On the Efficiency and Consistency of Visual-Inertial Localization and Mapping

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Kejian Wu

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Doctor of Philosophy

Stergios I. Roumeliotis, Advisor

February, 2024

Acknowledgements

This dissertation would not have been possible without the help and support from numerous people.

First, I would like to express my gratitude and appreciation to my advisor, Professor Stergios Roumeliotis, for the countless effort he put to pass me his knowledge, for the weekends he spent with me discussing research problems through reading groups, and for his continuous encouragement to make me achieve more. In addition to the guidance in academic matters, he has also taught me a serious research attitude by setting examples: Aim at the best and never give up. Without his endless mentoring, I cannot become the researcher I am now. I am also thankful for the time and valuable advice from the members of my committee, Professor Yousef Saad, Professor Andrew Lamperski, and Professor Mingyi Hong.

I would like to thank all my amazing colleagues and friends at the MARS lab. Thanks to the senior lab members, Faraz, Paul, Esha, Sam, Ke, and Joel, your priceless mentoring helped me build up fundamental knowledges in robotics. Thanks to my peers, Chao, Dimitrios, Ruipeng, Igor, and Luis. It's you who walked me through the most difficult times. I will never forget the nights we spent together trying to catch up deadlines. Lastly, thanks to the junior lab members, Kourosh, Ryan, Georgios, Mrinal, Elliot, Ahmed, Katherine, Tien, and Tong. I cannot accomplish this dissertation without the countless hours you spent with me on coding and writing papers.

Most importantly, I would like to thank my family. My parents have always done everything to support me so that I could follow my dreams, and have never failed to believe in me and care about me with love. Many thanks to Jia Li, for her love and companionship during all the happy and stressful times that we went through together. And finally, my deep thoughts to Ran Xu, I will always miss you and carry our dreams and future with me throughout my life.

Last but not least, I gratefully acknowledge financial support from the National Science

Foundation, the University of Minnesota Department of Computer Science and Engineering, the University of Minnesota Digital Technology Center, and Google Project Tango and Daydream.

Abstract

Simultaneous localization and mapping (SLAM) is a prerequisite for a wide range of applications, such as robot navigation in GPS-denied areas, autonomous driving, and augmented or virtual reality. As inertial measurement unit (IMU) and camera are becoming ubiquitous, visual-inertial SLAM (VI-SLAM) systems have prevailed in these applications, in part because of the complementary sensing capabilities and the decreasing costs and size of the sensors. Although successful VI-SLAM systems have been developed over the past decade, there still exist many challenges that limit the performance of such systems, especially when deployed on resource-constrained mobile devices (e.g., smart phones, tablets, and wearable computers). In this dissertation, we seek to address three key challenges for improving the efficiency, accuracy, and consistency of VI-SLAM.

The first part of this dissertation considers the problem of short-term VI-SLAM, aka visual-inertial odometry (VIO), where the system focuses its optimization over only a bounded-size sliding window of recent states (poses and features), for achieving constant processing time. While high computational efficiency is of critical importance for such systems, one of the main limitations of existing VIO algorithms is the requirement of using double-precision arithmetic for implementation, due to the ill-conditioning of the VIO problem. To address this issue, we present a square-root inverse sliding-window filter for highly efficient and accurate VIO. By maintaining and updating the upper-triangular Cholesky factor of the Hessian matrix, our estimator can yield the same effective precision of regular filters while using only half of the wordlength, thus enabling single-precision implementation. This leads to significant speedups as compared to double-precision alternatives, especially on mobile devices with co-processors that provide a 4-fold processing speed acceleration for 32-bit floating-point operations.

In the second part of this dissertation, we study the case when VI-SLAM systems are deployed on mobile platforms that have restricted motions (e.g., ground robots or self-driving cars). In such cases, we observe that the localization errors of VI-SLAM systems are significantly larger than those on the platforms moving freely in the 3D space. We investigate this issue and discover that the restricted motion that ground robots often undergo (e.g., constant speed or acceleration, or no rotation) alters the observability properties of VI-SLAM and renders additional unobservable directions (e.g., the scale, or roll and pitch angles). As a result,

little or no information can be obtained along these directions in the estimates, which will degrade the localization accuracy of the employed VI-SLAM estimator. To address this limitation, we extend the VI-SLAM system to incorporate extra information, from wheel-encoder data and planar-motion constraints, which leads to significant improvements in positioning accuracy for wheeled robots moving primarily on a plane.

Lastly, we address the long-term VI-SLAM problem. In such systems, in addition to the local optimization of the short-term VI-SLAM, global adjustment of past states is performed using loop-closure measurements (reobservations to previously-mapped features), so as to reduce global drifts in the estimates for long-term accuracy. In order to achieve real-time operation, however, existing approaches often assume previously-estimated states to be perfectly known (e.g., previous keyframes or maps), which leads to inconsistent estimates. This means that the estimated covariance is unduly small and does not represent correctly the uncertainty of the current state estimates, and combining these overly optimistic estimates with new measurements later on will further degrade the accuracy of the system. Instead, based on the idea of the Schmidt-Kalman filter, we derive a new consistent approximate method in the information domain, which has linear memory requirements and adjustable (constant to linear) processing cost. By employing this method with different configurations, we realize an efficient and accurate long-term VI-SLAM system, the RISE-SLAM, which improves estimation consistency.

Contents

List of Tables

List of Figures

Chapter 1

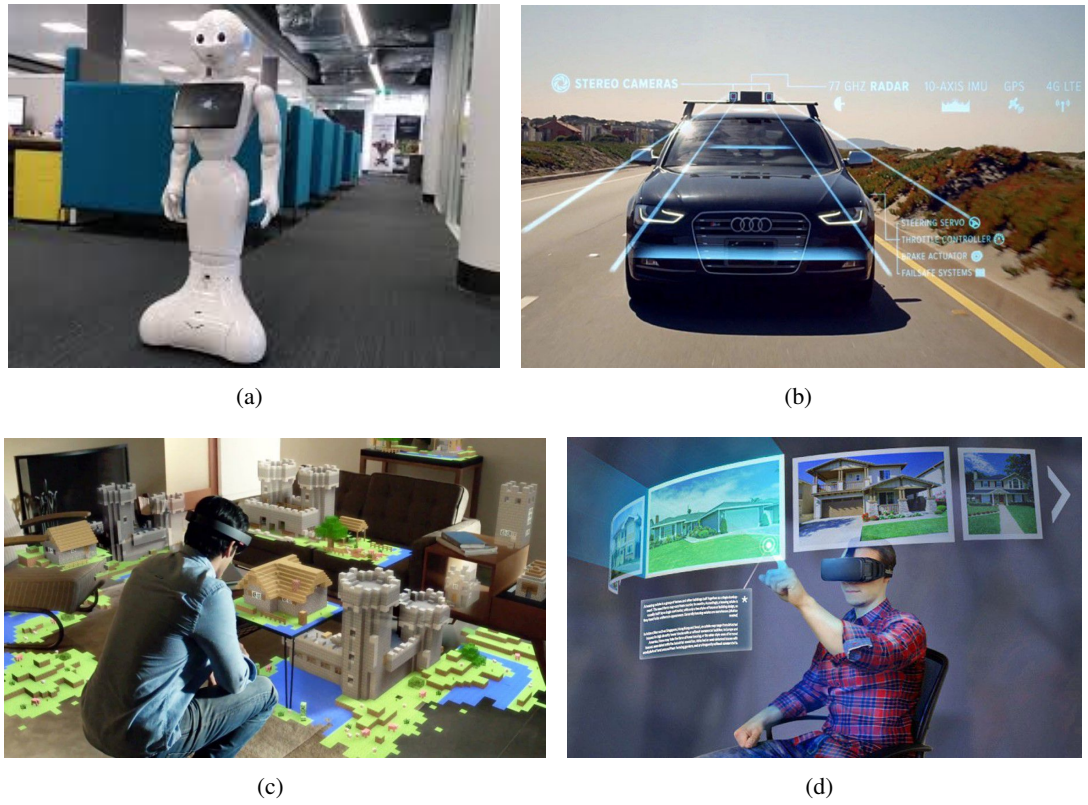
Introduction

1.1 Visual-Inertial Localization and Mapping

Localization, i.e., determining the position and orientation (pose) of a mobile platform in the three-dimensional (3D) space, is a fundamental technology for a wide range of applications (see Fig. 1.1), such as robot navigation [20, 21, 31, 81, 85], autonomous driving [3, 5, 17], and augmented/virtual reality (AR/VR) [1, 2, 4, 6]. As an example, mobile robots must be able to track their locations with respect to the environment, in order to perform path planning, obstacle avoidance, and other high-level tasks. Similarly, in AR applications, the current pose of the device is a crucial information for correctly rendering virtual contents onto surrounding scenes of the real world, so that a virtual object would appear static and hence more realistic when observed by humans.

Although GPS-based localization systems can directly provide positioning information, there exist many environments that preclude their usage, such as indoors, urban canyons, underground, underwater, outer space, etc. Moreover, the GPS accuracy, often in the order of meters, may be insufficient for high-precision applications, such as self-driving cars navigating in traffic. Furthermore, since GPS does not directly provide orientation information, sole reliance on it may lead a robot to become disoriented, especially during stationary intervals.

Instead of relying on GPS, a more flexible approach is to utilize onboard sensors to infer the location of the mobile platform. Sensors that are popular for localization purposes can be classified into two categories: proprioceptive sensors (e.g., inertial measurement units (IMUs) and odometers) and exteroceptive sensors (e.g., cameras, laser scanners, and sonars). In this

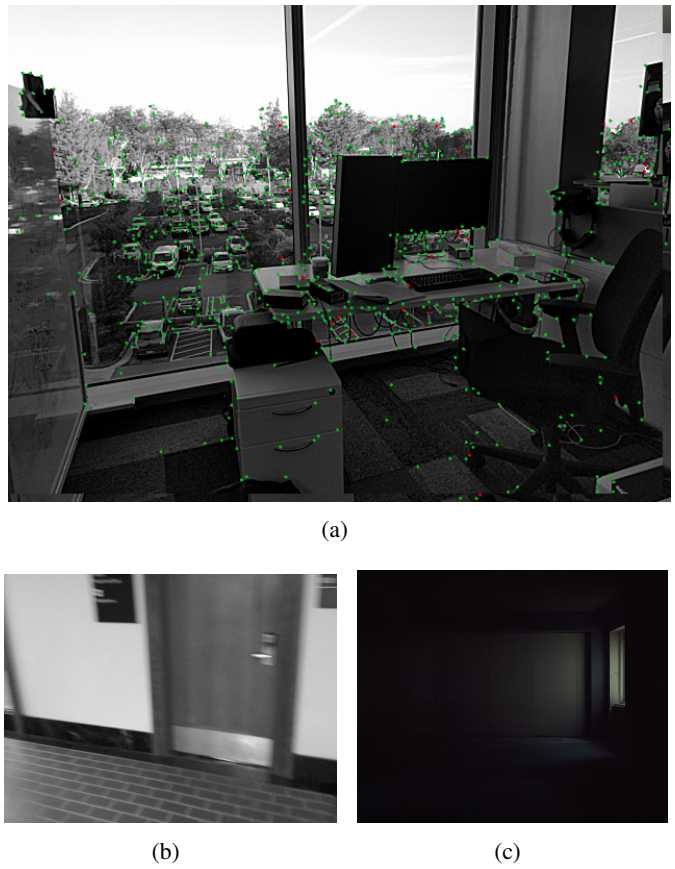


(a) (b)

(c) (d)

Figure 1.1: Applications of localization: (a) Indoor robot navigation. (b) Autonomous driving. (c) Augmented reality. (d) Virtual reality.

dissertation, we focus on two low-cost and lightweight sensors, which are readily available on most mobile devices (e.g., smart phones, tablets, and wearable computers): An IMU and a camera. An IMU measures the linear acceleration and rotational velocity of the platform to which it is rigidly connected. By integrating these measurements, one could track its 3D pose at a high frequency due to the low processing cost. Unfortunately, simple integration of high-rate IMU measurements that are corrupted by noise and bias, often results in fast error accumulation and hence unreliable pose estimates for long-term navigation. Although high-end tactical-grade IMUs exist, it remains prohibitively expensive for widespread deployments. On the other hand, images from a camera provide rich information about the environment. From these camera images, by extracting and tracking features of interest (e.g., points, see Fig. 1.2) that belong to the static scene, one could infer the 3D motions of the platform. In addition, by



(a)

(b) (c)

Figure 1.2: Images from a camera: (a) Point features (green dots) extracted and tracked. (b) A blurry image due to fast motion. (c) A dark image due to insufficient lighting.

performing mapping, a 3D map of the surrounding scene can be created in parallel, and utilized to bound the localization error when the camera returns to a previously-mapped area (i.e., loop-closures). Pure vision-based navigation systems, however, suffer from robustness issues under adverse operating conditions, such as image blur due to fast motions, insufficient lighting, and lack of scene texture (see Fig. 1.2). Due to these complementary sensing capabilities of an IMU and a camera, as well as their decreasing cost and size, over the past decade, it has become a trending technology to perform localization and mapping, by combining and fusing data from these two sensors, in the so-called visual-inertial navigation systems (VINS) [45]. Moreover, commercial-grade mobile devices, such as smart phones, have recently been recognized as promising platforms for realizing VINS, because of their sensing capabilities and wide-spread

availability.

In this dissertation, we address the problem of visual-inertial simultaneous localization and mapping (VI-SLAM), and develop algorithms that provide accurate and efficient solutions to it, so as to enable real-time perception and navigation applications on mobile devices with limited processing resources.

1.2 Challenges of Visual-Inertial Localization and Mapping on Mobile Devices

VI-SLAM can be considered as an instance of the general problem of SLAM [19] (using particular visual and inertial sensors), which arises when a platform navigates within unknown areas, i.e., it does not have access to a prior map of its environment. In these scenarios, in order to determine the platform's location with respect to its surroundings, a SLAM system is used to build a map of the environment online while simultaneously localizing the sensor platform within the same map. Therefore, the VI-SLAM problem is essentially a state estimation problem, where the state typically consists of sensor poses and map feature positions (as well as other states such as calibration parameters, sensor biases, velocity, etc), given the visual-inertial measurements from the IMU and the camera. It is well known that under the assumption of additive white Gaussian noises, finding the optimal Maximum a Posteriori (MAP) estimate for VI-SLAM can be cast as a nonlinear batch least-squares (BLS) problem, whose solution can be obtained in either a batch [24, 87] or an incremental [51, 52] form. As time goes by and the number of sensor poses and map features increases, however, these optimal approaches have an increasing processing cost with time, typically between linear and quadratic in the total number of states, and thus cannot provide high-frequency estimates when operating inside large areas.

In order to obtain efficient solutions, existing approaches in the VI-SLAM literature relax the optimal BLS formulation into two sub-problems: The **short-term** and **long-term** VI-SLAM. The short-term VI-SLAM, aka visual-inertial odometry (VIO) [42, 58, 69], focuses its optimization over only a bounded-size sliding window of recent poses (as well as a local map), by marginalizing out past states and measurements. The latency of such VIO methods is typically very low and does not increase with time, and hence is used for tracking the current pose of the platform in real time. This local optimization scheme, however, results in an ever-increasing

drift in the pose estimates, due to its inability to process loop-closure measurements and perform global adjustment of past states. To overcome this limitation, long-term VI-SLAM, aka full VI-SLAM, combines the advantages of both the optimal (global) and the VIO (local) approaches, by employing a multi-thread scheme [63, 71, 77]: A frontend thread runs a VIO to track the current pose of the platform at a high frequency, while a backend thread optimizes, at a higher processing cost and hence lower frequency, over the entire trajectory (using either the optimal BLS [24] or its approximations [56, 74, 84]), and generates more accurate keyframe pose estimates (and global maps) for relocalization. This way, loop-closure measurements are utilized to improve long-term accuracy, while efficiency is still achieved at the same time.

Although real-time systems have been successfully developed for both short-term and long-term VI-SLAM (e.g., [39, 59, 63, 77]), there still exist many challenges that limit the efficiency and accuracy of such systems, especially when deployed on resource-constrained mobile devices. First of all, as for the problem of short-term VI-SLAM (or VIO), the demand for high computational efficiency is critical, i.e., a VIO algorithm needs to be extremely fast. This is due to the fact that a VIO serves as the tracking frontend for real-time operations, and any significant latency in this part of the navigation system would cause immediate performance degradation. As an example, in VR applications, low motion-to-photon latency ($<$ 20ms) is necessary to deliver a convincing immersive experience. If the VIO motion-tracking module causes a large latency, the screen display will lag behind the user's movement, and the user can experience disorientation and motion sickness. Meanwhile, existing VIO algorithms in the literature suffer from several issues that negatively affect their speed. One of the main limitations is the requirement of using double-precision arithmetic for implementation, due to the ill-conditioning of the VIO problem (i.e., covariance/Hessian matrix's condition number $> 10^9$), otherwise the numerical errors can easily become the dominant error source affecting estimation accuracy, or even cause the estimator to diverge. On the other hand, to achieve more computationally and memory efficient VIO solutions on mobile devices, single-precision arithmetic is desired for the following reasons: i) A 32-bit single-precision representation occupies less hardware resources, and the corresponding operations are likely to be faster, than 64-bit double-precision representations, and ii) most current mobile devices feature ARM NEON co-processors that provide a 4-fold processing speed increase for 32-bit floating-point operations. Therefore, a VIO algorithm that enables single-precision arithmetic is necessary for pose tracking with extremely-low latency on mobile devices.

Another challenge of the VIO problem is the balancing between estimation accuracy and computational efficiency. Given the large number of feature observations from camera images (typically hundreds per image, see Fig. 1.2), we need to determine a visual-information processing scheme, i.e., to decide which feature measurements to use and how to use them. Although more visual observations in general result in better localization accuracy, it has a diminishing gain while increases the computational cost linearly with the number of measurements. Thus, we need to select a portion of all available feature observations for bounding the processing cost, while ensuring accuracy. Additionally, multiple observations towards a same feature can be processed either as a mapped landmark for accuracy, as in the conventional SLAM approaches, or as pose constraints for efficiency, as in the multi-state constraint Kalman filter (MSC-KF) approach [69]. Moreover, since the measurement model describing the camera observation is nonlinear, it is necessary to perform linearization of each measurement before processing, either once for efficiency, as in the filtering-based methods [61, 69], or multiple times (i.e., relinearization) to reduce the linearization error for accuracy, as in the optimization-based methods [58, 77]. Therefore, given a limited processing resource on a mobile device, it is of critical importance to establish a visual-information processing scheme, that chooses between these options, so as to maintain the low latency of the VIO algorithm while providing highly-accurate pose estimates.

A second major issue of localization using the VINS arises when it is employed on mobile platforms that have restricted motions (e.g., wheeled robots or self-driving cars). Consider mobile robots navigating over a flat terrain. In many cases, it is necessary to estimate the full 6-degree-of-freedom (dof) pose of the robot in 3D space, since its motion is only approximately planar, e.g., due to the unevenness or roughness of the surface, or the presence of ramps, bumps, and low-height obstacles on the floor. Otherwise, the unmodeled part of the robot's odometry error will significantly increase and, in the absence of external corrections, may even cause the estimator to diverge. On the other hand, VINS has been employed to successfully estimate the 6-dof pose of mobile platforms moving in 3D. Therefore, one would expect that it would be straightforward to deploy a VINS for localizing robots moving in 2D (i.e., on a planar surface), as this is an instance of the more general case of 3D motion. Surprisingly, however, this is not true. In practice, we observe that the positioning error of VINS for such cases is significantly larger (e.g., $>$ 1m) than what is typically expected (e.g., $<$ 0.1m). This issue raises a fundamental question about the VINS: What are the observability properties of VI-SLAM

under special motion profiles? Observability is an important property of a system, not only because the observable directions provide the knowledge of what can be estimated, but also the unobservable directions can be leveraged to avoid gaining incorrect information for improving accuracy (i.e., observability-constrained filters [42, 47]). In the literature, it is well-known that the VI-SLAM has only 4 unobservable dof corresponding to 3 dof of global translation and 1 dof of rotation around the gravity vector (yaw) [42, 66]. This result, however, holds only when the IMU-camera pair undergoes generic 3D motion. In contrast, we need to analyze the VI-SLAM's observability properties under common motion profiles for ground vehicles, such as constant speed/acceleration or no rotation, and identify the additional unobservable directions if they exist. Furthermore, after the observability results are obtained, we need to extend the VINS model to incorporate extra information (e.g., from wheel encoders, or the planar-motion constraint) to ensure that information about the unobservable directions is always available, so as to improve the localization accuracy of VINS when deployed on wheeled robots.

Finally, as for long-term VI-SLAM, recent algorithms [63, 71, 77] manage to achieve accurate and efficient localization and mapping, but at the cost of inconsistent estimates. Specifically, to limit the processing cost, all these approaches employ approximations, e.g., keyframes involved in the frontend's relocalization are assumed to be *perfectly known*. Ignoring the corresponding uncertainties of these states and their cross correlations with the current states, however, leads to *inconsistent* estimates.[1] This means that the estimated covariance is unduly small and does not represent correctly the uncertainty of the current state estimates (i.e., it does not offer a reliable measure of the tracking quality). More importantly, combining these overly optimistic estimates with new measurements later on can further degrade the accuracy of the system, as new, precise measurements are weighted less in favor of the current estimates. In fact, this problem of inconsistency has been acknowledged in the past, and remedies are often used to alleviate its negative impact on estimation accuracy, e.g., by inflating the assumed covariance of the noise corresponding to the relocalization visual observations [65, 70]. These heuristics, however, offer no guarantees on the estimation consistency or the system's performance. Instead, to guarantee consistent estimates, it is necessary to employ an approach that

[1] As defined in [48, 49], a state estimator is consistent if the estimation errors are zero-mean and have covariance matrix smaller than or equal to the one calculated by the estimator. For the purposes of this work, we focus on the covariance requirement. Note that there exist additional sources of inconsistency, due to linearization errors and local minima (see e.g., [47]). In this work, we focus on the inconsistency caused by the assumption that uncertain quantities, such as a map, are perfectly known.

uses only consistent approximations. There exists one such estimator in the filtering domain: The Schmidt-Kalman filter (SKF) [80]. The key idea of the SKF is to update only a subset of the states (e.g., recent poses and features) and their corresponding covariance and cross correlation terms, while leaving the rest (e.g., past poses and features) unaltered. By doing so, the computational cost is reduced from quadratic to linear in the (potentially very large) size of unchanged states. Meanwhile, the uncertainty of the past states is correctly accounted for to guarantee consistent estimates. The SKF and its variants have been applied to the SLAM problem [38,50,73], where their major drawback is their high memory requirements: Quadratic in the size of all states due to the dense covariance matrix. Thus, the SKF cannot be employed in large-scale VI-SLAM. On the other hand, it is well-known that the information-domain solutions are more suitable for large-scale VI-SLAM, as the Hessian matrix and its corresponding Cholesky factor are sparse [87]. Therefore, estimators following the Schmidt approximation, but operating in the information form, is required, in order to obtain efficient and consistent estimates for long-term VI-SLAM. Moreover, based on the requirements of a long-term VI-SLAM system during its exploration (i.e., when navigating through a new area) vs. relocalization (i.e., when revisiting a previously-mapped area) phase, different configurations of the new estimator need to be employed, so as to maximize accuracy while preserving efficiency.

1.3 Research Objectives

The primary objective of the research works in this dissertation is to improve the efficiency, accuracy, and consistency of VI-SLAM, for both the short-term and long-term sub-problems, by addressing the aforementioned challenges.

1.3.1 Efficient Short-Term VI-SLAM

In this work, we aim to design a short-term VI-SLAM (or VIO) algorithm that is highly efficient and accurate. Specifically, while existing estimators for VIO operate in the regular (i.e., covariance or Hessian) forms and require double-precision number representations and arithmetic, we propose to employ their square-root equivalent [14,68]. By maintaining and updating the square-root factor of the covariance/Hessian matrix, square-root filters can yield the same effective precision of regular filters while using only half of the wordlength, i.e., to enable single-precision implementation. First, based on four equivalent forms of the filtering methods,

we analyze the potential gain in speed of square-root methods for VIO, especially on mobile devices, as most modern smart phones and tablets are equipped with the ARM Neon processor that allows for up to 4-time speed acceleration for 32-bit floating-point operations. To this end, we present a square-root inverse sliding-window filter (SR-ISWF) for VIO, which maintains the upper-triangular Cholesky factor of the Hessian matrix of a sliding window of recent poses and features.

Additionally, we establish the correspondence between two main categories of estimation methods in the VIO literature, i.e., the filtering and optimization-based approaches, with focus on their (re-)linearization behaviors. Based on our analysis, we design a hybrid visual-information processing scheme for our VIO algorithm. In order to balance between estimation accuracy and computational efficiency, our scheme combines: i) the SLAM and MSCKF feature processing approaches, ii) relinearization of a selective portion of visual measurements, and iii) a limiting budget on the number of features to be processed that bounds the total computational cost.

Moreover, we further improve the efficiency of the algorithm by investigating the underlying VIO problem structures. Under the square-root inverse estimation formulation, the main numerical operation is a QR factorization. We split this entire QR process into multiple steps, to better exploit the nonzero patterns of the Jacobian and information factor matrices involved at each step, so as to achieve computational savings.

1.3.2 Planar VI-SLAM: Observability Analysis and Model Extensions

In this work, we investigate the fundamental cause of the large estimation error of VINS when deployed on ground robots moving primarily on a plane. Based on our analysis, the restricted motion that ground robots often undergo alters the observability properties of VINS and renders additional unobservable directions. Specifically, we analytically show that: i) Scale becomes unobservable if the platform moves with constant local linear acceleration (e.g., when following a straight line path with constant speed or acceleration, or when making turns along a circular arc with constant speed); ii) All 3-dof of global orientation become unobservable if the platform does not rotate (e.g., when moving on a straight line, or a holonomic vehicle sliding sideways). Since observability is a fundamental property of the VINS model itself, these additional unobservable directions will negatively impact the accuracy of any VI-SLAM estimator employed, e.g., either BLS [24, 34, 51] or sliding-window filters/smoothers [15, 58, 59, 70, 89]. Although,

in practice, these specific motion constraints are never *exactly* satisfied all the time, when the robot (even temporarily) *approximately* follows them, it acquires very limited information along the unobservable directions. This will cause the information (Hessian) matrix estimated by the VINS to be severely ill-conditioned, or even numerically rank-deficient, and hence degrades the localization performance. As an evidence, we experimentally demonstrate the impact of such motion on the VINS accuracy.

Furthermore, motivated by the key findings of our observability analysis, we focus on improving the localization accuracy of VINS when deployed on wheeled robots. First, in order to ensure that information about the scale is always available, we extend the VINS algorithm to incorporate wheel-encoder measurements. Since these are often noisy and of frequency significantly lower than that of the IMU, we process them in a robust manner, by first integrating the raw encoder data and then treating them as inferred displacement measurements between consecutive poses. Additionally, we take advantage of the fact that the robot moves on an *approximately* flat surface and introduce the manifold-(m)VINS, which explicitly considers the planar-motion constraint in the estimation algorithm, to reduce drift out of the plane in position estimates. This is achieved by analyzing the motion profile of the robot, and its deviation from planar motion (e.g., due to terrain unevenness or vibration of the IMU-camera's mounting platform) and formulating stochastic (i.e., "soft"), instead of deterministic (i.e., "hard") constraints, that allow to properly model the vehicle's almost-planar motion.

1.3.3 Efficient and Consistent Long-Term VI-SLAM

In this work, we seek a novel estimator for efficient and accurate long-term VI-SLAM, while improving estimation consistency. Specifically, motivated by the potential processing savings of the SKF, as well as the low-memory requirements of the Hessian (or equivalently its Cholesky factor) representation of the uncertainty, we first derive the exact equivalent of the SKF in its square-root inverse form, i.e., by maintaining the Cholesky factor of the Hessian, since the corresponding portion of the information factor does not change. Surprisingly, unlike the case of the SKF, the exact inverse-form Schmidt estimator does *not* provide any computational savings as compared to the optimal solver. Moreover, the involved operations introduce a large number of fill-ins, leading to an almost dense information factor. This eventually makes the system too slow, and hence unsuitable for real-time long-term VI-SLAM. To overcome these limitations, we further introduce the resource-aware inverse Schmidt estimator (RISE), which is derived as

a further approximation of the exact inverse Schmidt estimator. The key idea behind RISE is to drop a certain portion of the available information, so that: i) As in the exact inverse Schmidt, past states as well as their corresponding portion of the information factor remain unaltered, while at the same time, ii) Recent states are updated only *approximately*, instead of *optimally*, so as to reduce both the processing cost and the factor fill-ins. Hence, RISE achieves both computational and memory efficiency by keeping the information factor sparse. Meanwhile, it is a *consistent* approximation to the optimal approach, as it only drops information, instead of assuming any state to be perfectly known. More importantly, RISE allows trading accuracy for efficiency, by adjusting the size of the window of the states selected to be updated.

Furthermore, we employ the RISE algorithm in various configurations to realize an accurate and efficient long-term VI-SLAM system, the RISE-SLAM, which maintains a *consistent* sparse information factor corresponding to all estimated states. Specifically, our system alternates between two modes, *exploration* and *relocalization*, based on the availability of loop-closure measurements. In order to balance between accuracy and efficiency, in each mode, RISE is employed with various window sizes and different state orders. Similarly to most recent VI-SLAM systems, our implementation incorporates two threads running in parallel: A fast frontend thread for estimating the current poses and features at a high frequency, and a lower-rate backend thread for globally adjusting the past states to achieve high accuracy. A key difference, however, as compared to existing systems that solve *multiple* optimization problems *independently* in different threads [63, 71, 77], is that RISE-SLAM always solves a *single* optimization problem, partitioned into two components each assigned to one of the two threads. This is only possible because of the structure of RISE, whose approximation allows focusing resources on only a subset of states at a time. As a result, in our system, important global corrections from the backend are *immediately* reflected onto the frontend estimates, hence improving the current tracking accuracy.

1.4 Structure of the Manuscript

- Chapter 2 describes a square-root inverse sliding-window filter (SR-ISWF) for efficient short-term VI-SLAM.
- Chapter 3 presents observability analysis and model extensions for accurate planar VI-SLAM.

- Chapter 4 introduces the RISE-SLAM algorithm for efficient long-term VI-SLAM with improved estimation consistency.
- Chapter 5 provides the concluding remarks and an outlook on future research directions.

Chapter 2

Efficient Short-Term Visual-Inertial Localization and Mapping

In this chapter, we address the problem of short-term visual-inertial localization and mapping, aka visual-inertial odometry (VIO). Specifically, we present a square-root inverse sliding-window filter (SR-ISWF) for highly efficient and accurate VIO. While existing estimators in the covariance/information form suffer from numerical issues, employing their square-root equivalent enables the usage of single-precision number representations and arithmetic, thus achieving considerable speedups as compared to double-precision alternatives, especially on resource-constrained mobile platforms. Additionally, we establish the correspondence between two main categories of estimation methods in the VIO literature, i.e., the filtering and optimization-based approaches, with focus on their (re-)linearization behaviors. Based on our analysis, we design a hybrid visual information processing scheme for our VIO algorithm, that balances between estimation accuracy and computational efficiency. Moreover, a detailed description of our SR-ISWF estimation algorithm is presented, which focuses on the numerical procedures that enable exploiting the problem's structure for gaining in efficiency. Our proposed approaches to visual information processing and Jacobian structure utilization are fairly general and can benefit other existing VIO algorithms. Finally, as compared to state-of-the-art VIO algorithms on public datasets, our SR-ISWF achieves superior localization accuracy and speed. Additional experimental results on commercial-grade cell phones demonstrate our system's ability for (faster than) real-time operations on such resource-constrained mobile devices.

2.1 Introduction and Related Work

Visual-inertial simultaneous localization and mapping (VI-SLAM), where inertial data from an inertial measurement unit (IMU) are combined with visual observations from a camera, to compute the position and orientation (pose) of the sensing platform and build a map of the environment simultaneously, has a wide range of applications, such as augmented/virtual reality, autonomous driving, and robot navigation in GPS-denied areas [45]. Over the past years, commercial-grade mobile devices (e.g., cell phones) have been recognized as promising platforms for VI-SLAM, because of their sensing capabilities, and wide-spread, low-cost availability [4].

Under certain assumptions, it is well known that the Maximum a Posteriori (MAP) estimator for VI-SLAM can be cast as a nonlinear batch least-squares (BLS) problem, or often referred to as bundle adjustment (BA) in the computer vision literature. BLS methods (e.g., [24, 87]), however, that optimize over the entire state history, have processing and memory requirements that increase at least linearly (often quadratically) with time, and thus are not amenable to real-time implementations. Although incremental forms [51, 52] and approximations of the BLS have been proposed (e.g., [56, 74]) to reduce the processing cost, they are still far from being able to generate real-time estimates on resource-constrained mobile devices. To address this issue, recent VI-SLAM systems employ a multi-thread scheme (e.g., [63, 71, 77]), where a frontend thread estimates a small window of recent states for real-time performance, while a backend thread optimizes, at a higher cost and lower frequency, over the entire trajectory with loop-closure constraints, and generates more accurate keyframe pose estimates and global maps for relocalization. In this chapter, we focus on the estimation problem in the frontend, i.e., to design an algorithm for visual-inertial odometry (VIO) that optimizes over a bounded-size sliding window of recent poses (as well as a local map), by marginalizing out past states and measurements. We consider VIO estimators that require only *constant* processing time and memory usage, without any long-term loop-closures.

The VIO problem has been studied extensively in the literature. One of the earliest successful VIO algorithm is the multi-state constraint Kalman filter (MSC-KF) [69], where visual observations are transformed into pose constraints by marginalizing the features, thereby excluding these features from the state vector to reduce the computational cost. Extensions of

the MSC-KF have been developed to improve its performance. In particular, based on the observability properties of the VIO(-SLAM) problem, observability-constrained (OC) MSC-KF algorithms (e.g., [42, 61]) are introduced to improve the estimation consistency and accuracy. A hybrid visual information processing scheme, combining pose constraints as in the MSC-KF and a local map as in the conventional SLAM approaches, have been employed to increase the (relatively) long-term accuracy and robustness of the estimator (e.g., [42, 60]). To address the hardware limitations of mobile devices, online estimation and compensation for rolling-shutter cameras with inaccurate time synchronization are developed (e.g. [39, 59]), as well as the inclusion of IMU-camera intrinsic and extrinsic parameters (e.g., [62]). Recently, a robocentric formulation of the MSC-KF is introduced in [44], for improving the linearity properties of the nonlinear measurement model and eliminating the observability mismatch issue as in the standard world-centric counterpart. Besides the MSC-KF algorithm, other extended Kalman filter (EKF)-based VIO methods have been developed, such as [15] that directly uses pixel intensity errors of image patches to achieve high tracking accuracy and robustness. While being highly efficient, all these EKF-based approaches suffer from potentially large linearization errors, due to the one-time linearization of the nonlinear measurement models before processing, and hence degrading the estimation accuracy.

On the other hand, the EKF's information-domain counterpart, the (extended) inverse filter (EIF), allows for relinearization of the nonlinear measurements in a straightforward manner. By formulating the VIO problem as nonlinear least squares, the Gauss-Newton method is employed to solve for the state estimates iteratively (e.g., [25,46,82]). Similarly, optimization-based methods (e.g., [58,63,77]) aim to obtain the solution of the same nonlinear problem by using general optimization solvers. In order to achieve efficiency, these methods maintain the sparsity of the information (Hessian) matrix by discarding information during their marginalization step. Nevertheless, through relinearization, all these information-domain approaches reduce the linearization error but with a higher computational cost.

Although real-time performance has been successfully demonstrated, even on cell phones, from both filtering-based and optimization-based VIO systems (e.g., [39, 59, 77]), all existing estimators suffer from the ill-conditioning of the VIO problem (i.e., covariance/Hessian's condition number $\geq 10^9$), which necessitates using double-precision arithmetic for implementation. Otherwise, the numerical errors can easily become the dominant error source affecting estimation accuracy, or even cause the filter to diverge. Meanwhile, to achieve more computationally

and memory efficient VIO solutions on mobile devices, single-precision arithmetic is desired for the following reasons: i) A 32-bit single-precision representation occupies less hardware resources, and the corresponding operations are likely to be faster, than 64-bit double-precision representations, and ii) most current mobile devices feature ARM NEON co-processors that provide a 4-fold processing speed increase for 32-bit floating-point operations.

In order to overcome the numerical limitations of the estimators in the regular (i.e., covariance or Hessian) forms, square-root filters [14, 68] have been developed in the past that improve numerical stability by maintaining and updating the square-root factor of the covariance/Hessian matrix. Since the condition number of the square-root factor is the square root of that of the corresponding covariance/Hessian matrix, square-root filters can yield the same effective precision of regular filters while using only half of the wordlength.

Motivated by the superior numerical properties and hence the potential gain in speed of square-root methods, in this chapter, we present the square-root inverse sliding-window filter (SR-ISWF) for VIO, which maintains the upper-triangular Cholesky factor of the Hessian matrix of a sliding window of recent poses and features. Specifically, we choose the inverse form, rather than the covariance form, for its simplicity in performing relinearizations as in the optimization-based approaches. Also, under the square-root inverse formulation, we focus on taking advantage of the corresponding structures of the VIO problem for computational efficiency. Our main contributions are:

- To the best of our knowledge, we present the first VIO estimator in the square-root information domain, which enables single-precision numerical representation and arithmetic for implementation. This leads to significant speedups of the program, especially on mobile devices.

- We establish the equivalence and correspondence between the filtering-based and optimization-based methods, especially on their (re-)linearization behaviors. Based on our analysis, we identify three key factors for designing a state estimator.

- We introduce a hybrid visual information management and processing scheme, that is suitable for any sparse-feature-based sliding-window VIO system, for balancing between estimation accuracy and computational efficiency.

- We show the specific problem structures under our proposed information processing scheme and the square-root inverse estimation formulation, that we exploit to obtain an

efficient VIO algorithm, the SR-ISWF. Moreover, some of the most important structure findings and our approach for handling them, to achieve significant computational savings, are applicable to other popular VIO estimators, such as the MSC-KF and its extensions.

- Our implementation of the proposed SR-ISWF algorithm outperforms alternative state-of-the-art VIO systems on public datasets, in terms of localization accuracy and speed. Experiments on cell phones show our system's capability for real-time operations on resource-constrained mobile devices.

The rest of the chapter is structured as follows: In Section 2.2, we discuss various popular estimation methods for the VIO-SLAM problem, and identify their key distinctive factors, based on which we propose our VIO estimator. Then, Section 2.3 presents our visual-inertial information management and processing scheme. The detailed SR-ISWF estimation algorithm is derived in Section 2.4, with special attention to the problem structures and our approach of utilizing them to achieve an efficient implementation. In Section 2.5, our algorithm is experimentally evaluated in terms of accuracy and processing speed. Finally, Section 2.6 concludes the chapter.

2.2 VIO-SLAM Estimators

The problem of visual-inertial odometry and/or SLAM (VIO-SLAM) can be cast as a state estimation problem, where the state typically consists of sensor poses and map feature positions (as well as other states such as calibration parameters, sensor biases, velocity, etc), and the measurements are provided by the IMU and camera sensors. There exist two main categories of estimation methods that are popular in the literature: *Filtering-based* (e.g., [15,32,42,44,46,61, 70,82]) and *Optimization-based* (e.g., [33,52,54,58,63,71,77,87]). Although many successful VIO-SLAM systems have been developed in the past years based on these two approaches, the correspondence and connection between them and their variants remain unclear. In this section, we establish the mathematical equivalence between various forms of the filtering methods, and show their correspondence to the optimization-based approaches.

Our derivation for the equivalence first focuses on the basic, optimal forms of each approach. The incorporation of approximations for efficiency that are popular in the VIO-SLAM literature,

such as assuming previous states as perfectly known (e.g., [54, 71, 77]) or discarding certain information from nonkeyframes (e.g, [58, 63, 74, 77]), are then discussed.

2.2.1 Problem Formulation

We start by presenting the estimation problem formulation. Note that, in this section, the assumptions and models are kept abstract and general in purpose, so that the results and conclusions drawn are not restricted to specific sensors or their configurations (e.g., stereo vs. mono). Details of the particular visual-inertial states and models used in our proposed VIO algorithm are presented later.

Our discussion follows a *sliding-window* scheme, where at each time step, some new states come in, then all current states get updated, and finally some old states are removed from the window. We consider the most general case where the number of states added and/or removed per step can vary (from zero to multiple), and hence the size of the window may change from time to time. This is arguably the broadest description of a SLAM estimation problem, as it can scale from a single state (e.g., pose tracking) to the full/incremental batch formulation, depending on the horizon and processing scheme within the window: What states to add or remove, when to add or remove them, which measurements to use, and which ones to absorb into the prior. These are key design choices that affect the accuracy and efficiency of the estimator as we discuss later.

State

The state vector to be estimated is denoted by $\mathbf{x}$. It typically comprises sensor (e.g., IMU or camera) poses, feature states (e.g., for points, lines, planes, etc), platform velocity, sensor biases (e.g., for gyroscopes and accelerometers), intrinsic and extrinsic parameters, time synchronization values between sensor clocks, to name a few.

At each time step, per the sliding-window scheme, some new states $\mathbf{x}_\nu$ are added onto the current state vector $\mathbf{x}$, while some old states $\mathbf{x}_\mu$ are marginalized out of the window. After marginalization, the remaining states $\mathbf{x}_\rho$ serve as the state vector for the next time step. In order to compute the state estimates, three sources of information are combined: The prior knowledge from previous time steps, the process model for the new states, and the measurements within the window's horizon. We briefly describe the models for each of these hereafter.

Prior

Prior information is typically available for the initial states, and then evolves through the update and marginalization process at each time step. Although, in general, the prior constraint at the first time step for the initial states can be nonlinear, all subsequent prior terms are in a linearized form due to the marginalization process. Therefore, without loss of generality, the prior can be described as a linearized constraint over the state $\mathbf{x}$:

$$\mathbf{J}_0\mathbf{x} - \mathbf{r}_0 = \mathbf{n}_0, \quad \text{with } \mathbf{n}_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.1}$$

which forms the following prior cost term:

$$\mathcal{C}_p = \|\mathbf{J}_0\mathbf{x} - \mathbf{r}_0\|^2_{\mathbf{I}} \tag{2.2}$$

where $\mathbf{J}_0$ and $\mathbf{r}_0$ are the prior Jacobian and residual, respectively, and $\mathbf{I}$ denotes the identity matrix. If $\mathbf{J}_0$ has full column rank (as for most filtering-based methods), the prior constraint (2.1) and the cost term (2.2) can be written equivalently as:

$$\mathbf{x} - \hat{\mathbf{x}}_0 = \mathbf{n}'_0, \quad \text{with } \mathbf{n}'_0 \sim \mathcal{N}(\mathbf{0}, \mathbf{P}_0) \tag{2.3}$$

$$\mathcal{C}_p = \|\mathbf{x} - \hat{\mathbf{x}}_0\|^2_{\mathbf{P}_0} \tag{2.4}$$

with

$$\hat{\mathbf{x}}_0 = (\mathbf{J}_0^T\mathbf{J}_0)^{-1}\mathbf{J}_0^T\mathbf{r}_0 \tag{2.5}$$

$$\mathbf{P}_0 = \mathbf{A}_0^{-1} = (\mathbf{J}_0^T\mathbf{J}_0)^{-1} \tag{2.6}$$

Hence, the prior knowledge can be represented as a Gaussian distribution on the state vector, with mean $\hat{\mathbf{x}}_0$ and covariance $\mathbf{P}_0$ (or information $\mathbf{A}_0$), respectively.

Depending on the specific update and marginalization schemes, the prior term can vary in terms of the states involved (e.g., from the oldest pose only, up to all states in the window), as well as the amount of measurement information being absorbed into the prior after each marginalization step. As we will explain later, these variations of the prior term is a distinctive factor between the efficiency and accuracy for different estimators.

Process Model

The process model describes the state evolution from the previous state $\mathbf{x}$ to the new state $\mathbf{x}_\nu$. For VIO-SLAM, for example, the process model typically corresponds to the motion constraints between consecutive poses using IMU integration. In general, it can be described as a nonlinear constraint over the states $\mathbf{x}$ and $\mathbf{x}_\nu$:

$$\mathbf{x}_\nu = \mathbf{f}(\mathbf{x}, \mathbf{u}) + \mathbf{w}, \quad \text{with } \mathbf{w} \sim \mathcal{N}(\mathbf{0}, \mathbf{W}) \tag{2.7}$$

which gives the following nonlinear process cost term:

$$\mathcal{C}_u = \|\mathbf{x}_\nu - \mathbf{f}(\mathbf{x}, \mathbf{u})\|^2_{\mathbf{W}} \tag{2.8}$$

where $\mathbf{f}$ is a nonlinear function describing the process model, $\mathbf{u}$ is the known control input (e.g., from IMU data), and $\mathbf{w}$ and $\mathbf{W}$ are the additive white Gaussian noise of the process and its covariance, respectively. Linearizing (2.7) around the state estimate $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}_\nu$, we obtain the linearized process constraint and cost term:

$$\widetilde{\mathbf{x}}_\nu \simeq \mathbf{\Phi}\widetilde{\mathbf{x}} - \mathbf{r}_u + \mathbf{w} \tag{2.9}$$

$$\mathcal{C}_u \simeq \| \begin{bmatrix} \mathbf{\Phi} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}} \\ \widetilde{\mathbf{x}}_\nu \end{bmatrix} - \mathbf{r}_u \|^2_{\mathbf{W}} \tag{2.10}$$

where $\widetilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ denotes the corresponding error state, and

$$\mathbf{\Phi} = \left. \frac{\partial \mathbf{f}}{\partial \mathbf{x}} \right|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{2.11}$$

$$\mathbf{r}_u = \hat{\mathbf{x}}_\nu - \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u}) \tag{2.12}$$

are the state transition (Jacobian) matrix and residual, respectively. If the new state's estimate is computed using the process model (as is usually the case for non-iterative filtering-based methods), i.e., $\hat{\mathbf{x}}_\nu = \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u})$, the process residual vanishes (i.e., $\mathbf{r}_u = \mathbf{0}$) as from (2.12).

Measurement Model

Sensor measurements, such as visual observations from the camera to feature points, lines, etc, provide useful information for determining the state of a VIO-SLAM problem. In general, an observation can be modeled as a nonlinear function of the entire state $\mathbf{x}$:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{n}, \quad \text{with } \mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \tag{2.13}$$

which gives the following nonlinear measurement cost term:

$$\mathcal{C}_z = \|\mathbf{z} - \mathbf{h}(\mathbf{x})\|^2_{\mathbf{\Sigma}} \tag{2.14}$$

where $\mathbf{h}$ is a nonlinear function describing the measurement model, $\mathbf{z}$ is the obtained measurement (e.g., from camera image), and $\mathbf{n}$ and $\mathbf{\Sigma}$ are the additive white Gaussian noise of the measurement and its covariance, respectively. Linearizing (2.13) around the state estimate $\hat{\mathbf{x}}$, we obtain the linearized measurement constraint and cost term:

$$\mathbf{r} \simeq \mathbf{H}\widetilde{\mathbf{x}} + \mathbf{n} \tag{2.15}$$

$$\mathcal{C}_z \simeq \|\mathbf{H}\widetilde{\mathbf{x}} - \mathbf{r}\|^2_{\mathbf{\Sigma}} \tag{2.16}$$

where

$$\mathbf{H} = \left.\frac{\partial \mathbf{h}}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} \tag{2.17}$$

$$\mathbf{r} = \mathbf{z} - \mathbf{h}(\hat{\mathbf{x}}) \tag{2.18}$$

are the measurement Jacobian matrix and residual, respectively.

Given the prior, process model, and measurement information, in what follows, we first present the filtering and optimization-based methods that fuse this information in order to estimate the state in a sliding-window scheme, and subsequently establish the correspondence between their steps and show their equivalence.

