e}_{\mathbf{r},\mathrm{i}}^{\mathrm{abs}} &= \underset{\mathrm{euler}}{\angle}(\mathbf{R}_{\mathrm{error}}^{\mathrm{abs}}) = \begin{bmatrix} \theta_{\mathrm{i}}^{\mathrm{abs}} & \phi_{\mathrm{i}}^{\mathrm{abs}} & \psi_{\mathrm{i}}^{\mathrm{abs}} \end{bmatrix}^{\mathrm{T}}
\end{aligned} \tag{7.4}$$

where $\mathbf{T}_{\mathrm{error}}^{\mathrm{abs}}$ represents the transformation error between the estimated and true global poses, with $\mathbf{t}_{\mathrm{error}}^{\mathrm{abs}}$ and $\mathbf{R}_{\mathrm{error}}^{\mathrm{abs}}$ denoting its translation and rotation components, respectively.

7.3 Datasets

The system has been evaluated using two photo-realistic datasets:

- *Characterizing Visual Localization and Mapping Datasets* [22]: this dataset was constructed by combining real-world trajectories with a photo-realistic 3D environment. First, trajectories were recorded using a motion capture system in different setups, including a ground robot, a human wearing a helmet, and a MAV (Micro Air Vehicle). These recorded trajectories were then applied to a virtual camera moving within a rendered environment. To ensure a high level of realism, various visual artifacts were introduced into the images, such as motion blur, randomized lighting colour/intensity, and specular/transparent materials.
- *TartanAir: A Dataset to Push the Limits of Visual SLAM* [25]: this dataset is entirely synthetic, created using Unreal Engine and the AirSim plug-in to generate photo-realistic 3D environments. Unlike *Characterizing Visual Localization and Mapping Datasets*, both the trajectories and environments are procedurally generated. Trajectories follow diverse 6-DoF motion patterns, including aggressive rotations and translations. Additionally, it features dynamic lighting, weather conditions, and moving objects to simulate real-world challenges.

Both datasets provide ground-truth trajectory data, from which the IMU measurements were derived, as presented in Section 7.1.

7.3.1 Sequences

The sequences *Deer Running* and *Deer VR Slow* from *Characterizing Visual Localization and Mapping Datasets* were selected because they share the same environment while providing significantly different trajectory types, resulting in highly diverse and heterogeneous IMU data.

Figure 7.1 presents the ground-truth trajectories along with sample images of the environment.

The sequences *ME005* and *ME007* were selected from *TartanAir: A Dataset to Push the Limits of Visual SLAM*. The latter was chosen for its completely different environment compared to those selected from *Characterizing Visual Localization and Mapping Datasets*. In contrast, *ME005* was selected due to its numerous repetitive visual patterns. Additionally, both sequences from *TartanAir* feature significantly faster trajectories and introduce more visually challenging conditions for the camera compared to those selected from the other dataset. Figure 7.2 presents the ground-truth trajectories along with sample images of the environments.

For all selected sequences, the first 2500 frames were considered for evaluation.

7.4 Results

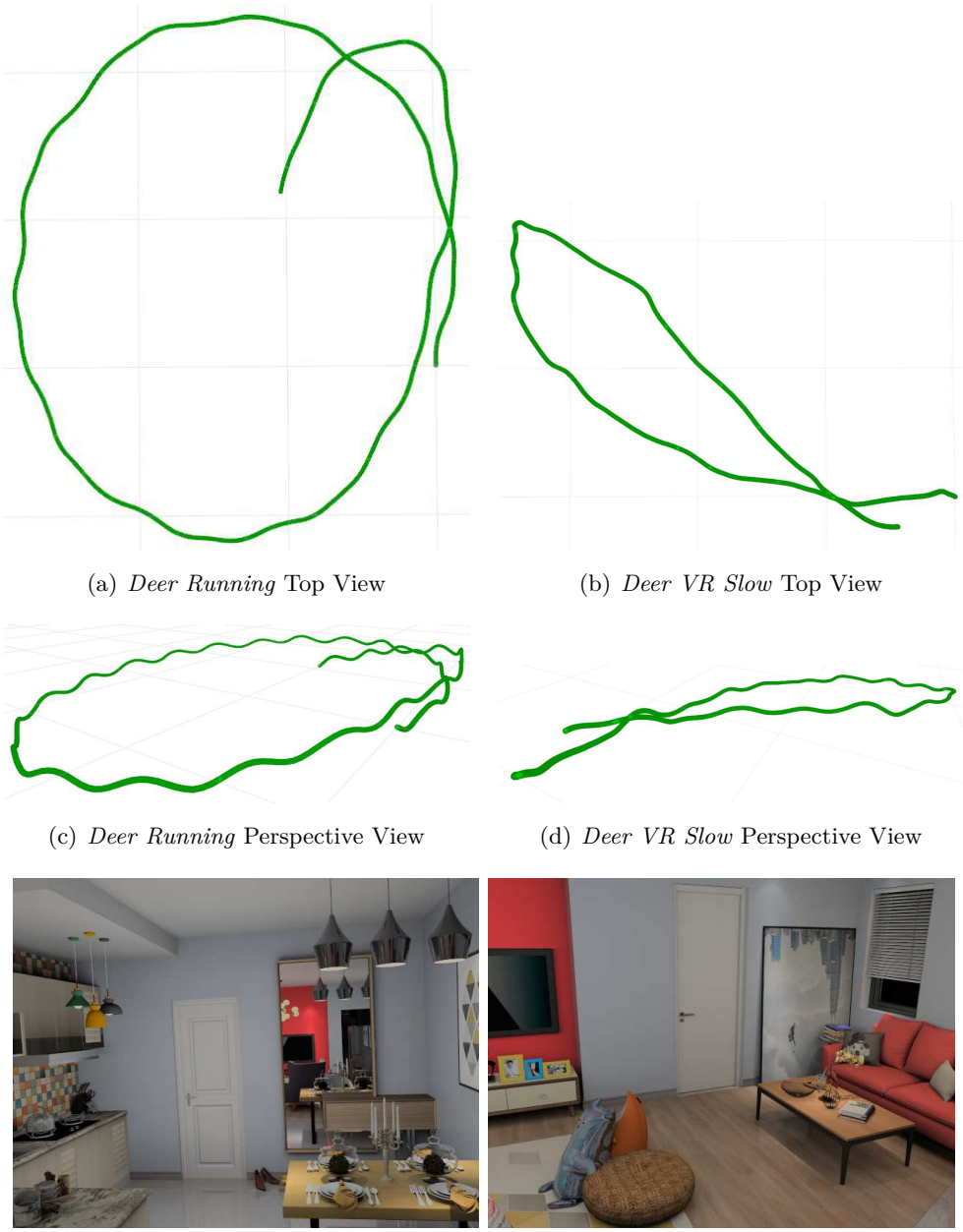
For each sequence, results are presented at different levels of IMU noise, including RPE for both translational and rotational components, along with their mean and standard deviation, as well as ATE with 3σ bounds.

