Simultaneous Localization and Mapping: A Rapprochement of Filtering and Optimization-Based Approaches

Chih-Yuan Chiu
S. Shankar Sastry

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-76
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-76.html

May 14, 2021

Acknowledgement

I thank my advisor, Professor Shankar Sastry, and Professor Yi Ma for their guidance and feedback on this project. I would also like to thank my collaborator Amay Saxena for his excellent work on designing simulations to support the theoretical results in this work.

Simultaneous Localization and Mapping: A Rapprochement of Filtering and Optimization-Based Approaches

by Chih-Yuan Chiu

Research Project

Submitted to the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley, in partial satisfaction of the requirements for the degree of **Master of Science, Plan II**.

Approval for the Report and Comprehensive Examination:

Committee:

[illegible]

Professor Shankar Sastry
Research Advisor

May 13, 2021

(Date)

* * * * * * *

Yi Ma

Professor Yi Ma
Second Reader

May 13, 2021

(Date)

Simultaneous Localization and Mapping: A Rapprochement of Filtering and Optimization-Based Approaches

by

Chih-Yuan Chiu

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Master of Science

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor S. Shankar Sastry, Chair
Professor Yi Ma

Spring 2021

Abstract

Simultaneous Localization and Mapping: A Rapprochement of Filtering and Optimization-Based Approaches

by

Chih-Yuan Chiu

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor S. Shankar Sastry, Chair

Simultaneous Localization and Mapping (SLAM) algorithms perform visual-inertial estimation via filtering or batch optimization methods [17]. Empirical evidence suggests that filtering algorithms are computationally more efficient, while optimization methods are more accurate [17, 24]. This paper presents an optimization-based framework that unifies these approaches, and allows the flexible implementation of different design choices, e.g., selecting the number and types of variables maintained in the algorithm at each time. We mathematically prove that filtering methods, e.g., EKF SLAM and MSCKF, correspond to specific design choices in our generalized framework. Finally, we reformulate the MSCKF using our framework, implement the reformulation on challenging image sequences in a baseline SLAM dataset in simulation, and use the proposed re-interpretation to contrast the performance characteristics of the two classes of algorithms. Finally, we describe future extensions of our work to the dynamic SLAM problem and multi-agent planning problems.

Contents

Chapter 1

Introduction

Simultaneous Localization and Mapping (SLAM) concerns a robotic agent in an uncharted or highly dynamic environment that seeks to compose a map of its surroundings while simultaneously locating itself within the constructed map. This fundamental problem appears in a myriad of applications, such as map construction in civilian and military applications, or search-and-rescue missions [5, 3, 6, 7]. To solve a *landmark-based SLAM problem*, one begins by locating identifiable landmarks, and extracting features to construct a map of the environment. The robotic agent then uses its dynamics model, in conjunction with these feature measurements, to pinpoint the position and orientation (pose) of the robotic agent relative to the map. Measurements of new features from newly identified landmarks, and new measurements of old features, can be used to iteratively update the SLAM states—that is, landmark positions and estimates of the robot's pose. This reduces errors in original estimates of landmark positions and the robot pose, due to either measurement noise or fluctuations in the environment.

A typical SLAM algorithm is composed of two components—the front end and the back end. The front end processes raw sensor data into information that facilitates mapping and localization. In particular, the front end performs feature extraction, data association, and outlier rejection, to match features across feature data, filter out spurious feature matches (outliers), process IMU data, and associate all of this information with relevant SLAM states. The processed data values are then supplied to the SLAM back end, which searches for a locally optimal state estimate that best agrees with the processed data. Back end algorithms are typically categorized into two groups—filtering and batch optimization. In filtering methods, propagation and update equations are iteratively applied to refine the probability distributions of a sliding window of recent poses, based on associated measurements [24, 18]. By contrast, batch optimization methods present the optimal pose estimate as the solution to a nonlinear least squares problem, formed by summing the squares of IMU measurement residuals and reproduction error terms. The problem is then solved via linearization techniques, such as Gauss-Newton or Levenberg-Marquardt methods. Empirically, both filtering and batch optimization algorithms have attained state-of-the-art performance, though optimization-based methods often attain higher accuracy at the cost of higher computation cost [3, 17].

This thesis presents a unified optimization-based framework for the SLAM back end, to facilitate the design of new algorithms whose computational speed and performance can flexibly interpolate between those of existing, state-of-the-art algorithms. Although similar results exist in the optimization literature, they do not analyze algorithmic submodules unique to SLAM, e.g., feature incorporation and discarding [1]. First, we illustrate that these algorithms all perform the same three key steps—linearization, optimization, and marginalization—and that mathematically, their only differences lie in the frequency with which each step is performed. From the perspective of our algorithm, filtering approaches are simply nonlinear optimization methods that iteratively optimize recent robot poses, while periodically marginalizing away past features and poses to reduce computational complexity. For example, the Extended Kalman Filter and iterated Extended Kalman Filter aggressively marginalize out all past poses, except the current pose, while retaining all features in the optimization window. [34]. Meanwhile, the Multi-State Constrained Kalman Filter and Fixed Lag Smoother retain a sliding window of recent poses, but marginalize out features no longer observed by the current pose. On the other hand, nonlinear optimization methods may retain a subset of the poses arbitrarily separated across time while marginalizing out all other poses; this is demonstrated by keyframe based methods presented in [25, 17], Alternatively, other batch optimization methods, such as bundle adjustment, graph SLAM, and pose-graph SLAM, opt to maintain and repeatedly re-estimate all poses and feature positions from the past, for increased accuracy. As expected, performing more linearization and optimization steps correspond to higher accuracy at the cost of higher computation time, while marginalizing more often reduces computational cost while sacrificing accuracy.

More concretely:

1. Our primary contribution is the reformulation of state-of-the-art SLAM filtering algorithms in the context of our unified optimization-based framework. In particular, we present the Extended Kalman Filter (EKF) and the Multi-State Constrained Kalman Filter (MSCKF) in terms of our algorithmic structure.

2. Our second contribution is to empirically study the performance of filtering-based and batch optimization-based algorithms on benchmark SLAM datasets in simulation environments. We illustrate the usefulness of this perspective by testing state-of-the-art algorithms, such as MSCKF and sliding window filters, on different datasets.

The remaining chapters are structured as follows. In Chapter 2.1, we formulate the SLAM problem for robotic agents on Euclidean spaces, by presenting the dynamic states and associated dynamics and measurement maps of these systems, as well as the residual costs maintained in the problem. As an example, we pose the states, dynamics, and measurement functions of the Extended Kalman Filter (EKF) within our framework. In Chapter 2.2, we generalize the above results to scenarios where the dynamic states of interest are overparameterized, and constrained to lie on a smooth manifold. As examples, we examine the setup for the Multi-State Constrained Kalman Filter (MSCKF) and Open Keyframe-Based Visual-Inertial SLAM (OK-Vis) algorithms. Next, in Chapter 3, we present the three key steps of our main algorithm—Optimization via Gauss-Newton descent, linear approximation, and marginalization of states. For simplicity, we will first formulate these steps for the Euclidean space formulation of the SLAM problem, before addressing their generalization to the over-parameterized case. We formulate the EKF, MSCKF, and OK-Vis algorithms within our framework in Chapter 4, by illustrating that the steps of these algorithms represent iterations of our three key steps at different frequencies. Chapter 5 demonstrates the experimental use of our novel framework. Here, we interpolate the frequency at which these three key steps are taken in state-of-the-art SLAM algorithms to formulate novel SLAM algorithms. We then test these novel SLAM algorithms on simulated datasets, to characterize the tradeoff between estimation accuracy and computational efficiency as a direct consequence of varying the relative frequency at which these three steps are taken. Finally, Chapter 6 summarizes our work and discusses directions for future work.

Chapter 2

SLAM: Setup and Formulation

This chapter describes the mathematical formulation of the SLAM problem from the perspective of nonlinear optimization. The objective of SLAM is to estimate state and feature positions that best enforce constraints posed by the given dynamics and measurement models, as well as noisy state and feature measurements collected over time. This is formally posed as the unconstrained minimization of a sum of a collection of weighted cost terms, which represent constraints imposed by the dynamics and measurement maps. First, in Section 2.1, we describe this approach in detail for scenarios in which all states evolve on Euclidean spaces, e.g., for a robot constrained to move on a 2D plane. Then, in Section 2.2, we generalize this formulation to the case where the optimization variables in SLAM lie on a smooth manifold, whose geometric properties may differ significantly from those of Euclidean spaces. To do so, we introduce the *boxplus* and *boxminus* operators on arbitrary smooth manifolds, which are commonly used in the SLAM literature to define cost functions for states defined on smooth manifolds. We also note that, when the manifold in question is a Lie group, the boxplus and boxminus operators become much more straightforward to characterize.

2.1 SLAM: Formulation on Euclidean Spaces

In SLAM, two types of variables are estimated: *states* and *features*. The state at each time t, denoted $x_t \in \mathbb{R}^{d_x}$, encapsulates information describing the robot, e.g., camera positions and orientations (poses). The feature positions available at time t in a global frame, denoted $\{f_{t,j} | j = 1, \cdots, p\} \subset \mathbb{R}^{d_f}$, can be obtained by analyzing information from image measurements $\{z_{t,j} | j = 1, \cdots, p\} \subset \mathbb{R}^{d_z}$ and state estimates; these provide information regarding the relative position of the robot in its environment. States and features are described by a smooth (i.e., infinitely continuously differentiable) dynamics map $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ and a smooth measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$, via additive noise models:

$$x_{t+1} = g(x_t) + w_t, \quad w_t \sim \mathcal{N}(0, \Sigma_w), \tag{2.1}$$

$$z_{t,j} = h(x_t, f_{t,j}) + v_{t,j}, \quad v_{t,j} \sim \mathcal{N}(0, \Sigma_v, \tag{2.2}$$

where $\Sigma_w \in \mathbb{R}^{d_x \times d_x}, \Sigma_w \succeq 0$ and $\Sigma_v \in \mathbb{R}^{d_z \times d_z}, \Sigma_v \succeq 0$.

To perform localization and mapping, SLAM algorithms maintain a *full state (vector)* $\overline{x_t} \in \mathbb{R}^d$, in which a number of past states and feature positions are concatenated. The exact number and time stamps of these states and features vary with the design choice of each SLAM algorithm. For example, sliding-window filters define the full state $\overline{x_t} := (x_{t-n+1}, \cdots, x_t, f_{t,p-q+1}, \cdots, f_{t,p}) \in \mathbb{R}^d$, with $d := d_x n + d_f q$, to be a sliding window of the most recent n states, consisting of one pose each, and the most recent estimates, at time t, of a collection of q features. Batch optimization problems, on the other hand, maintain all states and features ever encountered in the problem up to the current time.

The dynamics and measurement maps above do not involve overparameterized state variables, e.g., quaternion representations for poses. Such discussions are deferred to Section 2.2.

SLAM as an Optimization Problem on Euclidean Spaces

The objective of SLAM is to estimate state and feature positions that best enforce constraints posed by the given dynamics and measurement models, as well as noisy state and feature measurements collected over time. This can be formulated as the optimization problem of minimizing the sum of a collection of weighted residual terms, which represent constraints generated by the dynamics and measurement maps. For example, weighted residuals associated with the prior distribution over $\overline{x_t} \in \mathbb{R}^d$, the dynamics constraints between states $x_i, x_{i+1} \in \mathbb{R}^{d_x}$, and the reprojection error of feature $f_j \in \mathbb{R}^{d_f}$ corresponding to the state $x_i \in \mathbb{R}^{d_x}$ and image measurement $z_{t,j} \in \mathbb{R}^{d_z}$, may be given by $\Sigma_0^{-1/2}(\overline{x_t} - \mu_0) \in \mathbb{R}^d$, $\Sigma_w^{-1/2}\big(x_{i+1} - g(x_i)\big) \in \mathbb{R}^{d_x}$, and $\Sigma_v^{-1/2}\big(z_{i,j} - h(x_i, f_{t,j})\big) \in \mathbb{R}^{d_z}$, respectively (here, $1 \le i \le n-1$ and $1 \le j \le q$). We can then define the running cost, $c : \mathbb{R}^{d_x n + d_f q} \to \mathbb{R}$, as the sum of weighted norm squares of these residuals. For example, for a sliding-window filtering algorithm for SLAM, we may have:

$$\begin{aligned} c(\overline{x_t}) := \|\overline{x_t} - \mu_0)\|^2_{\Sigma_0^{-1}} + \sum_{i=t-n+1}^{t-1} \|x_{i+1} - g(x_i)\|^2_{\Sigma_w^{-1}} \\ + \sum_{j=p-q+1}^{p} \sum_{i=t-n+1}^{t} \|z_{i,j} - h(x_i, f_{t,j})\|^2_{\Sigma_v^{-1}}, \end{aligned} \tag{2.3}$$

where we have defined $\|v\|_A^2 := v^\top A v$ for any real vector v and real matrix A of compatible dimension.

To formulate SLAM as a nonlinear least-squares problem, we stack all residual terms into a single residual vector $C(\overline{x_t})$. For example, for the sliding-window filter above, we have:

$$\begin{aligned} C(\overline{x_t}) := \Big[& \big(\Sigma_0^{-1/2}(\tilde{x}_t - \mu_0)\big)^\top \\ & \big(\Sigma_w^{-1/2}(x_{t-n+1} - g(x_{t-n}))\big)^\top \cdots \big(\Sigma_w^{-1/2}(x_t - g(x_{t-1}))\big)^\top \end{aligned}$$

$$
\begin{aligned}
&\big(\Sigma_v^{-1/2}(z_{t-n+1,p-q+1} - h(x_{t-n+1}, f_{t,p-q+1}))\big)^\top \cdots \\
&\big(\Sigma_v^{-1/2}(z_{t-n+1,p} - h(x_{t-n+1}, f_{t,p}))\big)^\top \cdots \\
&\big(\Sigma_v^{-1/2}(z_{t,p-q+1} - h(x_t, f_{t,p-q+1}))\big)^\top \cdots \\
&\big(\Sigma_v^{-1/2}(z_{t,p} - h(x_t, f_{t,p}))\big)^\top\Big]^\top \\
&\quad \in \mathbb{R}^{(2n-1)d_x + pd_f + nqd_z}.
\end{aligned}
$$

As a result, $c(\overline{x_t}) = C(\overline{x_t})^\top C(\overline{x_t})$, and the SLAM problem is now reduced to the following nonlinear least squares problem:

$$
\min_{\overline{x_t}} . c(\overline{x_t}) = \min_{\overline{x_t}} . C(\overline{x_t})^\top C(\overline{x_t}) \tag{2.4}
$$

Section 3 introduces the main algorithmic submodules used to find an approximate solution to (2.4).

2.2 SLAM: Formulation on Manifolds

In this section, we generalize the SLAM problem formulation presented in Section 2.1 to situations where the dynamical states under study are defined on smooth manifolds, rather than on Euclidean spaces. This is necessary, since SLAM problems often involve the orientations of rigid bodies, which evolve on a smooth manifold embedded in an ambient space, e.g., rotation matrices expressed as unit quaternions. In such situations, we define *boxplus* ($\boxplus$) and *boxminus* ($\boxminus$) operators, to perform composition and difference operations in the iterative SLAM algorithm presented in Section 3, while enforcing constraints imposed by the manifold's geometric structure. The following subsections define $\boxplus$ and $\boxminus$ on general manifolds, and on specific types of manifolds relevant to the SLAM problem.

Box Operators on Manifolds

Suppose the full state $x \in \mathcal{M}$ evolves on an n dimensional smooth manifold $\mathcal{M}$. For each $x \in \mathcal{M}$, let $\pi_x : U_x \to V_x$ a homeomorphic chart from an open neighborhood U_x of $x \in \mathcal{M}$ to an open neighborhood $V_x \subset \mathbb{R}^n$ of 0 in $\mathbb{R}^n$. Without loss of generality, suppose $\pi_x(x) = 0$. The operators $\boxplus : U_x \times V_x \to U_x$ and $\boxminus : U_x \times U_x \to V_x$ are defined as follows:

$$
x \boxplus \delta = \pi_x^{-1}(\delta) \tag{2.5}
$$

$$
y \boxminus x = \pi_x(y) \tag{2.6}
$$

In essence, $\boxplus$ adds a perturbation $\delta \in \mathbb{R}^n$, in local coordinates, to a state $x \in M$, while $\boxminus$ extracts the perturbation $\delta \in \mathbb{R}^n$, in local coordinates, between states $x, x' \in M$ covered by the same chart. In subsequent sections, δ often describes an error or increment to a nominal state on the manifold.

The SLAM problem on manifolds concerns the minimization of a smooth function $f : \mathcal{M} \to \mathbb{R}$ via iterative descent, starting from an initial state $x_0 \in \mathcal{M}$. Let $x_k \in \mathcal{M}$ denote the candidate solution at iteration k, and define $\widehat{f}_{x_k} := f \circ \pi_{x_k}^{-1} : \mathbb{R}^n \to \mathbb{R}$ to be the coordinate representation of f w.r.t the coordinate map π_{x_k}. Since $\widehat{f}_{x_k}$ is smooth function between Euclidean spaces, we can compute a direction $\delta \in \mathbb{R}^n$ along which $\widehat{f}_{x_k}$ locally decreases the most. The update law is then:

$$x_{k+1} \leftarrow x_k \boxplus \delta \tag{2.7}$$

In words, the iterative algorithm finds a direction around x_k along which $\widehat{f}_{x_k}$ locally decreases the most, moves along it in local coordinates, and projects the result back onto $\mathcal{M}$.

When $\mathcal{M}$ is a *Lie Group*, only the coordinate chart $\pi_0 : \mathcal{M} \to \mathbb{R}^n$ on the identity element $I \in \mathcal{M}$ needs to be specified. This chart provides a natural choice for any other chart $\pi_x : \mathcal{M} \to \mathbb{R}^n$, by using the group multiplication operator $\circ : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$:

$$\pi_x(\delta) = \pi_0(x^{-1} \circ \delta), \qquad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n, \tag{2.8}$$

$$\pi_x^{-1}(\delta) = x \circ \pi_0^{-1}(\delta), \qquad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n. \tag{2.9}$$

On Lie groups, $\boxplus$ and $\boxminus$ can then be defined by:

$$x \boxplus \delta = x \circ \pi_0^{-1}(\delta), \qquad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n, \tag{2.10}$$

$$y \boxminus x = \pi_0(x^{-1} \circ y), \qquad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n. \tag{2.11}$$

defined in terms of the single map π_0. Lie groups, e.g., SO(3), occur commonly in the study of SLAM.

Manifold Examples

This subsection gives examples of the $\boxplus$, $\boxminus$ and π operators for manifolds that occur widely in SLAM: the space of unit quaternions, $\mathbb{H}_u$, and the space of rotation matrices, $SO(3)$.

1. **Unit Quaternions**: Each $q \in \mathbb{H}_u$ is expressed as $q = (q_u, \vec{q}_v)$ where $q_u \in \mathbb{R}$ and $\vec{q}_v \in \mathbb{R}^3$ denote the scalar part and the vector (imaginary) part, respectively, with $\|q\| = \sqrt{q_u^2 + \|\vec{q}_v\|^2} = 1$ (JPL convention). Here, the coordinate map $\pi : \mathbb{H}_u \to \mathbb{R}^3$ is defined as the Log map on $\mathbb{H}_u$, and its inverse π^{-1} is defined as the Exp map. Specifically, we first rewrite each $q \in \mathbb{H}_u$ as $q = (\cos(\theta/2),\ \sin(\theta/2)\vec{\omega})$ for some $\theta \in [0, \pi]$, $\vec{\omega} \in \mathbb{R}^3$ with $\|\vec{\omega}\| = 1$, i.e., the quaternion q implements a rotation about the unit axis $\vec{\omega}$ by θ radians counter-clockwise. Then, $\pi : \mathbb{H}_u \to \mathbb{R}^3$, and its inverse $\pi^{-1} : B_\pi(0) \to \mathbb{H}_u$ are defined as follows. (Here, $B_\pi(0) := \{x \in \mathbb{R}^3 : \|x\|_2 < \pi\}$ denotes the image of the map π.)

$$\pi(q) = \text{Log}(q) = \theta\vec{\omega},$$

$$\pi^{-1}(\theta\vec{\omega}) = \text{Exp}(\theta\vec{\omega}) = (\cos(\theta/2),\ \sin(\theta/2)\vec{\omega}). \tag{2.12}$$

The $\boxplus$ and $\boxminus$ maps are then implemented via the standard quaternion product $\star$: $\mathbb{H}_u \times \mathbb{H}_u \to \mathbb{H}_u$:

$$q_a \boxplus \vec{\omega} = q_a * \text{Exp}(\vec{\omega}) \tag{2.13}$$

$$q_a \boxminus q_b = \text{Log}(q_b^{-1} * q_a) \tag{2.14}$$

2. **The Rotation Group** $SO(3)$: Definitions of $\boxplus$ and $\boxminus$ for $SO(3)$ parallel those for quaternions, i.e., the coordinate map $\pi : SO(3) \to \mathbb{R}^3$ is defined as the Log operator on $SO(3)$, with output restricted to $B_\pi(0)$. Define:

$$R_a \boxplus \vec{\omega} = R_a * \text{Exp}(\vec{\omega}) \tag{2.15}$$

$$R_a \boxminus R_b = \text{Log}(R_b^T R_a) \tag{2.16}$$

3. **Cartesian Products of Manifolds**: Often, the full state maintained in the SLAM algorithm is defined on the Cartesian product of a finite collection of manifolds, since it contains poses and features which exist and evolve on their own manifolds. For a product manifold $M = M_1 \times M_2$, with projection, increment, and difference maps already defined on M_1 and M_2, we can define $\boxplus$ and $\boxminus$ on M by:

$$(g_1, g_2) \boxplus (\xi_1, \xi_2) = (g_1 \boxplus \xi_1, g_2 \boxplus \xi_2) \tag{2.17}$$

$$(g_1, g_2) \boxminus (h_1, h_2) = (g_1 \boxminus h_1, g_2 \boxminus h_2) \tag{2.18}$$

SLAM on Manifolds: Dynamics and Measurement Maps

To formulate SLAM on a manifold, we must alter our definitions of the state variables, features, image positions, dynamics map, and measurement map. Let $\mathcal{X}$ be a smooth manifold of dimension d_x, on which the system state are defined. Similarly, let $\mathcal{F}$ be a smooth manifold of dimension d_f, on which features are defined, and let $\mathcal{Z}$ be the smooth manifold of dimension d_z, on which image measurements are defined (Often, $\mathcal{F} = \mathbb{R}^{d_f}$ and $\mathcal{Z} \in \mathbb{R}^{d_z}$, e.g., with $d_f = 3$ and $d_z = 2$). We then have:

$$\begin{aligned} x_{t+1} &= g(x_t) \boxplus w_t, \qquad && w_t \sim \mathcal{N}(0, \Sigma_w), \\ z_{t,j} &= h(x_t, f_{t,j}) \boxplus v_{t,j}, \qquad && v_{t,j} \sim \mathcal{N}(0, \Sigma_v). \end{aligned}$$

where $x_t \in \mathcal{X}$ denotes the state at time t, $g : \mathcal{X} \to \mathcal{X}$ denotes the discrete-time dynamics map, and $w_t \in \mathbb{R}^{d_x}$ denotes the dynamics noise, with covariance $\Sigma_w \in \mathbb{R}^{d_x \times d_x}$, $\Sigma_w \succ 0$. Moreover, $f_{t,j} \in \mathcal{F}$ denotes feature position j estimated at the camera pose at time t, $z_{t,j} \in \mathcal{Z}$ denotes the image measurement of feature j measured from the camera pose at time t, $h : \mathcal{X} \times \mathcal{F} \to \mathcal{Z}$ denotes the measurement map, and $v_t \in \mathbb{R}^{d_z}$ denotes the measurement noise, with covariance $\Sigma_v \in \mathbb{R}^{d_z \times d_z}$, $\Sigma_v \succ 0$.

As before, SLAM concerns an optimization problem over a collection of poses and features, e.g., a sliding window of the most recent poses in the states $\{x_i \in \mathcal{X} | i = t-n+1, \cdots, t\}$ and features $\{f_{t,j} \in \mathcal{F} | j = p - q + 1, \cdots, p\}$:

$$\overline{x_t} := (x_{t-n+1}, \cdots, x_t, f_{p-q+1}, \cdots, f_p)$$

$$\in \mathcal{X}^n \times \mathcal{F}^q.$$

We assume that $\overline{x_t}$ is associated with a prior distribution with mean $\mu_0 \in \mathcal{X}^n \times \mathcal{F}^q$ and covariance $\Sigma_0 \in \mathbb{R}^{(nd_x+qd_f)\times(nd_x+qd_f)}$.

SLAM as an Optimization Problem on Manifolds

In this subsection, we interpret the SLAM problem on manifolds as the optimization of a cost function $c : \mathcal{X}^n \times \mathcal{F}^q \to \mathbb{R}$, constructed from residual terms of the same dimension of the minimal coordinates of $\overline{x_t}$, x_t and z_t. In particular, we must generalize (2.3) to the case where the states, dynamics and measurement maps are defined on and between manifolds. This involves replacing + and - operators with $\boxplus$ and $\boxminus$ operators, when necessary. For example, the sliding-window filter window presented in Section 2.1, would be associated with the cost $c : \mathcal{X}^n \times \mathcal{F}^q \to \mathbb{R}$, given by:

$$\begin{aligned} c(\overline{x_t}) = {} & \|\overline{x_t} \boxminus \mu_0\|^2_{\Sigma_0^{-1}} \\ & + \sum_{i=t-n+1}^{t-1} \|x_{i+1} \boxminus g(x_i)\|^2_{\Sigma_w^{-1}}(x_{i+1} \boxminus g(x_i)) \\ & + \sum_{j=p-q+1}^{p} \sum_{i=t-n+1}^{t} \|z_{ij} \boxminus h(x_i, f_{t,j})\|^2_{\Sigma_v^{-1}} \end{aligned}$$

Similar to Section 2.1, we stack all residual terms into a single residual vector $C(\overline{x_t})$. For example, for the sliding-window filter above, we have:

$$\begin{aligned} C(\overline{x_t}) := & \Big[\big(\Sigma_0^{-1/2}(\tilde{x}_t \boxminus \mu_0)\big)^\top \\ & \big(\Sigma_w^{-1/2}(x_{t-n+1} \boxminus g(x_{t-n}))\big)^\top \cdots \big(\Sigma_w^{-1/2}(x_t \boxminus g(x_{t-1}))\big)^\top \\ & \big(\Sigma_v^{-1/2}(z_{t-n+1,p-q+1} \boxminus h(x_{t-n+1}, f_{t,p-q+1}))\big)^\top \cdots \\ & \big(\Sigma_v^{-1/2}(z_{t-n+1,p} \boxminus h(x_{t-n+1}, f_{t,p}))\big)^\top \cdots \\ & \big(\Sigma_v^{-1/2}(z_{t,p-q+1} \boxminus h(x_t, f_{t,p-q+1}))\big)^\top \cdots \\ & \big(\Sigma_v^{-1/2}(z_{t,p} \boxminus h(x_t, f_{t,p}))\big)^\top\Big]^\top \\ & \in \mathbb{R}^{(2n-1)d_x+pd_f+nqd_z}. \end{aligned}$$

As a result, $c(\overline{x_t}) = C(\overline{x_t})^\top C(\overline{x_t})$, and the SLAM problem is now reduced to the following nonlinear least squares problem:

$$\min_{\overline{x_t}} .c(\overline{x_t}) = \min_{\overline{x_t}} .C(\overline{x_t})^\top C(\overline{x_t}) \tag{2.19}$$

Section 3.4 introduces the main algorithmic submodules used to find an approximate solution to (2.19).

To formulate optimization-based SLAM algorithms on manifolds in later sections, it is necessary to compute the derivatives (Jacobians) of smooth functions on manifolds. This is discussed in Section 2.2.

