

Simultaneous Localization and Mapping Through the Lens of Nonlinear Optimization

Amay Saxena



Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2021-248

<http://www2.eecs.berkeley.edu/Pubs/TechRpts/2021/EECS-2021-248.html>

December 1, 2021

Copyright © 2021, by the author(s).
All rights reserved.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission.

Simultaneous Localization and Mapping Through the Lens of Nonlinear Optimization

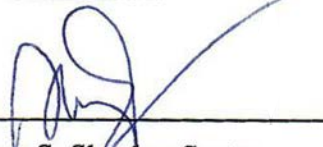
by Amay Saxena

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:



Professor S. Shankar Sastry
Research Advisor

(05/12/2021)



Professor Yi Ma
Second Reader

(05/12/2021)

Abstract

Simultaneous Localization and Mapping Through the Lens of Nonlinear Optimization

by

Amay Saxena

Master of Science in Electrical Engineering and Computer Science

University of California, Berkeley

Professor S. Shankar Sastry, Chair

Simultaneous localization and mapping (SLAM) is the problem of jointly estimating the state (such as pose, velocity, IMU biases etc) of a robot (localization) along with a map of the environment of the robot (mapping). The typical estimators used for landmark-based visual-inertial SLAM fall into one of two categories: batch optimization methods, and filtering methods. This paper analyzes the landmark-based SLAM problem through the lens of nonlinear optimization, and presents a framework that can be used to analyze, implement, and flexibly interpolate between a vast variety of SLAM algorithms. In particular, we demonstrate the equivalence between filtering based algorithms such as the Multi-state constraint Kalman filter (MSCKF) and optimization based algorithms like the sliding window filter. We present a re-interpretation of the MSCKF in terms of nonlinear optimization, and present a novel implementation based on it. We empirically compare the performance of sliding window filters and MSCKF on challenging image sequences, and use the proposed re-interpretation to explain the relative performance characteristics of the two classes of algorithms.

Contents

Contents	i
1 Introduction	1
1.1 Problem Set-up	2
2 Main Algorithm	5
2.1 Nonlinear Least-Squares	5
2.2 SLAM as a Nonlinear Least Squares Problem	6
2.3 A Recipe for SLAM	12
3 Optimization on Manifolds	13
3.1 Box Operators on Manifolds	13
3.2 Manifold Examples	14
3.3 SLAM on Manifolds	16
4 Analyzing SLAM Algorithms	19
4.1 Sliding Window Filter	19
4.2 Kalman Filtering Based SLAM	21
4.3 Equivalence of Filtering and Optimization Approaches	22
4.4 Other State-of-the-Art SLAM Algorithms	29
5 Implementation and Experimental Results	32
5.1 Dataset	32
5.2 Front-end	33
5.3 Dynamics Model	33
5.4 Image Measurement Model	35
5.5 Backend	35
5.6 Results and Discussion	35
5.7 Conclusion	36
Bibliography	39
6 Appendix	42

6.1	Algorithms from Chapter 4	42
6.2	Proofs from Chapter 4	48

Acknowledgments

Much of the research that led to this project was done in collaboration with Chih-Yuan (Frank) Chiu and Dr. Joseph Menke, and will be part of an upcoming publication. I would also like to thank my research advisor Prof. S. Shankar Sastry for his mentorship and guidance, for this project and beyond.

Chapter 1

Introduction

In many robotic applications where a robot is deployed in an unknown environment, it is necessary for the robot to localize itself in its surroundings and also build a map of those surroundings. *Simultaneous Localization and Mapping (SLAM)* is a category of problems that concern a robotic agent in an uncharted environment that must jointly estimate a map of its surroundings in addition to its location within that map. Applications of SLAM include map construction in military applications or search-and-rescue missions [2, 3, 4, 5].

A typical SLAM algorithm consists of the *front end* and *back end*. The front end performs feature extraction, data association, and outlier rejection, to process and interpret raw sensor data. The processed data are then supplied to the SLAM back end, which does inference to produce a state estimate compatible with the data, and with underlying dynamics and measurement models. Back end algorithms can be *filtering* or *batch optimization* based. Filtering methods use processed data to iteratively refine the distribution of recent states [28, 21, 15], and are typically stated in terms of matrix manipulations to the mean vector and covariance matrix of the estimated variables. By contrast, batch optimization iteratively estimates recent states as the solution to an optimization problem whose objective is constructed from odometry and landmark measurement error terms. 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 [2, 14, 9].

The main concrete contributions of this work are as follows:

1. A re-statement the popular Multi-State Constraint Kalman filter (MSCKF) [21] algorithm as an iterative optimization, and a proof of the equivalence thereof. We show how using our interpretation allows us to transparently analyze the properties of the algorithm and suggest straightforward improvements and modifications to it in a way that is conceptually much more difficult when the algorithm is stated in its standard form (in terms of sparse matrix operations).

2. A novel implementation of the MSCKF based on the above result.
3. An empirical evaluation and comparison of the MSCKF and other batch-optimization based sliding window techniques; it is shown how the presented framework allows us to easily explain the comparative features of the two techniques.
4. Experimental validation on real-world data.

Below, Section 1.1 formulates the SLAM problem for robotic agents on Euclidean spaces. Section ?? generalizes this framework to settings where the state is overparameterized, and constrained to evolve on a smooth manifold. Section ?? then presents our main algorithm in three submodules: Gauss-Newton steps, linear approximation, and marginalization. Section 4.3 demonstrates how our framework encompasses existing filtering algorithms, e.g., the Extended Kalman Filter (EKF) and Multi-State Constrained Kalman Filter (MSCKF). Section ?? describes implementation details and provides empirical evaluations of a number of sliding window techniques on real-world data.

1.1 Problem Set-up

Landmark-based SLAM

SLAM is an umbrella term for a large variety of problems and algorithms that generally differ in terms of the sensors that are assumed to be available to the robot and the format in which the map is to be specified. We focus in particular on the paradigm of *landmark-based SLAM*, where the environment is specified as an unordered collection of landmarks. Landmark-based SLAM is the dominant paradigm in situations where the full 6DOF pose of a robot needs to be estimated. It is also the dominant paradigm in the case of *Visual SLAM*, where the primary exteroceptive sensor available to the robot is a camera. The objective then, is to jointly estimate the robot states (at each timestep) along with the states of every landmark that comprises the map of the environment. Note further that we will generally assume that the environment is static, so that the true states of the landmarks do not vary with time.

In landmark-based SLAM, two types of variables must be jointly estimated: robot *states* and *features* or *landmarks*. 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 $\{f_j \mid j = 1, \dots, p\} \subset \mathbb{R}^{d_f}$, can be obtained by analyzing information from image measurements $\{z_{t,j} \mid j = 1, \dots, p\} \subset \mathbb{R}^{d_z}$ and state estimates, where $z_{t,j}$ is the *measurement* of landmark j as seen by the robot at time t (when it is in state x_t). These measurements provide information regarding the relative position of the robot in its environment, and should be used to constrain the robot states and feature states through the measurement of the latter from the former.

States and features are described by a smooth (i.e., infinitely continuously differentiable) dynamics map $g : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x}$ and a smooth measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_z}$, via additive noise models:

$$x_{t+1} = g(x_t) + w_t, \quad w_t \sim \mathcal{N}(0, \Sigma_w), \quad (1.1)$$

$$z_{t,j} = h(x_t, f_j) + v_{t,j}, \quad v_{t,j} \sim \mathcal{N}(0, \Sigma_v), \quad (1.2)$$

where $\Sigma_w \in \mathbb{R}^{d_x \times d_x}$, $\Sigma_w \succeq 0$ and $\Sigma_v \in \mathbb{R}^{d_z \times d_z}$, $\Sigma_v \succeq 0$. Here, g describes the discrete time evolution of the robot state, and represents our model of the robot's dynamics. Although this is not notated for simplicity, the dynamics map g may depend on an odometry measurement or a control input u_t in addition to the state x_t .

Equations 1.1 and 1.2 represent prior knowledge about the evolution of the robot states and the generative process for landmark measurements. Thus, the objective of a SLAM algorithm is to find the assignment to the unknown variables $\{x_t\}$, $\{f_j\}$ that best explains the observed set of landmark measurements $\{z_{t,j}\}$ and odometry measurements/control inputs $\{u_t\}$, subject to the constraints imposed by the dynamics map and measurement model.

SLAM as a Nonlinear Optimization Problem

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 unconstrained minimization of a sum of a collection of weighted cost terms, which represent constraints generated by the dynamics and measurement maps.

Say that at time t , we have detected and measured p landmarks. Let $S_t \subseteq \{0, \dots, t\} \times \{1, \dots, p\}$ be a set of index pairs such that $(i, j) \in S_t$ if and only if landmark j was measured by the robot at time i . The unknown variables to be estimated are the robot states $\{x_0, \dots, x_t\}$ and landmark states $\{f_1, \dots, f_p\}$. Denote the vector of unknown variables $\xi_t = (x_0, \dots, x_t, f_1, \dots, f_p)$. Then we wish to find $\min_{\xi_t} c(\xi_t)$ where

$$c(\xi_t) := \|\xi_t - \mu_0\|_{\Sigma_0^{-1}}^2 + \sum_{i=0}^{t-1} \|x_{i+1} - g(x_i)\|_{\Sigma_w^{-1}}^2 + \sum_{(i,j) \in S_t} \|z_{i,j} - h(x_i, f_j)\|_{\Sigma_v^{-1}}^2 \quad (1.3)$$

where we notate $\|v\|_A^2 := v^\top A v$ for any real vector v to be the Mahalanobis norm of v weighted with a positive semi-definite matrix A . We weigh each cost term by the inverse covariance matrix to best incorporate our prior belief distributions. The cost function above is comprised of the Mahalanobis norms of a number of vectors (such as $(x_{i+1} - g(x_i))$, $(z_{i,j} - h(x_i, f_j))$ etc.). We will refer to each such individual vector as a *residual vector* and the scalar Mahalanobis norm cost term as a *residual term*. The first term in the cost function $\|\xi_t - \mu_0\|_{\Sigma_0^{-1}}^2$ represents a prior belief distribution $\mathcal{N}(\mu_0, \Sigma_0)$ over the entire vector of unknowns ξ_t . An initial prior over the first pose x_0 is often necessary, since usually the

pose of the robot in some global frame is not observable. So the initial prior term lets us constrain the absolute pose of the initial robot state. Subsequently, this prior term may be updated to incorporate new information, as we shall see in later sections.

In subsequent sections, we will introduce techniques used to minimize this cost function. Since the above cost function is highly nonlinear and non-convex, we cannot, in general, find the global minimum. Rather, we will start off with an initial guess for the assignment ξ^* , and then use gradient based methods to improve upon this initial guess until convergence. If the initial guess is good enough (i.e. is within the attraction basin of the global minimum), then we can hope to converge to the global minimum. The standard way to do this iterative minimization is by interpreting the cost function above as a *nonlinear least squares* problem and then using methods such as the *Gauss-Newton method* or Levenberg-Marquardt (which can be thought of as Gauss-Newton using a trust-region approach).

As time evolves and we receive new landmark measurements and odometry measurements/control inputs, we add new residual terms to the cost and the cost function grows. Usually, SLAM is intended for real-time operation, i.e. we wish to produce real-time estimates of the robot's pose and environment map. As such, we also need techniques to manage the size of the optimization problem as time grows. To this end, we will describe *marginalization*, a technique in nonlinear optimization that can be used to eliminate variables from the cost function while maintaining, up to first order, the constraints imposed by the removed variables on the remaining variables. In this way, we can eliminate variables from our optimization problem without losing all the information they contain.

In the next section we describe, in detail, the Gauss-Newton and Marginalization algorithms. One of the central objectives of this work is to analyze the properties of these two algorithms as they are applied to the SLAM problem. We will show that a vast variety of SLAM algorithms in the literature can be described as the iterative application of Gauss-Newton descent and Marginalization, and that viewing them through this light gives us the analytic horsepower to compare and improve upon them. The objective is to provide a transparent framework for the analysis of landmark-based SLAM algorithms through the lens of nonlinear optimization, in such a way that makes it easy to compare different algorithms and interpolate between them or improve upon them.

Chapter 2

Main Algorithm

In this chapter, we describe techniques for solving the least-squares minimization of the SLAM cost-function introduced in the prequel. In particular, we will introduce the Gauss-Newton algorithm and the Marginalization algorithm, which are both widely used techniques for the real-time solution of the present optimization problem. As stated earlier, our cost function is highly nonlinear, and as such the solving strategy is to start off with an initial guess for the unknown variables and then iteratively improve upon this guess using gradient based methods (namely, Gauss-Newton descent), and converge to the nearest local minimum. If the initial guess is good enough, we can have some hope of converging to the global minimum. In this chapter, we will assume that we already have a way of producing such an initial guess. In a real SLAM system, this is the job of the *front-end* [2]. We will present an example of such a system in the sequel, when we describe our implementation and present experimental results on real data.

First, we will describe how to recast the cost function as a *nonlinear least-squares* cost, on which the Gauss-Newton method can be applied. The Gauss-Newton method will then be described. After that, we describe the technique of *marginalization*, which is used to eliminate variables from the cost function, thus keeping the cost function bounded and making real-time operation possible.