The IMU measurements were obtained by simulating a sensor with an update frequency of 200 Hz, corresponding to a time step of $\Delta t = 0.005$.

7.4.1 MSCKF Setup

The implemented MSCKF framework takes the following set of parameters as input:

- `sigma_image`: The standard deviation characterizing camera measurements. It is used in the filter to compute the measurement noise matrix $\mathbf{R}_i^j$ (see Section 6.4.1).
- `accelerometer_noise_density` and `gyroscope_noise_density`: The standard deviations characterizing the white noise of the accelerometer and gyroscope, respectively (see Section 4.3.1). These parameters are used in the filter to compute the continuous noise covariance matrix $\mathbf{Q}_{\text{IMU}}$ of the IMU (see Section 6.2.1).
- `accelerometer_random_walk` and `gyroscope_random_walk`: The standard deviations characterizing the brown noise of the accelerometer and gyroscope biases, respectively (see Section 4.3.2). These values are also used in the filter



(a) *Deer Running* Top View

(b) *Deer VR Slow* Top View

(c) *Deer Running* Perspective View

(d) *Deer VR Slow* Perspective View

(e) *Deer Running* / *Deer VR Slow* environment samples

Figure 7.1. *Characterizing Visual Localization and Mapping Datasets* [22] *Deer Running* and *Deer VR Slow* sequences. The background grid in images (a), (b), (c) and (d) has a resolution of 1 meter.

to compute the continuous noise covariance matrix $\mathbf{Q}_{\text{IMU}}$ of the IMU (see Section 6.2.1).

- `number_of_extracted_features`: The number of features extracted by *XFeat* in each image (see Section 6.3.2).

- `min_cosine_similarity`: The threshold on cosine similarity used to determine the validity of descriptor-based matches (see Section 6.3.2).
- `min_parallax`: The minimum parallax threshold between the initial and final observation of a feature required to trigger the parallax-based update strategy (see Section 6.4.3).
- `epipolar_rejection_threshold`: The epipolar score threshold above which a descriptor-based match is discarded during the epipolar refinement step (see Section 6.3.2).
- `homography_rejection_threshold`: The homography score threshold above which a descriptor-based match is discarded during the homography refinement step (see Section 6.3.2).
- `min_number_of_frames_to_be_tracked`: The minimum number of frames required for a feature to be considered successfully tracked (see Section 6.3.2).
- `min_number_of_frames_to_be_lost`: The minimum number of frames required for a feature to be considered lost (see Section 6.3.2).
- `max_number_of_camera_states`: The maximum number of camera states allowed in the EKF state. If this limit is reached, the corresponding update strategy is triggered (see Section 6.4.3).