Chapter 3

SLAM: A Generalized Optimization Framework

This chapter introduces a straightforward optimization-based algorithm for solving the least-squares minimization of the SLAM cost-function introduced in Sections 2.1 and 2.2. In particular, we formulate the two key submodules of our generalized SLAM algorithm—Gauss-Newton steps and Marginalization steps—both of which are widely used to solve the SLAM backend inference problem in real time. Since the cost function is highly nonlinear, we initialize the algorithm with a state estimate, typically produced by the SLAM front end [3]. Gauss-Newton steps are then applied to iteratively refine this estimate, until convergence to a local minimum is attained. To ensure real-time operation, we periodically eliminate variables from the cost function, and thus from the optimization problem, using the Marginalization submodule. Finally, we generalize the above algorithmic submodules to the case where the state space is a smooth manifold, but not necessarily an Euclidean space. In Chapter 5, we will present an example of such a system, when we describe our implementation and present simulation results on real data.

3.1 Algorithm Overview

In this section, we describe in detail the submodules of a straightforward, general SLAM algorithm, using the state variables and cost terms defined in Sections 2.1 and 2.2. As before, we first focus on the case where the state space is Euclidean. As before, denote the state and concatenated cost vector by $\overline{x_t} \in \mathbb{R}^d$ and $C : \mathbb{R}^d \to \mathbb{R}^{d_C}$, respectively. (e.g., the sliding window filter in Section 2.1 would correspond to $d = d_x n + d_f q$ and $d_C = (2n-1)d_x + qd_f + nqd_z$). Recall from Chapter 2.1 that the SLAM problem is equivalent to solving the nonlinear least-squares problem (2.4), reproduced below:

$$\min_{\overline{x_t}} . c(\overline{x_t}) = \min_{\overline{x_t}} . \|C(\overline{x_t})\|_2^2.$$

Construction of the Optimization Problem

:

The first step of the algorithm is to construct the objective function whose minimization defines the SLAM problem. From available visual and inertial measurements, we construct and concatenate a collection of residual terms to form a residual vector $C(\overline{x_t})$ of the form introduced in Section 2.1. The SLAM problem is then equivalent to solving the nonlinear least-squares problem (2.4), reproduced below with the indices considered here:

$$\min_{\overline{x_t}} .c(\overline{x_t}) = \min_{\overline{x_t}} .C(\overline{x_t})^\top C(\overline{x_t}).$$

Recall that $\overline{x_t} \in \mathbb{R}^{d_x n + d_f q}$ consists of the most recent n poses and position estimates of the q most salient features, measured at the most recent pose (at time t). Recall also that $C : \mathbb{R}^{d_x n + d_f q} \to \mathbb{R}^{d_x n + d_f q + d_x(n-1) + d_z nq}$ is defined by:

$$C(\overline{x_t}) := \Big[C_0(\overline{x_t})^\top \quad C_{g_{t-n+1}}^\top \quad \cdots \quad C_{g_{t-n+1}}^\top \quad C_{h_{p-q+1,t-n+1}}^\top \quad \cdots \quad C_{h_{t,p-q+1}}^\top \quad \cdots \\ C_{h_{t-n+1,p}}^\top \quad \cdots \quad C_{h_{t,p}}^\top \Big]^\top.$$

with weighted residual terms $C_0 : \mathbb{R}^{d_x n + d_f q} \to \mathbb{R}^{d_x n + d_f q}$, and $C_{g_i} : \mathbb{R}^{2d_x} \to \mathbb{R}^{d_x}$, $C_{h_{ij}} : \mathbb{R}^{d_x + d_f} \to \mathbb{R}^{d_z}$ for each $i \in \{t-n+1, \cdots, t-1\}$ and $j \in \{p-q+1, \cdots, p\}$ given by:

$$C_0(\overline{x_t}) := \Sigma_0^{-1/2}(\overline{x_t} - \mu_0), \tag{3.1}$$

$$C_{g_i}(x_i, x_{i+1}) := \Sigma_w^{-1/2}\big(x_{i+1} - g(x_i)\big), \tag{3.2}$$

$$C_{h_{ij}}(x_i, f_{t,j}) := \Sigma_v^{-1/2}\big(z_{i,j} - h(x_i, f_{t,j})\big). \tag{3.3}$$

Gauss-Newton Descent and Linear Approximation

:

The next step is the recursive update of the joint state $\overline{x_t} \in \mathbb{R}^{d_x n + d_f q}$ by performing Gauss-Newton descent steps using the cost function. More precisely, let $J \in \mathbb{R}^{(d_x n + d_f q + d_x(n-1) + d_z q) \times (d_x n + d_f q)}$ denote the Jacobian of C with respect to $\overline{x_t}$. Starting from an initial estimate $\overline{x_t}^{(0)} \in \mathbb{R}^{d_x n + d_f q}$ of $\overline{x_t}$, we recursively update iterates of our estimate $\{\overline{x_t}^{(k)} | k \geq 0\}$ via the Gauss-Newton algorithm:

$$\overline{x_t}^{(k+1)} \leftarrow \overline{x_t}^{(k)} - (J^T J)^{-1} J^T C(\overline{x_t}^{(k)}). \tag{3.4}$$

These Gauss-Newton steps are iteratively applied until the current iterate $\overline{x_t}^\star := \overline{x_t}^{(k)}$, for some sufficiently large $k \in \mathbb{N}$, is believed to correspond to a sufficiently small cost $c(\overline{x_t}^\star)$. Then, it is fixed, and all or part of the original SLAM optimization problem is replaced with the following linear least squares optimization problem (2.4):

$$\min_{\overline{x_t}} .c(\overline{x_t}) = \min_{\overline{x_t}} .C(\overline{x_t})^\top C(\overline{x_t})$$

$$
\begin{aligned}
&= \min_{\overline{x_t}} . \left[(\overline{x_t} - \mu)^\top \Sigma^{-1} (\overline{x_t} - \mu) + o(\overline{x_t} - \overline{x_t}^\star) \right] \\
&\approx \min_{\overline{x_t}} . (\overline{x_t} - \mu)^\top \Sigma^{-1} (\overline{x_t} - \mu) \qquad (3.5)
\end{aligned}
$$

where $\mu \in \mathbb{R}^{d_x n + d_f q}$ and $\Sigma \in \mathbb{R}^{(d_x n + d_f q) \times (d_x n + d_f q)}$ are given by:

$$
\begin{aligned}
\mu &\leftarrow \overline{x_t}^\star - (J^\top J)^{-1} J^\top C(\overline{x_t}^\star) \\
\Sigma &\leftarrow (J^\top J)^{-1}.
\end{aligned}
$$

Remark 3.1.1. *Alternatives to the Gauss-Newton algorithm, for computing incremental improvements to the initial guess are available, exist in abundance. These include the standard gradient descent algorithms [29], the Levenberg-Marquardt algorithm [21], or Powell's dog leg method [27, 20]. However, we focus on the Gauss-Newton algorithm because, as illustrated above, it has a meaningful interpretation in the case of filtering based SLAM algorithms. Moreover, for well-conditioned problems where a good initial estimate is available, Gauss-Newton tends to be faster than other methods [26].*

Marginalization

:

The *marginalization* step reduces the number of variables present in the SLAM problem to reduce computation time. In the context of the above setup, these variables are estimates of the n pose positions and q feature positions, encapsulated in the overall state $\overline{x_t}$, which are selected to optimize the overall cost $c(\overline{x_t})$. Intuitively, the marginalization procedure involves the following steps to reduce the number of pose and feature position estimates in our optimization problem. First, we partition the overall state $\overline{x_t} \in \mathbb{R}^{d_x n + d_f q}$ into *marginalized* and *non-marginalized* components. Likewise, we rewrite the overall cost $c(\overline{x_t}) \in \mathbb{R}$ as the sum of two cost terms, one of which depends only on the non-marginalized components, while the other depends on both the marginalized and non-marginalized components. Finally, the marginalization step is completed by approximating the latter cost term as an explicit function of the non-marginalized cost term. This process is described mathematically below.

Among the n poses and p features present in the overall state $\overline{x_t}$, let the marginalized state $\overline{x_M} \in \mathbb{R}^{d_x M_x + d_f M_f}$ encapsulate the M_x pose positions and M_f feature positions that we wish to discard from our optimization problem, where $1 \leq M_x \leq n - 1$ and $1 \leq M_f \leq q - 1$, and collect the remaining pose and feature position estimates into the non-marginalized state $\overline{x_K} \in \mathbb{R}^{d_x (n - M_x) + d_f (q - M_f)}$. The only state components kept in the optimization problem after marginalization, encapsulated in $\overline{x_K}$, are poses and feature position estimates that are sufficiently recent or informative to be considered irreplaceable in the optimization problem.

Next, we wish to approximate $c(\overline{x_t})$ using a cost function that depends entirely on $\overline{x_K}$. To do this, we first recall that $c(\overline{x_t})$ is the sum of squared residual terms. By collecting all terms which depend only on the non-marginalized state components $\overline{x_K}$, we can rewrite $c(\overline{x_t})$ as the sum of two costs:

$$
c(\overline{x_t}) = c(\overline{x_K}, \overline{x_M}) = c_1(\overline{x_K}) + c_2(\overline{x_K}, \overline{x_M})
$$

$$= C_1(\overline{x_K})^\top C_1(\overline{x_K}) + C_2(\overline{x_K}, \overline{x_M})^\top C_2(\overline{x_K}, \overline{x_M}),$$

where $c_1 : \mathbb{R}^{d_x(n-M_x)+d_f(q-M_f)} \to \mathbb{R}$ describes the sum of squared residuals in $c(\overline{x_t})$ with no dependence on $\overline{x_M}$, and $C_1 : \mathbb{R}^{d_x(n-M_x)+d_f(q-M_f)} \to \mathbb{R}^{d_{c,1}}$ denotes the concatenation of such squared residual terms, i.e., $c_1(\overline{x_K}) = C_1(\overline{x_K})^\top C_1(\overline{x_K})$, while $c_2 : \mathbb{R}^{d_x n + d_f q} \to \mathbb{R}$ and $C_2 : \mathbb{R}^{d_x n + d_f q} \to \mathbb{R}^{d_{c,2}}$ correspond to the remaining terms. The dimensions c_1 and c_2 depend on the specific way in which the state variables in $\overline{x_t}$ and the cost terms in $c(\overline{x_t})$ are partitioned. For example, consider the cost function (2.3):

$$c(\overline{x_t}) = (\overline{x_t} - \mu_0)^\top \Sigma_0^{-1} (\overline{x_t} - \mu_0) + \sum_{i=t-n+1}^{t-1} (x_{i+1} - g(x_i))^\top \Sigma_w^{-1} \big(x_{i+1} - g(x_i)\big)$$
$$+ \sum_{j=p-q+1}^{p} \sum_{i=t-n+1}^{t} \big(z_{i,j} - h(x_i, f_{t,j})\big)^\top \Sigma_v^{-1} \big(z_{i,j} - h(x_i, f_{t,j})\big).$$

Suppose we partition the full state vector by $\overline{x_t} := (\overline{x_K}, \overline{x_M})$, with:

$$\begin{aligned}
\mathcal{M}_x &:= \{t-n+1, \cdots, t-n+M_x\}, && \text{(poses to marginalize)},\\
\mathcal{M}_f &:= \{p-q+1, \cdots, p-q+M_f\}, && \text{(features to marginalize)},\\
\overline{x_K} &:= (x_{t-n+M_x+1}, \cdots, x_t, f_{t,p-q+M_f+1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x(n-M_x)+d_f(p-M_f)},\\
\overline{x_M} &:= (x_{t-n+1}, \cdots, x_{t-n+M_x}, f_{t,p-q+1}, \cdots, f_{t,p-q+M_f}) \in \mathbb{R}^{d_x M_x + d_f M_f}.
\end{aligned}$$

and the cost function by $c(\overline{x_t}) = c_1(\overline{x_K}) + c_2(\overline{x_M, x_K})$, with:

$$\begin{aligned}
C_1(\overline{x_K}) &:= \Big[C_{g_{t-n+M_x}}^\top \ \cdots \ C_{g_{t-1}}^\top \ C_{h_{t-n+M_x+1,p-q+M_f+1,t}}^\top \ \cdots \ C_{h_{t-n+M_x+1,p,t}}^\top \ \cdots\\
&\qquad C_{h_{t,p-q+M_f+1,t}}^\top \ \cdots \ C_{h_{t,p,t}}^\top\Big]^\top \in \mathbb{R}^{d_x(n-M_x)+d_z(n-M_x)(q-M_f)}.\\
C_2(\overline{x_K}, \overline{x_M}) &:= \Big[C_0(\overline{x_t})^\top \ C_{g_{t-n+1}}^\top \ \cdots \ C_{g_{t-n+M_x-1}}^\top \ C_{h_{t-n+1,p-q+1,t}}^\top \ \cdots \ C_{h_{t-n+1,p,t}}^\top \ \cdots\\
&\qquad C_{h_{t-n+M_x,p-q+1,t}}^\top \ \cdots \ C_{h_{t-n+M_x,p}}^\top \ C_{h_{t-n+M_x+1,p-q+1,t}}^\top \ \cdots \ C_{h_{t-n+M_x+1,p-q+M_f,t}}^\top\\
&\qquad C_{h_{t,p-q+1,t}} \cdots C_{h_{t,p-q+M_f,t}}\Big]^\top \in \mathbb{R}^{d_x n + d_f q + d_x(M_x-1)+d_z(M_x q + M_f n - M_x M_f)},\\
c_1(\overline{x_K}) &= C_1(\overline{x_K})^\top C_1(\overline{x_K}) \in \mathbb{R},\\
c_2(\overline{x_K}, \overline{x_M}) &= C_2(\overline{x_K} \in \mathbb{R}, \overline{x_M})^\top C_2(\overline{x_K}, \overline{x_M}) \in \mathbb{R},
\end{aligned}$$

where C_0 and each C_{g_i} and $C_{h_{ij,t}}$ are given by (3.1), (3.2), and (3.3), respectively. Note that in this case, the dimensions of C_1 and C_2 are given by $d_{c,1} = d_x(n-M_x) + d_z(n-M_x)(q-M_f)$ and $d_{c,2} = d_x n + d_f q + d_x(M_x - 1) + d_x(M_x q + M_f n - M_x M_f)$, respectively.

The SLAM problem can now be written as:

$$\min_{\overline{x_t}} . c(\overline{x_t}) = \min_{\overline{x_K}} . \Big[C_1(\overline{x_K})^\top C_1(\overline{x_K}) + \min_{\overline{x_M}} . C_2(\overline{x_K}, \overline{x_M})^\top C_2(\overline{x_K}, \overline{x_M})\Big]$$

To complete the marginalization step, we replace the output of the inner minimization, $\min_{\overline{x_M}} . C_2(\overline{x_K}, \overline{x_M})^\top C_2(\overline{x_K}, \overline{x_M})$, with an explicit function of $\overline{x_K}$. To do so, we apply first-order approximation to the cost term $C_2(\overline{x_K}, \overline{x_M})^\top C_2(\overline{x_K}, \overline{x_M})$ to obtain:

$$C_2(\overline{x_K}, \overline{x_M}) = C_2(\overline{x_K^\star}, \overline{x_M^\star}) + \begin{bmatrix} J_K^\top & J_M^\top \end{bmatrix} \begin{bmatrix} \overline{x_K} - \overline{x_K^\star} \\ \overline{x_M} - \overline{x_M^\star} \end{bmatrix} + o(\overline{x_K} - \overline{x_K^\star}, \overline{x_M} - \overline{x_M^\star}), \tag{3.6}$$

where $J_K \in \mathbb{R}^{c_2 \times (d_x(n-M_x)+d_f(q-M_f))}$ denotes the Jacobian of C_2 with respect to $\overline{x_K}$ and $J_M \in \mathbb{R}^{c_2 \times (d_x M_x + d_f M_f)}$ denotes the Jacobian of C_2 with respect to $\overline{x_M}$. Using (3.6) to approximate the inner minimization in the above optimization problem, we arrive at the approximate optimization problem below, which depends only on the non-marginalized state $\overline{x_K}$:

$$\begin{aligned} \min_{\overline{x_t}} . c(\overline{x_t}) &= \min_{\overline{x_K}} . \Big[C_1(\overline{x_K})^\top C_1(\overline{x_K}) + \min_{\overline{x_M}} . C_2(\overline{x_K}, \overline{x_M})^\top C_2(\overline{x_K}, \overline{x_M}) \Big] \\ &\approx \min_{\overline{x_K}} . \Big(C_1(\overline{x_K})^\top C_1(\overline{x_K}) + (\overline{x_K} - \mu_K)^\top \Sigma_K^{-1} (\overline{x_K} - \mu_K)) \Big) \end{aligned} \tag{3.7}$$

where the algorithm defines the mean $\mu_K \in \mathbb{R}^{d_x(n-M_x)+d_f(q-M_f)}$ and the covariance matrix $\Sigma_K \in \mathbb{R}^{(d_x(n-M_x)+d_f(q-M_f)) \times (d_x(n-M_x)+d_f(q-M_f))}$ of $\overline{x_K}$ by assigning:

$$\mu_K \leftarrow \overline{x_K^\star} - \Sigma_K J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] C_2(\overline{x^\star}) \tag{3.8}$$

$$\Sigma_K \leftarrow \big(J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] J_K \big)^{-1}, \tag{3.9}$$

This paper mathematically demonstrates the optimality of the approximations in (3.6) and (3.7), and examines the implications of varying the frequency in executing the *Gauss-Newton Descent*, *Linear Approximation*, and *Marginalization* steps. In particular, in Section 4, we will interpret a selection of mainstream filtering-based SLAM algorithms as the repeated iteration of the above three steps at different rates. Moreover, in Section 5, we will illustrate that, by varying the frequencies at which each of the above three steps is performed, we can construct novel SLAM algorithms whose accuracy and computational time interpolate smoothly between those of existing algorithms.

In the sections below, we consider the Gauss-Newton Descent, Linear Approximation, and Marginalization steps in detail.

3.2 Gauss-Newton Descent

Gauss-Newton descent involves solving for the minimization of $c(\overline{x_t})$ via Gauss-Newton steps, an iterative linearization method that approximates $c(\overline{x_t})$ about a given linearization point $\overline{x_t}^\star$ by a linear least-squares cost term, i.e.,

$$\min_{\overline{x_t}} . c(\overline{x_t}) = \min_{\overline{x_t}} . \|\overline{x_t} - \mu_t\|^2_{\Sigma_t^{-1}} + o(\overline{x_t} - \overline{x_t}^\star) \tag{3.10}$$

for some $\mu_t \in \mathbb{R}^d$ and $\Sigma_t \in \mathbb{R}^{d \times d}$. The linearization procedure required to obtain $\mu_t \in \mathbb{R}^d$ and $\Sigma_t \in \mathbb{R}^{d \times d}$, as well as the approximation involved, are detailed in the theorem below.

Theorem 3.2.1. *(**Gauss-Newton Step**) Let $\overline{x_t}^\star \in \mathbb{R}^d$ denote a given linearization point, and suppose $J := \frac{\partial C}{\partial \overline{x_t}} \in \mathbb{R}^{d_C \times d}$ has full column rank. Then applying a Gauss-Newton step to the cost $c(\overline{x_t})$, about $\overline{x_t}^\star \in \mathbb{R}^d$ yields the new cost:*

$$c(\overline{x_t}) = \|\overline{x_t} - \mu_t\|^2_{\Sigma_t^{-1}} + o(\overline{x_t} - \overline{x_t}^\star),$$

where $\mu_t \in \mathbb{R}^d$ and $\Sigma_t \in \mathbb{R}^{d \times d}$ are given by:

$$\begin{aligned} \Sigma_t &\leftarrow (J^\top J)^{-1}, \\ \mu_t &\leftarrow \overline{x_t}^\star - (J^\top J)^{-1} J^\top C(\overline{x_t}^\star). \end{aligned}$$

Proof. See Appendix (Section 7.2). □

Algorithm 1: Gauss-Newton Step.

```
Data: Objective $C^\top C$, linearization point $x_t^\star$.
Result: Mean $\mu$, covariance $\Sigma$ after a Gauss-Newton step.
1 $J \leftarrow \frac{\partial C}{\partial x_t}\Big|_{\mu_t}$
2 $\Sigma_t \leftarrow J^\top J$
3 $\mu_t \leftarrow x_t^\star - (J^\top J)^{-1} J^\top C(x_t^\star)$
4 return $\mu_t, \Sigma_t$
```

3.3 Marginalization of States

The marginalization step reduces the state dimension in our SLAM algorithm, which helps to reduce the computation time. First, we partition the overall state $\overline{x_t} \in \mathbb{R}^d$ into a *marginalized component* $\overline{x_{t,M}} \in \mathbb{R}^{d_M}$, to be discarded from $\overline{x_t}$, and a *non-marginalized component* $\overline{x_{t,K}} \in \mathbb{R}^{d_K}$, to be kept ($d = d_K + d_M$.) Then, we partition $c(\overline{x_t})$ into two cost terms: $c_1(\overline{x_{t,K}})$, which depends only on non-marginalized state components, and $c_2(\overline{x_{t,K}}, \overline{x_{t,M}})$ which depends on both marginalized and non-marginalized state components:

$$\begin{aligned} c(\overline{x_t}) = c(\overline{x_K}, \overline{x_M}) &= c_1(\overline{x_K}) + c_2(\overline{x_K}, \overline{x_M}) \\ &= \|C_1(\overline{x_K})\|_2^2 + \|C_2(\overline{x_K}, \overline{x_M})\|_2^2. \end{aligned}$$

Here, $C_1(\overline{x_K}) \in \mathbb{R}^{d_{C,1}}$ and $C_2(\overline{x_K}, \overline{x_M}) \in \mathbb{R}^{d_{C,2}}$ denote the concatenation of residuals associated with $c_1(\overline{x_K})$ and $c_2(\overline{x_K}, \overline{x_M})$ (with $d_C = d_{C,1} + d_{C,2}$). To remove $\overline{x_{t,M}} \in \mathbb{R}^{d_M}$ from the optimization problem , observe that:

$$\min_{\overline{x_t}} c(\overline{x_t}) = \min_{\overline{x_{t,K}}, \overline{x_{t,M}}} \Big(c_1(\overline{x_{t,K}}) + c_2(\overline{x_{t,K}}, \overline{x_{t,M}}) \Big)$$

$$= \min_{\overline{x_{t,K}}} \Big(\|C_1(\overline{x_{t,K}})\|_2^2 + \min_{\overline{x_{t,M}}} \|C_2(\overline{x_{t,K}}, \overline{x_{t,M}})\|_2^2 \Big).$$

To remove $\overline{x_{t,M}}$, it suffices to approximate the solution to the inner minimization problem by a linear least-squares cost, i.e.:

$$\min_{\overline{x_{t,M}}} \|C_2(\overline{x_{t,K}}, \overline{x_{t,M}})\|_2^2 \approx \|\overline{x_{t,K}} - \overline{\mu}_{t,K}\|^2_{\overline{\Sigma}_{t,K}^{-1}}$$

for some $\overline{\mu}_{t,K} \in \mathbb{R}^{d_K}$ and $\overline{\Sigma}_{t,K} \in \mathbb{R}^{d_K \times d_K}$. Since $\|C_2(\overline{x_{t,K}}, \overline{x_{t,M}})\|_2^2$ is in general non-convex, we obtain $\overline{\mu}_{t,K}$ and $\overline{\Sigma}_{t,K}$ by minimizing the first-order Taylor expansion of $\|C_2(\overline{x_{t,K}}, \overline{x_{t,M}})\|_2^2$ about some linearization point, instead of minimizing $\|C_2(\overline{x_{t,K}}, \overline{x_{t,M}})\|_2^2$ directly. Below, Theorem 3.3.1 details the derivation of $\overline{\mu}_{t,K}$ and $\overline{\Sigma}_{t,K}$.

Theorem 3.3.1 (Marginalization Step). *Let $\overline{x_t}^\star \in \mathbb{R}^d$ denote a given linearization point, and suppose $J := \frac{\partial C}{\partial \overline{x_t}} \in \mathbb{R}^{d_C \times d}$ has full column rank. Define $J_K := \frac{\partial C}{\partial \overline{x_{t,K}}} \in \mathbb{R}^{d_C \times d_K}$ and $J_M := \frac{\partial C}{\partial \overline{x_{t,M}}} \in \mathbb{R}^{d_C \times d_M}$. If $C(\overline{x_{t,M}}, \overline{x_{t,K}})$ were a linear function of $\overline{x_t} = (\overline{x_{t,M}}, \overline{x_{t,K}})$, then applying a Marginalization step to the cost $c(\overline{x_t})$, about the linearization point $\overline{x_t}^\star = (\overline{x_{t,K}^\star}, \overline{x_{t,M}^\star}) \in \mathbb{R}^d$ yields:*

$$\min_{\overline{x_{t,M}}} c(\overline{x_{t,K}}, \overline{x_{t,M}}) = \min_{\overline{x_{t,K}}} . \Big(c_1(\overline{x_{t,K}}) + \min_{\overline{x_{t,M}}} c_2(\overline{x_{t,K}}, \overline{x_{t,M}}) \Big), \tag{3.11}$$

where $\Sigma_{t,K} \in \mathbb{R}^{d_K \times d_K}$ and $\mu_{t,K} \in \mathbb{R}^{d_K}$ are given by:

$$\Sigma_{t,K} := \big(J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] J_K \big)^{-1}, \tag{3.12}$$

$$\mu_{t,K} := \overline{x_{t,K}^\star} - \Sigma_{t,K} J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] C_2(\overline{x_t^\star}). \tag{3.13}$$

Proof. See Appendix (Section 7.2). □

```
Algorithm 2: Marginalization
  Data: Objective $f = C^\top C$, vector of variables to marginzlie $x_{t,M}$, linearization
        point $x_t^\star$.
  Result: Mean $\mu_{t,K}$ and covariance $\Sigma_{t,K}$ of $\overline{x_{t,K}^\star}$ of non-margnalized variables.
1 $C \leftarrow$ subvector of $C$ containing entries dependent on $x_M$.
2 $J := \begin{bmatrix} J_K & J_M \end{bmatrix} \leftarrow \Big[ \frac{\partial C}{\partial \overline{x_{t,K}}}\big|_{\overline{x^\star}} \quad \frac{\partial C}{\partial \overline{x_{t,M}}}\big|_{\overline{x^\star}} \Big]$.
3 $\Sigma_{t,K} \leftarrow \big( J_K^\top \big[ I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] J_K \big)^{-1}$
4 $\mu_{t,K} \leftarrow \overline{x_{t,K}^\star} - \Sigma_{t,K} J_K^\top \big[ I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] C(x^\star)$
5 return $\mu_{t,K}, \Sigma_{t,K}$
```

3.4 Main Algorithm on Manifolds

Our SLAM framework, formulated above on Euclidean spaces, can be straightforwardly extended to a formulation on manifolds. This involves using manifold-related concepts in Section 2.2 to modify the Euclidean-space dynamics and measurement maps in Section 2.1, as well as the cost functions, Gauss-Newton steps, and marginalization steps in Sections 3.2, 3.3. In particular, when appropriate, plus and minus operations must be replaced with the $\boxplus$ and boxminus operators.