2.2.2 Filtering-based Methods

Filtering methods (e.g., [15, 61, 69]) take as input the prior (2.3)-(2.6) and the *linearized* process (2.9)-(2.12) and measurement (2.15)-(2.18) models. As mentioned earlier, there are two basic assumptions commonly made in most filtering-based VIO-SLAM algorithms: i) The prior covariance is always full rank, hence the estimate's uncertainty as well as its inverse (i.e., the information) is well defined [see (2.6)]; ii) The new state's estimate is computed using the process model (i.e., $\hat{\mathbf{x}}_\nu = \mathbf{f}(\hat{\mathbf{x}}, \mathbf{u})$), hence the process residual vanishes (i.e., $\mathbf{r}_u = \mathbf{0}$ [see (2.12)]). For the sake of simplicity, here we also make these two assumptions when discussing the filtering algorithms. Extensions to remove these assumptions, as appearing later in the optimization-based methods, will be discussed shortly.

As compared to the classic literature on the subject of filtering with the conventional two-step (prediction-update) process (e.g., [68]), our presentation of the filtering algorithms and the proofs of their equivalence follow a *three-step* process under the aforementioned sliding-window scheme. This is more general as it encompasses both filtering and optimization-based VIO-SLAM algorithms. Specifically, under this sliding-window scheme, at each time step, a filtering VIO-SLAM method performs the following three operations:

- **State augmentation**: The state vector $\mathbf{x}$, from the previous time step, is augmented with the new state $\mathbf{x}_\nu$. Both the state estimate and the corresponding covariance/information matrix are augmented, by combining the prior information (2.3)-(2.6) with the linearized process model (2.9)-(2.12).

- **Update**: The current (augmented) state vector $\mathbf{x}$ is updated to obtain the posterior estimate, as well as the corresponding covariance/information, using the linearized measurement model (2.15)-(2.18).

- **Marginalization**: A portion of the current state vector $\mathbf{x}$, typically the oldest states $\mathbf{x}_\mu$, are marginalized out of the window. Both the state estimate and the corresponding covariance/information matrix are marginalized, resulting in a prior (for the next time step) with respect to the remaining state $\mathbf{x}_\rho$.

Note that the order of the last two steps can be switched, i.e., the marginalization step can take place either before or after the update step. Also, this three-step process is a generalization of the conventional two-step (prediction-update) filtering process: If $\mathbf{x}_\nu$ and $\mathbf{x}$ represent the

same physical quantity (e.g., IMU pose) at two consecutive time steps, where the new state is initialized using the process model (e.g., IMU integration) while the old one is chosen to be marginalized, then the combined effect of the state augmentation and marginalization steps realizes the conventional prediction step.

To carry out the operations in these three steps, in the VIO-SLAM literature, filters have been employed in various forms. Based on their uncertainty/information representation, they can be classified into the following four basic forms [14, 68]:

- **Extended Kalman filter (EKF)** (e.g. [15, 69]): The covariance matrix $\mathbf{P}$ is maintained to represent the uncertainty of the state estimate.

- **Square-root extended Kalman filter (SR-EKF)**: For numerical stability, factorization methods utilize certain matrix decompositions of the covariance. Here, we focus on the SR-EKF where the upper-triangular Cholesky factor $\mathbf{U}$ of the covariance matrix $\mathbf{P}$ is maintained,[1] with $\mathbf{P} = \mathbf{U}^T\mathbf{U}$.

- **Extended inverse filter (EIF)** (e.g. [46, 82]): The EIF maintains the information matrix $\mathbf{A}$, which equals the inverse of the covariance $\mathbf{P}$, i.e., $\mathbf{A} = \mathbf{P}^{-1}$, and hence the name.[2]

- **Square-root extended inverse filter (SR-EIF)** (e.g. [89]): Similarly to the SR-EKF, the square-root form of the EIF maintains the information factor $\mathbf{R}$, which is the upper-triangular Cholesky factor of the information matrix $\mathbf{A}$, with $\mathbf{A} = \mathbf{R}^T\mathbf{R}$.

These four filters are well-established in the literature (e.g., [14, 68]), and we hereby consider their equations as readily known. The detailed algorithms are listed in Table 2.1, where the equations haven been adapted to follow the aforementioned three-step process for the sliding-window VIO-SLAM scheme, and organized according to the operations in each step, in order to show the correspondence between the four filters. Moreover, under the sliding-window scheme, when the new state is added during the state augmentation step in Table 2.1, two opposite state orderings are used between the covariance and the inverse forms respectively. The choice of

[1] Other factorization forms also exist, such as the U-D covariance filter, but are mathematically equivalent to the SR-EKF [14].

[2] A close relative to the EIF is the extended information filter [83, 88], where the information vector $\hat{\mathbf{y}} = \mathbf{P}^{-1}\hat{\mathbf{x}}$ is estimated instead of the state estimate $\hat{\mathbf{x}}$. These two formulations are very similar with each other and are mathematically equivalent. Our presentation here focuses on the EIF.

Table 2.1: The Filtering Methods: Four Equivalent Forms

	EKF	**SR-EKF**	**EIF**	**SR-EIF**
Input	Prior state estimate: $\mathbf{x}$ State transition matrix: $\mathbf{\Phi}$ Process noise covariance: $\mathbf{W}$ Measurement Jacobian: $\mathbf{H}$ Measurement residual: $\mathbf{r}$ Measurement noise covariance: $\mathbf{\Sigma}$			
	Prior covariance: $\mathbf{P}$	Prior cov. sqrt factor: $\mathbf{U}$ where $\mathbf{U}^T\mathbf{U} = \mathbf{P}$	Prior information matrix: $\mathbf{A}$ where $\mathbf{A}^{-1} = \mathbf{P}$	Prior information sqrt factor: $\mathbf{R}$ where $\mathbf{R}^T\mathbf{R} = \mathbf{A}$
State Augmentation	Add new state $\mathbf{x}_\nu$ to $\mathbf{x}$:			
	$\mathbf{x} \leftarrow \begin{bmatrix}\mathbf{x}_\nu \\ \mathbf{x}\end{bmatrix}$		$\mathbf{x} \leftarrow \begin{bmatrix}\mathbf{x} \\ \mathbf{x}_\nu\end{bmatrix}$	
	$\mathbf{P} \leftarrow \begin{bmatrix}\mathbf{\Phi P \Phi}^T + \mathbf{W} & \mathbf{\Phi P} \\ \mathbf{P \Phi}^T & \mathbf{P}\end{bmatrix}$	$\mathbf{U} \leftarrow \begin{bmatrix}\mathbf{W}^{\frac{T}{2}} & \mathbf{0} \\ \mathbf{U \Phi}^T & \mathbf{U}\end{bmatrix}$	$\mathbf{A} \leftarrow \begin{bmatrix}\mathbf{A} + \mathbf{\Phi}^T\mathbf{W}^{-1}\mathbf{\Phi} & -\mathbf{\Phi}^T\mathbf{W}^{-1} \\ -\mathbf{W}^{-1}\mathbf{\Phi} & \mathbf{W}^{-1}\end{bmatrix}$	$\mathbf{R} \leftarrow \begin{bmatrix}\mathbf{R} & \mathbf{0} \\ \mathbf{W}^{-\frac{1}{2}}\mathbf{\Phi} & -\mathbf{W}^{-\frac{1}{2}}\end{bmatrix}$
Update	$\mathbf{S} = \mathbf{HPH}^T + \mathbf{\Sigma}$ $\mathbf{P}^\oplus = \mathbf{P} - \mathbf{PH}^T\mathbf{S}^{-1}\mathbf{HP}$	$\begin{bmatrix}\mathbf{\Sigma}^{\frac{T}{2}} & \mathbf{0} \\ \mathbf{UH}^T & \mathbf{U}\end{bmatrix} \overset{\text{QR}}{=} \mathbf{T} \begin{bmatrix}\mathbf{S}^{\frac{T}{2}} & \mathbf{G} \\ \mathbf{0} & \mathbf{U}^\oplus\end{bmatrix}$	$\mathbf{A}^\oplus = \mathbf{A} + \mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{H}$ $\mathbf{b}^\oplus = \mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{r}$	$\begin{bmatrix}\mathbf{R} & \mathbf{0} \\ \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{H} & \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{r}\end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q} \begin{bmatrix}\mathbf{R}^\oplus & \mathbf{r}^\oplus \\ \mathbf{0} & \mathbf{e}\end{bmatrix}$
	$\mathbf{P} \leftarrow \mathbf{P}^\oplus$ $\Delta\mathbf{x} = \mathbf{PH}^T\mathbf{S}^{-1}\mathbf{r}$	$\mathbf{U} \leftarrow \mathbf{U}^\oplus$ $\Delta\mathbf{x} = \mathbf{G}^T\mathbf{S}^{-\frac{1}{2}}\mathbf{r}$	$\mathbf{A} \leftarrow \mathbf{A}^\oplus$ $\Delta\mathbf{x} = \mathbf{A}^{\oplus^{-1}}\mathbf{b}^\oplus$	$\mathbf{R} \leftarrow \mathbf{R}^\oplus$ $\Delta\mathbf{x} = \mathbf{R}^{\oplus^{-1}}\mathbf{r}^\oplus$
	Update state: $\mathbf{x}^\oplus = \mathbf{x} + \Delta\mathbf{x},\ \mathbf{x} \leftarrow \mathbf{x}^\oplus$			
Marginalization	Marginalize state $\mathbf{x}_\mu$ from $\mathbf{x}$, and keep remaining state $\mathbf{x}_\rho$: $\mathbf{x} \leftarrow \mathbf{x}_\rho$			
	Permutation $\mathbf{\Pi}$: $\mathbf{\Pi x} = \begin{bmatrix}\mathbf{x}_\rho \\ \mathbf{x}_\mu\end{bmatrix}$		Permutation $\mathbf{\Pi}'$: $\mathbf{\Pi}'\mathbf{x} = \begin{bmatrix}\mathbf{x}_\mu \\ \mathbf{x}_\rho\end{bmatrix}$	
	$\mathbf{\Pi P \Pi}^T = \begin{bmatrix}\mathbf{P}_{\rho\rho} & \mathbf{P}_{\rho\mu} \\ \mathbf{P}_{\mu\rho} & \mathbf{P}_{\mu\mu}\end{bmatrix}$	$\mathbf{U\Pi}^T = [\mathbf{U}_{:\rho}\ \mathbf{U}_{:\mu}]$ $\mathbf{U}_{:\rho} \overset{\text{QR}}{=} \mathbf{T}' \begin{bmatrix}\mathbf{U}'_{\rho\rho} \\ \mathbf{0}\end{bmatrix}$	$\mathbf{\Pi}'\mathbf{A}\mathbf{\Pi}'^T = \begin{bmatrix}\mathbf{A}_{\mu\mu} & \mathbf{A}_{\mu\rho} \\ \mathbf{A}_{\rho\mu} & \mathbf{A}_{\rho\rho}\end{bmatrix}$ $\mathbf{A}'_{\rho\rho} = \mathbf{A}_{\rho\rho} - \mathbf{A}_{\rho\mu}\mathbf{A}_{\mu\mu}^{-1}\mathbf{A}_{\mu\rho}$	$\mathbf{R\Pi}'^T = [\mathbf{R}_{:\mu}\ \mathbf{R}_{:\rho}]$ $[\mathbf{R}_{:\mu}\ \mathbf{R}_{:\rho}] \overset{\text{QR}}{=} \mathbf{Q}' \begin{bmatrix}\mathbf{R}'_{\mu\mu} & \mathbf{R}'_{\mu\rho} \\ \mathbf{0} & \mathbf{R}'_{\rho\rho}\end{bmatrix}$
	$\mathbf{P} \leftarrow \mathbf{P}_{\rho\rho}$	$\mathbf{U} \leftarrow \mathbf{U}'_{\rho\rho}$	$\mathbf{A} \leftarrow \mathbf{A}'_{\rho\rho}$	$\mathbf{R} \leftarrow \mathbf{R}'_{\rho\rho}$

these state orderings affects the computational efficiency of the covariance and inverse square-root filters as explained later. Also, for the clarity of presentation, the notation for the estimated quantities in Table 2.1 has been simplified (e.g., $\hat{\mathbf{x}}_0 \rightarrow \mathbf{x}$ and $\mathbf{P}_0 \rightarrow \mathbf{P}$).

In what follows, we establish the mathematical equivalence between the four filters in Table 2.1. We start by showing the pair-wise equivalence between the two regular forms (EKF vs. EIF), followed by pairs of each regular one with its corresponding square-root form (EKF vs. SR-EKF, and EIF vs. SR-EIF). For each step, we first describe the corresponding filter equations (considered as known from the literature), and then show their equivalence by proving that, given the same input (i.e., the prior, process model, and measurements), the filter pairs under consideration produce the exact same state and covariance/information estimates after each step. Finally, throughout these derivations, we also discuss the analogies between these different filter forms.

EKF $\Longleftrightarrow$ EIF

Establishing the equivalence between the EKF and the EIF requires proving that their state estimates are equal, and the resulting information matrix for the EIF is the inverse of the corresponding covariance matrix for the EKF. Note that in this derivation, certain permutations are involved due to the opposite state orderings that are used between the covariance and the inverse forms respectively (see Table 2.1 State Augmentation and Marginalization). While these state orderings have no impact on the EKF or the EIF, as it will become evident later on, they are chosen in order to improve the computational efficiency of the marginalization step of the corresponding square-root filters. Therefore, and to ensure easier correspondence later on with the square-root forms (SR-EKF and SR-EIF), these different state orderings are kept for the regular forms (EKF and EIF) here as well, and will not alter the equivalence result. In fact, our proof implies that the equivalence between these filters holds for any state ordering.

Proof. i) State augmentation: Given the prior in (2.3) and the process model in (2.9), the state

and covariance augmentation for the EKF can be written as:

$$\hat{\mathbf{x}}^{\ominus} = \begin{bmatrix} \hat{\mathbf{x}}_{\nu} \\ \hat{\mathbf{x}}_0 \end{bmatrix}, \;\; \hat{\mathbf{x}}_{\nu} = \mathbf{f}(\hat{\mathbf{x}}_0, \mathbf{u}) \tag{2.19}$$

$$\mathbf{P}^{\ominus} = \begin{bmatrix} \mathbf{\Phi}\mathbf{P}_0\mathbf{\Phi}^T + \mathbf{W} & \mathbf{\Phi}\mathbf{P}_0 \\ \mathbf{P}_0\mathbf{\Phi}^T & \mathbf{P}_0 \end{bmatrix} \tag{2.20}$$

On the other hand, from the equivalent prior cost term (2.4)-(2.6) and the process cost term (2.10), the state augmentation of the EIF is the same with that of the EKF in (2.19) with the permutation $\mathbf{\Pi}_0 = \begin{bmatrix} \mathbf{0} & \mathbf{I} \\ \mathbf{I} & \mathbf{0} \end{bmatrix}$, i.e., $\hat{\mathbf{x}}^{\ominus} = \begin{bmatrix} \hat{\mathbf{x}}_0 \\ \hat{\mathbf{x}}_{\nu} \end{bmatrix}$, while the corresponding augmented information matrix is:

$$\begin{aligned} \mathbf{A}^{\ominus} &= \begin{bmatrix} \mathbf{A}_0 & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} + \begin{bmatrix} \mathbf{\Phi}^T \\ -\mathbf{I} \end{bmatrix} \mathbf{W}^{-1} \begin{bmatrix} \mathbf{\Phi} & -\mathbf{I} \end{bmatrix} \\ &= \begin{bmatrix} \mathbf{A}_0 + \mathbf{\Phi}^T\mathbf{W}^{-1}\mathbf{\Phi} & -\mathbf{\Phi}^T\mathbf{W}^{-1} \\ -\mathbf{W}^{-1}\mathbf{\Phi} & \mathbf{W}^{-1} \end{bmatrix} \end{aligned} \tag{2.21}$$

The equivalence between (2.20) and (2.21) holds, as it can be easily verified that, given $\mathbf{P}_0 = \mathbf{A}_0^{-1}$ [see (2.6)], $\mathbf{A}^{\ominus}$ is the inverse of $\mathbf{P}^{\ominus}$ with the permutation $\mathbf{\Pi}_0$, i.e., $\mathbf{P}^{\ominus} = \mathbf{\Pi}_0\mathbf{A}^{\ominus^{-1}}\mathbf{\Pi}_0^T$, by employing the block matrix inversion lemma [12].

ii) Update: The EKF performs state and covariance update based on the current prior and the measurement model in (2.15). Meanwhile, the EIF minimizes the cost function that combines cost terms corresponding to the same prior and measurement in (2.16). To solve this (linearized) least-squares problem, the EIF formulates the corresponding normal equation by computing [68]:

$$\mathbf{A}^{\oplus} = \mathbf{A} + \mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{H} \tag{2.22}$$

$$\mathbf{b}^{\oplus} = \mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{r} \tag{2.23}$$

Then, the updated information matrix is $\mathbf{A}^{\oplus}$, while the updated state estimate $\hat{\mathbf{x}}^{\oplus}$ is given by $\hat{\mathbf{x}}^{\oplus} = \hat{\mathbf{x}} + \Delta\mathbf{x}$, where the state correction $\Delta\mathbf{x}$ is obtained by solving the normal equation:

$$\mathbf{A}^{\oplus}\Delta\mathbf{x} = \mathbf{b}^{\oplus} \tag{2.24}$$

which can be computed efficiently by a Cholesky factorization of $\mathbf{A}^{\oplus}$. To prove the equivalence with the EKF update, given that $\mathbf{A}^{-1} = \mathbf{P}$, we compute the inverse of the posterior information matrix $\mathbf{A}^{\oplus}$ in (2.22) using the matrix inversion lemma [12], i.e.,

$$\mathbf{A}^{\oplus^{-1}} = (\mathbf{A} + \mathbf{H}^T\boldsymbol{\Sigma}^{-1}\mathbf{H})^{-1} \tag{2.25}$$

$$= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}^T(\mathbf{H}\mathbf{A}^{-1}\mathbf{H}^T + \boldsymbol{\Sigma})^{-1}\mathbf{H}\mathbf{A}^{-1} \tag{2.26}$$

$$= \mathbf{P} - \mathbf{P}\mathbf{H}^T\underbrace{(\mathbf{H}\mathbf{P}\mathbf{H}^T + \boldsymbol{\Sigma})}_{\mathbf{S}}^{-1}\mathbf{H}\mathbf{P} = \mathbf{P}^{\oplus} \tag{2.27}$$

which becomes the standard covariance update for the EKF, with the residual covariance $\mathbf{S}$. On the other hand, the state correction from (2.24) is:

$$\Delta\mathbf{x} = \mathbf{A}^{\oplus^{-1}}\mathbf{b}^{\oplus} = \mathbf{P}^{\oplus}\mathbf{b}^{\oplus} \tag{2.28}$$

$$= (\mathbf{P} - \mathbf{P}\mathbf{H}^T\mathbf{S}^{-1}\mathbf{H}\mathbf{P})\mathbf{H}^T\boldsymbol{\Sigma}^{-1}\mathbf{r} \tag{2.29}$$

$$= \mathbf{P}\mathbf{H}^T\mathbf{S}^{-1}(\mathbf{S} - \mathbf{H}\mathbf{P}\mathbf{H}^T)\boldsymbol{\Sigma}^{-1}\mathbf{r} \tag{2.30}$$

$$= \underbrace{\mathbf{P}\mathbf{H}^T\mathbf{S}^{-1}}_{\mathbf{K}}\mathbf{r} = \mathbf{K}\mathbf{r} \tag{2.31}$$

which is the state correction for the EKF update, with the standard Kalman gain $\mathbf{K}$. Note that (2.31) can also be obtained directly from (2.28), using the alternative expression of the Kalman gain as $\mathbf{K} = \mathbf{P}^{\oplus}\mathbf{H}^T\boldsymbol{\Sigma}^{-1}$ [68].

iii) Marginalization: In order to separate the state $\mathbf{x}_\mu$ to be marginalized and the remaining state $\mathbf{x}_\rho$, the EKF first permutes and partitions the state vector and the covariance into:

$$\boldsymbol{\Pi}\hat{\mathbf{x}} = \begin{bmatrix}\hat{\mathbf{x}}_\rho \\ \hat{\mathbf{x}}_\mu\end{bmatrix}, \quad \boldsymbol{\Pi}\mathbf{P}\boldsymbol{\Pi}^T = \begin{bmatrix}\mathbf{P}_{\rho\rho} & \mathbf{P}_{\rho\mu} \\ \mathbf{P}_{\mu\rho} & \mathbf{P}_{\mu\mu}\end{bmatrix} \tag{2.32}$$

Then, for marginalization, the EKF simply takes $\hat{\mathbf{x}}_\rho$ and $\mathbf{P}_{\rho\rho}$ as the remaining state and covariance estimates. On the other hand, the EIF employs a different permutation where the remaining state $\mathbf{x}_\rho$ is placed at the bottom of the state vector:

$$\boldsymbol{\Pi}'\hat{\mathbf{x}} = \begin{bmatrix}\hat{\mathbf{x}}_\mu \\ \hat{\mathbf{x}}_\rho\end{bmatrix}, \quad \boldsymbol{\Pi}'\mathbf{A}\boldsymbol{\Pi}'^T = \begin{bmatrix}\mathbf{A}_{\mu\mu} & \mathbf{A}_{\mu\rho} \\ \mathbf{A}_{\rho\mu} & \mathbf{A}_{\rho\rho}\end{bmatrix} \tag{2.33}$$

These opposite state orderings are chosen in correspondence to the ones required by their square-root forms as shown later. Then, for marginalization, the EIF takes the same remaining state estimate $\hat{\mathbf{x}}_\rho$, while the information is computed using the Schur complement [68]:

$$\mathbf{A}'_{\rho\rho} = \mathbf{A}_{\rho\rho} - \mathbf{A}_{\rho\mu}\mathbf{A}_{\mu\mu}^{-1}\mathbf{A}_{\mu\rho} \tag{2.34}$$

To prove their equivalence, from (2.32)-(2.33) and given $\mathbf{A}^{-1} = \mathbf{P}$, it is easy to verify that $\begin{bmatrix} \mathbf{P}_{\rho\rho} & \mathbf{P}_{\rho\mu} \\ \mathbf{P}_{\mu\rho} & \mathbf{P}_{\mu\mu} \end{bmatrix} = \begin{bmatrix} \mathbf{A}_{\rho\rho} & \mathbf{A}_{\rho\mu} \\ \mathbf{A}_{\mu\rho} & \mathbf{A}_{\mu\mu} \end{bmatrix}^{-1}$. Hence, from the block matrix inversion lemma and the definition of $\mathbf{A}'_{\rho\rho}$ in (2.34), we obtain $\mathbf{P}_{\rho\rho} = (\mathbf{A}_{\rho\rho} - \mathbf{A}_{\rho\mu}\mathbf{A}_{\mu\mu}^{-1}\mathbf{A}_{\mu\rho})^{-1} = \mathbf{A}'^{-1}_{\rho\rho}$. □

Remark 1: Under the assumption of the multivariate Gaussian distribution with respect to the state vector and the residual, the EIF is the *dual* of the EKF, in terms of the fundamental operations of *conditioning* and *marginalization* [68]. Specifically, the update step computes the *conditional* distribution $p(\widetilde{\mathbf{x}}|\mathbf{r})$ of the jointly-Gaussian random vector $\begin{bmatrix} \mathbf{r} \\ \widetilde{\mathbf{x}} \end{bmatrix}$, which has the following covariance:

$$cov(\begin{bmatrix} \mathbf{r} \\ \widetilde{\mathbf{x}} \end{bmatrix}) = \begin{bmatrix} \mathbf{S} & \mathbf{HP} \\ \mathbf{PH}^T & \mathbf{P} \end{bmatrix} \triangleq \mathbf{\Psi} \tag{2.35}$$

and the information matrix (i.e., the inverse of the covariance):

$$\mathbf{\Psi}^{-1} = \begin{bmatrix} \mathbf{\Sigma}^{-1} & -\mathbf{\Sigma}^{-1}\mathbf{H} \\ -\mathbf{H}^T\mathbf{\Sigma}^{-1} & \mathbf{A} + \mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{H} \end{bmatrix} \triangleq \mathbf{\Xi} \tag{2.36}$$

While conditioning is *hard* in the covariance form, requiring the computation of the Schur complement in $\mathbf{\Psi}$ to obtain the updated covariance $\mathbf{P}^\oplus$ for the EKF [see (2.27)], it is *easy* in the information form, which simply requires the bottom-right block in $\mathbf{\Xi}$ as the updated information $\mathbf{A}^\oplus$ for the EIF [see (2.22)]. In contrast, the opposite is true regarding the *marginalization* operation, i.e., it is easy for the EKF [see (2.32)] while requires the Schur complement for the EIF [see (2.34)].

For the problem of SLAM, it is well-known that while the covariance matrix is always dense, the information (Hessian) matrix can be sparse, e.g., under the batch least-squares formulation [24, 87], or under the sliding-window scheme but with approximations such as discarding certain measurements [58, 77]. Therefore, the inverse-domain approaches (e.g., the EIF) are

more efficient for SLAM estimation problems where the size of the map is large.

EKF $\Longleftrightarrow$ SR-EKF

To prove the equivalence between them, we need to show that their state estimates are equal, and the resulting covariance factor for the SR-EKF is the (upper-triangular) square-root factor of the corresponding covariance matrix for the EKF.

Proof. i) State augmentation: As compared to the EKF, the state augmentation of the SR-EKF is the same as in (2.19), while the corresponding covariance factor is augmented as:

$$\mathbf{U}^{\ominus} = \begin{bmatrix} \mathbf{W}^{\frac{T}{2}} & \mathbf{0} \\ \mathbf{U}_0\mathbf{\Phi}^T & \mathbf{U}_0 \end{bmatrix} \tag{2.37}$$

where $\mathbf{U}_0^T\mathbf{U}_0 = \mathbf{P}_0$ and $\mathbf{W}^{\frac{1}{2}}\mathbf{W}^{\frac{T}{2}} = \mathbf{W}$. To prove the equivalence, it is straightforward to show that $\mathbf{U}^{\ominus^T}\mathbf{U}^{\ominus} = \mathbf{P}^{\ominus}$ from (2.20). Note that the resulting factor $\mathbf{U}^{\ominus}$ in (2.37) is square but not upper-triangular. In practice, an extra QR factorization step can be applied to upper triangularize this factor.

ii) Update: The SR-EKF update equations can be derived from the residual-error state joint covariance $\mathbf{\Psi}$ defined in (2.35). Specifically, it is easy to verify that $\mathbf{\Psi}$ has a square-root factor $\mathbf{\Psi}^{\frac{1}{2}} = \begin{bmatrix} \mathbf{\Sigma}^{\frac{T}{2}} & \mathbf{0} \\ \mathbf{U}\mathbf{H}^T & \mathbf{U} \end{bmatrix}$, on which the SR-EKF update performs a QR factorization as [9]:

$$\begin{bmatrix} \mathbf{\Sigma}^{\frac{T}{2}} & \mathbf{0} \\ \mathbf{U}\mathbf{H}^T & \mathbf{U} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{T} \begin{bmatrix} \mathbf{S}^{\frac{T}{2}} & \mathbf{G} \\ \mathbf{0} & \mathbf{U}^{\oplus} \end{bmatrix} \tag{2.38}$$

where $\mathbf{U}^T\mathbf{U} = \mathbf{P}$, $\mathbf{\Sigma}^{\frac{1}{2}}\mathbf{\Sigma}^{\frac{T}{2}} = \mathbf{\Sigma}$, and $\mathbf{T}$ is the orthogonal transformation matrix of the QR factorization with $\mathbf{T}^T\mathbf{T} = \mathbf{T}\mathbf{T}^T = \mathbf{I}$. Then, from the result of this QR factorization in (2.38), the SR-EKF takes $\mathbf{U}^{\oplus}$ as the updated upper-triangular covariance factor, while the state correction term $\Delta\mathbf{x}$ is computed as:

$$\Delta\mathbf{x} = \mathbf{G}^T\mathbf{S}^{-\frac{1}{2}}\mathbf{r} \tag{2.39}$$

Now we show its equivalence with the EKF update. First, from the two block columns of (2.38)

and the orthogonality of $\mathbf{T}$, we obtain:

$$\mathbf{S}^{\frac{1}{2}}\mathbf{S}^{\frac{T}{2}} = \mathbf{H}\mathbf{U}^T\mathbf{U}\mathbf{H}^T + \mathbf{\Sigma} = \mathbf{S} \tag{2.40}$$

$$\mathbf{U}^{\oplus^T}\mathbf{U}^{\oplus} = \mathbf{U}^T\mathbf{U} - \mathbf{G}^T\mathbf{G} \tag{2.41}$$

where $\mathbf{S}$ is defined in (2.27). Hence, from (2.40), the resulting top-left block $\mathbf{S}^{\frac{T}{2}}$ of the QR factor is indeed a square-root factor of the residual covariance $\mathbf{S}$. Partitioning $\mathbf{T} = \begin{bmatrix}\mathbf{T}_1 & \mathbf{T}_2\end{bmatrix}$, then, from (2.38), it is straightforward to show that $\mathbf{G} = \mathbf{T}_1^T \begin{bmatrix}\mathbf{0} \\ \mathbf{U}\end{bmatrix}$ and $\mathbf{T}_1 = \begin{bmatrix}\mathbf{\Sigma}^{\frac{T}{2}} \\ \mathbf{U}\mathbf{H}^T\end{bmatrix}\mathbf{S}^{-\frac{T}{2}}$, which gives the expression of $\mathbf{G}$ as:

$$\mathbf{G} = \mathbf{S}^{-\frac{1}{2}}\mathbf{H}\mathbf{U}^T\mathbf{U} = \mathbf{S}^{-\frac{1}{2}}\mathbf{H}\mathbf{P} \tag{2.42}$$

Finally, substituting (2.42) into (2.41), we obtain:

$$\mathbf{U}^{\oplus^T}\mathbf{U}^{\oplus} = \mathbf{U}^T\mathbf{U} - \mathbf{G}^T\mathbf{G} = \mathbf{P} - \mathbf{P}\mathbf{H}^T\mathbf{S}^{-1}\mathbf{H}\mathbf{P} = \mathbf{P}^{\oplus} \tag{2.43}$$

Hence, the updated factor $\mathbf{U}^{\oplus}$ is the square-root factor of the posterior covariance $\mathbf{P}^{\oplus}$ for the EKF as in (2.27). Similarly for the state correction term in (2.39):

$$\Delta\mathbf{x} = \mathbf{G}^T\mathbf{S}^{-\frac{1}{2}}\mathbf{r} = \mathbf{P}\mathbf{H}^T\mathbf{S}^{-1}\mathbf{r} = \mathbf{K}\mathbf{r} \tag{2.44}$$

which is the state correction for the EKF update, with the standard Kalman gain $\mathbf{K}$ as in (2.31).

iii) Marginalization: Similarly to the EKF, the SR-EKF employs the same permutation $\mathbf{\Pi}$ to partition the state vector and takes the remaining state $\hat{\mathbf{x}}_\rho$ as in (2.32). As for the covariance factor $\mathbf{U}$, only its columns need to be permuted, instead of a symmetric permutation as in the case of the covariance [see (2.32)], since row permutations have no impact on a square-root factor. Specifically, the factor $\mathbf{U}$ is permuted and then partitioned column-wise as:

$$\mathbf{U}\mathbf{\Pi}^T = \begin{bmatrix}\mathbf{U}_{:\rho} & \mathbf{U}_{:\mu}\end{bmatrix} \tag{2.45}$$

Note that this permutation $\mathbf{\Pi}$ is chosen so that $\hat{\mathbf{x}}_\rho$ is placed at the *top* of the state vector, and hence its corresponding columns $\mathbf{U}_{:\rho}$ in the covariance factor are permuted to the *left* [see (2.45)], as required by the following procedure. Then, a QR factorization is performed

on $\mathbf{U}_{:\rho}$ as:

$$\mathbf{U}_{:\rho} \overset{\text{QR}}{=} \mathbf{T}' \begin{bmatrix} \mathbf{U}'_{\rho\rho} \\ \mathbf{0} \end{bmatrix} \tag{2.46}$$

Given $\mathbf{U}^T\mathbf{U} = \mathbf{P}$, from (2.32) and (2.45)-(2.46), it is straightforward to show that $\mathbf{U}'^T_{\rho\rho}\mathbf{U}'_{\rho\rho} = \mathbf{U}^T_{:\rho}\mathbf{U}_{:\rho} = \mathbf{P}_{\rho\rho}$, and hence after this marginalization, the resulting upper-triangular QR factor $\mathbf{U}'_{\rho\rho}$ for the SR-EKF is the square-root factor of the corresponding covariance $\mathbf{P}_{\rho\rho}$ for the EKF. □

EIF $\Longleftrightarrow$ SR-EIF

Similarly to the previous case, to prove the equivalence between them, we need to show that their state estimates are equal, and the resulting information factor for the SR-EIF is the (upper-triangular) square-root factor of the corresponding information matrix for the EIF.

Proof. i) State augmentation: As compared to the EIF, the state augmentation of the SR-EIF is the same, while the corresponding information factor is augmented as:

$$\mathbf{R}^{\ominus} = \begin{bmatrix} \mathbf{R}_0 & \mathbf{0} \\ \mathbf{W}^{-\frac{1}{2}}\mathbf{\Phi} & -\mathbf{W}^{-\frac{1}{2}} \end{bmatrix} \tag{2.47}$$

where $\mathbf{R}_0^T\mathbf{R}_0 = \mathbf{A}_0$. To prove the equivalence, it is straightforward to show that $\mathbf{R}^{\ominus^T}\mathbf{R}^{\ominus} = \mathbf{A}^{\ominus}$ from (2.21). Note that, similarly to the factor augmentation for the SR-EKF, the resulting factor $\mathbf{R}^{\ominus}$ in (2.47) is square but not upper-triangular, and an extra QR factorization step can be applied to upper triangularize this factor.

ii) Update: As compared to the EIF that solves the corresponding least-squares problem by the normal equation as in (2.22)-(2.24), the SR-EIF operates in the square-root domain, i.e., using the information factor and the measurement Jacobian. Specifically, given $\mathbf{R}^T\mathbf{R} = \mathbf{A}$, it is easy to verify that the updated information $\mathbf{A}^{\oplus}$ in (2.22) has a square-root factor $\mathbf{A}^{\oplus^{\frac{1}{2}}} = \begin{bmatrix} \mathbf{R} \\ \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{H} \end{bmatrix}$, on which the SR-EIF update performs a QR factorization as [9]:

$$\begin{bmatrix} \mathbf{R} & \mathbf{0} \\ \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{H} & \mathbf{\Sigma}^{-\frac{1}{2}}\mathbf{r} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q} \begin{bmatrix} \mathbf{R}^{\oplus} & \mathbf{r}^{\oplus} \\ \mathbf{0} & \mathbf{e} \end{bmatrix} \tag{2.48}$$

where the factor $\mathbf{A}^{\oplus^{\frac{1}{2}}}$ on the left-hand side has been augmented with the prewhitened residual vector $\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{r}$, so as to obtain $\mathbf{r}^{\oplus}$ efficiently in-place [37], and $\mathbf{Q}$ is the orthogonal transformation matrix of the QR factorization. Then, from the result of this QR factorization in (2.48), the SR-EIF takes $\mathbf{R}^{\oplus}$ as the updated upper-triangular information factor, while the state correction term $\Delta\mathbf{x}$ is computed as:

$$\Delta\mathbf{x} = \mathbf{R}^{\oplus^{-1}}\mathbf{r}^{\oplus} \tag{2.49}$$

Now we show its equivalence with the EIF update. First, from the first block column of (2.48) and the orthogonality of $\mathbf{Q}$, we obtain:

$$\mathbf{R}^{\oplus^{T}}\mathbf{R}^{\oplus} = \mathbf{R}^{T}\mathbf{R} + \mathbf{H}^{T}\boldsymbol{\Sigma}^{-1}\mathbf{H} = \mathbf{A}^{\oplus} \tag{2.50}$$

Hence, the updated factor $\mathbf{R}^{\oplus}$ is the square-root factor of the posterior information $\mathbf{A}^{\oplus}$ for the EIF as in (2.22). Similarly, it can be shown that:

$$\mathbf{R}^{\oplus^{T}}\mathbf{r}^{\oplus} = \mathbf{H}^{T}\boldsymbol{\Sigma}^{-1}\mathbf{r} = \mathbf{b}^{\oplus} \tag{2.51}$$

where $\mathbf{b}^{\oplus}$ is defined in (2.23). Then, from (2.50)-(2.51), the state correction term in (2.49) can be written as:

$$\Delta\mathbf{x} = \mathbf{R}^{\oplus^{-1}}\mathbf{r}^{\oplus} = \mathbf{R}^{\oplus^{-1}}\mathbf{R}^{\oplus^{-T}}\mathbf{R}^{\oplus^{T}}\mathbf{r}^{\oplus} = \mathbf{A}^{\oplus^{-1}}\mathbf{b}^{\oplus} \tag{2.52}$$

which is the state correction for the EIF update as in (2.28).

iii) Marginalization: Similarly to the EIF, the SR-EIF employs the same permutation $\boldsymbol{\Pi}'$ to partition the state vector and takes the remaining state $\hat{\mathbf{x}}_{\rho}$ as in (2.33). Accordingly, the information factor $\mathbf{R}$ is permuted and then partitioned column-wise as:

$$\mathbf{R}\boldsymbol{\Pi}'^{T} = \begin{bmatrix}\mathbf{R}_{:\mu} & \mathbf{R}_{:\rho}\end{bmatrix} \tag{2.53}$$

Note that, in contrast to the state reordering for the SR-EKF's marginalization [see (2.45)], this permutation $\boldsymbol{\Pi}'$ is chosen so that $\hat{\mathbf{x}}_{\rho}$ is placed at the *bottom* of the state vector, and hence its corresponding columns $\mathbf{R}_{:\rho}$ in the information factor are permuted to the *right* [see (2.53)], as required by the following procedure. Then, a QR factorization is performed on the entire

permuted factor as:

$$\begin{bmatrix} \mathbf{R}_{:\mu} & \mathbf{R}_{:\rho} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q}' \begin{bmatrix} \mathbf{R}'_{\mu\mu} & \mathbf{R}'_{\mu\rho} \\ \mathbf{0} & \mathbf{R}'_{\rho\rho} \end{bmatrix} \tag{2.54}$$

from which the SR-EIF takes the bottom-right block $\mathbf{R}'_{\rho\rho}$ as the resulting upper-triangular information factor after marginalization. To show its equivalence with the EIF, given $\mathbf{R}^T\mathbf{R} = \mathbf{A}$, from (2.33) and (2.53)-(2.54), the QR factor $\begin{bmatrix} \mathbf{R}'_{\mu\mu} & \mathbf{R}'_{\mu\rho} \\ \mathbf{0} & \mathbf{R}'_{\rho\rho} \end{bmatrix}$ is indeed a square-root factor of the permuted information matrix $\begin{bmatrix} \mathbf{A}_{\mu\mu} & \mathbf{A}_{\mu\rho} \\ \mathbf{A}_{\rho\mu} & \mathbf{A}_{\rho\rho} \end{bmatrix}$, from which it can be easily verified that $\mathbf{R}'^T_{\rho\rho}\mathbf{R}'_{\rho\rho} = \mathbf{A}_{\rho\rho} - \mathbf{A}_{\rho\mu}\mathbf{A}^{-1}_{\mu\mu}\mathbf{A}_{\mu\rho} = \mathbf{A}'_{\rho\rho}$ as defined in (2.34). Therefore, for the marginalization, the resulting information factor $\mathbf{R}'_{\rho\rho}$ for the SR-EIF is the square-root factor of the corresponding information matrix $\mathbf{A}'_{\rho\rho}$ for the EIF. □

Remark 2: For the update step, the regular filters operate with the covariance/information matrix and employ the *Cholesky decomposition*, on the residual covariance matrix for the EKF [see (2.27)] or the posterior information matrix for the EIF [see (2.24)]. In contrast, the square-root filters operate with the square-root factors and Jacobians and employ the *QR decomposition* (see (2.38) for the SR-EKF and (2.48) for the SR-EIF). Moreover, since the update step essentially translates into adding more information to the prior, the EIF increases its information by *addition* of the measurement information [see (2.22)] while the EKF decreases its covariance by *subtraction* [see (2.27)], and hence correspondingly in their square-root forms, the update process can be represented as a Cholesky factor *update* for the SR-EIF [see (2.50)] vs. *downdate* for the SR-EKF [see (2.43)].

Remark 3: As compared to the regular filters (the EKF and EIF), these square-root forms have several advantages in numerical stability [9, 14]. Specifically, while the numerical representation of the covariance/information matrix can be *indefinite* in practice due to numerical errors, the product of the corresponding square-root factor is always *nonnegative definite*, and hence results in a better numerical representation. More importantly, the condition number of the square-root factor is the *square root* of that of the corresponding covariance/information matrix. This means that only *half* as many significant digits are required for the square-root filter computations as compared with those of the regular filters, leading to potential faster programs for real-world systems.

Remark 4: While the *state ordering* does not affect the computational complexity of the *regular* filters due to the symmetry of the covariance/information matrix, it is an important factor for all three steps of the *square-root* filters where the factor is kept upper-triangular. Specifically, as for marginalization, it requires less computation if the columns corresponding to the states to be marginalized are ordered to the right within the covariance factor, as from (2.45)-(2.46) for the SR-EKF, while the opposite is true for the SR-EIF marginalization [see (2.53)-(2.54)]. These two state orderings are reflected in Table 2.1 under the sliding-window scheme, where the new states are added to one end of the state vector so that the old states (to be marginalized) appear on the other end in favor of the marginalization computation for each square-root filter, respectively. Note that, however, this ordering is not necessarily optimal for the computation of the state augmentation or the update step. In practice, in order to choose the right state ordering for a square-root estimation algorithm (i.e., the ordering of the *columns* of the corresponding upper-triangular covariance/information factor) that minimizes the overall complexity, operations from all steps should be considered, with properties of the specific problem taken into account.

In summary, there are four basic forms for the filtering methods, i.e, the covariance and inverse filters with their corresponding square-root versions, and through the three-step process under the sliding-window scheme, we have shown that they are all mathematically equivalent to each other. This conclusion agrees with that of the classic filtering literature (e.g., [14, 68]). In addition, since these filters operate in different domains (covariance/information vs. its square root) and employ different matrix factorizations (Cholesky vs. QR decomposition), they have distinct numerical characteristics, as well as the impact of state ordering on their computational complexity. Note that, so far, we have only discussed these filters *in their basic forms*. Next, we establish their correspondence with the optimization-based methods, and discuss the extensions to these basic filters that make them equivalent to the optimization-based methods.

2.2.3 Optimization-based Methods

In order to reduce the linearization error for higher estimation accuracy, optimization-based methods (e.g., [58, 77]) employ *relinearization* of the *nonlinear* cost terms, both *within each time step* and *across multiple time steps*. In brief, at each time step, iterative optimization techniques are first used to compute the state estimates as the minimum of the nonlinear cost function corresponding to all available information in the current sliding window. Then, the

marginalization process absorbs only the necessary portion of all available information into the prior, while the remaining (nonlinear) cost terms can be reused and relinearized at subsequent time steps. In what follows, we discuss these two steps of the optimization-based methods in detail, and show how these operations for the relinearization can be carried out equivalently for the filtering-based methods.

Nonlinear Optimization

At each time step, the state estimates are computed using all cost terms within the current sliding window, including the prior in (2.2), the process model in (2.8), and the measurements in (2.14), which lead to the following nonlinear least-squares cost to be minimized:

$$\begin{aligned}\mathcal{C} &= \mathcal{C}_p + \mathcal{C}_u + \mathcal{C}_z \\ &= \|\mathbf{J}_0\mathbf{x} - \mathbf{r}_0\|_{\mathbf{I}}^2 + \|\mathbf{x}_\nu - \mathbf{f}(\mathbf{x},\mathbf{u})\|_{\mathbf{W}}^2 + \|\mathbf{z} - \mathbf{h}(\mathbf{x})\|_{\mathbf{\Sigma}}^2\end{aligned} \tag{2.55}$$

To solve this problem, one popular approach is to use the Gauss-Newton method [87] that iteratively relinearizes these nonlinear least-squares terms. Specifically, per iteration, the cost function $\mathcal{C}$ in (2.55) is approximated by its corresponding *linearized* terms around some state estimate $\hat{\mathbf{x}}$ [see (2.2), (2.10), and (2.16)]:

$$\mathcal{C} \simeq \|\mathbf{J}_0\widetilde{\mathbf{x}} - \mathbf{r}_0'\|_{\mathbf{I}}^2 + \| \begin{bmatrix}\mathbf{\Phi} & -\mathbf{I}\end{bmatrix} \begin{bmatrix}\widetilde{\mathbf{x}} \\ \widetilde{\mathbf{x}}_\nu\end{bmatrix} - \mathbf{r}_u\|_{\mathbf{W}}^2 + \|\mathbf{H}\widetilde{\mathbf{x}} - \mathbf{r}\|_{\mathbf{\Sigma}}^2 \tag{2.56}$$

where $\widetilde{\mathbf{x}} = \mathbf{x} - \hat{\mathbf{x}}$ is the corresponding error state, and $\mathbf{r}_0' = \mathbf{r}_0 - \mathbf{J}_0\hat{\mathbf{x}}$. Then, this linear least-squares problem in (2.56) is typically solved in the *regular information/inverse form*, i.e., by forming the corresponding normal equation with the information (Hessian) matrix $\mathbf{A}^\oplus$:

$$\mathbf{A}^\oplus \Delta\mathbf{x} = \mathbf{b}^\oplus \tag{2.57}$$

$$\mathbf{A}^\oplus = \begin{bmatrix}\mathbf{A}_0 + \mathbf{\Phi}^T\mathbf{W}^{-1}\mathbf{\Phi} & -\mathbf{\Phi}^T\mathbf{W}^{-1} \\ -\mathbf{W}^{-1}\mathbf{\Phi} & \mathbf{W}^{-1}\end{bmatrix} + \mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{H} \tag{2.58}$$

$$\mathbf{b}^\oplus = \begin{bmatrix}\mathbf{J}_0^T\mathbf{r}_0' \\ \mathbf{0}\end{bmatrix} + \begin{bmatrix}\mathbf{\Phi}^T\mathbf{W}^{-1}\mathbf{r}_u \\ -\mathbf{W}^{-1}\mathbf{r}_u\end{bmatrix} + \mathbf{H}^T\mathbf{\Sigma}^{-1}\mathbf{r} \tag{2.59}$$

followed by a Cholesky factorization of the Hessian matrix $\mathbf{A}^{\oplus}$ to obtain the state update $\Delta\mathbf{x}$. Finally, this process is repeated multiple times until certain convergence criterion is satisfied.

To establish its correspondence with the process of the filtering methods, if we compare (2.57)-(2.59) to the EIF's equations [see Table 2.1 and (2.21)-(2.24)], then it is obvious that the operations here are just a composition of the two steps, state augmentation and update, of the EIF. Therefore, given the equivalence between the EIF and the other three filter forms in Section 2.2.2, we conclude that, the nonlinear optimization process (with the Gauss-Newton algorithm) *per iteration* of the optimization-based methods, is equivalent to the combination of the state augmentation and update steps of the filtering methods. This result is expected as both methods eventually solve the same underlying linearized problem in (2.56).

Furthermore, the optimization-based methods iterate this process to obtain the optimal solution for the nonlinear problem, while the filters in their basic forms as in Table 2.1 effectively performs only one iteration of this process. The change of linearization points between iterations of the nonlinear optimization also leads to the existence of the prior and process residuals, i.e., the first two terms of $\mathbf{b}^{\oplus}$ in (2.59). Extensions of the filters, however, to incorporate multiple iterations, as well as the handling of these residual terms due to relinearization, are straightforward and have been developed in the literature. Specifically, the exact same equations as in (2.57)-(2.59) hold for the EIF [46, 82], while for the EKF it becomes the iterated EKF (IEKF) [16, 68]. These are summarized in the following remark:

Remark 5: The iterative optimization that relinearizes the nonlinear cost terms *within the sliding window at each time step*, can be carried out *equivalently* for the filters, in both covariance and inverse forms.