The experiments were conducted using the following set of parameters:

- `sigma_image`: 0.1
- `accelerometer_noise_density`: 0.001 / 0.005 / 0.01
- `gyroscope_noise_density`: 0.0001 / 0.0005 / 0.001
- `accelerometer_random_walk`: 0.0001 / 0.0005 / 0.001
- `gyroscope_random_walk`: 0.00001 / 0.00005 / 0.0001
- `number_of_extracted_features`: 300
- `min_cosine_similarity`: 0.95
- `min_parallax`: 45
- `epipolar_rejection_threshold`: 0.005
- `homography_rejection_threshold`: 5
- `min_number_of_frames_to_be_tracked`: 4
- `min_number_of_frames_to_be_lost`: 2
- `max_number_of_camera_states`: 30

7.4.2 Discussion

As shown in Figures 7.3 (a, b, c), 7.4 (a, b, c), 7.5 (a, b, c), and 7.6 (a, b, c), and summarized in terms of mean and standard deviation in Tables 7.1, 7.2, 7.3, and 7.4, the RPE results are generally promising. Specifically, the translational error remains below 4% for low and medium IMU noise levels across all sequences. However, at high IMU noise levels, the error increases, reaching up to 10% in sequence ME005, which contains the fewest available visual features due to the presence of numerous repetitive patterns.

While the translational RPE results are reasonable, the relative rotational errors are significantly lower, with a maximum error of only 0.7% across all sequences. This result can be attributed to two main factors. First, the gyroscope is inherently less noisy than the accelerometer and has been modeled accordingly. Second, the MSCKF parametrizes features using the Inverse Depth Parametrization (see Section 3.4), enabling accurate extraction of information from features with low parallax, either due to tracking loss or because the feature is far from the camera.

Conversely, accurate translational estimation relies on features exhibiting sufficient parallax throughout the observation period, which is essential for precise 3D reconstruction (see Section 3.3).

Regarding ATE, Figures 7.3 (d, e, f), 7.4 (d, e, f), 7.5 (d, e, f), and 7.5 (d, e, f) show that errors remain within the 3σ bounds across all sequences, confirming the filter's consistency. However, as the proposed framework is a tracking-based Visual-Inertial Odometry system without loop closure or global map optimization, estimates naturally drift over time. The most significant drift is observed in sequence ME005, likely due to the presence of repetitive visual patterns in the environment.

Table 7.5 presents the average loop frequencies of the system, evaluated across all sequences at a medium IMU noise level. The evaluation was conducted on a laptop equipped with an Intel i7-13700H processor, which is generally more powerful than the typical hardware used in robotics and embedded systems. However, the implementation was developed in Python, introducing additional computational overhead compared to C++, the standard programming language for robotics applications.

Despite these constraints, the system achieves real-time performance, maintaining an average loop frequency of approximately 45 Hz across all loops. More specifically, the frequency reaches approximately 171 Hz for loops relying solely on IMU measurements and around 7 Hz for loops incorporating camera measurements.

	avg $(e_{\mathbf{t}}^{\text{rel\%}})$	std $(e_{\mathbf{t}}^{\text{rel\%}})$	avg $(e_{\mathbf{r}}^{\text{rel\%}})$	std $(e_{\mathbf{r}}^{\text{rel\%}})$
Low IMU noise	0.00365	0.01019	0.00031	0.00032
Medium IMU noise	0.02447	0.02554	0.00161	0.00176
High IMU noise	0.04565	0.04534	0.00336	0.00377

Table 7.1. *Deer Running*: Mean and Standard Deviation of RPE.

	avg $(e_{\mathbf{t}}^{\text{rel\%}})$	std $(e_{\mathbf{t}}^{\text{rel\%}})$	avg $(e_{\mathbf{r}}^{\text{rel\%}})$	std $(e_{\mathbf{r}}^{\text{rel\%}})$
Low IMU noise	0.00994	0.01822	0.00028	0.00031
Medium IMU noise	0.03536	0.04293	0.00142	0.00167
High IMU noise	0.06428	0.07908	0.00292	0.00371

Table 7.2. *Deer VR Slow*: Mean and Standard Deviation of RPE.

	avg $(e_{\mathbf{t}}^{\text{rel\%}})$	std $(e_{\mathbf{t}}^{\text{rel\%}})$	avg $(e_{\mathbf{r}}^{\text{rel\%}})$	std $(e_{\mathbf{r}}^{\text{rel\%}})$
Low IMU noise	0.00759	0.03915	0.00072	0.00078
Medium IMU noise	0.04039	0.12304	0.00379	0.00489
High IMU noise	0.10024	0.25302	0.00772	0.01069

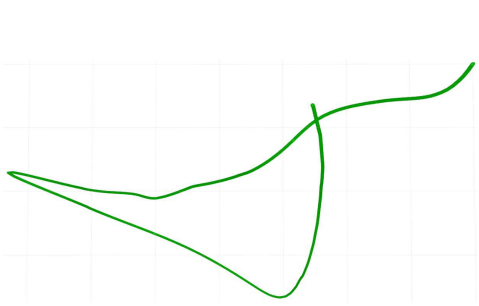
Table 7.3. *ME005*: Mean and Standard Deviation of RPE.

	avg $(e_{\mathbf{t}}^{\text{rel\%}})$	std $(e_{\mathbf{t}}^{\text{rel\%}})$	avg $(e_{\mathbf{r}}^{\text{rel\%}})$	std $(e_{\mathbf{r}}^{\text{rel\%}})$
Low IMU noise	0.00682	0.02501	0.00075	0.00131
Medium IMU noise	0.01867	0.05512	0.00391	0.00708
High IMU noise	0.03349	0.07849	0.00779	0.01357