2.1 Nonlinear Least-Squares

A *nonlinear least-squares* (NLLS) problem is an optimization problem of the form

$$\min_x \|f(x)\|_2^2$$

$f : \mathbb{R}^m \rightarrow \mathbb{R}^n$ is some smooth vector-valued function which we shall refer to as the *cost vector*. In the special case where $f(x) = Ax + b$ is affine, for some $A \in \mathbb{R}^{n \times m}$, $b \in \mathbb{R}^n$, this reduces to the standard *linear least-squares* problem $\min_x \|Ax + b\|$ whose solution x^* is given by the solution to the so-called *normal equations* $(A^\top A)x^* = -A^\top b$. If A has full column rank, then $A^\top A$ is positive definite and this solution is unique.

In general, since f is free to be highly nonlinear, we cannot find a global optimum of an NLLS problem. However, if we are given a good-enough initial guess x^* , we can hope to improve upon this guess. We can approximate f by a first-order Taylor expansion around x^* (which we therefore call the *linearization point*) as

$$f(x^* + \Delta x) \approx f(x^*) + J\Delta x$$

where

$$J = \left. \frac{\partial f}{\partial x} \right|_{x=x^*}$$

is the Jacobian of f at x^* . We can then approximate the original NLLS problem with the linear least squares problem

$$\min_{\Delta x} \|f(x^*) + J\Delta x\|_2^2$$

which seeks a deviation Δx from x^* that minimizes an approximation to the cost in a neighbourhood of x^* . Assuming the problem is well-conditioned, i.e. that the cost is, in fact, well approximated by the quadratic approximation $\|f(x^*) + J\Delta x\|_2^2$ and the Jacobian J has full column rank, the solution is given by solving the normal equations $(J^\top J)\Delta x = -J^\top f(x^*)$. Then, we can update our guess as $x^* \leftarrow x^* + \Delta x$. Assuming the initial guess was good enough and the problem was well conditioned, this new guess will be better than the original one.

We can then use this new guess as a new linearization point, and repeat the process above. We can continue to improve the guess in this way until some termination criterion is reached, such as a maximum iteration count or an improvement threshold. This process of iteratively linearizing the cost function and solving the linearized least squares problem to improve upon an initial guess is known as the *Gauss-Newton* algorithm, and can be used to tackle any well-conditioned nonlinear optimization problem where a good initial guess is available. It is worth noting that other policies for computing incremental improvements to the initial guess are available, such as standard gradient descent, the Levenberg-Marquardt algorithm [18], or Powell’s dog leg method [24, 17]. We focus on Gauss-Newton because, as we shall see, it has a meaningful interpretation in the case of filtering based SLAM algorithms, and because for well-conditioned problems where a good initial guess is available, Gauss-Newton tends to be faster than other methods [23].

2.2 SLAM as a Nonlinear Least Squares Problem

To formulate SLAM as a nonlinear least-squares problem, we start by noting that we can “whiten” the residual vectors in the cost function to get rid of the Mahalanobis norm and

replace with a standard 2-norm.

$$\begin{aligned} c(\xi_t) &:= \|\xi_t - \mu_0\|_{\Sigma_0^{-1}}^2 + \sum_{i=0}^{t-1} \|x_{i+1} - g(x_i)\|_{\Sigma_w^{-1}}^2 + \sum_{(i,j) \in \mathcal{S}_t} \|z_{i,j} - h(x_i, f_j)\|_{\Sigma_v^{-1}}^2 \\ &= \|\Sigma_0^{-1/2}(\xi_t - \mu_0)\|_2^2 + \sum_{i=0}^{t-1} \|\Sigma_w^{-1/2}(x_{i+1} - g(x_i))\|_2^2 + \sum_{(i,j) \in \mathcal{S}_t} \|\Sigma_v^{-1/2}(z_{i,j} - h(x_i, f_j))\|_2^2 \end{aligned}$$

This can then be easily seen as the 2-norm of a cost vector $C(\xi_t)$ constructed by stacking each of the residual vectors of the form $\Sigma^{-1/2}v$. As a result, $c(\xi_t) = C(\xi_t)^\top C(\xi_t)$, and the SLAM problem is now reduced to the following nonlinear least squares problem:

$$\min_{\xi_t} c(\xi_t) = \min_{\xi_t} C(\xi_t)^\top C(\xi_t) = \min_{\xi_t} \|C(\xi_t)\|_2^2 \quad (2.1)$$

The Gauss-Newton Method for SLAM

Gauss-Newton descent involves solving for the minimization of $c(\xi_t)$ via the Gauss-Newton method applied to the NLLS form of the SLAM cost function described above. The iterative linearization allows us to approximate $c(\xi_t)$ about a given linearization point ξ_t^* by a linear least-squares (hence quadratic) cost term, i.e.,

$$\min_{\xi_t} c(\xi_t) = \min_{\xi_t} \|\xi_t - \mu_t\|_{\Sigma_t^{-1}}^2 + o(\xi_t - \xi_t^*) \quad (2.2)$$

for some $\mu_t \in \mathbb{R}^d$ and symmetric positive definite $\Sigma_t \in \mathbb{R}^{d \times d}$. Then, the Gauss-Newton update gives us the new guess μ_t . 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 2.2.1. (Gauss-Newton Step) *Let $\xi_t^* \in \mathbb{R}^d$ denote a given linearization point, and suppose $J := \frac{\partial C}{\partial \xi_t} \in \mathbb{R}^{d_C \times d}$ has full column rank. Then up to first order in $\xi_t - \xi_t^*$ and some constant terms (which do not affect the minimization), we can write*

$$c(\xi_t) = \|\xi_t - \mu_t\|_{\Sigma_t^{-1}}^2 + o(\xi_t - \xi_t^*),$$

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

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

The Gauss-Newton algorithm then updates our initial guess to μ_t .

Algorithm 1: Gauss-Newton Step.

Data: Objective $C^\top C$, linearization point ξ_t^* .**Result:** Mean μ , covariance Σ after a Gauss-Newton step.

- 1 $J \leftarrow \left. \frac{\partial C}{\partial \xi_t} \right|_{\mu_t}$
 - 2 $\Sigma_t \leftarrow J^\top J$
 - 3 $\mu_t \leftarrow \xi_t^* - (J^\top J)^{-1} J^\top C(\xi_t^*)$
 - 4 **return** μ_t, Σ_t
-

Proof. We have:

$$\begin{aligned}
c(\xi_t) &= C(\xi_t)^\top C(\xi_t) \\
&= [C(\xi_t^*) + J(\xi_t - \xi_t^*)]^\top [C(\xi_t^*) + J(\xi_t - \xi_t^*)] \\
&\quad + o(\xi_t - \xi_t^*) \\
&= (\xi_t - \mu_t)^\top \Sigma_t^{-1} (\xi_t - \mu_t) + c_0(\xi_t^*) + o(\xi_t - \xi_t^*),
\end{aligned}$$

where $c_0(\xi_t^*) \in \mathbb{R}$ denotes a scalar-valued function of ξ_t^* that is independent of the variable ξ_t . This concludes the proof. \square

The above theorem and algorithm 1 describe one iteration of Gauss-Newton descent. As described above, we may choose to repeat the process with the updated guess as the new linearization point.

At each step, in addition to retrieving the new solution update, we can also retrieve the *information matrix* $\Sigma^{-1} = J^\top J$. When we replace the cost function with the quadratic cost from the theorem above, we say we have *linearized* the cost (even though we have in fact quadraticized the cost, but we have linearized the *cost vector*, which is the object we are more interested in anyway). When we believe that we have a good enough guess for our state variable ξ_t at any given timestep, we may choose to replace our nonlinear cost with a linearized cost for some time savings in subsequent Gauss-Newton steps. This puts the whole cost function into the form of a prior over the states ξ_t . This then takes the role of the first prior term in the cost function, onto which residual terms corresponding to new measurements can be added as time advances. Note that choosing to linearize the cost function is an algorithmic design decision. We point it out here, since in subsequent sections when we discuss filtering algorithms, we will see that some sub-modules of such algorithms are best described as the linearization of a certain cost function.

Marginalization

During real operation, we add new residual terms to the cost function each time a new measurement (of a landmark or of odometry) is registered, and as such, the optimization problem as stated can grow unbounded. To make real-time operation possible, we need to be able to discard states from the optimization problem. For instance, we may want to discard old poses (whose estimates we are confident of) or old features (that are no longer visible) from the optimization problem. However, these variables impose constraints on the variables that we wish to keep, and hence simply dropping them will lead to loss of information. So before we can discard them, we need to impose new constraints on the variables we wish to keep that capture the relationships imposed through the removed variables. We do this by modifying the prior term over the remaining variables to incorporate constraints (up to first order) between the remaining variables induced by the removed variables. This process is called *marginalization* and will be described next.

First, we partition the overall state $\xi_t \in \mathbb{R}^d$ into a *marginalized component* $\xi_M \in \mathbb{R}^{d_M}$, to be discarded from ξ_t , and a *non-marginalized component* $\xi_K \in \mathbb{R}^{d_K}$, to be kept ($d = d_K + d_M$.) Then, we partition $c(\xi_t)$ into two cost terms: $c_1(\xi_K)$, which depends only on non-marginalized state components, and $c_2(\xi_K, \xi_M)$ which depends on both marginalized and non-marginalized state components:

$$\begin{aligned} c(\xi_t) &= c(\xi_K, \xi_M) = c_1(\xi_K) + c_2(\xi_K, \xi_M) \\ &= \|C_1(\xi_K)\|_2^2 + \|C_2(\xi_K, \xi_M)\|_2^2. \end{aligned}$$

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

$$\begin{aligned} \min_{\xi_t} c(\xi_t) &= \min_{\xi_K, \xi_M} \left(c_1(\xi_K) + c_2(\xi_K, \xi_M) \right) \\ &= \min_{\xi_K} \left(\|C_1(\xi_K)\|_2^2 + \min_{\xi_M} \|C_2(\xi_K, \xi_M)\|_2^2 \right). \end{aligned}$$

To remove ξ_M , it suffices to approximate the solution to the inner minimization problem by a linear least-squares cost, i.e.:

$$\min_{\xi_M} \|C_2(\xi_K, \xi_M)\|_2^2 \approx \|\xi_K - \bar{\mu}_{t,K}\|_{\bar{\Sigma}_{t,K}^{-1}}^2$$

for some $\bar{\mu}_{t,K} \in \mathbb{R}^{d_K}$ and $\bar{\Sigma}_{t,K} \in \mathbb{R}^{d_K \times d_K}$. Since $\|C_2(\xi_K, \xi_M)\|_2^2$ is in general non-convex, we obtain $\bar{\mu}_{t,K}$ and $\bar{\Sigma}_{t,K}$ by minimizing the first-order Taylor expansion of $\|C_2(\xi_K, \xi_M)\|_2^2$ about some linearization point, instead of minimizing $\|C_2(\xi_K, \xi_M)\|_2^2$ directly. Below, Theorem 2.2.2 details the derivation of $\bar{\mu}_{t,K}$ and $\bar{\Sigma}_{t,K}$.

Theorem 2.2.2 (Marginalization Step). *Let $\xi_t^* \in \mathbb{R}^d$ denote a given linearization point, and suppose $J := \frac{\partial C}{\partial \xi_t} \in \mathbb{R}^{d_C \times d}$ has full column rank. Define $J_K := \frac{\partial C}{\partial \xi_K} \in \mathbb{R}^{d_C \times d_K}$ and $J_M := \frac{\partial C}{\partial \xi_M} \in \mathbb{R}^{d_C \times d_M}$ evaluated at the linearization point $\xi_t^* = (\xi_K^*, \xi_M^*)$. Up to first order, we can write*

$$\min_{\xi_M} c(\xi_K, \xi_M) = \min_{\xi_K} \left(c_1(\xi_K) + \min_{\xi_M} c_2(\xi_K, \xi_M) \right) \quad (2.3)$$

$$= \min_{\xi_K} \left(c_1(\xi_K) + \|\xi_K - \mu_K\|_{\Sigma_K^{-1}}^2 \right) \quad (2.4)$$

where $\Sigma_K \in \mathbb{R}^{d_K \times d_K}$ and $\mu_K \in \mathbb{R}^{d_K}$ are given by:

$$\Sigma_K := \left(J_K^\top [I - J_M (J_M^\top J_M)^{-1} J_M^\top] J_K \right)^{-1}, \quad (2.5)$$

$$\mu_K := \xi_K^* - \Sigma_K J_K^\top [I - J_M (J_M^\top J_M)^{-1} J_M^\top] C_2(\xi_t^*). \quad (2.6)$$

Proof. It suffices to show that:

$$\begin{aligned} & \min_{\xi_M} c_2(\xi_K, \xi_M) \\ &= (\xi_K - \mu_K)^\top \Sigma_K^{-1} (\xi_K - \mu_K) + c'(\xi_t^*). \end{aligned}$$

To do so, we first note that up to first order in deviations $\Delta \xi_t = (\Delta \xi_K, \Delta \xi_M)$ from the linearization point:

$$\begin{aligned} c_2(\xi_t) &= \|C_2(\xi_t)\|_2^2 = \|C_2(\xi_t^*) + J_2 \Delta \xi_t\|_2^2 \\ &= \|C_2(\xi_t^*) + J_K \Delta \xi_K + J_M \Delta \xi_M\|_2^2. \end{aligned}$$