Lastly, it is worth noting that, besides the basic form of the nonlinear optimization as presented here in (2.55)-(2.59), other common variations also exist, such as using robustified cost functions instead of least squares [41, 87], or employing alternative nonlinear-optimization solvers based on line-search or trust-region methods (e.g, Levenberg-Marquardt instead of Gauss-Newton algorithm) [8, 13]. These extensions, however, can also be realized under the filtering framework.

Marginalization

As old states are removed out of the window, optimization-based methods employ the same marginalization procedure as in the filtering methods to absorb information into the prior. The

equations are identical to that of the EIF, i.e., through the Schur complement as in (2.34). The main difference, however, is that *only a portion* of all available information is used for marginalization, i.e., only the cost terms that involve the states to be marginalized (e.g., the oldest pose in the window), while the remaining nonlinear terms can be reused and relinearized again at subsequent time steps. As a result, all the nonlinear cost terms in (2.55) in the current optimization window can be categorized into two types:

- **State-and-information (SI)** cost terms: They are used in the marginalization step to generate a new prior term consisting of *both state and information* estimates. Hence, after the marginalization step, these SI terms are completely absorbed into the prior with their corresponding linearization point fixed, and cannot be reused or relinearized at any subsequent time step.

- **State-only (SO)** cost terms: They are not involved in the marginalization process, and hence do not contribute to the prior's information estimates, but are used in the nonlinear optimization step (together with the SI terms) to obtain more accurate estimates for the *state only* (as the new linearization point). Hence, after the marginalization step, these SO terms are not absorbed into the prior, and can be reused and relinearized again at multiple subsequent time steps.

These two types of measurement processing represent the preference between computational efficiency vs. estimation accuracy. Specifically, since the information of the SI-type cost terms is efficiently contained in the prior, it avoids reprocessing and hence saves computational cost. This, however, comes at the cost of early fixation of linearization points for these SI terms, leading to suboptimal solutions for the nonlinear optimization problem in (2.55). In contrast, since the information of the SO-type cost terms is not contained in the prior and can be reused in the optimization problems at multiple subsequent time steps, it enables further relinearization for these terms in order to reduce the linearization errors through smoothing, and hence results in better solutions for the nonlinear optimization problem in (2.55). The reprocessing of these SO-type terms, however, leads to a higher computational cost.

Optimization-based methods typically minimize the employment of the SI-type processing at every time step, i.e, only the cost terms that involve the marginalized states are used in the marginalization, while all other terms are kept and processed as the SO-type (e.g., [58,77]). This way, they maximize the accuracy of their solutions to the nonlinear optimization problem. On

the other hand, the filters in their basic forms, as in Table 2.1, use all the available information in marginalization for efficient processing, i.e., all cost terms are treated as the SI-type and absorbed into the prior, and hence cannot be relinearized at subsequent time steps. Extensions, however, to include the SO-type processing equivalently in the filtering framework have been developed in the literature. Specifically, the exact same marginalization process (i.e., using the Schur complement operation), but with only the SI-type terms, holds for the EIF [46, 82]. As for the EKF, these extensions are more tricky, since in this case, the prior information matrix resulting from the marginalization can be singular, and hence the corresponding covariance matrix cannot be represented in a numerically stable manner. Nevertheless, it can be done equivalently as described in the iterative Kalman smoother (IKS) [57], by decomposing the prior term into a covariance for a subset of the prior states and a set of linearized constraints. These are summarized in the following remark:

Remark 6: The marginalization scheme of the optimization-based methods, that absorbs only a portion of all available cost terms (SI-type) into the prior while enabling relinearization of the remaining nonlinear cost terms (SO-type) *across multiple time steps*, can be carried out *equivalently* for the filters, in both covariance and inverse forms.

Finally, we would like to point out the possibility of executing the optimization-based methods in their equivalent square-root forms:

Remark 7: Similarly as in the case of the filtering methods, computations in the optimization-based methods can be done equivalently in the *square-root* form (e.g., [51, 52]): One simply needs to replace the corresponding linear systems in each optimization iteration and the marginalization process with the corresponding square-root ones, as that of the SR-EIF.

In summary, filtering and optimization-based methods are just two approaches of solving the same problem as in (2.55), and they are fundamentally equivalent. As compared to the filters *in their basic forms*, the optimization-based methods perform relinearization of the nonlinear cost terms, both within each time step and across multiple time steps. On the other hand, the filters can also be extended to incorporate these relinearization operations, both in the (regular or square-root) covariance and inverse forms, and in this case become exactly equivalent to those optimization-based methods. When compared, the basic filtering methods process each cost term only once, and hence are more efficient but have larger linearization errors, thus are less accurate in general, due to the lack of relinearization. In contrast, the optimization-based methods reprocess each cost term multiple times, and hence can achieve the optimal nonlinear

solution, thus are more accurate, but at a higher processing cost. Therefore, the choice in between these two extremes (instead of between these two approaches) should be made based on the trade-off between the accuracy and efficiency for the specific estimation task of the VIO-SLAM system.

2.2.4 Applications to VIO-SLAM

So far we have discussed the optimal forms of both filtering and optimization-based methods. In practice, when applied to the VIO-SLAM problem, it is common to include certain approximations in order to reduce the computational cost of these methods. One such approximation is to assume some previously-estimated states as perfectly known, e.g., previous poses or mapped features during relocalization (e.g., [54, 71, 77]). This improves efficiency, but at the cost of inconsistent estimates and hence less accuracy [27]. Another popular practice in keyframe-based approaches is to discard information from nonkeyframes during the marginalization step, either directly the visual measurements (e.g., [58, 77]), or feature correspondences (e.g., [63, 74]), in order to maintain the sparsity of the Hessian matrix. Hence, this loss of information trades estimation accuracy for computational efficiency. Regardless, we would like to point out that, all these extra approximations can be incorporated easily in both filtering and optimization-based methods, and do not affect their equivalence mentioned earlier.

When designing an estimator for VIO-SLAM, one can choose either the filtering or the optimization-based framework, since they are fundamentally equivalent. And under either framework, in order to define a particular estimator, several key factors need to be determined. These factors have been briefly mentioned earlier and are summarized here:

Prior vs. Relinearization

As explained earlier, at each time step, every measurement can be processed either as the SI-type or the SO-type: The SI-type cost terms are absorbed into the prior term at the current time step, which become the prior for the next time step. This is more efficient, but with potentially larger linearization errors; Meanwhile, the SO-type cost terms are used only when computing the state estimates and are not absorbed into the prior, thus can be reprocessed and relinearized at multiple time steps, which leads to more accurate estimates but at a higher cost. Therefore, the choice of a measurement cost term being processed as either the SI-type or the SO-type is a

distinctive factor that determines between *efficiency* and *accuracy* for different estimators. For example, in the VIO-SLAM literature, some methods employ only the SI-type processing for all measurements at each time step (e.g., [42, 61, 69]) in order to achieve high efficiency, while others enable the SO-type processing (e.g., [46, 57, 58, 77, 82]) for improved accuracy.

Optimization Window Horizon

Under the sliding-window scheme, depending on the size and horizon of the window, the estimator can scale from a single state filtering (e.g., pose tracking) to the full batch/incremental smoothing (e.g., the optimal full SLAM solution). For example, in the VIO-SLAM literature, there exist VIO methods that consider only a constant-sized window of recent states (e.g., poses and features) to serve as an efficient tracking frontend (e.g., [15, 32, 58, 69, 77]), as well as global-adjustment methods that optimize over the entire state history to serve as an accurate loop-closing backend (e.g., [51, 56, 71, 74, 84]).

Square-Root vs. Regular Forms

As presented earlier, an estimator (filtering or optimization-based), can be realized in either the regular (i.e., covariance or information) form, or equivalently its square-root form. As for the VIO-SLAM problem, estimators in the covariance or information form require *double-precision* numerical representation and arithmetic, due to the ill-conditioning of the covariance/information matrix in practice (i.e., condition number $\geq 10^9$), otherwise the numerical errors can easily become the dominant error source affecting estimation accuracy, or even cause the estimator to diverge. Meanwhile, to achieve the same numerical accuracy and stability, the square-root alternatives only require *single-precision* representation. Hence, implementations using the square-root methods may lead to significant speedups as compared to their regular-form counterparts. This is especially true when operating on mobile devices, as most modern smart phones and tablets are equipped with the ARM Neon processor that allows for up to 4-time speed acceleration for 32-bit floating-point operations.

2.2.5 Our Proposed VIO Estimator

Based on the three key factors in Section 2.2.4, we define the following distinctive characteristics of our proposed VIO estimator, the square-root inverse sliding-window filter (SR-ISWF).

First, for the underlying estimation framework, we choose to employ the *square-root* inverse form, hence the name, so as to enable *single-precision* arithmetic and operations for improved speed. We choose the inverse form, rather than the covariance form, for its simplicity in performing relinearizations as explained earlier. To the best our knowledge, this is the first VIO algorithm that allows for single-precision implementation.

As for the optimization window's horizon, we choose to employ a constant-sized sliding window of recent poses (as well as the features observed from them), which is a common practice in the VIO literature. This is to ensure *constant computational and memory cost* as time goes by, which is a basic requirement of a VIO algorithm. As a result, our proposed algorithm can be used as a fast while locally accurate tracking frontend in a multithread full VI-SLAM system.

Moreover, similarly to the optimization-based methods, our proposed SR-ISWF algorithm employs *relinearization* of the nonlinear cost terms, both within each time step by iterating the update procedure, and across multiple time steps by using the SO-type measurement processing in addition to the SI-type. In order to balance between accuracy and efficiency, however, we choose to relinearize *only a selective portion* of all available cost terms. This way, in terms of the relinearization behavior, our estimator is a mixture of the filtering (in basic forms) and the optimization-based methods. Our particular information management and processing scheme, as presented later in Section 2.3.4, is another major technical contribution of this chapter.

Finally, our SR-ISWF algorithm does not employ any aforementioned approximation, such as assuming previous states as perfectly known, or discarding information during marginalization. This, however, leads to a dense prior information factor matrix, and will slow down the algorithm when the size of the state vector becomes too large, e.g., when a large number of SLAM features are estimated. To overcome this limitation, as explained later in Section 2.3.4, we employ both SLAM and MSCKF type processing for the feature measurements, which allows for utilizing as much visual information as available while bounding the size of the state vector at the same time. As a result, our SR-ISWF algorithm achieves high tracking accuracy with great computational efficiency.

In what follows, Section 2.3 describes in detail our proposed visual-inertial information management and processing scheme that balances between accuracy and efficiency. Then, in Section 2.4, we show how to use these selected information terms to obtain the state and information estimates efficiently under the square-root inverse formulation, by exploiting the

underlying problem structures.

2.3 SR-ISWF: VIO Problem Formulation and Information Management

In correspondence to the general problem formulation in Section 2.2.1, in this section, we first present details of the particular visual-inertial states, process and measurement models, and their corresponding cost terms, used in our proposed SR-ISWF VIO algorithm. Then, we focus on the management and processing scheme of these visual-inertial states and information, so as to achieve estimation accuracy and computational efficiency at the same time.

2.3.1 State Vector

At each time step k, the following state vector is estimated:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_S^T & \mathbf{x}_C^T & \mathbf{x}_P^T & \mathbf{x}_E^T \end{bmatrix}^T \tag{2.60}$$

The first component $\mathbf{x}_S$ is the state of the currently-observed SLAM features, with $\mathbf{x}_S = \begin{bmatrix} \mathbf{x}_{f_1}^T & \cdots & \mathbf{x}_{f_N}^T \end{bmatrix}^T$, where $\mathbf{x}_{f_j}$, for $j = 1, \cdots, N$, denotes the state of the corresponding point feature $\mathbf{f}_j$. Here the feature state is represented using the inverse-depth parametrization (i.e., a 3×1 vector with homogeneous coordinates and inverse depth) with respect to its first observing camera pose within the current sliding window, for improved numerical accuracy [23]. These SLAM features are maintained as a local map for tracking the poses of the device accurately in the short term (see Section 2.3.4).

The state $\mathbf{x}_C$ consists of a sliding window of recent poses, from $\mathbf{x}_{C_{k-M+1}}$ to $\mathbf{x}_{C_k}$, where $\mathbf{x}_{C_i}$, for $i = k - M + 1, \cdots, k$, denotes the state of the cloned[3] IMU pose at time step i, and M is the size of the sliding window. Each cloned pose state $\mathbf{x}_{C_i}$ is defined as:

$$\mathbf{x}_{C_i} = \begin{bmatrix} {}^{I_i}\mathbf{q}_G^T & {}^G\mathbf{p}_{I_i}^T & t_{d_i} \end{bmatrix}^T \tag{2.61}$$

where ${}^{I_i}\mathbf{q}_G$ is the quaternion representation of the orientation of the global frame $\{G\}$ in the

[3] We refer to the same stochastic cloning as in [69], for maintaining past IMU poses in a sliding window estimator.

IMU's frame of reference $\{I_i\}$ at time step i, and ${}^{G}\mathbf{p}_{I_i}$ is the position of $\{I_i\}$ in $\{G\}$. Additionally as in [39], in order to handle the issue of time synchronization between the sensors on commercial-grade mobile devices, we also include in the cloned pose state the unknown and varying IMU-camera time offset[4] t_{d_i} at time step i.

The parameter state $\mathbf{x}_P$ consists of the constant but unknown parameters as:

$$\mathbf{x}_P = \begin{bmatrix} {}^{I}\mathbf{q}_C^T & {}^{I}\mathbf{p}_C^T & t_r \end{bmatrix}^T \tag{2.62}$$

where ${}^{I}\mathbf{q}_C$ is the quaternion representation of the orientation of the camera frame $\{C\}$ in the IMU frame $\{I\}$, and ${}^{I}\mathbf{p}_C$ is the position of $\{C\}$ in $\{I\}$. Additionally as in [39], we estimate the rolling-shutter time t_r of the camera (i.e., the readout time of each image, which is constant) for mobile devices. Moreover, this parameter state can also contain other quantities of interest, such as IMU and/or camera intrinsics as in [62].

Finally, the states necessary for modeling the IMU process, besides the pose, are kept in the IMU extra state $\mathbf{x}_E$ as:

$$\mathbf{x}_E = \begin{bmatrix} \mathbf{b}_{g_k}^T & {}^{G}\mathbf{v}_{I_k}^T & \mathbf{b}_{a_k}^T \end{bmatrix}^T \tag{2.63}$$

where $\mathbf{b}_{g_k}$ and $\mathbf{b}_{a_k}$ correspond to the gyroscope and accelerometer biases, respectively, and ${}^{G}\mathbf{v}_{I_k}$ is the velocity of $\{I_k\}$ in $\{G\}$, at the current time step k.

The evolution of the state vector across the different steps of our SR-ISWF algorithm, and the ordering of these states within the state vector that affects the computational cost, are described later in Section 2.4.

2.3.2 Inertial Measurement Model and Cost Terms

Integration of the IMU data provides motion constraints between two consecutive cloned IMU poses. This corresponds to the process model and its cost terms in Section 2.2.1. Here we follow the standard approach for modeling the IMU process as in [42, 86]. Specifically, the IMU measures the device's rotational velocity and linear acceleration contaminated by white Gaussian noises and time-varying biases. The gyroscope and accelerometer measurements,

[4] Similarly as in [39], we consider the general case where the IMU and camera operate at different frequencies, and their time stamps are reported in different clocks and are not accurate. Thus, the unknown time difference between the two sensors is time-varying and needs to be estimated per image.

$\boldsymbol{\omega}_m(t)$ and $\mathbf{a}_m(t)$, are:

$$\begin{aligned}\boldsymbol{\omega}_m(t) &= {}^I\boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_g(t) \\ \mathbf{a}_m(t) &= \mathbf{C}({}^I\mathbf{q}_G(t))({}^G\mathbf{a}(t) - {}^G\mathbf{g}) + \mathbf{b}_a(t) + \mathbf{n}_a(t)\end{aligned} \tag{2.64}$$

where the noise terms, $\mathbf{n}_g(t)$ and $\mathbf{n}_a(t)$, are modeled as zero-mean white Gaussian noise processes, $\mathbf{C}(\mathbf{q})$ is the rotation matrix corresponding to $\mathbf{q}$, and the gravitational acceleration ${}^G\mathbf{g}$ is considered a known deterministic constant. The device's rotational velocity ${}^I\boldsymbol{\omega}(t)$ and linear acceleration ${}^G\mathbf{a}(t)$, from (2.64), can be used to relate consecutive IMU poses through the continuous-time system equations:

$$\begin{aligned}{}^I\dot{\mathbf{q}}_G(t) &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m(t) - \mathbf{b}_g(t) - \mathbf{n}_g(t)){}^I\mathbf{q}_G(t) \\ {}^G\dot{\mathbf{v}}_I(t) &= \mathbf{C}^T({}^I\mathbf{q}_G(t))(\mathbf{a}_m(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)) + {}^G\mathbf{g} \\ {}^G\dot{\mathbf{p}}_I(t) &= {}^G\mathbf{v}_I(t), \;\; \dot{\mathbf{b}}_g(t) = \mathbf{n}_{wg}(t), \;\; \dot{\mathbf{b}}_a(t) = \mathbf{n}_{wa}(t) \\ {}^I\dot{\mathbf{q}}_C(t) &= \mathbf{0}, \;\; {}^I\dot{\mathbf{p}}_C(t) = \mathbf{0}, \;\; \dot{t}_d(t) = n_{td}(t), \;\; \dot{t}_r(t) = 0\end{aligned} \tag{2.65}$$

where the biases and the IMU-camera time offset are modeled as random walks driven by zero-mean white Gaussian noise processes $\mathbf{n}_{wg}(t)$, $\mathbf{n}_{wa}(t)$, and $n_{td}(t)$, respectively.

Given the inertial measurements $\mathbf{u}_{k-1:k} = \begin{bmatrix}\boldsymbol{\omega}_{m_{k-1:k}}^T & \mathbf{a}_{m_{k-1:k}}^T\end{bmatrix}^T$ within the time interval $\begin{bmatrix}t_{k-1}, \; t_k\end{bmatrix}$, analytical integration of (2.65) is used to determine the discrete-time system equations, which imposes a constraint between the consecutive cloned IMU states $\mathbf{x}_{I_{k-1}}$ and $\mathbf{x}_{I_k}$ (after linearizing with respect to the noise terms) as:

$$\mathbf{x}_{I_k} = \mathbf{f}(\mathbf{x}_{I_{k-1}}, \mathbf{u}_{k-1:k}) + \mathbf{w}_k \tag{2.66}$$

and the corresponding nonlinear cost term:

$$\mathcal{C}_u(\mathbf{x}_{I_{k-1}}, \mathbf{x}_{I_k}) = \|\mathbf{x}_{I_k} - \mathbf{f}(\mathbf{x}_{I_{k-1}}, \mathbf{u}_{k-1:k})\|^2_{\mathbf{W}_k} \tag{2.67}$$

where the cloned IMU state $\mathbf{x}_{I_k} = \begin{bmatrix}\mathbf{x}_{C_k}^T & \mathbf{x}_{E_k}^T\end{bmatrix}^T$ [see (2.61) and (2.63)], and $\mathbf{w}_k$ is the discrete-time zero-mean white Gaussian noise with covariance $\mathbf{W}_k$. Linearizing (2.66), around the state

estimates $\hat{\mathbf{x}}_{I_{k-1}}$ and $\hat{\mathbf{x}}_{I_k}$, results in the following linearized inertial constraint and cost term:

$$\widetilde{\mathbf{x}}_{I_k} \simeq \mathbf{\Phi}_{k,k-1}\widetilde{\mathbf{x}}_{I_{k-1}} - \mathbf{r}_{u_k} + \mathbf{w}_k \tag{2.68}$$

$$\mathcal{C}_u(\widetilde{\mathbf{x}}_{I_{k-1}}, \widetilde{\mathbf{x}}_{I_k}) \simeq \| \begin{bmatrix} \mathbf{\Phi}_{k,k-1} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_{I_{k-1}} \\ \widetilde{\mathbf{x}}_{I_k} \end{bmatrix} - \mathbf{r}_{u_k} \|^2_{\mathbf{W}_k} \tag{2.69}$$

where the error state $\widetilde{\mathbf{x}}_{I_k} = \mathbf{x}_{I_k} - \hat{\mathbf{x}}_{I_k}$, $\mathbf{\Phi}_{k,k-1}$ is the state transition (Jacobian) matrix, and $\mathbf{r}_{u_k} = \hat{\mathbf{x}}_{I_k} - \mathbf{f}(\hat{\mathbf{x}}_{I_{k-1}}, \mathbf{u}_{k-1:k})$ is the residual (i.e., IMU propagation error). Details of the IMU integration, including the analytical expressions for $\mathbf{\Phi}_{k,k-1}$ and $\mathbf{W}_k$, can be found in [42, 86]. Moreover, as in [42], the state transition matrix $\mathbf{\Phi}_{k,k-1}$ is modified to satisfy the observability constraints for improving the estimation consistency.

2.3.3 Visual Measurement Model and Cost Terms

Camera images provide observations to features from the camera poses. This corresponds to the measurement model and its cost terms in Section 2.2.1. Specifically, we extract point features and track them across images within the current window as visual measurements to be processed. When working with commercial-grade mobile devices (e.g., cell phones), the images suffer from the rolling-shutter effect, where the image pixel rows are read sequentially in time, so each row has a different actual observation time. In addition, there exists a time-varying offset between the IMU and the camera's time stamps, i.e., the two clocks are not synchronized and their time stamps are inaccurate. To handle these issues, we employ the interpolation model in [39], where each camera pose is interpolated using its two closest cloned poses.

In general, a point-feature measurement model can be written as:

$$\mathbf{z}_i^j = \mathbf{h}({}^{C_{i+t}}\mathbf{p}_{f_j}) + \mathbf{n}_i^j = \mathbf{h}(\mathbf{x}_{f_j}, \mathbf{x}_F) + \mathbf{n}_i^j \tag{2.70}$$

where $\mathbf{z}_i^j$ is a point-feature measurement in pixel coordinates, ${}^{C_{i+t}}\mathbf{p}_{f_j}$ is the feature's position expressed in the camera frame of reference at the exact acquisition time instant of this pixel measurement, $\mathbf{n}_i^j$ is zero-mean white Gaussian noise with covariance $\sigma^2\mathbf{I}_2$, and $\mathbf{I}_2$ is the 2×2 identity matrix. As in (2.70), a visual measurement is a nonlinear function of the states involved: i) The feature's state $\mathbf{x}_{f_j}$, which is expressed with respect to its first observing camera pose within the current window; ii) Some poses in the cloned pose states $\mathbf{x}_C$ and the parameter state

$\mathbf{x}_P$. And for convenience, we have defined:

$$\mathbf{x}_F = \begin{bmatrix} \mathbf{x}_C^T & \mathbf{x}_P^T & \mathbf{x}_E^T \end{bmatrix}^T \tag{2.71}$$

which is a subset of the entire state vector in (2.60), comprising all states except the SLAM features. Thus, each feature measurement contributes a nonlinear cost term:

$$\mathcal{C}_z(\mathbf{x}_{f_j}, \mathbf{x}_F) = \|\mathbf{z}_i^j - \mathbf{h}(\mathbf{x}_{f_j}, \mathbf{x}_F)\|^2_{\sigma^2\mathbf{I}_2} \tag{2.72}$$

Linearizing (2.70), around the state estimates $\hat{\mathbf{x}}_{f_j}$ (initially from point-feature triangulation) and $\hat{\mathbf{x}}_F$, results in the following linearized visual constraint and cost term:

$$\mathbf{r}_{z,i}^j \simeq \mathbf{H}_{f,i}^j \widetilde{\mathbf{x}}_{f_j} + \mathbf{H}_{F,i}^j \widetilde{\mathbf{x}}_F + \mathbf{n}_i^j \tag{2.73}$$

$$\mathcal{C}_z(\widetilde{\mathbf{x}}_{f_j}, \widetilde{\mathbf{x}}_F) \simeq \|\mathbf{H}_{f,i}^j \widetilde{\mathbf{x}}_{f_j} + \mathbf{H}_{F,i}^j \widetilde{\mathbf{x}}_F - \mathbf{r}_{z,i}^j\|^2_{\sigma^2\mathbf{I}_2} \tag{2.74}$$

where $\mathbf{H}_{f,i}^j$ and $\mathbf{H}_{F,i}^j$ are the corresponding feature and pose Jacobians, respectively, and $\mathbf{r}_{z,i}^j$ is the measurement residual (i.e., reprojection error). Note that $\mathbf{H}_{F,i}^j$ has nonzero columns corresponding to only the involved poses of this measurement and the parameter states. Details of the specific camera measurement model that we adopt, including the analytical expressions for the Jacobians and their modifications to satisfy the observability constraints, can be found in [39].

As a feature is tracked across multiple images, we define the **feature track** as the collection of all the visual measurements to this feature from all its observing poses. Combining together all its ℓ_j observations within the current window, from (2.74), each feature track contributes a visual cost term:

$$\begin{aligned} \mathcal{C}_{z^0}(\widetilde{\mathbf{x}}_{f_j}, \widetilde{\mathbf{x}}_F) &= \sum_{i=k-M+1}^{k} \mathcal{C}_z(\widetilde{\mathbf{x}}_{f_j}, \widetilde{\mathbf{x}}_F) \\ &= \|\mathbf{H}_f^j \widetilde{\mathbf{x}}_{f_j} + \mathbf{H}_F^j \widetilde{\mathbf{x}}_F - \mathbf{r}_z^j\|^2_{\sigma^2\mathbf{I}_{2\ell_j}} \end{aligned} \tag{2.75}$$

where $\mathbf{H}_f^j$, $\mathbf{H}_F^j$, and $\mathbf{r}_z^j$ are the stacked feature Jacobian, pose Jacobian, and residual, respectively. Note that due to the nonzero pattern of each $\mathbf{H}_{F,i}^j$, $\mathbf{H}_F^j$ has a specific structure that we will utilize later. To further reveal the information portions of the cost term $\mathcal{C}_{z^0}$, consider a $2\ell_j \times 2\ell_j$

orthogonal matrix $\mathbf{Q}_L^j$, partitioned as $\mathbf{Q}_L^j = \begin{bmatrix}\mathbf{Q}_{L_1}^j & \mathbf{Q}_{L_2}^j\end{bmatrix}$, where the 3 columns of $\mathbf{Q}_{L_1}^j$ span the column space of $\mathbf{H}_f^j$, while the $2\ell_j - 3$ columns of $\mathbf{Q}_{L_2}^j$ span its left nullspace [69]. If we apply the *orthogonal* transformation, defined by $\mathbf{Q}_L^{j^T}$, to the Jacobians and residual in (2.75) as:

$$\begin{aligned}
&\mathbf{Q}_{L_1}^{j^T}\mathbf{H}_f^j = \mathbf{R}_f^j, && \mathbf{Q}_{L_1}^{j^T}\mathbf{H}_F^j = \mathbf{F}_1^j, && \mathbf{Q}_{L_1}^{j^T}\mathbf{r}_z^j = \boldsymbol{\zeta}_1^j \\
&\mathbf{Q}_{L_2}^{j^T}\mathbf{H}_f^j = \mathbf{0}, && \mathbf{Q}_{L_2}^{j^T}\mathbf{H}_F^j = \mathbf{F}_2^j, && \mathbf{Q}_{L_2}^{j^T}\mathbf{r}_z^j = \boldsymbol{\zeta}_2^j
\end{aligned} \tag{2.76}$$

which we name the **left-nullspace** (**LNS**) **transformation**, then, each feature track's cost term $\mathcal{C}_{z^0}$ in (2.75) becomes:

$$\begin{aligned}
&\mathcal{C}_{z^0}(\widetilde{\mathbf{x}}_{f_j}, \widetilde{\mathbf{x}}_F) = \|\mathbf{H}_f^j\widetilde{\mathbf{x}}_{f_j} + \mathbf{H}_F^j\widetilde{\mathbf{x}}_F - \mathbf{r}_z^j\|^2_{\sigma^2\mathbf{I}_{2\ell_j}} \\
&= \|\mathbf{Q}_L^{j^T}(\mathbf{H}_f^j\widetilde{\mathbf{x}}_{f_j} + \mathbf{H}_F^j\widetilde{\mathbf{x}}_F - \mathbf{r}_z^j)\|^2_{\sigma^2\mathbf{I}_{2\ell_j}} \\
&= \underbrace{\|\mathbf{R}_f^j\widetilde{\mathbf{x}}_{f_j} + \mathbf{F}_1^j\widetilde{\mathbf{x}}_F - \boldsymbol{\zeta}_1^j\|^2_{\sigma^2\mathbf{I}_3}}_{\mathcal{C}_{z^1}(\widetilde{\mathbf{x}}_{f_j}, \widetilde{\mathbf{x}}_F)} + \underbrace{\|\mathbf{F}_2^j\widetilde{\mathbf{x}}_F - \boldsymbol{\zeta}_2^j\|^2_{\sigma^2\mathbf{I}_{2\ell_j-3}}}_{\mathcal{C}_{z^2}(\widetilde{\mathbf{x}}_F)}
\end{aligned} \tag{2.77}$$

As a result, $\mathcal{C}_{z^0}$ is equivalently transformed into two cost terms. Since $\mathbf{R}_f^j$ is invertible, for any $\widetilde{\mathbf{x}}_F$, there always exists an $\widetilde{\mathbf{x}}_{f_j}$ that makes the first term $\mathcal{C}_{z^1}(\widetilde{\mathbf{x}}_{f_j}, \widetilde{\mathbf{x}}_F)$ zero. Therefore, minimizing $\mathcal{C}_{z^0}$ is equivalent to minimizing only the second term $\mathcal{C}_{z^2}(\widetilde{\mathbf{x}}_F)$, which corresponds to marginalizing the feature state. Thus we can conclude that, $\mathcal{C}_{z^1}(\widetilde{\mathbf{x}}_{f_j}, \widetilde{\mathbf{x}}_F)$ contains *only* information about the feature's state, while *all* the information on the cloned poses (and the parameter state) is contained in $\mathcal{C}_{z^2}(\widetilde{\mathbf{x}}_F)$, which is exactly the pose constraint used by the MSC-KF algorithm [69].

2.3.4 Visual-Inertial Information Management

Given the visual-inertial measurements and their cost terms, we need to determine the specific approach of information management, i.e., the way in which the state vector evolves and the corresponding measurement processing scheme for estimating these states.

Cloned State Management Scheme

When a new camera image arrives at time step k, the corresponding IMU frame's pose state $\mathbf{x}_{C_k}$ [see (2.61)] and extra state $\mathbf{x}_{E_k}$ [see (2.63)] are cloned and added to the state vector, if

the camera has undergone sufficient motion from the previous cloned pose, in order to provide geometry for point feature triangulation. The specific IMU frame to be cloned is chosen at the instance of the IMU data that is closest to the image in terms of their time stamps. Meanwhile, the oldest cloned pose $\mathbf{x}_{C_{k-M}}$ is marginalized, so as to maintain a *constant* size of the pose sliding window.

Inertial Information Processing Scheme

Once the new cloned IMU states are added to the state vector, we use the corresponding inertial measurements to initialize these states. This corresponds to the state augmentation step in Section 2.2.2. Specifically, we use the inertial cost term, $\mathcal{C}_u(\widetilde{\mathbf{x}}_{I_{k-1}}, \widetilde{\mathbf{x}}_{I_k})$ from (2.69), which represents the information of IMU propagation from the previous cloned IMU state $\mathbf{x}_{I_{k-1}}$ to the current new one $\mathbf{x}_{I_k}$. Similarly to most filtering-based VIO methods (e.g., [42, 61, 69]), we choose *not* to relinearize or reprocess the IMU cost term, i.e., it is processed as the SI-type and absorbed into the prior immediately, so as to reduce the computational cost. This is due to the fact that, although relinearization can be carried out efficiently by performing IMU preintegration [33], reprocessing the inertial cost terms requires to include all the IMU extra states for all the clones in the window. This will significantly increase the state vector's size and hence the computational cost. Instead, by not reprocessing the inertial cost terms, the previous IMU extra state $\mathbf{x}_{E_{k-1}}$ can be marginalized right after the cloned state augmentation step, as it is no longer needed afterwards. Meanwhile, *only* the current IMU extra state $\mathbf{x}_{E_k}$ is kept [see (2.63)] during the subsequent update step using the visual information.

Visual Information Processing Scheme

The visual information from camera observations to point features are utilized to correct the state estimates. This corresponds to the update step in Section 2.2.2. Given the large amount of visual observations (easily hundreds of them per image), however, we need to choose carefully their processing scheme, so as to achieve high estimation accuracy while maintaining great computational efficiency at the same time, especially when operating on resource-constrained platforms. To do so, we employ two types of hybrid visual processing approaches: i) SLAM and MSCKF feature processing, and ii) SI-type and SO-type measurement processing.

First, similarly as in [42, 60], our algorithm employs a hybrid feature processing scheme of

both SLAM and MSCKF features:

- **SLAM features**: As in typical SLAM approaches, these features' states $\mathbf{x}_S$ are added into the estimator's state vector [see (2.60)] and updated across time. By maintaining a map of the scene, SLAM features increase the estimation accuracy and improve the estimator's robustness, especially when viewed over many frames. In terms of the cost terms in (2.77), $\mathcal{C}_{z^1}$ is used for initializing new SLAM features, while $\mathcal{C}_{z^2}$ is exploited for update. Then at subsequent time steps, reobservations to these estimated SLAM features, i.e., $\mathcal{C}_z$ in (2.74), are used to further improve the feature estimates, as well as localizing the camera poses. Finally, when a SLAM feature is no longer observed by the newest pose in the sliding window, it is marginalized and removed from the state vector.

- **MSCKF features**: These features are processed as in the MSC-KF approach [69], hence the name,[5] where the feature states are marginalized on-the-fly to generate efficient local multi-state constraints with respect to their observing camera poses [see (2.77)]. In terms of the cost terms, this approach employs only the cost term $\mathcal{C}_{z^2}$ in (2.77) for update. As a result, MSCKF features are not mapped, i.e., the feature states are not added into the estimator's state vector for direct estimation, so as to reduce the computational cost.

Under the *sliding-window* scheme, the SLAM feature processing is in general more accurate, while the MSCKF scheme is more efficient, depending on the feature's track length. Specifically, the **track length** of a feature is defined as the number of camera observations of the entire feature track, which depends on the appearance of the feature in the camera images across time and the results of the image processing module, while is *independent* of the specific size of the sliding window. If a feature's track length is smaller than or equal to the window size, i.e., the entire feature track is contained within one epoch of the sliding window, then processing it as a SLAM or MSCKF feature will give *exactly the same* pose estimates, since in this case, as explained in Section 2.3.3 [see (2.77)], all the information on the poses from this feature track is represented as the cost term $\mathcal{C}_{z^2}$. Otherwise, if a feature track is longer than the window's size, then applying the MSCKF scheme actually cuts the entire track into *multiple* pieces of shorter tracks, where each one of these shorter tracks is processed as a *different* feature. As a result, the effective track length of a MSCKF feature can never exceed the size of the window,

[5] Throughout this chapter, we use the term "MSC-KF" to denote the overall filtering algorithm in [69], while "MSCKF features" signifies the features processed in such a manner.

and this scheme trades accuracy for efficiency, by dropping information on feature correspondences. In contrast, the SLAM processing scheme preserves the correspondence information of a feature track, by adding the feature state into the state vector and hence allowing to process subsequent reobservations to the *same* feature, and thus is more accurate and robust than the MSCKF scheme. This advantage, however, comes with a higher processing cost, as it increases the size of the state vector, as well as that of the corresponding covariance/information matrix (or its factor), especially when the matrix is dense. Therefore, to summarize, we choose to process features whose track lengths are smaller than the sliding window's size as MSCKF features for efficiency, while those with longer tracks are preferably to be processed as SLAM features for accuracy.

At this point, it is worth noting that, when processing the visual information of a feature track to initialize it as a new SLAM feature, all existing VIO-SLAM methods directly employ the cost term $\mathcal{C}_{z^0}$ in (2.75) that corresponds to the original feature measurements. Instead, in this work, we propose to use the alternative but *equivalent* cost terms after the LNS transformation, i.e., $\mathcal{C}_{z^1} + \mathcal{C}_{z^2}$ in (2.77), so that: i) The update operation of new SLAM feature initialization can be carried out in a *uniform* manner with that of the MSCKF features (i.e., both using the cost term $\mathcal{C}_{z^2}$), which results in an easier implementation of the algorithm, and more importantly, ii) the *computational savings* achieved by exploiting the Jacobian structures of the MSCKF feature tracks during the update procedure also apply to the case of new SLAM feature initialization. Details of the SLAM and MSCKF feature measurement processing in our algorithm are presented later in Section 2.4.

Moreover, besides the hybrid processing scheme with both SLAM and MSCKF features, another mechanism for balancing between accuracy and efficiency, as explained earlier in Section 2.2.4, is by choosing between the SI-type vs. SO-type measurement processing, i.e., by deciding on which measurements to be absorbed into the prior at the current time step, while others to update only the state estimates so that they can be reprocessed and relinearized at subsequent time steps. As mentioned earlier in Section 2.2.3, optimization-based methods typically maximize the usage of SO-type processing *per measurement* for accuracy, by selecting as SO-type all the measurements whose absorption into the prior can be postponed to later time steps (e.g., [58, 77]). Instead, in this work, we choose to relinearize *only a selective portion* of all these measurements, by performing a coarser selection of the processing scheme *per feature track*, where each track comprises multiple measurements to a feature. This is due to the fact

that, the MSCKF features can only be processed per track (using the transformed cost term $\mathcal{C}_{z^2}$) instead of per measurement, as well as the new SLAM feature initialization as we choose to for the reasons mentioned earlier. As a result, our measurement processing scheme trades accuracy for efficiency. Specifically, under the *sliding-window* scheme, at each time step, in order to determine the choice between the SI-type and SO-type processing for each feature track, we classify all available feature tracks in the current window into the following two categories (see Fig. 2.1):

- **Immature feature tracks**: These are the features that are observed by some cloned poses in the current window, but *not* by the *oldest* cloned pose yet. All feature tracks start as immature, as in the beginning a track only covers some newest poses of the window at these epochs. Since these tracks have not reached the tail of the window, their measurements can be reprocessed and relinearized across multiple time steps. Hence, we choose to use them as the SO-type for accuracy. After being processed by the estimator, these SO-type feature track measurements remain as available for the next time step.

- **Mature feature tracks**: These are the features that are observed by the *oldest* cloned pose in the current window. As time goes by and the estimation window slides forward in time, all immature feature tracks eventually become mature, i.e., when a feature's first observing pose becomes the oldest cloned pose of the window at a certain epoch. Since these tracks have reached the tail of the window, and the oldest pose will be marginalized at the next time step, this is the last chance to absorb their information into the prior, i.e., postponing their processing to future epochs of the sliding window will lead to loss of information. Hence, we choose to use these tracks' measurements within the current window as the SI-type for efficiency. After being processed by the estimator, these SI-type feature track measurements can no longer be used in the future, and hence are removed.

Given the two types of hybrid visual processing approaches (i.e., SLAM+MSCKF features and SI+SO-type measurement processing), in this work, we propose to combine them to create our visual information processing strategy under the sliding-window scheme. Specifically, based on our discussions so far, the key factors for determining between these processing choices are the maturity and the track length of a feature track within the current window. When a new feature track enters the sliding window and is *immature*, it is processed as the SO-type.

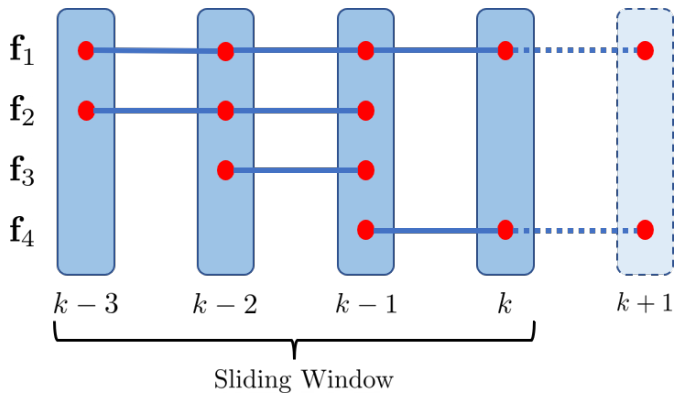


Figure 2.1: An example of feature tracks in the sliding window and our corresponding visual information processing scheme: At the current time step k, the window with size 4 contains cloned poses from time step $k-3$ to k. Within this window, the feature tracks $\mathbf{f}_1$ and $\mathbf{f}_2$ are mature, as they are observed by the oldest cloned pose at $k-3$, while $\mathbf{f}_3$ and $\mathbf{f}_4$ are immature. We choose to initialize $\mathbf{f}_1$ as a new (SI-)SLAM feature, since its track is mature and spans the entire window. Meanwhile, the other mature feature track $\mathbf{f}_2$ is processed as SI-MSCKF, since it is not observed by the newest pose at time step k. On the other hand, the immature tracks $\mathbf{f}_3$ and $\mathbf{f}_4$ are processed as SO-MSCKF. After being processed by the estimator, the SI-type feature track measurements of $\mathbf{f}_1$ and $\mathbf{f}_2$ are removed, since their information has been absorbed into the prior, while the SO-type measurements of $\mathbf{f}_3$ and $\mathbf{f}_4$ remain as available for the next time step. At the next time step $k+1$, the window slides forward to contain poses from $k-2$ to $k+1$, where $\mathbf{f}_3$ becomes mature, while $\mathbf{f}_4$ remains to be immature. Additionally, the SLAM feature $\mathbf{f}_1$ is reobserved from the newest pose at time step $k+1$, and this measurement is processed as the SI-type.

Note that for the SO-type processing, the SLAM and MSCKF feature approaches are identical regardless of the track length, since after the SO-type processing the prior will not contain any information on the feature's state, and the entire feature track can be reprocessed freely at following time steps. In fact, the only impact of processing such SO-type feature tracks is to have a better estimate (linearization point) for the state vector, by using these tracks' multi-state constraint cost terms $\mathcal{C}_{z^2}$. Hence, we use **SO-MSCKF** to denote any feature track for the SO-type processing. On the other hand, as time goes by and a feature track becomes *mature*, we choose to process it as the SI-type, and either as **SI-SLAM** (or simply denoted by **SLAM**) or **SI-MSCKF** based on its *track length* within the current window: If the track spans the entire window, then it has chance to last longer and to be reobserved, hence it is initialized as a new SLAM feature, by processing as the SI-type its entire track within the current window. Reobservations to this SLAM feature, from the newest pose of the window at subsequent time steps, are immediately processed as the SI-type for efficiency; Otherwise, if the track is shorter than the window span, i.e., the feature is not observed from the newest poses of the current window, and hence this track is unlikely to continue, it is processed as the SI-MSCKF. An example of our visual information processing scheme is illustrated in Fig. 2.1.

As compared to the sliding-window VIO methods that employ both SLAM and MSCKF features but with only the SI-type processing (e.g., [42, 60]), by adding the SO-type measurement processing, our hybrid visual information processing scheme has the following advantages in practice: i) It enables using visual observations *as soon as they become available* for obtaining a better estimate for the current states, while allowing for *delayed decision* on a feature track to be processed as either a SLAM or MSCKF feature, depending on its track length after it becomes mature. ii) By using the same information repeatedly for updating the state, it improves the estimator's *robustness* by avoiding the case of running out of visual measurements at certain epochs of the sliding window, which can happen when only the SI-type processing is employed, especially under adverse conditions (e.g., in areas with limited number of features). iii) By relinearizing the nonlinear visual cost terms across multiple time steps, it reduces the linearization error and hence improves accuracy.

Finally, in order to address the limitations on processing capabilities of resource-constrained platforms, we bound the total computational cost of each estimator run, by setting a *budget* on

the number of feature tracks that can be processed in each update, for each of the aforementioned processing types (i.e., SLAM, SI-MSCKF, and SO-MSCKF). When the number of available feature tracks is larger than the allowed budget, features are selected based on their track lengths, with higher priority given to the *longer* tracks, since they contain more information in general.

In summary, we have proposed a hybrid visual information processing scheme for sliding-window VIO. Our scheme combines the SLAM+MSCKF feature approach with the SI+SO-type measurement processing, while complying with the limiting budget for each type, in order to balance between the estimation accuracy and computational efficiency. Note that our proposed scheme is not restricted to any specific estimator, i.e., it can be used for any sparse-feature-based sliding-window VIO algorithm regardless of the estimation framework (see Section 2.2). The detailed steps for carrying out our proposed scheme are listed below:

(i) **Feature track management**: At each time step, all available feature tracks in the current window are classified into the mature and immature groups, and are arranged by their track lengths. This can be accomplished efficiently by updating the result from the previous time step, considering the evolution of the cloned poses under the sliding-window scheme.

(ii) **Current SLAM feature reobservation [$\mathbb{Z}_R$ ($\mathcal{C}_{\mathbb{Z}_R}$)]**: For each SLAM feature in the current state vector: If the feature is reobserved from the newest pose, this measurement is selected to be processed as the SI-type; Otherwise, if the feature is not reobserved, its state is set to be marginalized.

(iii) **New SLAM feature initialization [$\mathbb{Z}_S$ ($\mathcal{C}_{\mathbb{Z}_S^1} + \mathcal{C}_{\mathbb{Z}_S^2}$)]**: If the number of remaining SLAM features is smaller than the SLAM budget, new SLAM features are selected from the mature feature tracks that span the entire window. These features' states are set to be added into the state vector, and their tracks are to be processed in the SI-type SLAM manner.

(iv) **SI-MSCKF feature selection [$\mathbb{Z}_M$ ($\mathcal{C}_{\mathbb{Z}_M^2}$)]**: Feature tracks are selected to fulfill the SI-MSCKF budget, from the mature group and in the descending order of track length. These tracks are to be processed in the SI-type MSCKF manner.

(v) **SO-MSCKF feature selection [$\bar{\mathbb{Z}}_M$ ($\bar{\mathcal{C}}_{\mathbb{Z}_M^2}$)]**: Feature tracks are selected to fulfill the SO-MSCKF budget, from both the immature and mature groups and in the descending order of track length. These tracks are to be processed in the SO-type MSCKF manner.

(vi) **Estimator run**: All selected measurements (cost terms) are processed by the estimator (see Section 2.4).

(vii) **Feature measurement removal**: All measurements processed as the SI-type are removed, while the ones as the SO-type remain as available for future uses.

2.4 SR-ISWF: Estimation Algorithm

Given the selected visual-inertial measurements (cost terms) and their corresponding processing scheme as described in Section 2.3.4, in this section, we present our SR-ISWF estimation algorithm that performs the numerical operations to compute an estimate of the state vector [see (2.60)], as well as its corresponding information factor, with special attention to specific problem structures that are exploited for an efficient implementation.