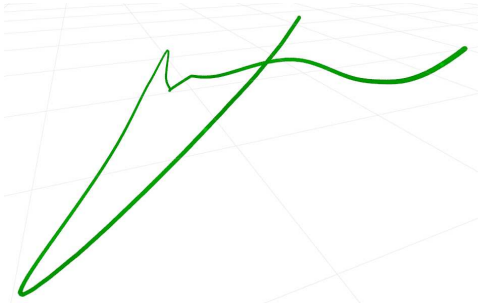
Table 7.4. *ME007*: Mean and Standard Deviation of RPE.

	Overall [Hz]	IMU-Only [Hz]	Camera-Only [Hz]
Deer Running	42.4	169.7	6.1
Deer VR Slow	48.2	173.2	7.2
ME005	44.8	168.1	6.4
ME007	47.4	173.8	7.1
Average	45.7	171.2	6.7

Table 7.5. Average Loop Frequencies: Performance profiling of the proposed system, implemented in Python and evaluated on a laptop with an Intel i7-13700H processor.

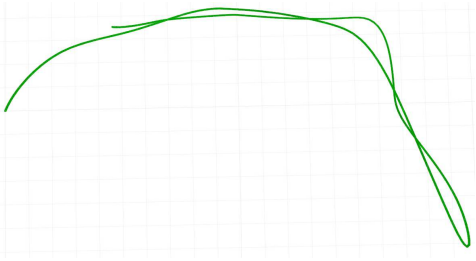


(a) *ME005* Top View

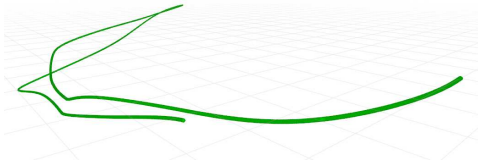


(b) *ME005* Perspective View

(c) *ME005* environment samples



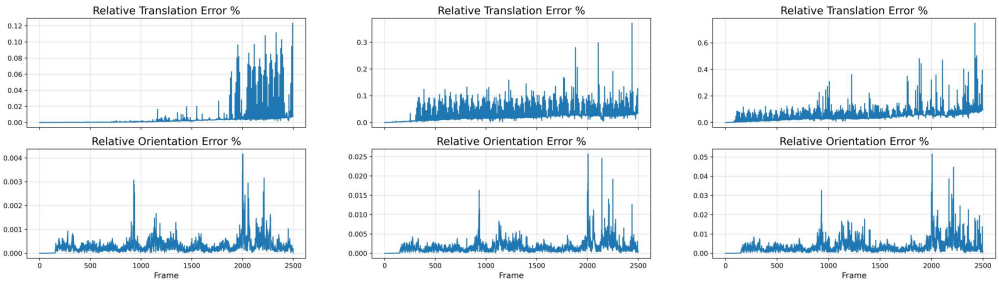
(d) *ME007* Top View



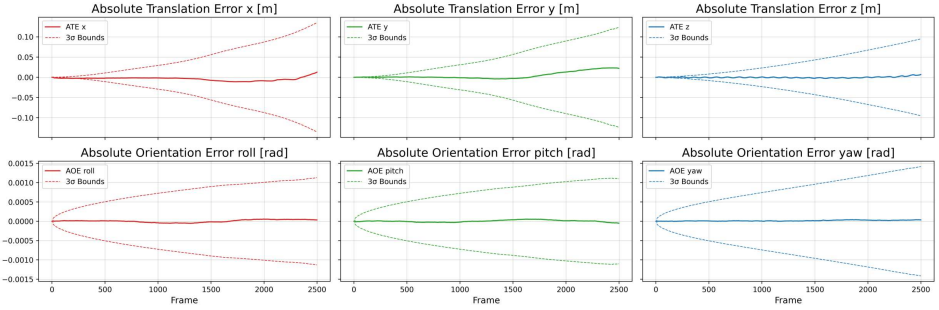
(e) *ME007* Perspective View

(f) *ME007* environment samples

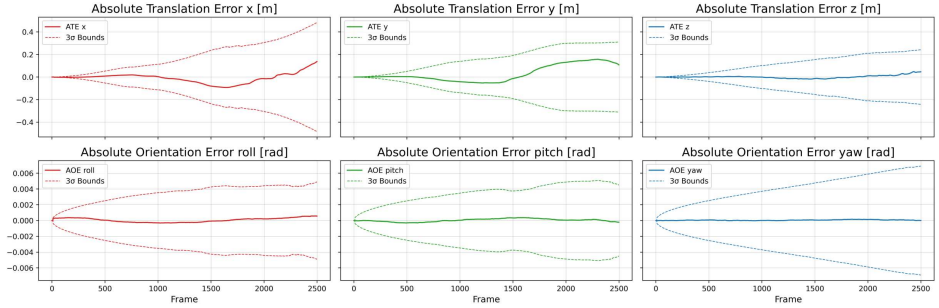
Figure 7.2. *TartanAir: A Dataset to Push the Limits of Visual SLAM* [25] *ME005* and *ME007* sequences. The background grid in images (a) and (b) has a resolution of 1 meter.