Recall the states, poses, features, dynamics and measurement maps, and costs defined in Section 2.2. In particular, the cost $c(\overline{x})$ is given by:

$$c(\overline{x}) = C(\overline{x})^\top C(\overline{x}) \tag{3.14}$$

Now, let $\bar{x}^*$ be a chosen linearization point. Let $\widehat{C}_{\bar{x}^*} := C \circ \pi_{\bar{x}^\star}^{-1}$ be the coordinate representation of the function C near $\bar{x}^*$. Recall that $\widehat{C}_{\bar{x}^*}$ is simply a function from one Euclidean space to another. We can now Taylor expand to write:

$$\begin{aligned} C(\bar{x}) &= (C \circ \pi_{\overline{x}^\star}^{-1})\big(\pi_{\overline{x}^\star}(\overline{x})\big) = \widehat{C}_{\bar{x}^\star}(\Delta\overline{\chi}) \\ &= \widehat{C}_{\bar{x}^\star}(0) + J\Delta\overline{\chi} + o(\Delta\overline{\chi}), \end{aligned} \tag{3.15}$$

where $\Delta\overline{\chi} = \overline{x} \boxminus \overline{x}^\star$ and J is the Jacobian of $\widehat{C}_{\bar{x}^*}$ with respect to $\Delta\overline{\chi}$ evaluated at zero. We then apply a modified version of the algorithms from Section 7.2:

Gauss-Newton Descent

:

Gauss-Newton steps update the current linearization point $\bar{x}^{\{k\}}$ to a new linearization point $\bar{x}^{\{k+1\}}$.

$$\overline{x}^{(k+1)} \leftarrow \overline{x}^{(k)} \boxplus \big(- (J^T J)^{-1} J^T C(\overline{x}^{(k)})\big) \tag{3.16}$$

After Gauss-Newton steps have been taken, the linearization point $\overline{x}^\star$ is fixed, and all or part of the original optimization problem, reproduced below:

$$\min_{\overline{x}} \; c(\overline{x}) = \min_{\overline{x}} \; C(\overline{x})^\top C(\overline{x})$$

is replaced with the following linear least squares optimization problem:

$$\min_{\overline{x}} . (\overline{x} \boxminus \mu)^\top \Sigma^{-1} (\overline{x} \boxminus \mu) \tag{3.17}$$

where the algorithm assigns:

$$\begin{aligned} \mu &\leftarrow \overline{x}^\star \boxplus (J^\top J)^{-1} J^\top C(\overline{x}^\star) \\ \Sigma &\leftarrow (J^\top J)^{-1}. \end{aligned}$$

Marginalization

:

Marginalization steps remove variables $\overline{x_M}$ from the optimization problem by applying linear approximation to C—in particular, the optimization problem:

$$\min_{\overline{x}} .c(\overline{x})$$

is approximated by:

$$\min_{\overline{x_K}} .(\overline{x_K} \boxminus \mu_K)^\top \Sigma_K^{-1} (\overline{x_K} \boxminus \mu_K),$$

where the algorithm assigns:

$$\begin{aligned} \mu_K &\leftarrow \overline{x_K^\star} \boxplus \big(- \Sigma_K J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] C_2(\overline{x^\star}) \big) \\ \Sigma_K &\leftarrow \big(J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] J_K \big)^{-1}, \end{aligned}$$

Chapter 4

Filtering as Nonlinear Optimization

This chapter presents the equivalence of filtering and batch optimization based SLAM algorithms, using the Extended Kalman Filter (EKF, in Section 4.1) and Multi-State Constrained Kalman Filter (MSCKF, in Section 4.2), as examples. (For an introduction to the classical formulation of EKF SLAM and MSCKF, please see Appendices 7.3, 6).

First, we demonstrate how popular *filtering-based* SLAM algorithms can be reformulated as the iterative minimization of a nonlinear least squares cost function for different design choices (more precisely, marginalization schemes). We show that the Extended Kalman Filter (EKF SLAM) is equivalent to a sliding window filter of window size 1. Although several versions of this result exist in the literature, the presented proof lays the groundwork for our analysis.

The main result of this section is a demonstration of the equivalence of the popular Multi-State Constrained Kalman Filter (MSCKF) [24] to iterative nonlinear least squares optimization with a specific marginalization scheme. This result implies that the MSCKF can be formulated as an optimization-based algorithm with a specific marginalization scheme, instead of using matrix operations, as in the classical formulation [24]. This, in turn, allows the MSCKF and its variants to be implemented more straightforwardly, while also allowing their empirical performance to be more directly compared with that of sliding-window optimization algorithms. Indeed, due to the user flexibility granted by our generalized optimization-based framework, improvements or modification to the algorithm become trivial to derive and implement. To summarize, we describe the classic sliding window filter, a workhorse of optimization-based SLAM and the motivating application of marginalization to the SLAM problem. Finally, at the end of the chapter, we list a number of popular visual SLAM algorithms and algorithm paradigms and characterize them in terms of descent and marginalization schemes.

4.1 Extended Kalman Filter (EKF)

The EKF SLAM algorithm constantly maintains the EKF full state vector, $\tilde{x}_t := (x_t, f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x+pd_f}$, consisting of the most recent state, $x_t \in \mathbb{R}^{d_x}$, and the most recent position estimates $f_{t,1}, \cdots, f_{t,p} \in \mathbb{R}^{d_f}$ of all features maintained in the EKF full state $\tilde{x}_t$. At initialization $(t = 0)$, no feature has yet been detected $(p = 0)$, and the EKF full state is simply the initial state $\tilde{x}_0 = x_0 \in \mathbb{R}^{d_x}$, with mean $\mu_0 \in \mathbb{R}^{d_x}$ and covariance $\Sigma_0 \in \mathbb{R}^{d_x \times d_x}$. Suppose that at the current time t, the running cost maintained in the optimization formulation of the EKF SLAM algorithm, $c_{EKF,t,0} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}^{d_x+pd_f}$, is:

$$c_{EKF,t,0} = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}},$$

where $\tilde{x}_t := (x_t, f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x+pd_f}$ denotes the EKF full state at time t, with mean $\mu_t \in \mathbb{R}^{d_x+pd_f}$ and covariance $\Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$. First, the *feature augmentation step* appends position estimates of new features $f_{p+1}, \cdots, f_{p+p'} \in \mathbb{R}^{d_f}$ to the EKF full state $\tilde{x}_t$, and updates its mean and covariance. In particular, feature measurements $(z_{t,p+1}, \cdots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}$ are incorporated by adding measurement residual terms to the current running cost $c_{EKF,t,0}$, resulting in a new cost $c_{EKF,t,1} : \mathbb{R}^{d_x+(p+p')d_f} \to \mathbb{R}$:

$$\begin{aligned} &c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1}, \cdots, f_{t,p+p'}) \\ :=\, &\|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\Sigma_v^{-1}}. \end{aligned}$$

In effect, $c_{EKF,t,1}$ appends new feature positions to $\tilde{x}_t$, and constrains it using feature measurements residuals.

The *feature update* step uses measurements of features already described by $\tilde{x}_t$ to update the mean and covariance of $\tilde{x}_t$. More precisely, feature measurements $z_{t,1:p} := (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{d_z p}$, of the p features $f_1, \cdots, f_p$ included in $\tilde{x}_t$, are introduced by incorporating associated measurement residuals to the running cost, resulting in a new cost $c_{EKF,t,3} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$:

$$c_{EKF,t,3}(\tilde{x}_t) := \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\Sigma_v^{-1}}.$$

A Gauss-Newton step is then applied to construct an updated mean $\overline{\mu_t} \in \mathbb{R}^{d_x+pd_f}$ and covariance $\overline{\Sigma}_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$ for $\tilde{x}_t$, resulting in a new cost $c_{EKF,t,4} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$:

$$c_{EKF,t,4}(\tilde{x}_t) := \|\tilde{x}_t - \overline{\mu_t}\|^2_{\overline{\Sigma}_t^{-1}},$$

which returns the running cost to the form of $c_{EKF,t,0}$.

The *state propagation* step propagates the EKF full state forward by one time step, via the EKF state propagation map $g : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}^{d_x+pd_f}$. To propagate $\tilde{x}_t$ forward in time, we incorporate the dynamics residual to the running cost $c_{EKF,t,0}$, thus creating a new cost $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \to \mathbb{R}$:

$$c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) := \|\tilde{x}_t - \overline{\mu_t}\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}.$$

In effect, $c_{EKF,t,5}$ appends the new state $x_{t+1} \in \mathbb{R}^{d_x}$ to $\tilde{x}_t$, while adding new constraints posed by the dynamics residuals. One marginalization step, with $\tilde{x}_{t,K} := (x_{t+1}, f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x+pd_f}$ and $\tilde{x}_{t,M} := x_t \in \mathbb{R}^{d_x}$, is then applied to remove the previous state $x_t \in \mathbb{R}^{d_x}$ from the running cost. This step produces a mean $\mu_{t+1} \in \mathbb{R}^{d_x+pd_f}$ and a covariance $\Sigma_{t+1} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$ for the new EKF full state, $\tilde{x}_{t+1} := \tilde{x}_{t,K}$. Accordingly, the running cost maintained in the optimization framework is updated to $c_{EKF,t+1,0} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$:

$$c_{EKF,t+1,0}(\tilde{x}_{t+1}) := \|\tilde{x}_{t+1} - \mu_{t+1}\|^2_{\Sigma_{t+1}^{-1}},$$

which returns the running cost to the form of $c_{EKF,t,0}$.

The theorems below establish that the feature update, and propagation steps of the EKF, presented above in our optimization framework, correspond precisely to those presented in the standard EKF SLAM algorithm (Algorithm 6) [34] [33].

Theorem 4.1.1. *The feature augmentation step of the standard EKF SLAM algorithm (Alg. 7) is equivalent to applying a Gauss-Newton step to $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, given by:*

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1}, \cdots, f_{t,p+p'}) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\tilde{\Sigma}_v^{-1}}.$$

Proof. See Appendix (Section 6). ☐

Theorem 4.1.2. *The feature update step of the standard EKF SLAM algorithm (Alg. 8) is equivalent to applying a Gauss-Newton step on $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$, given by:*

$$c_{EKF,t,3}(\tilde{x}_t) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\Sigma_v^{-1}}.$$

Proof. See Appendix (Section 6). ☐

Theorem 4.1.3. *The state propagation step of the standard EKF SLAM algorithm (Alg. 9) is equivalent to applying a Marginalization step to $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \to \mathbb{R}$, given by:*

$$c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) = \|\tilde{x}_t - \overline{\mu}_t\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}.$$

with $\tilde{x}_{t,K} := (x_{t+1}, f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{d_x+pd_f}$ and $\tilde{x}_{t,M} = x_t \in \mathbb{R}^d$.

Proof. See Appendix (Section 6). ☐

Algorithm 3: EKF SLAM, as an iterative optimization problem.

```
Data: Prior $\mathcal{N}(\mu_0, \Sigma_0)$ on $x_0 \in \mathbb{R}^{d_x}$, noise covariances $\Sigma_w$, $\Sigma_v$, dynamics map $g$,
      measurement map $h$, time horizon $T$.
Result: Estimates $\hat{x}_t \in \mathbb{R}^{d_x}$, $\forall t \in \{1, \cdots, T\}$.
1  $f_0(x) \leftarrow \|x_0 - \mu_0\|^2_{\Sigma_0^{-1}}$
2  $p \leftarrow 0$.
3  for $t = 0, 1, \cdots T$ do
4      $(z_{t,1}, \cdots, z_{t,p}) \leftarrow$ Measurements of existing features.
5      $\text{cost}_t \leftarrow \text{cost}_t + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_k)\|^2_{\Sigma_v^{-1}}$
6      $\bar{\mu}_t, \bar{\Sigma}_t, \text{cost}_t \leftarrow$ 1 Gauss-Newton step on $\text{cost}_t$, about $\mu_t$, (Alg. 1).
7      $\hat{x}_t \leftarrow \bar{\mu}_t \in \mathbb{R}^{d_x + pd_f}$.
8      $(z_{t,p+1}, \cdots, z_{t,p+p'}) \leftarrow$ Measurements of new features.
9      $\text{cost}_t \leftarrow \text{cost}_t + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_k)\|^2_{\Sigma_v^{-1}}$
10     $\bar{\mu}_t \leftarrow \big(\bar{\mu}_t, \ell(x_t, z_{t,p+1}), \cdots, \ell(x_t, z_{t,p+p'})\big) \in \mathbb{R}^{d_x + (p+p')d_f}$.
11     $\bar{\mu}_t, \bar{\Sigma}_t, \text{cost}_t \leftarrow$ 1 Gauss-Newton step on $\text{cost}_t$, about $\overline{\mu_t}$ (Alg. 1).
12     $p \leftarrow p + p'$
13     if $t < T$ then
14         $\text{cost}_t \leftarrow \text{cost}_t + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}$
15         $\mu_{t+1}, \Sigma_{t+1}, \text{cost}_t \leftarrow$ 1 Marginalization step on $\text{cost}_{t+1}$ with $x_M = x_t$, about
             $(\overline{\mu_t}, g(\overline{\mu_t}))$ (Alg. 2).
16         $\text{cost}_{t+1} \leftarrow \|x_{t+1} - \mu_{t+1}\|^2_{\Sigma_{t+1}^{-1}}$
17     end
18 end
19 return $\hat{x}_0, \cdots, \hat{x}_T$
```

Remark 4.1.1. *The earliest solutions to the SLAM problem were stated and solved in terms of the EKF as the central estimator, with the seminal solution provided by Smith and Cheeseman in 1987 [31]. Solutions based on the EKF are favoured due to their relatively cheap computational cost, and remain a popular choice for real-time state estimation.*

Remark 4.1.2. *In practice, the application of the Gauss-Newton algorithm for the feature augmentation step can be delayed, and instead done in conjunction with the feature update step.*

4.2 Multi-State Constrained Kalman Filter

The MSCKF algorithm constantly maintains a full state, $\tilde{x}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, containing the most recent IMU state, $x_{\text{IMU}} \in \mathcal{X}_{\text{IMU}}$ and n recent poses, $(x_1, \cdots, x_n) \in (\mathcal{X}_p)^n$:

$$\tilde{x}_t := (x_{t,\text{IMU}}, x_1, \cdots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n,$$

with mean $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and covariance $\Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$. As new poses are introduced, old poses are discarded, and features are marginalized to update $\tilde{x}_t$, the mean μ_t, covariance Σ_t, and the integer n accordingly.

At initialization ($t = 0$), no pose has yet been recorded ($n = 0$), and the full state $\tilde{x}_0$ is the initial IMU state $\tilde{x}_{0,\text{IMU}} \in \mathcal{X}_{\text{IMU}}$, with mean $\mu_0 \in \mathcal{X}_{\text{IMU}}$ and covariance $\Sigma_0 \in \mathbb{R}^{d_{\text{IMU}} \times d_{\text{IMU}}}$. In other words, $\tilde{x}_0 = \mu_0$ optimizes the initial instantiation $c_{MSCKF,0} : \mathcal{X}_{\text{IMU}} \to \mathbb{R}$ of the running cost in our framework:

$$c_{MSCKF,0,0}(\tilde{x}_0) = \|\tilde{x}_0 \boxminus \mu_0\|^2_{\Sigma_0^{-1}}.$$

Suppose that at the current time t, the running cost $c_{MSCKF,t,0} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \to \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ is:

$$c_{MSCKF,t,0}(\tilde{x}_t) = \|\tilde{x}_t \boxminus \mu_t\|^2_{\Sigma_t^{-1}},$$

where $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and $\Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$ denote the mean and covariance of the full state $\tilde{x}_t := (x_{t,\text{IMU}}, x_1, \cdots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ at time t, consisting of the current IMU state and n poses. When a new image is received, the *pose augmentation step* adds a new pose $x_{n+1} \in \mathcal{X}^p$ (global frame) to $\tilde{x}_t$, derived from $x_{n+1}^{\text{IMU}} \in \mathcal{X}^p$, the IMU position estimate in the global frame, via the map $\psi : (\mathcal{X}_{\text{IMU}})^2 \times (\mathcal{X}_p)^n \to \mathcal{X}_p$, i.e.,

$$x_{n+1} := \psi(\tilde{x}_t, x_{n+1}^{\text{IMU}}) \in \mathcal{X}_p.$$

The *feature update step* uses features measurements to update the mean and covariance of $\tilde{x}_t$. In the MSCKF, features are discarded if (A) they are not observed in the current pose, or (B) $n \geq N_{\max}$, a specified upper bound, in which case $\lfloor N_{\max} \rfloor/3$ of the n poses are dropped after features common to these poses are marginalized. Let $S_{z,1}$ and $S_{z,2}$ denote the set of pose-feature pairs (x_i, f_j) described in cases (A) and (B) above, respectively, and let S_f denote the set of all features to be marginalized. (Algorithm 10.) These feature constraints are then incorporated into the running cost, resulting in a new cost $c_{MSCKF,t,3} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \to \mathbb{R}$:

$$\begin{aligned} &c_{MSCKF,t,3}(\tilde{x}_t) \\ :=& \|\tilde{x}_t \boxminus \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{(x_i,f_j)\in S_{z,1}\cup S_{z,2}} \|z_{i,j} \boxminus h(x_i, f_j)\|^2_{\Sigma_v^{-1}}, \end{aligned}$$

where $z_{i,j} \in \mathbb{R}^{d_z}$ denotes the feature measurement of feature j observed at pose $x_i \in \mathcal{X}_p$. By using Gauss-Newton linearization, we leverage constraints posed by the measurement residuals to construct an updated mean for $\tilde{x}_t$, denoted $\overline{\mu_t} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, and an updated covariance for $\tilde{x}_t$, denoted $\overline{\Sigma}_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$. As a result, our cost will be updated to $c_{MSCKF,t,4} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \to \mathbb{R}$:

$$c_{MSCKF,t,4}(\tilde{x}_t) := \|\tilde{x}_t \boxminus \overline{\mu_t}\|^2_{\overline{\Sigma}_t^{-1}},$$

which assumes the form of $c_{MSCKF,t,0}$.

The *state propagation* step propagates the full state by incorporating dynamics residuals into the running cost $c_{MSCKF,t,0}$, resulting in a new cost $c_{MSCKF,t,5} : \mathbb{R}^{2d_{\text{IMU}}+nd_x} \to \mathbb{R}$:

$$\begin{aligned}
&c_{MSCKF,t,5}(\tilde{x}_t, x_{t+1,\text{IMU}}) \\
:=&\|\tilde{x}_t \boxminus \overline{\mu_t}\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1,\text{IMU}} \boxminus g_{\text{IMU}}(x_{t,\text{IMU}})\|^2_{\Sigma_t^{-1}}.
\end{aligned}$$

In effect, $c_{MSCKF,t,5}$ appends the new IMU variable $x_{t+1,\text{IMU}} \in \mathcal{X}_{\text{IMU}}$ to the current full state $\tilde{x}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, and constrains this new full state via the dynamics residuals. One marginalization step, with $\tilde{x}_{t,K} := (x_{t+1,\text{IMU}}, x_1, \cdots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and $\tilde{x}_{t,M} := x_{t,\text{IMU}} \in \mathcal{X}_{\text{IMU}}$, is then applied to remove the previous IMU state, $x_{t,\text{IMU}}$, from the running cost. This produces a mean $\mu_{t+1} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and a covariance $\Sigma_{t+1} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$ for the new MSCKF full state, $\tilde{x}_{t+1} := \tilde{x}_{t,K} = (x_{t+1,\text{IMU}}, x_1, \cdots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$. Accordingly, the running cost maintained in the optimization framework is updated to $c_{MSCKF,t+1,0} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \to \mathbb{R}$:

$$c_{MSCKF,t+1,0}(\tilde{x}_{t+1}) := \|\tilde{x}_{t+1} \boxminus \mu_{t+1}\|^2_{\Sigma_{t+1}^{-1}},$$

which returns the running cost to the form of $c_{MSCKF,t,0}$.

The theorems below establish that the feature update, and propagation steps of the MSCKF, presented above in our optimization framework, correspond precisely to those presented in the standard MSCKF (Algorithm 10) [24].

Theorem 4.2.1. *The pose augmentation step of the standard MSCKF SLAM algorithm (Alg. 11) is equivalent to applying a Gauss-Newton step to $c_{MSCKF,t,1} : \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \to \mathbb{R}$, given by:*

$$c_{MSCKF,t,1}(\tilde{x}_t, x_{n+1}) = \|\tilde{x}_t \boxminus \mu_t\|^2_{\Sigma_t^{-1}} + \epsilon^{-1}\|x_{n+1} \boxminus \psi(\tilde{x}_t, x_{n+1}^{IMU})\|^2_2,$$

and taking $\epsilon \to 0$ in the resulting (augmented) mean μ_t and covariance Σ_t.

Proof. See Appendix (Section 5). □

Theorem 4.2.2. *The feature update step of the standard MSCKF algorithm (Alg. 12) is equivalent to applying a marginalization step to $c_{MSCKF,t,3} : \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \times \mathbb{R}^{|S_f|d_f} \to \mathbb{R}$, given by:*

$$c_{MSCKF,t,3}(\tilde{x}_t, f_{S_f}) = \|\tilde{x}_t \boxminus \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{(x_i,f_j)\in S_{z,1}\cup S_{z,2}} \|z_{i,j} \boxminus h(x_i, f_j)\|^2_{\Sigma_v^{-1}},$$

Algorithm 4: Multi-State Constrained Kalman Filter, as iterative optimization.

```
Data: Prior $\mathcal{N}(\mu_0, \Sigma_0)$ on $x_{\text{IMU},0} \in \mathcal{X}_{\text{IMU}}$, noise covariances $\Sigma_w$, $\Sigma_v$, dynamics $g_{\text{IMU}}$,
      measurement map $h$, time horizon $T$, Pose transform $\psi$ (IMU $\rightarrow$ global) ,
      $\epsilon > 0$.
Result: Estimates $\hat{x}_t$ for all desired timesteps $t \in \{1, \cdots, T\}$ .
1  $\text{cost}_t \leftarrow \|x_0 \boxminus \mu_0\|^2_{\Sigma_0}$. (Initialize objective function).
2  $S_z, S_x, S_{z,1}, S_{z,2} \leftarrow \phi$
3  $(n, p) \leftarrow (0, 0)$
4  for $t = 0, \cdots, T$ do
5      while new pose $x_{n+1} \in \mathcal{X}_p$ recorded, new IMU measurement not received do
6          $\text{cost}_t \leftarrow \text{cost}_t + \epsilon^{-1}\|x_{n+1} \boxminus \psi(\tilde{x}_t, x^{\text{IMU}}_{n+1})\|^2_2$.
7          $\mu_t, \Sigma_t, \text{cost}_t \leftarrow$ 1 Gauss-Newton $\text{cost}_t$ (Alg. 1), about $(\mu_t, \psi(\mu_t, x^{\text{IMU}}_{n+1}))$ with
            $\epsilon \rightarrow 0$.
8          $\{z_{n+1,j}\} \leftarrow$ Feature measurements at $x_{n+1}$
9          $S_z \leftarrow S_z \cup \{(x_{n+1}, f_j) | f_j \text{ observed at } n+1\}$
10         $n \leftarrow n + 1$
11         if $n \geq N_{\max} - 1$ then
12             $S_x \leftarrow \{x_i | i \bmod 3 = 2, \text{ and } 1 \leq i \leq n.\}$
13             $S_{z,1} \leftarrow \{(x_i, f_j) \in S_z \big| x_i \in S_x, \text{feature } j \text{ observed at each pose in } S_x\}$
14         end
15         $S_{z,2} \leftarrow \{(x_i, f_j) \in S_z | f_j \text{ not observed at } x_n\}$.
16         $\text{cost}_t \leftarrow \text{cost}_t + \sum_{(x_i, f_j) \in S_{z,1} \cup S_{z,2}} \|z_{i,j} \boxminus h(x_i, f_{t,j})\|_{\Sigma_v^{-1}}$
17         $\overline{\mu_t}$, $\overline{\Sigma_t}$, $\text{cost}_t \leftarrow$ 1 Gauss-Newton step on $\text{cost}_t$, about $\mu_t$ (Alg. 1)
18         $\hat{x}_t \leftarrow \overline{\mu_t} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
19         $S_z \leftarrow S_z \backslash (S_{z,1} \cup \{(x_i, f_j) | x_i \in S_x\})$
20         Reindex poses and features in ascending order.
21         $(p, n) \leftarrow (p - |S_f|, n - |S_x|)$
22     end
23     if $t < T$ then
24         $\text{cost}_t \leftarrow \text{cost}_t + \|x_{t+1,\text{IMU}} \boxminus g_{\text{IMU}}(x_{t,\text{IMU}})\|^2_{\Sigma_w^{-1}}$.
25         $\mu_{t+1}, \Sigma_{t+1}, \text{cost}_t \leftarrow$ 1 Marginalization step on $\text{cost}_t$, about $(\overline{\mu_t}, g(\mu_{t,\text{IMU}}))$ (Alg.
            2)
26     end
27 end
28 return $\hat{x}_0, \cdots \hat{x}_T \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n$
```

where $f_{S_f} \in \mathbb{R}^{|S_f|d_f}$ denotes the stacked vector of all feature positions in S_f (see Algorithm 10).

Proof. See Appendix (Section 5). □

Theorem 4.2.3. *The state propagation step of the standard MSCKF SLAM algorithm (Alg. 13) is equivalent to applying a Marginalization step once to $c_{MSCKF,t,5} : \mathbb{R}^{2d_{IMU}+nd_x} \to \mathbb{R}$, given by:*

$$c_{MSCKF,t,5}(\tilde{x}_t, x_{t+1,IMU}) = \|\tilde{x}_t \boxminus \overline{\mu_t}\|^2_{\Sigma_t^{-1}} + \|x_{t+1,IMU} \boxminus g_{IMU}(x_{t,IMU})\|^2_{\Sigma_t^{-1}}.$$

with $\tilde{x}_{t,K} := (x_{t+1,IMU}, x_1, \cdots, x_n) \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n$ and $\tilde{x}_{t,M} = x_{t,IMU} \in \mathcal{X}_{IMU}$.

Proof. See Appendix (Section 5). □

Remark 4.2.1. *In practice, the application of the Gauss-Newton algorithm for the pose augmentation step can be delayed, and instead done in conjunction with the feature update step.*

4.3 State-of-the-Art SLAM Algorithms

This paper aims to construct a generalized, optimization-based SLAM algorithm that balances the need for computational efficiency, estimation accuracy, and map precision. These tradeoffs are clearly observed in the design choices of SLAM algorithms in the existing literature.

- **Extended Kalman Filter (EKF)** [34] [33] [36] [35] –The EKF iteratively updates a full state consisting of the current pose, and position estimates of all features observed; all past poses are marginalized. This design favors computational speed over localization precision. The *iterated Extended Kalman Filter* (iEKF), a variant of EKF, takes multiple Gauss-Newton steps before each marginalization step, to tune the linearization point about which marginalization occurs. This improves mapping and localization accuracy but increases computation time.

- **Multi-State Constrained Kalman Filter** [24] [18] [19]–The MSCKF iteratively updates a full state consisting of the current IMU state and n past poses, with $n \leq N_{\max}$, a specified upper bound (features are stored separately). Here, the choice of $N_{\max}$ most directly characterizes the tradeoff between accuracy and computational speed.

- **Sliding Window Smoother, Fixed-Lag Smoother** [22] [30] [10]–The fixed-lag smoother resembles the MSCKF, but performs multiple steps of Gauss-Newton descent before the marginalization step, to tune the linearization point. This improves mapping and localization accuracies of the MSCKF at the cost of increasing computation time.

- **Open Keyframe Visual-Inertial SLAM (OKVis)** [17] [23] [25] –OKVis updates a sliding window of "keyframe" poses, that are deemed the most informative and may be arbitrarily spaced in time. Keyframe poses leaving the sliding window are marginalized, while non-keyframe poses are dropped. This design choice improves estimation accuracy by maximizing information encoded by the stored poses, without increasing computation time.

- **Graph SLAM and Bundle Adjustment** [34] [13] [14] –These algorithms solve the full SLAM problem, with no marginalization. Their state estimation can be more accurate than the above algorithms, but their computational times are often longer by one to two orders of magnitude.

4.4 A Generalized Sliding Window SLAM Algorithm

The main application of marginalization to SLAM is illustrated by the classical sliding window filter [30]. As stated earlier, the SLAM cost function grows unbounded with time as additional measurements are recorded. To enable real-time operation of optimization-based SLAM, a *sliding-window filter* is proposed, which only maintains the most recent k camera poses and landmarks visible from those poses in the cost function. This is done by marginalizing away the oldest pose as soon as the number of poses in the optimization problem exceeds k. Moreover, any landmarks that are no longer visible from any of the most recent k poses are discarded.