Specifically, at each time step k, the input of the algorithm consists of the prior and the selected visual-inertial measurements, as well as the current state estimate to be used as the linearization point. First, the visual-inertial measurements to be processed are described in detail in Section 2.3.4, and their corresponding cost terms are summarized here: The inertial cost term $\mathcal{C}_u$ [see (2.69)] represents the information arising from the IMU measurements $\mathbf{u}_{k-1:k}$, the visual cost terms $\mathcal{C}_{\mathbb{Z}_R}$ [see (2.74)] from all the reobservation measurements $\mathbb{Z}_R$ to the current SLAM features, $\mathcal{C}_{\mathbb{Z}_S^1} + \mathcal{C}_{\mathbb{Z}_S^2}$ [see (2.77)] from all the new SLAM feature (to be initialized) track measurements $\mathbb{Z}_S$, and $\mathcal{C}_{\mathbb{Z}_M^2}$ and $\bar{\mathcal{C}}_{\mathbb{Z}_M^2}$ from all the SI-MSCKF and SO-MSCKF feature track measurements $\mathbb{Z}_M$ and $\bar{\mathbb{Z}}_M$, respectively. Finally, the prior information, which is generated by our estimator from the previous time step $k-1$, is represented by the prior cost term:

$$\mathcal{C}_p(\widetilde{\mathbf{x}}_{k-1}) = \|\mathbf{R}_{k-1}\widetilde{\mathbf{x}}_{k-1} - \mathbf{r}_{k-1}\|^2 \tag{2.78}$$

where $\|\cdot\|$ denotes the standard vector 2-norm, $\mathbf{R}_{k-1}$ and $\mathbf{r}_{k-1}$ are the prior information square-root factor matrix (i.e., the *upper-triangular* Cholesky factor of the prior information/Hessian matrix) and the prior residual vector, respectively. $\widetilde{\mathbf{x}}_{k-1} = \mathbf{x}_{k-1} - \hat{\mathbf{x}}_{k-1}$ is the error state

corresponding to the current state estimate (linearization point) $\hat{\mathbf{x}}_{k-1}$. Note that in (2.78), the prior residual $\mathbf{r}_{k-1} \neq \mathbf{0}$ due to the processing of the SO-type cost terms (see Section 2.4.11).

The main objective of our estimation algorithm is to compute the new state estimate, by minimizing the total cost function $\mathcal{C}_k^\star$ that contains all the information from the prior, inertial, and visual cost terms:

$$\mathcal{C}_k^\star = \mathcal{C}_p + \mathcal{C}_u + \mathcal{C}_\mathbb{Z} \tag{2.79}$$

where $\mathcal{C}_\mathbb{Z}$ represents the collection of the nonlinear cost terms $\mathcal{C}_z$ in (2.72) from all aforementioned visual measurements $\mathbb{Z}$. In order to solve this nonlinear least-squares problem, our estimator employs the Gauss-Newton optimization, where each iteration minimizes the linearized least-squares cost function:

$$\mathcal{C}_k^\star = \underbrace{\mathcal{C}_p + \mathcal{C}_u + \mathcal{C}_{\mathbb{Z}_R} + \mathcal{C}_{\mathbb{Z}_S^1} + \mathcal{C}_{\mathbb{Z}_S^2} + \mathcal{C}_{\mathbb{Z}_M^2}}_{\mathcal{C}_k^\oplus} + \bar{\mathcal{C}}_{\mathbb{Z}_M^2} \tag{2.80}$$

which is solved in the square-root inverse form, following the basic numerical procedure of the SR-EIF (see Table 2.1), so as to enable single-precision implementation for improved speed. Note that, as explained in Section 2.2.3, we use the total cost function $\mathcal{C}_k^\star$, including *both* the SI-type and SO-type terms, to compute the new state estimate (as the linearization point), while the posterior cost function $\mathcal{C}_k^\oplus$, containing *only* the SI-type terms, to generate the new prior for the next time step.

Same as in the SR-EIF (see Table 2.1), the main numerical operation of our algorithm's update procedure, where the visual cost terms are processed, is a QR factorization as in (2.48). A naive approach would stack all the measurement Jacobians together with the prior information factor, and perform a large QR factorization, considering all the involved matrices as dense. Instead, in our algorithm, we split this entire QR process into multiple steps, where each step executes a portion of this process, according to the different measurement types in (2.80). Although mathematically equivalent, this enables us to better utilize the underlying *problem structures* of each measurement type, i.e., we explicitly analyze and exploit the *nonzero patterns* of the Jacobian and information factor matrices involved at each step, and implement specialized QR functions for them, so as to achieve computational savings. Additionally, as explained in Section 2.2.2, the *state ordering* is an important factor that affects the computational cost of the square-root estimators. We choose the specific ordering as in (2.60), which defines the column

ordering of the (upper-triangular) information factor and all Jacobian matrices, so that it allows for exploiting the nonzero patterns of these matrices to achieve a lower cost for our QR factorizations. Finally, as explained later (see Section 2.4.6 and 2.4.8), some of these most important structural findings and our approach for handling them, can be applied to many other popular VIO estimators, besides our SR-ISWF, to achieve significant computational savings. This is another major contribution of this chapter.

In what follows, we describe in detail all the steps of our SR-ISWF estimation algorithm. We show the effect of each step on the cost function to be minimized in (2.80), the evolution of the state vector from $\mathbf{x}_{k-1}$ to $\mathbf{x}_k$, and the specific problem structures with our state ordering. An overview of our SR-ISWF estimation algorithm is shown in Algorithm 1.

2.4.1 Cloned State Augmentation

The estimator run is triggered when a new IMU clone becomes available at time step k. As described in Section 2.3.4, the current state vector $\mathbf{x}_{k-1}$, from the previous time step $k-1$, is augmented with the new cloned IMU states $\mathbf{x}_{I_k}$ [see (2.66)] as:

$$\mathbf{x}_k^{\alpha} = \begin{bmatrix} \mathbf{x}_{k-1}^T & \mathbf{x}_{I_k}^T \end{bmatrix}^T \tag{2.81}$$

where $\mathbf{x}_{k-1}$ follows the form in (2.60). In terms of the cost function [see (2.80)], the prior term $\mathcal{C}_p$ in (2.78) is combined with the inertial cost term $\mathcal{C}_u$ in (2.69), which is obtained by integrating the IMU measurements $\mathbf{u}_{k-1:k}$, to yield:

$$\begin{aligned} &\mathcal{C}_{\alpha}(\widetilde{\mathbf{x}}_k^{\alpha}) = \mathcal{C}_p(\widetilde{\mathbf{x}}_{k-1}) + \mathcal{C}_u(\widetilde{\mathbf{x}}_{I_{k-1}}, \widetilde{\mathbf{x}}_{I_k}) \\ &= \|\mathbf{R}_{k-1}\widetilde{\mathbf{x}}_{k-1} - \mathbf{r}_{k-1}\|^2 + \|\begin{bmatrix} \mathbf{\Phi}_{k,k-1} & -\mathbf{I} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_{I_{k-1}} \\ \widetilde{\mathbf{x}}_{I_k} \end{bmatrix} - \mathbf{r}_{u_k}\|_{\mathbf{W}_k}^2 \\ &= \|\mathbf{R}_{\alpha}\widetilde{\mathbf{x}}_k^{\alpha} - \mathbf{r}_{\alpha}\|^2 \end{aligned} \tag{2.82}$$

where

$$\begin{aligned} \mathbf{R}_{\alpha} = \begin{bmatrix} \mathbf{R}_{k-1} & \mathbf{0} \\ \mathbf{W}_k^{-\frac{1}{2}}\breve{\mathbf{\Phi}}_{k,k-1} & -\mathbf{W}_k^{-\frac{1}{2}} \end{bmatrix}, \quad \mathbf{r}_{\alpha} = \begin{bmatrix} \mathbf{r}_{k-1} \\ \mathbf{W}_k^{-\frac{1}{2}}\mathbf{r}_{u_k} \end{bmatrix} \\ \breve{\mathbf{\Phi}}_{k,k-1} = \begin{bmatrix} \mathbf{0}\cdots\mathbf{0} & \mathbf{\Phi}_{k,k-1}^{(C)} & \mathbf{0}\cdots\mathbf{0} & \mathbf{\Phi}_{k,k-1}^{(E)} \end{bmatrix} \end{aligned} \tag{2.83}$$

Algorithm 1 SR-ISWF Estimation Algorithm

Input:

- Current state estimate $\hat{\mathbf{x}}_{k-1}$
- Prior info sqrt factor $\mathbf{R}_{k-1}$ and prior residual $\mathbf{r}_{k-1}$
- Inertial measurements $\mathbf{u}_{k-1:k}$
- Current SLAM feature reobservations $\mathbb{Z}_R$
- New SLAM feature track measurements $\mathbb{Z}_S$
- SI-MSCKF feature track measurements $\mathbb{Z}_M$
- SO-MSCKF feature track measurements $\bar{\mathbb{Z}}_M$

Estimator Run:

- Cloned state augmentation [(2.69) (2.81) (2.83)]
- SLAM feature propagation [(2.84) (2.85) (2.86)]
- Marginalization [(2.88) (2.90)]
- Covariance factor recovery [(2.95)]
- Update (iterated):
  - Linearization: Jacobians and residuals [(2.74)]
  - SI Update: Current SLAM feature reobservations [(2.98) (2.99)]
  - Left-nullspace transformation [(2.75) (2.101)]
  - SI Update: New SLAM feature initialization [(2.104) (2.106) (2.107)]
  - SI Update: New SLAM and SI-MSCKF pose constraints [(2.110) (2.111) (2.113)]
  - SO Update: SO-MSCKF pose constraints [(2.115)]
  - Computing new state estimate [(2.117) (2.118)]
- Computing new prior term [(2.120)]

Output:

- New state estimate $\hat{\mathbf{x}}_k$
- New prior info sqrt factor $\mathbf{R}_k$ and prior residual $\mathbf{r}_k$

where $\mathbf{\Phi}_{k,k-1}^{(C)}$ and $\mathbf{\Phi}_{k,k-1}^{(E)}$ are block columns of the Jacobian $\mathbf{\Phi}_{k,k-1}$ with respect to the cloned IMU pose and extra states [see (2.61) and (2.63)], respectively. Thus, same as in the SR-EIF [see (2.47)], this cloned state augmentation step simply requires augmenting the prior information factor with the IMU propagation Jacobians, as well as for the prior residual as in (2.83).

2.4.2 SLAM Feature Propagation

Due to the inverse-depth feature parametrization used in our algorithm for improved numerical accuracy, the SLAM features' states in the state vector $\mathbf{x}_{k-1}$ in (2.81) are expressed with respect to their first observing camera pose within the current sliding window, which is simply the *oldest* cloned pose $\mathbf{x}_{C_{k-M}}$ of the window, as we choose to initialize new SLAM features from only the *mature* feature tracks (see Section 2.3.4). Under the sliding-window scheme, however, this "anchor" pose is the tail of the window and is about to be marginalized (see Section 2.4.3). Meanwhile, the SLAM feature tracks can last longer than the window's horizon, and in order to process reobservation measurements to these SLAM features (see Section 2.4.5) in a consistent manner, their states need to be re-anchored to another cloned pose within the current window. Hence, at every time step k, prior to the marginalization process, our algorithm shifts the SLAM features' anchor pose from $\mathbf{x}_{C_{k-M}}$ to its next cloned pose $\mathbf{x}_{C_{k-M+1}}$, which will be the new tail of the window after marginalization. Specifically, this corresponds to replacing the current state vector $\mathbf{x}_k^\alpha$ by a new one $\mathbf{x}_k^\beta$, where the SLAM features' states [see (2.81) and (2.60)] are propagated from ${}^{C_{k-M}}\mathbf{x}_{f_j}$ to ${}^{C_{k-M+1}}\mathbf{x}_{f_j}$, for $j = 1, \cdots, N$. To do so, we employ the geometric relation between the two feature states, through the two involved cloned poses and the IMU-camera extrinsics, which can be described as a nonlinear *deterministic* constraint:

$$ {}^{C_{k-M+1}}\mathbf{x}_{f_j} = \mathbf{g}({}^{C_{k-M}}\mathbf{x}_{f_j}, \mathbf{x}_{C_{k-M}}, \mathbf{x}_{C_{k-M+1}}, \mathbf{x}_P) \tag{2.84} $$

where $\mathbf{x}_P$ is the parameter state in (2.62). Given the current state estimate $\hat{\mathbf{x}}_k^\alpha$, from (2.84) we compute the propagated feature estimate ${}^{C_{k-M+1}}\hat{\mathbf{x}}_{f_j}$. In terms of the cost function, it needs to be transformed to be expressed in the new error state vector $\widetilde{\mathbf{x}}_k^\beta$. To do this, we first linearize (2.84) around the current state estimate, and multiply it from the left by the inverse of the Jacobian

with respect to ${}^{C_{k-M}}\widetilde{\mathbf{x}}_{f_j}$, to obtain:

$$\begin{aligned} {}^{C_{k-M}}\widetilde{\mathbf{x}}_{f_j} \simeq \mathbf{G}_f^{j\,{}^{C_{k-M+1}}}\widetilde{\mathbf{x}}_{f_j} + \mathbf{G}_{C_{k-M}}^{j}\widetilde{\mathbf{x}}_{C_{k-M}} \\ + \mathbf{G}_{C_{k-M+1}}^{j}\widetilde{\mathbf{x}}_{C_{k-M+1}} + \mathbf{G}_P^j\widetilde{\mathbf{x}}_P \end{aligned} \tag{2.85}$$

where the $\mathbf{G}$ matrices are the corresponding Jacobians. Then, substituting (2.85) into $\mathcal{C}_\alpha(\widetilde{\mathbf{x}}_k^\alpha)$ in (2.82), for all the SLAM features $j = 1, \cdots, N$, we get the new cost function $\mathcal{C}_\beta(\widetilde{\mathbf{x}}_k^\beta)$ as:

$$\mathcal{C}_\alpha(\widetilde{\mathbf{x}}_k^\alpha) = \|\mathbf{R}_\alpha\widetilde{\mathbf{x}}_k^\alpha - \mathbf{r}_\alpha\|^2 = \|\mathbf{R}_\beta\widetilde{\mathbf{x}}_k^\beta - \mathbf{r}_\beta\|^2 = \mathcal{C}_\beta(\widetilde{\mathbf{x}}_k^\beta) \tag{2.86}$$

where the new factor $\mathbf{R}_\beta$ is obtained by modifying the original factor $\mathbf{R}_\alpha$ using the $\mathbf{G}$ matrices in (2.85), while the residual remains unchanged, i.e., $\mathbf{r}_\beta = \mathbf{r}_\alpha$.

Note that the factor modification in (2.86) can be carried out very efficiently: Only four small blocks, whose columns correspond to the involved states in (2.85), at the top rows of the factor need to be modified, thanks to the (almost) *upper-triangular* structure of $\mathbf{R}_\alpha$, with the columns corresponding to the SLAM feature states to the *left* due to our *state ordering* in (2.60). Moreover, we only need to propagate the SLAM features to be kept at the current time step, while the others are about to be marginalized (see Section 2.4.3) and thus their anchor pose does not matter.

As a result of this SLAM feature propagation step, in our algorithm, all currently estimated SLAM features are always anchored to the *oldest* cloned pose of the sliding window at any time step.

2.4.3 Marginalization

As described in Section 2.3.4, in order to maintain constant complexity, at every time step k, our algorithm marginalizes the following states within the current sliding window: The oldest cloned pose $\mathbf{x}_{C_{k-M}}$, the previous IMU extra state $\mathbf{x}_{E_{k-1}}$, and the SLAM features $\mathbf{x}_S^\mu$ that are no longer tracked. If we define the state vector consisting of all these states to be marginalized as:

$$\mathbf{x}_k^\mu = \begin{bmatrix} \mathbf{x}_S^{\mu^T} & \mathbf{x}_{C_{k-M}}^T & \mathbf{x}_{E_{k-1}}^T \end{bmatrix}^T \tag{2.87}$$

and $\mathbf{x}_k^\rho$, following the form in (2.60), denotes the remaining states after removing $\mathbf{x}_k^\mu$ from the current state vector $\mathbf{x}_k^\beta$, then we have:

$$\mathbf{\Pi}_M \mathbf{x}_k^\beta = \begin{bmatrix} \mathbf{x}_k^{\mu^T} & \mathbf{x}_k^{\rho^T} \end{bmatrix}^T \tag{2.88}$$

where $\mathbf{\Pi}_M$ is a permutation matrix. Consequently, from (2.86) and (2.88), the columns of the factor $\mathbf{R}_\beta$ are permuted correspondingly, since:

$$\begin{aligned} \mathcal{C}_\beta(\widetilde{\mathbf{x}}_k^\beta) &= \|\mathbf{R}_\beta \widetilde{\mathbf{x}}_k^\beta - \mathbf{r}_\beta\|^2 = \|\mathbf{R}_\beta \mathbf{\Pi}_M^T \begin{bmatrix} \widetilde{\mathbf{x}}_k^\mu \\ \widetilde{\mathbf{x}}_k^\rho \end{bmatrix} - \mathbf{r}_\beta\|^2 \\ &= \| \begin{bmatrix} \mathbf{R}_{\beta_{:\mu}} & \mathbf{R}_{\beta_{:\rho}} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_k^\mu \\ \widetilde{\mathbf{x}}_k^\rho \end{bmatrix} - \mathbf{r}_\beta\|^2 = \mathcal{C}_\beta(\widetilde{\mathbf{x}}_k^\mu, \widetilde{\mathbf{x}}_k^\rho) \end{aligned} \tag{2.89}$$

where the column-permuted factor is partitioned as $\mathbf{R}_\beta \mathbf{\Pi}_M^T = \begin{bmatrix} \mathbf{R}_{\beta_{:\mu}} & \mathbf{R}_{\beta_{:\rho}} \end{bmatrix}$. In terms of the cost function, marginalization corresponds to removing the error state $\widetilde{\mathbf{x}}_k^\mu$ from the cost function $\mathcal{C}_\beta(\widetilde{\mathbf{x}}_k^\mu, \widetilde{\mathbf{x}}_k^\rho)$, by minimizing with respect to it. To achieve this, we perform the following QR factorization:

$$\begin{bmatrix} \mathbf{R}_{\beta_{:\mu}} & \mathbf{R}_{\beta_{:\rho}} & \mathbf{r}_\beta \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q}_M \begin{bmatrix} \mathbf{R}_{\mu\mu} & \mathbf{R}_{\mu\rho} & \mathbf{r}_\mu \\ \mathbf{0} & \mathbf{R}_{\rho\rho} & \mathbf{r}_\rho \end{bmatrix} \tag{2.90}$$

Substituting (2.90) into (2.89) yields:

$$\begin{aligned} \mathcal{C}_\beta(\widetilde{\mathbf{x}}_k^\mu, \widetilde{\mathbf{x}}_k^\rho) &= \left\| \begin{bmatrix} \mathbf{R}_{\mu\mu} & \mathbf{R}_{\mu\rho} \\ \mathbf{0} & \mathbf{R}_{\rho\rho} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_k^\mu \\ \widetilde{\mathbf{x}}_k^\rho \end{bmatrix} - \begin{bmatrix} \mathbf{r}_\mu \\ \mathbf{r}_\rho \end{bmatrix} \right\|^2 \\ &= \|\mathbf{R}_{\mu\mu} \widetilde{\mathbf{x}}_k^\mu + \mathbf{R}_{\mu\rho} \widetilde{\mathbf{x}}_k^\rho - \mathbf{r}_\mu\|^2 + \|\mathbf{R}_{\rho\rho} \widetilde{\mathbf{x}}_k^\rho - \mathbf{r}_\rho\|^2 \end{aligned} \tag{2.91}$$

Since $\mathbf{R}_{\mu\mu}$ is invertible, for any $\widetilde{\mathbf{x}}_k^\rho$, there always exists an $\widetilde{\mathbf{x}}_k^\mu$ that makes the first cost term in (2.91) zero. Hence, from (2.91), the cost function after marginalization, $\mathcal{C}_\rho(\widetilde{\mathbf{x}}_k^\rho)$, is simply the second cost term:

$$\mathcal{C}_\rho(\widetilde{\mathbf{x}}_k^\rho) = \min_{\widetilde{\mathbf{x}}_k^\mu} \mathcal{C}_\beta(\widetilde{\mathbf{x}}_k^\mu, \widetilde{\mathbf{x}}_k^\rho) = \|\mathbf{R}_{\rho\rho} \widetilde{\mathbf{x}}_k^\rho - \mathbf{r}_\rho\|^2 \tag{2.92}$$

Thus, same as in the SR-EIF [see (2.53)-(2.54)], this marginalization step requires permuting the factor column-wise as in (2.89), and then performing a QR factorization on the permuted

factor as in (2.90), where the factor is augmented with the residual so that $\mathbf{r}_\rho$ is obtained in-place (i.e., $\mathbf{Q}_M$ does not need to be formed explicitly [37]). During this QR process, we take advantage of the (almost) upper-triangular structure of the factor $\mathbf{R}_\beta$ before the permutation to achieve computational savings.

As a result of this marginalization step, only the remaining state vector $\mathbf{x}_k^\rho$ is kept, and it follows the form and *ordering* in (2.60). Additionally, the corresponding information factor $\mathbf{R}_{\rho\rho}$, whose columns have the same state ordering, is upper-triangular.

2.4.4 Covariance Factor Recovery

In order to improve the accuracy and robustness of our estimator, we perform the Mahalanobis distance test on all visual measurement cost terms for outlier rejection. It is in general more effective than the reprojection error check, since it takes into account the uncertainties of the current state estimates that are used to compute the measurement residual (i.e., reprojection error). Specifically, the Mahalanobis distance γ is defined as:

$$\gamma = \|\mathbf{r}_z\|_{\mathbf{S}}^2 = \mathbf{r}_z^T \mathbf{S}^{-1} \mathbf{r}_z, \quad \mathbf{S} = \mathbf{H}\mathbf{P}_{\rho\rho}\mathbf{H}^T + \sigma^2\mathbf{I} \tag{2.93}$$

where the measurement Jacobian $\mathbf{H}$ and residual $\mathbf{r}_z$ are as in (2.74) and (2.77). The residual covariance $\mathbf{S}$ requires the covariance $\mathbf{P}_{\rho\rho}$ for the current state estimate, which is not explicitly maintained in our square-root inverse estimator. Instead, we have the information factor $\mathbf{R}_{\rho\rho}$ [see (2.92)], and they are related as:

$$\mathbf{P}_{\rho\rho} = (\mathbf{R}_{\rho\rho}^T \mathbf{R}_{\rho\rho})^{-1} = \mathbf{R}_{\rho\rho}^{-1} \mathbf{R}_{\rho\rho}^{-T} = \mathbf{U}_{\rho\rho}\mathbf{U}_{\rho\rho}^T \tag{2.94}$$

where

$$\mathbf{U}_{\rho\rho} = \mathbf{R}_{\rho\rho}^{-1} \tag{2.95}$$

is the upper-triangular covariance factor. Hence, prior to the subsequent update procedure, our algorithm first recovers this covariance factor $\mathbf{U}_{\rho\rho}$ as in (2.95). Then, from (2.93) and (2.94), $\mathbf{S}$ is computed as:

$$\mathbf{S} = (\mathbf{H}\mathbf{U}_{\rho\rho})(\mathbf{H}\mathbf{U}_{\rho\rho})^T + \sigma^2\mathbf{I} \tag{2.96}$$

This way we need not compute explicitly the covariance matrix $\mathbf{P}_{\rho\rho}$, which is numerically unstable since its condition number is the square of that of the factor $\mathbf{U}_{\rho\rho}$ or $\mathbf{R}_{\rho\rho}$.

While the Mahalanobis distance test for outlier rejection is commonly used in EKF-based VIO methods (e.g., [15, 62, 69]), it is typically absent in estimators in the information/inverse form where a large Hessian matrix is kept (e.g., [58, 77, 82]), due to the highly expensive operation of recovering the *dense* covaraince matrix/factor, despite the sparsity of the Hessian matrix. In contrast, in our algorithm, the size of the state vector and the corresponding information factor is much *smaller*, because of the MSCKF-type feature processing and the marginalization of all previous IMU extra states within the current window. Moreover, in (2.95), since $\mathbf{R}_{\rho\rho}$ is *upper-triangular*, its inverse can be computed efficiently by solving with the identity matrix using backward substitution. As a result, the Mahalanobis distance test is enabled in our algorithm by recovering the covariance factor with minimal overhead, which leads to reliable outlier rejection and hence higher estimation accuracy and robustness.

2.4.5 SI Update: Current SLAM Feature Reobservations

From this point on, our estimation algorithm enters the update procedure, where all the input visual measurements are processed: i) Reobservations ($\mathbb{Z}_R$), from the newest cloned pose of the window, to the *current* SLAM features whose states are kept in the state vector; ii) Feature track measurements ($\mathbb{Z}_S$, $\mathbb{Z}_M$, and $\bar{\mathbb{Z}}_M$), from multiple cloned poses within the window, to some *new* features. Since these two types of visual measurements have different Jacobian structures, they are processed separately in our algorithm. We start with the update step using the reobservations, followed by the ones using the new feature tracks.

Each SLAM feature reobservation corresponds to a *single* visual measurement $\mathbf{z}_k^j$ as in (2.70), from the newest cloned pose $\mathbf{x}_{C_k}$, to a current SLAM feature $\mathbf{f}_j$ in the state vector $\mathbf{x}_k^\rho$. Here the SLAM feature's state ${}^{C_{k-M+1}}\mathbf{x}_{f_j}$ is expressed with respect to the oldest cloned pose $\mathbf{x}_{C_{k-M+1}}$ of the window, due to the SLAM feature propagation step earlier (see Section 2.4.2). After linearization, each measurement's Jacobian and residual in (2.74) are first used to perform the Mahalanobis distance test [see (2.93) and (2.96)]. Then, all the inlier measurements are employed for update. In terms of the cost function [see (2.80)], this update step corresponds to combining the current cost function $\mathcal{C}_\rho$ in (2.92) with $\mathcal{C}_{\mathbb{Z}_R}$, which consists of visual cost terms

Figure 2.2: An example of the Jacobian structure in the SLAM feature reobservation update.

$\mathcal{C}_z$ in (2.74) from these reobservation measurements, to obtain:

$$\begin{aligned}
&\mathcal{C}_{sr}(\widetilde{\mathbf{x}}_k^\rho) = \mathcal{C}_\rho(\widetilde{\mathbf{x}}_k^\rho) + \mathcal{C}_{\mathbb{Z}_R}(\widetilde{\mathbf{x}}_S, \widetilde{\mathbf{x}}_F) \\
&= \|\mathbf{R}_{\rho\rho}\widetilde{\mathbf{x}}_k^\rho - \mathbf{r}_\rho\|^2 + \sum_{j=1}^{N_{sr}} \|\mathbf{H}_{f,k}^j \widetilde{\mathbf{x}}_{f_j} + \mathbf{H}_{F,k}^j \widetilde{\mathbf{x}}_F - \mathbf{r}_{z,k}^j\|_{\sigma^2\mathbf{I}}^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}_{\rho\rho} \\ \frac{1}{\sigma}\mathbf{H}_{sr} \end{bmatrix} \widetilde{\mathbf{x}}_k^\rho - \begin{bmatrix} \mathbf{r}_\rho \\ \frac{1}{\sigma}\mathbf{r}_{z_{sr}} \end{bmatrix} \right\|^2
\end{aligned} \tag{2.97}$$

where

$$\begin{aligned}
\mathbf{H}_{sr} = \begin{bmatrix} \cdots & \mathbf{H}_{sr}^{j^T} & \cdots \end{bmatrix}^T, \quad \mathbf{r}_{z_{sr}} = \begin{bmatrix} \cdots & \mathbf{r}_{z,k}^{j^T} & \cdots \end{bmatrix}^T \\
\mathbf{H}_{sr}^j = \begin{bmatrix} \mathbf{0}\cdots\mathbf{0} & \mathbf{H}_{f,k}^j & \mathbf{0}\cdots\mathbf{0} & \mathbf{H}_{F,k}^j \end{bmatrix}
\end{aligned} \tag{2.98}$$

with N_{sr} the total number of inlier reobservation measurements. If we perform the following QR factorization:

$$\begin{bmatrix} \mathbf{R}_{\rho\rho} & \mathbf{r}_\rho \\ \frac{1}{\sigma}\mathbf{H}_{sr} & \frac{1}{\sigma}\mathbf{r}_{z_{sr}} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q}_{sr} \begin{bmatrix} \mathbf{R}_{sr} & \mathbf{r}_{sr} \\ \mathbf{0} & \mathbf{e}_{sr} \end{bmatrix} \tag{2.99}$$

then, after dropping the constant term $\|\mathbf{e}_{sr}\|^2$, the cost function $\mathcal{C}_{sr}$ in (2.97) after the reobservation update becomes:

$$\mathcal{C}_{sr}(\widetilde{\mathbf{x}}_k^\rho) = \|\mathbf{R}_{sr}\widetilde{\mathbf{x}}_k^\rho - \mathbf{r}_{sr}\|^2 \tag{2.100}$$

Thus, same as in the SR-EIF [see (2.48)], this update step requires performing a QR factorization as in (2.99), where $\mathbf{r}_{sr}$ is obtained in-place through the QR process.

To carry out the QR factorization in (2.99), we exploit the fact that $\mathbf{R}_{\rho\rho}$ is *upper-triangular*, and that the Jacobian matrix $\mathbf{H}_{sr}$ in (2.98) has a specific *structure*, as shown in Fig. 2.2. Hence, with our state ordering in (2.60) where the Jacobian's columns corresponding to the SLAM feature states are to the left, we first permute the rows of the stacked matrix according to their

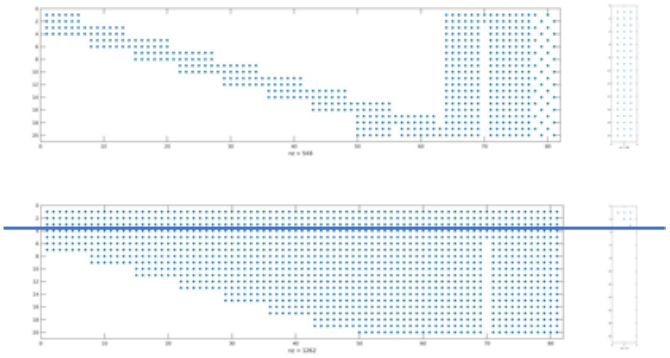


Figure 2.3: An example of the Jacobian structures before and after the left-nullspace transformation.

leading entry positions. As a result, the permuted matrix is almost upper-triangular, with a large portion of zeros at the bottom-left corner, which allows an efficient QR factorization.

2.4.6 Left-Nullspace Transformation

Now we use the new feature tracks to perform updates, where measurements of the new SLAM features ($\mathbb{Z}_S$), the SI-MSCKF features ($\mathbb{Z}_M$), and the SO-MSCKF features ($\bar{\mathbb{Z}}_M$), are all processed in a *uniform* manner. Specifically, as described in Section 2.3.3, each feature track $\mathbf{f}_j$ contributes a linearized cost term $\mathcal{C}_{z^0}$ as in (2.75), which after the LNS transformation in (2.76), becomes equivalently $\mathcal{C}_{z^1} + \mathcal{C}_{z^2}$ as in (2.77). To realize this LNS transformation, there exist infinitely many $\mathbf{Q}_{L_2}^j$ that span the LNS of the feature Jacobian $\mathbf{H}_f^j$. An arbitrary choice, however, would in general make the transformed pose Jacobian $\mathbf{F}_2^j$ in (2.76) *dense*, despite the structure of the original pose Jacobian $\mathbf{H}_F^j$ [see (2.75) and Fig. 2.3]. To this end, we propose a *particular* $\mathbf{Q}_{L_2}^j$ (implicitly), so that the resulting $\mathbf{F}_2^j$ has a favorable structure, which will be exploited in the update steps afterwards to achieve computational savings (see Section 2.4.8).

Specifically, we carry out the LNS transformation in (2.76) through the following QR factorization:

$$\begin{bmatrix}\mathbf{H}_f^j & \mathbf{H}_F^j & \mathbf{r}_z^j\end{bmatrix} \overset{\text{QR}}{=} \underbrace{\begin{bmatrix}\mathbf{Q}_{L_1}^j & \mathbf{Q}_{L_2}^j\end{bmatrix}}_{\mathbf{Q}_L^j} \begin{bmatrix}\mathbf{R}_f^j & \mathbf{F}_1^j & \boldsymbol{\zeta}_1^j \\ \mathbf{0} & \mathbf{F}_2^j & \boldsymbol{\zeta}_2^j\end{bmatrix} \tag{2.101}$$

where we employ Givens rotations [37] to triangularize $\mathbf{H}_f^j$ into $\mathbf{R}_f^j$, with the following *specific order* of elimination: The elements in $\mathbf{H}_f^j$ are zeroed out column-wise from left to right, and

within each column from bottom to top. Meanwhile, this sequence of Givens rotations are applied to $\mathbf{H}_F^j$ and $\mathbf{r}_z^j$, so that $\mathbf{F}^j$ and $\boldsymbol{\zeta}^j$ are obtained *in-place* through this QR process. Note that $\mathbf{Q}_L^j$ is never formed explicitly. Rather, we are implicitly defining our particular $\mathbf{Q}_L^j$ as the underlying orthogonal matrix of this Givens-based QR factorization in (2.101).

As a result of this specialized LNS transformation, in (2.101), the transformed feature Jacobian $\mathbf{R}_f^j$ is square and *upper-triangular*. More importantly, based on the structure of the original pose Jacobian $\mathbf{H}_F^j$ and our specific order of the Givens process, it can be verified that the resulting pose Jacobian $\mathbf{F}_2^j$ has a *block-upper-triangular structure*, as shown in Fig. 2.3. By doing this, the computational savings achieved are twofold: i) During the LNS transformation, each Givens rotation needs to be applied to only a subset of the columns of $\mathbf{H}_F^j$, that have nonzero elements in the two rows undergoing the current Givens rotation. ii) After the LNS transformation, the resulting block-upper-triangular structure of $\mathbf{F}_2^j$ is utilized afterwards, when many such pose Jacobians from all the feature tracks are stacked together for update (see Section 2.4.8). Moreover, the structure information of these pose Jacobians can be obtained *analytically*, i.e., no numerical search is required to find out the exact locations of their nonzero elements. First, the structure of $\mathbf{H}_F^j$ is determined by the feature track's observation pattern within the current window. Then, given this information and our Givens process, the row and column indices of the *structural corners* (marked by red dots in Fig. 2.3) are easily derived, which define uniquely the exact block-upper-triangular structure of the resulting $\mathbf{F}_2^j$.

To ensure this block-upper-triangular structure of $\mathbf{F}_2^j$, the *dense Jacobian columns* in $\mathbf{H}_F^j$ must be placed at the *rightmost* position (see Fig. 2.3). This leads to our choice for a portion of the state ordering in (2.60). Specifically, these dense columns correspond to two sets of states: i) The parameter state $\mathbf{x}_P$, which we arrange next to the cloned pose states $\mathbf{x}_C$; ii) The cloned pose state of a feature's first observing pose within the window, which varies among different features. Therefore, there exists no ordering for $\mathbf{x}_C$ that will generate the desired Jacobian structure for *all* the feature tracks. In practice, under our visual processing scheme (see Section 2.3.4), the majority of selected feature tracks are *mature* (including all of $\mathbb{Z}_S$ and $\mathbb{Z}_M$, as well as some of $\bar{\mathbb{Z}}_M$), whose first observing pose is the *oldest* cloned pose $\mathbf{x}_{C_{k-M+1}}$. Hence, at each time step k, we reorder $\mathbf{x}_{C_{k-M+1}}$ to the end of all other cloned pose states, i.e., in (2.60) we have:

$$\mathbf{x}_C = \begin{bmatrix} \mathbf{x}_{C_{k-M+2}}^T & \cdots & \mathbf{x}_{C_k}^T & \mathbf{x}_{C_{k-M+1}}^T \end{bmatrix}^T \tag{2.102}$$

This reordering is carried out during the marginalization step within the remaining state vector $\mathbf{x}_k^\rho$ [see (2.88)]. As a result, for mature feature tracks, the block-upper-triangular structure of $\mathbf{F}_2^j$ is guaranteed (see Fig. 2.3). On the other hand, as an *immature* feature track may start from anywhere in the window, there will be a dense block column corresponding to its first observing pose in the front part of $\mathbf{F}_2^j$, and thus we consider this entire Jacobian block as dense. Note that this is a drawback in terms of efficiency, due to the inverse-depth feature parametrization with respect to *local* frames, in exchange for numerical accuracy.[6]

As a result of this LNS transformation step, for each new feature track, we split its information into two cost terms as in (2.77), where $\mathcal{C}_{z^1}$ contains *only* information about the new feature's state, while $\mathcal{C}_{z^2}$ holds *all* the information on the cloned poses and the parameter states. Then, the Mahalanobis distance test [see (2.93) and (2.96)] is performed using only the $\mathcal{C}_{z^2}$ part to determine inlier feature tracks. Next, in our algorithm, the $\mathcal{C}_{z^1}$ terms from all the inlier new SLAM feature tracks (i.e., $\mathcal{C}_{\mathbb{Z}_S^1}$) are used to initialize these new SLAM features (see Section 2.4.7), while those from the MSCKF feature tracks are discarded as these features' states are not to be estimated. Additionally, the $\mathcal{C}_{z^2}$ terms from all the inlier feature tracks (i.e., $\mathcal{C}_{\mathbb{Z}_S^2}$, $\mathcal{C}_{\mathbb{Z}_M^2}$, and $\bar{\mathcal{C}}_{\mathbb{Z}_M^2}$) are employed to perform updates (see Section 2.4.8 and 2.4.9).

2.4.7 SI Update: New SLAM Feature Initialization

In this step, states of the inlier new SLAM features, denoted by $\mathbf{x}_S^\nu$, are added into the current state vector $\mathbf{x}_k^\rho$. Specifically, $\mathbf{x}_S^\nu = \begin{bmatrix}\mathbf{x}_{f_1}^{\nu^T} & \cdots & \mathbf{x}_{f_{N_{ns}}}^{\nu^T}\end{bmatrix}^T$, where each feature's state $\mathbf{x}_{f_j}^\nu$ is expressed with respect to the oldest cloned pose $\mathbf{x}_{C_{k-M+1}}$ as its track is mature, and N_{ns} is the total number of these new SLAM features. Recall that, from (2.60) and (2.71), the current state vector $\mathbf{x}_k^\rho$ is partitioned as:

$$\mathbf{x}_k^\rho = \begin{bmatrix}\mathbf{x}_S^T & \mathbf{x}_F^T\end{bmatrix}^T \tag{2.103}$$

where $\mathbf{x}_S$ consists of the current (existing) SLAM features' states. Now we append $\mathbf{x}_S^\nu$ to the *end* of $\mathbf{x}_S$ as:

$$\mathbf{x}_k = \begin{bmatrix}\mathbf{x}_S^T & \mathbf{x}_S^{\nu^T} & \mathbf{x}_F^T\end{bmatrix}^T \tag{2.104}$$

[6] Throughout this chapter, we address the problem in the challenging case where both local feature parametrization and interpolation model are employed, which result in more complicated problem structures. Modifications to our approach, to handle cases without these complications, are straightforward.

which forms the final state vector $\mathbf{x}_k$ for time step k. In terms of the cost function [see (2.80)], the current cost $\mathcal{C}_{sr}$ in (2.100) is combined with $\mathcal{C}_{\mathbb{Z}_S^1}$, which consists of cost terms $\mathcal{C}_{z^1}$ in (2.77) from all inlier new SLAM feature tracks after the LNS transformation, to obtain:

$$\begin{aligned}
&\mathcal{C}_{ns}(\widetilde{\mathbf{x}}_k) = \mathcal{C}_{sr}(\widetilde{\mathbf{x}}_k^{\rho}) + \mathcal{C}_{\mathbb{Z}_S^1}(\widetilde{\mathbf{x}}_S^{\nu}, \widetilde{\mathbf{x}}_F) \\
&= \|\mathbf{R}_{sr}\widetilde{\mathbf{x}}_k^{\rho} - \mathbf{r}_{sr}\|^2 + \sum_{j=1}^{N_{ns}} \|\mathbf{R}_{f_S}^{j}\widetilde{\mathbf{x}}_{f_j}^{\nu} + \mathbf{F}_{1_S}^{j}\widetilde{\mathbf{x}}_F - \boldsymbol{\zeta}_{1_S}^{j}\|_{\sigma^2\mathbf{I}}^2 \\
&= \|\mathbf{R}_{ns}\widetilde{\mathbf{x}}_k - \mathbf{r}_{ns}\|^2
\end{aligned} \tag{2.105}$$

where

$$\mathbf{R}_{ns} = \left[\begin{array}{c:ccc:c} \mathbf{R}_{SS} & & \mathbf{0} & & \mathbf{R}_{SF} \\ \hdashline & \frac{1}{\sigma}\mathbf{R}_{f_S}^{1} & & & \frac{1}{\sigma}\mathbf{F}_{1_S}^{1} \\ \mathbf{0} & & \ddots & & \vdots \\ & & & \frac{1}{\sigma}\mathbf{R}_{f_S}^{N_{ns}} & \frac{1}{\sigma}\mathbf{F}_{1_S}^{N_{ns}} \\ \hdashline \mathbf{0} & & \mathbf{0} & & \mathbf{R}_{FF} \end{array}\right] \tag{2.106}$$

$$\mathbf{r}_{ns} = \left[\begin{array}{c:ccc:c} \mathbf{r}_S^T & \frac{1}{\sigma}\boldsymbol{\zeta}_{1_S}^{1^T} & \cdots & \frac{1}{\sigma}\boldsymbol{\zeta}_{1_S}^{N_{ns}^T} & \mathbf{r}_F^T \end{array}\right]^T \tag{2.107}$$

and we have partitioned the current information factor $\mathbf{R}_{sr}$ and residual $\mathbf{r}_{sr}$ from (2.100), according to the state partition of $\mathbf{x}_k^{\rho}$ in (2.103), as:

$$\mathbf{R}_{sr} = \begin{bmatrix} \mathbf{R}_{SS} & \mathbf{R}_{SF} \\ \mathbf{0} & \mathbf{R}_{FF} \end{bmatrix}, \quad \mathbf{r}_{sr} = \begin{bmatrix} \mathbf{r}_S \\ \mathbf{r}_F \end{bmatrix} \tag{2.108}$$

Thus, this step of new SLAM feature initialization simply requires augmenting the current information factor, by inserting the corresponding Jacobians into it as in (2.106), according to the state augmentation in (2.104), and similarly for the residual as in (2.107). Note that this resulting factor $\mathbf{R}_{ns}$ in (2.106) is already *upper-triangular*, since both $\mathbf{R}_{SS}$ and $\mathbf{R}_{FF}$ are upper-triangular from $\mathbf{R}_{sr}$ in (2.108), and $\mathbf{R}_{f_S}^{j}$, for $j = 1, \cdots, N_{ns}$, is upper-triangular from the QR factorization of the LNS transformation in (2.101).

2.4.8 SI Update: New SLAM and SI-MSCKF Pose Constraints

Pose constraints, from all the inlier new feature tracks to be processed as the SI-type, are incorporated during this update step. They correspond to cost terms $\mathcal{C}_{z^2}$ in (2.77) after the LNS transformation, from the new SLAM and the SI-MSCKF feature tracks, i.e., $\mathcal{C}_{\mathbb{Z}_S^2}$ and $\mathcal{C}_{\mathbb{Z}_M^2}$, respectively, which are processed together in a uniform manner. In terms of the cost function [see (2.80)], these terms are added to the current cost $\mathcal{C}_{ns}$ in (2.105), to yield:

$$\begin{aligned}\mathcal{C}_k^{\oplus}(\widetilde{\mathbf{x}}_k) &= \mathcal{C}_{ns}(\widetilde{\mathbf{x}}_k) + \mathcal{C}_{\mathbb{Z}_S^2}(\widetilde{\mathbf{x}}_F) + \mathcal{C}_{\mathbb{Z}_M^2}(\widetilde{\mathbf{x}}_F) \\ &= \|\mathbf{R}_{ns}\widetilde{\mathbf{x}}_k - \mathbf{r}_{ns}\|^2 + \sum_{j=1}^{N_{ns}} \|\mathbf{F}_{2_S}^j \widetilde{\mathbf{x}}_F - \boldsymbol{\zeta}_{2_S}^j\|_{\sigma^2\mathbf{I}}^2 \\ &\quad + \sum_{i=1}^{N_m} \|\mathbf{F}_{2_M}^i \widetilde{\mathbf{x}}_F - \boldsymbol{\zeta}_{2_M}^i\|_{\sigma^2\mathbf{I}}^2 \end{aligned} \tag{2.109}$$

In order to take advantage of the different structures of the involved matrices here, we adopt a two-step approach to obtain $\mathcal{C}_k^{\oplus}$: First, we compute $\mathcal{C}_{\mathbb{Z}_S^2} + \mathcal{C}_{\mathbb{Z}_M^2}$, since the Jacobians of these two cost terms share similar structures. Then, we perform an efficient update to combine the resulting term with $\mathcal{C}_{ns}$.

Specifically, to compute $\mathcal{C}_{\mathbb{Z}_S^2} + \mathcal{C}_{\mathbb{Z}_M^2}$, we stack together all their Jacobians and residuals in (2.109) as:

$$\begin{aligned}\mathbf{F}_{2_{SM}} &= \begin{bmatrix}\cdots & \mathbf{F}_{2_S}^{j^T} & \cdots & \cdots & \mathbf{F}_{2_M}^{i^T} & \cdots\end{bmatrix}^T \\ \boldsymbol{\zeta}_{2_{SM}} &= \begin{bmatrix}\cdots & \boldsymbol{\zeta}_{2_S}^{j^T} & \cdots & \cdots & \boldsymbol{\zeta}_{2_M}^{i^T} & \cdots\end{bmatrix}^T \end{aligned} \tag{2.110}$$

where $\mathbf{F}_{2_{SM}}$ can be very tall if many feature tracks are processed. Then, we perform the following QR factorization on these stacked Jacobian and residual:

$$\begin{bmatrix}\mathbf{F}_{2_{SM}} & \boldsymbol{\zeta}_{2_{SM}}\end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q}_{F_2} \begin{bmatrix}\mathbf{R}_{F_2} & \mathbf{r}_{\zeta_2} \\ \mathbf{0} & \mathbf{e}_{\zeta_2}\end{bmatrix} \tag{2.111}$$

which, after dropping the constant term $\|\mathbf{e}_{\zeta_2}\|^2$, transforms $\mathcal{C}_{\mathbb{Z}_S^2} + \mathcal{C}_{\mathbb{Z}_M^2}$ into:

$$\mathcal{C}_{\mathbb{Z}_S^2}(\widetilde{\mathbf{x}}_F) + \mathcal{C}_{\mathbb{Z}_M^2}(\widetilde{\mathbf{x}}_F) = \|\mathbf{R}_{F_2}\widetilde{\mathbf{x}}_F - \mathbf{r}_{\zeta_2}\|_{\sigma^2\mathbf{I}}^2 \tag{2.112}$$

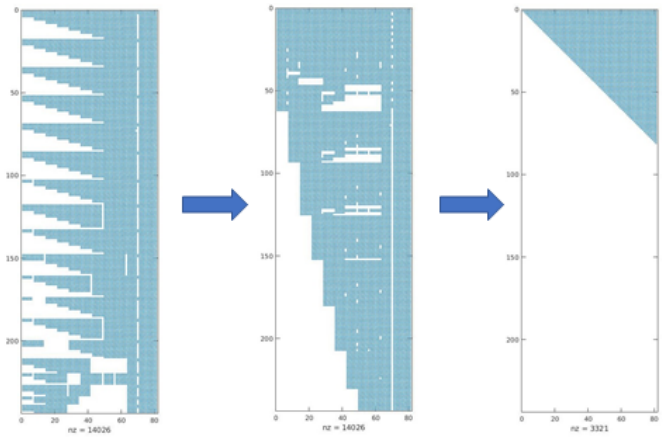


Figure 2.4: An example of the Jacobian structures during the measurement-compression transformation.

where the transformed pose Jacobian $\mathbf{R}_{F_2}$ is upper-triangular.