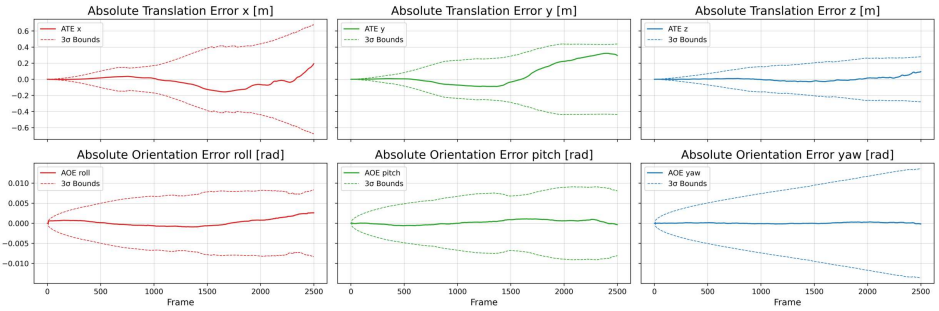
(a) RPE - Low IMU noise. (b) RPE - Medium IMU noise. (c) RPE - High IMU noise.



(d) ATE with 3σ bounds - Low IMU noise.

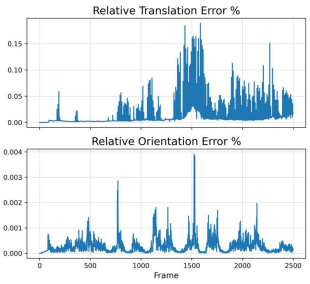


(e) ATE with 3σ bounds - Medium IMU noise.

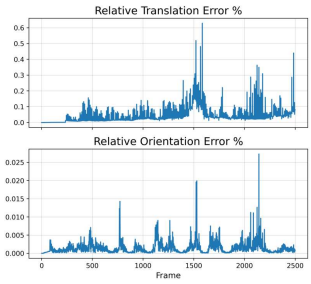


(f) ATE with 3σ bounds - High IMU noise.

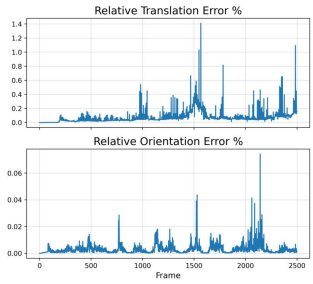
Figure 7.3. *Deer Running*: Relative Pose Error and Absolute Trajectory Error results.



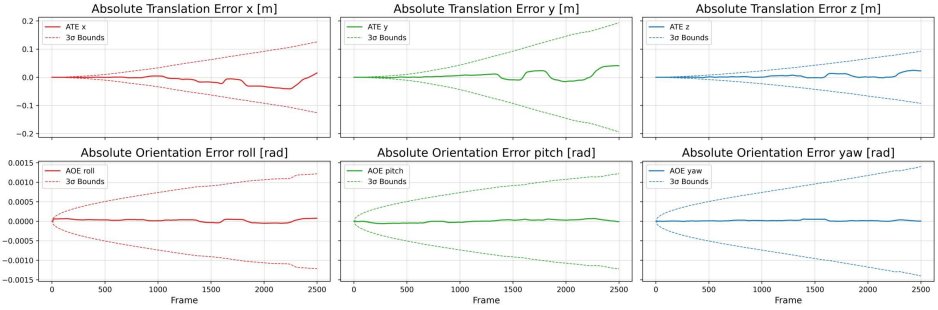
(a) RPE - Low IMU noise.



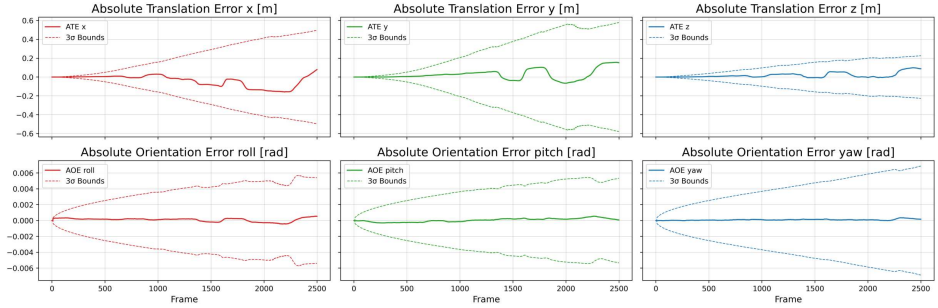
(b) RPE - Medium IMU noise.