In this section, we use our optimization-based framework to generalize the algorithms listed in (4.3) into a unified framework. Presented in Algorithm 5 this generalized sliding-window SLAM algorithm allows the human operator to flexibly tune critical design choices, e.g., if and when poses and features are incorporated into and marginalized (or discarded) from the algorithm, and how many steps of iterative optimization are performed on the resulting running cost. These choices allow explicit fine-tuning of the tradeoff between computational efficiency, localization accuracy, and map construction precision.

We can summarize the algorithm as follows. We start off with a prior (μ_0, Σ_0) over the initial robot pose x_0. The cost function at time 0 includes only the variable $\overline{x} = (x_0)$, and is $c_0(\overline{x}) = \|x_0 - \mu_0\|^2_{\Sigma_0^{-1}}$. The best estimate $\overline{x}_0^*$ is simply the prior pose μ_0. Then at the n-th timestep, the following processing steps are performed.

1. **Cost function:** The current cost has the form

$$c_n(\overline{x}) = \|\overline{x} - \mu_n\|^2_{\Sigma_n^{-1}} + \sum_{t=n-k+1}^{n-1} \|x_{t+1} - g(x_t)\|^2_{\Sigma_v^{-1}} + \sum_{t,j \in S_n} \|z_{tj} - h(x_t, f_j)\|^2_{\Sigma_w^{-1}}$$

   where S_n is a set of index pairs such that $(t, j) \in S_n$ if and only if a measurement of landmark j from robot pose x_t is available at timestep n.

Algorithm 5: Generalized Sliding Window Filter, as an iterative optimization problem .

```
Data: Prior on $x_{\text{IMU},0} \in \mathcal{X}_p$: $\mathcal{N}(\mu_0, \Sigma_0)$, noise covariances $\Sigma_w$, $\Sigma_v$, dynamics $g_{\text{IMU}}$,
      measurement map $h$, time horizon $T$, pose transformation $\psi$ (IMU $\rightarrow$ global).
Result: Estimates $\hat{x}_t, \forall t \in \{1, \cdots, T\}$ .
1  $\text{cost}_t \leftarrow \|x_0 \boxminus \mu_0\|^2_{\Sigma_0} \in \mathbb{R}$. (Initialize objective function).
2  $x_0^\star \leftarrow \mu_0 \in \mathcal{X}_p$. (Initialize linearization point).
3  $S_z, S_x, S_{z,1}, S_{z,2} \leftarrow \phi$
4  $(n, p) \leftarrow (0, 0)$
5  for $t = 0, \cdots, T$ do
6      $x_{n+1} \in \mathcal{X}_p \leftarrow$ New pose recorded in new image
7      $\text{cost}_t \leftarrow \text{cost}_t + \epsilon^{-1}\|x_{n+1} \boxminus \psi(\tilde{x}_t, x_{n+1}^{\text{IMU}})\|_2^2 \in \mathbb{R}$.
8      $\mu_t, \Sigma_t, \text{cost}_t \leftarrow$ Gauss-Newton step(s) on $\text{cost}_t$, about $\big(\mu_t, \psi(\mu_t, x_{n+1}^{\text{IMU}})\big)$ (Alg. 1),
        with $\epsilon \rightarrow 0$
9      $(z_{t,1}, \cdots, z_{t,p+p'}) \leftarrow$ Measurements of existing features.
10     $\text{cost}_t \leftarrow \text{cost}_t + \sum_{k=1}^{p+p'} \|z_{t,k} - h(x_t, f_k)\|^2_{\Sigma_v^{-1}} \in \mathbb{R}$.
11     $\overline{\mu}_t, \overline{\Sigma}_t, \text{cost}_t \leftarrow$ Gauss-Newton step(s) on $\text{cost}_t$, about $\mu_t$, 1 iteration (Alg. 1 ).
12     $S_x \leftarrow \{x_i | x_i$ is a pose to marginalize$\}$
13     $\overline{\mu}_t, \overline{\Sigma}_t, \text{cost}_t \leftarrow$ Marginalization step on $\text{cost}_t$, about $\mu_t$ (Alg. 2).
14     $S_z \leftarrow \big\{(x_i, f_j) \big|$Feature measurements to marginalize$\big\}$
15     $S_f \leftarrow \big\{f_j \big| \exists x_i \in x_{1:n}$ s.t. $(x_i, f_j) \in S_z\big\}$
16     $\text{cost}_t \leftarrow \text{cost}_t + \sum_{(x_i, f_j) \in S_z} \|z_{i,j} \boxminus h(x_i, f_j)\|_{\Sigma_v^{-1}} \in \mathbb{R}$
17     $\overline{\mu_t}$, $\overline{\Sigma_t}$, $\text{cost}_t \leftarrow$ Gauss-Newton step(s) on $\text{cost}_t$ (Alg. 1)
18     $\hat{x}_t \leftarrow \overline{\mu_t} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \times (\mathbb{R}^{d_f})^p$.
19     Reindex poses and features, in ascending order .
20     $(p, n) \leftarrow (p - |S_f|, n - |S_x|)$
21     if $t < T$ then
22         $\text{cost}_t \leftarrow \text{cost}_t + \|x_{t+1,\text{IMU}} \boxminus g_{\text{IMU}}(x_{t,\text{IMU}})\|^2_{\Sigma_w^{-1}} \in \mathbb{R}$.
23         $\mu_{t+1}, \Sigma_{t+1}, \text{cost}_t \leftarrow$ Marginalization step on $\text{cost}_t$, about $\big(\mu_{t,\text{IMU}}, g_{\text{IMU}}(\mu_{t,\text{IMU}})\big)$
            (Alg. 2)
24     end
25 end
26 return $\hat{x}_0, \cdots \hat{x}_T \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \times (\mathbb{R}^{d_f})^p$
```

2. **State augmentation:** Add the latest pose x_{n+1} to the optimization problem using the odometry measurement made at time n.

$$c_n(\overline{x}) \leftarrow c_n(\overline{x}) + \|x_{n+1} - g(x_n)\|^2_{\Sigma_v^{-1}}$$

   If the best available estimate of the existing optimization variables is $\overline{x}^*_n = (x^*_0, \cdots, x^*_n, f^*_1, \cdots, f^*_p)$, then initialize variable x_{n+1} with the guess $x^*_{n+1} = g(x^*_n)$.

3. **Feature augmentation:** Add new landmark measurements from the current pose x_{n+1}. Let $\{j_1, \cdots, j_m\}$ be the indices of landmarks (old and new) measured from the current pose.

$$c_n(\overline{x}) \leftarrow c_n(\overline{x}) + \sum_{i=j_1,\cdots,j_m} \|z_{n+1,i} - h(x_{n+1}, f_i)\|^2_{\Sigma_w^{-1}}$$

   In addition, update $S_n \leftarrow S_n \cup \{(n+1, j_k)\}$. Additionally, initialize all newly detected landmarks with initial guesses by adding them to the vector $\overline{x}^*_n$.

4. **Cost propagation:** if $n > k$, some poses must be removed. Let S_f be the set of landmark indices that are not visible from poses $(x_{n-k+2}, \cdots, x_{n+1})$. Apply marginalization to remove the variables $\overline{x}_M = S_f \cup \{x_{n-k+1}\}$. Update $s_n \leftarrow S_n \setminus \{(t, j) : j \in S_f, t = 0, \cdots, n+1\}$.

5. **Measurement Update:** Update the best estimate of all variables currently in the optimization problem by taking Gauss-Newton steps until convergence. Reset $c_{n+1} \leftarrow c_n, \overline{x}^*_{n+1} \leftarrow \overline{x}^*_n, S_{n+1} \leftarrow S_n$, and proceed to the next timestep.

In other words, the sliding window filter marginalizes away the oldest pose whenever the number of poses in the optimization problem exceeds k, marginalizes features as soon as they are no longer visible from the last k poses, appends feature measurement cost terms from the current timestep, and takes Gauss-Newton steps until convergence to update estimates of all variables. Under the assumption that landmarks are more-or-less evenly distributed in the scene, this marginalization policy ensures that the number of variables in the optimization problem remains constant, thus ensuring constant runtime per timestep. The time to process each frame can be tuned simply by changing the window size. The smaller k is, the less time it will take to process each frame. Note that most of the runtime is spent in taking Gauss-Newton steps.

The sliding window filter is a SLAM algorithm paradigm that explicitly depends on nonlinear optimization. Sliding window filters are used as the backbone of many state of the art SLAM systems such as [17, 25, 28]. Such approaches have received renewed prominence in online optimization recently, due to the rapid increase in the accessibility of computational resources.

Chapter 5

Implementation and Experimental Results

This chapter presents the empirical performance of our optimization-based SLAM framework on the quality of pose tracking in real-world data. We designed a general SLAM back-end that modularly implements utilities to keep track of the current cost function, performs Gauss-Newton descent, and marginalizes out variables. We then analyze and implement each state-of-the-art algorithm as a particular choice of marginalization scheme in this general back-end algorithm. Next, we implemented a sliding window filter, to examine the effect of window size on state estimation accuracy, and contrast its performance to that of the our optimization-based reformulation of the MSCKF.

5.1 Simulation Settings

Dataset

All experiments are performed on the EuRoC MAV dataset [2], a popular public SLAM dataset of stereo image sequences and inertial measurement unit (IMU) measurements. The stereo images and IMU measurements arrive at rates of 20Hz and 200Hz, respectively. The sensor suite is mounted on board a micro aerial vehicle. Ground-truth poses, recorded using a Vicon moion capture system and IMU biases, are also available for each sequence. Our experiments are carried out on the Vicon Room 2 01 and 02 sequences, which contain about 2300 stereo images each and span about 2 minutes of real-time operation. The first of these sequences is easier to analyze via our SLAM algorithms, as it corresponds to simple and slow evolution of the camera, while the latter contains some jerky and quick motions that prove challenging to some algorithms.

Front-end and Back-end

Since the focus of this work is the SLAM back-end, we standardize the front-end across all experiments, altering only the back-end used to process the abstracted data produced by the front-end. We use keypoint features as environment landmarks, as is standard in visual SLAM. First, BRISK features are extracted from both images of the input stereo pair. Then, feature matching is carried out between the left and right image frames. We then carry out brute force matching using Hamming distance on the binary BRISK descriptors, and filter outliers via an epipolar constraint check, using the known relative pose between the two cameras in the stereo set-up. Only keypoints for which a stereo match was found are kept. Next, a four-way consistence check is carried out. i.e. a match between two stereo frames S_1, S_2 is accepted if and only if both observations of a given feature in S_1 are matched to the respective observations of the same feature in S_2. Finally, outlier matches are rejected by projecting the best estimate of the matched feature onto the best estimate of the current camera pose, and rejecting matches that have a high reprojection error. Any stereo matches in the current frame that were not matched with a previously seen is recorded as a newly detected landmark, and initialized using stereo triangulation from the best estimate of the current camera pose. The front-end maintains data structures allowing two-way access between features and camera poses: for each feature index, it is possible to look up all camera poses from which that feature is visible, and likewise for each camera pose it is possible to query which features are visible in that frame.

For book-keeping the cost function in the back-end, computing Jacobians, and implementing Gauss-Newton optimization, we use GTSAM in C++ [8, 9].

Dynamics and Image Measurement Models

We use an on-board IMU to collect odometry measurements, i.e., body-frame angular velocity and linear acceleration, and apply the IMU pre-integration scheme detailed in [11], as summarized below. The objective of IMU preintegration is to establish a discrete-time dynamics map $x_{t+1} = g(x_t)$ that allows us to predict the pose of the robot at time x_{t+1} given the pose at time t and the IMU measurement at time t. Here, each timestep corresponds to a new *image* measurement. Since IMU measurements arrive at a faster rate than image measurements, to compute this map, we must stack several IMU measurements into a relative state measurement, which can then be concatenated with the state at time x_t to get the predicted state at time x_{t+1}.

The robot state x_t consists of the orientation, position, and velocity of the body frame relative to the world frame, and the IMU biases, so that $x_t = (R_t, p_t, v_t, b_t)$, where $(R_t, p_t) \in SE(3)$, $v_t \in \mathbb{R}^3$ and $b_t = (b_t^g, b_t^a) \in \mathbb{R}^3 \times \mathbb{R}^3 \simeq \mathbb{R}^6$ are the IMU biases in the gyroscope and accelerometer respectively. The IMU biases are slowly varying and generally unknown, so they are included in the robot state and are also jointly estimated.

The IMU measures angular velocity and accelerations. The measurements are denoted ${}_B\tilde{a}$ and ${}_B\tilde{\omega}_{WB}$. Here, B refers to the robot's body frame and W the world frame. The prefix

B means that the quantity is expressed in the B frame, and the suffix WB denotes that the quantity represents the motion of the B frame relative to W. So the pose of the robot is $(R_{WB}, {}_Wp) \in SE(3)$. The measurements are affected by additive white noise η and the slowly varying IMU biases:

$$\begin{aligned} {}_B\tilde{\omega}_{WB}(t) &= {}_B\omega_{WB}(t) + b^g(t) + \eta^g(t) \\ {}_B\tilde{a}(t) &= R_{WB}^{\top}(t)({}_Wa(t) - {}_Wg) + b^g(t) + \eta^g(t) \end{aligned}$$

where ${}_W\omega_{WB}$ is the true angular velocity, ${}_Wa$ the true acceleration, and ${}_Wg$ the gravity acceleration vector in the world frame. Assume that between timestep t and $t+1$, we received m IMU measurements, at constant time increments Δt. [11] provides expressions for $(x_{t+1} \boxminus g(x_t))$ directly in terms of the IMU measurements. Additionally, expressions for noise propagation are also provided, which allows us to compute the covariance Σ_v over the error $(x_{t+1} \boxminus g(x_t))$ in terms of the measurement noise η, which is what we need to compute the required cost function $\|x_{t+1} \boxminus g(x_t)\|^2_{\Sigma_v^{-1}}$ and Jacobians.

Given a camera pose $x_t = (R_t, p_t)$ and a 3D feature location $f_j = (f_j^x, f_j^y, f_j^z)$, the camera measurement model $h(x_t, f_j)$ predicts the projected pixel location of the point in both stereo images. We assume the stereo camera pair is calibrated and rectified, so that both cameras have the same pinhole camera matrix K, and the epilines are horizontal. As such, the two measurements of a point in the two cameras will share the same v-coordinate in (u, v) image space. Therefore, an image measurement will be stored as a 3-vector (u_L, u_R, v), where the coordinates of the measurement in the left and right image are (u_L, v) and (u_R, v) respectively. The measurement map h predicts the image location (u_L, u_R, v) by projecting the point f_j onto both image frames (with the standard pinhole projection), using the known poses of the two cameras in the robot's body-frame. Due to noise, the actual measurement will not have the exact same v coordinate, so the image measurement is collected by averaging the two v-coordinates of the two keypoints. This measurement z_{tj} is then compared to the predicted coordinates by a Mahalanobis distance in $\mathbb{R}^3$ space to get the cost function $\|z_{tj} - h(x_t, f_j)\|^2_{\Sigma_w^{-1}}$. The covariance Σ_w is the expected noise in image space, which is a design parameter. In our experiments we choose $\Sigma_w = \sigma I_3$ where I_3 is the 3×3 identity matrix and $\sigma = 0.05$ pixels, which was found to work well in practice.

5.2 Results and Discussion

The localization results for a sliding window filter with filter sizes 5, 10, 30, and 50 on the Vicon room 2 01 dataset (easy) are shown in Figure (5.1) along with an MSCKF with window size 10. Recall that the images arrive at 20Hz, so a window size of 20 corresponds to 1 second of real-time. Simulation results indicate that most algorithms do well on this sequence. More surprising is the fact that the MSCKF, despite its lack of multiple nonlinear Gauss-Newton updates at each iteration, performs approximately as well the large sliding window filters which are much more computationally intensive. The results for the Vicon room 2 02 dataset

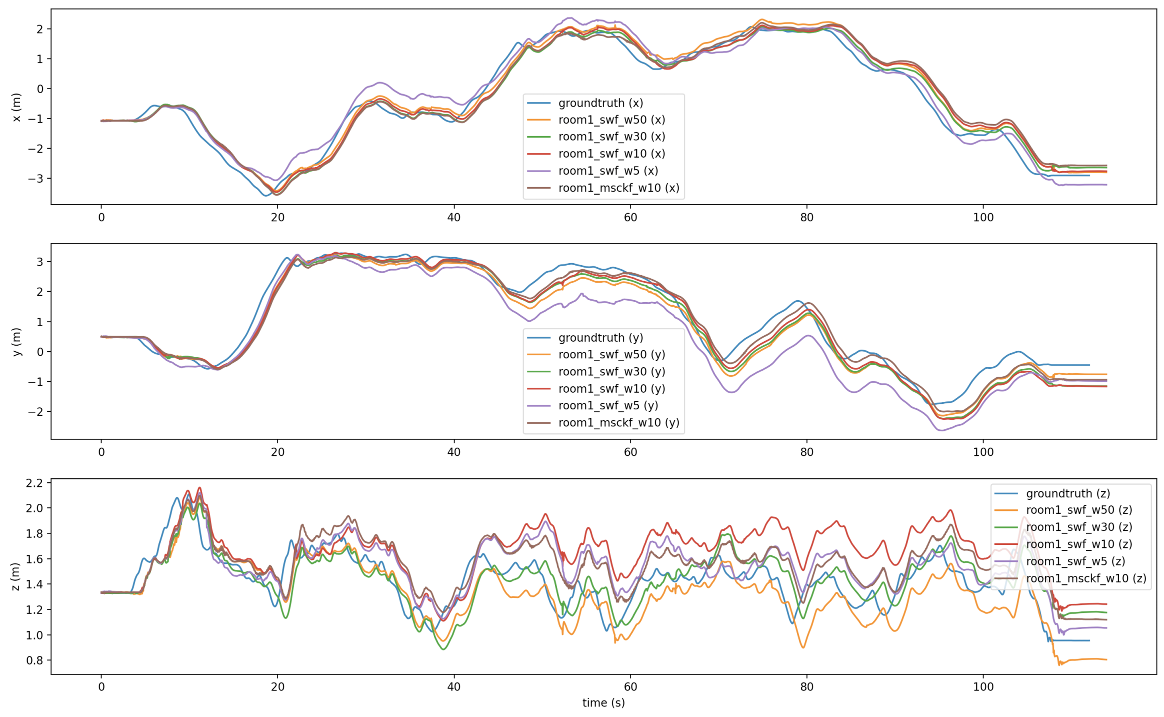

Figure 5.1: Localization results for various sliding window filters and the MSCKF on the Vicon room 2 01 (easy) dataset.

(medium) are shown in figure (5.2). Note that on this dataset, sliding window filters of size smaller than 30 fail completely, since there are parts of the sequence where the front-end loses tracking; as a result, filters are unable to recover previously collected data. Results for the sliding window filter (window size 30) and the MSCKF (window size 10) indicate that the MSCKF outperforms the larger sliding window filter on this more challenging dataset.

A few observations can be made immediately. First, on the easy dataset, there is not much improvement in the performance of the sliding window filter when the window size is increased past 10. This indicates a critical limit past which point features and poses are maximally “matured,” and can be safely marginalized away. However, on the harder dataset, a small window size can be fatal, where losing tracking for any significant chunk of the duration of the window size can prove fatal for the algorithm’s performance.

Second, despite the fact that the MSCKF does not perform any costly multiple nonlinear Gauss-Newton updates, its performance is comparable to even sliding window filters with a much larger window size. On the harder dataset, the MSCKF is able to recover from lost tracking whereas sliding window filters of comparable sizes are not. This can be attributed to a few factors. First of all, the MSCKF employs a marginalization scheme wherein poses that are evenly spaced in the optimization window are dropped. As such, at any point in time,

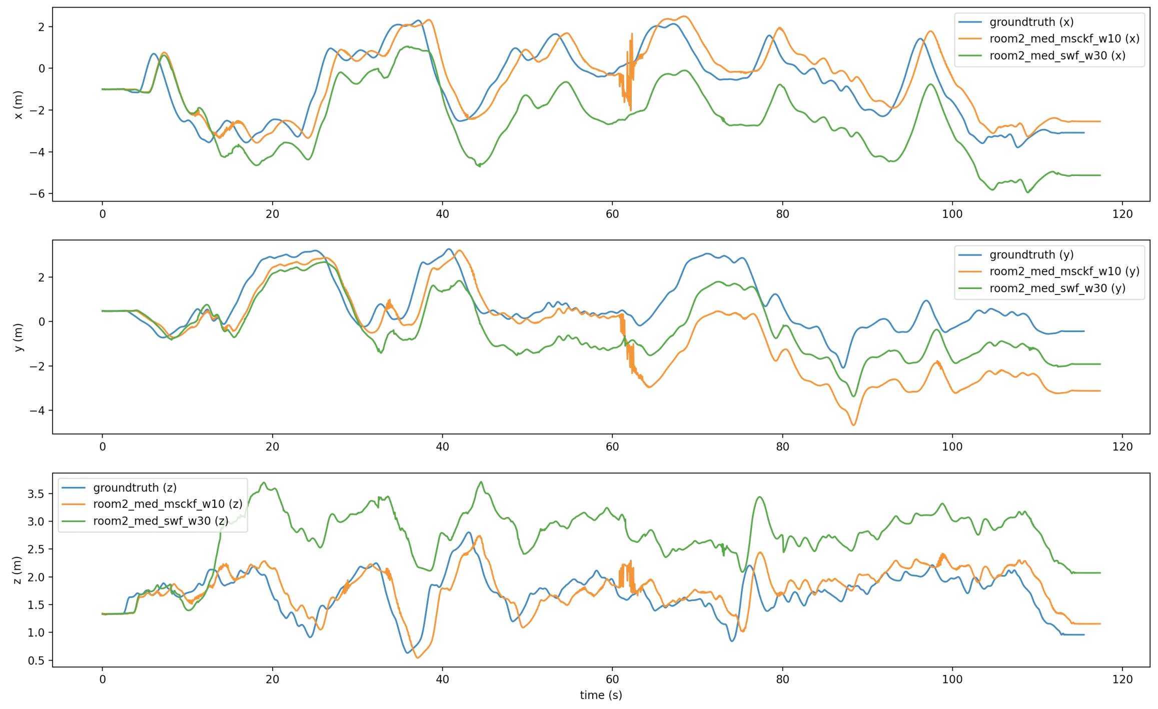

Figure 5.2: Localization results for various sliding window filters and the MSCKF on the Vicon room 2 02 (medium) dataset.

the optimization window contains poses from arbitrarily far in the past, with the density of included poses being highest near the current time. This is significant, since older poses generally represent higher baselines with respect to more recent poses, and hence features common to more recent poses and these older poses contain better localization information [24]. This also allows the MSCKF to recover from losing track, since the older poses in the optimization window can give it the localization information it needs. Further, features are only included into the optimization window once they have "matured" i.e. they have been viewed from multiple camera poses and are no longer in view. This also allows us to maximally utilize their localization information with fewer updates.

In summary, the MSCKF can be seen as a sliding window filter with a special marginalization scheme where localization "updates" are carried out by introducing and then immediately marginalizing away features. This way, we only use first-order localization information from the features to perform updates. When running the MSCKF, dropping evenly spaced poses from the optimization window and only incorporating matured features makes up for the absence of costly nonlinear updates through multiple iterations of Gauss-Newton. Moreover, by only incorporating matured features, we ensure that the feature is always initialized through multiple-view triangulation instead of just stereo triangulation; this minimizes the linearization error when the feature is marginalized away. All of this means that through the choice of a clever marginalization scheme, the MSCKF is able to compete with large sliding window filters, despite the fact that the latter employ multiple nonlinear Gauss-Newton updates for each feature before marginalization.

Chapter 6

Conclusion and Future Work

In this thesis, we presented a nonlinear optimization-based framework for analyzing and implementing SLAM algorithms. Using this framework, we demonstrated the equivalence between popular filtering based techniques, such as EKF-SLAM or MSCKF, and nonlinear optimization-based formulations, such as OK-Vis. In particular, we recast the MSCKF algorithm as an iterative optimization solver is presented, and contrasted the empirical performance of this reformulated MSCKF algorithm against explicitly optimization-based sliding window filters on a benchmark SLAM dataset in simulation. The results indicate that MSCKF with a design choice of smaller optimization windows is able to outperform a sliding window filter with large window size, despite the fact that the MSCKF does not perform multiple nonlinear feature measurement updates through Gauss-Newton steps. Further, we verified that the MSCKF recovers more efficiently from lost tracking, compared to the sliding window filter. By interpreting the MSCKF as iterative optimization with a specific marginalization scheme, the characteristics of this marginalization scheme are used to explain the observed performance characteristics. We believe that the presented approach and analysis serves as a basis for visual odometry algorithm design, specifically for filter design. Indeed, by simply tweaking marginalization policies, we can see that new performance characteristics can be extracted from the *same implementation*, as we see from the comparison between implementations of the sliding window filter and the MSCKF that are identical up to the marginalization scheme.

An immediate and natural direction for future work is to validate the generalized algorithm proposed in this work in real-life scenarios, by validating its performance on hardware experiments. Future work will also study the *dynamic SLAM* problem using the generalized optimization-based framework proposed in this thesis. In the dynamic SLAM problem, landmarks (and thus their associated features), are assumed to be mobile [15] [3]. Thus, rather than performing tracking estimation while simultaneously mapping a static environment, a dynamic SLAM algorithm must perform tracking while simultaneously updating a potentially highly dynamic environment. This problem is of growing interest in the SLAM community, due to the natural occurrence of moving features in practical multi-agent applications, such as real-life traffic scenarios, and due to the difficulty of solving SLAM in rapidly evolving surroundings [4] [12] [16]. We anticipate that the generalized SLAM algorithm introduced in this thesis will be useful in tackling the dynamic SLAM problem, due to the great flexibility offered by the user-selected design choices in its optimization-based framework.

Bibliography

[1] B. Bell. “The Iterated Kalman Smoother as a Gauss-Newton Method”. In: *SIAM J. Optim.* 4 (1994), pp. 626–636.

[2] Michael Burri et al. “The EuRoC Micro Aerial Vehicle Datasets”. In: *The International Journal of Robotics Research* (2016). DOI: `10.1177/0278364915620033`. eprint: `http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.full.pdf+html`. URL: `http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract`.

[3] C. Cadena et al. “Past, Present, and Future of Simultaneous Localization and Mapping: Toward the Robust-Perception Age”. In: *IEEE Transactions on Robotics* 32.6 (2016), pp. 1309–1332.

[4] Chih-Yuan Chiu, David Fridovich-Keil, and Claire J. Tomlin. *Encoding Defensive Driving as a Dynamic Nash Game.* 2021. arXiv: `2011.04815 [cs.RO]`.

[5] Anna Dai et al. “Fast Frontier-based Information-driven Autonomous Exploration with an MAV”. In: *2020 IEEE International Conference on Robotics and Automation (ICRA).* 2020, pp. 9570–9576. DOI: `10.1109/ICRA40945.2020.9196707`.

[6] Andrew J. Davison. *FutureMapping: The Computational Structure of Spatial AI Systems.* 2018. eprint: `1803.11288` (cs.AI).

[7] Andrew J. Davison and Joseph Ortiz. *FutureMapping 2: Gaussian Belief Propagation for Spatial AI.* 2019. eprint: `1910.14139` (cs.AI).

[8] Frank Dellaert et al. “Gtsam”. In: *URL: https://borg. cc. gatech. edu* (2012).

[9] Frank Dellaert, Michael Kaess, et al. “Factor Graphs for Robot Perception”. In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[10] Frank Dellaert and Michael Kaess. “Square Root SAM: Simultaneous Localization and Mapping via Square Root Information Smoothing”. In: *The International Journal of Robotics Research* 25.12 (2006), pp. 1181–1203. DOI: `10.1177/0278364906072768`. eprint: `https://doi.org/10.1177/0278364906072768`. URL: `https://doi.org/10.1177/0278364906072768`.