The procedure from (2.110) to (2.112) is exactly the measurement-compression (MC) transformation on the pose constraints in the MSC-KF algorithm [69]. Here, in our case, the QR factorization in (2.111) is carried out efficiently, by utilizing the block-upper-triangular structure of each $\mathbf{F}_2^j$ in $\mathbf{F}_{2_{SM}}$ (see Fig. 2.4), resulting from our specialized LNS transformation (see Section 2.4.6). To do so, as shown in Fig. 2.4, we first perform a row permutation on the stacked Jacobian $\mathbf{F}_{2_{SM}}$, as well as on $\boldsymbol{\zeta}_{2_{SM}}$, by interleaving its rows according to their leading entry positions, which are determined by the indices of the structural corners of each $\mathbf{F}_2^j$ block (see Fig. 2.3). After this row permutation, the Jacobian matrix has a structure similar to block-upper-triangular, with a large portion of zeros at the bottom-left corner. Again, indices of the structural corners (marked by red dots in Fig. 2.4) of this permuted matrix are easily computed, from those of each $\mathbf{F}_2^j$ as an output of the LNS transformation step. As a result, an efficient QR factorization is executed on the permuted matrix based on this structure information.

At this point, we would like to note that, both the LNS and MC transformations involve *only* measurement Jacobians and residuals, but not the information factor, and hence are not restricted to our specific estimator. For example, these steps are found in the MSC-KF algorithm [69]. Therefore, our approach for exploiting the corresponding structures during these transformations, can be applied to any VIO method that employs the MSCKF feature track processing (with or without SLAM features), so as to achieve computational savings. And

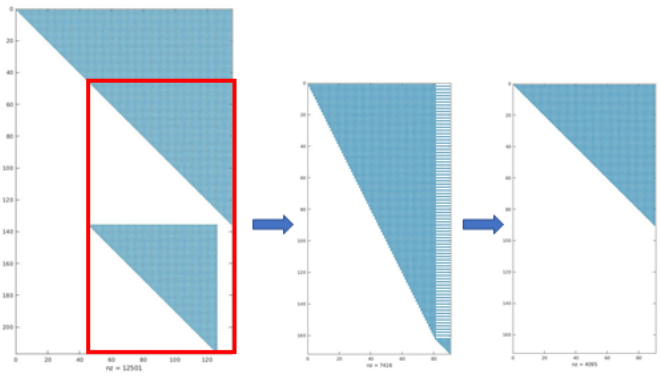


Figure 2.5: An example of the information factor and Jacobian structures during the pose-constraint update.

this holds regardless of the specific estimation framework used, including the four filter forms and optimization-based methods described in Section 2.2, such as the original MSC-KF algorithm [69] or its extensions (e.g., [42, 57, 61]). In fact, we have implemented our specialized LNS and MC transformations for the MSC-KF algorithm in the covariance form and our SR-ISWF presented here in the square-root inverse form, and observed *significant* speedups in both cases.

Next, given the cost term $\mathcal{C}_{\mathbb{Z}_S^2} + \mathcal{C}_{\mathbb{Z}_M^2}$ in (2.112) after the MC transformation, we combine it with $\mathcal{C}_{ns}$ as in (2.109) to perform the update. Same as in the previous update step [see (2.99)], this requires a QR factorization on the stacked matrix of the information factor $\mathbf{R}_{ns}$ and the Jacobian $\mathbf{R}_{F_2}$, together with the residuals. This QR factorization can be carried out efficiently for two reasons: First, the pose constraint in (2.112) involves *only* the state $\mathbf{x}_F$, which, by the design of our state ordering in (2.60), is at the *end* of the state vector $\mathbf{x}_k$ [see (2.104)]. Therefore, as shown in Fig. 2.5, the QR factorization affects only the corresponding $\mathbf{R}_{FF}$ block of $\mathbf{R}_{ns}$ in (2.106), instead of the entire factor, as:

$$\begin{bmatrix} \mathbf{R}_{FF} & \mathbf{r}_F \\ \frac{1}{\sigma}\mathbf{R}_{F_2} & \frac{1}{\sigma}\mathbf{r}_{\zeta_2} \end{bmatrix} \overset{\text{QR}}{=} \mathbf{Q}_{FF}^{\oplus} \begin{bmatrix} \mathbf{R}_{FF}^{\oplus} & \mathbf{r}_F^{\oplus} \\ \mathbf{0} & \mathbf{e}_F \end{bmatrix} \tag{2.113}$$

Second, since both $\mathbf{R}_{FF}$ and $\mathbf{R}_{F_2}$ are *upper-triangular*, a row permutation is performed on the stacked matrix, where we alternate between these two matrices and take one row at a time. As a result, the permuted matrix is almost upper-triangular, with a large portion of zeros at the

bottom-left corner, which allows an efficient QR factorization (see Fig. 2.5).

Finally, after the pose-constraint update in (2.113), the cost function $\mathcal{C}_k^{\oplus}$ in (2.109) becomes:

$$\mathcal{C}_k^{\oplus}(\widetilde{\mathbf{x}}_k) = \|\mathbf{R}_k^{\oplus}\widetilde{\mathbf{x}}_k - \mathbf{r}_k^{\oplus}\|^2 \tag{2.114}$$

where $\mathbf{R}_k^{\oplus}$ and $\mathbf{r}_k^{\oplus}$ are the same as $\mathbf{R}_{ns}$ and $\mathbf{r}_{ns}$ in (2.106) and (2.107), except that their $\mathbf{R}_{FF}$ and $\mathbf{r}_F$ blocks (corresponding to the state $\mathbf{x}_F$) have been updated to $\mathbf{R}_{FF}^{\oplus}$ and $\mathbf{r}_F^{\oplus}$, respectively.

At this point of our estimation algorithm, all the input SI-type measurements ($\mathbb{Z}_R$, $\mathbb{Z}_S$, and $\mathbb{Z}_M$) have been used for update, i.e., information from all the SI-type visual cost terms ($\mathcal{C}_{\mathbb{Z}_R}$, $\mathcal{C}_{\mathbb{Z}_S^1}$, $\mathcal{C}_{\mathbb{Z}_S^2}$, and $\mathcal{C}_{\mathbb{Z}_M^2}$) are absorbed to generate the posterior cost function $\mathcal{C}_k^{\oplus}$ in (2.114) [see (2.80)]. A copy of its information factor $\mathbf{R}_k^{\oplus}$ and residual $\mathbf{r}_k^{\oplus}$ is saved at this moment, and to be used later when we compute the new prior term for the next time step (see Section 2.4.11).

2.4.9 SO Update: SO-MSCKF Pose Constraints

After all the SI-type cost terms are processed, we now use the SO-type ones to perform a further update. They correspond to the pose constraints $\bar{\mathcal{C}}_{\mathbb{Z}_M^2}$ from the inlier SO-MSCKF feature tracks $\bar{\mathbb{Z}}_M$ after the LNS transformation. In terms of the cost function [see (2.80)], $\bar{\mathcal{C}}_{\mathbb{Z}_M^2}$ is added to the posterior cost $\mathcal{C}_k^{\oplus}$ in (2.114), to yield:

$$\mathcal{C}_k^{\star}(\widetilde{\mathbf{x}}_k) = \mathcal{C}_k^{\oplus}(\widetilde{\mathbf{x}}_k) + \bar{\mathcal{C}}_{\mathbb{Z}_M^2}(\widetilde{\mathbf{x}}_F) = \|\mathbf{R}_k^{\star}\widetilde{\mathbf{x}}_k - \mathbf{r}_k^{\star}\|^2 \tag{2.115}$$

which is computed following the same procedure of the SI-type pose-constraint update as in (2.109)-(2.114). Similarly to the case of $\mathbf{R}_k^{\oplus}$ [see (2.113)-(2.114)], this updated information factor $\mathbf{R}_k^{\star}$ is upper-triangular and has the same structure as in (2.106), where *only* its $\mathbf{R}_{FF}^{\oplus}$ block (corresponding to the state $\mathbf{x}_F$) is further updated (and similarly for $\mathbf{r}_k^{\star}$).

At this point, all the information, from both the SI-type and SO-type cost terms, has been incorporated into the total cost function $\mathcal{C}_k^{\star}$ in (2.115) [see (2.80)]. As compared to the posterior cost $\mathcal{C}_k^{\oplus}$ in (2.114), $\mathcal{C}_k^{\star}$ contains more information, by including the SO-type cost terms in addition to the SI-type ones, and hence will lead to a better state estimate in general. Note that, however, because of this inclusion of the SO-type terms, $\mathcal{C}_k^{\star}$ is used for computing *only* the new state estimate (see Section 2.4.10), but *not* the new prior term (see Section 2.4.11).

2.4.10 Computing New State Estimate

The last step of our estimation algorithm's update procedure is to obtain the new state estimate. Specifically, we compute the state correction term $\Delta\mathbf{x}_k$, by minimizing the total cost function $\mathcal{C}_k^\star$ in (2.115), as:

$$\Delta\mathbf{x}_k = \underset{\widetilde{\mathbf{x}}_k}{\operatorname{argmin}}\, \mathcal{C}_k^\star(\widetilde{\mathbf{x}}_k) = \underset{\widetilde{\mathbf{x}}_k}{\operatorname{argmin}}\, \|\mathbf{R}_k^\star\widetilde{\mathbf{x}}_k - \mathbf{r}_k^\star\|^2 \tag{2.116}$$

Since, from the previous update step, $\mathbf{R}_k^\star$ is invertible and upper-triangular, this is equivalent to solving the linear equation:

$$\mathbf{R}_k^\star\Delta\mathbf{x}_k = \mathbf{r}_k^\star \tag{2.117}$$

which, same as in the SR-EIF [see (2.49)], simply requires a backward substitution. Moreover, this operation is carried out even more efficiently, by taking advantage of the specific structure of $\mathbf{R}_k^\star$, where its top-left corner is block-diagonal [see (2.106)]. Then, after solving for $\Delta\mathbf{x}_k$, the state update is given by:

$$\hat{\mathbf{x}}_k^\star = \hat{\mathbf{x}}_k + \Delta\mathbf{x}_k \tag{2.118}$$

where $\hat{\mathbf{x}}_k$ is the current state estimate, and $\hat{\mathbf{x}}_k^\star$ is the new state estimate to serve as our new linearization point.

This is the end of the update procedure. In order to achieve smaller linearization error and hence higher estimation accuracy, besides the relinearization across multiple time steps by using the SO-type cost terms for update, our estimation algorithm also performs relinearization within each time step by iterating the update procedure (see Algorithm 1). This corresponds to the iterative Gauss-Newton optimization, where the nonlinear cost terms $\mathcal{C}_\mathbb{Z}$, from all the visual measurements $\mathbb{Z}$, are relinearized [see (2.72)-(2.74)] around the newest state estimate.

2.4.11 Computing New Prior Term

After the update procedure is complete, our estimation algorithm employs a final step to generate the new prior term for the next time step. The new prior term is essentially the posterior cost $\mathcal{C}_k^\oplus$ in (2.114), which consists of information from all the SI-type cost terms, but *not* the SO-type ones (as opposed to $\mathcal{C}_k^\star$). This posterior cost $\mathcal{C}_k^\oplus(\widetilde{\mathbf{x}}_k)$, however, is with respect to the *old* state estimate (linearization point) $\hat{\mathbf{x}}_k$ before the state update, i.e., $\widetilde{\mathbf{x}}_k = \mathbf{x}_k - \hat{\mathbf{x}}_k$. Now that the state estimate has been updated to the new $\hat{\mathbf{x}}_k^\star$ as in (2.118), we need to modify $\mathcal{C}_k^\oplus(\widetilde{\mathbf{x}}_k)$ so that it corresponds to this *new* linearization point, in order to make it usable for the next time

step. To do so, we substitute (2.118) into (2.114) to obtain:

$$\mathcal{C}_k^{\oplus}(\widetilde{\mathbf{x}}_k) = \|\mathbf{R}_k^{\oplus}\widetilde{\mathbf{x}}_k - \mathbf{r}_k^{\oplus}\|^2 = \|\mathbf{R}_k^{\oplus}\widetilde{\mathbf{x}}_k^{\star} - \mathbf{r}'_k\|^2 = \mathcal{C}_p(\widetilde{\mathbf{x}}_k^{\star}) \tag{2.119}$$

where $\mathbf{r}'_k = \mathbf{r}_k^{\oplus} - \mathbf{R}_k^{\oplus}\Delta\mathbf{x}_k$, and $\mathcal{C}_p(\widetilde{\mathbf{x}}_k^{\star})$ is the new prior term with respect to the new linearization point $\hat{\mathbf{x}}_k^{\star}$, i.e., $\widetilde{\mathbf{x}}_k^{\star} = \mathbf{x}_k - \hat{\mathbf{x}}_k^{\star}$. Moreover, from (2.117) and the fact that $\mathbf{R}_k^{\star}$ equals $\mathbf{R}_k^{\oplus}$ except only the $\mathbf{R}_{FF}^{\oplus}$ block (and similarly for $\mathbf{r}_k^{\star}$) (see Section 2.4.9), it can be verified that $\mathbf{r}'_k$ is further simplified into:

$$\mathbf{r}'_k = \mathbf{r}_k^{\oplus} - \mathbf{R}_k^{\oplus}\Delta\mathbf{x}_k = \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_F^{\oplus} - \mathbf{R}_{FF}^{\oplus}\Delta\mathbf{x}_F \end{bmatrix} \tag{2.120}$$

where $\Delta\mathbf{x}_F$ is the tail segment of the state correction term $\Delta\mathbf{x}_k$, corresponding to the state $\mathbf{x}_F$ [see (2.104)]. Note that $\mathbf{r}'_k \neq \mathbf{0}$ due to the update step using the SO-type cost terms.

As a result of this step, we have obtained the new prior term $\mathcal{C}_p(\widetilde{\mathbf{x}}_k^{\star})$ in (2.119) for the next time step [as in (2.78)]. Note that, since this prior includes *no* information from the SO-type cost terms, all the currently-processed SO-MSCKF feature track measurements $\bar{\mathbb{Z}}_M$ can be reused and relinearized freely at the next time step.

This marks the end of our SR-ISWF estimation algorithm. The outputs are (see Algorithm 1): The new state estimate $\hat{\mathbf{x}}_k \leftarrow \hat{\mathbf{x}}_k^{\star}$, the new prior information factor $\mathbf{R}_k \leftarrow \mathbf{R}_k^{\oplus}$, and the new prior residual $\mathbf{r}_k \leftarrow \mathbf{r}'_k$, for the next time step.

2.5 Experimental Results

In order to evaluate our proposed SR-ISWF algorithm, we implemented this VIO system and tested its performance on visual-inertial datasets. We first compare our system with state-of-the-art VIO implementations, in terms of the estimation accuracy and running speed, using publicly-available datasets. Then, we also show its performance on commercial-grade cell phones with limited processing powers.

2.5.1 System Setup

We implemented a single-threaded pipeline consisting of feature extraction and tracking to provide visual measurements to the filter. First, we extract 400 ORB [79] features per image and

match them based on their descriptors against previous images in the sliding window to generate feature tracks. Then, a 2-Point RANSAC [55] is used for initial outlier rejection. As for a VIO system, no loop-closure measurement is generated or used. The visual measurements are assumed to be contaminated by zero-mean white Gaussian noises with $\sigma = 1.5$ pixels. After that, the information manager selects within the window a maximum of 20 SLAM, 30 SI-MSCKF, and 30 SO-MSCKF features. These feature measurements, as well as the corresponding IMU data, are then processed by our estimator, which maintains a sliding window of $M = 10$ cloned poses. These clones are selected based on the motion of the platform, i.e., when the IMU frame has moved for more than 5 centimeters (cm) or rotated for 5 degrees. Finally, our implementation uses only single-precision floating-point arithmetic, enabled by our square-root formulation.

2.5.2 Performance on the EuRoC Datasets

We compare the performance of our proposed SR-ISWF on the EuRoC [18] datasets, against state-of-the-art open-source VIO estimators: i) Optimization-based methods, including OKVIS [58], VINS-Mono [77] (without loop closure), and ICE-BA [63] (without loop closure); ii) Filtering-based methods, including ROVIO [16] (EKF-based) and OpenVINS [36] (MSC-KF-based). Since our implementation focuses on VIO, we did not compare to vision-only systems, such as [71]. The datasets contain stereo images from global shutter cameras (20 Hz) and IMU measurements (200 Hz), along with ground-truth poses from VICON. We use only the left-camera images, as our evaluation focuses on the setup with a monocular camera and an IMU. Code of each compared system is downloaded from its Github repository and run with its provided configuration file for EuRoC datasets.

Localization Accuracy

We compute the root-mean-square error (RMSE) of the estimated positions and orientations, of the current (latest) pose at each time step, by comparing to the ground truth, i.e., the absolute trajectory error (ATE), to evaluate the localization accuracy of all estimators considered. Each estimated trajectory is aligned with the ground-truth coordinate frame by a 3D-to-3D matching using [43]. The results are shown in Table 2.2. As evident, our algorithm achieves the highest accuracy on the majority of sequences and also on average. This is because: i) Our algorithm

Table 2.2: RMSE of Position (cm) / Orientation (degree) Estimates on EuRoC Datasets

Dataset	OKVIS	VINS-Mono no lc	ICE-BA no lc	ROVIO	OpenVINS	**SR-ISWF (ours)**
MH_01_easy	34.56 / 3.51	15.58 / **1.50**	13.09 / 2.40	21.39 / 3.97	15.59 / 3.24	**12.00** / 1.91
MH_02_easy	40.94 / 3.53	17.82 / 2.31	17.63 / 3.14	37.78 / 3.98	**11.92** / 1.48	12.10 / **1.37**
MH_03_medium	22.08 / 1.59	19.52 / 1.64	20.43 / 1.36	31.22 / 3.20	17.45 / 1.74	**10.77 / 0.93**
MH_04_difficult	33.68 / 1.46	34.84 / 1.49	30.48 / 2.09	57.31 / 1.34	32.13 / 1.39	**28.37 / 0.65**
MH_05_difficult	42.52 / 1.37	30.26 / **0.71**	23.10 / 2.82	69.80 / 1.28	40.21 / 1.53	**16.34** / 1.22
V1_01_easy	10.63 / 5.94	8.87 / 6.34	12.42 / 6.02	15.39 / 6.14	6.31 / 5.60	**5.16 / 5.57**
V1_02_medium	11.60 / 2.23	11.03 / 3.28	11.80 / 1.91	13.01 / 1.81	**6.82 / 1.80**	10.64 / 1.97
V1_03_difficult	22.66 / 4.42	18.71 / 6.22	19.12 / **2.30**	15.48 / 3.49	6.67 / 3.16	**6.11** / 2.50
V2_01_easy	16.18 / **1.12**	8.61 / 2.03	14.66 / 1.99	10.40 / 1.44	8.87 / 2.41	**7.80** / 1.84
V2_02_medium	18.06 / 3.25	15.99 / 4.36	14.17 / 1.89	15.24 / 1.72	**6.48** / 1.58	9.56 / **1.13**
V2_03_difficult	28.48 / 3.76	27.79 / 3.24	15.32 / 1.97	**11.48** / 1.43	22.15 / 2.09	13.01 / **1.40**
Average	25.58 / 2.93	19.00 / 3.01	17.47 / 2.54	27.14 / 2.71	15.87 / 2.36	**11.99 / 1.86**

optimally processes all available measurements (within the budget), without discarding any visual information as in those optimization-based methods compared. ii) As compared to those filtering-based methods, the extra SO-type processing employed in our algorithm effectively reduces the linearization errors. iii) A better outlier rejection due to the Mahalanobis-distance test, which is enabled efficiently in our algorithm as explained in Section 2.4.4. Examples of our estimated trajectories vs. ground truth are shown in Fig. 2.6.

Computational Efficiency

We run all systems on a laptop with Intel Core i7-6700HQ 2.60GHz x 8 CPU to evaluate the efficiency of each estimator. For speed comparisons, we focus only on the estimation portion of each system and do not consider the time spent in other modules (e.g., image processing). Average timing results, in milliseconds (msec), per estimator run are shown in Table 2.3. As evident, our SR-ISWF is significantly faster than all other estimators.[7] This is mainly because: i) Our state size is much smaller, due to the MSCKF feature processing and the exclusion of all IMU extra states except for only the current clone. ii) Only a selective portion of all visual measurements (the SO-type ones) are relinearized by our information processing scheme. iii) We exploit fully the specific problem structures for computational savings. iv) Single-precision arithmetic enabled by our square-root formulation, as compared to all other systems that require

[7] Although we do not compare to ORB-SLAM as their open-source implementation do not use IMU, it is worth noting that according to [63] and Table 2.3, ORB-SLAM is more than 10 times slower than ours.

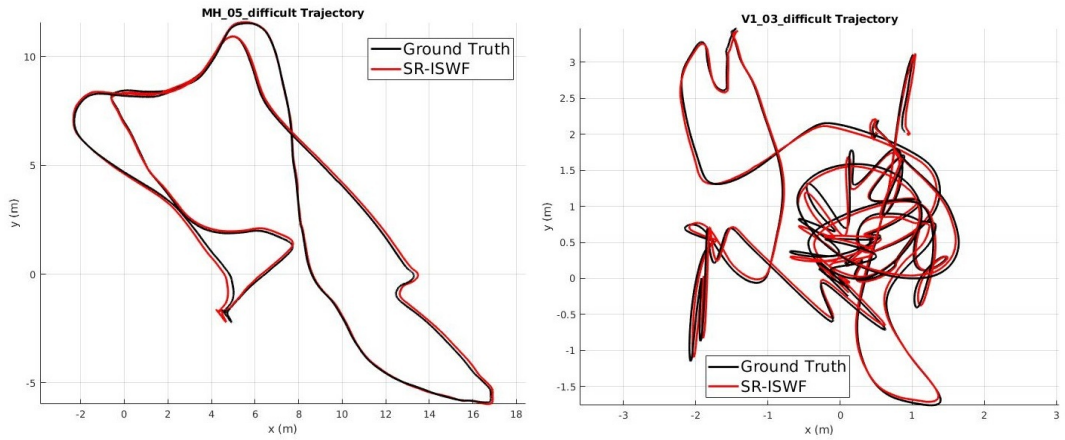


Figure 2.6: EuRoC sequences: Estimated trajectories vs. ground truth, in MH05 (left) and V103 (right).

Table 2.3: Laptop Timing Results per Estimator Run (msec)

OKVIS	VINS-Mono no lc	ICE-BA no lc	ROVIO	OpenVINS	**SR-ISWF (ours)**
27	52	11	27	19	**7**

double-precision.

2.5.3 Performance on Cell-Phone Datasets

To examine the capability of our proposed SR-ISWF for real-time operation on resource-constrained devices, we evaluate its performance on commercial cell phones. Two cell phones are used as our test beds: The Samsung S4 and the Google Pixel phone. The S4 is equipped with a 1.6 GHz quad-core Cortex-A15 ARM CPU, a MEMS-quality IMU running at 100 Hz, and a rolling-shutter camera providing images with resolution 640×480 at 30 Hz. The camera and the IMU have separate clocks, while the image time-stamps are inaccurate. Similar sensor settings hold for the Pixel phone, while it has a faster Quad-core CPU (2.15 GHz dual core + 1.6 GHz dual core). We use the same system setup as mentioned earlier, but with a different image-processing module for higher speed on cell phones. Specifically, the pipeline extracts 300 FAST corners [78] from images at 15 Hz, which are tracked by the KLT algorithm [64] across images. When running on the cell phones, in addition to single-precision arithmetic, our implementation further accelerates the major computational steps of our algorithm (i.e., all QR

Table 2.4: Cell-Phone Datasets: Loop-Closure Error Percentages (%)

	Trajectory Length (m)	MSC-KF+	SR-ISWF Naive	SR-ISWF
Dataset 1	285	0.65	0.48	0.44
Dataset 2	56	0.52	0.71	0.77
Dataset 3	97	0.50	0.52	0.55
Dataset 4	105	0.58	0.74	0.70
Dataset 5	198	0.42	0.35	0.27
Average		0.53	0.56	0.55

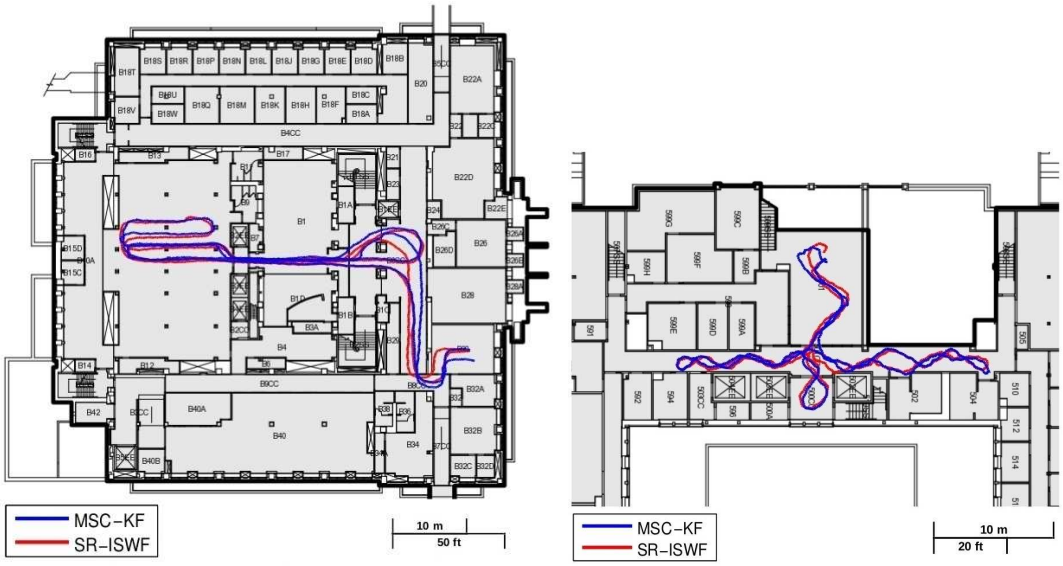


Figure 2.7: Cell-phone datasets: Estimated trajectories from MSC-KF+ vs. our SR-ISWF, overlaid onto the blueprints of the floor plans, in Dataset 1 (left) and Dataset 4 (right).

factorizations) by using ARM NEON intrinsics.

For speed comparisons, we also implemented two equivalent estimators to our SR-ISWF: i) The "MSC-KF+", which is an extension to the basic MSC-KF algorithm [69], so that it follows the same information processing scheme as ours, i.e., including in addition the SLAM feature and SO-type processing. ii) The "naive" SR-ISWF, which is the same as our algorithm while ignoring the problem structures, by considering all matrices involved in its QR factorizations as dense.

Table 2.5: S4 Cell-Phone Timing Results per Estimator Run (msec)

	MSC-KF+	SR-ISWF Naive	SR-ISWF
Estimator Run Mean / Std	50.7 / 6.8	52.1 / 7.0	23.5 / 4.3
Total Pipeline Mean	114.4	100.2	71.8

Table 2.6: Pixel Cell-Phone Timing Results per Estimator Run (msec)

	ORB Matching	FAST-KLT
Estimator Run	11	15
Image Processing	32	20
Total Pipeline	51	44

Localization Accuracy

Five indoor datasets are collected using the S4 cell phone. Due to lack of ground truth, in all datasets, the device is returned back to its starting position. This allows us to quantitatively evaluate the accuracy of the estimators using the loop-closure error percentage, which is computed as the ratio of the distance between the estimated starting and ending points against the total distance travelled. The results are shown in Table 2.4. In addition, to visualize the estimation accuracy, we overlay the estimated trajectories onto the floor plans' blueprints as reference, which are depicted in Fig. 2.7. As evident, all three estimators achieve comparable levels of accuracy (about half percent loop-closure error on average), and their trajectories overlay each other in most places, which are expected as these estimators are equivalent.

Computational Efficiency

Table 2.5 shows the run time of the three algorithms on the S4 cell phone. First, we observe that the MSC-KF+, which operates in the covariance form, has a similar speed to that of the naive SR-ISWF. Then, by switching to our specialized QR functions designed for the problem structures and accelerated with the NEON co-processor, the estimator run time of our SR-ISWF is reduced to less than half of the others. Finally, timing results of our SR-ISWF pipeline on the Pixel phone are reported in Table 2.6, where we compare the two aforementioned image-processing options, i.e., ORB descriptor matching vs. FAST-KLT tracking. Due to the higher speed of FAST-KLT, this approach reduces the latency of the overall pipeline, although it slightly increases the estimation time due to more tracked corners. Based on these

results, our SR-ISWF algorithm achieves (faster than) real-time operations on both cell phones.

2.6 Summary

In this chapter, we studied the problem of designing a state estimator for visual-inertial odometry (VIO). As a theoretical foundation, we showed four equivalent forms of the filtering methods, and established their correspondence to one iteration of the nonlinear optimization-based methods. Furthermore, we discussed extensions of the basic filters, for performing the same relinearization processes as in the optimization-based methods. In our analysis, we introduced the concept of state-only (SO) processing, that relinearizes the same measurement across multiple time steps, for obtaining better state estimates. Based on our discussion of these estimators and their applications to the VIO-SLAM problem, we determined the key aspects of our proposed VIO estimator, the square-root inverse sliding-window filter (SR-ISWF): i) It operates in the square-root inverse form, so as to enable using single-precision format for performing numerical operations very fast. ii) It employs a hybrid visual information processing scheme, which combines the SLAM+MSCKF feature approach with the SI+SO type measurement processing, for achieving estimation accuracy and computational efficiency at the same time. Given the selected visual-inertial information, we derived in detail all steps of our estimation algorithm in the square-root inverse form. Moreover, further computational savings were gained by taking advantage of the nonzero patterns of the information factor and Jacobian matrices involved. Our approach for exploiting the Jacobian structures during the left-nullspace (LNS) and measurement-compression (MC) transformations is also applicable to other popular VIO methods, such as the MSC-KF algorithm. A complete VIO system with the proposed SR-ISWF algorithm was implemented and compared to open-source state-of-the-art VIO alternatives on the EuRoC datasets. Results showed that our SR-ISWF provides better pose tracking accuracy, with significantly reduced processing time. And it achieves fast running speed even on cell phones, allowing for real-time operations on these resource-constrained mobile devices.

Chapter 3

Planar Visual-Inertial Localization and Mapping: Observability Analysis and Model Extensions

In this chapter, we consider the problem of visual-inertial localization and mapping, under the special case of planar motions. Specifically, we present a vision-aided inertial navigation system (VINS) for localizing wheeled robots. In particular, we prove that VINS has additional unobservable directions, such as the scale, when deployed on a ground vehicle that is constrained to move along straight lines or circular arcs. To address this limitation, we extend VINS to incorporate low-frequency wheel-encoder data, and show that the scale becomes observable. Furthermore, and in order to improve the localization accuracy, we introduce the manifold-(m)VINS that exploits the fact that the vehicle moves on an approximately planar surface. In our experiments, we first show the performance degradation of VINS due to special motions, and then demonstrate that by utilizing the additional sources of information, our system achieves significantly higher positioning accuracy, while operating in real-time on a commercial-grade mobile device.

3.1 Introduction and Related Work

Over the past 20 years, extensive research has focused on simultaneous localization and mapping (SLAM) with mobile robots navigating over flat terrain [10, 26]. In the absence of GPS, various exteroceptive sensors (e.g., ultrasonic, laser scanners, cameras, and, more recently, RGB-D) have been used in conjunction with 2D wheel odometry to determine, typically, the 3-degree-of-freedom (dof) position and orientation (pose) of the robot. In most cases, however, the underlying planar-motion assumption is only approximately satisfied (e.g., due to the unevenness, or roughness, of the surface, the presence of ramps, bumps, and low-height obstacles on the floor), thus significantly increasing the unmodeled part of the robot's odometry error and leading to low-accuracy estimates or, in the absence of external corrections, even divergence.

On the other hand, vision-aided inertial navigation systems (VINS), where visual observations from a camera are combined with data from an inertial measurement unit (IMU) to estimate the 6-dof pose of a platform navigating in 3D, have been shown to achieve high-accuracy localization results (e.g., [58, 70]), even on low-cost mobile devices (e.g., [59, 89]). Therefore, one would expect that it would be straightforward to deploy a VINS for localizing robots moving in 2D. Surprisingly, however, this is not the case. And one of the main reasons is that the restricted motion (approximately planar and, for the most part, along arcs or straight lines at constant speed or acceleration) that ground robots often undergo when navigating, e.g., indoors, alters the observability properties of VINS and renders certain, additional, dof unobservable.

Specifically, as proven in [42, 66], a VINS has 4 unobservable dof corresponding to 3 dof of global translation and 1 dof of rotation around the gravity vector (yaw). This result, however, holds only when the IMU-camera pair undergoes generic 3D motion. In contrast, and as shown in this chapter, additional dof, such as the scale, become unobservable when the robot is restricted to move with constant acceleration.[1] In particular, under the simplifying assumption of perfectly-known gyroscope biases, [67] showed that the VINS's initial state cannot be uniquely determined for certain motions, but without specifying which are the additional unobservable directions. In this work, we consider the most general case of unknown gyroscope biases and determine these unobservable directions *analytically*.[2]

[1] Note that although the motion constraints considered are never *exactly* satisfied, as explained later on and shown experimentally, motion profiles close to these significantly reduce the information along the unobservable directions, and hence degrade the localization accuracy.

[2] Note that observability is a fundamental property of the VINS model itself, and does not depend on the specific estimator employed for SLAM. Thus, the additional unobservable directions of monocular-VINS will negatively

Furthermore, motivated by the key findings of our observability analysis, in this chapter we focus on improving the localization accuracy of VINS when deployed on wheeled robots. Firstly, in order to ensure that information about the scale is always available (e.g., even for the periods of time when the robot moves with almost constant acceleration), we extend the VINS algorithm to incorporate wheel-odometry measurements. Since these are often noisy and of frequency significantly lower than that of the IMU, we process them in a robust manner, by first integrating the raw encoder data and then treating them as inferred displacement measurements between consecutive poses. Additionally, we take advantage of the fact that the robot moves on an *approximately* flat surface and introduce the manifold-(m)VINS, which explicitly considers the planar-motion constraint in the estimation algorithm to reduce the localization error. This is achieved by analyzing the motion profile of the robot, and its deviation from planar motion (e.g., due to terrain unevenness or vibration of the IMU-camera's mounting platform) and formulating stochastic (i.e., "soft"), instead of deterministic (i.e., "hard") constraints, that allow to properly model the vehicle's almost-planar motion. In summary, our main contributions are:

- We analytically determine the unobservable dof of VINS under special, restrictive motions.
- We extend VINS to process low-frequency odometric measurements, thus rendering scale always observable.
- We introduce the mVINS which incorporates constraints about the motion of the vehicle (in this case, approximately planar) to improve the localization accuracy.
- Through experiments, we validate the key findings of our theoretical analysis, and demonstrate the increased accuracy of the proposed VINS-algorithm extensions when deployed on a tablet onboard a wheeled robot that navigates within a large-scale building.

In what follows, we first present an overview of VINS (Sect. 3.2) followed by its observability analysis when undergoing specific motion profiles (Sect. 3.3). Sect. 3.4 describes the process for incorporating additional information in the VINS, and in particular odometric measurements (Sect. 3.4.1) and motion constraints (Sect. 3.4.2). In Sect. 3.5, we present the experimental validation of the proposed algorithm. Finally, Sect. 3.6 summarizes this chapter.

---

impact the accuracy of both batch-least-squares (e.g., [24, 34, 51]) and sliding-window filters/smoothers (e.g., [15, 58, 59, 70, 89]).

3.2 Preliminaries on Vision-aided Inertial Navigation System (VINS)

In this section, we provide a brief review of the monocular-VINS which serves as the key component of our system. The VINS estimates the following state vector:

$$\mathbf{x} = \begin{bmatrix} {}^{I}\mathbf{q}_G^T & \mathbf{b}_g^T & {}^{G}\mathbf{v}_I^T & \mathbf{b}_a^T & {}^{G}\mathbf{p}_I^T \mid {}^{G}\mathbf{f}_1^T & \cdots & {}^{G}\mathbf{f}_N^T \end{bmatrix}^T \tag{3.1}$$

where ${}^{I}\mathbf{q}_G$ is the unit quaternion that represents the orientation of the global frame $\{G\}$ in the IMU frame $\{I\}$ at time t. ${}^{G}\mathbf{v}_I$ and ${}^{G}\mathbf{p}_I$ are the the velocity and position of $\{I\}$ in $\{G\}$, respectively, and the gyroscope and accelerometer biases are denoted by $\mathbf{b}_g$ and $\mathbf{b}_a$, respectively. Finally, the positions of point features in $\{G\}$ are denoted by ${}^{G}\mathbf{f}_j,\ j = 1, ..., N$.

The IMU provides measurements of the rotational velocity, $\boldsymbol{\omega}_m$, and the linear acceleration, $\mathbf{a}_m$, as:

$$\begin{aligned} \boldsymbol{\omega}_m(t) &= {}^{I}\boldsymbol{\omega}(t) + \mathbf{b}_g(t) + \mathbf{n}_g(t) \\ \mathbf{a}_m(t) &= \mathbf{C}({}^{I}\mathbf{q}_G(t))({}^{G}\mathbf{a}(t) - {}^{G}\mathbf{g}) + \mathbf{b}_a(t) + \mathbf{n}_a(t) \end{aligned} \tag{3.2}$$

where the noise terms, $\mathbf{n}_g(t)$ and $\mathbf{n}_a(t)$ are modelled as zero-mean, white Gaussian noise processes, while the gravitational acceleration, ${}^{G}\mathbf{g}$, is considered a known constant. The IMU's rotational velocity ${}^{I}\boldsymbol{\omega}(t)$ and linear acceleration ${}^{G}\mathbf{a}(t)$, in (3.2), can be used to derive the continuous-time system equations:

$$\begin{aligned} {}^{I}\dot{\mathbf{q}}_G(t) &= \frac{1}{2}\boldsymbol{\Omega}(\boldsymbol{\omega}_m(t) - \mathbf{b}_g(t) - \mathbf{n}_g(t)){}^{I}\mathbf{q}_G(t) \\ \dot{\mathbf{b}}_g(t) &= \mathbf{n}_{wg}(t) \\ {}^{G}\dot{\mathbf{v}}_I(t) &= \mathbf{C}({}^{I}\mathbf{q}_G(t))^T(\mathbf{a}_m(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)) + {}^{G}\mathbf{g} \\ \dot{\mathbf{b}}_a(t) &= \mathbf{n}_{wa}(t) \\ {}^{G}\dot{\mathbf{p}}_I(t) &= {}^{G}\mathbf{v}_I(t) \\ {}^{G}\dot{\mathbf{f}}_j(t) &= \mathbf{0},\ \ j = 1, \ldots, N \end{aligned} \tag{3.3}$$

where, $\boldsymbol{\Omega}(\boldsymbol{\omega}) \triangleq \begin{bmatrix} -\lfloor\boldsymbol{\omega}\rfloor & \boldsymbol{\omega} \\ -\boldsymbol{\omega}^T & 0 \end{bmatrix}$ for $\boldsymbol{\omega} \in \mathbb{R}^3$, $\lfloor\cdot\rfloor$ denotes the skew-symmetric matrix, while the IMU biases are modelled as random walks driven by white, zero-mean Gaussian noise processes $\mathbf{n}_{wg}(t)$ and $\mathbf{n}_{wa}(t)$, respectively.

As the camera-IMU pair moves, the camera provides measurements of point features extracted from the images. Each such measurement, $\mathbf{z}_j$, is modeled as the perspective projection of the point feature $\mathbf{f}_j$, expressed in the current IMU frame[3] $\{I\}$, onto the image plane:

$$\mathbf{z}_j = \frac{1}{z}\begin{bmatrix} x \\ y \end{bmatrix} + \mathbf{n}_j, \quad \begin{bmatrix} x \\ y \\ z \end{bmatrix} \triangleq {}^I\mathbf{f}_j = \mathbf{C}({}^I\mathbf{q}_G)({}^G\mathbf{f}_j - {}^G\mathbf{p}_I) \tag{3.4}$$

where the measurement noise, $\mathbf{n}_j$, is modeled as zero mean, white Gaussian. For modeling the IMU propagation [see (3.3)] and camera observations [see (3.4)], including their error equations and analytical Jacobians, we follow [42].

3.3 VINS: Observability Analysis Under Specific Motion Profiles

Observability is a fundamental property of a dynamic system and provides important insights. Previous works have studied the observability properties of VINS, and employed the results of their analysis to improve the consistency of the estimator [42]. Specifically, in [42, 66], it was shown that, *for generic motions*, a VINS has four unobservable directions (three for global translation and one for global yaw).

In this chapter, we are interested in the case when the VINS is deployed on a ground vehicle, whose motion is approximately planar, and, for the most part, along a straight line (e.g., when moving forward) or a circular arc (e.g., when turning). In particular, we are interested in the impact that such motions have on the VINS's observability properties, and hence the accuracy of the corresponding estimator.

3.3.1 Constant Acceleration

Consider that the platform moves with constant *local* linear acceleration (e.g., on a circle), i.e.,

$${}^I\mathbf{a}(t) \triangleq \mathbf{C}({}^I\mathbf{q}_G(t)){}^G\mathbf{a}(t) \equiv {}^I\mathbf{a}, \;\; \forall\, t \geq t_0 \tag{3.5}$$

where ${}^I\mathbf{a}$ is a constant vector with respect to time, we prove the following theorem:

[3] For clarity of presentation, we assume that the IMU-camera frames coincide. In practice, we estimate the IMU-camera extrinsics online.

Theorem 1: The linearized monocular-VINS model of (2.65) - (3.4) has the following *additional* unobservable direction, besides the global translation and yaw, *if and only if* condition (3.5) is satisfied:

$$\mathbf{N}_s = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ \mathbf{0}_{3\times 1} \\ {}^{G}\mathbf{v}_{I_0} \\ -{}^{I}\mathbf{a} \\ {}^{G}\mathbf{p}_{I_0} \\ {}^{G}\mathbf{f}_1 \\ \vdots \\ {}^{G}\mathbf{f}_N \end{bmatrix} \tag{3.6}$$

Proof: See Appendix A.1.

Remark: The unobservable direction in (3.6) corresponds to the scale, as shown in Appendix A.4.

The physical interpretation of Thm. 1 is that, when the local acceleration is non-varying, one *cannot* distinguish the magnitude of the true body acceleration from that of the accelerometer bias, as both of them are, at least temporarily, constant. As a consequence, the magnitude of the true body acceleration can be arbitrary, leading to scale ambiguity.

At this point, we should note that in most cases in practice, a ground vehicle moves on a plane with (almost) constant acceleration, such as when following a straight line path with constant speed or acceleration, or when making turns along a circular arc with constant speed, etc. Based on Thm. 1, these motions render the scale estimated by the VINS inaccurate.

3.3.2 No Rotation

Consider that the platform has no rotational motion, i.e., the orientation remains the same across time:

$$ {}^{I_t}_{G}\mathbf{C} \triangleq \mathbf{C}({}^{I}\mathbf{q}_G(t)) \equiv {}^{I_0}_{G}\mathbf{C}, \ \ \forall\, t \geq t_0 \tag{3.7}$$

where I_t denotes the IMU frame at time t. Then, the following theorem regarding observability holds:

Theorem 2: The linearized VINS model of (2.65) - (3.4) has the following *additional* unobservable directions, besides the global translation, *if and only if* condition (3.7) is satisfied:

$$\mathbf{N}_o = \begin{bmatrix} {}^{I_0}_{G}\mathbf{C} \\ \mathbf{0}_{3\times 3} \\ -\lfloor {}^{G}\mathbf{v}_{I_0} \rfloor \\ {}^{I_0}_{G}\mathbf{C}\lfloor {}^{G}\mathbf{g} \rfloor \\ -\lfloor {}^{G}\mathbf{p}_{I_0} \rfloor \\ -\lfloor {}^{G}\mathbf{f}_1 \rfloor \\ \vdots \\ -\lfloor {}^{G}\mathbf{f}_N \rfloor \end{bmatrix} \tag{3.8}$$

Proof: See Appendix A.2.

Remark: The unobservable directions in (3.8) correspond to all 3 dof of global orientation instead of only yaw, as shown in Appendix A.5.

The physical interpretation of Thm. 2 is that, when there is no rotational motion, one *cannot* distinguish the direction of the local gravitational acceleration from that of the accelerometer bias, as both of them are, at least temporarily, constant. As a consequence, the roll and pitch angles become ambiguous.

The motion profile considered in Thm. 2 is the case typically followed by a robot moving on a straight line, or (for a holonomic vehicle) sliding sideways. In such cases, due to the lack of observability, the orientation estimates generated by the VINS become inaccurate.

In summary, moving with constant acceleration or without rotating can introduce extra unobservable directions to the VINS model. At this point, we should reiterate that, although, in practice, these specific motion constraints are never *exactly* satisfied all the time, when the robot (even temporarily) *approximately* follows them, it acquires very limited information along the unobservable directions. This will cause the information (Hessian) matrix estimated by the VINS to be severely ill-conditioned, or even numerically rank-deficient, and hence degrades the localization performance. The impact of such motion on the VINS accuracy is demonstrated experimentally in Sect. 3.5.

Among the two cases of unobservability, that of global orientation (see Thm. 2) is the one that can be easily alleviated by allowing the robot to deviate from its straight-line path. On the other hand, rendering scale observable is quite challenging as it would require the robot

to constantly change its acceleration, which would increase the wear and tear of its mobility system. Instead, in what follows, we propose to address this issue and ensure scale observability by extending the VINS to incorporate measurements provided by the robot's wheel odometer.

3.4 VINS: Incorporating Extra Information

In order to improve the performance of VINS for wheeled vehicles, we hereafter present our methodology for incorporating two additional sources of information: (i) Odometry measurements and (ii) Planar-motion constraints.

3.4.1 VINS with Odometer

Most ground vehicles are equipped with wheel encoders that provide low-frequency, often noisy, and maybe only intermittently, reliable measurements of the motion of each wheel. On the other hand, these measurements contain scale information necessary for improving the accuracy of VINS under constant-acceleration motions. In particular, the wheel-encoder data can be transformed into local 2D linear and rotational velocity measurements by employing the odometer intrinsics,[4] i.e.,

$$v = \frac{r_l w_l + r_r w_r}{2}, \quad w = \frac{r_r w_r - r_l w_l}{a} \tag{3.9}$$

where w_l, w_r are the rotational velocities of the left and right wheels, respectively, r_l, r_r are their corresponding radii, and a denotes the vehicle's baseline.

First, we show that adding these odometric measurements makes the scale of VINS observable:

Theorem 3: Given the odometry measurements of (3.9), the scale direction in (3.6) of the linearized VINS model [see (2.65) - (3.4)] becomes observable.

Proof: See Appendix A.3.

In particular, the odometer's linear velocity measurements contain the absolute scale information. Thus, an odometric sensor improves the localization accuracy of VINS not only

[4] In this work, we compute offline the batch least-squares estimates of the wheel encoder intrinsics, including the baseline and the radius of each wheel, based on visual, inertial, and odometry data. Subsequently, we consider them as known quantities. Note that these intrinsic states are observable within VINS.

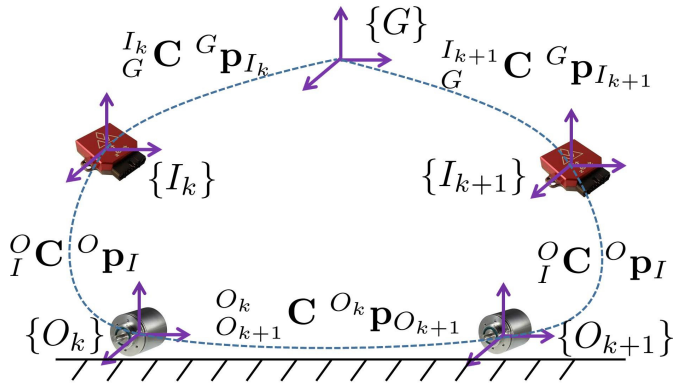


Figure 3.1: Geometric relation between the IMU, $\{I\}$, and odometer, $\{O\}$, frames when the robot moves from time step k to $k+1$.

by recording additional motion measurements, but primarily by providing critical information along the VINS's scale direction which often becomes unobservable due to the vehicle's motion.