By the method of least-squares, the optimal $\Delta \xi_M$ is given by the normal equation:

$$\Delta \xi_M = -(J_M^\top J_M)^{-1} J_M^\top (C_2(\xi_t^*) + J_K \Delta \xi_K)$$

Substituting back into our expression for $c(\xi_t)$, we have:

$$\begin{aligned}
& \min_{\xi_M} c_2(\xi_t) \\
&= \|(I - J_M(J_M^\top J_M)^{-1} J_M^\top)(C_2(\xi_t^*) + J_K \Delta \xi_K)\|_2^2 \\
&= (C_2(\xi_t^*) + J_K \Delta \xi_K)^\top [I - J_M(J_M^\top J_M)^{-1} J_M^\top] \\
&\quad (C_2(\xi_t^*) + J_K \Delta \xi_K) \\
&= (\xi_K - \xi_K^*)^\top \underbrace{J_K^\top [I - J_M(J_M^\top J_M)^{-1} J_M^\top] J_K}_{:= \Sigma_K^{-1}} \\
&\quad (\xi_K - \xi_K^*) \\
&\quad + 2(\xi_K - \xi_K^*)^\top J_K^\top [I - J_M(J_M^\top J_M)^{-1} J_M^\top] C_2(\xi_t^*) \\
&\quad + C_2(\xi_t^*)^\top [I - J_M(J_M^\top J_M)^{-1} J_M^\top] C_2(\xi_t^*) \\
&= (\xi_K - \xi_K^* + \underbrace{\Sigma_K J_K^\top [I - J_M(J_M^\top J_M)^{-1} J_M^\top] C_2(\xi_t^*)}_{:= -\mu_K})^\top \Sigma_K^{-1} \\
&\quad (\xi_K - \xi_K^* + \underbrace{\Sigma_K J_K^\top [I - J_M(J_M^\top J_M)^{-1} J_M^\top] C_2(\xi_t^*)}_{:= -\mu_K}) \\
&\quad + C_2(\xi_t^*) (I - J_M(J_M^\top J_M)^{-1} J_M^\top) \\
&\quad + c'(\xi_t^*) \\
&= (\xi_K - \mu_K)^\top \Sigma_K^{-1} (\xi_K - \mu_K) + c'(\xi_t^*).
\end{aligned}$$

with Σ_K and μ_K as defined in the theorem statement, and $c'(\xi_t^*) \in \mathbb{R}$ as a term independent of ξ_t . □

The important takeaway from the above is that, up to first order, we were able to replace all terms involving the marginalized variables ξ_M with a new prior $\|\xi_K - \mu_K\|_{\Sigma_K^{-1}}^2$ with the expressions for (μ_K, Σ_K) as given in the theorem. Note that now, we are left simply with this new prior and the original residual terms involving only ξ_K . In this way, the variables in ξ_M have been removed from the cost function, and the cost function has been brought back to its original form (with a prior term and a number of measurement residuals), except in terms of only ξ_K . We can now proceed with this new cost function.

Note that the only non-optimality introduced in this process comes from linearization errors in ξ_M . Indeed, if our cost functions were linear, this marginalization process would be ideal. Hence, so long as we make sure that the variables being marginalized are "mature", i.e. we already have good estimates for them and hence the linearization error will be small, we can minimize the introduced errors.

Algorithm 2: Marginalization

Data: Objective $f = C^\top C$, vector of variables to marginalize ξ_M , linearization point ξ^* .

Result: Mean μ_K and covariance Σ_K for non-marginalized variables ξ_K .

1 $C \leftarrow$ subvector of C containing entries dependent on x_M .

2 $J := \begin{bmatrix} J_K & J_M \end{bmatrix} \leftarrow \begin{bmatrix} \frac{\partial C}{\partial \xi_K} \Big|_{\xi^*} & \frac{\partial C}{\partial \xi_M} \Big|_{\xi^*} \end{bmatrix}$.

3 $\Sigma_K \leftarrow (J_K^\top [I - J_M (J_M^\top J_M)^{-1} J_M^\top] J_K)^{-1}$

4 $\mu_K \leftarrow \xi_K^* - \Sigma_{t,K} J_K^\top [I - J_M (J_M^\top J_M)^{-1} J_M^\top] C(x^*)$

5 **return** μ_K, Σ_K

2.3 A Recipe for SLAM

Now, we have the tools to describe how, at a high-level, we may approach solving the SLAM problem. A SLAM algorithm can be described in terms of a policy for applying one or more of the following at each timestep:

1. **Cost construction:** Given all measurements up to the current timestep, construct a nonlinear least squares cost function.
2. **Gauss-Newton descent:** Given an initial guess for newly initialized variables (such as newly detected landmarks or newly added poses) and the best estimate so far for existing variables, improve upon the estimate by performing one or more iterations of the Gauss-Newton algorithm.
3. **Marginalization:** Marginalize away some subset of variables, thus updating the estimate of the mean and covariance over the other states, taking into account information from the removed states.

While this may seem to be somewhat abstract at the moment, in chapter 4 we will make this intuition concrete by looking at a wide family of state-of-the-art SLAM algorithms and characterizing them in terms of policies for cost-construction, Gauss-Newton descent, and marginalization. In particular, we will be able to analyze algorithms based on nonlinear optimization and those based on filtering through the same lens, thus giving us a language with which to compare and analyze them.

But first, we must talk about a particular complication which arises due to the fact that often our optimization variables do not lie in Euclidean space where vector addition and subtraction are possible and Taylor expansion has the usual meaning. Instead, many types of commonly seen variables (most notably, 6DOF poses) lie on *smooth manifolds* instead. Hence, special care needs to be taken when talking about optimizing over them. Formalizing this is the topic of the next chapter.

Chapter 3

Optimization on Manifolds

In this chapter, we generalize the SLAM problem formulation presented in Section 1.1 to situations where the dynamical states under study are defined on smooth manifolds, rather than on Euclidean spaces. For instance, the SLAM problem in three dimensions requires keeping track of the orientations of rigid bodies on $\text{SO}(3)$, which are usually embedded in a higher-dimensional ambient space, e.g., via unit quaternions or rotation matrices. To address this requirement, we need new operators to perform the required composition and difference operations directly on the smooth manifold constraining the system states, thus ensuring that the constraints imposed by the over-parameterization remain enforced. This implements the most natural way of composing manifold-valued states with small perturbations in the desired manner.

In the next section we will provide specific definitions for operators on smooth manifolds commonly used in the SLAM literature, known as the *boxplus* and *boxminus* operators. These operations will allow us to write down a more general version of the cost function in (1.3) for states defined on smooth manifolds. In addition, to take the derivative of variables with respect to their minimal coordinates, we formalize our definitions of such operators on general smooth manifolds, acknowledging the specific case when the manifold in question is also a *Lie Group*. Then, we will provide some salient examples of manifolds that are evidently relevant to the SLAM problem, specifying *boxplus* and *boxminus* operators for each, before presenting the generalized version of our SLAM formulation.

3.1 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 $\pi_x : U_x \rightarrow V_x$ a smooth co-ordinate 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 $\boxplus : U_x \times V_x \rightarrow U_x$ and $\boxminus : U_x \times U_x \rightarrow V_x$ are defined as follows:

$$x \boxplus \delta = \pi_x^{-1}(\delta) \quad (3.1)$$

$$y \boxminus x = \pi_x(y) \quad (3.2)$$

The SLAM problem on manifolds concerns the minimization of a smooth function $f : \mathcal{M} \rightarrow \mathbb{R}$ 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 $\hat{f}_{x_k} := f \circ \pi_{x_k}^{-1} : U_{x_k} \rightarrow \mathbb{R}$ to be the coordinate representation of f w.r.t the coordinate map π_{x_k} . Since \hat{f}_{x_k} is smooth function between Euclidean spaces, we can compute a direction $\delta \in \mathbb{R}^n$ along which \hat{f}_{x_k} should be updated (for instance, using Gauss-Newton on the coordinate representation of f). The update law is then:

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

In words, the iterative algorithm finds a direction around x_k along which \hat{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 $\pi_0 : \mathcal{M} \rightarrow \mathbb{R}^n$ at the identity element $I \in \mathcal{M}$ needs to be specified. This chart provides a natural choice for any other chart $\pi_x : \mathcal{M} \rightarrow \mathbb{R}^n$, by using the group multiplication operator $\circ : \mathcal{M} \times \mathcal{M} \rightarrow \mathcal{M}$:

$$\pi_x(\delta) = \pi_0(x^{-1} \circ \delta), \quad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n, \quad (3.3)$$

$$\pi_x^{-1}(\delta) = x \circ \pi_0^{-1}(\delta), \quad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n. \quad (3.4)$$

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

$$x \boxplus \delta = x \circ \pi_0^{-1}(\delta), \quad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n, \quad (3.5)$$

$$y \boxminus x = \pi_0(x^{-1} \circ y), \quad \forall x \in \mathcal{M}, \delta \in \mathbb{R}^n. \quad (3.6)$$

defined in terms of the single map π_0 . Lie groups occur commonly in the study of SLAM, so it is useful to have this simplification in hand. In particular, the rotation group $SO(3)$ is an example of a Lie group.

3.2 Manifold Examples

In this section, we exhibit examples of the \boxplus , \boxminus and π operators for certain manifolds that prove useful in describing state spaces for SLAM applications—in particular, Euclidean spaces, the space \mathbb{H}_u of unit quaternions and the rotation space $SO(3)$.

1. Unit Quaternions:

In this paper, we use the JPL convention to describe the Lie Group of unit quaternions \mathbb{H}_u , by expressing each unit quaternion q as $q = (q_u, \vec{q}_v)$ where $q_u \in \mathbb{R}$

is the scalar part of the quaternion and $\vec{q}_v \in \mathbb{R}^3$ is the imaginary or vector part, constrained so that $\|q\| = \sqrt{q_u^2 + \|\vec{q}_v\|^2} = 1$. The Lie group \mathbb{H}_u is equipped with the standard quaternion multiplication \star , and its minimal representation is completely specified by its *Lie Algebra*. Thus, the local projection function $\pi : \mathbb{H}_u \rightarrow \mathbb{R}^3$ can be defined as the respective Log map, and its inverse $\pi^{-1} : \mathbb{R}^3 \rightarrow \mathbb{H}_u$ can be defined as the Exp map. To define the Log map, we first rewrite each unit quaternion $q = (q_u, \vec{q}_v)$ as $q = (\cos(\theta/2), \sin(\theta/2)\vec{\omega})$ for some $\theta \in [0, \pi]$, and $\vec{\omega} \in \mathbb{R}^3$ with $\|\vec{\omega}\| = 1$. In this form, q is interpreted as the quaternion that implements a rotation about the unit axis $\vec{\omega}$ by θ radians counter-clockwise. When q is not the identity, this decomposition is unique; when it is the identity, we have $\theta = 0$, but $\vec{\omega}$ can be arbitrary. The projection $\pi_g : \mathbb{H}_u \rightarrow \mathbb{R}^3$ to local coordinates is then defined as:

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

and the inverse map π_g^{-1} is defined as:

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

Note that the image of π_g is the open ball of radius π , and so the inverse map is only defined there.

Equipped with the Exp and Log maps, we can implement the \boxplus and \boxminus operations using the standard quaternion product $\star : \mathbb{H}_u \times \mathbb{H}_u \rightarrow \mathbb{H}_u$:

$$q_a \boxplus \vec{\omega} = q_a * \text{Exp}(\vec{\omega}) \tag{3.7}$$

$$q_a \boxminus q_b = \text{Log}(q_b^{-1} * q_a) \tag{3.8}$$

2. The Rotation Group $SO(3)$:

The definition of the boxplus and boxminus operators for $SO(3)$ parallels those for quaternions. In particular, we use as our projection operator $\pi : SO(3) \rightarrow \mathbb{R}^3$ the standard Log operator on $SO(3)$, with output restricted to vectors of length less than π . In this case, the relevant increment and difference operators are:

$$R_a \boxplus \vec{\omega} = R_a * \text{Exp}(\vec{\omega}) \tag{3.9}$$

$$R_a \boxminus R_b = \text{Log}(R_b^T R_a) \tag{3.10}$$

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. Fortunately, the product of a finite collection of smooth manifolds is also a smooth manifold, and likewise for Lie groups,

with the group operation defined simply as element-wise multiplication. 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 the boxplus and boxminus operators on M by:

$$(g_1, g_2) \boxplus (\xi_1, \xi_2) = (g_1 \boxplus \xi_1, g_2 \boxplus \xi_2) \quad (3.11)$$

$$(g_1, g_2) \boxminus (h_1, h_2) = (g_1 \boxminus h_1, g_2 \boxminus h_2) \quad (3.12)$$

The projection map π , too, can be defined similarly via element-wise projection.