[11] Christian Forster et al. “IMU Preintegration on Manifold for Efficient Visual-Inertial Maximum-a-Posteriori Estimation”. In: Georgia Institute of Technology. 2015.

[12] David Fridovich-Keil et al. "Efficient Iterative Linear-Quadratic Approximations for Nonlinear Multi-Player General-Sum Differential Games". In: *arXiv preprint arXiv:1909.04694* (2019).

[13] G. Grisetti et al. "A Tutorial on Graph-Based SLAM". In: *IEEE Intelligent Transportation Systems Magazine* 2.4 (2010), pp. 31–43. DOI: `10.1109/MITS.2010.939925`.

[14] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. USA: Cambridge University Press, 2003. ISBN: 0521540518.

[15] Mina Henein et al. *Dynamic SLAM: The Need For Speed*. 2020. arXiv: `2002.08584 [cs.RO]`.

[16] Forrest Laine et al. "The Computation of Approximate Generalized Feedback Nash Equilibria". In: *arXiv preprint arXiv:2101.02900* (2021).

[17] Stefan Leutenegger et al. "Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization". In: *The International Journal of Robotics Research* 34 (2015), pp. 314–334.

[18] Mingyang Li and Anastasios I. Mourikis. "Improving the Accuracy of EKF-based Visual-Inertial Odometry". In: *2012 IEEE International Conference on Robotics and Automation* (2012), pp. 828–835.

[19] Mingyang Li and Anastasios I. Mourikis. "Optimization-based Estimator Design for Vision-aided Inertial Navigation: Supplemental Materials". In: *Robotics: Science and Systems* (2012).

[20] MLA Lourakis and Antonis A Argyros. "Is Levenberg-Marquardt the most efficient optimization algorithm for implementing bundle adjustment?" In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. Vol. 2. IEEE. 2005, pp. 1526–1531.

[21] Donald W Marquardt. "An algorithm for least-squares estimation of nonlinear parameters". In: *Journal of the society for Industrial and Applied Mathematics* 11.2 (1963), pp. 431–441.

[22] Peter S. Maybeck et al. *Stochastics Models, Estimation, and Control: Introduction*. 1979.

[23] Christopher Mei et al. "RSLAM: A System for Large-Scale Mapping in Constant-Time Using Stereo". In: *International Journal of Computer Vision* 94 (Sept. 2011), pp. 198–214. DOI: `10.1007/s11263-010-0361-7`.

[24] Anastasios I. Mourikis and Stergios I. Roumeliotis. "A Multi-State Constraint Kalman Filter for Vision-aided Inertial Navigation". In: *Proceedings 2007 IEEE International Conference on Robotics and Automation* (2007), pp. 3565–3572.

[25] Esha Nerurkar, Kejian Wu, and Stergios Roumeliotis. “C-KLAM: Constrained Keyframe-based Localization and Mapping”. In: *Proceedings - IEEE International Conference on Robotics and Automation* (May 2014), pp. 3638–3643. DOI: 10.1109/ICRA.2014.6907385.

[26] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

[27] Michael JD Powell. “A new algorithm for unconstrained optimization”. In: *Nonlinear programming*. Elsevier, 1970, pp. 31–65.

[28] Tong Qin, Peiliang Li, and Shaojie Shen. “VINS-Mono: A Robust and Versatile Monocular Visual-Inertial State Estimator”. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[29] Benjamin Recht and Stephen Wright. *Optimization for Modern Data Analysis*. Cambridge University Press, 2021. ISBN: 1316518981.

[30] Gabe Sibley, Larry H. Matthies, and Gaurav S. Sukhatme. “Sliding window filter with application to planetary landing.” In: *J. Field Robotics* 27.5 (2010), pp. 587–608. URL: http://dblp.uni-trier.de/db/journals/jfr/jfr27.html#SibleyMS10.

[31] Randall C Smith and Peter Cheeseman. “On the representation and estimation of spatial uncertainty”. In: *The international journal of Robotics Research* 5.4 (1986), pp. 56–68.

[32] J. Solà, Jérémie Deray, and Dinesh Atchuthan. “A Micro Lie Theory for State Estimation in Robotics”. In: *ArXiv* abs/1812.01537 (2018).

[33] Joan Solà. “Simultaneous localization and mapping with the extended Kalman filter”. In: *arXiv* (2014).

[34] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 0262201623.

[35] S. Tully et al. “Iterated Filters for Bearing-only SLAM”. In: *2008 IEEE International Conference on Robotics and Automation*. 2008, pp. 1442–1448. DOI: 10.1109/ROBOT.2008.4543405.

[36] Zhengyou Zhang. “Parameter Estimation Techniques: a Tutorial with Application to Conic Fitting”. In: *Image and Vision Computing* 15.1 (1997), pp. 59–76. ISSN: 0262-8856. DOI: https://doi.org/10.1016/S0262-8856(96)01112-2. URL: http://www.sciencedirect.com/science/article/pii/S0262885696011122.

Chapter 7

Appendix

7.1 Appendix for Chapter 2

Jacobians Under Box Operators

In subsequent proofs, we require the following results regarding the behavior of Jacobian matrices under the $\boxplus$ and $\boxminus$ operators. In particular, we will focus on the case where the box operator acts on elements of $SO(3)$, since $SO(3)$ is the non-Euclidean-space Lie group that appears most often in this paper.

Definition 7.1.1 (Jacobian on $SO(3)$). *We say that $f : SO(3) \to SO(3)$ is differentiable at $R \in SO(3)$, with Jacobian denoted by:*

$$\frac{\partial}{\partial \theta} f(R) \in \mathbb{R}^{3\times 3},$$

if the following equality holds:

$$\lim_{\delta\theta \to 0} \frac{\left\| f(R \boxplus \delta\theta) \boxminus f(R) - \frac{\partial f}{\partial \theta}\delta\theta \right\|_2}{\|\delta\theta\|_2} = 0.$$

Theorem 7.1.2. *Suppose $f : SO(3) \to SO(3)$ is differentiable. Then, for any fixed $y \in SO(3)$:*

$$\frac{\partial}{\partial \theta}\big(y \boxminus f(R)\big) = -\frac{\partial f}{\partial \theta} + O(y \boxminus f(R)).$$

Proof. Using the definition of the $\boxminus$ operator for SO(3), we have, for any $\delta\theta \in \mathbb{R}^3$:

$$\begin{aligned} y \boxminus f(R \boxplus \delta\theta) &= y \boxminus \left(f(R) \boxplus \frac{\partial f}{\partial \theta}\delta\theta \right) + o(\delta\theta) \\ &= y \boxminus \left(f(R)\,\mathrm{Exp}\left(\frac{\partial f}{\partial \theta}\delta\theta \right) \right) + o(\delta\theta) \end{aligned}$$

$$
\begin{aligned}
&= \operatorname{Log}\left(\operatorname{Exp}\left(-\frac{\partial f}{\partial \theta}\delta\theta\right) f(R)^\top y\right) + o(\delta\theta) \\
&= \operatorname{Log}\left(\operatorname{Exp}\left(-\frac{\partial f}{\partial \theta}\delta\theta\right) \operatorname{Exp}\big(y \boxminus f(R)\big)\right) + o(\delta\theta) \\
&= y \boxminus f(R) + J_\ell^{-1}\big(y \boxminus f(R)\big) \cdot \left(-\frac{\partial f}{\partial \theta}\delta\theta\right) + o(\delta\theta).
\end{aligned}
$$

Here, we have defined $J_\ell^{-1} : \mathbb{R}^3 \backslash \{0\} \to SO(3)$ by:

$$
J_\ell^{-1}(\theta) = I - \frac{1}{2}\theta^\wedge + \left(\frac{1}{\|\theta\|^2} - \frac{1 + \cos\|\theta\|}{2\|\theta\| \cdot \sin\|\theta\|}\right)(\theta^\wedge)^2,
$$

in accordance with [32]. Here, $\wedge : \mathbb{R}^3 \to \mathbb{R}^{3\times 3}$ denotes the wedge operator. Thus, we have:

$$
\frac{\partial}{\partial \theta}\big(y \boxminus f(R)\big) = -J_\ell^{-1}\big(y \boxminus f(R)\big) \cdot \frac{\partial f}{\partial \theta}.
$$

The theorem statement now follows. □

Theorem 7.1.3. *Suppose $f : SO(3) \to SO(3)$ is differentiable. Then for any fixed $y \in SO(3)$:*

$$
\frac{\partial}{\partial \theta}\big(f(R) \boxminus y\big) = \frac{\partial f}{\partial \theta} + O(f(R) \boxminus y).
$$

Proof. Using the definition of the $\boxminus$ operator for SO(3), we have, for any $\delta\theta \in \mathbb{R}^3$:

$$
\begin{aligned}
f(R \boxplus \delta\theta) \boxminus y &= \left(f(R) \boxplus \frac{\partial f}{\partial \theta}\delta\theta\right) \boxminus y + o(\delta\theta) \\
&= \left(f(R)\operatorname{Exp}\left(\frac{\partial f}{\partial \theta}\delta\theta\right)\right) \boxminus y + o(\delta\theta) \\
&= \operatorname{Log}\left(y^\top f(R) \operatorname{Exp}\left(\frac{\partial f}{\partial \theta}\delta\theta\right)\right) + o(\delta\theta) \\
&= \operatorname{Log}\left(\operatorname{Exp}\big(f(R) \boxminus y\big)\operatorname{Exp}\left(\frac{\partial f}{\partial \theta}\delta\theta\right)\right) + o(\delta\theta) \\
&= f(R) \boxminus y + J_r^{-1}\big(f(R) \boxminus y\big) \cdot \left(\frac{\partial f}{\partial \theta}\delta\theta\right) + o(\delta\theta),
\end{aligned}
$$

which implies:

$$
\frac{\partial}{\partial \theta}\big(f(R) \boxminus y\big) = J_r^{-1}\big(f(R) \boxminus y\big) \cdot \frac{\partial f}{\partial \theta}.
$$

Here, we have defined $J_r^{-1} : \mathbb{R}^3 \backslash \{0\} \to SO(3)$ by:

$$
J_r^{-1}(\theta) = I + \frac{1}{2}\theta^\wedge + \left(\frac{1}{\|\theta\|^2} - \frac{1 + \cos\|\theta\|}{2\|\theta\| \cdot \sin\|\theta\|}\right)(\theta^\wedge)^2,
$$

in accordance with [32]. The theorem statement now follows. □

In subsequent discussions, when we consider dynamics and measurement models of the form $y = f(R) \boxplus n$, where $n \in \mathbb{R}^3$ denotes small-magnitude, zero-sum noise, terms of order $O(y \boxminus f(R))$ and $O(f(R) \boxminus y)$ are often ignored.

7.2 Appendix for Chapter 3

Proof of Theorem 3.2.1 (Gauss-Newton Steps)

Here, we present the proof of Theorem 3.2.1, reproduced below.

Theorem 7.2.1. ***(Gauss-Newton Step)*** *Let $\overline{x_t}^\star \in \mathbb{R}^d$ denote a given linearization point, and suppose $J := \frac{\partial C}{\partial \overline{x_t}} \in \mathbb{R}^{d_C \times d}$ has full column rank. Then applying a Gauss-Newton step to the cost $c(\overline{x_t})$, about $\overline{x_t}^\star \in \mathbb{R}^d$ yields the new cost:*

$$c(\overline{x_t}) = \|\overline{x_t} - \mu_t\|^2_{\Sigma_t^{-1}} + o(\overline{x_t} - \overline{x_t}^\star),$$

where $\mu_t \in \mathbb{R}^d$ and $\Sigma_t \in \mathbb{R}^{d \times d}$ are given by:

$$\begin{aligned}\Sigma_t &\leftarrow (J^\top J)^{-1},\\ \mu_t &\leftarrow \overline{x_t}^\star - (J^\top J)^{-1} J^\top C(\overline{x_t}^\star).\end{aligned}$$

Proof. We have:

$$\begin{aligned}c(\overline{x_t}) &= C(\overline{x_t})^\top C(\overline{x_t})\\ &= \big[C(\overline{x_t}^\star) + J(\overline{x_t} - \overline{x_t}^\star)\big]^\top \big[C(\overline{x_t}^\star) + J(\overline{x_t} - \overline{x_t}^\star)\big] + o(\overline{x_t} - \overline{x_t}^\star)\\ &= (\overline{x_t} - \mu_t)^\top \Sigma_t^{-1} (\overline{x_t} - \mu_t) + c_0(\overline{x_t}^\star) + o(\overline{x_t} - \overline{x_t}^\star),\end{aligned}$$

where $c_0(\overline{x_t}^\star) \in \mathbb{R}$ denotes a scalar-valued function of $\overline{x_t}^\star$ that is independent of the variable $\overline{x_t}$. This concludes the proof. □

Proof of Theorem 3.3.1 (Maringalization Steps)

Theorem 7.2.2 (**Marginalization Step**)**.** *Let $\overline{x_t}^\star \in \mathbb{R}^d$ denote a given linearization point, and suppose $J := \frac{\partial C}{\partial \overline{x_t}} \in \mathbb{R}^{d_C \times d}$ has full column rank. Define $J_K := \frac{\partial C}{\partial \overline{x_{t,K}}} \in \mathbb{R}^{d_C \times d_K}$ and $J_M := \frac{\partial C}{\partial \overline{x_{t,M}}} \in \mathbb{R}^{d_C \times d_M}$. If $C_2(\overline{x_{t,M}}, \overline{x_{t,K}})$ were a linear function of $\overline{x_t} = (\overline{x_{t,M}}, \overline{x_{t,K}})$, then applying a Marginalization step to the cost $c(\overline{x_t})$, about the linearization point $\overline{x_t}^\star = (\overline{x_{t,K}^\star}, \overline{x_{t,M}^\star}) \in \mathbb{R}^d$ yields:*

$$\min_{\overline{x_{t,M}}} c_2(\overline{x_{t,K}}, \overline{x_{t,M}}) = \|\overline{x_{t,K}} - \mu_{t,K}\|^2_{\Sigma_{t,K}^{-1}},$$

where $\Sigma_{t,K} \in \mathbb{R}^{d_K \times d_K}$ and $\mu_{t,K} \in \mathbb{R}^{d_K}$ are given by:

$$\begin{aligned}\Sigma_{t,K} &:= \big(J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top\big] J_K\big)^{-1},\\ \mu_{t,K} &:= \overline{x_{t,K}^\star} - \Sigma_{t,K} J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top\big] C(\overline{x_t^\star}).\end{aligned}$$

Proof. It suffices to show that:

$$\min_{\overline{x_{t,M}}} c_2(\overline{x_{t,K}}, \overline{x_{t,M}}) = (\overline{x_{t,K}} - \mu_K)^\top \Sigma_K^{-1} (\overline{x_{t,K}} - \mu_K) + c'(\overline{x_t^\star}).$$

To do so, we first note that since $C_2(\overline{x_t})$ is linear in $\overline{x_t}$:

$$\begin{aligned} c_2(\overline{x_t}) &= \|C_2(\overline{x_t})\|_2^2 = \|C_2(\overline{x_t^\star}) + J_2 \Delta \overline{x_t}\|_2^2 \\ &= \|C_2(\overline{x_t^\star}) + J_K \Delta \overline{x_{t,K}} + J_M \Delta \overline{x_{t,M}}\|_2^2. \end{aligned}$$

By the method of least-squares, the optimal $\Delta \overline{x_{t,M}}$ is given by the normal equation:

$$\Delta \overline{x_{t,M}} = -(J_M^\top J_M)^{-1} J_M^\top \big(C_2(\overline{x_t^\star}) + J_K \Delta \overline{x_{t,K}}\big)$$

Substituting back into our expression for $c(\overline{x_t})$, we have:

$$\begin{aligned} \min_{\overline{x_{t,M}}} .c_2(\overline{x_t}) =& \|\big(I - J_M(J_M^\top J_M)^{-1} J_M^\top\big)\big(C_2(\overline{x_t^\star}) + J_K \Delta \overline{x_{t,K}}\big)\|_2^2 \\ =& \big(C_2(\overline{x_t^\star}) + J_K \Delta \overline{x_{t,K}}\big)^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top\big]\big(C_2(\overline{x_t^\star}) + J_K \Delta \overline{x_{t,K}}\big) \\ =& (\overline{x_{t,K}} - \overline{x_{t,K}^\star})^\top \underbrace{J_K^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top\big] J_K}_{:=\Sigma_K^{-1}} (\overline{x_{t,K}} - \overline{x_{t,K}^\star}) \\ &+ 2(\overline{x_{t,K}} - \overline{x_{t,K}^\star})^\top J_K^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top\big] C_2(\overline{x_t^\star}) \\ &+ C_2(\overline{x_t^\star})^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top\big] C_2(\overline{x_t^\star}) \\ =& \big(\overline{x_{t,K}} \underbrace{- \overline{x_{t,K}^\star} + \Sigma_K J_K^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top C_2(\overline{x_t^\star})}_{:=-\mu_K}\big]\big)^\top \Sigma_K^{-1} \\ &\big(\overline{x_{t,K}} \underbrace{- \overline{x_{t,K}^\star} + \Sigma_K J_K^\top \big[I - J_M(J_M^\top J_M)^{-1} J_M^\top\big] C_2(\overline{x_t^\star})}_{:=-\mu_K}\big) \\ &+ C_2(\overline{x_t^\star})\big(I - J_M(J_M^\top J_M)^{-1} J_M^\top\big) \\ &+ c'(\overline{x_t^\star}) \\ =& (\overline{x_{t,K}} - \mu_K)^\top \Sigma_K^{-1} (\overline{x_{t,K}} - \mu_K) + c'(\overline{x_t^\star}). \end{aligned}$$

with Σ_K and μ_K as defined in the theorem statement, and $c'(\overline{x_t^\star} \in \mathbb{R}$ as a term independent of $\overline{x_t}$.

□

7.3 Appendix for Chapter 4

Continuous to Discrete Time Formulation

The robotic systems analyzed by most mainstream SLAM algorithms are described using continuous-time dynamics models. The implementation of these algorithms thus involves discrete-time propagation. We sketch this process below, in a manner which is both mathematically aligned with the existing literature and convenient for illustrating our optimization algorithm, which operates using a discrete-time dynamics model.

To begin, let $\mathbb{H}$ and $\mathbb{H}_u$ denote the space of quaternions and the space of unit quaternions, respectively. Recall that the time derivative of a smooth curve on $\mathbb{H}_u$, e.g., $q \in \mathbb{H}_u$, can be expressed as an element of $\mathbb{H}$; in particular, there exists some $\omega \in \mathbb{R}^3$ such that:

$$\dot{q} = q \star \begin{bmatrix} 0 \\ \omega \end{bmatrix} \in \mathbb{H}. \tag{7.1}$$

Let $\mathcal{X}$ be the d_x-dimensional Lie group inhabited by the robot state $x \in \mathcal{X}$, given by the finite Cartesian product of Euclidean spaces and unit quaternions. In other words, $\mathcal{X}$ can be written as $\mathcal{X} := \mathcal{X}_1 \times \cdots \mathcal{X}_N,$, where $\mathcal{X}_k$ is either $\mathbb{H}_u$ or an Euclidean space, for each $k \in \{1, \cdots, N\}$. Inspired by (7.1), we define $\mathcal{X}' := \mathcal{X}_1' \times \cdots \mathcal{X}_N'$, where, for each $k \in \{1, \cdots, N\}$, we have $\mathcal{X}_k' = \mathbb{H}$ if $\mathcal{X}_k = \mathbb{H}_u$, and $\mathcal{X}_k' = \mathcal{X}_k$ otherwise.

The fundamental question underlying the time discretization process is as follows. Suppose the continuous-time dynamics model for the robot state is given by:

$$\dot{x}(s) = g_{ct}\big(x(s), w_c(s)\big), \qquad \forall\, s \in \mathbb{R}, \tag{7.2}$$

with $g_{ct} : \mathcal{X} \times \mathbb{R}^{d_x} \to \mathcal{X}'$ smooth, and with $w_c(s) \in \mathbb{R}^{d_x}$ as zero-mean white noise with autocorrelation $\mathbb{E}[w_c(s) w_c(s')] = K \cdot \delta(s - s')$ for each $s \in \mathbb{R}$, where $K \in \mathbb{R}^{d_x \times d_x}$, $K \succ 0$, and $\delta(\cdot)$ denotes the Dirac delta function. Suppose $w_c(s)$ and deviations in $x(s)$ are treated as first-order perturbations. Can we straightforwardly construct an approximate discrete-time additive noise model, of the form:

$$x_{t+1} = g(x_t) \boxplus w(t), \tag{7.3}$$

where $x_t := x(t_0) \in \mathcal{X}$, $g : \mathcal{X} \to \mathcal{X}$ smooth, and $x_{t+1} := x(t_1) \in \mathcal{X}$ and $w(t) \in \mathbb{R}^{d_x}$, whose evolution agrees with that of (7.2) up to first-order perturbations?

We answer the above question in the affirmative, by detailing below the time discretization process used throughout the remainder of this section. With a slight abuse of notation, we write the discrete-time states as $x_t := x(t_0) \in \mathcal{X}$ and $x_{t+1} := x(t_1) \in \mathcal{X}$ for some $t_0, t_1 \in \mathbb{R}$, with $t_0 < t_1$. The main idea is to first perform non-linear integration on a set of nominal dynamics, which ignores error in the initial state and the dynamics noise. We then correct for these perturbations by assuming that they evolve as a linear dynamical system.

- Nominal state dynamics:

  We define the *nominal state dynamics* as:

  $$\dot{\hat{x}}(s) = g_{ct}(\hat{x}(s), 0) \qquad \forall\, s \in \mathbb{R}.$$

  This is the set of dynamics that would be obeyed in the absence of the white noise $w_c(s)$. If the solution $\hat{x}(s)$ uniquely exists, we can construct a smooth function $\hat{g} : \mathcal{X} \to \mathcal{X}$ that maps $\hat{x}(t_0)$ to $\hat{x}(t_1)$, i.e.:

  $$\hat{x}(t_1) = \hat{g}\big(\hat{x}(t_0)\big), \tag{7.4}$$

which describes the discrete-time propagation of the nominal state $x(t)$.

- Error state dynamics:

  To characterize the drift of the true state $x(s)$ away from the nominal state $x(s)$, we define the error state $\delta x(s) := x(s) \boxminus \delta\hat{x}(s) \in \mathbb{R}^{d_x}$, and characterize the corresponding *error-state dynamics* as:

$$\begin{aligned}\dot{\delta x}(s) &= g_{ct}(x(s), w_c(s)) \boxminus g_{ct}(\delta x(s), 0) \\ &= \frac{\partial f}{\partial x}(\hat{x}(s), 0) \cdot \delta x(s) + \frac{\partial f}{\partial w_c}(\hat{x}(s), 0) \cdot w_c(s) + o\big(\delta x(s) \cdot w_c(s)\big),\end{aligned}$$

  By approximating the dynamics of the error state $\delta x(s) \in \mathbb{R}^{d_x}$ as a linear, time-varying system, we can write, for each $s \in (t_0, t_1)$:

$$\delta x(s) = \Phi(s, t_0)\delta x(t_0) + \int_{t_0}^{s} \Phi(s, \tau) \cdot \frac{\partial f}{\partial \omega_c}(\hat{x}(\tau), 0) \cdot w_c(\tau)\, d\tau, \tag{7.5}$$

  where $\Phi(s, t_0) \in \mathbb{R}^{d_x \times d_x}$ is the state transition matrix satisfying:

$$\frac{d}{ds}\Phi(s, t_0) = \frac{\partial f}{\partial x}(\hat{x}(s), 0) \cdot \Phi(s, t_0).$$

- True State dynamics—Discrete-time propagation of mean and covariance:

  From the discrete-time propagation of the nominal and error dynamics, i.e., (7.4) and (7.5), we have:

$$\begin{aligned}x(t_1) &= \hat{x}(t_1) \boxplus \delta x(t_1) \\ &= \hat{g}(\hat{x}(t_0)) \boxplus \left(\Phi(t_1, t_0)\delta x(t_0) + \int_{t_0}^{t_1} \Phi(t_1, \tau) \cdot \frac{\partial f}{\partial \omega_c}(\hat{x}(\tau), 0) \cdot w_c(\tau)\, d\tau \right) \\ &= \Big(\hat{g}(\hat{x}(t_0)) \boxplus \Phi(t_1, t_0)\delta x(t_0)\Big) \boxplus \left(\int_{t_0}^{t_1} \Phi(t_1, \tau) \cdot \frac{\partial f}{\partial \omega_c}(\hat{x}(\tau), 0) \cdot w_c(\tau)\, d\tau \right) \\ &\qquad + o\big(\delta x(t_0), w_c(\cdot)\big),\end{aligned}$$

  where we have used the fact that the $\boxplus$ operator, which operates through:

$$\text{Exp}(\delta\theta + \delta\phi) = \text{Exp}(\delta\theta) \cdot \text{Exp}(\delta\phi) + o(\delta\theta \cdot \delta\phi)$$

  for any $x \in \mathbb{H}$ and $\delta\theta, \delta\phi \in \mathbb{R}^3$.

  Finally, by identifying:

$$\begin{aligned}x_{t+1} &\leftarrow x(t_1) \in \mathcal{X}, \\ g(x_t) &\leftarrow \hat{x}(t_1) \boxplus \Phi(t_1, t_0)\delta x(t_0) \in \mathcal{X},\end{aligned}$$

$$w_t \leftarrow \int_{t_0}^{t_1} \Phi(t_1, \tau) \cdot \frac{\partial f}{\partial \omega_c}(\hat{x}(\tau), 0) \cdot w_c(\tau)\, d\tau \in \mathbb{R}^{d_x},$$

we obtain our discrete-time dynamics model:

$$x_{t+1} = g(x_t) \boxplus w_t.$$

Notice that $w_t \in \mathbb{R}^{d_x \times d_x}$ is a zero-mean random variable, with covariance $\Sigma_w \in \mathbb{R}^{d_x \times d_x}$ given by:

$$\begin{aligned}\Sigma_w := \mathbb{E}[w_t w_t^\top] = \mathbb{E}\Bigg[\int_{t_0}^{t_1} \int_{t_0}^{t_1} \Phi(t_1, \tau) \frac{\partial f}{\partial \omega_c}(\hat{x}(\tau), 0) \cdot w_c(\tau) w_c(\tau')^\top \\ \cdot \left(\frac{\partial f}{\partial \omega_c}(\hat{x}(\tau'), 0) \right)^\top \Phi(t_1, \tau')^\top \, d\tau d\tau' \Bigg] \\ = \int_{t_0}^{t_1} \Phi(t_1, \tau) \frac{\partial f}{\partial \omega_c}(\hat{x}(\tau), 0) \cdot K \cdot \left(\frac{\partial f}{\partial \omega_c}(\hat{x}(\tau), 0) \right)^\top \Phi(t_1, \tau)^\top \, d\tau\end{aligned}$$

where we have used the fact that $\mathbb{E}[w_c(\tau) w_c(\tau')^\top] = K \cdot \delta(\tau - \tau')$, with $\delta(\cdot)$ denoting the Dirac delta function.

EKF, Setup

The Extended Kalman Filter (EKF), whose standard formulation is presented in Algorithm 6, is an iterative algorithm for updating estimates of the current pose x_t (i.e. $n = 1$) and positions of all observed features at the current time, $f_t := (f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{pd_f}$. This corresponds to the sliding window filter in our formulation, with $n = 1$ and $q = p$. Below, as an application of our optimization-based SLAM framework, we present the dynamics and measurement maps of the EKF algorithm in $\mathbb{R}^2$, as well as the associated cost functions. Dimension-wise, in its standard formulation, the 2D EKF is an instantiation of Algorithm 6 with $d_x = 3$, $d_f = 2$, and $d_z = 2$. To unify our notation, we will suppose that d_x, d_f, d_z assume these values throughout the rest of this section.