In order to process the noisy odometer data in a robust manner, instead of using the velocity measurements in (3.9), we propose to integrate them and fuse the resulting 2D displacement estimates into the 3D VINS. We start by deriving the measurement model, where we assume that, between consecutive odometer readings, the motion is planar. Hence, the transformation between two consecutive odometer frames, $\{O_k\}$ and $\{O_{k+1}\}$, involves a rotation around only the z axis by an angle ${}^{O_k}\phi_{O_{k+1}}$:

$$ {}^{O_k}_{O_{k+1}}\mathbf{C} = \mathbf{C}_z({}^{O_k}\phi_{O_{k+1}}) \tag{3.10} $$

and a translation within the x-y plane, i.e., the first two elements of the translation vector ${}^{O_k}\mathbf{p}_{O_{k+1}}$. Integrating the linear and rotational velocities obtained from the odometer provides measurements to these 3-dof quantities, i.e.,

$$ \phi_k = {}^{O_k}\phi_{O_{k+1}} + n_\phi \tag{3.11} $$

$$ \mathbf{d}_k = \Lambda\, {}^{O_k}\mathbf{p}_{O_{k+1}} + \mathbf{n}_d, \quad \Lambda = \begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2\end{bmatrix}^T \tag{3.12} $$

where $\begin{bmatrix}n_\phi & \mathbf{n}_d^T\end{bmatrix}^T$ is a 3×1 zero-mean Gaussian noise vector.

Furthermore, from the geometric constraints, depicted in Fig. 3.1, the transformation between two consecutive odometer frames, at time steps k and $k+1$, can be written as:

$$
\begin{aligned}
{}^{O_k}_{O_{k+1}}\mathbf{C} &= {}^{O}_{I}\mathbf{C}\,{}^{I_k}_{G}\mathbf{C}\,{}^{I_{k+1}}_{G}\mathbf{C}^T\,{}^{O}_{I}\mathbf{C}^T && (3.13)\\
{}^{O_k}\mathbf{p}_{O_{k+1}} &= {}^{O}\mathbf{p}_I + {}^{O}_{I}\mathbf{C}\,{}^{I_k}_{G}\mathbf{C}({}^{G}\mathbf{p}_{I_{k+1}} - {}^{G}\mathbf{p}_{I_k} - {}^{I_{k+1}}_{G}\mathbf{C}^T\,{}^{O}_{I}\mathbf{C}^T{}^{O}\mathbf{p}_I) && (3.14)
\end{aligned}
$$

where ${}^{O}_{I}\mathbf{C}$ and ${}^{O}\mathbf{p}_I$ are the rotation and translation of the odometer-IMU extrinsics,[5] respectively.

Next, we employ (3.11)-(3.14) to derive the Jacobians and residuals of the corresponding measurement models to be used by the VINS estimator.

Rotational Component

By equating (3.10) to (3.13), and employing small-angle approximations in the rotation matrices involved, we obtain the following error equation:

$$
\begin{aligned}
\delta\boldsymbol{\phi} &= {}^{O}_{I}\mathbf{C}\,\delta\boldsymbol{\theta}_{I_k} - {}^{O}_{I}\mathbf{C}\,{}^{I_k}_{G}\hat{\mathbf{C}}\,{}^{I_{k+1}}_{G}\hat{\mathbf{C}}^T\delta\boldsymbol{\theta}_{I_{k+1}} - n_\phi\mathbf{e}_3 && (3.15)\\
\text{with}\quad \lfloor\delta\boldsymbol{\phi}\rfloor &= \mathbf{I}_3 - \mathbf{C}_z(\phi_k)\,{}^{O_k}_{O_{k+1}}\hat{\mathbf{C}}^T\\
{}^{O_k}_{O_{k+1}}\hat{\mathbf{C}} &= {}^{O}_{I}\mathbf{C}\,{}^{I_k}_{G}\hat{\mathbf{C}}\,{}^{I_{k+1}}_{G}\hat{\mathbf{C}}^T\,{}^{O}_{I}\mathbf{C}^T
\end{aligned}
$$

where $\hat{\mathbf{C}}$ denotes the estimate of the rotation matrix $\mathbf{C}$, and $\delta\boldsymbol{\theta}$ is the error state of the corresponding quaternion parameterization. The third element of the vector $\delta\boldsymbol{\phi}$ represents the angular error between the measured and the estimated in-plane rotation. Multiplying both sides of (3.15) with $\mathbf{e}_3^T$ yields the Jacobians and residual:

$$
\begin{aligned}
\mathbf{H}_{\delta\theta_{I_k}} &= \mathbf{e}_3^T\,{}^{O}_{I}\mathbf{C}\ ,\ \ \mathbf{H}_{\delta\theta_{I_{k+1}}} = -\mathbf{e}_3^T\,{}^{O}_{I}\hat{\mathbf{C}}\,{}^{I_k}_{G}\hat{\mathbf{C}}\,{}^{I_{k+1}}_{G}\hat{\mathbf{C}}^T\\
\mathbf{r} &= \mathbf{e}_3^T\,\delta\boldsymbol{\phi} && (3.16)
\end{aligned}
$$

[5] In this work, we compute offline the batch least-squares estimates of the odometer-IMU extrinsics, based on visual, inertial, and odometry data. Subsequently, we consider them as known quantities. The observable directions of the VINS with odometer extrinsics are presented in [40].

Translational Component

By substituting (3.14) into (3.12) and linearizing, it is straightforward to obtain the following Jacobians and residual:

$$\begin{aligned}
\mathbf{H}_{\delta\theta_{I_k}} &= \Lambda_I^O \mathbf{C} \lfloor \boldsymbol{\xi} \rfloor,\ \mathbf{H}_{\delta\theta_{I_{k+1}}} = \Lambda_I^O \mathbf{C}_G^{I_k} \hat{\mathbf{C}}_G^{I_{k+1}} \hat{\mathbf{C}}^T \lfloor {}_I^O\mathbf{C}^{TO} \mathbf{p}_I \rfloor \\
\mathbf{H}_{p_{I_k}} &= -\Lambda_I^O \mathbf{C}_G^{I_k} \hat{\mathbf{C}}\ ,\ \ \mathbf{H}_{p_{I_{k+1}}} = \Lambda_I^O \mathbf{C}_G^{I_k} \hat{\mathbf{C}} \\
\mathbf{r} &= \mathbf{d}_k - \Lambda({}^O\mathbf{p}_I + {}_I^O\mathbf{C}\,\boldsymbol{\xi}) \\
\text{with}\ \ \boldsymbol{\xi} &\triangleq {}_G^{I_k}\hat{\mathbf{C}}({}^G\hat{\mathbf{p}}_{I_{k+1}} - {}^G\hat{\mathbf{p}}_{I_k} - {}_G^{I_{k+1}}\hat{\mathbf{C}}\,{}_I^O\mathbf{C}^{TO}\mathbf{p}_I).
\end{aligned} \tag{3.17}$$

Finally, (3.16) and (3.17) represent stochastic constraints between the poses of the platform, and can be combined in a tightly-coupled manner into standard VINS estimators.

3.4.2 mVINS: VINS within a Manifold

In many cases in practice, the trajectory of a moving object often lies within some manifold. Ground vehicles, for example, travel mostly on a plane, especially when navigating indoors. The knowledge of this specific motion manifold can provide additional information for improving the localization accuracy of VINS.

A motion manifold can be described mathematically as geometric constraints, $\mathbf{g}(\mathbf{x}) = \mathbf{0}$, where $\mathbf{g}$ is, in general, a nonlinear function of the state $\mathbf{x}$. There are two approaches for incorporating such information into a VINS:

Deterministic Constraints

A standard VINS estimator (e.g., a filter or a smoother) optimizes a cost function $\mathcal{C}(\mathbf{x})$ arising from the information in the sensor (visual, inertial, and potentially odometric) data (e.g., [24,34, 58,89]), while the motion manifold is described as a deterministic constraint of the optimization problem, i.e.,

$$\begin{aligned}
&\min\ \ \mathcal{C}(\mathbf{x}) \\
&\text{s.t.}\ \ \mathbf{g}(\mathbf{x}) = \mathbf{0}
\end{aligned} \tag{3.18}$$

For VINS, the cost function $\mathcal{C}(\mathbf{x})$ typically takes the form of nonlinear least squares,

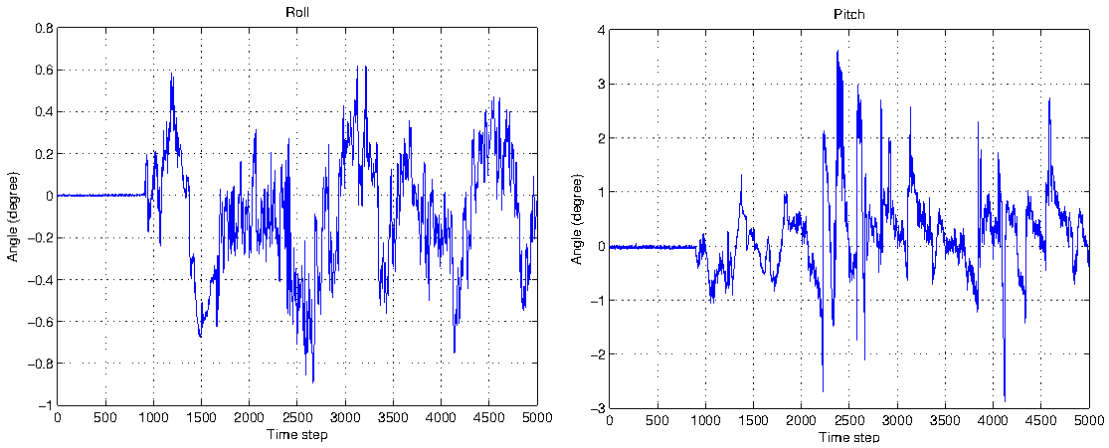


Figure 3.2: The roll (left) and pitch (right) angles in degrees across time, when the robot is moving on a flat surface. The mean is -0.08 and 0.2 degree, and the standard deviation is 0.3 and 0.7 degree, respectively.

and (3.18) can be solved by employing iterative Gauss-Newton minimization [76].

Stochastic Constraints

In practice, the motion manifold is never exactly satisfied. Fig. 3.2 depicts the platform's roll and pitch angles across time, when a ground robot (a Pioneer 3 in our case) is moving on a flat surface. During an ideal planar motion, the roll and pitch angles would have remained constant. As evident, however, this is not the case in practice due to the vibrations of the moving platform and the unevenness of the surface. To account for such deviations, we model the manifold as a stochastic constraint $\mathbf{g}(\mathbf{x}) = \mathbf{n}$, where $\mathbf{n}$ is assumed to be a zero-mean Gaussian process with covariance $\mathbf{R}$, and incorporate this information as an additional cost term:

$$\min \;\; \mathcal{C}(\mathbf{x}) + ||\mathbf{g}(\mathbf{x})||^2_{\mathbf{R}} \tag{3.19}$$

Note that (3.19) can be solved by employing standard VINS estimators. Moreover, this stochastic approach (as compared to the deterministic one) provides more flexibility for rejecting false information due to outliers. Specifically, we employ the Mahalanobis distance test to detect and temporally remove the constraints when they are least likely (in the probabilistic sense) to be satisfied (e.g., when the robot goes over a bump).

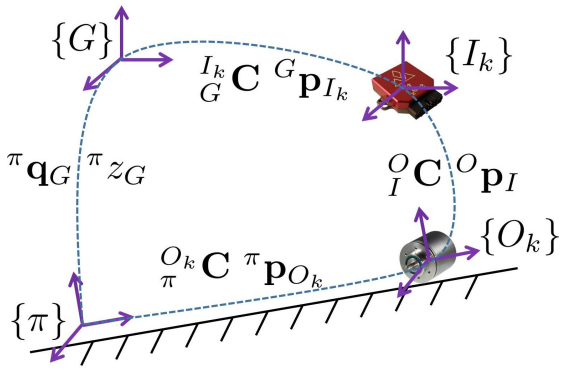


Figure 3.3: Geometric relationship between the IMU, $\{I\}$, odometer, $\{O\}$, and plane, $\{\pi\}$, frames when the robot moves on the plane, at time step k.

In what follows, we focus on a specific manifold, the one corresponding to planar motion, and present in detail how to employ this information in VINS. The frame of the plane, $\{\pi\}$, is defined so that the $x-y$ plane coincides with the physical plane. We parameterize the plane with a 2-dof quaternion, ${}^{\pi}\mathbf{q}_G$, representing the orientation between the plane frame and the global frame, and a scalar ${}^{\pi}z_G$, denoting the perpendicular distance from the origin of the global frame to the plane. The error state of the quaternion ${}^{\pi}\mathbf{q}_G$ is defined as a 2×1 vector $\delta\boldsymbol{\theta}_\pi$ so that the error quaternion is given by $\delta\mathbf{q} \triangleq \begin{bmatrix}\frac{1}{2}\delta\boldsymbol{\theta}_\pi^T & 0 & 1\end{bmatrix}^T$. Note that our parameterization agrees with the fact that a plane in the 3D space has 3 dof. As depicted in Fig. 3.3, we can express the geometric constraint of the odometer frame, $\{O\}$, moving within the plane, as:

$$\mathbf{g}(\mathbf{x}) = \begin{bmatrix} \Lambda^O_I \mathbf{C}^{I_k}_G \mathbf{C}^\pi_G \mathbf{C}^T \mathbf{e}_3 \\ {}^{\pi}z_G + \mathbf{e}_3^T {}^{\pi}_{G}\mathbf{C}({}^{G}\mathbf{p}_{I_k} - {}^{I_k}_{G}\mathbf{C}^T {}^{O}_{I}\mathbf{C}^T {}^{O}\mathbf{p}_I) \end{bmatrix} = \mathbf{0} \tag{3.20}$$

where the first block element (2×1 vector) corresponds to the planar rotational constraint, i.e., that the roll and pitch angles are zero between $\{\pi\}$ and $\{O\}$, while the second block element (scalar) corresponds to the planar translational constraint, i.e., that the position displacement along the z-axis is zero between $\{\pi\}$ and $\{O\}$.

Lastly, we provide the Jacobians of the planar model, derived from (3.20), employed by the VINS estimator:

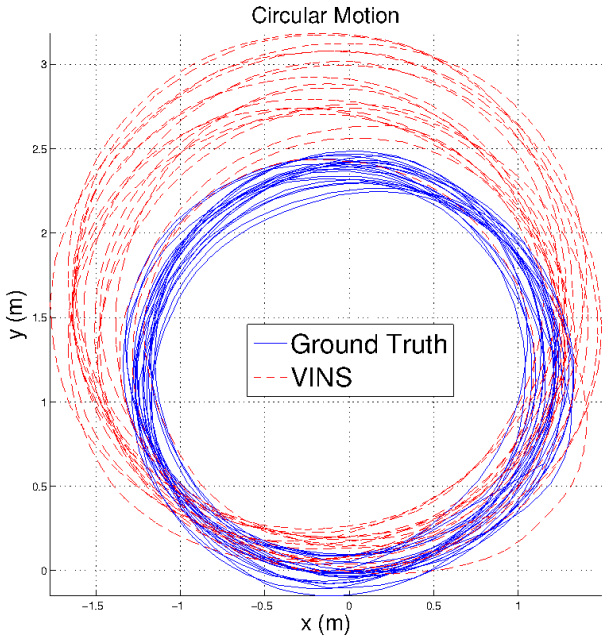


Figure 3.4: $x-y$ overview of the Pioneer robot's trajectory during the circular-motion experiment: The ground truth is shown in blue solid line, while the VINS estimate is shown in red dashed line.

i) Rotational component:

$$\begin{aligned}\mathbf{H}_{\delta\theta_{I_k}} &= \Lambda_I^O \mathbf{C} \lfloor {}^{I_k}_{G}\hat{\mathbf{C}}\, {}^{\pi}_{G}\hat{\mathbf{C}}^T \mathbf{e}_3 \rfloor \\ \mathbf{H}_{\delta\theta_{\pi}} &= \Lambda_I^O \mathbf{C}\, {}^{I_k}_{G}\hat{\mathbf{C}}\, {}^{\pi}_{G}\hat{\mathbf{C}}^T \begin{bmatrix} -\mathbf{e}_2 & \mathbf{e}_1 \end{bmatrix}\end{aligned} \tag{3.21}$$

ii) Translational component:

$$\begin{aligned}\mathbf{H}_{\delta\theta_{I_k}} &= \mathbf{e}_3^T\, {}^{\pi}_{G}\hat{\mathbf{C}}\, {}^{I_k}_{G}\hat{\mathbf{C}}^T \lfloor {}^{O}_{I}\mathbf{C}^{TO}\mathbf{p}_I \rfloor \\ \mathbf{H}_{\delta\theta_{\pi}} &= \left({}^{G}\hat{\mathbf{p}}_{I_k} - {}^{I_k}_{G}\hat{\mathbf{C}}\, {}^{TO}_{I}\mathbf{C}^{TO}\mathbf{p}_I\right)^T {}^{\pi}_{G}\hat{\mathbf{C}}^T \begin{bmatrix} -\mathbf{e}_2 & \mathbf{e}_1 \end{bmatrix} \\ \mathbf{H}_{p_{I_k}} &= \mathbf{e}_3^T\, {}^{\pi}_{G}\hat{\mathbf{C}}\,, \quad \mathbf{H}_{z_G} = 1.\end{aligned} \tag{3.22}$$

3.5 Experimental Results

We aim to examine the impact of different motions on the localization accuracy of VINS, as well as to validate the proposed methods for incorporating information from the odometer and the motion manifold. Note that our observability findings and the proposed methods are generic and not restricted to any particular VINS estimator. In our experiments, we chose the square-root inverse sliding window filter (SR-ISWF) [89] that is implemented with single-precision data types, in order to obtain highly-efficient localization results on mobile devices.[6]

Our testing platform involves commercial-grade sensors and CPU: A Pioneer 3 DX robot,[7] with a Project Tango tablet [7] mounted on it for visual and inertial sensing, as well as for processing. This tablet has a 2.3 GHz quad-core NVIDIA Tegra K1 CPU and 4 GB on-chip RAM, and is able to record: (i) MEMS-based IMU data, at 100 Hz, and (ii) Grayscale images from its wide field-of-view camera, with a resolution of 640×480, at 30 Hz. Around 200 FAST corners [78] are extracted from each image and tracked using the Kanade-Lucas-Tomasi (KLT) algorithm [64] at a frequency of 15 Hz. Then, a 2-pt RANSAC [55] is used for initial outlier rejection. The SR-ISWF estimator maintains a sliding window of 15 poses, which are selected at a frequency of about 7 Hz (depending on the motion).

3.5.1 Assessment of the Motion's Impact

We compare the localization results of the VINS, within the same environment, between two motion profiles: (i) Generic motion, where we hand-hold the tablet and walk regularly, and (ii) Constant (local) acceleration motion, where the tablet is mounted on the Pioneer robot that follows a circular motion. Fig. 3.4 illustrates the VICON[8] ground truth and the VINS filter's estimated trajectory. Note that as expected in practice, the vehicle's path (ground truth) is not a perfect circle; Instead, it only approximately follows one of the special motions considered here. Regardless, as evident from Fig. 3.4, significant scale error appears in the VINS estimates, which validates the conclusion of Thm. 1. We further compare the *scale ratio* between the two motions considered, as shown in Fig. 3.5. The scale ratio is computed as the estimated distance

[6] Similar results were observed when using the native Google Tango [7] VINS onboard the tablet.
[7] http://www.mobilerobots.com/ResearchRobots/PioneerP3DX.aspx
[8] http://www.vicon.com/

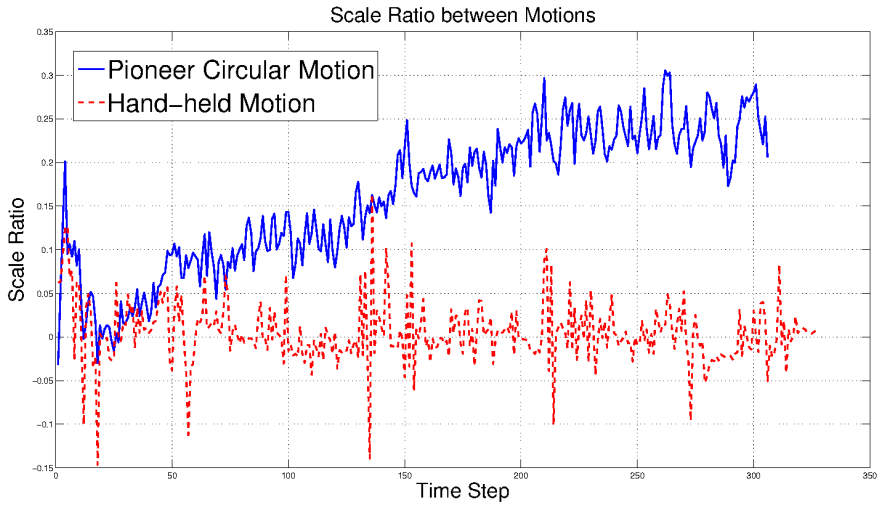


Figure 3.5: Scale ratio results for: (i) Pioneer circular motion (blue solid line) with mean 0.16 and std 0.08; (ii) Hand-held motion (red dashed line) with mean 3e-3 and std 0.03.

Table 3.1: Comparisons: Positioning RMSE (in Meters) of Different Methods Across Datasets (xy - z - **xyz** - %)

DS	Path (m)	VINS	VO	VOD	VOS
1	1080	9.8 - 1.5 - **9.9** - 0.91%	2.4 - 1.6 - **2.9** - 0.27%	4.3 - 0.1 - **4.3** - 0.4%	2.7 - 0.08 - **2.7** - 0.25%
2	876	13.8 - 1.1 - **13.8** - 1.6%	1.9 - 1.2 - **2.2** - 0.26%	4.5 - 0.14 - **4.5** - 0.52%	1.9 - 0.09 - **1.9** - 0.22%
3	954	8.3 - 1.2 - **8.4** - 0.88%	3.2 - 1.5 - **3.5** - 0.37%	7.8 - 0.22 - **7.8** - 0.82%	3.1 - 0.11 - **3.1** - 0.32%
4	1048	11.7 - 0.99 - **11.7** - 1.1%	3.7 - 1.0 - **3.8** - 0.37%	7.6 - 0.26 - **7.6** - 0.73%	3.6 - 0.07 - **3.6** - 0.34%
5	1034	9.7 - 0.99 - **9.8** - 0.94%	1.6 - 1.4 - **2.1** - 0.2%	3.2 - 0.12 - **3.2** - 0.31%	1.6 - 0.08 - **1.6** - 0.15%

between consecutive poses divided by that of the ground truth, and shifted by one:

$$SR = \frac{d_{est}}{d_{gt}} - 1 \tag{3.23}$$

which measures the quality of the scale estimates, i.e., the closer this quantity is to zero, the better is the scale estimate. As evident, the scale ratio corresponding to the hand-held motion stays around zero, while that of the circular motion drifts away. Finally, the positioning root mean square error (RMSE) of the hand-held vs. circular motion is 14 cm vs. 81 cm, respectively. These results confirm that when a vehicle undergoes (even approximately) special motions, the reduced information available to the VINS along the unobservable directions significantly degrades the localization accuracy of the corresponding estimator.

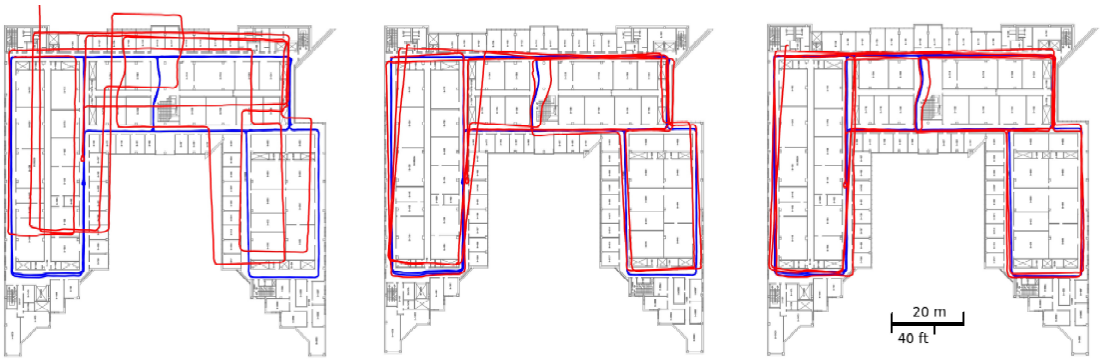


Figure 3.6: Illustration of the indoor Pioneer navigation trajectories, shown in red, estimated by the VINS only (left), the VOD (middle), and the VOS (right), overlayed on the floor plan. The ground truth, computed from the BLS method offline, is shown in blue.

3.5.2 System Performance Test

We further test the localization accuracy of our system on the Pioneer robot. Five datasets are collected by driving the Pioneer each time for $\sim$ 1 km through a large building. In addition to the IMU-camera data, the Pioneer wheel encoders provide readings at 10 Hz. We compare the localization results among the following setups: (i) VINS only, (ii) VINS plus odometer (VO), (iii) VINS plus odometer plus deterministic planar constraint (VOD), and (iv) VINS plus odometer plus stochastic planar constraint (VOS). The ground truth is computed from the batch least squares (BLS) offline, using all available (visual, inertial, and odometric) measurements.

Fig. 3.6 illustrates the estimated trajectories, overlayed on the building's floor plan as reference. As evident, the pure VINS suffers from very large errors due to the restricted motion (mostly constant-speed, on straight lines), while as more information becomes available, the positioning accuracy improves significantly. Also, the VOS outperforms the VOD, since the stochastic constraint better models the *approximately* planar motion due to the vibrations of the moving platform and the unevenness of the ground surface. Table 3.1 compares quantitatively the positioning error between different methods across all datasets (DS), where each block contains the following RMSE (in meters) results: xy - z - xyz total position - as percentage of the total distance traveled. From these results, we draw the following conclusions: First, between VO and VINS, when the odometer measurements are added, the x-y positioning accuracy is improved dramatically, since more scale information is injected. Second, by comparing VOD

and VOS to VINS and VO, it is evident that the planar motion constraints improve mostly the estimates in the z direction, as the error along the perpendicular direction is restricted by the constraint. Lastly, the stochastic constraint of VOS consistently improves the positioning accuracy, while the deterministic one of VOD has a negative impact, due to its modeling error.

In terms of efficiency, our system runs in real time on the tablet. Specifically, the whole VINS pipeline is taking 68 msec per cloned pose, including the 36 msec spent on the SR-ISWF filter update. Note also that our efficient implementation of the proposed methods (for processing odometer data and planar constraints) takes less than 1 msec for each. Overall, $\sim 50\%$ of the total CPU is used by our program when performing updates at ~ 7 Hz.

Finally, it is worth mentioning that our system is able to work robustly in both indoor and outdoor environments. Demonstrating videos are available at: http://mars.cs.umn.edu/research/VINSodometry.php

3.6 Summary

In this chapter, we proved that the VINS scale, or 2 additional dof of its global orientation, become unobservable when the robot moves with constant acceleration, or it is not rotating, respectively. For this reason, and as demonstrated in our experiments, directly employing VINS on a wheeled robot results in inaccurate pose estimates. To address this issue, we incorporated wheel-encoder measurements into VINS and showed that the scale becomes observable. Furthermore, we introduced mVINS that properly models the ground robot's almost-planar motion and directly employs this information in the estimator. Experimental results showed that special motions indeed lead to larger positioning errors when using VINS on a wheeled robot. Incorporating, however, odometry measurements, as well as stochastic constraints modeling the vehicle's planar motion, provide additional information and lead to significant improvements in positioning accuracy.

Chapter 4

Efficient and Consistent Long-Term Visual-Inertial Localization and Mapping

In this chapter, we address the problem of long-term visual-inertial localization and mapping, aka the full simultaneous localization and mapping (SLAM) that enables global adjustment with loop closures. In particular, we present the RISE-SLAM algorithm for performing efficient visual-inertial SLAM, while improving estimation consistency. Specifically, in order to achieve real-time operation, existing approaches often assume previously-estimated states to be perfectly known, which leads to inconsistent estimates. Instead, based on the idea of the Schmidt-Kalman filter, which has processing cost linear in the size of the state vector but quadratic memory requirements, we derive a new consistent approximate method in the information domain, which has linear memory requirements and adjustable (constant to linear) processing cost. In particular, this method, the resource-aware inverse Schmidt estimator (RISE), allows trading estimation accuracy for computational efficiency. Furthermore, and in order to better address the requirements of a SLAM system during an exploration vs. a relocalization phase, we employ different configurations of RISE (in terms of the number and order of states updated) to maximize accuracy while preserving efficiency. Lastly, we evaluate the proposed RISE-SLAM algorithm on publicly-available datasets and demonstrate its superiority, both in terms of accuracy and efficiency, as compared to alternative visual-inertial SLAM systems.

4.1 Introduction and Related Work

Simultaneous localization and mapping (SLAM) is necessary in a wide range of applications, such as robot navigation in GPS-denied areas, autonomous driving, and augmented/virtual reality. Recently, successful vision-only SLAM systems have emerged that employ one or multiple cameras [30, 54, 71]. Another popular choice is to combine the visual information with inertial data, from an inertial measurement unit (IMU), for increased robustness and accuracy [58, 63, 69, 72, 77]. In both cases, it is well known that under certain assumptions, finding the Maximum a Posteriori (MAP) estimate for SLAM can be cast as a nonlinear batch least-squares (BLS) problem, and the optimal solution, for the camera poses and feature positions, can be obtained in either a batch [24, 87] or an incremental [51, 52] form. These optimal approaches, however, have an increasing processing cost with time, typically between linear and quadratic in the number of poses and features, and thus cannot provide high-frequency estimates when operating inside large areas. On the other end of the spectrum, visual(-inertial) odometry systems [16, 28, 29, 35, 42, 58, 69, 89] focus their optimization over only a bounded sliding window of recent poses. The latency of these methods is typically very low and does not increase with time, but this comes at the expense of an ever-increasing drift in the pose estimates, due to their inability to process loop-closure measurements and perform global adjustment.

In order to achieve accuracy and efficiency at the same time, recent visual(-inertial) SLAM systems aim to combine the advantages of both the optimal (global) and the odometry (local) approaches, by employing a multi-thread scheme [54, 63, 71, 77]: A frontend thread estimates the current or several recent poses (as well as a local map) in constant time for real-time performance, while a backend thread optimizes, at a higher cost and lower frequency, over the entire trajectory (using either the optimal BLS [24] or its approximations [56, 74, 84]), and generates more accurate keyframe pose estimates and global maps for relocalization. To limit the processing cost, however, all these approaches employ approximations, e.g., keyframes involved in the frontend's relocalization are assumed to be *perfectly known*. Ignoring the corresponding uncertainties of these states and their cross correlations with the current states, however, leads to *inconsistent* estimates.[1] This means that the estimated covariance is unduly small and does

[1] As defined in [48, 49], a state estimator is consistent if the estimation errors are zero-mean and have covariance matrix smaller than or equal to the one calculated by the estimator. For the purposes of this work, we focus on the covariance requirement. Note that there exist additional sources of inconsistency, due to linearization errors and local minima (see e.g., [47]). In this work, we focus on the inconsistency caused by the assumption that uncertain quantities, such as a map, are perfectly known.

not represent correctly the uncertainty of the current state estimates (i.e., it does not offer a reliable measure of the tracking quality). More importantly, combining these overly optimistic estimates with new measurements later on can further degrade the accuracy of the system, as new, precise measurements are weighted less in favor of the current estimates. In fact, this problem of inconsistency has been acknowledged in the past, and remedies are often used to alleviate its negative impact on estimation accuracy, e.g., by inflating the assumed covariance of the noise corresponding to the relocalization visual observations [65, 70]. These heuristics, however, offer no guarantees on the estimation consistency or the system's performance.

On the other hand, the sparse extended information filter (SEIF) of [83, 88] is a consistent approximate SLAM algorithm, whose cost (between linear and cubic in the map's size) though for recovering the state estimate from the information vector makes it prohibitive for real-time operation. Specifically, although approximations involving early-terminating iterative solvers reduce processing during exploration, the required number of iterations for loop closures makes the cost often larger than that of direct solvers.

At this point, we should note that there exists a consistent approximation in the *filtering* domain: The Schmidt-Kalman filter (SKF) [80]. The key idea of the SKF is to update *optimally* only a subset of the states (e.g., recent poses and features) and their corresponding covariance and cross correlation terms, while leaving the rest (e.g., past poses and features) unaltered. By doing so, the computational cost is reduced from quadratic to linear in the (potentially very large) size of unchanged states. Meanwhile, the uncertainty of the past states is correctly accounted for to guarantee consistent estimates. The SKF and its variants have been applied to the SLAM problem [38, 50, 73], where their major drawback is their high memory requirements: Quadratic in the size of *all* states due to the dense covariance matrix. Thus, the SKF cannot be employed in large-scale SLAM. On the other hand, it is well-known that the information-domain solutions are more suitable for large-scale SLAM, as the Hessian matrix and its corresponding Cholesky factor are sparse [87]. To leverage this fact, [27] adapted the SKF to incorporate a previously-computed sparse Cholesky factor of a given map's Hessian. The approach, however, is a filtering one, and can only be used to perform *map-based localization* given an *offline-built* map, but not SLAM.

Motivated by the potential processing savings of the SKF, as well as the low-memory requirements of the Hessian (or equivalently its Cholesky factorization) representation of the uncertainty, in this work, we seek to derive a Schmidt-type estimator in the *information* domain,

that we can apply to the SLAM problem. To do so, we first derive the exact equivalent of the SKF in its square-root inverse form, i.e., by maintaining the Cholesky factor of the Hessian, since the corresponding portion of the information factor does not change [90]. Surprisingly, unlike the case of the SKF, the exact inverse-form Schmidt estimator does *not* provide any computational savings as compared to the optimal solver [53]. Moreover, the involved operations introduce a large number of fill-ins, leading to an almost dense information factor. This eventually makes the system too slow, and hence unsuitable for real-time long-term SLAM.

To overcome these limitations, we further introduce the resource-aware inverse Schmidt estimator (RISE), which is derived as a further approximation of the exact inverse Schmidt estimator [90]. The key idea behind RISE is to drop a certain portion of the available information, so that: i) As in the exact inverse Schmidt, past states as well as their corresponding portion of the information factor remain unaltered, while at the same time, ii) Recent states are updated only *approximately*, instead of *optimally*, so as to reduce both the processing cost and the factor fill-ins. Hence, RISE achieves both computational and memory efficiency by keeping the information factor sparse. Meanwhile, it is a *consistent* approximation to the optimal approach, as it only drops information, instead of assuming any state to be perfectly known. More importantly, RISE allows trading accuracy for efficiency, by adjusting the size of the window of the states selected to be updated. In the extreme case when all states are chosen for update, RISE becomes exactly equivalent to the optimal solver without any information loss.

Furthermore, we employ the proposed RISE algorithm in various configurations to realize an accurate and efficient visual-inertial SLAM system, the RISE-SLAM, which maintains a *consistent* sparse information factor corresponding to all estimated states. Specifically, our system alternates between two modes, *exploration* and *relocalization*, based on the availability of loop-closure measurements. In order to balance between accuracy and efficiency, in each mode, RISE is employed with various window sizes and different state orders. Similarly to most recent SLAM systems, our implementation incorporates two threads running in parallel: A fast frontend thread for estimating the current poses and features at a high frequency, and a lower-rate backend thread for globally adjusting the past states to achieve high accuracy. A key difference, however, as compared to existing systems that solve *multiple* optimization problems *independently* in different threads [63, 71, 77], is that RISE-SLAM always solves a *single* optimization problem, partitioned into two components each assigned to one of the two threads. This is only possible because of the structure of RISE, whose approximation allows focusing resources on

only a subset of states at a time. As a result, in our system, important global corrections from the backend are *immediately* reflected onto the frontend estimates, hence improving the current tracking accuracy. In summary, our main contributions are:

- To the best of our knowledge, we derive the first equivalent of the Schmidt-Kalman filter in its inverse form, the exact inverse Schmidt estimator (exact ISE).
- We derive the resource-aware inverse Schmidt estimator (RISE), which approximates the exact inverse Schmidt and has adjustable processing cost, while preserving sparsity and ensuring consistency.
- We introduce RISE-SLAM, for building 3D maps and relocalizing within previously-mapped areas in a consistent manner with constant cost.
- We implement RISE-SLAM and assess its performance. As compared to state-of-the-art approaches, our algorithm achieves the best performance in terms of estimation accuracy and processing time.

The rest of the chapter is structured as follows: Sec. 4.2 presents in detail our derivation of the exact inverse Schmidt estimator, as well as its approximation, the resource-aware inverse Schmidt estimator. Then, in Sec. 4.3, we apply these inverse Schmidt estimators to the problem of visual-inertial SLAM, and describe our RISE-SLAM algorithm. Finally, Sec. 4.4 concludes the chapter.

4.2 Inverse Schmidt Estimators

The Kalman filter (KF) and its information-domain equivalent, inverse filter (IF), are popular algorithms for localization and navigation applications [68]. And in order to improve the numerical stability, their square-root forms have been developed, i.e., the square-root Kalman filter (SR-KF) and the square-root inverse filter (SR-IF) [14]. When nuisance parameters exist, the Schmidt-Kalman filter (SKF) [80] can be used for reducing the dimensionality of the state estimate, while still considering the uncertainty of these parameters and ensuring the correctness of the covariance matrix, i.e., to guarantee estimation consistency. In this section, we introduce the SKF's information-domain equivalent in its square-root form, i.e., the square-root inverse

Schmidt estimator (SR-ISE or ISE). To the best of our knowledge, this is the first time that the information form of the SKF has been developed.

Furthermore, we provide complexity analysis of the exact ISE, and identify its limitations in terms of computational efficiency. In order to reduce the processing cost even more, we introduce further approximations based on the exact ISE and obtain other consistent alternatives. One important outcome is the resource-aware inverse Schmidt estimator (RISE), that enables trading estimation accuracy for computational efficiency, and hence is more flexible when employed to provide sufficiently accurate solutions with low processing and memory cost at the same time.

This section is organized as follows: We start by providing background knowledge in Sec. 4.2.1 on estimation consistency, which is an important concept ensured by the Schmidt approach. The problem of finding the exact equivalent of the Schmidt-Kalman filter (SKF) in the information domain is formulated in Sec. 4.2.2. Then, Sec. 4.2.3 presents in detail a complete derivation of the exact inverse Schmidt estimator (ISE) algorithm, as well as its complexity analysis. Further approximations of the exact ISE, for gaining in computational efficiency, are introduced in Sec. 4.2.4, including the resource-aware inverse Schmidt estimator (RISE).

4.2.1 Background: Estimation Consistency

Definition of Estimation Consistency

As defined in [48, 49], a state estimator is consistent if the estimation errors are zero-mean and have covariance matrix smaller than or equal to the one calculated by the estimator. For the purposes of this work, we focus on the covariance requirement, while the zero-mean error requirement is typically satisfied in general. In other words, an estimator is consistent, if the estimated covariance (or the inverse of the estimated Hessian matrix for estimators in the information form) is greater than or equal to, in the matrix positive-semidefinite sense, the true covariance computed by an optimal approach (e.g., KF or IF).

Consistency is an important concept in the theory of estimation, as it is a fundamental property of an estimator, with inconsistency typically indicating a bad performance or even failure of the estimator. Specifically, according to the definition, inconsistency means that the estimated covariance (or its information equivalent) is overly confident and does not represent correctly the uncertainty of the current estimate. Hence, the estimated covariance does not offer

a reliable measure of the quality of the state estimate. More importantly, combining these overly optimistic estimates with new measurements later on will further degrade the accuracy of the results.

Inconsistent Approximations

In practice, however, in order to limit the computational cost and achieve efficient solutions, approximations that lead to inconsistent estimates are commonly used. For example, in the literature of simultaneous localization and mapping (SLAM), most state-of-the-art systems employ approximations where some previously-estimated states are assumed to be perfectly known, such as past keyframes involved in the system's frontend thread during relocalization. Apparently, assuming some imperfect states as perfectly-known ignores the uncertainty of these states, and hence the estimated covariance based on this assumption will become falsely optimistic as compared to the true uncertainty. Therefore, these approaches generate inconsistent estimates according to the definition, and the estimation accuracy will suffer, in exchange for faster processing speed.

In fact, the problem of inconsistency has been acknowledged in the SLAM community in the past, and remedies are often used to alleviate its negative impact on estimation accuracy, e.g., by artificially inflating the covariance of the noise corresponding to the relocalization visual observations [65, 70]. These heuristics, however, offer no guarantees on the estimation consistency or the system's performance. To address this issue, in this work, we focus on consistent approximations and present novel estimators that ensure estimation consistency.

4.2.2 Problem Formulation

Kalman Filter (KF) and Schmidt-Kalman Filter (SKF)

The standard Kalman filter (KF) estimates a state vector $\mathbf{x} \in \mathbb{R}^{n\times 1}$ and the corresponding error covariance matrix $\mathbf{P} \in \mathbb{R}^{n\times n}$. It is optimal in the minimum mean-squared-error (MMSE) sense, and has a quadratic complexity in the size of all estimated states, in both computational and memory requirements. To achieve more efficient solutions, approximations have been introduced to the KF. One of such existing methods is the Schmidt-Kalman filter (SKF) [80]. Specifically, when nuisance parameters exist, the SKF reduces the dimensionality of the estimated state, and hence lowers the computational cost down to linear. Meanwhile, it considers

the uncertainty of these parameters and ensures the correctness of the updated covariance, i.e., to guarantee estimation consistency.

Specifically, if $\mathbf{x}$ is splitted into

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^T & \mathbf{x}_2^T \end{bmatrix}^T, \quad \text{with } \mathbf{x}_1 \in \mathbb{R}^{n_1 \times 1}, \ \mathbf{x}_2 \in \mathbb{R}^{n_2 \times 1}, \ \text{and } n_1 + n_2 = n \tag{4.1}$$

and correspondingly for the covariance matrix:

$$\mathbf{P} = \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{21} & \mathbf{P}_{22} \end{bmatrix}, \quad \text{with } \mathbf{P}_{11} \in \mathbb{R}^{n_1 \times n_1}, \ \mathbf{P}_{21}^T = \mathbf{P}_{12} \in \mathbb{R}^{n_1 \times n_2}, \ \text{and } \mathbf{P}_{22} \in \mathbb{R}^{n_2 \times n_2} \tag{4.2}$$

where $\mathbf{x}_1$ consists of the states of interest (to be updated) and $\mathbf{x}_2$ consists of all other states (not to be updated), then the SKF performs the following update:

$$\mathbf{S} = \mathbf{H}\mathbf{P}\mathbf{H}^T + \mathbf{I}_m \tag{4.3}$$

$$\mathbf{K} = \mathbf{P}\mathbf{H}^T\mathbf{S}^{-1} \triangleq \begin{bmatrix} \mathbf{K}_1 \\ \mathbf{K}_2 \end{bmatrix}, \quad \text{with } \mathbf{K}_1 \in \mathbb{R}^{n_1 \times m} \text{ and } \mathbf{K}_2 \in \mathbb{R}^{n_2 \times m} \tag{4.4}$$

$$\hat{\mathbf{x}}^s = \begin{bmatrix} \hat{\mathbf{x}}_1 + \mathbf{K}_1\mathbf{r} \\ \hat{\mathbf{x}}_2 \end{bmatrix} \tag{4.5}$$

$$\mathbf{P}^s = \begin{bmatrix} \mathbf{P}_{11} - \mathbf{K}_1\mathbf{S}\mathbf{K}_1^T & \mathbf{P}_{12} - \mathbf{K}_1\mathbf{S}\mathbf{K}_2^T \\ \mathbf{P}_{21} - \mathbf{K}_2\mathbf{S}\mathbf{K}_1^T & \mathbf{P}_{22} \end{bmatrix} \tag{4.6}$$

where $\mathbf{H} \in \mathbb{R}^{m \times n}$ denotes the pre-whitened measurement Jacobian (so that the measurement noise has covariance equal to the $m \times m$ identity matrix $\mathbf{I}_m$), $\mathbf{r} \in \mathbb{R}^{m \times 1}$ the pre-whitened measurement residual, $\mathbf{S}$ the residual covariance, $\mathbf{K}$ the Kalman gain, and $\hat{\mathbf{x}}$ the prior state estimate, respectively. Note that, after the SKF update, $\hat{\mathbf{x}}_1$, $\mathbf{P}_{11}$, and $\mathbf{P}_{12}$ become exactly the same as the result of the standard KF update, while $\hat{\mathbf{x}}_2$ and $\mathbf{P}_{22}$ remain unchanged. This way, the SKF updates only the states of interest and the corresponding covariance blocks, and hence, achieves computational savings as compared to the KF.

The SKF is an approximation to the KF (or equivalently the SR-IF), in the sense that it drops a certain amount of available information, i.e., the portion corresponding to $\mathbf{x}_2$. This can

be shown analytically as follows: The posterior covariance after the KF update is given by:

$$\mathbf{P}^{\oplus} = \mathbf{P} - \mathbf{K}\mathbf{S}\mathbf{K}^T \tag{4.7}$$

Therefore, the SKF covariance update in (4.6) can be written as:

$$\mathbf{P}^s = \mathbf{P}^{\oplus} + \bar{\mathbf{K}}_2 \mathbf{S} \bar{\mathbf{K}}_2^T, \quad \text{with } \bar{\mathbf{K}}_2 \triangleq \begin{bmatrix} \mathbf{0}_{n_1 \times m} \\ \mathbf{K}_2 \end{bmatrix} \tag{4.8}$$

and hence, the corresponding information is:

$$\begin{aligned} \boldsymbol{\mathcal{H}}^s \triangleq \mathbf{P}^{s^{-1}} &= (\mathbf{P}^{\oplus} + \bar{\mathbf{K}}_2 \mathbf{S} \bar{\mathbf{K}}_2^T)^{-1} = \mathbf{P}^{\oplus^{-1}} - \mathbf{P}^{\oplus^{-1}} \bar{\mathbf{K}}_2 (\mathbf{S}^{-1} + \bar{\mathbf{K}}_2^T \mathbf{P}^{\oplus^{-1}} \bar{\mathbf{K}}_2)^{-1} \bar{\mathbf{K}}_2^T \mathbf{P}^{\oplus^{-1}} \\ &= \boldsymbol{\mathcal{H}}^{\oplus} - \boldsymbol{\mathcal{H}}^{\oplus} \bar{\mathbf{K}}_2 (\mathbf{S}^{-1} + \bar{\mathbf{K}}_2^T \boldsymbol{\mathcal{H}}^{\oplus} \bar{\mathbf{K}}_2)^{-1} \bar{\mathbf{K}}_2^T \boldsymbol{\mathcal{H}}^{\oplus}, \quad \text{with } \boldsymbol{\mathcal{H}}^{\oplus} \triangleq \mathbf{P}^{\oplus^{-1}} \end{aligned} \tag{4.9}$$

If we define the Cholesky factorization of the following symmetric positive-definite (SPD) matrix:

$$\mathbf{S}^{-1} + \bar{\mathbf{K}}_2^T \boldsymbol{\mathcal{H}}^{\oplus} \bar{\mathbf{K}}_2 \triangleq \mathbf{L}_s \mathbf{L}_s^T \tag{4.10}$$

then (4.9) becomes:

$$\begin{aligned} \boldsymbol{\mathcal{H}}^s &= \boldsymbol{\mathcal{H}}^{\oplus} - \boldsymbol{\mathcal{H}}^{\oplus} \bar{\mathbf{K}}_2 \mathbf{L}_s^{-T} \mathbf{L}_s^{-1} \bar{\mathbf{K}}_2^T \boldsymbol{\mathcal{H}}^{\oplus} \\ &= \boldsymbol{\mathcal{H}}^{\oplus} - \mathbf{J}^{s^T} \mathbf{J}^s, \quad \text{with } \mathbf{J}^s \triangleq \mathbf{L}_s^{-1} \bar{\mathbf{K}}_2^T \boldsymbol{\mathcal{H}}^{\oplus} \in \mathbb{R}^{m \times n} \end{aligned} \tag{4.11}$$

which shows that the information term $\mathbf{J}^{s^T} \mathbf{J}^s$ is discarded during the SKF update. Due to this fact, the SKF is a *consistent* approximation of the KF [see (4.11)], i.e.,

$$\mathbf{P}^s \geq \mathbf{P}^{\oplus} \tag{4.12}$$

in the matrix positive-semidefinite sense.