3.3 SLAM on Manifolds

To formulate SLAM on manifold-valued variables, we must slightly alter our definition of the state variables, feature and image positions, and dynamics and measurement maps. Let \mathcal{X} be a smooth manifold of dimension d_x , embedded in $\mathbb{R}^{d_{x,e}}$, on which camera poses are defined. Similarly, let \mathcal{F} be a smooth manifold of dimension d_f , embedded in an ambient space $\mathbb{R}^{d_{f,e}}$, on which features are defined, and let \mathcal{Z} be the smooth manifold of dimension d_z , embedded in the ambient space $\mathbb{R}^{d_{z,e}}$, on which image measurements are defined. We then have:

$x_t \in \mathcal{X} :$	Camera pose at time t
$g : \mathcal{X} \rightarrow \mathcal{X} :$	Discrete-time dynamics,
$w_t \in \mathbb{R}^{d_x} :$	Discrete-time dynamics (process) noise,
$\Sigma_w \in \mathbb{R}^{d_x \times d_x} :$	Covariance matrix of dynamics (process) noise, with $\Sigma_w \succ 0$.
$f_{t,j} \in \mathcal{F} :$	Feature position j in the world frame, obtained at the camera pose at time t ,
$z_{tj} \in \mathcal{Z} :$	Observation of feature j relative to pose x_t , (Feature measurement),
$h : \mathcal{X} \times \mathcal{F} \rightarrow \mathcal{Z} :$	Measurement map,
$v_{tj} \in \mathbb{R}^{d_z} :$	Measurement noise,
$\Sigma_v \in \mathbb{R}^{d_z \times d_z} :$	Covariance matrix of measurement (process) noise, with $\Sigma_v \succ 0$.

As $x_t \in \mathcal{X}$ only has d_x degrees of freedom, our process noise will be of dimension d_x . Here, we use our definition of the boxplus operator (\boxplus) to enable the perturbation of $g(x_t)$ by our noise w_t . Thus our additive noise dynamics model becomes:

$$x_{t+1} = g(x_t) \boxplus w_t \quad (3.13)$$

where $w_t \sim \mathcal{N}(0, \Sigma_w)$ is the process noise in \mathbb{R}^{d_x} (of the same dimension as the minimal coordinates of x_t). Similarly, as our measurement z_{tj} only has d_z degrees of freedom, our

measurement noise will be of dimension d_z . By incorporating the \boxplus operator, our measurement function is defined as:

$$z_{tj} = h(x_t, f_{t,j}) \boxplus v_{tj} \quad (3.14)$$

where $v_t \sim \mathcal{N}(0, \Sigma_v)$ is the measurement noise in \mathbb{R}^{d_z} (of the same dimension as \mathcal{Z}).

SLAM as an Optimization Problem on Manifolds

Next, we interpret the SLAM problem on manifolds as the optimization of a cost function $c : \mathcal{X}^t \times \mathcal{F}^p \rightarrow \mathbb{R}$, constructed from residual terms of the same dimension of the minimal coordinates of ξ_t , x_t and z_t :

$$c(\xi_t) := \|\xi_t \boxminus \mu_0\|_{\Sigma_0^{-1}}^2 + \sum_{i=0}^{t-1} \|x_{i+1} \boxminus g(x_i)\|_{\Sigma_w^{-1}}^2 + \sum_{(i,j) \in S_t} \|z_{i,j} \boxminus h(x_i, f_j)\|_{\Sigma_v^{-1}}^2 \quad (3.15)$$

In this way we can generalize our previous results with minimal changes to the algorithm. Note that (1.3) is a special case of this function where all variables live in Euclidean space, and \boxplus and \boxminus are just the standard vector $+$ and $-$ operators respectively.

Further note that since the coordinate maps are assumed smooth, we can differentiate through them. In particular, we can define

$$J := \left. \frac{\partial C(\xi^* \boxplus \delta)}{\partial \delta} \right|_{\delta=0}$$

at the linearization point ξ^* , which is just the Jacobian of a standard vector valued function $\delta \mapsto C(\xi^* \boxplus \delta)$. With these operations in hand, we can define the Gauss-Newton and Marginalization algorithms as

1. **Gauss-Newton:** Given linearization point ξ^* , solve the normal equations $(J^\top J)\Delta\xi = J^\top C(\xi^*)$, and then update using

$$\xi^* \leftarrow \xi^* \boxplus \Delta\xi$$

2. **Linear approximation:** Given linearization point ξ^* , a linearized version of the cost function $c(\xi) = \|C(\xi)\|_2^2$ is given by

$$c(\xi) \approx \|(\xi \boxminus \xi^*) - \hat{\mu}\|_{\hat{\Sigma}^{-1}}^2$$

where $\hat{\Sigma} = (J^\top J)^{-1}$, $\hat{\mu} = -(J^\top J)^{-1} J^\top C(\xi^*)$

3. **Marginalization:** Given linearization point $\xi^* = (\xi_K^*, \xi_M^*)$, up to first order, we have

$$\min_{\xi_K} \left(c_1(\xi_K) + \min_{\xi_M} \|C_2(\xi_K, \xi_M)\|_2^2 \right) = \min_{\xi_K} \left(c_1(\xi_K) + \|(\xi_K \boxminus \xi_K^*) - \hat{\xi}_K\|_{\hat{\Sigma}_K^{-1}}^2 \right)$$

for

$$\begin{aligned}\widehat{\Sigma}_K^{-1} &= J_K^\top \left[I - J_M (J_M^\top J_M)^{-1} J_M^\top \right] J_K \\ \widehat{\xi}_K &= -\widehat{\Sigma}_K J_K^\top \left[I - J_M (J_M^\top J_M)^{-1} J_M^\top \right] C_2(\xi_K^*, \xi_M^*)\end{aligned}$$

where J_K and J_M are the Jacobians of $C_2(\xi_K + \Delta\xi_K, \xi_M + \Delta\xi_M)$ with respect to $\Delta\xi_K$ and $\Delta\xi_M$ respectively, evaluated at 0.

Chapter 4

Analyzing SLAM Algorithms

In this section, we utilize the framework developed so far to analyze a wide family of SLAM algorithms, with a focus on visual inertial odometry (although the analytical insights can be extended to algorithms with any sensory modalities). First, we describe the classic sliding window filter, a workhorse of optimization-based SLAM and the motivating application of marginalization to the SLAM problem. Then, we demonstrate how popular *filtering-based* SLAM algorithms can also be equivalently viewed as the iterative minimization of a NLLS cost function for different marginalization policies. The equivalence of the Extended Kalman Filter (EKF SLAM) to a sliding window filter with window size 1 is shown. 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) [21] to iterative NLLS optimization with a specific marginalization scheme. This result allows us to restate the MSCKF algorithm in simple terms by specifying a marginalization scheme, instead of in terms of matrix operations, as Kalman filter based algorithms are usually stated. This allows us to analyze the MSCKF and its variants on equal footing with the sliding-window optimization algorithms, and also hints at a more straight-forward implementation. We will further see that some improvements or modification to the algorithm become trivial to derive and implement when it is viewed through this perspective.

Finally, we list a number of popular visual SLAM algorithms and algorithm paradigms and characterize them in terms of descent and marginalization schemes.

4.1 Sliding Window Filter

The main application of marginalization to SLAM is the sliding window filter [26]. 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.

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 $\xi = (x_0)$, and is $c_0(\xi) = \|x_0 - \mu_0\|_{\Sigma_0^{-1}}^2$. The best estimate ξ_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

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

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 .

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

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

If the best available estimate of the existing optimization variables is

$\xi_n^* = (x_0^*, \dots, x_n^*, f_1^*, \dots, f_p^*)$, 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 $\{j_1, \dots, j_m\}$ be the indices of landmarks (old and new) measured from the current pose.

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

Also 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 ξ_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 $(x_{n-k+2}, \dots, x_{n+1})$. Apply marginalization to remove the variables $\xi_M = S_f \cup \{x_{n-k+1}\}$. Update $s_n \leftarrow S_n \setminus \{(t, j) : j \in S_f, t = 0, \dots, n+1\}$.
5. **Measurement Update:** Update the best estimate of all variables currently in the optimization problem by taking Gauss-Newton steps until convergence. Reset $c_{n+1} \leftarrow c_n$, $\xi_{n+1}^* \leftarrow \xi_n^*$, $S_{n+1} \leftarrow S_n$, 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, and takes Gauss-Newton steps until convergence to update the estimates of all variables after including feature measurement cost terms from the current timestep. 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 an example of 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 [14, 22, 25]. Such approaches have received prominence more recently as computational resources have made online optimization possible.

4.2 Kalman Filtering Based SLAM

The other class of SLAM systems are based on Kalman-type filtering. Here, a mean vector and covariance matrix over the estimated variables is maintained explicitly and recursively updated when new information is made available through matrix operations. Perhaps the most popular algorithmic paradigm for the SLAM problem is the Extended Kalman Filter (which when applied to the SLAM problem is termed EKF-SLAM) [28]. Indeed, 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 [27]. Solutions based on the EKF are favoured due to their relatively cheap computational cost, and remain a popular choice for real-time state estimation.

EKF-SLAM recursively estimates the current pose of the robot in addition to the states of all landmarks detected in the scene. Note that this means that the estimator never revisits estimates of previous robot states; only the estimate of the current robot state is kept around. Mourikis et al. [21] proposed the Multi-State Constrained Kalman Filter (MSCKF), which keeps n robot poses as part of the state vector, and updates its estimates by taking into account the constraints imposed by landmark measurements common to multiple poses at the same time (hence “multi-state constrained”). Empirically, the MSCKF achieves state-of-the-art performance, and remains a popular choice of real-time estimator in modern SLAM systems.

In the next section, we will establish that filtering-based approaches to SLAM can be interpreted as an iterative optimization of the form discussed above with particular marginalization schemes. First, we will show that the standard EKF-SLAM is equivalent to a sliding window filter with window size 1, where we only take one Gauss-Newton step at each iteration (instead of iterating till convergence). Several versions of this observation can be found

in the literature [30, 13, 26]. We present our own proof and formalize the EKF-SLAM algorithm in terms of an iterative optimization with a chosen marginalization scheme. Then, we establish a similar result for the MSCKF, and show that it also corresponds to an iterative optimization with a chosen marginalization scheme. We will then formalize the MSCKF algorithm as an iterative optimization; this is the main result of this section. As we shall see towards the end of this chapter and the sequel, this allows us to put the MSCKF on equal footing with other optimization based approaches such as sliding window filters, and allow us to explain its performance characteristics and compare them directly to other approaches.

4.3 Equivalence of Filtering and Optimization Approaches

Here, we demonstrate the equivalence of filtering and batch optimization based SLAM algorithms, using the Extended Kalman Filter (EKF, in Section 4.3) and Multi-State Constrained Kalman Filter (MSCKF, in Section 19), as examples.

Extended Kalman Filter (EKF)

The EKF SLAM algorithm maintains the EKF full state vector, $\xi_t := (x_t, f_1, \dots, f_p)$, consisting of the most recent robot state, $x_t \in \mathbb{R}^{d_x}$, and the most recent landmark estimates f_1, \dots, f_p of all detected features. We also maintain a covariance matrix P_t . The EKF proceeds by updating the estimate ξ_t^* and covariance matrix P_t based on odometry and landmark measurements. We base the details of the EKF-SLAM algorithm for the purposes of our discussion on [28]. The EKF-SLAM algorithm, in the standard formulation, proceeds by alternating between the following three steps:

1. **Feature augmentation:** When new landmark are detected, the state vector ξ_t needs to be grown to accommodate these new landmarks. This is the *state augmentation* step.
2. **Measurement Update:** When existing landmarks are re-observed, the estimate and covariance matrix are updated by means of an EKF update, using Jacobians of the measurement model h .
3. **State propagation:** When a new odometry measurement is registered, the robot state is propagated forward according to the dynamics map $g(x_t)$. Since the landmarks are assumed static, they are not propagated. The covariance matrix is updated based on Jacobians of the dynamics map g .

See algorithm 5 for the precise operations performed in each of these steps. In this section, we will show how to re-interpret the EKF-SLAM algorithm as the solution to an

iterative optimization problem. First, let us walk through the proposed optimization based algorithm.

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 $\Sigma_0 \in \mathbb{R}^{d_x \times d_x}$. So at time 0, the cost function is simply $\|\tilde{x}_0 - \mu_0\|_{\Sigma_0^{-1}}^2$.

Suppose that at the current time t , the running cost maintained in the optimization formulation of the EKF SLAM algorithm, $c_{EKF,t,0} : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}^{d_x + pd_f}$, is:

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

where $\tilde{x}_t := (x_t, f_1, \dots, f_p) \in \mathbb{R}^{d_x + pd_f}$ 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 $f_{p+1}, \dots, f_{p+p'} \in \mathbb{R}^{d_f}$ to the EKF full state \tilde{x}_t , and updates its mean and covariance. In particular, feature measurements $(z_{t,p+1}, \dots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}$ are incorporated by adding measurement residual terms to the current running cost $c_{EKF,t,0}$, resulting in a new cost $c_{EKF,t,1} : \mathbb{R}^{d_x + (p+p')d_f} \rightarrow \mathbb{R}$:

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

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 $z_{t,1:p} := (z_{t,1}, \dots, z_{t,p}) \in \mathbb{R}^{d_z p}$, of the p features f_1, \dots, f_p included in \tilde{x}_t , are introduced by incorporating associated measurement residuals to the running cost, resulting in a new cost $c_{EKF,t,3} : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}$:

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

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

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

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 $g : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}^{d_x + pd_f}$. 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 $c_{EKF,t,5} : \mathbb{R}^{2d_x + pd_f} \rightarrow \mathbb{R}$:

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

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 $\tilde{x}_{t,K} := (x_{t+1}, f_{t,1}, \dots, f_{t,p}) \in \mathbb{R}^{d_x + pd_f}$ 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 $c_{EKF,t+1,0} : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}$:

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

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 5) [29] [28].