Let $x_t := (x_t^1, x_t^2, \theta_t) \in \mathbb{R}^{d_x}$ denote the *robot pose*, comprising its position and angle in $\mathbb{R}^{d_f}$, let $f_{t,k} := (f_{t,k}^1, f_{t,k}^2) \in \mathbb{R}^{d_f}$ denote the position of each *feature* $f_k \in \{f_1, \cdots, f_p\}$ visible at time t, and let $z_{t,k} := (z_{t,k}^1, z_{t,k}^2) \in \mathbb{R}^{d_z}$ denote the measurement of feature f_k at time t. The dynamics map $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, with $\dot{x}_t = g(x_t)$ is obtained by performing numerical integration on the continuous-time dynamics:

$$\begin{aligned}\dot{x}_t^1 &= v \cos\theta + w_t^1, \\ \dot{x}_t^2 &= v \sin\theta + w_t^2, \\ \dot{\theta}_t &= \omega + w_t^3,\end{aligned}$$

where $w_t := (w_t^1, w_t^2, w_t^3) \in \mathbb{R}^{d_x}$ denotes additive zero-mean Gaussian noise on the (x, y, θ) coordinates of the state variable, respectively, with joint covariance $w_t \sim \mathcal{N}(0, \Sigma_w)$ for some covariance matrix $\Sigma_w \in \mathbb{R}^{d_x \times d_x}$, $\Sigma_w \succ 0$. For more details regarding the numerical integration process, see Section 7.3 in the Appendix.

The measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$ is given by:

$$z_{t,k}^1 = f_{t,k}^1 - x_t^1 + v_t^1,$$
$$z_{t,k}^2 = f_{t,k}^2 - x_t^2 + v_t^2,$$

where $v_t := (v_t^1, v_t^2) \in \mathbb{R}^{d_z}$ denotes additive zero-mean Gaussian noise on the measurements $z_{t,j}^1$, $z_{t,j}^2 \in \mathbb{R}$, respectively, with joint covariance $v_t \sim \mathcal{N}(0, \Sigma_v)$ for some covariance matrix $\Sigma_v \in \mathbb{R}^{d_z \times d_z}$, $\Sigma_v \succ 0$. The measurement vector $z_t \in \mathbb{R}^{pd_f}$ is then given by concatenating each of the q residual measurements obtained at time t, i.e. $z_t := (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{pd_z}$.

Algorithm 6: Extended Kalman Filter SLAM, Standard Formulation.

```
Data: Prior distribution on $x_0 \in \mathbb{R}^{d_x}$: $\mathcal{N}(\mu_0, \Sigma_0)$, dynamics and measurement noise
      covariances $\Sigma_w \in \mathbb{R}^{d_x \times d_x}, \Sigma_v \in \mathbb{R}^{d_z \times d_z}$, (discrete-time) dynamics map
      $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$, measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{pd_f} \to \mathbb{R}^{d_z}$, time horizon $T \in \mathbb{N}$.
Result: Estimates $\hat{x}_t$ for all desired timesteps $t \leq T$.
1  for $t = 0, \cdots, T$ do
2      if detect new feature measurements $z_{t,p+1:p+p'} := (z_{t,p+1}, \cdots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}$
        then
3          $\mu_t, \Sigma_t, p \leftarrow$ Alg. 7, EKF feature augmentation $\big(\mu_t, \Sigma_t, p, z_{t,p+1:p+p'}, h(\cdot)\big)$
4      end
5      $z_{t,1:p} := (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{pd_z} \leftarrow$ New measurements of existing features.
6      $\overline{\mu_t}, \overline{\Sigma_t} \leftarrow$ Alg. 8, EKF feature update $\big(\overline{\mu_t}, \overline{\Sigma_t}, z_{t,1:p}, h(\cdot)\big)$.
7      if $t < T$ then
8          $\mu_{t+1}, \Sigma_{t+1} \leftarrow$ Alg. 9, EKF state propagation $\big(\mu_t, \Sigma_t, g(\cdot)\big)$
9      end
10 end
11 return $\hat{x}_0, \cdots \hat{x}_T \in \mathbb{R}^{d_x}$.
```

Proofs from Section 4.1

Theorem 7.3.1. *The feature augmentation step of the standard EKF SLAM algorithm (Alg. 7) is equivalent to applying a Gauss-Newton step to $c_{EKF,t,1} : \mathbb{R}^{d_x + pd_f} \to \mathbb{R}$, given by:*

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1}, \cdots, f_{t,p+p'}) = \|\tilde{x}_t - \mu_t\|_{\Sigma_t^{-1}}^2 + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_{t,k})\|_{\Sigma_v^{-1}}^2.$$

Algorithm 7: Extended Kalman Filter, Feature Augmentation Sub-block.

```
Data: Current EKF state $\tilde{x}_t \in \mathbb{R}^{d_x+pd_f}$, with mean $\mu_t \in \mathbb{R}^{d_x+pd_f}$ and covariance
      $\Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$, current number of features $p$, observations of new
      features at current pose $z_{t,p+1:p+p'} := (z_{t,p+1}, \cdots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}$,
      measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$, inverse measurement map
      $\ell : \mathbb{R}^{d_x} \times \mathbb{R}^{d_z} \to \mathbb{R}^{d_f}$.
Result: Updated number of features $p$, updated EKF state mean $\mu_t \in \mathbb{R}^{d_x+pd_f}$,
        covariance $\Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$ (with $p$ already updated)
1  $(\mu_{t,x}, \mu_{t,f,1:p}) \leftarrow \mu_t \in \mathbb{R}^{d_x+pd_f}$, with $\mu_{t,x} \in \mathbb{R}^{d_x}$, $\mu_{t,f,1:p} \in \mathbb{R}^{pd_f}$.
2  $\ell : \mathbb{R}^{d_x} \times \mathbb{R}^{d_z} \to \mathbb{R}^{d_f} \leftarrow$ Inverse measurement map, satisfying $z_{t,k} = h\big(x_t, \ell(x_t, z_{t,k})\big)$
    for each $x_t \in \mathbb{R}^{d_x}$, $z_{t,k} \in \mathbb{R}^{d_z}$, $\forall k = p+1, \cdots, p+p'$.
3  $\tilde{\ell}(\mu_{t,x}, z_{t,p+1}, \cdots, z_{t,p+p'}) \leftarrow \big(\ell(\mu_{t,x}, z_{t,p+1}), \cdots, \ell(\mu_{t,x}, z_{t,p+p'})\big) \in \mathbb{R}^{p'd_f \times (d_x+p'd_z)}$
4  $\mu_t \leftarrow \big(\mu_t, \tilde{\ell}(\mu_{t,x}, z_{t,p+1}, \cdots, z_{t,p+p'})\big) \in \mathbb{R}^{d_x+(p+p')d_f}$
5  $\begin{bmatrix} \Sigma_{t,xx} & \Sigma_{t,xf} \\ \Sigma_{t,fx} & \Sigma_{t,ff} \end{bmatrix} \leftarrow \Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$, with $\Sigma_{t,xx} \in \mathbb{R}^{d_x \times d_x}$,
    $\Sigma_{t,xf} = \Sigma_{t,fx}^\top \in \mathbb{R}^{d_x \times pd_f}$, $\Sigma_{t,ff} \in \mathbb{R}^{pd_f \times pd_f}$.
6  $L_x \leftarrow \frac{\partial \tilde{\ell}}{\partial x}\big|_{(\mu_t, z_t')} \in \mathbb{R}^{p'd_f \times d_x}$
7  $L_z \leftarrow \frac{\partial \tilde{\ell}}{\partial z}\big|_{(\mu_t, z_t')} \in \mathbb{R}^{p'd_f \times p'd_z}$
8  $\tilde{\Sigma}_v \leftarrow \text{diag}\{\Sigma_v, \cdots, \Sigma_v\} \in \mathbb{R}^{p'd_z \times p'd_z}$
9  $\Sigma_t \leftarrow \begin{bmatrix} \Sigma_{t,xx} & \Sigma_{t,xf} & \Sigma_{t,xx}L_x^\top \\ \Sigma_{t,fx} & \Sigma_{t,ff} & \Sigma_{t,fx}L_x^\top \\ L_x\Sigma_{t,xx} & L_x\Sigma_{t,xf} & L_x\Sigma_{t,xx}L_x^\top + L_z\tilde{\Sigma}_v L_z^\top \end{bmatrix} \in \mathbb{R}^{(d_x+(p+p')d_f)\times(d_x+(p+p')d_f)}$
10 $p \leftarrow p + p'$
11 return $\mu_t \in \mathbb{R}^{d_x+pd_f}, \Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}, p \geq 0$
```

Proof. To simplify the analysis below, we assume all degrees of freedom of new features are observed. More specifically, we assume the existence of an *inverse observation map* $\ell : \mathbb{R}^{d_x} \times \mathbb{R}^{d_z} \to \mathbb{R}^{d_f}$, satisfying $h(x_t, \ell(x_t, z_t)) = z_t$ for each $x_t \in \mathbb{R}^{d_x}, z_t \in \mathbb{R}^{d_z}$, which directly generates position estimates of new features from their feature measurements and the current pose, by effectively "inverting" the measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$ [33]. When full observations are unattainable, the missing degrees of freedom are introduced as a prior to the system [33]; in this case, similar results follow.

First, to simplify notation, define:

$$\begin{aligned} z_{t,p+1:p+p'} &= (z_{t,p+1}, \cdots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}, \\ f_{t,p+1:p+p'} &= (f_{t,p+1}, \cdots, f_{t,p+p'}) \in \mathbb{R}^{p'd_f}, \\ \tilde{h}(x_t, f_{t,p+1:p+p'}) &:= \big(h(x_t, f_{t,p+1}), \cdots, h(x_t, f_{t,p+p'})\big) \in \mathbb{R}^{p'd_z}, \\ \tilde{\Sigma}_v &= \text{diag}\{\Sigma_v, \cdots, \Sigma_v\} \in \mathbb{R}^{p'd_z \times p'd_z}. \end{aligned}$$

Algorithm 8: Extended Kalman Filter, Feature Update Sub-block.

```
Data: Current EKF state $\tilde{x}_t \in \mathbb{R}^{d_x+pd_f}$, with mean $\mu_t \in \mathbb{R}^{d_x+pd_f}$ and covariance
      $\Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$, new measurements of existing features
      $z_{t,1:p} := (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{pd_z}$, measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$
Result: Updated EKF state mean $\mu_t \in \mathbb{R}^{d_x+pd_f}$ and covariance
        $\Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$
1 $f_{t,1:p} \leftarrow (f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{pd_f}$.
2 $\tilde{h}(x_t, f_{t,1:p}) \leftarrow \big(h(x_t, f_{t,1}), \cdots, h(x_t, f_{t,p})\big) \in \mathbb{R}^{pd_z}$
3 $H_t \leftarrow \frac{\partial \tilde{h}}{\partial(x_t, f_{t,1:p})}\Big|_{\mu_t}$ Jacobian of $\tilde{h} : \mathbb{R}^{d_x} \times \mathbb{R}^{pd_f} \to \mathbb{R}^{pd_z}$ evaluated at $\mu_t \in \mathbb{R}^{d_x+pd_f}$.
4 $\tilde{\Sigma}_v \leftarrow \text{diag}\{\Sigma_v, \cdots, \Sigma_v\} \in \mathbb{R}^{pd_z\times pd_z}$.
5 $\overline{\mu_t} \leftarrow \mu_t + \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1}\big(z_{t,1:p} - \tilde{h}(\mu_t, f_{t,1:p})\big) \in \mathbb{R}^{d_x+pd_f}$.
6 $\overline{\Sigma}_t \leftarrow \Sigma_t - \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1} H_t \Sigma_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$.
7 return $\overline{\mu}_t \in \mathbb{R}^{d_x+pd_f}, \overline{\Sigma}_t \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$.
```

Algorithm 9: Extended Kalman Filter, State Propagation Sub-block.

```
Data: Current EKF state $\tilde{x}_t \in \mathbb{R}^{d_x+pd_f}$, with mean $\overline{\mu_t} \in \mathbb{R}^{d_x+pd_f}$ and covariance
      $\overline{\Sigma_t} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$, (discrete-time) dynamics map $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$
Result: Propagated EKF state mean $\mu_{t+1} \in \mathbb{R}^{d_x+pd_f}$ and covariance
        $\Sigma_{t+1} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$
1 $(\overline{\mu}_{t,x}, \overline{\mu}_{t,f,1:p}) \leftarrow \overline{\mu_t}$, with $\overline{\mu}_{t,x} \in \mathbb{R}^{d_x}$, $\overline{\mu}_{t,f,1:p} \in \mathbb{R}^{pd_f}$.
2 $\begin{bmatrix} \overline{\Sigma}_{t,xx} & \overline{\Sigma}_{t,xf} \\ \overline{\Sigma}_{t,fx} & \overline{\Sigma}_{t,ff} \end{bmatrix} \leftarrow \overline{\Sigma}_t \in \mathbb{R}^{d_x\times d_x}$, with $\overline{\Sigma}_{t,xx} \in \mathbb{R}^{d_x\times d_x}, \overline{\Sigma}_{t,xf} = \overline{\Sigma}_{t,fx}^\top \in \mathbb{R}^{d_x\times pd_f}$,
  $\overline{\Sigma}_{t,ff} \in \mathbb{R}^{pd_f\times pd_f}$.
3 $G_t \leftarrow \frac{\partial g}{\partial x}\Big|_{\overline{\mu}_{t,x}}$.
4 $\mu_{t+1} \leftarrow \big(g(\overline{\mu_t}), \overline{\mu}_{t,f,1:p}\big) \in \mathbb{R}^{d_x+pd_f}$.
5 $\Sigma_{t+1} \leftarrow \begin{bmatrix} G_t\overline{\Sigma}_{t,xx}G_t^\top + \Sigma_w & G_t\overline{\Sigma}_{t,xf} \\ \overline{\Sigma}_{t,fx}G_t^\top & \overline{\Sigma}_{t,ff} \end{bmatrix} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$.
6 return $\mu_{t+1} \in \mathbb{R}^{d_x+pd_f}, \Sigma_{t+1} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$.
```

We can now rewrite the cost $c_{EKF,t,1}$ as:

$$c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1:p+p'}) = \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \|z_{t,p+1:p+p'} - \tilde{h}(x_t, f_{t,p+1:p+p'})\|^2_{\tilde{\Sigma}_v^{-1}}.$$

To apply a Gauss-Newton step, our first task is to find a vector $C_1(\tilde{x}_t, f_{t,p+1:p+p'})$ of an appropriate dimension such that $c_{EKF,t,1}(\tilde{x}_t, f_{t,p+1:p+p'}) = C_1(\tilde{x}_t, f_{t,p+1:p+p'})^\top C_1(\tilde{x}_t, f_{t,p+1:p+p'})$. A natural choice is furnished by $C_1(\tilde{x}_t, f_{t,p+1:p+p'}) \in \mathbb{R}^{d_x+pd_f+p'd_z}$, as defined below:

$$C_1(\tilde{x}_t, f_{t,p+1:p+p'}) := \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t - \mu_t) \\ \Sigma_v^{-1/2}\big(z_{t,p+1:p+p'} - \tilde{h}(x_t, f_{t,p+1:p+p'})\big) \end{bmatrix}.$$

Thus, our parameters for the Gauss-Newton algorithm submodule are:

$$\begin{aligned} \tilde{x}_t^\star &:= (x_t^\star, f_{t,1:p}^\star, f_{t,p+1:p+p'}^\star) \\ &= \big(\overline{\mu_t}, \ell(x_t^\star, z_{t,p+1}), \cdots, \ell(x_t^\star, z_{t,p+p'})\big) \in \mathbb{R}^{d_x+(p+p')d_f}, \\ &\qquad \text{where } x_t^\star \in \mathbb{R}^{d_x}, f_{t,1:p}^\star \in \mathbb{R}^{pd_f}, f_{t,p+1:p+p'}^\star \in \mathbb{R}^{p'd_f}, \\ C_1(\tilde{x}_t^\star) &= \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t^\star - \mu_t) \\ \tilde{\Sigma}_v^{\ -1/2}\big(z_{t,p+1:p+p'} - \tilde{h}(x_t^\star, f_{t,p+1:p+p'}^\star)\big) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{d_x+pd_f+p'd_z}, \\ J &= \begin{bmatrix} \Sigma_t^{-1/2} & O \\ -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,x}\begin{bmatrix} I & O \end{bmatrix} & -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f} \end{bmatrix} \in \mathbb{R}^{(d_x+pd_f+p'd_z)\times(d_x+(p+p')d_f)}, \end{aligned}$$

where $\tilde{H}_t := \begin{bmatrix} \tilde{H}_{t,x} & \tilde{H}_{t,f} \end{bmatrix} \in \mathbb{R}^{p'd_z\times(d_x+p'd_f)}$ is defined as the Jacobian of $\tilde{h} : \mathbb{R}^{d_x}\times\mathbb{R}^{p'd_f} \to \mathbb{R}^{p'd_z}$ at $(x_t^\star, f_{t,p+1:p+p'}^\star) \in \mathbb{R}^{d_x+p'd_f}$, with $\tilde{H}_{t,x} \in \mathbb{R}^{p'd_z\times d_x}$ and $\tilde{H}_{t,f} \in \mathbb{R}^{p'd_z\times pd_f}$. By Algorithm 1, the Gauss-Newton update is thus given by:

$$\begin{aligned} \Sigma_t &\leftarrow (J^\top J)^{-1} \\ &= \left(\begin{bmatrix} \Sigma_t^{-1/2} & -\begin{bmatrix} I \\ O \end{bmatrix}\tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1/2} \\ O & -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f} \end{bmatrix} \begin{bmatrix} \Sigma_t^{-1/2} & O \\ -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,x}\begin{bmatrix} I & O \end{bmatrix} & -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f} \end{bmatrix} \right)^{-1} \\ &= \begin{bmatrix} \Sigma_t^{-1} + \begin{bmatrix} I \\ O \end{bmatrix}\tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,x}\begin{bmatrix} I & O \end{bmatrix} & \begin{bmatrix} I \\ O \end{bmatrix}\tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f} \\ \tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,x}\begin{bmatrix} I & O \end{bmatrix} & \tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,f} \end{bmatrix}^{-1} \\ &= \begin{bmatrix} \Omega_{t,xx} + \tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,x} & \Omega_{t,xf} & \tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,f} \\ \Omega_{t,fx} & \Omega_{t,ff} & O \\ \tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,x} & O & \tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{tf} \end{bmatrix}^{-1}, \qquad (7.6) \\ \overline{\mu_t} &\leftarrow \tilde{x}_t^\star - (J^\top J)^{-1}J^\top C_1(\tilde{x}_t^\star) \\ &= \big(\overline{\mu_t}, \ell(x_t^\star, z_{t,p+1}), \cdots, \ell(x_t^\star, z_{t,p+p'})\big). \end{aligned}$$

Here, we have defined $\Omega_{t,xx} \in \mathbb{R}^{d_x\times d_x}, \Omega_{t,xf} = \Omega_{t,fx}^\top \in \mathbb{R}^{d_x\times pd_f}$ and $\Omega_{t,ff} \in \mathbb{R}^{pd_f\times pd_f}$ by:

$$\begin{bmatrix} \Omega_{t,xx} & \Omega_{t,xf} \\ \Omega_{t,fx} & \Omega_{t,ff} \end{bmatrix} := \begin{bmatrix} \Sigma_{t,xx} & \Sigma_{t,xf} \\ \Sigma_{t,fx} & \Sigma_{t,ff} \end{bmatrix}^{-1} \qquad (7.7)$$

To conclude the proof, we must show that (7.6) is identical to the update equations for covariance matrix in the standard formulation of the Extended Kalman Filter algorithm, i.e., we must show that:

$$\begin{bmatrix} \Sigma_{t,xx} & \Sigma_{t,xf} & \Sigma_{t,xx}L_x^\top \\ \Sigma_{t,fx} & \Sigma_{t,ff} & \Sigma_{t,fx}L_x^\top \\ L_x\Sigma_{t,xx} & L_x\Sigma_{t,xf} & L_x\Sigma_{t,xx}L_x^\top + L_z\Sigma_v L_z^\top \end{bmatrix} \cdot \begin{bmatrix} \Omega_{t,xx} + \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,x} & \Omega_{t,xf} & \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f} \\ \Omega_{t,fx} & \Omega_{t,ff} & O \\ \tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,x} & O & \tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{tf} \end{bmatrix}$$

equals the $(d_x + (p+p')d_f) \times (d_x + (p+p')d_f)$ identity matrix. This follows by applying (7.7), as well as the matrix equalities resulting from taking the derivative of the equation $z_t := h\big(x_t, \ell(x_t, z_t)\big)$ with respect to $x_t \in \mathbb{R}^{d_x}$ and $z_t \in \mathbb{R}^{d_z}$, respectively:

$$\begin{aligned} I &= \tilde{H}_{t,f}L_z, \\ O &= \tilde{H}_{t,x} + H_{t,f}L_x. \end{aligned}$$

□

Theorem 7.3.2. *The feature update step of the standard EKF SLAM algorithm (Alg. 8) is equivalent to applying a Gauss-Newton step on* $c_{EKF,t,1} : \mathbb{R}^{d_x+pd_f} \to \mathbb{R}$*, given by:*

$$c_{EKF,t,3}(\tilde{x}_t) := \|\tilde{x}_t - \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{k=1}^{p} \|z_{t,k} - h(x_t, f_{t,k})\|^2_{\Sigma_v^{-1}}.$$

Proof. First, to simplify notation, define:

$$\begin{aligned} z_{t,1:p} &:= (z_{t,1}, \cdots, z_{t,p}) \in \mathbb{R}^{pd_z}, \\ f_{t,1:p} &:= (f_{t,1}, \cdots, f_{t,p}) \in \mathbb{R}^{pd_f}, \\ \tilde{h}(x_t, f_{t,1:p}) &:= \big(h(x_t, f_{t,1}), \cdots, h(x_t, f_{t,p})\big) \in \mathbb{R}^{pd_z}, \\ \tilde{\Sigma}_v &:= \text{diag}\{\Sigma_v, \cdots, \Sigma_v\} \in \mathbb{R}^{pd_z \times pd_z}. \end{aligned}$$

We can then rewrite the cost as:

$$c_{EKF,t,1}(\tilde{x}_t) = \|\tilde{x}_t^\star - \mu_t\|^2_{\Sigma_t^{-1}} + \|z_{t,1:p} - \tilde{h}(\tilde{x}_t^\star)\|^2_{\tilde{\Sigma}_v^{-1}}.$$

To apply a Gauss-Newton step, our first task is to find a vector $C_2(\tilde{x}_t)$ of an appropriate dimension such that $c_{EKF,t,1}(\tilde{x}_t) = C_2(\tilde{x}_t)^\top C_2(\tilde{x}_t)$. A natural choice is furnished by $C_2(\tilde{x}_t) \in \mathbb{R}^{d_x+pd_f+pd_z}$, as defined below:

$$C_2(\tilde{x}_t) := \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t - \mu_t) \\ \tilde{\Sigma}_v^{-1/2}(z_{t,1:p} - \tilde{h}(\tilde{x}_t)) \end{bmatrix}.$$

Thus, our parameters for the Gauss-Newton algorithm submodule are:

$$\tilde{x}_t^\star = \mu_t \in \mathbb{R}^{d_x+pd_f},$$

$$\begin{aligned} C_2(\tilde{x}_t^\star) &= \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t^\star - \mu_t) \\ \tilde{\Sigma}_v^{-1/2}(z_{t,1:p} - \tilde{h}(\tilde{x}_t^\star)) \end{bmatrix} = \begin{bmatrix} 0 \\ \tilde{\Sigma}_v^{-1/2}(z_{t,1:p} - \tilde{h}(\mu_t)) \end{bmatrix} \in \mathbb{R}^{d_x+pd_f+pd_z}, \\ J &= \begin{bmatrix} \Sigma_t^{-1/2} \\ -\tilde{\Sigma}_v^{-1/2} H_t \end{bmatrix} \in \mathbb{R}^{(d_x+pd_f+pd_z)\times(d_x+pd_f)}, \end{aligned}$$

where $\tilde{H}_t \in \mathbb{R}^{pd_z} \times \mathbb{R}^{d_x+pd_f}$ is defined as the Jacobian of $\tilde{h} : \mathbb{R}^{d_x} \times \mathbb{R}^{pd_f} \to \mathbb{R}^{pd_z}$ at $\tilde{x}_t^\star \in \mathbb{R}^{d_x+pd_f}$. By Algorithm 1, the Gauss-Newton update is thus given by:

$$\begin{aligned} \overline{\Sigma}_t &\leftarrow (J^\top J)^{-1} \\ &= (\Sigma_t^{-1} + H_t^\top \Sigma_v H_t)^{-1} \\ &= \Sigma_t - \Sigma_t H_t^\top (\Sigma_v^{-1} + H_t \Sigma_t^{-1} H^\top)^{-1} H_t \Sigma_t, \\ \overline{\mu_t} &\leftarrow \mu_t - (J^\top J)^{-1} J^\top C_2(\tilde{x}_t^\star) \\ &= \mu_t - (\Sigma_t^{-1} + H_t^\top \Sigma_v^{-1} H_t)^{-1} \begin{bmatrix} \Sigma_t^{-1/2} & -H_t^\top \Sigma_v^{-1/2} \end{bmatrix} \begin{bmatrix} 0 \\ \Sigma_v^{-1/2}(z_{t,1:p} - \tilde{h}(\mu_t)) \end{bmatrix} \\ &= \mu_t + (\Sigma_t^{-1} + H_t^\top \Sigma_v^{-1} H_t)^{-1} H_t^\top \Sigma_v^{-1} \big(z_{t,1:p} - \tilde{h}(\mu_t)\big), \\ &= \mu_t + \Sigma_v^{-1} H_t^\top (\Sigma_t^{-1} + H_t^\top \Sigma_v^{-1} H_t)^{-1} \big(z_{t,1:p} - \tilde{h}(\mu_t)\big), \end{aligned}$$

which are identical to the feature update equations for the mean and covariance matrix in the Extended Kalman Filter algorithm, i.e. (4) and (5) respectively. Note that, in the final step, we have used a variant of the Woodbury Matrix Identity. □

Theorem 7.3.3. *The state propagation step of the standard EKF SLAM algorithm (Alg. 9) is equivalent to applying a Marginalization step to* $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \to \mathbb{R}$*, given by:*

$$c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) := \|\tilde{x}_t - \overline{\mu}_t\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}.$$

Proof. Intuitively, the state propagation step marginalizes out $\tilde{x}_t \in \mathbb{R}^{d_x}$ and retain $x_{t+1} \in \mathbb{R}^{d_x}$. In other words, in the notation of our Marginalization algorithm submodule, we have:

$$\begin{aligned} \tilde{x}_{t,K} &= x_{t+1} \in \mathbb{R}^{d_x+pd_f}, \\ \tilde{x}_{t,M} &= \tilde{x}_t \in \mathbb{R}^{d_x+pd_f}. \end{aligned}$$

To apply a marginalization step, our first task is to find vectors $C_K(x_K) = C_K(\tilde{x}_t)$ and $C_M(x_K, x_M) = C_M(\tilde{x}_t, x_{t+1})$ of appropriate dimensions such that $c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) = C_K(x_{t+1})^\top C_K(x_{t+1}) + C_M(\tilde{x}_t, x_{t+1})^\top C_M(\tilde{x}_t, x_{t+1})$. A natural choice is furnished by $C_K(x_{t+1}) \in \mathbb{R}$ and $C_M(\tilde{x}_t, x_{t+1}) \in \mathbb{R}^{d_x}$, as defined below:

$$\begin{aligned} c_K(x_{t+1}) &= 0 \\ c_M(\tilde{x}_t, x_{t+1}) &= \|\tilde{x}_t - \overline{\mu_t}\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1} - g(x_t)\|^2_{\Sigma_w^{-1}}. \end{aligned}$$

where we have identified the following parameters, in the language of a Marginalization step (Section 2.1):

$$\begin{aligned} C_K(\tilde{x}_{t,K}) &= 0 \in \mathbb{R} \\ C_M(\tilde{x}_{t,K}, \tilde{x}_{t,M}) &= \begin{bmatrix} \bar{\Sigma}_t^{-1/2}(\tilde{x}_t - \overline{\mu_t}) \\ \Sigma_w^{-1/2}\big(x_{t+1} - g(x_t)\big) \end{bmatrix} \in \mathbb{R}^{2d_x + pd_f}. \end{aligned}$$