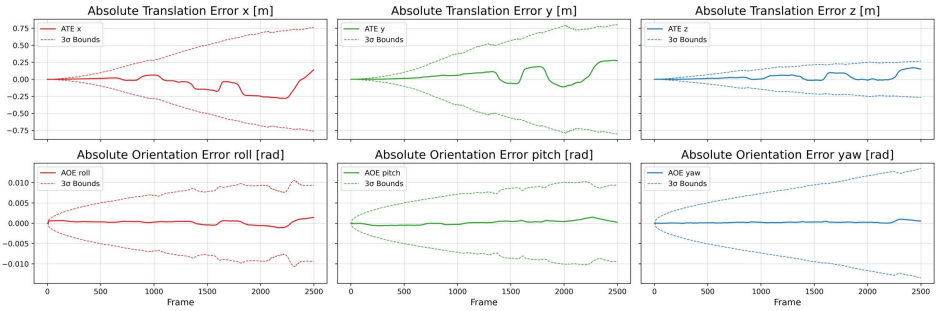
(c) RPE - High IMU noise.



(d) ATE with 3σ bounds - Low IMU noise.

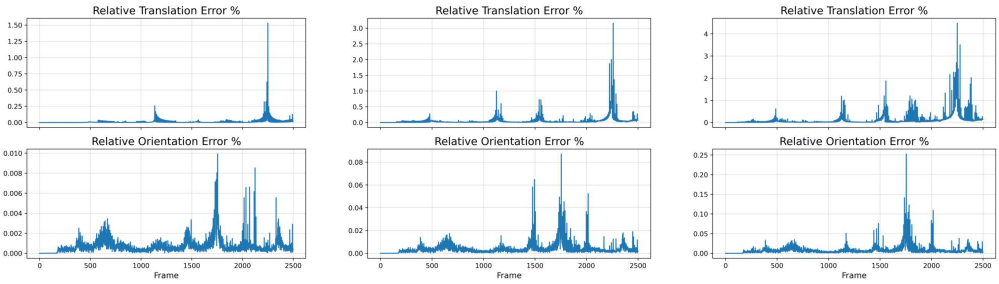


(e) ATE with 3σ bounds - Medium IMU noise.

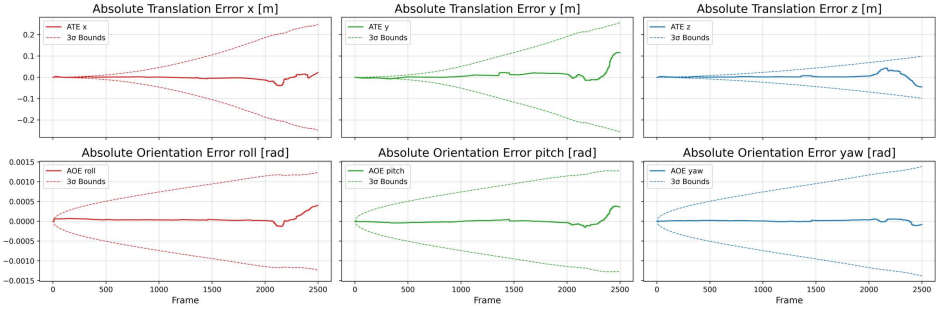


(f) ATE with 3σ bounds - High IMU noise.

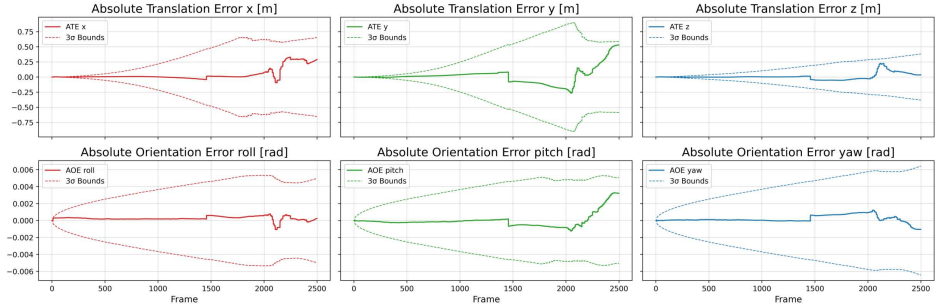
Figure 7.4. *Deer VR Slow*: Relative Pose Error and Absolute Trajectory Error results.



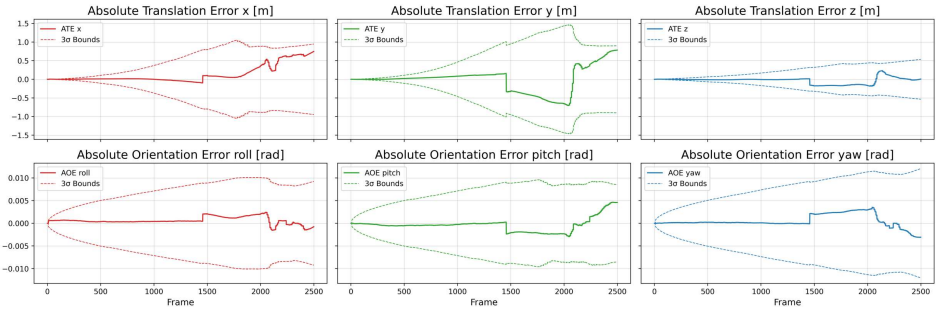
(a) RPE - Low IMU noise. (b) RPE - Medium IMU noise. (c) RPE - High IMU noise.