On the other hand, note that, among all possible approximate algorithms that do not update $\mathbf{x}_2$, the SKF is the "best" one in the sense that all the information on $\mathbf{x}_1$ has been absorbed. This is obvious as the updated state estimate and covariance of $\mathbf{x}_1$ are exactly the same as those of the optimal KF.

When applied to the SLAM problem, similarly to the optimal KF, the major drawback of

the SKF is its high memory requirements: Quadratic in the size of *all* states due to the dense covariance matrix. Thus, the SKF cannot be employed in large-scale SLAM tasks.

Inverse Filter (IF)

The inverse filter (IF) is the information-domain equivalent of the KF, where the Hessian matrix $\boldsymbol{\mathcal{H}} = \mathbf{P}^{-1}$ is maintained and estimated instead of the covariance $\mathbf{P}$, and hence the name. For updates, it simply adds new information contribution terms from the measurements to the prior information, as:

$$\boldsymbol{\mathcal{H}}^{\oplus} = \boldsymbol{\mathcal{H}} + \mathbf{H}^T\mathbf{H} \tag{4.13}$$

And the state is updated by solving the normal equation using the Cholesky factorization of $\boldsymbol{\mathcal{H}}^{\oplus}$.

It is well-known that the information domain solutions are more suitable for large-scale SLAM, as the Hessian matrix and its corresponding Cholesky factor are sparse [87].

Square-Root Inverse Filter (SR-IF)

The square-root inverse filter (SR-IF) is the square-root form equivalent of the IF (hence equivalent to the KF as well), and it maintains the (upper-triangular) Cholesky factor $\mathbf{R} \in \mathbb{R}^{n\times n}$ of the Hessian matrix:

$$\mathbf{P}^{-1} = \boldsymbol{\mathcal{H}} = \mathbf{R}^T\mathbf{R} \tag{4.14}$$

At each update step, the SR-IF solves the following optimization problem:

$$\min_{\widetilde{\mathbf{x}}} \; ||\mathbf{R}\widetilde{\mathbf{x}} - \mathbf{r}_0||^2 + ||\mathbf{H}\widetilde{\mathbf{x}} - \mathbf{r}||^2 \tag{4.15}$$

where $\widetilde{\mathbf{x}} \triangleq \mathbf{x} - \hat{\mathbf{x}}$ denotes the error state. In the above cost function $\mathcal{C}$, the first term corresponds to the prior information, and the second term arises from the (linearized) measurement equations. Note that, the prior residual, $\mathbf{r}_0$, is nonzero if the linearization and solve are carried out iteratively, in order to reduce linearization errors.

To solve this least-squares problem, a QR factorization is performed on the stacked matrix

of the prior information factor $\mathbf{R}$ and the measurement Jacobian $\mathbf{H}$:

$$\begin{bmatrix}\mathbf{R}\\ \mathbf{H}\end{bmatrix} = \mathbf{Q}^{\oplus}\begin{bmatrix}\mathbf{R}^{\oplus}\\ \mathbf{0}_{m\times n}\end{bmatrix}, \quad \text{with } \mathbf{Q}^{\oplus^T}\mathbf{Q}^{\oplus} = \mathbf{Q}^{\oplus}\mathbf{Q}^{\oplus^T} = \mathbf{I}_{n+m} \tag{4.16}$$

where $\mathbf{R}^{\oplus} \in \mathbb{R}^{n\times n}$ is the upper-triangular factor of the QR factorization. In terms of the cost function, this update process can be interpreted as:

$$\mathcal{C} = ||\mathbf{R}\widetilde{\mathbf{x}} - \mathbf{r}_0||^2 + ||\mathbf{H}\widetilde{\mathbf{x}} - \mathbf{r}||^2 = \left\|\begin{bmatrix}\mathbf{R}\\ \mathbf{H}\end{bmatrix}\widetilde{\mathbf{x}} - \begin{bmatrix}\mathbf{r}_0\\ \mathbf{r}\end{bmatrix}\right\|^2 \tag{4.17}$$

$$= \left\|\mathbf{Q}^{\oplus^T}\begin{bmatrix}\mathbf{R}\\ \mathbf{H}\end{bmatrix}\widetilde{\mathbf{x}} - \mathbf{Q}^{\oplus^T}\begin{bmatrix}\mathbf{r}_0\\ \mathbf{r}\end{bmatrix}\right\|^2 \tag{4.18}$$

$$= \left\|\begin{bmatrix}\mathbf{R}^{\oplus}\\ \mathbf{0}_{m\times n}\end{bmatrix}\widetilde{\mathbf{x}} - \mathbf{Q}^{\oplus^T}\begin{bmatrix}\mathbf{r}_0\\ \mathbf{r}\end{bmatrix}\right\|^2 \tag{4.19}$$

$$= \left\|\mathbf{R}^{\oplus}\widetilde{\mathbf{x}} - \mathbf{r}_1^{\oplus}\right\|^2 + \left\|\mathbf{r}_2^{\oplus}\right\|^2, \quad \text{with } \begin{bmatrix}\mathbf{r}_1^{\oplus}\\ \mathbf{r}_2^{\oplus}\end{bmatrix} \triangleq \mathbf{Q}^{\oplus^T}\begin{bmatrix}\mathbf{r}_0\\ \mathbf{r}\end{bmatrix}, \ \mathbf{r}_1^{\oplus} \in \mathbb{R}^{n\times 1} \text{ and } \mathbf{r}_2^{\oplus} \in \mathbb{R}^{m\times 1} \tag{4.20}$$

where from (4.17) to (4.18) we have used the fact that multiplying the unitary matrix $\mathbf{Q}^{\oplus}$ (from the QR factorization) preserves the norm of a vector. By minimizing (4.20), the solution is given by:

$$\widetilde{\mathbf{x}} = \mathbf{R}^{\oplus^{-1}}\mathbf{r}_1^{\oplus} \tag{4.21}$$

and hence the updated state estimate and information factor are given by $\hat{\mathbf{x}}^{\oplus} = \hat{\mathbf{x}} + \widetilde{\mathbf{x}}$ and $\mathbf{R}^{\oplus}$, respectively. As mentioned before, this process can be repeated iteratively till convergence to reduce linearization errors.

As compared to the IF in the Hessian form, the SR-IF operates on the square-root factor domain, and the update employs QR factorization on the sqaure-root factor instead of Cholesky factorization on the Hessian. This provides better numerical stability and acuuracy, but at the expense of slower processing speed [37].

Inverse Schmidt Estimator (ISE) Problem Formulation

Motivated by the potential processing savings of the SKF, as well as the low-memory requirements of the Hessian (or equivalently its Cholesky factorization) representation of the uncertainty, in this work, we seek to derive the Schmidt estimator, but in the information domain. Specifically, our objective is to find the exact inverse Schmidt estimator (ISE), i.e., to derive an algorithm that is exactly equivalent to the SKF, while operating in the information domain.

As mentioned in the previous subsections, an estimator in the information domain can take two forms: Either the Hessian form (e.g., the IF) or the square-root form (e.g., the SR-IF). For the task of finding the inverse equivalent of the SKF, however, the Hessian form does not seem to provide any computational savings, because all the blocks of the Hessian matrix will need to be updated. To be specific, if we compute the inverse of the SKF's updated covariance matrix in (4.6), the corresponding updated Schmidt Hessian matrix will require changes in all of its four blocks, as compared to the prior Hessian before the Schmidt update. This is in contrast to the SKF, as the (potentially very large) portion of $\mathbf{P}_{22}$ remains unchanged, from which the SKF achieves computational savings. As a result, compared to the optimal IF, the Schmidt approximation in the Hessian form will not lead to any processing savings, while its solution is only suboptimal, and hence is meaningless.

Instead, as shown later, the Schmidt approximation preserves the corresponding portion ($\mathbf{R}_{22}$) of the square-root information factor, and hence may bring potential computational benefits due to the saved work for this portion. Therefore, in this work, we focus on finding the information-domain Schmidt estimators in the square-root inverse form. And from this point on, by saying "inverse Schmidt estimator", we refer to only the square-root inverse form estimator that operates on the Cholesky factor of the Hessian.

Under the square-root inverse form, the problem of finding the exact square-root inverse Schmidt estimator (SR-ISE, or simply ISE) can be formulated as follows: Given the (invertible and upper-triangular) prior information factor $\mathbf{R}$ and the measurement Jacobian $\mathbf{H}$:

$$\mathbf{R} = \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0}_{n_2 \times n_1} & \mathbf{R}_{22} \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{bmatrix} \tag{4.22}$$

where $\mathbf{R}_{11} \in \mathbb{R}^{n_1 \times n_1}$, $\mathbf{R}_{12} \in \mathbb{R}^{n_1 \times n_2}$, $\mathbf{R}_{22} \in \mathbb{R}^{n_2 \times n_2}$, $\mathbf{H}_1 \in \mathbb{R}^{m \times n_1}$, and $\mathbf{H}_2 \in \mathbb{R}^{m \times n_2}$, find

the algorithm that computes the updated Schmidt information factor $\mathbf{R}^s \in \mathbb{R}^{n\times n}$ (invertible and upper-triangular as well), such that it reflects the covariance matrix of the SKF [see (4.6)], i.e.,

$$\mathbf{P}^{s^{-1}} = \mathbf{R}^{s^T}\mathbf{R}^s, \quad \text{with } \mathbf{R}^s = \begin{bmatrix} \mathbf{R}^s_{11} & \mathbf{R}^s_{12} \\ \mathbf{0}_{n_2\times n_1} & \mathbf{R}^s_{22} \end{bmatrix} \tag{4.23}$$

with the same block sizes as defined in $\mathbf{R}$. Similarly for the state update: The updated state estimate $\hat{\mathbf{x}}^s$ should be equal to that of the SKF [see (4.5)]. Note that there are several basic underlying requirements of the algorithm: At any step, the algorithm (i) should not compute explicitly the covariance matrix, since we seek an inverse (information-domain) approach, and (ii) should not compute explicitly the information matrix, since we require the square-root form.

4.2.3 Exact Inverse Schmidt Estimator (ISE)

In this section, we present the detailed derivation of the exact ISE algorithm. First, we propose a generic framework that operates in the square-root information domain, inspired by the SKF and the SR-IF. Then, we derive several sufficient and necessary conditions, under which this framework becomes equivalent to the SKF. Last, based on one of these equivalent conditions, an efficient exact ISE algorithm is presented.

A Generic Framework

As a first step, our objective is to seek a square-root information-domain algorithm that shares two fundamental properties with the SKF: (i) The $\mathbf{x}_2$ part of the state vector is not updated during the Schmidt process, i.e., the state estimate, $\hat{\mathbf{x}}_2$, and the corresponding covariance block, $\mathbf{P}_{22}$, remain intact [see (4.5) and (4.6)]; (ii) It is a consistent approximation of the optimal SR-IF, i.e., the resulting inferred covariance is conservative as compared to that of the SR-IF [see (4.12)].

Our proposed algorithm follows the approach of the SR-IF, where a unitary transformation, defined by the QR factor $\mathbf{Q}^{\oplus}$, is performed on the cost function [see (4.18)]. In our algorithm, instead of employing this specific unitary matrix $\mathbf{Q}^{\oplus}$, a generic unitary matrix $\breve{\mathbf{U}}$ is used.

Specifically, in terms of the cost function to be minimized [see (4.17)]:

$$\mathcal{C} = ||\mathbf{R}\widetilde{\mathbf{x}} - \mathbf{r}_0||^2 + ||\mathbf{H}\widetilde{\mathbf{x}} - \mathbf{r}||^2 = \left\| \begin{bmatrix} \mathbf{R} \\ \mathbf{H} \end{bmatrix} \widetilde{\mathbf{x}} - \begin{bmatrix} \mathbf{r}_0 \\ \mathbf{r} \end{bmatrix} \right\|^2 \tag{4.24}$$

$$= \left\| \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{0} & \mathbf{R}_{22} \\ \mathbf{H}_1 & \mathbf{H}_2 \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{0}_{n_2 \times 1} \\ \mathbf{r} \end{bmatrix} \right\|^2 \tag{4.25}$$

$$= \left\| \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{r} \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 \tag{4.26}$$

$$= \left\| \breve{\mathbf{U}}^T \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{H}_1 & \mathbf{H}_2 \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} - \breve{\mathbf{U}}^T \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{r} \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 \tag{4.27}$$

where $\breve{\mathbf{U}} \in \mathbb{R}^{(n+m)\times(n+m)}$ is unitary, i.e., $\breve{\mathbf{U}}^T\breve{\mathbf{U}} = \breve{\mathbf{U}}\breve{\mathbf{U}}^T = \mathbf{I}_{n+m}$. Note that from (4.24) to (4.25), we have used the fact that $\mathbf{r}_0 = \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix}$, i.e., the prior residual has only a nonzero block corresponding to the $\mathbf{x}_1$ part, since $\mathbf{x}_2$ will not be updated in the process.

As mentioned earlier, the first objective is to preserve the state $\mathbf{x}_2$. The following lemma states the condition when this requirement is satisfied:

Lemma 1. *Given an invertible and upper-triangular prior information factor* $\mathbf{R}$ *as in* (4.22)*, and the corresponding prior covariance matrix* $\mathbf{P}$ *as in* (4.2)*. An algorithm updates the factor into another invertible and upper-triangular matrix* $\mathbf{R}'$*, with the corresponding covariance* $\mathbf{P}'$*. Then the covariance corresponding to* $\mathbf{x}_2$ *remains unchanged, i.e.,* $\mathbf{P}'_{22} = \mathbf{P}_{22}$*, if and only if the corresponding information factor remains unchanged, i.e.,* $\mathbf{R}'_{22} = \mathbf{R}_{22}$ *(up to the sign of each row).*

Proof. From (4.14) and (4.22), using block matrix inversion, the (2,2) block of the covariance matrix can be written in terms of the information factor blocks as:

$$\mathbf{P} = (\mathbf{R}^T\mathbf{R})^{-1} = \begin{bmatrix} \mathbf{R}_{11}^T\mathbf{R}_{11} & \mathbf{R}_{11}^T\mathbf{R}_{12} \\ \mathbf{R}_{12}^T\mathbf{R}_{11} & \mathbf{R}_{12}^T\mathbf{R}_{12} + \mathbf{R}_{22}^T\mathbf{R}_{22} \end{bmatrix}^{-1} \Rightarrow \mathbf{P}_{22} = \mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T} \tag{4.28}$$

Similarly, we have $\mathbf{P}'_{22} = \mathbf{R}'^{-1}_{22}\mathbf{R}'^{-T}_{22}$. Hence, $\mathbf{P}'_{22} = \mathbf{P}_{22}$ is equivalent to $\mathbf{R}'^{-1}_{22}\mathbf{R}'^{-T}_{22} = \mathbf{R}^{-1}_{22}\mathbf{R}^{-T}_{22}$, or equivalently $\mathbf{R}'^{T}_{22}\mathbf{R}'_{22} = \mathbf{R}^{T}_{22}\mathbf{R}_{22}$. Since both $\mathbf{R}'_{22}$ and $\mathbf{R}_{22}$ are invertible and upper-triangular matrices, they are the Cholesky factors of the same SPD matrix. Hence, $\mathbf{R}'_{22} = \mathbf{R}_{22}$ (up to the sign of each row) from the uniqueness of the Cholesky factorization of a SPD matrix. □

Therefore, from Lemma 1, in order to preserve $\mathbf{x}_2$, the updated (upper-triangular) information factor should take the form:

$$\mathbf{R}' = \begin{bmatrix} \mathbf{R}'_{11} & \mathbf{R}'_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \tag{4.29}$$

with the $\mathbf{R}_{22}$ block stays unchanged. Based on this key result, we propose the matrix $\breve{\mathbf{U}}$ to take the following form:

$$\breve{\mathbf{U}} = \begin{bmatrix} \mathbf{U} & \\ & \mathbf{I}_{n_2} \end{bmatrix} \tag{4.30}$$

where $\mathbf{U} \in \mathbb{R}^{(n_1+m)\times(n_1+m)}$ is a unitary matrix. Note that here $\breve{\mathbf{U}}$ is unitary if and only if $\mathbf{U}$ is unitary. Substituting (4.30) into (4.27), the cost function $\mathcal{C}$ becomes:

$$\mathcal{C} = \left\| \begin{bmatrix} \mathbf{U}^T & \\ & \mathbf{I}_{n_2} \end{bmatrix} \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{H}_1 & \mathbf{H}_2 \\ \cdots\cdots & \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{U}^T & \\ & \mathbf{I}_{n_2} \end{bmatrix} \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{r} \\ \cdots\cdots \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} \right\|^2 \tag{4.31}$$

Partition the columns of $\mathbf{U}$ as $\mathbf{U} = \begin{bmatrix} \mathbf{U}_1 & \mathbf{U}_2 \end{bmatrix}$, with $\mathbf{U}_1 \in \mathbb{R}^{(n_1+m)\times n_1}$ and $\mathbf{U}_2 \in \mathbb{R}^{(n_1+m)\times m}$,

and define:

$$\mathbf{R}'_{11} \triangleq \mathbf{U}_1^T \begin{bmatrix} \mathbf{R}_{11} \\ \mathbf{H}_1 \end{bmatrix} \in \mathbb{R}^{n_1 \times n_1}, \;\; \mathbf{R}'_{12} \triangleq \mathbf{U}_1^T \begin{bmatrix} \mathbf{R}_{12} \\ \mathbf{H}_2 \end{bmatrix} \in \mathbb{R}^{n_1 \times n_2} \tag{4.32}$$

$$\mathbf{J}_1 \triangleq \mathbf{U}_2^T \begin{bmatrix} \mathbf{R}_{11} \\ \mathbf{H}_1 \end{bmatrix} \in \mathbb{R}^{m \times n_1}, \;\; \mathbf{J}_2 \triangleq \mathbf{U}_2^T \begin{bmatrix} \mathbf{R}_{12} \\ \mathbf{H}_2 \end{bmatrix} \in \mathbb{R}^{m \times n_2}, \;\; \mathbf{J} \triangleq \begin{bmatrix} \mathbf{J}_1 & \mathbf{J}_2 \end{bmatrix} \in \mathbb{R}^{m \times n} \tag{4.33}$$

$$\mathbf{r}' \triangleq \mathbf{U}_1^T \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{r} \end{bmatrix} \in \mathbb{R}^{n_1 \times 1}, \;\; \mathbf{r}_J \triangleq \mathbf{U}_2^T \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{r} \end{bmatrix} \in \mathbb{R}^{m \times 1} \tag{4.34}$$

$$\implies \begin{bmatrix} \mathbf{R}'_{11} & \mathbf{R}'_{12} \\ \mathbf{J}_1 & \mathbf{J}_2 \end{bmatrix} = \mathbf{U}^T \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{H}_1 & \mathbf{H}_2 \end{bmatrix} \text{ and } \begin{bmatrix} \mathbf{r}' \\ \mathbf{r}_J \end{bmatrix} = \mathbf{U}^T \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{r} \end{bmatrix} \tag{4.35}$$

then the cost function $\mathcal{C}$ in (4.31) can be written as:

$$\mathcal{C} = \left\| \begin{bmatrix} \mathbf{R}'_{11} & \mathbf{R}'_{12} \\ \mathbf{J}_1 & \mathbf{J}_2 \\ \cdots\cdots & \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{r}' \\ \mathbf{r}_J \\ \cdots\cdots \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 \tag{4.36}$$

$$= \left\| \begin{bmatrix} \mathbf{R}'_{11} & \mathbf{R}'_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 + \left\| \begin{bmatrix} \mathbf{J}_1 & \mathbf{J}_2 \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_1 \\ \widetilde{\mathbf{x}}_2 \end{bmatrix} - \mathbf{r}_J \right\|^2 \tag{4.37}$$

$$= \left\| \mathbf{R}' \widetilde{\mathbf{x}} - \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 + \left\| \mathbf{J} \widetilde{\mathbf{x}} - \mathbf{r}_J \right\|^2 \tag{4.38}$$

with $\mathbf{R}'$ defined in (4.29). Note that, at this point, the cost function is still equal to the original cost $\mathcal{C}$ as in (4.17) (since any unitary transformation will not change the cost value), and minimizing it would generate the exact same update results as in the optimal SR-IF.

With the expression of the cost function $\mathcal{C}$ in (4.38), we are ready to introduce the necessary approximation step in our proposed algorithm, in order to imitate the SKF: The second term in the cost function is discarded. As a result, the new cost function $\mathcal{C}'$ consists of only the first

term in (4.38), i.e.,

$$\mathcal{C} = \left\| \mathbf{R}'\widetilde{\mathbf{x}} - \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 + \left\| \mathbf{J}\widetilde{\mathbf{x}} - \mathbf{r}_J \right\|^2 \quad \Longrightarrow \quad \mathcal{C}' = \left\| \mathbf{R}'\widetilde{\mathbf{x}} - \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 \tag{4.39}$$

which leads to the new optimization problem:

$$\min_{\widetilde{\mathbf{x}}} \ \left\| \mathbf{R}'\widetilde{\mathbf{x}} - \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \right\|^2 \tag{4.40}$$

and from which the solution (state correction) is given by:

$$\widetilde{\mathbf{x}}^\star = \mathbf{R}'^{-1} \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}'_{11} & \mathbf{R}'_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} = \begin{bmatrix} \mathbf{R}'^{-1}_{11}\mathbf{r}' \\ \mathbf{0}_{n_2 \times 1} \end{bmatrix} \tag{4.41}$$

and hence the state update can be written as:

$$\hat{\mathbf{x}}' = \hat{\mathbf{x}} + \widetilde{\mathbf{x}}^\star = \begin{bmatrix} \hat{\mathbf{x}}_1 + \mathbf{R}'^{-1}_{11}\mathbf{r}' \\ \hat{\mathbf{x}}_2 \end{bmatrix} \tag{4.42}$$

Clearly, only the $\mathbf{x}_1$ portion of the state vector has been changed, while $\mathbf{x}_2$ remains the same. Meanwhile, the updated information factor is given by $\mathbf{R}'$ [from (4.40)], and from its expression in (4.29), the block $\mathbf{R}_{22}$ stays unchanged, and hence, so as the corresponding covariance block $\mathbf{P}_{22}$ due to Lemma 1. Note that, in the above analysis, we have assumed that the updated factor block $\mathbf{R}'_{11}$ is invertible, which is a fundamental requirement for the proposed procedure to be a valid update.

Lastly, as desired, this approach is a consistent approximation to the optimal SR-IF, due to the step of discarding the second cost term in (4.38). Specifically, dropping a cost term corresponds to ignoring the information contained in it, and hence leads to less amount of information to be absorbed in the update, or equivalently larger uncertainty (covariance) in the updated estimates. This result matches the fact that the SKF drops information as well, as shown in (4.11), and hence justifies the necessity of this approximation step. The consistency of the proposed algorithm is stated formally in the following lemma:

Lemma 2. *Given an invertible prior information factor* $\mathbf{R}$ *and a measurement Jacobian* $\mathbf{H}$*. Let*

$\mathbf{P}^{\oplus}$ *denote the posterior covariance matrix given by an optimal approach (e.g., KF or SR-IF), and let* $\mathbf{P}'$ *denote the covariance matrix corresponding to the updated information factor* $\mathbf{R}'$ *in* (4.40)*. Then* $\mathbf{P}' \geq \mathbf{P}^{\oplus}$*, in the matrix positive-semidefinite sense.*

Proof. The posterior covariance matrix represents the uncertainty of the posterior state estimates considering all available information, i.e.,

$$\mathbf{P}^{\oplus} = (\mathbf{R}^T\mathbf{R} + \mathbf{H}^T\mathbf{H})^{-1} \tag{4.43}$$

and by the same arguments from (4.24) to (4.38), i.e., by introducing the unitary transformation defined in (4.30), the posterior covariance matrix is equal to:

$$\mathbf{P}^{\oplus} = (\mathbf{R}^T\mathbf{R} + \mathbf{H}^T\mathbf{H})^{-1} = (\mathbf{R}'^T\mathbf{R}' + \mathbf{J}^T\mathbf{J})^{-1} \tag{4.44}$$

Meanwhile, by definition, $\mathbf{P}' = (\mathbf{R}'^T\mathbf{R}')^{-1}$ from (4.40). Hence, by the matrix inversion lemma, we obtain:

$$\begin{aligned}\mathbf{P}' - \mathbf{P}^{\oplus} &= (\mathbf{R}'^T\mathbf{R}')^{-1} - (\mathbf{R}'^T\mathbf{R}' + \mathbf{J}^T\mathbf{J})^{-1} \\ &= (\mathbf{R}'^T\mathbf{R}')^{-1}\mathbf{J}^T\left[\mathbf{I}_m + \mathbf{J}(\mathbf{R}'^T\mathbf{R}')^{-1}\mathbf{J}^T\right]^{-1}\mathbf{J}(\mathbf{R}'^T\mathbf{R}')^{-1} \geq \mathbf{0}\end{aligned} \tag{4.45}$$

□

To summarize, our proposed approach first combines the prior knowledge with the measurement information [see (4.24)], then decomposes part of the combined information into two pieces by projecting it onto the two block columns of a unitary matrix [see (4.31) - (4.38)], and last drops one of these two resulting information pieces to obtain the final updated factor [see (4.39)]. This procedure is shown in Alg. 2. Note that, as mentioned before and written out in Alg. 2, we need to guarantee that the updated information factor is meaningful and structured, i.e., the final factor $\mathbf{R}'$ is invertible and upper-triangular. Based on the assumption that the input factor $\mathbf{R}$, and hence the $\mathbf{R}_{22}$ block, is invertible and upper-triangular, this requirement is satisfied if and only if the resulting $\mathbf{R}'_{11}$ block is invertible and upper-triangular. As a result, this requirement poses basic constraints on the matrix $\mathbf{U}_1$, in addition to having orthonormal columns.

To conclude, Alg. 2 satisfies the following desired properties (regardless of the choice of the

Algorithm 2 A Generic Framework

1: **Input:** Current state estimate $\hat{\mathbf{x}}$, prior information factor $\mathbf{R}$ and residual $\mathbf{r}_0$, pre-whitened measurement Jacobian $\mathbf{H}$ and residual $\mathbf{r}$
2: **procedure** UPDATE
3: Obtain a unitary matrix: $\mathbf{U} = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix}$, such that $\mathbf{R}'_{11}$ is invertible and upper-triangular
4: Perform the unitary transformation: $\mathbf{R}'_{11} \leftarrow \mathbf{U}_1^T \begin{bmatrix}\mathbf{R}_{11} \\ \mathbf{H}_1\end{bmatrix}$, $\mathbf{R}'_{12} \leftarrow \mathbf{U}_1^T \begin{bmatrix}\mathbf{R}_{12} \\ \mathbf{H}_2\end{bmatrix}$, $\mathbf{r}' \leftarrow \mathbf{U}_1^T \begin{bmatrix}\mathbf{r}_0^1 \\ \mathbf{r}\end{bmatrix}$
5: Information factor update: $\mathbf{R}' \leftarrow \begin{bmatrix}\mathbf{R}'_{11} & \mathbf{R}'_{12} \\ \mathbf{0} & \mathbf{R}_{22}\end{bmatrix}$
6: State update: $\hat{\mathbf{x}}' \leftarrow \begin{bmatrix}\hat{\mathbf{x}}_1 + \mathbf{R}'^{-1}_{11}\mathbf{r}' \\ \hat{\mathbf{x}}_2\end{bmatrix}$
7: **end procedure**
8: **Output:** Updated state estimate $\hat{\mathbf{x}}'$ and information factor $\mathbf{R}'$

unitary matrix $\mathbf{U}$): (i) It preserves the estimate and the information factor block of $\mathbf{x}_2$ (and hence the corresponding covariance block), and (ii) it is a consistent approximation of the optimal SR-IF. These are the two fundamental properties shared with the SKF. Note that, Alg. 2 presents a generic framework in the sense that it represents a family of algorithms by choosing a different matrix $\mathbf{U}$. One could potentially employ this framework to obtain an algorithm that generates an approximate solution and achieves computational savings, as compared to the SR-IF, due to the fact that $\mathbf{x}_2$ and $\mathbf{R}_{22}$ are not updated. In what follows, we show the procedure to obtain the specific $\mathbf{U}$ (either explicitly or implicitly) that leads to the exact square-root inverse Schmidt estimator, which was our initial goal. Later on, we also present alternative algorithms, special cases of this generic framework that have lower processing requirements and cause fewer fill-ins, when the factor is sparse.

Exact ISE: Sufficient and Necessary Conditions

Under the framework of Alg. 2, our objective is to find the specific matrix $\mathbf{U}$ such that this algorithm becomes the exact ISE, i.e., the updated state estimates and covariance matrix are equal to those of the SKF. Note that, since Alg. 2 is a square-root inverse approach, it does not compute explicitly the covariance matrix, but instead, the information factor. Hence, by saying the updated covariance of Alg. 2, we refer to the underlying covariance matrix corresponding to

the updated information factor, as can be computed using (4.14). Moreover, from this point on, we assume that the input prior factor $\mathbf{R}$ is invertible and upper-triangular, and will not repeat this assumption in our subsequent statements.

So far, we have considered only the equivalence with respect to the $\mathbf{x}_2$ portion of the entire state (i.e., the *unchanged* part). What remains to be accomplished is to establish the same equivalence with respect to $\mathbf{x}_1$ (i.e., the *changed*, or updated part). For this reason, we analyze the condition on the equivalence of the covariance blocks that are changed in the SKF update, i.e., the $(1,1)$, $(1,2)$, and $(2,1)$ blocks. The result is stated in the following lemma:

Lemma 3. *Let* $\mathbf{P}' \triangleq (\mathbf{R}'^T\mathbf{R}')^{-1}$ *denote the covariance matrix corresponding to the updated information factor* $\mathbf{R}'$ *of Alg. 2, and let* $\mathbf{P}^s$ *denote the updated covariance matrix of the SKF. Then*

$$\mathbf{P}'_{11} = \mathbf{P}^s_{11},\ \ \mathbf{P}'_{12} = \mathbf{P}^s_{12},\ \ \textit{and}\ \ \mathbf{P}'_{21} = \mathbf{P}^s_{21} \tag{4.46}$$

if and only if $\mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0}_{m\times n_1}$, *where* $\mathbf{P}'_{(:,1)} \triangleq \begin{bmatrix}\mathbf{P}'_{11}\\ \mathbf{P}'_{21}\end{bmatrix}$ *denotes the first block column of the matrix* $\mathbf{P}'$.

Proof. In the proof of Lemma 2, we have shown that [see (4.45)]:

$$\mathbf{P}' - \mathbf{P}^{\oplus} = (\mathbf{R}'^T\mathbf{R}')^{-1}\mathbf{J}^T\left[\mathbf{I}_m + \mathbf{J}(\mathbf{R}'^T\mathbf{R}')^{-1}\mathbf{J}^T\right]^{-1}\mathbf{J}(\mathbf{R}'^T\mathbf{R}')^{-1} \tag{4.47}$$

where $\mathbf{P}^{\oplus}$ denotes the posterior covariance after the KF update and $\mathbf{J}$ is defined in (4.33). Substituting $\mathbf{P}' = (\mathbf{R}'^T\mathbf{R}')^{-1}$ gives:

$$\mathbf{P}' - \mathbf{P}^{\oplus} = \mathbf{P}'\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\mathbf{P}' \tag{4.48}$$

If we partition the block columns of $\mathbf{P}'$ corresponding to $\mathbf{x}_1$ and $\mathbf{x}_2$ as $\mathbf{P}' = \begin{bmatrix}\mathbf{P}'_{(:,1)} & \mathbf{P}'_{(:,2)}\end{bmatrix}$, then we obtain:

$$\begin{aligned}\mathbf{P}' - \mathbf{P}^{\oplus} &= \begin{bmatrix}\mathbf{P}'^T_{(:,1)}\\ \mathbf{P}'^T_{(:,2)}\end{bmatrix}\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\begin{bmatrix}\mathbf{P}'_{(:,1)} & \mathbf{P}'_{(:,2)}\end{bmatrix}\\ &= \begin{bmatrix}\mathbf{P}'^T_{(:,1)}\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\mathbf{P}'_{(:,1)} & \mathbf{P}'^T_{(:,1)}\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\mathbf{P}'_{(:,2)}\\ \mathbf{P}'^T_{(:,2)}\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\mathbf{P}'_{(:,1)} & \mathbf{P}'^T_{(:,2)}\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\mathbf{P}'_{(:,2)}\end{bmatrix}\end{aligned} \tag{4.49}$$

On the other hand, the SKF and the KF have the same updated covariance blocks $(1,1)$, $(1,2)$,

and $(2,1)$ [see (4.6)]:

$$\mathbf{P}^s_{11} = \mathbf{P}^{\oplus}_{11}, \;\; \mathbf{P}^s_{12} = \mathbf{P}^{\oplus}_{12}, \;\; \text{and} \;\; \mathbf{P}^s_{21} = \mathbf{P}^{\oplus}_{21} \tag{4.50}$$

Therefore, (4.46) is equivalent to:

$$\mathbf{P}'_{11} = \mathbf{P}^{\oplus}_{11}, \;\; \mathbf{P}'_{12} = \mathbf{P}^{\oplus}_{12}, \;\; \text{and} \;\; \mathbf{P}'_{21} = \mathbf{P}^{\oplus}_{21} \tag{4.51}$$

or in the block matrix form as:

$$\mathbf{P}' - \mathbf{P}^{\oplus} = \begin{bmatrix} \mathbf{0}_{n_1 \times n_1} & \mathbf{0}_{n_1 \times n_2} \\ \mathbf{0}_{n_2 \times n_1} & * \end{bmatrix} \tag{4.52}$$

which holds true if and only if $\mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0}$ in (4.49), since $(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}$ is a SPD matrix. $\square$

As Lemma 3 demands, in order to absorb all the information of $\mathbf{x}_1$ (as is the case of the SKF), the Jacobian matrix $\mathbf{J}$ of the discarded information must be orthogonal to the directions of the updated covariance blocks corresponding to $\mathbf{x}_1$.

At this point, with Lemma 1 and Lemma 3 in place, we are ready to state our first main result, that presents a *sufficient and necessary* condition for finding the exact ISE as a mathematical equivalent to the SKF, under the framework of Alg. 2:

Theorem 1. *Given the same input, Alg. 2 and the SKF output the same updated state estimates and covariance matrices, respectively, i.e.,* $\hat{\mathbf{x}}' = \hat{\mathbf{x}}^s$ *and* $\mathbf{P}' = \mathbf{P}^s$*, if and only if* $\mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0}$.

Proof. (i) Necessity: The necessity of the condition follows directly from Lemma 3: If $\mathbf{P}' = \mathbf{P}^s$, then the three sub-blocks must equal, as in (4.46). Hence, the condition $\mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0}$ holds from its necessity in Lemma 3.

(ii) Sufficiency: (a) As for the covariance, from the condition's sufficiency in Lemma 3, we know that (4.46) holds, i.e., the $(1,1)$, $(1,2)$, and $(2,1)$ blocks of the covariance are equal. The last remaining block, $(2,2)$, are equal as a property of Alg. 2 that we have shown earlier: Since $\mathbf{R}'_{22} = \mathbf{R}_{22}$ (see the information factor update in Alg. 2), $\mathbf{P}'_{22} = \mathbf{P}_{22} = \mathbf{P}^s_{22}$ holds true from Lemma 1 and the property of the SKF [see (4.6)]. Hence, $\mathbf{P}' = \mathbf{P}^s$ holds as all the (four) sub-blocks of the covariance are equal. (b) As for the state estimates, $\hat{\mathbf{x}}'_2 = \hat{\mathbf{x}}_2 = \hat{\mathbf{x}}^s_2$ holds, again because of the property of Alg. 2 (see the state update in Alg. 2) and the SKF [see (4.5)].

To show that $\hat{\mathbf{x}}'_1 = \hat{\mathbf{x}}^s_1$, we start from the fact that $\hat{\mathbf{x}}^s_1 = \hat{\mathbf{x}}^{\oplus}_1$, where $\hat{\mathbf{x}}^{\oplus}$ denotes the updated state estimates of the KF [see (4.5)], or equivalently of the SR-IF. As we have shown earlier, the SR-IF updates the state by solving the optimization problem (4.15), where the cost function $\mathcal{C}$ can be written equivalently as in (4.38) [see the arguments from (4.24) to (4.38)]. By minimizing this least-squares cost function, the state correction computed by the SR-IF can be obtained through the normal equation as:

$$\begin{aligned}
\widetilde{\mathbf{x}}^{\oplus} &= (\mathbf{R}'^T\mathbf{R}' + \mathbf{J}^T\mathbf{J})^{-1}(\mathbf{R}'^T \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} + \mathbf{J}^T\mathbf{r}_J) \\
&= (\mathbf{P}'^{-1} + \mathbf{J}^T\mathbf{J})^{-1}(\mathbf{R}'^T \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} + \mathbf{J}^T\mathbf{r}_J) \\
&= \left[\mathbf{P}' - \mathbf{P}'\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\mathbf{P}'\right] (\mathbf{R}'^T \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} + \mathbf{J}^T\mathbf{r}_J) \qquad (4.53)
\end{aligned}$$

From the condition $\mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0} \iff \mathbf{P}'_{(1,:)}\mathbf{J}^T = \mathbf{0}$, since $\mathbf{P}'$ is symmetric $\iff \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} \mathbf{P}'\mathbf{J}^T = \mathbf{0}$, and the state partitioning $\widetilde{\mathbf{x}}^{\oplus} = \begin{bmatrix} \widetilde{\mathbf{x}}^{\oplus}_1 \\ \widetilde{\mathbf{x}}^{\oplus}_2 \end{bmatrix} \implies \widetilde{\mathbf{x}}^{\oplus}_1 = \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} \widetilde{\mathbf{x}}^{\oplus}$, we obtain:

$$\begin{aligned}
\widetilde{\mathbf{x}}^{\oplus}_1 &= \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} \widetilde{\mathbf{x}}^{\oplus} \\
&= \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} \left[\mathbf{P}' - \mathbf{P}'\mathbf{J}^T(\mathbf{I}_m + \mathbf{J}\mathbf{P}'\mathbf{J}^T)^{-1}\mathbf{J}\mathbf{P}'\right] (\mathbf{R}'^T \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} + \mathbf{J}^T\mathbf{r}_J) \\
&= \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} \mathbf{P}'\mathbf{R}'^T \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} (\mathbf{R}'^T\mathbf{R}')^{-1}\mathbf{R}'^T \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} \mathbf{R}'^{-1} \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} \\
&= \begin{bmatrix} \mathbf{I}_{n_1} & \mathbf{0}_{n_1\times n_2} \end{bmatrix} \begin{bmatrix} \mathbf{R}'_{11} & \mathbf{R}'_{12} \\ \mathbf{0} & \mathbf{R}_{22} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{r}' \\ \mathbf{0}_{n_2\times 1} \end{bmatrix} = \mathbf{R}'^{-1}_{11}\mathbf{r}' \qquad (4.54)
\end{aligned}$$

which leads to the updated state estimate $\hat{\mathbf{x}}_1^{\oplus}$ of the SR-IF as:

$$\hat{\mathbf{x}}_1^{\oplus} = \hat{\mathbf{x}}_1 + \widetilde{\mathbf{x}}_1^{\oplus} = \hat{\mathbf{x}}_1 + \mathbf{R}_{11}'^{-1}\mathbf{r}' \tag{4.55}$$

Comparing this to the state update in Alg. 2, we can see that $\hat{\mathbf{x}}_1' = \hat{\mathbf{x}}_1^{\oplus} = \hat{\mathbf{x}}_1^s$. Combining it with the previous result $\hat{\mathbf{x}}_2' = \hat{\mathbf{x}}_2^s$, we conclude that $\hat{\mathbf{x}}' = \hat{\mathbf{x}}^s$, as both (two) sub-blocks of the state estimates are equal. □

Recall that, the one and only degree of freedom in Alg. 2 is the unitary matrix $\mathbf{U}$, and by choosing different $\mathbf{U}$ matrices, it results in different matrices $\mathbf{J}$ and $\mathbf{R}'$ (hence $\mathbf{P}'$) [see (4.35)]. Therefore, as Theorem 1 demands, in order to find the exact ISE as a specific realization of Alg. 2, the task now becomes to seek a unitary matrix $\mathbf{U}$ such that the condition $\mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0}$ is satisfied. This condition poses an *implicit* constraint on the matrix $\mathbf{U}$, and in order to come up with an actual algorithm, we have further derived several other equivalent but *explicit* constraints, based on this one. The result is stated in the following lemma:

Lemma 4. *Assume that* $\mathbf{U} = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix}$ *is a unitary matrix. Define the following matrices:*

$$\mathbf{G}_1 \triangleq \begin{bmatrix}\mathbf{R}_{11}\\ \mathbf{H}_1\end{bmatrix} \in \mathbb{R}^{(n_1+m)\times n_1},\ \ \mathbf{G}_2 \triangleq \begin{bmatrix}\mathbf{R}_{12}\\ \mathbf{H}_2\end{bmatrix} \in \mathbb{R}^{(n_1+m)\times n_2},\ \ \mathbf{A} \triangleq \mathbf{R}_{22}^{-T}\mathbf{G}_2^T \in \mathbb{R}^{n_2\times(n_1+m)} \tag{4.56}$$

Then the following statements are equivalent:

(C1) $\mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0}$.

(C2) $\left[\mathbf{G}_1(\mathbf{U}_1^T\mathbf{G}_1)^{-1}\mathbf{U}_1^T - \mathbf{I}_{n_1+m}\right](\mathbf{I}_{n_1+m} + \mathbf{A}^T\mathbf{A})\,\mathbf{U}_1 = \mathbf{0}$.

(C3) The columns of $\mathbf{U}_1$ *form an orthonormal basis for the column space of* $(\mathbf{I}_{n_1+m} + \mathbf{A}^T\mathbf{A})^{-1}\mathbf{G}_1$.

(C4) The columns of $\mathbf{U}_1$ *form an orthonormal basis for the right null space of* $\mathbf{Q}_2^T(\mathbf{I}_{n_1+m} + \mathbf{A}^T\mathbf{A})$*, where the columns of* $\mathbf{Q}_2$ *form a basis for the left null space of* $\mathbf{G}_1$.