Theorem 4.3.1. *The feature augmentation step of the standard EKF SLAM algorithm (Alg. 6) is equivalent to applying a Gauss-Newton step to $c_{EKF,t,1} : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}$, given by:*

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

Proof. See Appendix (Section 6.2). □

Theorem 4.3.2. *The feature update step of the standard EKF SLAM algorithm (Alg. 7) is equivalent to applying a Gauss-Newton step on $c_{EKF,t,1} : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}$, given by:*

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

Proof. See Appendix (Section 6.2). □

Theorem 4.3.3. *The state propagation step of the standard EKF SLAM algorithm (Alg. 8) is equivalent to applying a Marginalization step to $c_{EKF,t,5} : \mathbb{R}^{2d_x + pd_f} \rightarrow \mathbb{R}$, given by:*

$$\begin{aligned} & c_{EKF,t,5}(\tilde{x}_t, x_{t+1}) \\ &:= \|\tilde{x}_t - \bar{\mu}_t\|_{\Sigma_t^{-1}}^2 + \|x_{t+1} - g(x_t)\|_{\Sigma_w^{-1}}^2. \end{aligned}$$

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

Proof. See Appendix (Section 6.2). □

Remark 4.3.1. *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.*

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 Σ_w, Σ_v , dynamics map g , measurement map h , time horizon T .

Result: Estimates $\hat{x}_t \in \mathbb{R}^{d_x}, \forall t \in \{1, \dots, T\}$.

```

1  $f_0(x) \leftarrow \|x_0 - \mu_0\|_{\Sigma_0^{-1}}^2$ 
2  $p \leftarrow 0$ .
3 for  $t = 0, 1, \dots, T$  do
4    $(z_{t,1}, \dots, z_{t,p}) \leftarrow$  Measurements of existing features.
5    $\text{cost}_t \leftarrow \text{cost}_t + \sum_{k=1}^p \|z_{t,k} - h(x_t, f_k)\|_{\Sigma_v^{-1}}^2$ 
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    $(z_{t,p+1}, \dots, z_{t,p+p'}) \leftarrow$  Measurements of new features.
9    $\text{cost}_t \leftarrow \text{cost}_t + \sum_{k=p+1}^{p+p'} \|z_{t,k} - h(x_t, f_k)\|_{\Sigma_v^{-1}}^2$ 
10   $\bar{\mu}_t \leftarrow (\bar{\mu}_t, \ell(x_t, z_{t,p+1}), \dots, \ell(x_t, z_{t,p+p'})) \in \mathbb{R}^{d_x + (p+p')d_f}$ .
11   $\bar{\mu}_t, \bar{\Sigma}_t, \text{cost}_t \leftarrow$  1 Gauss-Newton step on  $\text{cost}_t$ , about  $\bar{\mu}_t$  (Alg. 1).
12   $p \leftarrow p + p'$ 
13  if  $t < T$  then
14     $\text{cost}_t \leftarrow \text{cost}_t + \|x_{t+1} - g(x_t)\|_{\Sigma_w^{-1}}^2$ 
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
       $(\bar{\mu}_t, g(\bar{\mu}_t))$  (Alg. 2).
16     $\text{cost}_{t+1} \leftarrow \|x_{t+1} - \mu_{t+1}\|_{\Sigma_{t+1}^{-1}}^2$ 
17  end
18 end
19 return  $\hat{x}_0, \dots, \hat{x}_T$ 

```

Multi-State Constrained Kalman Filter

The Multi-State Constrained Kalman Filter [21] is an algorithm based on the Kalman filter that improves upon the classic EKF-SLAM by utilizing constraints imposed by feature measurements across *multiple* robot poses. In particular, the filter maintains a maximum of n robot poses as part of the filter state. In contrast to the classic EKF-SLAM, landmark positions are not included as part of the state vector, which means the state vector always stays bounded and small. Occasionally, some subset of features observed in two or more of the poses in the filter state are selected to perform an EKF update with. When this happens, the location of the chosen features are estimated by multi-view triangulation using the current best estimates of the involved robot poses. Then, the constraints imposed by the feature measurement across all poses it is visible from is used to perform an EKF update.

Features are chosen for the EKF update according to the following policy:

1. Every feature that is no longer visible in the most recent pose is selected for a feature

update.

2. When the number of states exceeds the maximum number of states n , some subset of them must be discarded. The MSCKF selects $\lfloor n/3 \rfloor$ poses evenly spaced in the state vector, starting from the second pose. Then, all features visible from these poses are utilized for an EKF update, before these poses are removed from the state vector.

The exact update equations are outlined in algorithm 9. In this section, we will show that the MSCKF is equivalent to an iterative optimization algorithm that proceeds as follows. Classically, the MSCKF maintains a filter state vector ξ_t that consists of the most recent IMU state (which contains the pose of the robot body frame, velocity of the body frame, and the IMU biases at the latest timestep), along with n camera poses (without any IMU biases or velocities). So we will first present the reductions with this state in mind, but later we will be able to simplify it to a more standard state vector.

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

$$\xi_t := (x_{t,\text{IMU}}, x_1, \dots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n,$$

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 ξ_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 ξ_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 $c_{\text{MSCKF},0} : \mathcal{X}_{\text{IMU}} \rightarrow \mathbb{R}$ of the running cost in our framework:

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

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

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

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 $\xi_t := (x_{t,\text{IMU}}, x_1, \dots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ 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 ξ_t , derived from $x_{n+1}^{\text{IMU}} \in \mathcal{X}^p$, the IMU position estimate in the global frame. Essentially, we append a copy of the camera pose associated with the current IMU pose to the state vector ξ_t . So now, there is an IMU state $x_{t,\text{IMU}}$, along with a new pose x_{n+1} which is just the pose of the camera at time t .

The *feature update step* uses features measurements to update the mean and covariance of ξ_t . In the MSCKF, features are chosen for an EKF update according to the criterion mentioned above. Let S_f denote the set of all features chosen to be used for an EKF update. (Algorithm 9.) These feature constraints are then incorporated into the running cost, resulting in a new cost $c_{MSCKF,t,3} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \rightarrow \mathbb{R}$:

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

where i in the sum above ranges over camera poses in the state vector from which feature f_j is visible, and $z_{i,j} \in \mathbb{R}^{d_z}$ denotes the feature measurement of feature j observed at pose $x_i \in \mathcal{X}_p$. Initial guesses for these features are produced using multi-view triangulation from the camera poses where they are visible. Then, we immediately perform a *marginalization* step to marginalize away all features f_j . This produces an updated mean for ξ_t , denoted $\bar{\mu}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, and an updated covariance for ξ_t , denoted $\bar{\Sigma}_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)}$. As a result, our cost will be updated to $c_{MSCKF,t,4} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \rightarrow \mathbb{R}$:

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

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 $c_{MSCKF,t,5} : \mathbb{R}^{2d_{\text{IMU}}+nd_x} \rightarrow \mathbb{R}$:

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

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 $\xi_K := (x_{t+1, \text{IMU}}, x_1, \dots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and $\xi_M := x_{t, \text{IMU}} \in \mathcal{X}_{\text{IMU}}$, 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, $\tilde{x}_{t+1} := \tilde{x}_{t, K} = (x_{t+1, \text{IMU}}, x_1, \dots, x_n) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$. Accordingly, the running cost maintained in the optimization framework is updated to $c_{MSCKF,t+1,0} : \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \rightarrow \mathbb{R}$:

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

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 9) [21].

Theorem 4.3.4. *The feature update step of the standard MSCKF algorithm (Alg. 11) is equivalent to applying a marginalization step to $c_{MSCKF,t,3} : \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \times \mathbb{R}^{|S_f|d_f} \rightarrow \mathbb{R}$, given by:*

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

where $f_{S_f} \in \mathbb{R}^{|S_f|d_f}$ denotes the stacked vector of all feature positions in S_f (see Algorithm 9).

Proof. See Appendix (Section 6.2). □

Theorem 4.3.5. *The state propagation step of the standard MSCKF SLAM algorithm (Alg. 12) is equivalent to applying a Marginalization step once to $c_{MSCKF,t,5} : \mathbb{R}^{2d_{IMU}+nd_x} \rightarrow \mathbb{R}$, given by:*

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

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

Proof. See Appendix (Section 6.2). □

Remark 4.3.2. *With the above formal results in hand, we are ready to make a few observations that will make an optimization based implementation of the MSCKF a lot more natural:*

1. *The standard statement of the MSCKF maintains a state vector consisting of one IMU state and multiple old camera poses. Due to this invariant, we need to go through the trouble of introducing a new camera pose and then marginalizing the old IMU state at each iteration. Instead, it is no loss of generality for us to instead keep n IMU states in the state vector. This way, no complicated pose augmentation step is needed, and we can simply add the robot state to the optimization problem and constrain it using a dynamics residual as usual.*
2. *Instead of simply dropping the poses to be removed, in the optimization-based approach, we can marginalize them away instead. This is a cheap operation, since each pose is only constrained to the pose before and after it through the dynamics residual terms.*

Finally, we arrive at the following optimization-based formulation of the MSCKF. At each timestep, we must:

1. Receive a new odometry measurement, and add a dynamics residual of the form $\|x_{t+1} - g(x_t)\|_{\Sigma_v^{-1}}^2$ to the optimization problem.

2. Pick features to perform an EKF update with. Let S_f be the set of features that are no longer visible in the most recent pose.
3. Pick poses that need to be dropped. This is only done if the number of poses in the optimization problem has exceeded the maximum allowable number n . If this is the case, let S_x be the set of $\lfloor n/3 \rfloor$ poses spaced out evenly along the vector of optimization states. Add all features visible from these poses into S_f .
4. Initialize estimates for the features in S_f by multi-view triangulation, then add feature measurement terms of the form $\|z - h(x, f)\|_{\Sigma^{-1}}^2$ for each feature in S_f to the cost function.
5. **Feature update:** Update the mean and covariance of the robot state estimate by marginalizing away all features in S_f .
6. **Dropping states:** Remove states in S_x from the optimization problem by marginalizing them away.

Note that now, the MSCKF is seen as a sliding window filter with a different, more sophisticated marginalization scheme. In the classical sliding window filter, features are added to the optimization problem as soon as they are seen, the oldest pose is marginalized away, and features are only marginalized away when they are no longer visible. At each step, the estimate is improved by taking multiple costly nonlinear Gauss-Newton steps. In the MSCKF, the introduction of features to the optimization problem is delayed until they are “matured”, i.e. have been viewed from multiple views and are no longer visible in the most recent pose. Then, features are immediately marginalized. When poses are to be dropped, evenly spaced poses are marginalized away instead of just marginalizing the oldest pose.

Note that in the MSCKF, no nonlinear Gauss-Newton based updates with feature constraints are performed at all, so only first-order effects of feature constraints are used (through the marginalization). In the next chapter, we will see if and how this affects the performance of the filter. It should be noted that there are already two safeguards against this: first, since the initialization of features is delayed until they have matured and have been seen from multiple views, the linearization error (which is the only error involved with marginalization) is minimized. Additionally, by removing robot states in this evenly-spaced manner, we keep around robot states from arbitrarily far in the past. Such poses tend to have a higher baseline with more recent poses, and hence possess better localization information.

4.4 Other State-of-the-Art SLAM Algorithms

In this section, we briefly overview some other common SLAM/structure from motion algorithmic paradigms, and interpret them through the lens of marginalization and update policies.

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 Σ_w, Σ_v , dynamics g_{IMU} , measurement map h , time horizon T , Pose transform ψ (IMU \rightarrow global), $\epsilon > 0$.