(d) ATE with 3σ bounds - Low IMU noise.

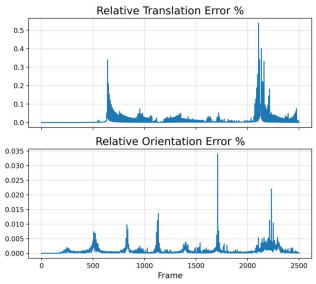


(e) ATE with 3σ bounds - Medium IMU noise.

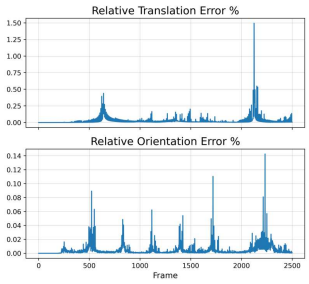


(f) ATE with 3σ bounds - High IMU noise.

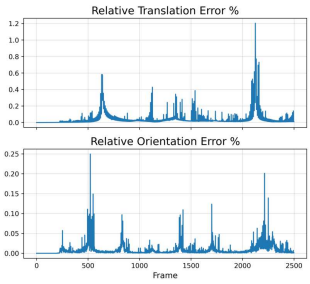
Figure 7.5. *ME005*: Relative Pose Error and Absolute Trajectory Error results.



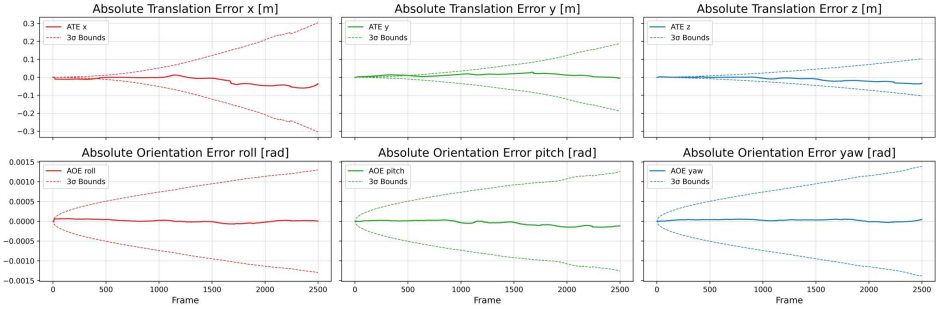
(a) RPE - Low IMU noise.



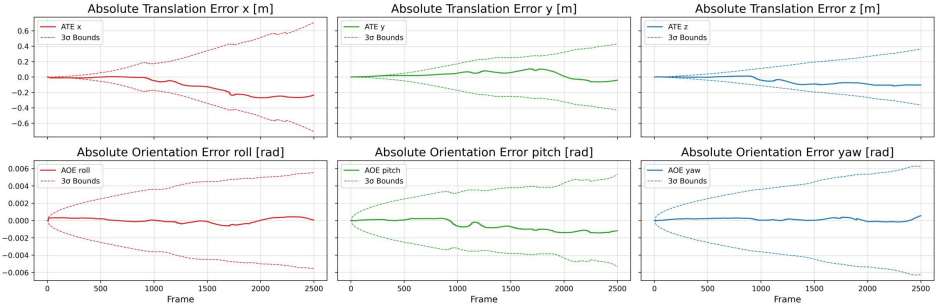
(b) RPE - Medium IMU noise.



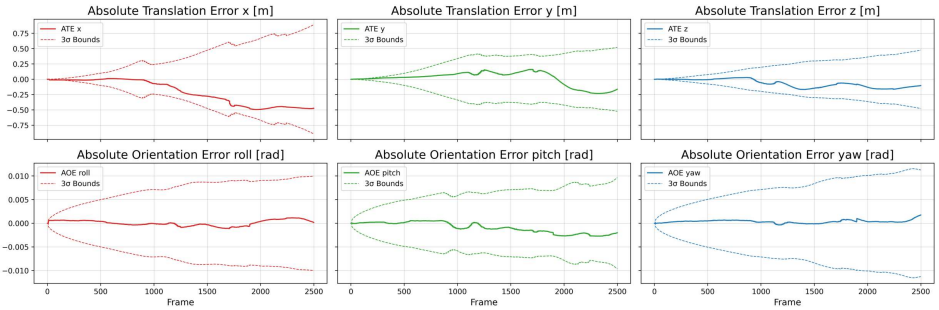
(c) RPE - High IMU noise.



(d) ATE with 3σ bounds - Low IMU noise.



(e) ATE with 3σ bounds - Medium IMU noise.



(f) ATE with 3σ bounds - High IMU noise.

Figure 7.6. *ME007*: Relative Pose Error and Absolute Trajectory Error results.

Chapter 8

Conclusion and Future Work

This thesis presents an extensive study and implementation of the Multi-State Constraint Kalman Filter (MSCKF) for monocular Visual-Inertial Odometry (VIO), together with a detailed theoretical foundation covering projective geometry, Inertial Measurement Unit (IMU) modeling, and Bayesian filtering techniques.