For convenience, we will define the pose and feature track components of the mean $\mu_t \in \mathbb{R}^{d_x+pd_f}$ by $\mu_t := (\mu_{t,x}, \mu_{t,f}) \in \mathbb{R}^{d_x+pd_f}$, with $\mu_{t,x} \in \mathbb{R}^{d_x}$ and $\mu_{t,f} \in \mathbb{R}^{pd_f}$, respectively. This mirrors our definition of $x_t \in \mathbb{R}^{d_x}$ and $f_{t,1:p} \in \mathbb{R}^{pd_f}$ as the components of the full state $\tilde{x}_t := (x_t, f_{t,1:p}) \in \mathbb{R}^{d_x+pd_f}$. In addition, we will define the components of $\bar{\Sigma}_t^{-1/2} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$ and $\bar{\Sigma}_t^{-1} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}$ by:

$$\begin{aligned} \begin{bmatrix} \Omega_{t,xx} & \Omega_{t,xf} \\ \Omega_{t,fx} & \Omega_{t,ff} \end{bmatrix} &:= \bar{\Sigma}_t^{-1} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}, \\ \begin{bmatrix} \Lambda_{t,xx} & \Lambda_{t,xf} \\ \Lambda_{t,fx} & \Lambda_{t,ff} \end{bmatrix} &:= \bar{\Sigma}_t^{-1/2} \in \mathbb{R}^{(d_x+pd_f)\times(d_x+pd_f)}, \end{aligned}$$

where $\Sigma_{t,xx}, \Lambda_{t,xx} \in \mathbb{R}^{d_x \times d_x}$, $\Sigma_{t,xf}, \Lambda_{t,xf} \in \mathbb{R}^{d_x \times pd_f}$, $\Sigma_{t,fx}, \Lambda_{t,fx} \in \mathbb{R}^{pd_f \times d_x}$, and $\Sigma_{t,ff}, \Lambda_{t,ff} \in \mathbb{R}^{pd_f \times pd_f}$. Using the above definitions, we can reorder the residuals in $C_K \in \mathbb{R}$ and $C_M \in \mathbb{R}^{2d_x+pd_f}$, and thus redefine them by:

$$\begin{aligned} C_K(\tilde{x}_{t,K}) &= 0 \in \mathbb{R} \\ C_M(\tilde{x}_{t,K}, \tilde{x}_{t,M}) &= \begin{bmatrix} \Lambda_{t,xx}(x_t - \mu_{t,x}) + \Lambda_{t,xf}(f_{t,1:p} - \mu_{t,f}) \\ \Sigma_w^{-1/2}(x_{t+1} - g(x_t)) \\ \Lambda_{t,fx}(x_t - \mu_{t,x}) + \Lambda_{t,ff}(f_{t,1:p} - \mu_{t,f}) \end{bmatrix} \in \mathbb{R}^{2d_x + pd_f}. \end{aligned}$$

Our state variables and cost functions for the Gauss-Newton algorithm submodule are:

$$\begin{aligned} \overline{x_M^\star} &= \tilde{x}_t^\star = \overline{\mu_t} \in \mathbb{R}^{d_x+pd_f}, \\ \overline{x_K^\star} &= g(\tilde{x}_t^\star) = g(\overline{\mu_t}) \in \mathbb{R}^{d_x+pd_f}, \\ C_K(\tilde{x}_{t,K}^\star) &= 0 \in \mathbb{R}, \\ C_M(\tilde{x}_{t,K}^\star, \tilde{x}_{t,M}^\star) &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{2d_x+pd_f}, \\ J_M &= \begin{bmatrix} O & \Lambda_{xf} \\ \Sigma_w^{-1/2} & O \\ O & \Lambda_{ff} \end{bmatrix} \in \mathbb{R}^{(2d_x+pd_f)\times(d_x+pd_f)} \\ J_K &= \begin{bmatrix} \Lambda_{xx} \\ -\Sigma_w^{-1/2} G_t \\ \Lambda_{xf} \end{bmatrix} \in \mathbb{R}^{(2d_x+pd_f)\times d_x}, \end{aligned}$$

where we have defined G_t to be the Jacobian of $g : \mathbb{R}^{d_x} \to \mathbb{R}^{d_x}$ at $\overline{\mu_{t,x}} \in \mathbb{R}^{d_x}$, i.e.:

$$G_t := \left. \frac{\partial g}{\partial x_t} \right|_{x_t = \overline{\mu_{t,x}}}$$

Applying the Marginalization equations, we thus have:

$$\begin{aligned}
\mu_{t+1} &\leftarrow \tilde{x}_{t,K} - \Sigma_{t+1} J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] C_M(\overline{x_K^\star}, \overline{x_M^\star}) \\
&= g(\overline{\mu_t}), \\
\Sigma_{t+1} &\leftarrow \big(J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top \big] J_K \big)^{-1}, \\
&= \big(J_K^\top J_K - J_K^\top J_M (J_M^\top J_M)^{-1} J_M^\top J_K \big)^{-1}, \\
&= \Bigg(\begin{bmatrix} \Sigma_w^{-1} & O \\ O & \Lambda_{fx}\Lambda_{xf} + \Lambda_{ff}^2 \end{bmatrix} - \begin{bmatrix} -\Sigma_w^{-1} G_t \\ \Lambda_{fx}\Lambda_{xx} + \Lambda_{ff}\Lambda_{fx} \end{bmatrix} (\Lambda_{xx}^2 + \Lambda_{xf}\Lambda_{fx} + G_t^\top \Sigma_w^{-1} G_t)^{-1} \\
&\qquad \cdot \begin{bmatrix} -G_t^\top \Sigma_w^{-1} & \Lambda_{xx}\Lambda_{xf} + \Lambda_{fx}\Lambda_{ff} \end{bmatrix} \Bigg)^{-1} \\
&= \left(\begin{bmatrix} \Sigma_w^{-1} & O \\ O & \Omega_{ff} \end{bmatrix} - \begin{bmatrix} -\Sigma_w^{-1} G_t \\ \Omega_{fx} \end{bmatrix} (\Omega_{xx} + G_t^\top \Sigma_w^{-1} G_t)^{-1} \begin{bmatrix} -G_t^\top \Sigma_w^{-1} & \Omega_{xf} \end{bmatrix} \right)^{-1}
\end{aligned}$$

To show that this is indeed identical to the propagation equation for the covariance matrix in the Extended Kalman Filter algorithm, i.e. Algorithm 6, Line 5, we must show that:

$$\begin{aligned}
&\left(\begin{bmatrix} \Sigma_w^{-1} & O \\ O & \Omega_{ff} \end{bmatrix} - \begin{bmatrix} -\Sigma_w^{-1} G_t \\ \Omega_{fx} \end{bmatrix} (\Omega_{xx} + G_t^\top \Sigma_w^{-1} G_t)^{-1} \begin{bmatrix} -G_t^\top \Sigma_w^{-1} & \Omega_{xf} \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} G_t \overline{\Sigma}_{t,xx} G_t^\top + \Sigma_w & G_t \overline{\Sigma}_{t,xf} \\ \overline{\Sigma}_{t,xf} G_t^\top & \overline{\Sigma}_{t,ff} \end{bmatrix}
\end{aligned}$$

This follows by brute-force expanding the above block matrix components, and applying Woodbury's Matrix Identity, along with the definitions of $\Sigma_{t,xx}$, $\Lambda_{t,xx}$, $\Sigma_{t,xf}$, $\Lambda_{t,xf}$, $\Sigma_{t,fx}$, $\Lambda_{t,fx}$, $\Sigma_{t,ff}$, and $\Lambda_{t,ff}$. □

MSCKF, Setup

The Multi-State Constrained Kalman Filter (MSCKF) algorithm iteratively refines the mean and covariance of a MSCKF full state, consisting of the most recent IMU state and a sliding window of n poses. (Algorithm 10) In particular, when a set of new IMU measurements is obtained, the MSCKF full state is propagated forward in time. When a new image measurement arrives, the current pose is appended to the MSCKF full state vector. Features not observed in the current pose are marginalized. If the number of poses maintained in the MSCKF full state, denoted n, exceeds a pre-specified upper bound $N_{\max}$, then features common to every third currently maintained pose, starting from the second oldest pose, are marginalized. Below, we discuss the key components of the MSCKF algorithm—the IMU state, poses, MSCKF full states, features, image measurements, dynamics, pose augmentation and measurement maps—in more detail.

The IMU state $x_{t,\text{IMU}}$ takes the form:

$$x_{t,\text{IMU}} := (q_{WS}, v_S, b_g, b_a, r_{WS})(t) \in \mathbb{R}^3 \times \mathbb{H}_u \times \mathbb{R}^9, \tag{7.8}$$

where we use S and W to represent the sensor frame and world frame, respectively. Here, $r_{WS} \in \mathbb{R}^3$ denotes the position of the IMU sensor frame represented in the world frame, $q_{WS} \in \mathbb{H}_u$ denotes the unit quaternion of axis rotation from world frame to IMU sensor frame, and $R(q_{WS}) \in SO(3)$ denotes the rotation matrix associated with q_{WS}. Moreover, $v_S \in \mathbb{R}^3$ denotes the linear velocity of the IMU sensor frame relative to the world frame, as represented in the world frame, while $b_g \in \mathbb{R}^3$ and $b_a \in \mathbb{R}^3$ denote the sensor biases of the gyroscope and accelerometer, respectively. Finally, $\tilde{\omega}_S \in \mathbb{R}^3$ and $\tilde{a}_S \in \mathbb{R}^3$ denote gyroscope and accelerometer measurements, respectively.

For convenience, define $\mathcal{X}_{\text{IMU}} := \mathbb{R}^3 \times \mathbb{H}_u \times \mathbb{R}^9$ and $\mathcal{X}'_{\text{IMU}} := \mathbb{R}^3 \times \mathbb{H} \times \mathbb{R}^9$. The continuous-time IMU dynamics map $g_{\text{IMU},ct} : \mathcal{X}_{\text{IMU}} \to \mathcal{X}'_{\text{IMU}}$ is given by:

$$\begin{aligned}
\dot{q}_{WS} &= q_{WS} \star \frac{1}{2} \begin{bmatrix} 0 \\ \tilde{\omega}_S - b_g - w_g \end{bmatrix}, \\
\dot{b}_g &= w_{b_g}, \\
\dot{v}_S &= R(q_{WS})^\top \big(\tilde{a}_S - b_a + w_a\big) + g_W, \\
\dot{b}_a &= w_{b_a}, \\
\dot{r}_{WS} &= v_S.
\end{aligned}$$

where $\star$ denotes quaternion multiplication, and $w_g, w_a, w_{b_g}, w_{b_a} \in \mathbb{R}^3$ denote zero-mean standard Gaussian noise.

Each pose $x_k \in \{x_1, \cdots, x_n\}$ currently maintained in the sliding window of poses takes the form $x_k := (q_{WC_k}, r_{WC_k}) \in \mathbb{H}_u \times \mathbb{R}^3$, where $r_{WC_k} \in \mathbb{R}^3$ denotes the position of the camera at pose x_k in the world frame, while $q_{WC_k} \in \mathbb{H}_u$ denotes the quaternion associated with the axis rotation from the world frame to the camera frame at pose $x_k \in \mathbb{H} \times \mathbb{R}^3$. For convenience, we define $\mathcal{X}_p := \mathbb{H}_u \times \mathbb{R}^3$.

The MSCKF full state maintained throughout the operation of the MSCKF algorithm contains the IMU state at the current time, as well as a collection of n poses, where n is constrained to remain below a pre-specified, fixed upper bound $N_{\max}$:

$$\tilde{x}_t := (x_{t,\text{IMU}}, x_1, \cdots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n. \tag{7.9}$$

The state space is thus $\mathcal{X} := \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.

When a new image measurement arrives, the estimate of the current camera pose in the IMU frame, denoted $x_{n+1}^{\text{IMU}} \in \mathcal{X}_{\text{IMU}}$, is transformed to the global frame, and appended

Algorithm 10: Multi-State Constrained Kalman Filter, Standard Formulation.

```
Data: Prior distribution on $x_{\text{IMU},0} \in \mathcal{X}_p$: $\mathcal{N}(\mu_0, \Sigma_0)$, dynamics and measurement noise
      covariances $\Sigma_w \in \mathbb{R}^{d_x \times d_x}$, $\Sigma_v \in \mathbb{R}^{d_x \times d_z}$, discrete-time dynamics map
      $g_{\text{IMU}} : \mathbb{R}^{d_{\text{IMU}}} \times \mathbb{R}^{d_{\text{IMU}}}$, measurement map $h : \mathcal{X}_p \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$, time horizon $T$, pose
      transformation $\psi : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \times \mathcal{X}_p \to \mathcal{X}_p$ (IMU $\to$ global).
Result: Estimates $\hat{x}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ for all desired timesteps $t \leq T$, where $n :=$ number
        of poses in $\hat{x}_t$ at time $t$.
1  $S_z, S_x, S_{z,1}, S_{z,2} \leftarrow \phi$
2  $(n, p) \leftarrow (0, 0)$
3  for $t = 0, \cdots, T$ do
4      while new image $\mathcal{I}$ with new pose $x_{n+1} \in \mathcal{X}_p$ recorded, next IMU measurement not
        yet received do
5          $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n, \Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)} \leftarrow$ Alg. 11 ($\tilde{x}_t$, $\mu_t$, $\Sigma_t$, $x_{n+1}$, $x_{n+1}^{\text{IMU}}$,
            $\psi(\cdot)$)
6          $\{z_{n+1,j} |$ feature $j$ is observed at $x_{n+1}\} \leftarrow$ Feature measurements at $x_{n+1}$
7          $\{f_j^\star |$ Feature $j$ is observed at $x_{n+1}\} \leftarrow$ Feature position estimates at $x_{n+1}$.
8          Record new estimates of existing features and first estimate of new features at
            $x_{n+1} \in \mathcal{X}_p$.
9          $S_z \leftarrow S_z \cup \big\{(x_{n+1}, f_j) |$ Feature $j$ observed at $n+1\big\}$
10         $n \leftarrow n + 1$
11         if $n \geq N_{\max} - 1$ then
12             $S_x \leftarrow \{x_i | i \bmod 3 = 2,$ and $1 \leq i \leq n.\}$
13             $S_{z,1} \leftarrow \big\{(x_i, f_j) \in S_z \big| x_i \in S_x,$ feature $j$ observed at each pose in $S_x\big\}$
14         end
15         $S_{z,2} \leftarrow \big\{(x_i, f_j) \in S_z | x_i \in x_{1:n},$ feature $j$ observed at $x_i$ but not at $x_n\big\}$.
16         $S_f \leftarrow \big\{f_j \big| \exists x_i \in x_{1:n}$ s.t. $(x_i, f_j) \in S_{z,1} \cup S_{z,2}\big\}$
17         if $S_f \neq \phi$ then
18             $\overline{\mu_t} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n, \overline{\Sigma_t} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)} \leftarrow$ Alg. 12 ($\tilde{x}_t$, $\mu_t$, $\Sigma_t$, $x_{n+1}$,
                $S_{z,1} \cup S_{z,2}$, $S_f$, $h(\cdot)$)
19             $\hat{x}_t \leftarrow \overline{\mu_t} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
20         end
21         $S_z \leftarrow S_z \backslash (S_{z,1} \cup \{(x_i, f_j) | x_i \in S_x\})$
22         Reindex poses and features, in ascending order of index, i.e., $\{x_1, \cdots, x_{n-|S_x|}\}$ and
            $\{f_1, \cdots, f_{p-|S_f|}\}$.
23         $(p, n) \leftarrow (p - |S_f|, n - |S_x|)$
24     end
25     if $t < T$ then
26         $\mu_{t+1} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n, \Sigma_{t+1} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)} \leftarrow$ Alg. 13, MSCKF State
            Propagation ($\tilde{x}_t$, $\overline{\mu_t}$, $\overline{\Sigma_t}$)
27     end
28 end
29 return $\hat{x}_0, \cdots \hat{x}_T \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n$
```

to the MSCKF full state $\tilde{x}_t$. This coordinate transformation is realized by the map $\psi : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \to \mathcal{X}_p$, defined by:

$$\begin{aligned}
&\psi\big(q_{WS}, v_S, b_g, b_a, r_{WS}, q_{WC_1}, r_{WC_1}, \cdots, q_{WC_n}, r_{WC_n}, q_{WI_{n+1}}, r_{WI_{n+1}}\big) \\
=&\big(q_{IC} \star q_{WI_{n+1}}, r_{WI_{n+1}} + C(q_{WI_{n+1}}) r_{IC}\big) \\
:=&\big(q_{WC_{n+1}}, r_{WC_{n+1}}\big),
\end{aligned}$$

where q_{IC} denotes the quaternion encoding the (fixed) transformation from the IMU frame to the camera frame. In summary, the MSCKF algorithm defines the new pose x_{n+1} and updates the MSCKF full state $\tilde{x}_t$ as follows:

$$\begin{aligned}
x_{n+1} &\leftarrow \psi(\tilde{x}_t, x_{n+1}^{\text{IMU}}) \in \mathcal{X}_p, \\
\tilde{x}_t &\leftarrow (\tilde{x}_t, x_{n+1}) = \big(\tilde{x}_t, \psi(\tilde{x}_t, x_{n+1}^{\text{IMU}})\big) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^{n+1},
\end{aligned}$$

with the map ψ as defined above.

When new feature position estimates are detected from a new image measurement, the new camera pose corresponding to this image measurement is appended to $\tilde{x}_t$, and n is incremented by 1. If $n = N_{\max}$ the upper limit $N_{\max}$, a third of all old poses in $\tilde{x}_t$ is discarded, starting from the second oldest pose. Then, feature measurements, corresponding to features unobserved at the current pose, are marginalized and used to update the mean and covariance of the new MSCKF full state $\tilde{x}_t$.

As is the case with the EKF algorithm, we assume that the image measurement space and feature space are given by $\mathbb{R}^{d_z}$ and $\mathbb{R}^{d_f}$, respectively, with $d_z = 2$ and $d_f = 3$. Throughout the duration of the MSCKF algorithm, poses and features are added into, dropped from, and marginalized from the MSCKF full state. Suppose at a given time, the MSCKF maintains n poses in the *MSCKF full state* $\tilde{x}_t$, and retains measurements of p features. For each pose $i \in \{1, \cdots, n\}$ and feature $j \in \{1, \cdots, p\}$ currently maintained in the SLAM algorithm, if feature j were detected at pose i, let $z_{i,j} \in \mathbb{R}^{d_z}$ denote the associated feature measurement. For the MSCKF, the measurement map $h : \mathcal{X}_{\text{IMU}} \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$ is given by:

$$z_{i,j} = h(x_i, f_j) = \frac{1}{(R(q_{WC_k}) f_j - r_{WC_k})_z} \begin{bmatrix} (R(q_{WC_k}) f_j - r_{WC_k})_x \\ (R(q_{WC_k}) f_j - r_{WC_k})_y \end{bmatrix} (t) + v_{i,j}.$$

where $R(q_{WC_k}) \in SO(3)$ denotes the rotation matrix associated with the quaternion q_{WC_k}, $f_j \in \mathbb{R}^3$ denotes the position of feature j in the world frame, while the subscript indices "x, y, z" refer to the respective coordinates of the vector $R(q_{WS}) f_j - r_{WS} \in \mathcal{X}_p$. Meanwhile, $v_{i,j} \in \mathbb{R}^{d_z}$ denotes zero-mean standard Gaussian noise in the measurement at time t, with covariance matrix $\Sigma_v \in \mathbb{R}^{d_z \times d_z}$, $\Sigma_v \succ 0$.

When a new image measurement is received, the MSCKF algorithm performs marginalization, described in Section 3.3, using two sets of feature measurements—the set of all feature measurements common to old poses x_i to be dropped, denoted $S_{z,1}$, as well as the set of all feature measurements of features f_j not seen in the current pose, denoted $S_{z,2}$. These are more precisely defined in Section 4.2. The measurement vector used for marginalization, denoted $\tilde{z} \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}| d_z}$, is then given by concatenating the q residual measurements obtained at times $t-n+1, \cdots, t$, i.e.:

$$\tilde{z} := \{z_{i,j} | (x_i, f_j) \in S_{z,1} \cup S_{z,2}\} \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}| d_z}.$$

Algorithm 11: Multi-State Constrained Kalman Filter, Pose Augmentation Sub-block.

```
Data: MSCKF state $\tilde{x}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, with mean $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and
      covariance $\Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$, New pose $x_{n+1} \in \mathcal{X}_p$, measurement of
      new pose in IMU frame $x_{n+1}^{\text{IMU}} \in \mathcal{X}_p$, Transformation of poses from IMU frame
      to global frame $\psi : \mathbb{R}^{(d_{\text{IMU}}+nd_x)} \times \mathcal{X}_p \to \mathcal{X}_p$
Result: Updated MSCKF state mean $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and covariance
        $\Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$, updated number of poses $n$.
1 $\tilde{x}_t \leftarrow (\tilde{x}_t, x_{n+1}) \in \mathbb{R}^{d_{\text{IMU}}+(n+1)d_x}$, where $x_{n+1} \in \mathcal{X}_p$ is the new pose vector.
2 $\{z_{n+1,j} |$ Feature $j$ is observed at pose $n+1\} \leftarrow$ Feature measurements at pose $x_{n+1}$
3 $\{f_j^\star |$ Feature $j$ is observed at pose $x_{n+1}\} \leftarrow$ Feature position estimates at pose $x_{n+1}$.
4 $\mu_t \leftarrow (\mu_t, \psi(\mu_t, x_{n+1}^{\text{IMU}})) \in \mathbb{R}^{d_{\text{IMU}}+(n+1)d_x}$, where $\mu_{t,\text{IMU}} \in \mathbb{R}^{d_{\text{IMU}}} :=$ IMU component of
    $\mu_t$, $x_{n+1}^{\text{IMU}} \in \mathcal{X}_p :=$ pose estimate of $x_{n+1}$ from the IMU frame.
5 $\Sigma_t \leftarrow \begin{bmatrix} I_{d_{\text{IMU}}+(n+1)d_x} \\ \frac{\partial \psi}{\partial(\tilde{x}_t, x_{n+1}^{\text{IMU}})} \end{bmatrix} \Sigma_t \begin{bmatrix} I_{d_{\text{IMU}}+(n+1)d_x} \\ \frac{\partial \psi}{\partial(\tilde{x}_t, x_{n+1}^{\text{IMU}})} \end{bmatrix}^\top$
6 return $\mu_t \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n$, $\Sigma_t \in \mathbb{R}^{(d_{IMU}+nd_x)\times(d_{IMU}+nd_x)}$, $n \geq 0$
```

Proofs from Section 4.2

Theorem 7.3.4. *The pose augmentation step of the standard MSCKF SLAM algorithm (Alg. 11) is equivalent to applying a Gauss-Newton step to $c_{MSCKF,t,1} : \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \to \mathbb{R}$, given by:*

$$c_{MSCKF,t,1}(\tilde{x}_t, x_{n+1}) = \|\tilde{x}_t \boxminus \mu_t\|_{\Sigma_t^{-1}}^2 + \epsilon^{-1} \|x_{n+1} \boxminus \psi(\tilde{x}_t, x_{n+1}^{IMU})\|_2^2,$$

and taking $\epsilon \to 0$ in the resulting (augmented) mean μ_t and covariance Σ_t.

Proof. We claim that from an optimization perspective, the state augmentation step is equivalent to applying one Gauss-Newton step to the cost function $c_{MSCKF,t,1}(\tilde{x}_t, x_{n+1})$, specified above, and then taking the limit $\epsilon \to 0$ in the resulting augmented mean $\mu_t(\epsilon) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^{(n+1)}$ and augmented covariance $\mu_t(\epsilon) \in \mathbb{R}^{(d_{\text{IMU}}+(n+1)d_x)\times(d_{\text{IMU}}+(n+1)d_x)}$.