Result: Estimates \hat{x}_t for all desired timesteps $t \in \{1, \dots, T\}$.

```

1 costt ← ||x0 ⊖ μ0||Σ02. (Initialize objective function).
2 Sz, Sx, Sz,1, Sz,2 ← φ
3 (n, p) ← (0, 0)
4 for t = 0, ⋯, T do
5   while new pose xn+1 ∈ Xp recorded, new IMU measurement not received do
6     costt ← costt + ε-1||xn+1 ⊖ ψ(x̃t, xn+1IMU)||22.
7     μt, Σt, costt ← 1 Gauss-Newton costt (Alg. 1), about (μt, ψ(μt, xn+1IMU)) with ε → 0.
8     {zn+1,j} ← Feature measurements at xn+1
9     Sz ← Sz ∪ {(xn+1, fj) | fj observed at n + 1}
10    n ← n + 1
11    if n ≥ Nmax - 1 then
12      Sx ← {xi | i mod 3 = 2, and 1 ≤ i ≤ n.}
13      Sz,1 ← {(xi, fj) ∈ Sz | xi ∈ Sx, feature j observed at each pose in Sx}
14    end
15    Sz,2 ← {(xi, fj) ∈ Sz | fj not observed at xn}.
16    costt ← costt + ∑(xi, fj) ∈ Sz,1 ∪ Sz,2 ||zi,j ⊖ h(xi, ft,j)||Σv-1
17    μ̄t, Σ̄t, costt ← 1 Gauss-Newton step on costt, about μt (Alg. 1)
18    x̂t ← μ̄t ∈ XIMU × (Xp)n.
19    Sz ← Sz \ (Sz,1 ∪ {(xi, fj) | xi ∈ Sx})
20    Reindex poses and features in ascending order.
21    (p, n) ← (p - |Sf|, n - |Sx|)
22  end
23  if t < T then
24    costt ← costt + ||xt+1,IMU ⊖ gIMU(xt,IMU)||Σw-12.
25    μt+1, Σt+1, costt ← 1 Marginalization step on costt, about (μ̄t, g(μt,IMU)) (Alg. 2)
26  end
27 end
28 return x̂0, ⋯, x̂T ∈ XIMU × (Xp)n

```

- **Extended Kalman Filter (EKF)** [29] [28] [31] [30] –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** [21] [15] [16]–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** [19] [26] [8]–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)** [14] [20] [22] –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** [29] [11] [12] –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.

Chapter 5

Implementation and Experimental Results

In this section we present empirical results showing the effects of different marginalization and update schemes to the quality of pose tracking in real-world data. We designed a general SLAM backend that modularly implements utilities to keep track of the current cost function, perform Gauss-Newton descent, and marginalize out variables. This backend is then called from different algorithms to implement various marginalization and state-update schemes. We implement a sliding window filter and examine the effect of window size on state estimation accuracy. We also provide a novel implementation of the MSCKF as an iterative optimization algorithm, and compare its performance to the sliding window filters. Due to our approach of analyzing and implementing each algorithm as a particular choice of marginalization scheme, we are able to make concrete statements about the relative characteristics of the two types of algorithms, and hence comment on their relative performance.

5.1 Dataset

All experiments are carried out using the popular public EuRoC MAV dataset [1], which includes stereo image sequences along with inertial measurement unit (IMU) measurements. The stereo images arrive at a rate of 20Hz and the IMU measurements at a rate of 200Hz. The sensor suite is mounted on board a micro aerial vehicle, as shown in figure 5.1. Ground-truth poses recorded using a Vicon motion 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, 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.

5.2 Front-end

Since the focus of this work is the SLAM backend, we standardize the front-end across all experiments, altering only the backend 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. First, brute force matching using Hamming distance on the binary BRISK descriptors is carried out, and then outliers are filtered using 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.

After stereo matching, all keypoint pairs observed in the previous k frames are considered for matching with the features in the current frame. First, brute-force matching based on Hamming distance is carried out in the left and right frames separately. Then, 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 feature, it is considered a newly detected landmark, and is 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.

5.3 Dynamics Model

We have access to odometry measurements in the form of an on-board IMU, which gives us measurements of body-frame angular velocity and linear acceleration. We use the IMU pre-integration scheme detailed in [10]. We wish 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 . Note that for us, one timestep corresponds to a new *image* measurement. Since IMU measurements arrive at a faster rate than image measurements, computing this map requires us to accumulate several IMU measurements together into a relative state measurement, which can then be composed with the state at time x_t to get the predicted state at time x_{t+1} . This is called IMU *pre-integration*, since we must integrate a number of IMU measurements in advance.

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 $(R_t, p_t) \in$

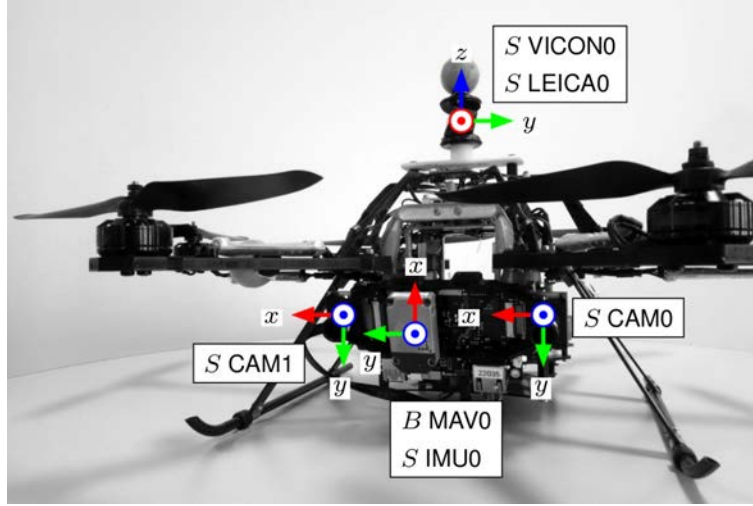


Figure 5.1: The micro aerial vehicle used to collect the EuRoC MAV dataset.

$SE(3)$, $v_t \in \mathbb{R}^3$ 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:

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

where ${}^W\omega_{WB}$ is the true angular velocity, ${}^W a$ the true acceleration, and ${}^W g$ 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 . [10] 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 $\|x_{t+1} \boxminus g(x_t)\|_{\Sigma_v^{-1}}^2$ and Jacobians.

5.4 Image Measurement Model

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, so that 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 $\|z_{tj} - h(x_t, f_j)\|_{\Sigma_w}^2$. 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.

5.5 Backend

For book-keeping the cost function in the backend, computing Jacobians, and implementing Gauss-Newton optimization, we use GTSAM [6, 7]. The whole system is implemented in C++.

5.6 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 (5.2) 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. We can see that most algorithms are able to do well on this sequence. What may be surprising is that the MSCKF, despite its lack of multiple nonlinear Gauss-Newton updates at each iteration, is able to perform about as well the large sliding window filters which are much more computationally intensive. The results for the Vicon room 2 02 dataset (medium) are shown in figure (5.3). Note that on this dataset, sliding window filters of size smaller than 30 fail completely; there are parts of the sequence where the front-end loses tracking, and the filters are unable to recover. Results are shown for the sliding window filter with window size 30 and the MSCKF for window size 10, and we can see that the MSCKF scheme outperforms the larger sliding window filter on this 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 tells us that there is a critical limit at 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, since if tracking is lost for any significant chunk of the duration of the window size, then we can lose track entirely.

Additionally, 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 couple of factors. First of all, the MSCKF employs a marginalization scheme wherein poses that are evenly spaced in the optimization window are dropped. This means that any given point in time, 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. This was also noted in the original MSCKF paper [21]. 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.

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. The absence of costly nonlinear updates through multiple iterations of Gauss-Newton is made up for by the pose marginalization scheme of dropping poses from evenly spaced out spots in the optimization window, and only incorporating features that have matured. The latter technique further ensures that when the feature is initialized this is done through multiple-view triangulation instead of just stereo triangulation, so that the linearization error when the feature is marginalized away is also minimal. 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 it is marginalized away.

5.7 Conclusion

We have presented a framework based on nonlinear optimization for analyzing and implementing SLAM algorithms. The equivalence between popular filtering based techniques such as EKF-SLAM or MSCKF and nonlinear optimization is shown. A novel re-statement of the

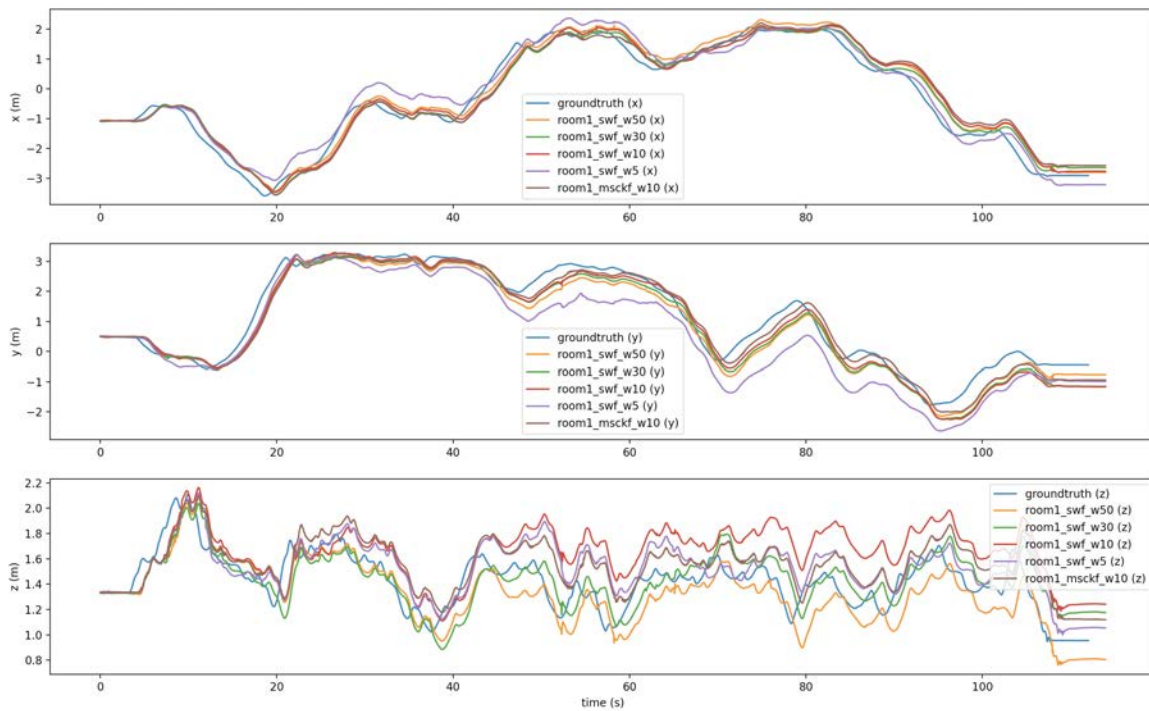


Figure 5.2: Localization results for various sliding window filters and the MSCKF on the Vicon room 2 01 (easy) dataset.

MSCKF as an iterative optimization solver is presented, and is used as the basis for a novel implementation of the algorithm. We evaluate the performance of the sliding window filter, which is explicitly optimization based, and the presented implementation of the MSCKF, and show that the MSCKF with a smaller optimization window 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. Further, we find that the MSCKF is able to better recover from lost tracking than the sliding window filter. By interpreting the MSCKF as an iterative optimization with a special marginalization scheme, the characteristics of this marginalization scheme are used to explain the observed performance characteristics. We believe that the presented approach and analysis can serve 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 the sliding window filter and the MSCKF. We implemented both of those algorithms in the exact same way, with only the marginalization scheme swapped out. As such, by examining the effects of various marginalization schemes in more detail, we can aid the design and implementation of SLAM algorithms.

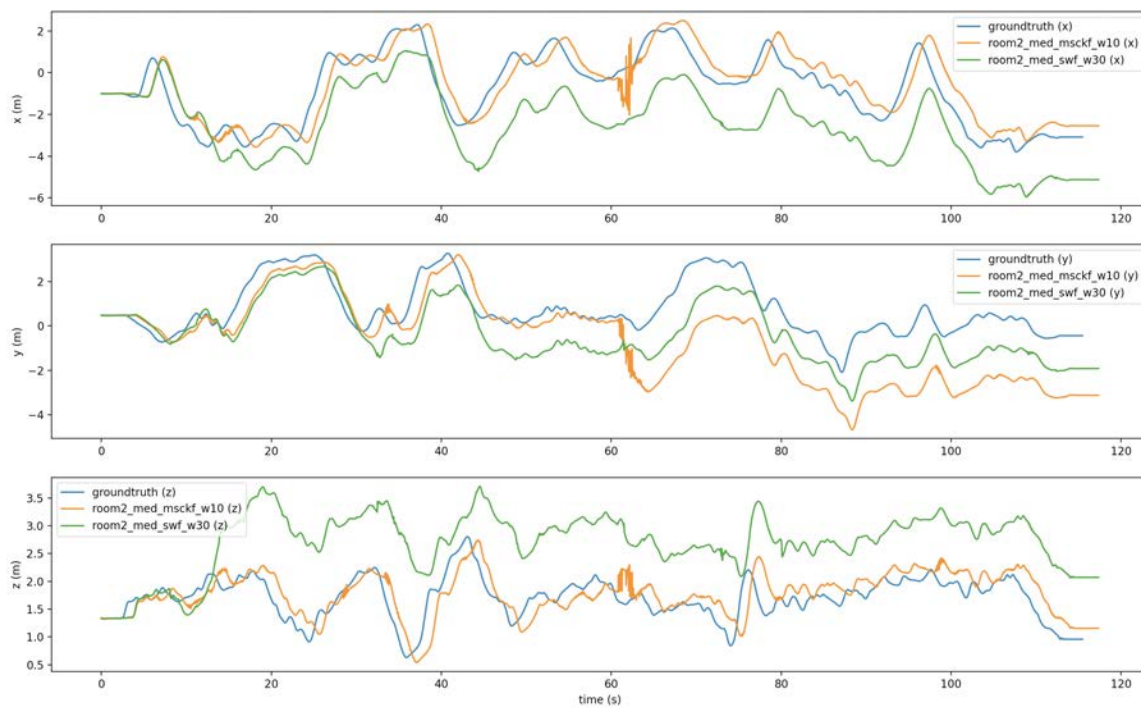


Figure 5.3: Localization results for various sliding window filters and the MSCKF on the Vicon room 2 02 (medium) dataset.