The MSCKF framework demonstrated its effectiveness in achieving accurate real-time pose estimation by tightly integrating visual features and inertial measurements within an Extended Kalman Filter (EKF) based system. To further enhance accuracy in feature extraction and matching, the XFeat library was utilized. XFeat is a novel neural architecture based on Convolutional Neural Network (CNN) that provides a fast, accurate, and hardware-independent solution, significantly improving feature matching performance. Additionally, an epipolar matching refinement step was successfully integrated into the MSCKF pipeline, substantially improving accuracy compared to standard descriptor-based matching without notably increasing computational overhead.

Experimental evaluations across various photo-realistic datasets demonstrated the robustness and adaptability of the proposed approach, consistently achieving good accuracy even under challenging visual conditions such as reflections, motion blur, or high IMU noise levels. Performance profiling highlighted that, despite computational overhead associated with the Python-based implementation, the system successfully achieved real-time performance on standard laptop hardware, suggesting considerable potential for efficiency improvements if ported to languages like C++.

In conclusion, the MSCKF-based algorithm represents a practical solution for precise localization in resource-constrained environments, effectively balancing computational efficiency and estimation accuracy.

Future research directions may include:

- Developing a robust and reliable IMU initialization pipeline that does not solely depend on a V-SLAM system, which is the predominant approach in

state-of-the-art methods;

- Improving data association to enhance robustness in dynamic environments;
- Integrating loop closure detection to further enhance reliability in real-world robotic applications.

Bibliography

[1] D.W. Allan. Statistics of atomic frequency standards. *Proceedings of the IEEE*, 54(2):221–230, 1966.

[2] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Y. Siegwart. Robust visual inertial odometry using a direct ekf-based approach. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 298–304, 2015.

[3] Carlos Campos, Richard Elvira, Juan J. Gómez Rodríguez, José M. M. Montiel, and Juan D. Tardós. Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 37:1874–1890, 2020.

[4] Javier Civera, Andrew J. Davison, and J. M. MartÍnez Montiel. Inverse depth parametrization for monocular slam. *IEEE Transactions on Robotics*, 24(5):932–945, 2008.

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

[6] Jakob J. Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40:611–625, 2016.

[7] Jakob J. Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European Conference on Computer Vision*, 2014.

[8] Christian Forster, Luca Carlone, Frank Dellaert, and Davide Scaramuzza. On-manifold preintegration for real-time visual–inertial odometry. *IEEE Transactions on Robotics*, 33:1–21, 2015.

[9] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Paul Huang. Openvins: A research platform for visual-inertial estimation. *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4666–4672, 2020.

[10] Giorgio Grisetti, Rainer Kümmerle, C. Stachniss, and Wolfram Burgard. A tutorial on graph-based slam. *IEEE Intelligent Transportation Systems Magazine*, 2:31–43, 2010.

[11] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003.

[12] Georg S. W. Klein and David William Murray. Parallel tracking and mapping for small ar workspaces. *2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality*, pages 225–234, 2007.

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

[14] Mingyang Li and Anastasios Mourikis. Improving the accuracy of ekf-based visual-inertial odometry. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 828–835, 05 2012.

[15] Hugh Christopher Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

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

[17] Raul Mur-Artal, José M. M. Montiel, and Juan D. Tardós. Orb-slam: A versatile and accurate monocular slam system. *IEEE Transactions on Robotics*, 31:1147–1163, 2015.

[18] Raul Mur-Artal and Juan D. Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2:796–803, 2016.

[19] Guilherme A. Potje, Felipe Cadar, Andre Araujo, Renato Martins, and Erickson Rangel do Nascimento. Xfeat: Accelerated features for lightweight image matching. *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2682–2691, 2024.

[20] Tong Qin, Peiliang Li, and Shaojie Shen. Vins-mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34:1004–1020, 2017.

[21] Arturo Rankin, Mark Maimone, Jeffrey Biesiadecki, Nikunj Patel, Dan Levine, and Olivier Toupet. Mars curiosity rover mobility trends during the first 7 years. *Journal of Field Robotics*, 38(5):759–800, 2021.

[22] Sajad Saeedi, Eduardo D C Carvalho, Wenbin Li, Dimos Tzoumanikas, Stefan Leutenegger, Paul H. J. Kelly, and Andrew J. Davison. Characterizing visual localization and mapping datasets. *2019 International Conference on Robotics and Automation (ICRA)*, pages 6699–6705, 2019.

[23] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo Jose Taylor, and Vijay R. Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3:965–972, 2017.

[24] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic robotics*. MIT Press, Cambridge, Mass., 2005.

[25] Wenshan Wang, Delong Zhu, Xiangwei Wang, Yaoyu Hu, Yuheng Qiu, Chen Wang, Yafei Hu, Ashish Kapoor, and Sebastian A. Scherer. Tartanair: A dataset to push the limits of visual slam. *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4909–4916, 2020.

[26] Oliver J. Woodman. An introduction to inertial navigation. Technical Report UCAM-CL-TR-696, University of Cambridge, Computer Laboratory, August 2007.