To apply a Gauss-Newton step, our first task is to find a vector $C(\tilde{x}_t, x_{n+1})$ of an appropriate dimension such that $c_{MSCKF,t,1}(\tilde{x}_t, x_{n+1}) = C_1(\tilde{x}_t, x_{n+1})^\top C_1(\tilde{x}_t, x_{n+1})$. A natural

Algorithm 12: Multi-State Constrained Kalman Filter, Feature Update Sub-block.

```
Data: MSCKF state $\tilde{x}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, with mean $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and
      covariance $\Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$, Set of image measurements for
      marginalization $S_{z,1} \cup S_{z,2}$, Set of features to marginalize $S_f$, measurement
      map $h : \mathcal{X}_p \times \mathbb{R}^{d_f} \to \mathbb{R}^{d_z}$.
Result: Updated MSCKF state mean $\overline{\mu_t} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and covariance
        $\overline{\Sigma_t} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$.
1  $f_{S_f} \in \mathbb{R}^{|S_f|d_f} \leftarrow$ Concatenation of all features in $S_f$
2  $f^\star_{S_f} \in \mathbb{R}^{|S_f|d_f} \leftarrow$ Concatenation of position estimate of all features in $S_f$
3  $\tilde{h}(\tilde{x}_t, f_{S_f}) \in \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z} \leftarrow$ Concatenation of measurement map outputs
     $\{h(x_i, f_j) | (x_i, f_j) \in S_{z,1} \cup S_{z,2}\}$.
4  $\tilde{z} \in \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z} \leftarrow$ Concatenation of feature measurements
     $\{z_{ij} | (x_i, f_j) \in S_{z,1} \cup S_{z,2}\}$.
5  $\tilde{H}_{t,x} \leftarrow \frac{\partial \tilde{h}}{\partial \tilde{x}_t}(\mu_t, f^\star_{S_f}) \in \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z \times (d_{\text{IMU}}+nd_x)}$.
6  $\tilde{H}_{t,f} \leftarrow \frac{\partial \tilde{h}}{\partial f_{S_f}}(\mu_t, f^\star_{S_f}) \in \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z \times |S_f|d_f}$.
7  $\{a_1, \cdots, a_{|S_{z,1}\cup S_{z,2}|d_z - |S_f|d_f}\} \subset \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z} \leftarrow$ Orthonormal basis for $N(\tilde{H}_{t,f}^\top)$.
8  $A \leftarrow \begin{bmatrix} a_1 & \cdots & a_{|S_{z,1}\cup S_{z,2}|d_z - |S_f|d_f} \end{bmatrix} \in \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z \times (|S_{z,1}\cup S_{z,2}|d_z - |S_f|d_f)}$.
9  $QT \leftarrow$ QR Decomposition of $A^\top \tilde{H}_{t,x}$, with
     $Q \in \mathbb{R}^{(|S_{z,1}\cup S_{z,2}|d_z - |S_f|d_f)\times(|S_{z,1}\cup S_{z,2}|d_z - |S_f|d_f)}$, $T \in \mathbb{R}^{(|S_{z,1}\cup S_{z,2}|d_z - |S_f|d_f)\times(d_{\text{IMU}}+nd_x)}$.
10 $\overline{\Sigma_t}^{-1} \leftarrow \Sigma_t^{-1} + T^\top (Q^\top A^\top R A Q)^{-1} T \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$.
11 $\overline{\mu_t} \leftarrow \mu_t \boxplus \big(\Sigma_t^{-1} + T^\top (Q^\top A^\top R A Q)^{-1} T\big)^{-1} T^\top (Q^\top A^\top R A Q)^{-1} \big(\tilde{z} \boxminus \tilde{h}(\tilde{x}_t)\big) \in$
     $\mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
12 $\hat{x}_t \leftarrow \overline{\mu_t} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
13 return $\overline{\mu_t} \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n$, $\overline{\Sigma_t} \in \mathbb{R}^{(d_{IMU}+nd_x)\times(d_{IMU}+nd_x)}$
```

Algorithm 13: Multi-State Constrained Kalman Filter, State Propagation Sub-block.

```
Data: MSCKF state $\tilde{x}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, with mean $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and
      covariance $\Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$, (discrete-time) dynamics map
      $g : \mathbb{R}^{d_{\text{IMU}}} \to \mathbb{R}^{d_{\text{IMU}}}$.
Result: Updated MSCKF state mean $\mu_{t+1} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and covariance
        $\Sigma_{t+1} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$.
1  $(\overline{\mu}_{t,\text{IMU}}, \overline{\mu}_{t,x,1:n}) \leftarrow \overline{\mu_t}$, with $\overline{\mu}_{t,\text{IMU}} \in \mathbb{R}^{d_{\text{IMU}}}$, $\overline{\mu}_{t,x,1:n} \in \mathbb{R}^{nd_x}$.
2  $G_t \leftarrow$ Jacobian of $g_{\text{IMU}} : \mathbb{R}^{d_{\text{IMU}}} \to \mathbb{R}^{d_{\text{IMU}}}$ evaluated at $\overline{\mu}_{t,\text{IMU}} \in \mathbb{R}^{d_{\text{IMU}}}$.
3  $\mu_{t+1} \leftarrow \big(g_{\text{IMU}}(\overline{\mu}_{t,\text{IMU}}), \overline{\mu}_{t,x,1:n}\big) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
4  $\Sigma_{t+1} \leftarrow \begin{bmatrix} G_t & O \\ O & I_{nd_x} \end{bmatrix} \overline{\Sigma_t} \begin{bmatrix} G_t^\top & O \\ O & I_{nd_x} \end{bmatrix} + \begin{bmatrix} \Sigma_w & O \\ O & O \end{bmatrix} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x)\times(d_{\text{IMU}}+nd_x)}$.
5  return $\mu_{t+1} \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n$, $\Sigma_{t+1} \in \mathbb{R}^{(d_{IMU}+nd_x)\times(d_{IMU}+nd_x)}$.
```

choice is furnished by $C_1(\tilde{x}_t, x_{n+1}) \in \mathbb{R}^{d_{\text{IMU}}+(n+1)d_x}$, as defined below:

$$C_1(\tilde{x}_t, x_{n+1}) := \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t \boxminus \mu_t) \\ \epsilon^{-1/2}\big(x_{n+1} \boxminus \psi(\tilde{x}_t, x_{n+1}^{\text{IMU}})\big) \end{bmatrix}.$$

Thus, our parameters for the Gauss-Newton algorithm submodule are:

$$\begin{aligned} (\tilde{x}_t^\star, x_{n+1}^\star) &:= (\mu_t, \psi(\mu_t, x_{n+1}^{\text{IMU}})) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n, \\ C_1(\tilde{x}_t^\star, x_{n+1}^\star) &= \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t^\star - \mu_t) \\ \epsilon^{-1/2}\big(x_{n+1}^\star - \psi(\tilde{x}_t^\star, x_{n+1}^{\text{IMU}})\big) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{d_{\text{IMU}}+(n+1)d_x}, \\ J &= \begin{bmatrix} \Sigma_t^{-1/2} & O \\ -\epsilon^{-1/2}\Psi & \epsilon^{-1/2} I_{d_x} \end{bmatrix} \in \mathbb{R}^{(d_{\text{IMU}}+(n+1)d_x)\times(d_{\text{IMU}}+(n+1)d_x)}, \end{aligned}$$

where $\Psi \in \mathbb{R}^{d_x \times (d_{\text{IMU}}+nd_x)}$ is defined as the Jacobian of $\psi : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \to \mathcal{X}_p$ with respect to $\tilde{x}_t$ at $(\tilde{x}_t^\star, x_{n+1}^{\text{IMU}}) \in \mathbb{R}^{d_{\text{IMU}}+(n_1)d_x}$. By Algorithm 1, the Gauss-Newton update is thus:

$$\begin{aligned} \Sigma_t(\epsilon) \leftarrow (J^\top J)^{-1} &= \begin{bmatrix} \Sigma_t^{1/2} & O \\ \Psi\Sigma_t^{1/2} & \epsilon^{1/2} I_{d_x} \end{bmatrix} \begin{bmatrix} \Sigma_t^{1/2} & \Sigma_t^{1/2}\Psi^\top \\ O & \epsilon^{1/2} I_{d_x} \end{bmatrix} \\ &= \begin{bmatrix} \Sigma_t & \Sigma_t\Phi^\top \\ \Psi\Sigma_t & \Psi\Sigma_t\Psi^\top + \epsilon I_{d_x} \end{bmatrix}, \\ \mu_t(\epsilon) \leftarrow \tilde{x}_t^\star &- (J^\top J)^{-1} J^\top C_1(\tilde{x}_t^\star, x_{n+1}^\star) \\ &= 0. \end{aligned}$$

Taking $\epsilon \to 0$ concludes the proof. □

Theorem 7.3.5. *The feature update step of the standard MSCKF algorithm (Alg. 12) is equivalent to applying a marginalization step to* $c_{MSCKF,t,3} : \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \times \mathbb{R}^{|S_f|d_f} \to \mathbb{R}$, *given by:*

$$c_{MSCKF,t,3}(\tilde{x}_t, f_{S_f}) := \|\tilde{x}_t \boxminus \mu_t\|^2_{\Sigma_t^{-1}} + \sum_{(x_i, f_j) \in S_{z,1} \cup S_{z,2}} \|z_{i,j} \boxminus h(x_i, f_j)\|^2_{\Sigma_v^{-1}},$$

where $f_{S_f} \in \mathbb{R}^{|S_f|d_f}$ *denotes the stacked vector of all feature positions in* S_f *(see Alg. 10).*

Proof. First, we rewrite $c_{MSCKF,t,3}$ as:

$$c_{MSCKF,t,3}(\tilde{x}_t, f_{S_f}) := \|\tilde{x}_t \boxminus \mu_t\|^2_{\Sigma_t^{-1}} + \|\tilde{z} \boxminus \tilde{h}(\tilde{x}_t, f_{S_f})\|^2_{\tilde{\Sigma}_v^{-1}},$$

where $\tilde{z} \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z}$, $\tilde{h} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \times \mathbb{R}^{|S_f|d_f} \to \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z}$: are defined as follows—$\tilde{z}$ denotes the stacked measurement vectors in $\{z_{i,j} | (x_i, f_j) \in S_{z,1} \cup S_{z,2}\} \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z}$, $\tilde{h}(\tilde{x}_t, f_{S_f})$ denotes the stacked outputs of the measurement map in $\{h(x_i, f_j) | (x_i, f_j) \in S_{z,1} \cup S_{z,2}\} \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z}$, and $\tilde{\Sigma}_v := \text{diag}\{\Sigma_v, \cdots, \Sigma_v\} \in \mathbb{R}^{|S_z|d_z \times |S_z|d_z}$.

Essentially, by marginalizing the feature position estimates, this step utilizes information from feature measurements to constrain our state estimates. To accomplish this, we choose our algorithm variables as follows:

$$\begin{aligned}
\tilde{x}_{t,K} &:= \tilde{x}_t = (x_{t,\mathrm{IMU}}, x_1, \cdots, x_n) \in \mathbb{R}^{d_{\mathrm{IMU}}+nd_x+|S_f|d_f}, \\
\tilde{x}_{t,M} &:= f_{S_f} \in \mathbb{R}^{|S_f|d_f}, \\
\overline{x} &:= (\tilde{x}_{t,K}, \tilde{x}_{t,M}) \in \mathbb{R}^{d_{\mathrm{IMU}}+nd_x+|S_f|d_f}, \\
C_M(\tilde{x}_{t,K}, \tilde{x}_{t,M}) &:= \begin{bmatrix} \Sigma_t^{-1/2}(\tilde{x}_t \boxminus \mu_t) \\ \tilde{\Sigma}_v^{-1x/2}\big(\tilde{z} \boxminus \tilde{h}(\tilde{x}_t, f_{S_f})\big) \end{bmatrix} \in \mathbb{R}^{d_{\mathrm{IMU}}+nd_x+|S_{z,1}\cup S_{z,2}|d_z}.
\end{aligned}$$

The Marginalization algorithm block then implies that:

$$\begin{aligned}
J_K &:= \frac{\partial C_M}{\partial \tilde{x}_t}(\overline{\mu_t}, f^\star_{S_f}) = \begin{bmatrix} \Sigma_t^{-1/2} \\ -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,x} \end{bmatrix} \in \mathbb{R}^{(d_{\mathrm{IMU}}+nd_x+|S_{z,1}\cup S_{z,2}|d_z)\times(d_{\mathrm{IMU}}+nd_x)}, \\
J_M &:= \frac{\partial C_M}{\partial f_{S_f}}(\overline{\mu_t}, f^\star_{S_f}) = \begin{bmatrix} O \\ -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f} \end{bmatrix} \in \mathbb{R}^{(d_{\mathrm{IMU}}+nd_x+|S_{z,1}\cup S_{z,2}|d_z)\times|S_f|d_f},
\end{aligned}$$

where we have defined:

$$\begin{aligned}
f^\star_{S_f} \in \mathbb{R}^{|S_f|d_f} &\leftarrow \text{Stacked position estimates of features in } S_f, \\
\tilde{H}_{t,x} &:= \frac{\partial \tilde{h}}{\partial \tilde{x}_t}\tilde{h}(\overline{\mu_t}, f^\star_{S_f}) \in \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z\times(d_{\mathrm{IMU}}+nd_x)}, \\
\tilde{H}_{t,f} &:= \frac{\partial \tilde{h}}{\partial f_{S_f}}(\overline{\mu_t}, f^\star_{S_f}) \in \mathbb{R}^{|S_{z,1}\cup S_{z,2}|d_z\times|S_f|d_f}.
\end{aligned}$$

Recall that the marginalization equations (3.8) and (3.9) in our formulation read:

$$\begin{aligned}
\mu_K &\leftarrow \mu_K - \Sigma_K J_K^\top\big[I - J_M(J_M^\top J_M)^{-1}J_M^\top\big]C_M\big(\tilde{x}_{t,K}, \tilde{x}_{t,M}\big), \\
\Sigma_K &\leftarrow \big(J_K^\top(I - J_M(J_M^\top J_M)^{-1}J_M^\top)J_K\big)^{-1}.
\end{aligned}$$

Substituting in the above expressions for J_K, J_M, and $C_M\big(\overline{\mu_t}, f^\star_{S_f}\big)$, we have:

$$\begin{aligned}
\overline{\Sigma}_t &\leftarrow (J_K^\top(I - J_M(J_M^\top J_M)^{-1}J_M^\top)J_K)^{-1}, \\
&= \left(\begin{bmatrix} \Sigma_t^{-1/2} & -\tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1/2} \end{bmatrix} \begin{bmatrix} I & O \\ O & I - \tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1/2} \end{bmatrix} \begin{bmatrix} \Sigma_t^{1/2} \\ -\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,x} \end{bmatrix} \right)^{-1} \\
&= \big(\Sigma_t^{-1} + \tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1/2}\big[I - \tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1/2}\big]\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,x}\big)^{-1} \\
\overline{\mu_t} &\leftarrow \mu_K - \Sigma_K J_K^\top\big[I - J_M(J_M^\top J_M)^{-1}J_M^\top\big]C_M\big(\overline{\mu_t}, f^\star_{S_f}\big) \\
&= \mu_t + \big(\Sigma_t^{-1} + \tilde{H}_{t,x}^\top\tilde{\Sigma}_v^{-1/2}\big[I - \tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top\tilde{\Sigma}_v^{-1/2}\big]\cdot\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,x}\big)^{-1}
\end{aligned}$$

$$\cdot \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1/2}\big[I - \tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1} \cdot \tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1/2}\big]\tilde{\Sigma}_v^{-1/2}\big(\tilde{z} - \tilde{h}(\tilde{x}_t, f_{S_f})\big).$$

Comparing with the update step in MSCKF, i.e., (10) and (9), reproduced below:

$$\begin{aligned}\overline{\Sigma}_t^{-1} &\leftarrow \Sigma_t^{-1} + T^\top(Q^\top A^\top \tilde{\Sigma}_v AQ)^{-1}T, \\ \overline{\mu_t} &\leftarrow \mu_t + \big(\Sigma_t^{-1} + T^\top(Q^\top A^\top \tilde{\Sigma}_v AQ)^{-1}T\big)^{-1}T^\top(Q^\top A^\top \tilde{\Sigma}_v AQ)^{-1}\big(\tilde{z} - \tilde{h}(\tilde{x}_t, f_{S_f})\big),\end{aligned}$$

we find that it suffices to show:

$$\begin{aligned}T^\top(Q^\top A^\top \tilde{\Sigma}_v AQ)^{-1} &= \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1/2}\big[I - \tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1/2}\big] \cdot \tilde{\Sigma}_v^{-1/2} \\ &= \tilde{H}_{t,x}\tilde{\Sigma}_v^{-1}\tilde{H}_{t,x} - \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}.\end{aligned}$$

To see this, recall that A is defined as a full-rank matrix whose columns span $N(\tilde{H}_{t,f}^\top)$. Thus:

$$(\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f})^\top \cdot \tilde{\Sigma}_v^{1/2}AQ = \tilde{H}_{t,f}^\top AQ = O.$$

In other words, the columns of $\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}$ and of $\tilde{\Sigma}_v^{1/2}AQ$ form bases of orthogonal subspaces whose direct sum equals $\mathbb{R}^{nqd_z}$. We thus have:

$$\tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1/2} + \tilde{\Sigma}_v^{1/2}AQ(Q^\top A^\top \tilde{\Sigma}_v AQ)^{-1}Q^\top A^\top \tilde{\Sigma}_v^{1/2} = I,$$

which in turn implies that:

$$\begin{aligned}T^\top(Q^\top A^\top \tilde{\Sigma}_v AQ)^{-1} &= \tilde{H}_{t,x}^\top AQ(Q^\top A^\top \tilde{\Sigma}_v AQ)^{-1}Q^\top A^\top \\ &= \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1/2}(\tilde{\Sigma}_v^{1/2}AQ)(Q^\top A^\top \tilde{\Sigma}_v^{1/2} \cdot \tilde{\Sigma}_v^{1/2}AQ)^{-1}(Q^\top A^\top \tilde{\Sigma}_v^{1/2})\tilde{\Sigma}_v^{-1/2} \\ &= \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1/2}\big(I - \tilde{\Sigma}_v^{-1/2}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1/2}\big)\tilde{\Sigma}_v^{-1/2} \\ &= \tilde{H}_{t,x}\tilde{\Sigma}_v^{-1}\tilde{H}_{t,x} - \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f}(\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1}\tilde{H}_{t,f})^{-1}\tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1},\end{aligned}$$

as claimed.

□

Theorem 7.3.6. *The state propagation step of the standard MSCKF SLAM algorithm (Alg. 13) is equivalent to applying a Marginalization step once to* $c_{MSCKF,t,5} : \mathbb{R}^{2d_{IMU}+nd_x} \to \mathbb{R}$, *given by:*

$$c_{MSCKF,t,5}(\tilde{x}_t, x_{t+1,IMU}) := \|\tilde{x}_t \boxminus \overline{\mu_t}\|^2_{\overline{\Sigma}_t^{-1}} + \|x_{t+1,IMU} \boxminus g_{IMU}(x_{t,IMU})\|^2_{\Sigma_t^{-1}}.$$

Proof. We claim that from an optimization perspective, the update step is equivalent to applying one marginalization step to the cost function $c_{MSCKF,t,5}(\tilde{x}_t, x_{t+1,\text{IMU}})$ specified above. In particular, we wish to marginalize out $x_{t,\text{IMU}} \in \mathcal{X}_{\text{IMU}}$ and retain $x_{t+1,\text{IMU}} \in \mathcal{X}_{\text{IMU}}$; in other words, in the notation of our Marginalization algorithm submodule, we have:

$$\tilde{x}_{t,K} := (x_{t+1,\text{IMU}}, x_1, \cdots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n,$$

$$\tilde{x}_{t,M} := x_{t,\mathrm{IMU}} \in \mathcal{X}_{\mathrm{IMU}}.$$

To apply a marginalization step, our first task is to find vectors $C_K(x_K) = C_K(\tilde{x}_t)$ and $C_M(x_K, x_M) = C_M(\tilde{x}_t, x_{t+1,\mathrm{IMU}})$ of appropriate dimensions such that $c_{MSCKF,t,5}(\tilde{x}_t, x_{t+1,\mathrm{IMU}}) = C_K(x_{t+1,\mathrm{IMU}})^\top C_K(x_{t+1,\mathrm{IMU}}) + C_M(\tilde{x}_t, x_{t+1,\mathrm{IMU}})^\top C_M(\tilde{x}_t, x_{t+1,\mathrm{IMU}})$. A natural choice is furnished by $C_K(x_{t+1,\mathrm{IMU}}) \in \mathbb{R}$ and $C_M(\tilde{x}_t, x_{t+1,\mathrm{IMU}}) \in \mathcal{X}_p$, as defined below:

$$\begin{aligned} C_K(\tilde{x}_{t,K}) &= 0 \in \mathbb{R} \\ C_M(\tilde{x}_{t,K}, \tilde{x}_{t,M}) &= \begin{bmatrix} \bar{\Sigma}_t^{-1/2}(\tilde{x}_t - \overline{\mu_t}) \\ \Sigma_w^{-1/2}\big(x_{t+1,\mathrm{IMU}} - g_{\mathrm{IMU}}(x_{t,\mathrm{IMU}})\big) \end{bmatrix} \in \mathbb{R}^{2d_{\mathrm{IMU}}+nd_x}. \end{aligned}$$

For convenience, we will define the IMU state and pose components of the mean $\mu_t \in \mathcal{X}_{\mathrm{IMU}} \times (\mathcal{X}_p)^n$ by $\mu_t := (\mu_{t,\mathrm{IMU}}, \mu_{t,\mathrm{IMU}}) \in \mathcal{X}_{\mathrm{IMU}} \times (\mathcal{X}_p)^n$, with $\mu_{t,\mathrm{IMU}} \in \mathcal{X}_p$ and $\mu_{t,x} \in (\mathcal{X}_p)^n$, respectively. This mirrors our definition of $x_t \in \mathcal{X}_p$ and $x_{n+1} \in (\mathcal{X}_p)^n$ as the components of the full state $\tilde{x}_t := (x_t, x_{n+1}) \in \mathcal{X}_{\mathrm{IMU}} \times (\mathcal{X}_p)^n$. In addition, we will define the components of $\bar{\Sigma}_t^{-1/2} \in \mathbb{R}^{(d_{\mathrm{IMU}}+nd_x)\times(d_{\mathrm{IMU}}+nd_x)}$ and $\bar{\Sigma}_t^{-1} \in \mathbb{R}^{(d_{\mathrm{IMU}}+nd_x)\times(d_{\mathrm{IMU}}+nd_x)}$ by:

$$\begin{aligned} \begin{bmatrix} \Omega_{t,\mathrm{IMU},\mathrm{IMU}} & \Omega_{t,\mathrm{IMU},x} \\ \Omega_{t,x,\mathrm{IMU}} & \Omega_{t,x,x} \end{bmatrix} &:= \bar{\Sigma}_t^{-1} \in \mathbb{R}^{(d_{\mathrm{IMU}}+nd_x)\times(d_{\mathrm{IMU}}+nd_x)}, \\ \begin{bmatrix} \Lambda_{t,\mathrm{IMU},\mathrm{IMU}} & \Lambda_{t,\mathrm{IMU},x} \\ \Lambda_{t,x,\mathrm{IMU}} & \Lambda_{t,x,x} \end{bmatrix} &:= \bar{\Sigma}_t^{-1/2} \in \mathbb{R}^{(d_{\mathrm{IMU}}+nd_x)\times(d_{\mathrm{IMU}}+nd_x)}, \end{aligned}$$

with the dimensions of the above block matrices given by $\Sigma_{t,\mathrm{IMU},\mathrm{IMU}}, \Lambda_{t,\mathrm{IMU},\mathrm{IMU}} \in \mathbb{R}^{d_{\mathrm{IMU}}\times d_{\mathrm{IMU}}}$, $\Sigma_{t,\mathrm{IMU},x}, \Lambda_{t,\mathrm{IMU},x} \in \mathbb{R}^{d_{\mathrm{IMU}}\times nd_x}$, $\Sigma_{t,x,\mathrm{IMU}}, \Lambda_{t,x,\mathrm{IMU}} \in \mathbb{R}^{pd_x\times d_{\mathrm{IMU}}}$, and $\Sigma_{t,x,x}, \Lambda_{t,x,x} \in \mathbb{R}^{nd_x\times nd_x}$. Using the above definitions, we can reorder the residuals in $C_K \in \mathbb{R}$ and $C_M \in \mathbb{R}^{2d_{\mathrm{IMU}}+nd_x}$, and thus redefine them by:

$$\begin{aligned} C_K(\tilde{x}_{t,K}) &= 0 \in \mathbb{R} \\ C_M(\tilde{x}_{t,K}, \tilde{x}_{t,M}) &= \begin{bmatrix} \Lambda_{t,\mathrm{IMU},\mathrm{IMU}}(x_{t,\mathrm{IMU}} - \mu_{t,\mathrm{IMU}}) + \Lambda_{t,\mathrm{IMU},x}(x_{1:n} - \mu_{t,x}) \\ \Sigma_w^{-1/2}(x_{t+1,\mathrm{IMU}} - g_{\mathrm{IMU}}(x_{t,\mathrm{IMU}})) \\ \Lambda_{t,x,\mathrm{IMU}}(x_{t,\mathrm{IMU}} - \mu_{t,\mathrm{IMU}}) + \Lambda_{t,x,x}(x_{1:n} - \mu_{t,x}) \end{bmatrix} \in \mathbb{R}^{2d_{\mathrm{IMU}}+nd_x}, \end{aligned}$$

where $x_{1:n} := (x_1, \cdots, x_n) \in (\mathcal{X}_p)^n$.

Our state variables and cost functions for the Gauss-Newton algorithm submodule are:

$$\begin{aligned} \overline{x_M^\star} &= \tilde{x}_t^\star = \overline{\mu_t} \in \mathcal{X}_{\mathrm{IMU}} \times (\mathcal{X}_p)^n, \\ \overline{x_K^\star} &= g(\tilde{x}_t^\star) = g(\overline{\mu_t}) \in \mathcal{X}_{\mathrm{IMU}} \times (\mathcal{X}_p)^n, \\ C_K(\tilde{x}_{t,K}^\star) &= 0 \in \mathbb{R}, \\ C_M(\tilde{x}_{t,K}^\star, \tilde{x}_{t,M}^\star) &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{2d_{\mathrm{IMU}}+nd_x}, \\ J_K &= \begin{bmatrix} O & \Lambda_{\mathrm{IMU},x} \\ \Sigma_w^{-1/2} & O \\ O & \Lambda_{xx} \end{bmatrix} \in \mathbb{R}^{(2d_{\mathrm{IMU}}+nd_x)\times(d_{\mathrm{IMU}}+nd_x)} \end{aligned}$$

$$J_M = \begin{bmatrix} \Lambda_{\text{IMU,IMU}} \\ -\Sigma_w^{-1/2} G_t \\ \Lambda_{x,\text{IMU}} \end{bmatrix} \in \mathbb{R}^{(2d_{\text{IMU}}+nd_x)\times d_x},$$

where we have defined G_t to be the Jacobian of $g_{\text{IMU}} : \mathcal{X}_{\text{IMU}} \to \mathcal{X}_{\text{IMU}}$ at $\overline{\mu_{t,\text{IMU}}} \in \mathcal{X}_{\text{IMU}}$, i.e.:

$$G_t := \frac{\partial g}{\partial x_{t,\text{IMU}}}\bigg|_{x_{t,\text{IMU}}=\overline{\mu_{t,\text{IMU}}}}$$

Applying the Marginalization update equations, we thus have:

$$\begin{aligned}
\mu_{t+1} &\leftarrow \tilde{x}_{t,K} - \Sigma_{t+1} J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top\big] C_M(\overline{x_K^\star}, \overline{x_M^\star}) \\
&= g(\overline{\mu_t}), \\
\Sigma_{t+1} &\leftarrow \big(J_K^\top \big[I - J_M (J_M^\top J_M)^{-1} J_M^\top\big] J_K\big)^{-1}, \\
&= \big(J_K^\top J_K - J_K^\top J_M (J_M^\top J_M)^{-1} J_M^\top J_K\big)^{-1}, \\
&= \Bigg(\begin{bmatrix} \Sigma_w^{-1} & O \\ O & \Lambda_{x,\text{IMU}}\Lambda_{\text{IMU},x} + \Lambda_{xx}^2 \end{bmatrix} - \begin{bmatrix} -\Sigma_w^{-1} G_t \\ \Lambda_{x,\text{IMU}}\Lambda_{\text{IMU,IMU}} + \Lambda_{xx}\Lambda_{x,\text{IMU}} \end{bmatrix} \cdot \\
&\qquad \cdot (\Lambda_{\text{IMU,IMU}}^2 + \Lambda_{\text{IMU},x}\Lambda_{x,\text{IMU}} + G_t^\top \Sigma_w^{-1} G_t)^{-1} \\
&\qquad \cdot \begin{bmatrix} -G_t^\top \Sigma_w^{-1} & \Lambda_{\text{IMU,IMU}}\Lambda_{\text{IMU},x} + \Lambda_{x,\text{IMU}}\Lambda_{xx} \end{bmatrix} \Bigg)^{-1} \\
&= \left(\begin{bmatrix} \Sigma_w^{-1} & O \\ O & \Omega_{xx} \end{bmatrix} - \begin{bmatrix} -\Sigma_w^{-1} G_t \\ \Omega_{x,\text{IMU}} \end{bmatrix} (\Omega_{\text{IMU,IMU}} + G_t^\top \Sigma_w^{-1} G_t)^{-1} \begin{bmatrix} -G_t^\top \Sigma_w^{-1} & \Omega_{\text{IMU},x} \end{bmatrix} \right)^{-1}
\end{aligned}$$

To show that this is indeed identical to the propagation equation for the covariance matrix in the Extended Kalman Filter algorithm, i.e. Algorithm 6, Line 5, we must show that:

$$\begin{aligned}
&\left(\begin{bmatrix} \Sigma_w^{-1} & O \\ O & \Omega_{xx} \end{bmatrix} - \begin{bmatrix} -\Sigma_w^{-1} G_t \\ \Omega_{x,\text{IMU}} \end{bmatrix} (\Omega_{\text{IMU,IMU}} + G_t^\top \Sigma_w^{-1} G_t)^{-1} \begin{bmatrix} -G_t^\top \Sigma_w^{-1} & \Omega_{\text{IMU},x} \end{bmatrix} \right)^{-1} \\
&= \begin{bmatrix} G_t \overline{\Sigma}_{t,\text{IMU,IMU}} G_t^\top + \Sigma_w & G_t \overline{\Sigma}_{t,\text{IMU},x} \\ \overline{\Sigma}_{t,\text{IMU},x} G_t^\top & \overline{\Sigma}_{t,x,x} \end{bmatrix}
\end{aligned}$$

This follows by brute-force expanding the above block matrix components, and applying Woodbury's Matrix Identity, along with the definitions of $\Sigma_{t,\text{IMU,IMU}}$, $\Lambda_{t,\text{IMU,IMU}}$, $\Sigma_{t,\text{IMU},x}$, $\Lambda_{t,\text{IMU},x}$, $\Sigma_{t,x,\text{IMU}}$, $\Lambda_{t,x,\text{IMU}}$, $\Sigma_{t,x,x}$, and $\Lambda_{t,x,x}$.

$\square$