Bibliography

- [1] 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>.
- [2] 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.
- [3] 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.
- [4] Andrew J. Davison. *FutureMapping: The Computational Structure of Spatial AI Systems*. 2018. eprint: 1803.11288 (cs.AI).
- [5] Andrew J. Davison and Joseph Ortiz. *FutureMapping 2: Gaussian Belief Propagation for Spatial AI*. 2019. eprint: 1910.14139 (cs.AI).
- [6] Frank Dellaert et al. “Gtsam”. In: URL: <https://borg.cc.gatech.edu> (2012).
- [7] Frank Dellaert, Michael Kaess, et al. “Factor graphs for robot perception”. In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.
- [8] 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>.
- [9] Kevin Ekenhoff, Patrick Geneva, and Guoquan Huang. “Closed-Form Preintegration Methods for Graph-based Visual-Inertial Navigation”. In: *The International Journal of Robotics Research* 38 (2019), pp. 563–586.
- [10] Christian Forster et al. “IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation”. In: Georgia Institute of Technology. 2015.
- [11] 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.

- [12] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2nd ed. USA: Cambridge University Press, 2003. ISBN: 0521540518.
- [13] Dimitrios G. Kottas and Stergios I. Roumeliotis. “An Iterative Kalman Smoother for Robust 3D Localization and Mapping”. In: *ISRR*. 2015.
- [14] Stefan Leutenegger et al. “Keyframe-based Visual-Inertial Odometry using Nonlinear Optimization”. In: *The International Journal of Robotics Research* 34 (2015), pp. 314–334.
- [15] 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.
- [16] Mingyang Li and Anastasios I. Mourikis. “Optimization-based Estimator Design for Vision-aided Inertial Navigation: Supplemental Materials”. In: *Robotics: Science and Systems* (2012).
- [17] 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.
- [18] 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.
- [19] Peter S. Maybeck et al. *Stochastics Models, Estimation, and Control: Introduction*. 1979.
- [20] 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.
- [21] 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.
- [22] 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.
- [23] Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.
- [24] Michael JD Powell. “A new algorithm for unconstrained optimization”. In: *Nonlinear programming*. Elsevier, 1970, pp. 31–65.

- [25] 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.
- [26] 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>.
- [27] 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.
- [28] Joan Solà. “Simultaneous localization and mapping with the extended Kalman filter”. In: *arXiv* (2014).
- [29] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 0262201623.
- [30] 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.
- [31] 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](https://doi.org/10.1016/S0262-8856(96)01112-2). URL: <http://www.sciencedirect.com/science/article/pii/S0262885696011122>.

Chapter 6

Appendix

6.1 Algorithms from Chapter 4

Algorithm 5: 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 $g : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x}$, measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{p d_f} \rightarrow \mathbb{R}^{d_z}$, time horizon $T \in \mathbb{N}$.

Result: Estimates \hat{x}_t for all desired timesteps $t \leq T$.

```

1 for  $t = 0, \dots, T$  do
2   if detect new feature measurements  $z_{t,p+1:p+p'} := (z_{t,p+1}, \dots, z_{t,p+p'}) \in \mathbb{R}^{p' d_z}$ 
   then
3      $\mu_t, \Sigma_t, p \leftarrow$  Alg. 6, EKF feature augmentation ( $\mu_t, \Sigma_t, p, z_{t,p+1:p+p'}, h(\cdot)$ )
4   end
5    $z_{t,1:p} := (z_{t,1}, \dots, z_{t,p}) \in \mathbb{R}^{p d_z} \leftarrow$  New measurements of existing features.
6    $\bar{\mu}_t, \bar{\Sigma}_t \leftarrow$  Alg. 7, EKF feature update ( $\bar{\mu}_t, \bar{\Sigma}_t, z_{t,1:p}, h(\cdot)$ ).
7   if  $t < T$  then
8      $\mu_{t+1}, \Sigma_{t+1} \leftarrow$  Alg. 8, EKF state propagation ( $\mu_t, \Sigma_t, g(\cdot)$ )
9   end
10 end
11 return  $\hat{x}_0, \dots, \hat{x}_T \in \mathbb{R}^{d_x}$ .
```

Algorithm 6: 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 $z_{t,p+1:p+p'} := (z_{t,p+1}, \dots, z_{t,p+p'}) \in \mathbb{R}^{p'd_z}$, measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_z}$, inverse measurement map $\ell : \mathbb{R}^{d_x} \times \mathbb{R}^{d_z} \rightarrow \mathbb{R}^{d_f}$.

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 $\ell : \mathbb{R}^{d_x} \times \mathbb{R}^{d_z} \rightarrow \mathbb{R}^{d_f} \leftarrow$ Inverse measurement map, satisfying $z_{t,k} = h(x_t, \ell(x_t, z_{t,k}))$ for each $x_t \in \mathbb{R}^{d_x}$, $z_{t,k} \in \mathbb{R}^{d_z}$, $\forall k = p+1, \dots, p+p'$.
- 3 $\tilde{\ell}(\mu_{t,x}, z_{t,p+1}, \dots, z_{t,p+p'}) \leftarrow (\ell(\mu_{t,x}, z_{t,p+1}), \dots, \ell(\mu_{t,x}, z_{t,p+p'})) \in \mathbb{R}^{p'd_f \times (d_x+p'd_z)}$
- 4 $\mu_t \leftarrow (\mu_t, \tilde{\ell}(\mu_{t,x}, z_{t,p+1}, \dots, z_{t,p+p'})) \in \mathbb{R}^{d_x+(p+p')d_f}$
- 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 $\tilde{\Sigma}_v \leftarrow \text{diag}\{\Sigma_v, \dots, \Sigma_v\} \in \mathbb{R}^{p'd_z \times p'd_z}$
- 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** $\mu_t \in \mathbb{R}^{d_x+pd_f}$, $\Sigma_t \in \mathbb{R}^{(d_x+pd_f) \times (d_x+pd_f)}$, $p \geq 0$

Algorithm 7: 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

$z_{t,1:p} := (z_{t,1}, \dots, z_{t,p}) \in \mathbb{R}^{pd_z}$, measurement map $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_z}$

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 $f_{t,1:p} \leftarrow (f_{t,1}, \dots, f_{t,p}) \in \mathbb{R}^{pd_f}$.
 - 2 $\tilde{h}(x_t, f_{t,1:p}) \leftarrow (h(x_t, f_{t,1}), \dots, h(x_t, f_{t,p})) \in \mathbb{R}^{pd_z}$
 - 3 $H_t \leftarrow \left. \frac{\partial \tilde{h}}{\partial (x_t, f_{t,1:p})} \right|_{\mu_t}$ Jacobian of $\tilde{h} : \mathbb{R}^{d_x} \times \mathbb{R}^{pd_f} \rightarrow \mathbb{R}^{pd_z}$ evaluated at $\mu_t \in \mathbb{R}^{d_x+pd_f}$.
 - 4 $\tilde{\Sigma}_v \leftarrow \text{diag}\{\Sigma_v, \dots, \Sigma_v\} \in \mathbb{R}^{pd_z \times pd_z}$.
 - 5 $\bar{\mu}_t \leftarrow \mu_t + \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1} (z_{t,1:p} - \tilde{h}(\mu_t, f_{t,1:p})) \in \mathbb{R}^{d_x+pd_f}$.
 - 6 $\bar{\Sigma}_t \leftarrow \Sigma_t - \Sigma_t H_t^T (H_t \Sigma_t H_t^T + \tilde{\Sigma}_v)^{-1} H_t \Sigma_t \in \mathbb{R}^{(d_x+pd_f) \times (d_x+pd_f)}$.
 - 7 **return** $\bar{\mu}_t \in \mathbb{R}^{d_x+pd_f}$, $\bar{\Sigma}_t \in \mathbb{R}^{(d_x+pd_f) \times (d_x+pd_f)}$.
-

Algorithm 8: Extended Kalman Filter, State Propagation Sub-block.

Data: Current EKF state $\tilde{x}_t \in \mathbb{R}^{d_x+pd_f}$, with mean $\bar{\mu}_t \in \mathbb{R}^{d_x+pd_f}$ and covariance $\bar{\Sigma}_t \in \mathbb{R}^{(d_x+pd_f) \times (d_x+pd_f)}$, (discrete-time) dynamics map $g : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x}$

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 $(\bar{\mu}_{t,x}, \bar{\mu}_{t,f,1:p}) \leftarrow \bar{\mu}_t$, with $\bar{\mu}_{t,x} \in \mathbb{R}^{d_x}$, $\bar{\mu}_{t,f,1:p} \in \mathbb{R}^{pd_f}$.
 - 2 $\begin{bmatrix} \bar{\Sigma}_{t,xx} & \bar{\Sigma}_{t,xf} \\ \bar{\Sigma}_{t,fx} & \bar{\Sigma}_{t,ff} \end{bmatrix} \leftarrow \bar{\Sigma}_t \in \mathbb{R}^{d_x \times d_x}$, with $\bar{\Sigma}_{t,xx} \in \mathbb{R}^{d_x \times d_x}$, $\bar{\Sigma}_{t,xf} = \bar{\Sigma}_{t,fx}^\top \in \mathbb{R}^{d_x \times pd_f}$, $\bar{\Sigma}_{t,ff} \in \mathbb{R}^{pd_f \times pd_f}$.
 - 3 $G_t \leftarrow \left. \frac{\partial g}{\partial x} \right|_{\bar{\mu}_{t,x}}$.
 - 4 $\mu_{t+1} \leftarrow (g(\bar{\mu}_t), \bar{\mu}_{t,f,1:p}) \in \mathbb{R}^{d_x+pd_f}$.
 - 5 $\Sigma_{t+1} \leftarrow \begin{bmatrix} G_t \bar{\Sigma}_{t,xx} G_t^\top + \Sigma_w & G_t \bar{\Sigma}_{t,xf} \\ \bar{\Sigma}_{t,fx} G_t^\top & \bar{\Sigma}_{t,ff} \end{bmatrix} \in \mathbb{R}^{(d_x+pd_f) \times (d_x+pd_f)}$.
 - 6 **return** $\mu_{t+1} \in \mathbb{R}^{d_x+pd_f}$, $\Sigma_{t+1} \in \mathbb{R}^{(d_x+pd_f) \times (d_x+pd_f)}$.
-

Algorithm 9: 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 $h: \mathcal{X}_p \times \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_z}$, time horizon T , pose transformation $\psi: \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n \times \mathcal{X}_p \rightarrow \mathcal{X}_p$ (IMU \rightarrow 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  $t = 0, \dots, T$  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. 10  $(\tilde{x}_t, \mu_t, \Sigma_t, x_{n+1}, x_{n+1}^{\text{IMU}},$ 
      $\psi(\cdot))$ 
6      $\{z_{n+1,j} | \text{feature } j \text{ is observed at } x_{n+1}\} \leftarrow$  Feature measurements at  $x_{n+1}$ 
7      $\{f_j^* | \text{Feature } j \text{ is observed at } x_{n+1}\} \leftarrow$  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      $S_z \leftarrow S_z \cup \{(x_{n+1}, f_j) | \text{Feature } j \text{ observed at } n+1\}$ 
10     $n \leftarrow n+1$ 
11    if  $n \geq N_{\text{max}} - 1$  then
12       $S_x \leftarrow \{x_i | i \bmod 3 = 2, \text{ and } 1 \leq i \leq n.\}$ 
13       $S_{z,1} \leftarrow \{(x_i, f_j) \in S_z | x_i \in S_x, \text{ feature } j \text{ observed at each pose in } S_x\}$ 
14    end
15     $S_{z,2} \leftarrow \{(x_i, f_j) \in S_z | x_i \in x_{1:n}, \text{ feature } j \text{ observed at } x_i \text{ but not at } x_n\}$ .
16     $S_f \leftarrow \{f_j | \exists x_i \in x_{1:n} \text{ s.t. } (x_i, f_j) \in S_{z,1} \cup S_{z,2}\}$ 
17    if  $S_f \neq \phi$  then
18       $\bar{\mu}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n, \bar{\Sigma}_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)} \leftarrow$  Alg. 11  $(\tilde{x}_t, \mu_t, \Sigma_t, x_{n+1},$ 
       $S_{z,1} \cup S_{z,2}, S_f, h(\cdot))$ 
19       $\hat{x}_t \leftarrow \bar{\mu}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ .
20    end
21     $S_z \leftarrow S_z \setminus (S_{z,1} \cup \{(x_i, f_j) | x_i \in S_x\})$ 
22    Reindex poses and features, in ascending order of index, i.e.,  $\{x_1, \dots, x_{n-|S_x|}\}$  and
       $\{f_1, \dots, f_{p-|S_f|}\}$ .
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. 12, MSCKF State
    Propagation  $(\tilde{x}_t, \bar{\mu}_t, \bar{\Sigma}_t)$ 
27  end
28 end
29 return  $\hat{x}_0, \dots, \hat{x}_T \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ 

```

Algorithm 10: 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 $\psi : \mathbb{R}^{(d_{\text{IMU}}+nd_x)} \times \mathcal{X}_p \rightarrow \mathcal{X}_p$

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 $\{f_j^*\}$ 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 μ_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** $\mu_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, $\Sigma_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)}$, $n \geq 0$
-

Algorithm 11: 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 $h : \mathcal{X}_p \times \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_z}$.

Result: Updated MSCKF state mean $\bar{\mu}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$ and covariance $\bar{\Sigma}_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)}$.

- 1 $f_{S_f} \in \mathbb{R}^{|S_f|d_f} \leftarrow$ Concatenation of all features in S_f
- 2 $f_{S_f}^* \in \mathbb{R}^{|S_f|d_f} \leftarrow$ 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 $\tilde{H}_{t,x} \leftarrow \frac{\partial \tilde{h}}{\partial \tilde{x}_t}(\mu_t, f_{S_f}^*) \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z \times (d_{\text{IMU}}+nd_x)}$.
- 6 $\tilde{H}_{t,f} \leftarrow \frac{\partial \tilde{h}}{\partial f_{S_f}}(\mu_t, f_{S_f}^*) \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z \times |S_f|d_f}$.
- 7 $\{a_1, \dots, a_{|S_{z,1} \cup S_{z,2}|d_z - |S_f|d_f}\} \subset \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z} \leftarrow$ Orthonormal basis for $N(\tilde{H}_{t,f}^\top)$.
- 8 $A \leftarrow [a_1 \ \dots \ a_{|S_{z,1} \cup S_{z,2}|d_z - |S_f|d_f}] \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z \times (|S_{z,1} \cup S_{z,2}|d_z - |S_f|d_f)}$.
- 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 $\bar{\Sigma}_t^{-1} \leftarrow \Sigma_t^{-1} + T^\top (Q^\top A^\top R A Q)^{-1} T \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)}$.
- 11 $\bar{\mu}_t \leftarrow \mu_t \boxplus (\Sigma_t^{-1} + T^\top (Q^\top A^\top R A Q)^{-1} T)^{-1} T^\top (Q^\top A^\top R A Q)^{-1} (\tilde{z} \boxminus \tilde{h}(\tilde{x}_t)) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
- 12 $\hat{x}_t \leftarrow \bar{\mu}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
- 13 **return** $\bar{\mu}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, $\bar{\Sigma}_t \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)}$

Algorithm 12: 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 $g : \mathbb{R}^{d_{\text{IMU}}} \rightarrow \mathbb{R}^{d_{\text{IMU}}}$.

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 $(\bar{\mu}_{t,\text{IMU}}, \bar{\mu}_{t,x,1:n}) \leftarrow \bar{\mu}_t$, with $\bar{\mu}_{t,\text{IMU}} \in \mathbb{R}^{d_{\text{IMU}}}$, $\bar{\mu}_{t,x,1:n} \in \mathbb{R}^{nd_x}$.
- 2 $G_t \leftarrow$ Jacobian of $g_{\text{IMU}} : \mathbb{R}^{d_{\text{IMU}}} \rightarrow \mathbb{R}^{d_{\text{IMU}}}$ evaluated at $\bar{\mu}_{t,\text{IMU}} \in \mathbb{R}^{d_{\text{IMU}}}$.
- 3 $\mu_{t+1} \leftarrow (g_{\text{IMU}}(\bar{\mu}_{t,\text{IMU}}), \bar{\mu}_{t,x,1:n}) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$.
- 4 $\Sigma_{t+1} \leftarrow \begin{bmatrix} G_t & O \\ O & I_{nd_x} \end{bmatrix} \bar{\Sigma}_t \begin{bmatrix} G_t^\top & O \\ O & I_{nd_x} \end{bmatrix} + \begin{bmatrix} \Sigma_w & O \\ O & O \end{bmatrix} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)}$.
- 5 **return** $\mu_{t+1} \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n$, $\Sigma_{t+1} \in \mathbb{R}^{(d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)}$.

6.2 Proofs from Chapter 4

Proofs from Section 4.3

Theorem 6.2.1. *The feature augmentation step of the standard EKF SLAM algorithm (Alg. 6) is equivalent to applying a Gauss-Newton step to $c_{EKF,t,1} : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}$, given by:*

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

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* $\ell : \mathbb{R}^{d_x} \times \mathbb{R}^{d_z} \rightarrow \mathbb{R}^{d_f}$, 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 $h : \mathbb{R}^{d_x} \times \mathbb{R}^{d_f} \rightarrow \mathbb{R}^{d_z}$ [28]. When full observations are unattainable, the missing degrees of freedom are introduced as a prior to the system [28]; in this case, similar results follow.

First, to simplify notation, define:

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

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

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

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:

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

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

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

where $\tilde{H}_t := [\tilde{H}_{t,x} \ \tilde{H}_{t,f}] \in \mathbb{R}^{p'd_z \times (d_x+p'd_f)}$ is defined as the Jacobian of $\tilde{h} : \mathbb{R}^{d_x} \times \mathbb{R}^{p'd_f} \rightarrow \mathbb{R}^{p'd_z}$ at $(x_t^*, f_{t,p+1:p+p'}^*) \in \mathbb{R}^{d_x+p'd_f}$, 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:

$$\begin{aligned}
&\Sigma_t \\
\leftarrow (J^\top J)^{-1} & \tag{6.1}
\end{aligned}$$

$$\begin{aligned}
&= \left(\begin{bmatrix} \Sigma_t^{-1/2} & - \begin{bmatrix} I \\ O \end{bmatrix} \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1/2} \\ O & -\tilde{\Sigma}_v^{-1/2} \tilde{H}_{t,f} \end{bmatrix} \right. \\
&\quad \left. \begin{bmatrix} \Sigma_t^{-1/2} & O \\ -\tilde{\Sigma}_v^{-1/2} \tilde{H}_{t,x} [I \ O] & -\tilde{\Sigma}_v^{-1/2} \tilde{H}_{t,f} \end{bmatrix} \right)^{-1} & \tag{6.2}
\end{aligned}$$

$$\begin{aligned}
&= \begin{bmatrix} \Sigma_t^{-1} + \begin{bmatrix} I \\ O \end{bmatrix} \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1} \tilde{H}_{t,x} [I \ O] & \begin{bmatrix} I \\ O \end{bmatrix} \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1/2} \tilde{H}_{t,f} \\ \tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1} \tilde{H}_{t,x} [I \ O] & \tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1} \tilde{H}_{t,f} \end{bmatrix}^{-1} \\
&= \begin{bmatrix} \Omega_{t,xx} + \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1} \tilde{H}_{t,x} & \Omega_{t,xf} & \tilde{H}_{t,x}^\top \tilde{\Sigma}_v^{-1} \tilde{H}_{t,f} \\ \Omega_{t,fx} & \Omega_{t,ff} & O \\ \tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1} \tilde{H}_{t,x} & O & \tilde{H}_{t,f}^\top \tilde{\Sigma}_v^{-1} \tilde{H}_{t,f} \end{bmatrix}^{-1}, & \tag{6.3}
\end{aligned}$$

$$\begin{aligned}
\bar{\mu}_t &\leftarrow \tilde{x}_t^* - (J^\top J)^{-1} J^\top C_1(\tilde{x}_t^*) \\
&= (\bar{\mu}_t, \ell(x_t^*, z_{t,p+1}), \dots, \ell(x_t^*, z_{t,p+p'})).
\end{aligned}$$

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:

$$\begin{bmatrix} \Omega_{t,xx} & \Omega_{t,xf} \\ \Omega_{t,fx} & \Omega_{t,ff} \end{bmatrix} := \begin{bmatrix} \Sigma_{t,xx} & \Sigma_{t,xf} \\ \Sigma_{t,fx} & \Sigma_{t,ff} \end{bmatrix}^{-1} \tag{6.4}$$

To conclude the proof, we must show that (6.3) 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:

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

equals the $(d_x + (p + p')d_f) \times (d_x + (p + p')d_f)$ identity matrix. This follows by applying (6.4), as well as the matrix equalities resulting from taking the derivative of the equation $z_t := h(x_t, \ell(x_t, z_t))$ 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 6.2.2. *The feature update step of the standard EKF SLAM algorithm (Alg. 7) is equivalent to applying a Gauss-Newton step on $c_{EKF,t,1} : \mathbb{R}^{d_x + pd_f} \rightarrow \mathbb{R}$, given by:*

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

Proof. First, to simplify notation, define:

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

We can then rewrite the cost as:

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

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:

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

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

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

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. \square

Theorem 6.2.3. *The state propagation step of the standard EKF SLAM algorithm (Alg. 8) is equivalent to applying a Marginalization step to $c_{EKF,t,5} : \mathbb{R}^{2d_x+pd_f} \rightarrow \mathbb{R}$, given by:*

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

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:

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

where we have identified the following parameters, in the language of a Marginalization step (Section 1.1):

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

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:

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

where we have defined G_t to be the Jacobian of $g : \mathbb{R}^{d_x} \rightarrow \mathbb{R}^{d_x}$ 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:

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

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

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

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 $\Lambda_{t,ff}$. \square

Proofs from Section 19

Theorem 6.2.4. *The feature update step of the standard MSCKF algorithm (Alg. 11) is equivalent to applying a marginalization step to $c_{MSCKF,t,3} : \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \times \mathbb{R}^{|S_f|d_f} \rightarrow \mathbb{R}$, given by:*

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

where $f_{S_f} \in \mathbb{R}^{|S_f|d_f}$ denotes the stacked vector of all feature positions in S_f (see Algorithm 9).

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

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

where $\tilde{z} \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z}$, $\tilde{h} : \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n \times \mathbb{R}^{|S_f|d_f} \rightarrow \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z}$: 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 $\tilde{\Sigma}_v := \text{diag}\{\Sigma_v, \dots, \Sigma_v\} \in \mathbb{R}^{|S_{z,1} \cup S_{z,2}|d_z \times |S_{z,1} \cup S_{z,2}|d_z}$.

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:

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

The Marginalization algorithm block then implies that:

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

where we have defined:

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

Recall that the marginalization equations (2.6) and (2.5) in our formulation read:

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

Substituting in the above expressions for J_K , J_M , and $C_M(\bar{\mu}_t, f_{S_f}^*)$, we have:

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

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

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

we find that it suffices to show:

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

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

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

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

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

which in turn implies that:

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

as claimed. □

Theorem 6.2.5. *The state propagation step of the standard MSCKF SLAM algorithm (Alg. 12) is equivalent to applying a Marginalization step once to $c_{MSCKF,t,5} : \mathbb{R}^{2d_{IMU} + nd_x} \rightarrow \mathbb{R}$, given by:*

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

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

$$\begin{aligned} \tilde{x}_{t,K} & := (x_{t+1,IMU}, x_1, \dots, x_n) \in \mathcal{X}_{IMU} \times (\mathcal{X}_p)^n, \\ \tilde{x}_{t,M} & := x_{t,IMU} \in \mathcal{X}_{IMU}. \end{aligned}$$

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

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

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

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

with the dimensions of the above block matrices given by $\Sigma_{t,\text{IMU},\text{IMU}}, \Lambda_{t,\text{IMU},\text{IMU}} \in \mathbb{R}^{d_{\text{IMU}} \times d_{\text{IMU}}}$, $\Sigma_{t,\text{IMU},x}, \Lambda_{t,\text{IMU},x} \in \mathbb{R}^{d_{\text{IMU}} \times nd_x}$, $\Sigma_{t,x,\text{IMU}}, \Lambda_{t,x,\text{IMU}} \in \mathbb{R}^{nd_x \times d_{\text{IMU}}}$, 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 $C_M \in \mathbb{R}^{2d_{\text{IMU}}+nd_x}$, and thus redefine them by:

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

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

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

$$\begin{aligned} \bar{x}_M^* &= \tilde{x}_t^* = \bar{\mu}_t \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n, \\ \bar{x}_K^* &= g(\tilde{x}_t^*) = g(\bar{\mu}_t) \in \mathcal{X}_{\text{IMU}} \times (\mathcal{X}_p)^n, \\ C_K(\tilde{x}_{t,K}^*) &= 0 \in \mathbb{R}, \\ C_M(\tilde{x}_{t,K}^*, \tilde{x}_{t,M}^*) &= \begin{bmatrix} 0 \\ 0 \end{bmatrix} \in \mathbb{R}^{2d_{\text{IMU}}+nd_x}, \\ J_K &= \begin{bmatrix} O & \Lambda_{\text{IMU},x} \\ \Sigma_w^{-1/2} & O \\ O & \Lambda_{xx} \end{bmatrix} \\ &\in \mathbb{R}^{(2d_{\text{IMU}}+nd_x) \times (d_{\text{IMU}}+nd_x)} \\ J_M &= \begin{bmatrix} \Lambda_{\text{IMU},\text{IMU}} \\ -\Sigma_w^{-1/2} G_t \\ \Lambda_{x,\text{IMU}} \end{bmatrix} \in \mathbb{R}^{(2d_{\text{IMU}}+nd_x) \times d_x}, \end{aligned}$$

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

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

Applying the Marginalization update equations, we thus have:

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

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

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

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

□