Proof. (C1) $\Longleftrightarrow$ *(C2)*: From the definition of $\mathbf{P}'$, and the expression of $\mathbf{R}'$ in (4.29), we obtain:

$$\mathbf{P}' = (\mathbf{R}'^T\mathbf{R}')^{-1} = \begin{bmatrix} \mathbf{R}_{11}'^{-1}\mathbf{R}_{11}'^{-T} + \mathbf{R}_{11}'^{-1}\mathbf{R}_{12}'\mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T}\mathbf{R}_{12}'^T\mathbf{R}_{11}'^{-T} & -\mathbf{R}_{11}'^{-1}\mathbf{R}_{12}'\mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T} \\ -\mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T}\mathbf{R}_{12}'^T\mathbf{R}_{11}'^{-T} & \mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T} \end{bmatrix} \tag{4.57}$$

which gives the expression of the first block column of $\mathbf{P}'$, i.e., $\mathbf{P}'_{(:,1)}$. Hence,

$$\begin{aligned}
& \mathbf{J}\mathbf{P}'_{(:,1)} = \mathbf{0} \\
\iff & \mathbf{J}_1(\mathbf{R}_{11}'^{-1}\mathbf{R}_{11}'^{-T} + \mathbf{R}_{11}'^{-1}\mathbf{R}_{12}'\mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T}\mathbf{R}_{12}'^T\mathbf{R}_{11}'^{-T}) - \mathbf{J}_2\mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T}\mathbf{R}_{12}'^T\mathbf{R}_{11}'^{-T} = \mathbf{0} \\
\iff & \mathbf{J}_1(\mathbf{R}_{11}'^{-1} + \mathbf{R}_{11}'^{-1}\mathbf{R}_{12}'\mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T}\mathbf{R}_{12}'^T) - \mathbf{J}_2\mathbf{R}_{22}^{-1}\mathbf{R}_{22}^{-T}\mathbf{R}_{12}'^T = \mathbf{0} \quad \text{(since } \mathbf{R}_{11}' \text{ is invertible)} \\
\iff & \mathbf{U}_2^T \underbrace{\left[\mathbf{G}_1(\mathbf{U}_1^T\mathbf{G}_1)^{-1}\mathbf{U}_1^T(\mathbf{I} + \mathbf{A}^T\mathbf{A})\mathbf{U}_1 - \mathbf{A}^T\mathbf{A}\mathbf{U}_1\right]}_{\triangleq \mathbf{\Gamma}_1} = \mathbf{0} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{(from (4.32), (4.33), and (4.56))} \\
\iff & \mathbf{U}_2^T\mathbf{\Gamma}_1 = \mathbf{0} \\
\iff & \mathbf{\Gamma}_1 = \mathbf{U}_1 \qquad\qquad\qquad\qquad\qquad\qquad \text{(since } \mathbf{U} = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix} \text{ is unitary and } \mathbf{U}_1^T\mathbf{\Gamma}_1 = \mathbf{I}\text{)} \\
\iff & \mathbf{G}_1(\mathbf{U}_1^T\mathbf{G}_1)^{-1}\mathbf{U}_1^T(\mathbf{I} + \mathbf{A}^T\mathbf{A})\mathbf{U}_1 - \mathbf{A}^T\mathbf{A}\mathbf{U}_1 = \mathbf{U}_1 \qquad \text{(from the definition of } \mathbf{\Gamma}_1\text{)} \\
\iff & \left[\mathbf{G}_1(\mathbf{U}_1^T\mathbf{G}_1)^{-1}\mathbf{U}_1^T - \mathbf{I}\right](\mathbf{I} + \mathbf{A}^T\mathbf{A})\,\mathbf{U}_1 = \mathbf{0}.
\end{aligned}$$

(C2) $\iff$ *(C3)*: Define $\mathbf{\Gamma}_2 \triangleq \mathbf{G}_1(\mathbf{U}_1^T\mathbf{G}_1)^{-1}\mathbf{U}_1^T - \mathbf{I} \in \mathbb{R}^{(n_1+m)\times(n_1+m)}$. We can show that:

$$\begin{aligned}
& \mathbf{\Gamma}_2\mathbf{G}_1 = \mathbf{0} \implies \dim\{null(\mathbf{\Gamma}_2)\} \geq n_1 \quad \text{(since } rank(\mathbf{G}_1) = n_1 \text{ because } \mathbf{R}_{11} \text{ is invertible)} \\
& \mathbf{\Gamma}_2\mathbf{U}_2 = -\mathbf{U}_2 \implies rank(\mathbf{\Gamma}_2) \geq m \\
& \qquad\qquad\qquad\qquad\qquad\qquad \text{(since } \mathbf{U} = \begin{bmatrix}\mathbf{U}_1 & \mathbf{U}_2\end{bmatrix} \text{ is unitary and } rank(\mathbf{U}_2) = m\text{)} \\
& rank(\mathbf{\Gamma}_2) + \dim\{null(\mathbf{\Gamma}_2)\} = n_1 + m \qquad \text{(rank-nullity theorem)}
\end{aligned}$$

from which we have:

$$\dim\{null(\mathbf{\Gamma}_2)\} = n_1 \tag{4.58}$$

$$\implies col(\mathbf{G}_1) = null(\mathbf{\Gamma}_2) \tag{4.59}$$

where $col(\cdot)$ denotes the column space of a matrix, while $null(\cdot)$ the right null space. Hence,

$$
\begin{aligned}
& \left[\mathbf{G}_1(\mathbf{U}_1^T\mathbf{G}_1)^{-1}\mathbf{U}_1^T - \mathbf{I}\right](\mathbf{I}+\mathbf{A}^T\mathbf{A})\,\mathbf{U}_1 = \mathbf{0} \\
\iff\ & \mathbf{\Gamma}_2\,(\mathbf{I}+\mathbf{A}^T\mathbf{A})\,\mathbf{U}_1 = \mathbf{0} && \text{(from the definition of } \mathbf{\Gamma}_2) \\
\iff\ & col((\mathbf{I}+\mathbf{A}^T\mathbf{A})\,\mathbf{U}_1) = null(\mathbf{\Gamma}_2) && \text{(from (4.58) and } rank((\mathbf{I}+\mathbf{A}^T\mathbf{A})\,\mathbf{U}_1) = n_1) \\
\iff\ & col((\mathbf{I}+\mathbf{A}^T\mathbf{A})\,\mathbf{U}_1) = col(\mathbf{G}_1) && \text{(from (4.59))} \\
\iff\ & col(\mathbf{U}_1) = col((\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\,\mathbf{G}_1) && \text{(since } \mathbf{I}+\mathbf{A}^T\mathbf{A} \text{ is invertible)} \\
\iff\ & \text{The columns of } \mathbf{U}_1 \text{ form an orthonormal basis for } col((\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\,\mathbf{G}_1). \\
& && \text{(since } \mathbf{U}_1 \text{ has orthonormal columns)}
\end{aligned}
$$

(C3) $\Longleftrightarrow$ *(C4)*: Given the matrix $\mathbf{Q}_2$, whose columns form a basis for the left null space of $\mathbf{G}_1$, i.e., $\mathbf{Q}_2^T\mathbf{G}_1 = \mathbf{0}$. Since $\mathbf{G}_1 \in \mathbb{R}^{(n_1+m)\times n_1}$ and is of full column rank, the dimension of its left null space must be m, and hence, we have $\mathbf{Q}_2 \in \mathbb{R}^{(n_1+m)\times m}$ and is of full column rank as a basic requirement of being a basis. Moreover, since $\mathbf{I}+\mathbf{A}^T\mathbf{A}$ is invertible, we obtain:

$$
\begin{cases}
(\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\,\mathbf{G}_1 \in \mathbb{R}^{(n_1+m)\times n_1}, \text{ with full column rank } n_1 \\
\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A}) \in \mathbb{R}^{m\times(n_1+m)}, \text{ with full row rank } m \\
\qquad \Longrightarrow \ \dim\{null(\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A}))\} = n_1 \\
\mathbf{Q}_2^T\mathbf{G}_1 = [\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})]\left[(\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\mathbf{G}_1\right] = \mathbf{0}
\end{cases}
$$

$$
\Longrightarrow \quad null(\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})) = col((\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\mathbf{G}_1) \tag{4.60}
$$

which shows that the right null space of $\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})$ is the same subspace as the column space of $(\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\mathbf{G}_1$. Therefore, if the columns of $\mathbf{U}_1$ form an orthonormal basis for either one subspace, so they do for the other. □

Among those conditions listed in Lemma 4, the first one is our starting point (from Theorem 1) as an implicit constraint on the matrix $\mathbf{U}$, the second presents an explicit constraint involving just the $\mathbf{U}_1$ portion, which is the only necessary part needed in Alg. 2, while the last two conditions are direct statements on how to obtain such a matrix $\mathbf{U}_1$. Hence, if one computes a matrix $\mathbf{U}_1$, based on condition *(C3)* or *(C4)*, and substitute it into Alg. 2, then this would give a square-root inverse algorithm that is equivalent to the SKF. On the other hand, in addition to being equivalent to the SKF, our desired ISE in the square-root form has further requirements

to be satisfied as mentioned before, i.e., the resulting information factor $\mathbf{R}'$ should be invertible and upper-triangular. As for the invertibility, it turns out that this is automatically guaranteed by the condition *(C3)* or *(C4)*, as stated in the following lemma:

Lemma 5. *In Alg. 2, if* $\mathbf{U}_1$ *satisfies the condition (C3) or (C4) in* Lemma 4*, then the updated factor* $\mathbf{R}'$ *is invertible.*

Proof. Since the condition *(C3)* and *(C4)* are equivalent [see (4.60)], it suffices to show the proof for just one of them, and here we choose *(C3)* for simplicity: Since both $\mathbf{G}_1$ and $\mathbf{U}_1$ have full column rank, from the condition *(C3)*, there must exist an invertible matrix $\mathbf{E}$, such that:

$$\begin{aligned} & (\mathbf{I}+\mathbf{A}^T\mathbf{A})^{-1}\mathbf{G}_1 = \mathbf{U}_1\mathbf{E} \\ \Longrightarrow\ & \mathbf{G}_1 = (\mathbf{I}+\mathbf{A}^T\mathbf{A})\mathbf{U}_1\mathbf{E} \\ \Longrightarrow\ & \mathbf{R}'_{11} = \mathbf{U}_1^T\mathbf{G}_1 = \mathbf{U}_1^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})\mathbf{U}_1\mathbf{E} = (\mathbf{I}+\mathbf{U}_1^T\mathbf{A}^T\mathbf{A}\mathbf{U}_1)\mathbf{E} \end{aligned} \tag{4.61}$$

based on the definition of $\mathbf{R}'_{11}$ in (4.32) and that $\mathbf{U}_1$ has orthonormal columns as state in condition *(C3)*. Therefore, from (4.61), we can see that $\mathbf{R}'_{11}$ must be invertible as a product of two invertible matrices. Moreover, given the assumption that $\mathbf{R}$ is invertible, and hence $\mathbf{R}_{22}$ as well, we conclude that $\mathbf{R}'$ is invertible as a block upper-triangular matrix with invertible diagonal blocks [see its expression in (4.29)]. □

As for the upper-triangular structure of $\mathbf{R}'$, it poses an additional constraint on the matrix $\mathbf{U}_1$. Note that, the conditions in Lemma 4 do not give a unique choice of $\mathbf{U}_1$. In fact, there are infinitely many matrices that will satisfy these conditions: It can be verified that, if a matrix $\mathbf{U}_1^\star$ satisfies the conditions in Lemma 4, then so does the matrix $\mathbf{U}_1^\star\mathbf{Q}$, where $\mathbf{Q} \in \mathbb{R}^{n_1\times n_1}$ is any arbitrary unitary matrix. This is reasonable because, for the underlying Schmidt Hessian matrix, its corresponding square-root factor [see (4.14)] is nonunique, if no other requirement of the factor is specified. With the additional restriction that $\mathbf{R}'$ be upper-triangular, or equivalently $\mathbf{R}'_{11}$, the choice of $\mathbf{U}_1$ is then unique (up to the sign of each column), due to the uniqueness of the Cholesky factorization of a SPD matrix (i.e., the Schmidt Hessian). Hence, as our goal is to obtain an exact ISE in the square-root form that maintains an upper-triangular information factor, we need to add this additional constraint on $\mathbf{U}_1$.

To conclude, we summarize the results of our previous analyses with the following statement:

Theorem 2. *Alg. 2 is an exact ISE in the square-root form, with the updated information factor being invertible and upper-triangular, if and only if the matrix* $\mathbf{U}_1$ *satisfies the condition (C3) or (C4) in* Lemma 4*, and* $\mathbf{R}'_{11} = \mathbf{U}_1^T \mathbf{G}_1$ *is upper-triangular.*

Theorem 2 is a direct combination of the results of Theorem 1, Lemma 4, and Lemma 5, hence no further proof is needed. The sufficient and necessary condition stated here consists of two parts: The first part guarantees the equivalence to the SKF and the invertibility of the updated factor, while the second regularizes the upper-triangular structure of the factor as required. Theorem 2 provides useful guidelines for realizing the exact ISE algorithm. In what follows, we utilize this result and present our proposed exact ISE algorithm that is both computationally efficient and numerically stable.

Exact ISE: The Algorithm

The condition stated in Theorem 2 can lead directly to the exact ISE algorithm. In fact, starting from either one of the constraints *(C3)* and *(C4)* on the matrix $\mathbf{U}_1$, we have found several versions of the algorithm. All these realizations are mathematical equivalents of the same exact ISE, but meanwhile, they differ in terms of the computational efficiency and numerical stability.

As an example, since the condition in Theorem 2 consists of two parts, one approach would accordingly decompose the matrix $\mathbf{U}_1$ into the following form:

$$\mathbf{U}_1 = \mathbf{U}'_1 \mathbf{U}''_1, \quad \text{with } \mathbf{U}'_1 \in \mathbb{R}^{(n_1+m)\times n_1} \text{ and } \mathbf{U}''_1 \in \mathbb{R}^{n_1 \times n_1} \tag{4.62}$$

where $\mathbf{U}'_1$ satisfies the constraint *(C3)* or *(C4)*, while the unitary matrix $\mathbf{U}''_1$ is chosen such that the resulting factor is upper-triangular. It is straightforward to verify that their product $\mathbf{U}_1$ obeys the condition in Theorem 2. Therefore, this is a valid approach, and the corresponding exact ISE algorithm can be carried out as (if the constraint *(C3)* is selected, for instance):

1. Obtain $\mathbf{G}_1 \leftarrow \begin{bmatrix} \mathbf{R}_{11} \\ \mathbf{H}_1 \end{bmatrix}$ and $\mathbf{G}_2 \leftarrow \begin{bmatrix} \mathbf{R}_{12} \\ \mathbf{H}_2 \end{bmatrix}$
2. Compute $\mathbf{A} \leftarrow \mathbf{R}_{22}^{-T} \mathbf{G}_2^T$
3. Compute $\mathbf{\Gamma} \leftarrow (\mathbf{I}_{n_1+m} + \mathbf{A}^T \mathbf{A})^{-1} \mathbf{G}_1$

4. Compute the thin QR factorization of $\mathbf{\Gamma} = \mathbf{U}_1' \mathbf{R}_\Gamma$, then compute $\mathbf{R}_{11}' \leftarrow \mathbf{U}_1'^T \mathbf{G}_1$, $\mathbf{R}_{12}' \leftarrow \mathbf{U}_1'^T \mathbf{G}_2$, and $\mathbf{r}' \leftarrow \mathbf{U}_1'^T \begin{bmatrix} \mathbf{r}_0^1 \\ \mathbf{r} \end{bmatrix}$
5. Compute the QR factorization of $\mathbf{R}_{11}' = \mathbf{U}_1'' \mathbf{R}_{11}^s$, then compute $\mathbf{R}_{12}^s \leftarrow \mathbf{U}_1''^T \mathbf{R}_{12}'$ and $\mathbf{r}^s \leftarrow \mathbf{U}_1''^T \mathbf{r}'$
6. Information factor update: $\mathbf{R}^s \leftarrow \begin{bmatrix} \mathbf{R}_{11}^s & \mathbf{R}_{12}^s \\ \mathbf{0}_{n_2 \times n_1} & \mathbf{R}_{22} \end{bmatrix}$
7. State update: $\hat{\mathbf{x}}^s \leftarrow \begin{bmatrix} \hat{\mathbf{x}}_1 + \mathbf{R}_{11}^{s^{-1}} \mathbf{r}^s \\ \hat{\mathbf{x}}_2 \end{bmatrix}$

where the superscript s is used to denote the fact that this algorithm is the Schmidt estimator, i.e., it is equivalent to the SKF (see the definitions in Sec. 4.2.2). As evident, this procedure is a special realization of Alg. 2, where a specific $\mathbf{U}_1$ matrix is obtained implicitly based on Theorem 2, as the product of two matrices: (i) $\mathbf{U}_1'$, whose columns, due to the thin QR factorization in Step 4, become an orthonormal basis for the column space of $\mathbf{\Gamma}$, following the constraint *(C3)*, and (ii) $\mathbf{U}_1''$, which transforms the previous Schmidt factor block $\mathbf{R}_{11}'$ into another equivalent but upper-triangular factor block $\mathbf{R}_{11}^s$, due to the QR factorization in Step 5.

Although this is a correct exact ISE algorithm, it has several drawbacks for numerical implementations: Firstly, the matrices $\mathbf{U}_1'$ and $\mathbf{U}_1''$ are formed explicitly in the two QR processes, hence reducing the computational efficiency. Second, the matrix inversion in Step 3 can cause loss in numerical accuracy. Lastly, the upper-triangular structure of the input factor block $\mathbf{R}_{11}$ is destroyed during the process, i.e., $\mathbf{R}_{11}'$ is in general dense, and hence results in the need of the triangularization step (see Step 5), which brings extra computational cost. While the first issue can be solved by using in-place QR operations so as to gain speed, the last two problems are caused by the approach itself, and cannot be eliminated easily without switching to a different scheme. In what follows, we present the "best" algorithm in terms of the computational efficiency and numerical stability, among many potential algorithms that we have found based on Theorem 2. This proposed method consists of several QR factorizations as its major operations, ensuring superior numerical behaviors. Moreover, the upper-triangular structure of the factor is respected throughout the entire process, achieving high efficiency by taking full advantage of this property.

We start from the constraint *(C4)*. Instead of a direct usage of this constraint to find $\mathbf{U}_1$, however, we first apply a unitary transformation on $\mathbf{G}_1$ and $\mathbf{G}_2$, and then follow the conditions in *(C4)*, but now with respect to the new resulting matrices. Specifically, we first perform the QR factorization of $\mathbf{G}_1$ as:

$$\mathbf{G}_1 = \begin{bmatrix} \mathbf{R}_{11} \\ \mathbf{H}_1 \end{bmatrix} = \mathbf{Q}_{G_1} \begin{bmatrix} \bar{\mathbf{R}}_{11} \\ \mathbf{0}_{m \times n_1} \end{bmatrix} \tag{4.63}$$

$$\implies \left[\mathbf{G}_1 \vdots \mathbf{G}_2\right] \xrightarrow{\text{QR}} \mathbf{Q}_{G_1}^T \left[\mathbf{G}_1 \vdots \mathbf{G}_2\right] = \begin{bmatrix} \bar{\mathbf{R}}_{11} & \vdots & \bar{\mathbf{R}}_{12} \\ \mathbf{0}_{m \times n_1} & \vdots & \bar{\mathbf{H}}_2 \end{bmatrix} = \left[\bar{\mathbf{G}}_1 \vdots \bar{\mathbf{G}}_2\right] \tag{4.64}$$

$$\text{with} \quad \bar{\mathbf{G}}_1 \triangleq \begin{bmatrix} \bar{\mathbf{R}}_{11} \\ \mathbf{0}_{m \times n_1} \end{bmatrix} = \mathbf{Q}_{G_1}^T \mathbf{G}_1 \ \text{ and } \ \bar{\mathbf{G}}_2 \triangleq \begin{bmatrix} \bar{\mathbf{R}}_{12} \\ \bar{\mathbf{H}}_2 \end{bmatrix} \triangleq \mathbf{Q}_{G_1}^T \mathbf{G}_2 \tag{4.65}$$

where the QR factor $\bar{\mathbf{R}}_{11} \in \mathbb{R}^{n_1 \times n_1}$ is guaranteed to be a full-rank upper-triangular matrix, since $\mathbf{G}_1$ is of full column rank. Define the following matrix $\bar{\mathbf{A}}$, similarly to the matrix $\mathbf{A}$ in (4.56), as:

$$\bar{\mathbf{A}} \triangleq \mathbf{R}_{22}^{-T} \bar{\mathbf{G}}_2^T = \mathbf{A}\mathbf{Q}_{G_1} \tag{4.66}$$

Now we apply the statement in the constraint *(C4)* to these new inputs, i.e., we find a matrix $\bar{\mathbf{U}}_1$, whose columns form an orthonormal basis for the right null space of $\bar{\mathbf{Q}}_2^T(\mathbf{I} + \bar{\mathbf{A}}^T\bar{\mathbf{A}})$, where the columns of $\bar{\mathbf{Q}}_2$ form a basis for the left null space of $\bar{\mathbf{G}}_1$. The advantage brought by this pre-transformation procedure is twofold: (i) Given the expression of $\bar{\mathbf{G}}_1$, its left null space can be found in closed-form as $\bar{\mathbf{Q}}_2 = \begin{bmatrix} \mathbf{0}_{m \times n_1} & \mathbf{I}_m \end{bmatrix}^T$, and (ii) given the upper-triangular structure of $\bar{\mathbf{G}}_1$, it turns out that there is an efficient way to compute the updated factor block $\bar{\mathbf{U}}_1^T\bar{\mathbf{G}}_1$, where the upper-triangular structure is always maintained. Specifically, we have:

$$\bar{\mathbf{Q}}_2 = \begin{bmatrix} \mathbf{0}_{m \times n_1} & \mathbf{I}_m \end{bmatrix}^T \implies \bar{\mathbf{Q}}_2^T(\mathbf{I} + \bar{\mathbf{A}}^T\bar{\mathbf{A}}) = \begin{bmatrix} \mathbf{0}_{m \times n_1} & \mathbf{I}_m \end{bmatrix} (\mathbf{I}_{n_1+m} + \bar{\mathbf{A}}^T\bar{\mathbf{A}}) = \bar{\mathbf{B}}^T \tag{4.67}$$

where we have defined:

$$\bar{\mathbf{A}} \triangleq \begin{bmatrix} \bar{\mathbf{A}}_1 & \bar{\mathbf{A}}_2 \end{bmatrix}, \quad \text{with } \bar{\mathbf{A}}_1 \in \mathbb{R}^{n_2 \times n_1} \text{ and } \bar{\mathbf{A}}_2 \in \mathbb{R}^{n_2 \times m} \tag{4.68}$$

$$\bar{\mathbf{B}} \triangleq \begin{bmatrix} \bar{\mathbf{B}}_1 \\ \bar{\mathbf{B}}_2 \end{bmatrix}, \quad \text{with } \bar{\mathbf{B}}_1 \triangleq \bar{\mathbf{A}}_1^T\bar{\mathbf{A}}_2 \in \mathbb{R}^{n_1 \times m} \text{ and } \bar{\mathbf{B}}_2 \triangleq \mathbf{I}_m + \bar{\mathbf{A}}_2^T\bar{\mathbf{A}}_2 \in \mathbb{R}^{m \times m} \tag{4.69}$$

As mentioned above, the columns of $\bar{\mathbf{U}}_1$ should form an orthonormal basis for the right null space of $\bar{\mathbf{Q}}_2^T(\mathbf{I} + \bar{\mathbf{A}}^T\bar{\mathbf{A}}) = \bar{\mathbf{B}}^T$, or equivalently, the left null space of $\bar{\mathbf{B}}$. Note that $\bar{\mathbf{B}} \in \mathbb{R}^{(n_1+m)\times m}$ is of full column rank, since $\bar{\mathbf{B}}_2$ is SPD [see (4.69)]. Therefore, after $\bar{\mathbf{B}}$ is computed, our next task is to obtain the updated factor blocks $\bar{\mathbf{U}}_1^T\bar{\mathbf{G}}_1$ and $\bar{\mathbf{U}}_1^T\bar{\mathbf{G}}_2$, where $\bar{\mathbf{U}}_1$ forms the left null space of $\bar{\mathbf{B}}$. A naive approach would first compute $\bar{\mathbf{U}}_1$ explicitly, and then obtain the factor blocks through matrix multiplications. This is inefficient, and it can be improved by performing a QR factorization on $\bar{\mathbf{B}}$, while computing $\bar{\mathbf{U}}_1^T\bar{\mathbf{G}}_1$ and $\bar{\mathbf{U}}_1^T\bar{\mathbf{G}}_2$ through in-place operations. Moreover, special care must be taken for this QR process, in order to take full advantage of the specific structure of $\bar{\mathbf{G}}_1$ for efficiency. Specifically, a standard QR (e.g., through Householder transformations [37]) would destroy the upper-triangular structure of $\bar{\mathbf{G}}_1$, and results in a dense factor block $\bar{\mathbf{U}}_1^T\bar{\mathbf{G}}_1$, and hence, a follow-up step will be required to triangularize it. It would be ideal if we can find a way to compute the factor blocks such that the structure is fully exploited, and furthermore, result in an upper-triangular matrix for $\bar{\mathbf{U}}_1^T\bar{\mathbf{G}}_1$ automatically at no extra cost. Indeed, we find that this is possible, through the following two-step approach: First, a QR factorization is performed only on the lower portion of $\bar{\mathbf{B}}$, i.e., the $\bar{\mathbf{B}}_2$ matrix, to triangularize it:

$$\left[\bar{\mathbf{B}} \vdots \bar{\mathbf{G}}_1 \vdots \bar{\mathbf{G}}_2\right] = \begin{bmatrix} \bar{\mathbf{B}}_1 \vdots & \bar{\mathbf{R}}_{11} & \vdots \bar{\mathbf{R}}_{12} \\ \bar{\mathbf{B}}_2 \vdots & \mathbf{0}_{m\times n_1} & \vdots \bar{\mathbf{H}}_2 \end{bmatrix}, \quad \bar{\mathbf{B}}_2 = \bar{\mathbf{Q}}_{B_2}\bar{\mathbf{R}}_{B_2} \tag{4.70}$$

$$\Longrightarrow \quad \left[\bar{\mathbf{B}} \vdots \bar{\mathbf{G}}_1 \vdots \bar{\mathbf{G}}_2\right] \xrightarrow{\text{QR}} \begin{bmatrix} \mathbf{I}_{n_1} & \\ & \bar{\mathbf{Q}}_{B_2}^T \end{bmatrix} \left[\bar{\mathbf{B}} \vdots \bar{\mathbf{G}}_1 \vdots \bar{\mathbf{G}}_2\right] = \begin{bmatrix} \bar{\mathbf{B}}_1 \vdots & \bar{\mathbf{R}}_{11} & \vdots \bar{\mathbf{R}}_{12} \\ \bar{\mathbf{R}}_{B_2} \vdots & \mathbf{0}_{m\times n_1} & \vdots \bar{\mathbf{H}}_2' \end{bmatrix} \tag{4.71}$$

$$\text{with} \quad \bar{\mathbf{H}}_2' \triangleq \bar{\mathbf{Q}}_{B_2}^T\bar{\mathbf{H}}_2 \tag{4.72}$$

where the QR factor $\bar{\mathbf{R}}_{B_2} \in \mathbb{R}^{m\times m}$ is guaranteed to be a full-rank upper-triangular matrix, since $\bar{\mathbf{B}}_2$ is SPD. Note that, in this step, we have utilized the fact that the lower portion of $\bar{\mathbf{G}}_1$ is zero, and hence it remains to be zero during the in-place QR process. This way, with $\bar{\mathbf{G}}_1$ unchanged, this first step can be carried out with minimal operations. Next, in the second step, we aim to zero out the $\bar{\mathbf{B}}_1$ block, through another in-place QR factorization, to obtain the desired left null space and its multiplication with the stacked factor blocks, but in an implicit way for efficiency. Moreover, as mentioned before, the resulting factor block $(1,1)$ should be guaranteed to be upper-triangular. To achieve this, we employ the QR with Givens rotations,

which has the flexibility in terms of the rows involved in each update and also the order in which the zeros are introduced [37], and hence keeping the desired upper-triangular structure. To be specific, based on the fact that $\bar{\mathbf{G}}_1$ and $\bar{\mathbf{R}}_{B_2}$ are both upper-triangular, we propose a Given process with the following rule:

(R1) *Use the diagonal elements of $\bar{\mathbf{R}}_{B_2}$ to zero out all the elements of $\bar{\mathbf{B}}_1$, in the following order: From left to right in terms of columns, and within each column, from bottom to top in terms of rows. Specifically, for each element of $\bar{\mathbf{B}}_1$ with row index i and column index j, $i \in \{1, 2, \ldots, n_1\}$ and $j \in \{1, 2, \ldots, m\}$, perform a Givens rotation to zero out this element, involving the following two rows: The i-th row of $\bar{\mathbf{B}}_1$ and the j-th row of $\bar{\mathbf{R}}_{B_2}$.*

By tracing the evolution of the non-zero pattern during the Givens process following this rule, it can be verified that, the upper-triangular structure of the $\bar{\mathbf{R}}_{11}$ block is kept all the way to the end, i.e., when $\bar{\mathbf{B}}_1$ becomes completely zero. Mathematically, this Givens QR process can be presented as:

$$\begin{bmatrix} \bar{\mathbf{B}}_1 \\ \bar{\mathbf{R}}_{B_2} \end{bmatrix} = \begin{bmatrix} \bar{\mathbf{U}}'_B & \bar{\mathbf{Q}}'_B \end{bmatrix} \begin{bmatrix} \mathbf{0}_{n_1 \times m} \\ \bar{\mathbf{R}}'_{B_2} \end{bmatrix} \tag{4.73}$$

$$\Longrightarrow \quad \begin{bmatrix} \bar{\mathbf{B}}_1 & \vdots & \bar{\mathbf{R}}_{11} & \vdots & \bar{\mathbf{R}}_{12} \\ \bar{\mathbf{R}}_{B_2} & \vdots & \mathbf{0}_{m \times n_1} & \vdots & \bar{\mathbf{H}}'_2 \end{bmatrix} \xrightarrow{\text{QR}}$$

$$\begin{bmatrix} \bar{\mathbf{U}}'_B & \bar{\mathbf{Q}}'_B \end{bmatrix}^T \begin{bmatrix} \bar{\mathbf{B}}_1 & \vdots & \bar{\mathbf{R}}_{11} & \vdots & \bar{\mathbf{R}}_{12} \\ \bar{\mathbf{R}}_{B_2} & \vdots & \mathbf{0}_{m \times n_1} & \vdots & \bar{\mathbf{H}}'_2 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{n_1 \times m} & \vdots & \mathbf{R}^s_{11} & \vdots & \mathbf{R}^s_{12} \\ \bar{\mathbf{R}}'_{B_2} & \vdots & \mathbf{J}^s_1 & \vdots & \mathbf{J}^s_2 \end{bmatrix} \tag{4.74}$$

$$\begin{aligned} \text{with} \quad & \mathbf{R}^s_{11} \triangleq \bar{\mathbf{U}}'^T_B \begin{bmatrix} \bar{\mathbf{R}}_{11} \\ \mathbf{0}_{m \times n_1} \end{bmatrix} \in \mathbb{R}^{n_1 \times n_1}, \ \mathbf{R}^s_{12} \triangleq \bar{\mathbf{U}}'^T_B \begin{bmatrix} \bar{\mathbf{R}}_{12} \\ \bar{\mathbf{H}}'_2 \end{bmatrix} \in \mathbb{R}^{n_1 \times n_2} \\ & \mathbf{J}^s_1 \triangleq \bar{\mathbf{Q}}'^T_B \begin{bmatrix} \bar{\mathbf{R}}_{11} \\ \mathbf{0}_{m \times n_1} \end{bmatrix} \in \mathbb{R}^{m \times n_1}, \ \mathbf{J}^s_2 \triangleq \bar{\mathbf{Q}}'^T_B \begin{bmatrix} \bar{\mathbf{R}}_{12} \\ \bar{\mathbf{H}}'_2 \end{bmatrix} \in \mathbb{R}^{m \times n_2} \end{aligned} \tag{4.75}$$

where $\bar{\mathbf{U}}'_B \in \mathbb{R}^{(n_1+m) \times n_1}$ spans the left null space of the matrix $\begin{bmatrix} \bar{\mathbf{B}}_1 \\ \bar{\mathbf{R}}_{B_2} \end{bmatrix}$, while $\bar{\mathbf{Q}}'_B \in \mathbb{R}^{(n_1+m) \times m}$ spans the column space of it, and they together form a unitary matrix, whose transpose represents the product of the sequence of the Givens rotations, according to the order in (R1). As mentioned earlier, due to this specific Given process, the resulting factor block $\mathbf{R}^s_{11}$

is guaranteed to be upper-triangular. Note that, this is possible because of the upper-triangular structure of both $\bar{\mathbf{G}}_1$ and $\bar{\mathbf{R}}_{B_2}$, thanks to the previous two QR processes, in (4.64) and (4.71). Finally, after all these three QR steps, the updated information factor blocks are given by $\mathbf{R}_{11}^s$ and $\mathbf{R}_{12}^s$ in (4.75), while $\mathbf{J}_1^s$ and $\mathbf{J}_2^s$ are the Jacobian blocks corresponding to the dropped information [see (4.32) – (4.39) for correspondence].

Now we prove the correctness of this proposed approach, i.e., it is indeed an exact ISE algorithm. Based on the generic framework of Alg. 2, we first find the specific underlying $\mathbf{U}_1$ matrix, that describes the entire unitary transformation defined in the above procedure, and then show that it satisfies the condition in Theorem 2. Specifically, as described above, our approach consists of three consecutive QR steps, where each of them defines a unitary transformation, converting the input information Jacobians, step by step, into the final updated factor blocks [see (4.64), (4.71), and (4.74)]:

$$\left[\mathbf{G}_1 \vdots \mathbf{G}_2\right] \xrightarrow{\mathbf{Q}_{G_1}^T} \left[\bar{\mathbf{G}}_1 \vdots \bar{\mathbf{G}}_2\right] \xrightarrow{\begin{bmatrix} \mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}^T \end{bmatrix}} \begin{bmatrix} \bar{\mathbf{R}}_{11} \vdots \bar{\mathbf{R}}_{12} \\ \mathbf{0} \ \vdots \ \bar{\mathbf{H}}_2' \end{bmatrix} \xrightarrow{\begin{bmatrix} \bar{\mathbf{U}}_B' & \bar{\mathbf{Q}}_B' \end{bmatrix}^T} \begin{bmatrix} \mathbf{R}_{11}^s \vdots \mathbf{R}_{12}^s \\ \mathbf{J}_1^s \ \vdots \ \mathbf{J}_2^s \end{bmatrix} \tag{4.76}$$

$$\implies \left[\mathbf{R}_{11}^s \vdots \mathbf{R}_{12}^s\right] = \bar{\mathbf{U}}_B'^T \begin{bmatrix} \mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}^T \end{bmatrix} \mathbf{Q}_{G_1}^T \left[\mathbf{G}_1 \vdots \mathbf{G}_2\right] = \left(\mathbf{Q}_{G_1} \begin{bmatrix} \mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2} \end{bmatrix} \bar{\mathbf{U}}_B'\right)^T \left[\mathbf{G}_1 \vdots \mathbf{G}_2\right] \tag{4.77}$$

Hence, if we define:

$$\mathbf{U}_1 \triangleq \mathbf{Q}_{G_1} \begin{bmatrix} \mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2} \end{bmatrix} \bar{\mathbf{U}}_B', \quad \text{with } \mathbf{U}_1 \in \mathbb{R}^{(n_1+m)\times n_1} \text{ and } \mathbf{U}_1^T\mathbf{U}_1 = \mathbf{I}_{n_1} \tag{4.78}$$

then $\mathbf{R}_{11}^s = \mathbf{U}_1^T\mathbf{G}_1$ and $\mathbf{R}_{12}^s = \mathbf{U}_1^T\mathbf{G}_2$. Comparing this result to Step 4 of Alg. 2, it is evident that this specific $\mathbf{U}_1$ in (4.78) is exactly the underlying matrix representing our entire unitary transformation. As expected, this orthonormal matrix $\mathbf{U}_1$ is the combination of the three orthonormal Q factors, corresponding to the three QR steps in our proposed approach. Next, we show that this specific $\mathbf{U}_1$ satisfies the condition stated in Theorem 2. First of all, as required by the second part of the condition, the resulting factor block $\mathbf{R}_{11}^s = \mathbf{U}_1^T\mathbf{G}_1$ is guaranteed to be upper-triangular, by the design of the last Given QR with the specific rule (R1). As for the first

part of the condition, we show that *(C4)* holds true for this specific $\mathbf{U}_1$, i.e., the columns of $\mathbf{U}_1$ form an orthonormal basis for the right null space of $\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})$, where $\mathbf{Q}_2$ spans the left null space of $\mathbf{G}_1$. Given that $\mathbf{Q}_{G_1}$ is the Q factor of the QR factorization of $\mathbf{G}_1$, from (4.63), we choose $\mathbf{Q}_2$ to be the second block column of $\mathbf{Q}_{G_1}$, hence it spans the left null space of $\mathbf{G}_1$ as required. Moreover, since $\mathbf{Q}_{G_1}$ is unitary, i.e., $\mathbf{Q}_{G_1}^T\mathbf{Q}_{G_1}=\mathbf{I}$, taking the second block row of this equation gives $\mathbf{Q}_2^T\mathbf{Q}_{G_1}=\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}$. Now we can show that:

$$
\begin{aligned}
&\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})\mathbf{U}_1 \\
&= \mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})\mathbf{Q}_{G_1}\begin{bmatrix}\mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}\end{bmatrix}\bar{\mathbf{U}}'_B && \text{(from the definition of } \mathbf{U}_1 \text{ in (4.78))} \\
&= \mathbf{Q}_2^T\mathbf{Q}_{G_1}(\mathbf{I}+\mathbf{Q}_{G_1}^T\mathbf{A}^T\mathbf{A}\mathbf{Q}_{G_1})\begin{bmatrix}\mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}\end{bmatrix}\bar{\mathbf{U}}'_B \\
& && \text{(since } \mathbf{Q}_{G_1}\mathbf{Q}_{G_1}^T=\mathbf{I} \text{ because } \mathbf{Q}_{G_1} \text{ is unitary)} \\
&= \begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}(\mathbf{I}+\mathbf{Q}_{G_1}^T\mathbf{A}^T\mathbf{A}\mathbf{Q}_{G_1})\begin{bmatrix}\mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}\end{bmatrix}\bar{\mathbf{U}}'_B && \text{(since } \mathbf{Q}_2^T\mathbf{Q}_{G_1}=\begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}) \\
&= \begin{bmatrix}\mathbf{0} & \mathbf{I}\end{bmatrix}(\mathbf{I}+\bar{\mathbf{A}}^T\bar{\mathbf{A}})\begin{bmatrix}\mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}\end{bmatrix}\bar{\mathbf{U}}'_B && \text{(from the definition of } \bar{\mathbf{A}} \text{ in (4.66))} \\
&= \bar{\mathbf{B}}^T\begin{bmatrix}\mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}\end{bmatrix}\bar{\mathbf{U}}'_B = \left(\bar{\mathbf{U}}'^T_B\begin{bmatrix}\mathbf{I} & \\ & \bar{\mathbf{Q}}_{B_2}^T\end{bmatrix}\begin{bmatrix}\bar{\mathbf{B}}_1 \\ \bar{\mathbf{B}}_2\end{bmatrix}\right)^T \\
& && \text{(from the definition of } \bar{\mathbf{B}} \text{ in (4.67) and (4.69))} \\
&= \left(\bar{\mathbf{U}}'^T_B\begin{bmatrix}\bar{\mathbf{B}}_1 \\ \bar{\mathbf{R}}_{B_2}\end{bmatrix}\right)^T && \text{(from the QR factorization in (4.70))} \\
&= \mathbf{0} && \text{(from the QR factorization in (4.73))}
\end{aligned}
$$

Hence, $\mathbf{U}_1$ belongs to the right null space of $\mathbf{Q}_2^T(\mathbf{I}+\mathbf{A}^T\mathbf{A})$. Moreover, since this null space is of dimension n_1 as shown in (4.60), and $\mathbf{U}_1$ has n_1 orthonormal columns, these columns must form an orthonormal basis for this null space. Therefore, by Theorem 2 with *(C4)*, we conclude that this specific $\mathbf{U}_1$ in (4.78), which is a summarized representation of our proposed approach, leads to a valid exact ISE algorithm.

Algorithm 3 The Exact Inverse Schmidt Estimator (ISE)

Input: Current state estimate $\hat{\mathbf{x}}$, prior information factor $\mathbf{R}$ and residual $\mathbf{r}_0$, pre-whitened measurement Jacobian $\mathbf{H}$ and residual $\mathbf{r}$

procedure UPDATE

Perform the in-place QR of $\begin{bmatrix}\mathbf{R}_{11}\\ \mathbf{H}_1\end{bmatrix}$ as: $\begin{bmatrix}\mathbf{R}_{11} \vdots \mathbf{R}_{12} \vdots \mathbf{r}_0^1\\ \mathbf{H}_1 \vdots \mathbf{H}_2 \vdots \mathbf{r}\end{bmatrix} \xrightarrow{\text{QR}} \begin{bmatrix}\bar{\mathbf{R}}_{11} \vdots \bar{\mathbf{R}}_{12} \vdots \bar{\mathbf{r}}_1\\ \mathbf{0} \vdots \bar{\mathbf{H}}_2 \vdots \bar{\mathbf{r}}_2\end{bmatrix}$

Compute $\begin{bmatrix}\bar{\mathbf{A}}_1 \vdots \bar{\mathbf{A}}_2\end{bmatrix} \leftarrow \mathbf{R}_{22}^{-T}\begin{bmatrix}\bar{\mathbf{R}}_{12}^T \vdots \bar{\mathbf{H}}_2^T\end{bmatrix}$, then compute $\bar{\mathbf{B}}_1 \leftarrow \bar{\mathbf{A}}_1^T\bar{\mathbf{A}}_2$ and $\bar{\mathbf{B}}_2 \leftarrow \mathbf{I} + \bar{\mathbf{A}}_2^T\bar{\mathbf{A}}_2$

Perform the in-place QR of $\bar{\mathbf{B}}_2$ as: $\begin{bmatrix}\bar{\mathbf{B}}_2 \vdots \bar{\mathbf{H}}_2 \vdots \bar{\mathbf{r}}_2\end{bmatrix} \xrightarrow{\text{QR}} \begin{bmatrix}\bar{\mathbf{R}}_{B_2} \vdots \bar{\mathbf{H}}_2' \vdots \bar{\mathbf{r}}_2'\end{bmatrix}$

Perform the in-place QR of $\begin{bmatrix}\bar{\mathbf{B}}_1\\ \bar{\mathbf{R}}_{B_2}\end{bmatrix}$ as: $\begin{bmatrix}\bar{\mathbf{B}}_1 \vdots \bar{\mathbf{R}}_{11} \vdots \bar{\mathbf{R}}_{12} \vdots \bar{\mathbf{r}}_1\\ \bar{\mathbf{R}}_{B_2} \vdots \mathbf{0} \vdots \bar{\mathbf{H}}_2' \vdots \bar{\mathbf{r}}_2'\end{bmatrix} \xrightarrow{\text{QR}}$ $\begin{bmatrix}\mathbf{0} \vdots \mathbf{R}_{11}^s \vdots \mathbf{R}_{12}^s \vdots \mathbf{r}^s\\ \bar{\mathbf{R}}_{B_2}' \vdots \mathbf{J}_1^s \vdots \mathbf{J}_2^s \vdots \mathbf{r}_J^s\end{bmatrix}$, by Givens rotations following (R1)

Information factor update: $\mathbf{R}^s \leftarrow \begin{bmatrix}\mathbf{R}_{11}^s & \mathbf{R}_{12}^s\\ \mathbf{0} & \mathbf{R}_{22}\end{bmatrix}$

State update: $\hat{\mathbf{x}}^s \leftarrow \begin{bmatrix}\hat{\mathbf{x}}_1 + \mathbf{R}_{11}^{s^{-1}}\mathbf{r}^s\\ \hat{\mathbf{x}}_2\end{bmatrix}$

end procedure

Output: Updated Schmidt state estimate $\hat{\mathbf{x}}^s$ and information factor $\mathbf{R}^s$

The proposed exact ISE algorithm is summarized in Alg. 3.[2] By Theorem 2, this estimator is mathematically equivalent to the SKF but in the square-root inverse form (see the definitions in Sec. 4.2.2), and the updated information factor is guaranteed to be invertible and upper-triangular. Note that, it is a special realization of the generic framework in Alg. 2, i.e., Steps 3 – 6 in Alg. 3 are particular realizations of Steps 3 – 4 in Alg. 2, with the specific $\mathbf{U}_1$ as defined in (4.78), but obtained implicitly through three consecutive QR factorizations.

In order to achieve efficient implementations, all the QR factorizations should be performed in place, i.e., the transpose of the Q factor is applied in its factored form to the stacked matrices, without explicitly computing the Q factor. Furthermore, as designed, the upper-triangular structure of the information factor's $(1,1)$ block is maintained throughout the entire algorithm, and hence should be taken into account during each QR process. Lastly, the two matrix-inversion operations in Steps 4 and 8 involve only upper (or lower) triangular matrices, and hence can be computed efficiently by backward (or forward) substitutions.

As for the computational complexity, it can be shown that the total complexity of Alg. 3 sums up to $\mathcal{O}\left((n_1+m)[n_2^2+m(n_2+n_1+m)]\right)$. Typically, in practice, we would apply the Schmidt estimator when a major portion of the state vector is to remain the same, i.e., when $n_2 \gg n_1+m$. Under this assumption, the total cost is simplified to $\mathcal{O}\left((n_1+m)n_2^2\right)$. This quadratic cost in n_2 is due to the solve operation with the upper-triangular matrix $\mathbf{R}_{22} \in \mathbb{R}^{n_2 \times n_2}$ in Step 4, which in this case becomes the dominant step in terms of the computational cost. Furthermore, if in addition, the input factor block $\mathbf{R}_{22}$ is sparse, and the solve can be done in linear time in n_2, then the total cost is reduced to $\mathcal{O}\left(m(n_1+m)n_2\right)$, which is now linear with respect to n_2. One such practical example is the problem of SLAM, where the square-root information factor is indeed sparse [24, 87], and hence potentially may allow efficient approximate solutions by using the exact ISE.

4.2.4 Approximate Inverse Schmidt Estimators

So far we have successfully derived the exact inverse Schmidt estimator (ISE), which is the information-domain equivalent of the SKF. And as mentioned before, the exact ISE is a special realization of the generic framework presented in Alg. 2. Furthermore, among all those algorithms that this generic framework covers, the exact ISE is the optimal one, in the sense that

[2] As mentioned before, in order to reduce linearization errors, this update procedure (including the linearization of the measurement equations) can be repeated iteratively till convergence.

it minimizes the mean squared error of the posterior estimate of $\mathbf{x}_1$, as is also the case for its covariance-domain equivalent SKF. In other words, the exact ISE absorbs all the information of $\mathbf{x}_1$, and updates its estimate to be the same as that of the optimal KF or SR-IF. Due to this optimality, the computational complexity of the exact ISE algorithm, as mentioned earlier, is still high (up to quadratic in the size of $\mathbf{x}_2$). Hence, we would like to further relax this optimality constraint, and seek to obtain alternative algorithms that are further approximations with respect to the exact ISE, while gaining in efficiency. Specifically, while $\mathbf{x}_2$ remains unaltered as before, we propose to update $\mathbf{x}_1$ only *approximately*, instead of *optimally* as in the case of the exact ISE, so as to reduce the computational cost.

To achieve this, we first identify the computationally-dominant step of the exact ISE algorithm, i.e., the back-solve operation with $\mathbf{R}_{22}$ in Step 4 of Alg. 3, since the size of $\mathbf{R}_{22}$ can be potentially very large. Then, instead of using the exact factor block $\mathbf{R}_{22}$, we propose to employ various approximate versions of it in the solving operation so that this step (as well as other steps) becomes less expensive. In what follows, we present two such algorithms, both of which are derived as further approximations of the exact ISE using this idea. Meanwhile, they are also special realizations of the generic framework in Alg. 2, and because of this, they inherit the properties of Alg. 2, i.e., $\mathbf{x}_2$ is preserved and the estimates are consistent.

Resource-aware Inverse Schmidt Estimator (RISE)

The first approximate algorithm is obtained by simply setting $\mathbf{R}_{22} = \infty$ in Step 4 of the exact ISE algorithm (see Alg. 3). This immediately eliminates Step 4-6 from the procedure. Hence, comparing this to the three-step QR procedure in Alg. 3, we can see that this algorithm only takes the first QR step from the exact ISE, while removing the next two. The proposed algorithm is presented in Alg. 4. In fact, this algorithm is a special realization of the generic framework in Alg. 2, as it can be obtained by choosing $\mathbf{U} = \mathbf{Q}_{G_1}$, where $\mathbf{Q}_{G_1}$ is the Q factor of the QR factorization of $\mathbf{G}_1$, as defined in (4.63).

The gain in speed of this algorithm is obvious: Since it is a "trimmed" version of Alg. 3, the computational cost is strictly lower than that of the exact ISE. In fact, the block $\mathbf{R}_{22}$ is never involved in the procedure, and the total cost is at most linear in n_2. If in addition, $\mathbf{R}_{12}$ and $\mathbf{H}_2$ are sparse, then the complexity of this entire algorithm becomes constant (depending on n_1) with respect to n_2. Hence, the advantage of Alg. 4 is its fast speed. The disadvantage, however, is that this estimator does not provide any performance guarantee in terms of the accuracy of

Algorithm 4 The Resource-aware Inverse Schmidt Estimator (RISE)

1: **Input:** Current state estimate $\hat{\mathbf{x}}$, prior information factor $\mathbf{R}$ and residual $\mathbf{r}_0$, pre-whitened measurement Jacobian $\mathbf{H}$ and residual $\mathbf{r}$
2: **procedure** UPDATE
3: Perform the in-place QR of $\begin{bmatrix}\mathbf{R}_{11}\\ \mathbf{H}_1\end{bmatrix}$ as: $\begin{bmatrix}\mathbf{R}_{11} \vdots \mathbf{R}_{12} \vdots \mathbf{r}_0^1\\ \mathbf{H}_1 \vdots \mathbf{H}_2 \vdots \mathbf{r}\end{bmatrix} \xrightarrow{\text{QR}} \begin{bmatrix}\bar{\mathbf{R}}_{11} \vdots \bar{\mathbf{R}}_{12} \vdots \bar{\mathbf{r}}_1\\ \mathbf{0} \vdots \bar{\mathbf{H}}_2 \vdots \bar{\mathbf{r}}_2\end{bmatrix}$
4: Information factor update: $\mathbf{R}^r \leftarrow \begin{bmatrix}\bar{\mathbf{R}}_{11} & \bar{\mathbf{R}}_{12}\\ \mathbf{0} & \mathbf{R}_{22}\end{bmatrix}$
5: State update: $\hat{\mathbf{x}}^r \leftarrow \begin{bmatrix}\hat{\mathbf{x}}_1 + \bar{\mathbf{R}}_{11}^{-1}\bar{\mathbf{r}}_1\\ \hat{\mathbf{x}}_2\end{bmatrix}$
6: **end procedure**
7: **Output:** Updated state estimate $\hat{\mathbf{x}}^r$ and information factor $\mathbf{R}^r$

the updated state $\mathbf{x}_1$. Specifically, unlike the exact ISE algorithm in Alg. 3, the state update in Alg. 4 only depends on the stacked Jacobian matrix $\begin{bmatrix}\mathbf{R}_{11}\\ \mathbf{H}_1\end{bmatrix}$ corresponding to $\mathbf{x}_1$, as well as the residual vector $\begin{bmatrix}\mathbf{r}_0^1\\ \mathbf{r}\end{bmatrix}$, while $\mathbf{R}_{12}$, $\mathbf{R}_{22}$, and $\mathbf{H}_2$ have no impact on the state update. Hence, this state update step is equivalent to that of the *inconsistent* approach, where $\mathbf{R}_{12}$, $\mathbf{R}_{22}$, and $\mathbf{H}_2$ are ignored, i.e., $\mathbf{x}_2$ is assumed to be *known* perfectly with zero uncertainty. The key difference, however, is that by updating and keeping track of the cross term $\mathbf{R}_{12}$, Alg. 4 maintains the correct information factor and thus achieves *consistent* estimates.

More importantly, this proposed algorithm is resource-aware, i.e., it allows trading estimation accuracy for computational efficiency according to the availability of processing resources, by adjusting the size of the window of the states selected to be updated (i.e., the size of $\mathbf{x}_1$ with respect to the entire $\mathbf{x}$). Smaller windows provide less accurate estimates at higher rates, while larger windows allow for higher accuracy at the cost of longer processing times. In the extreme case when all states are chosen for update (i.e., $\mathbf{x}_1 = \mathbf{x}$), this algorithm becomes exactly equivalent to the optimal SR-IF without any information loss, since in this case the $\mathbf{x}_2$ part vanishes and no cost term is dropped. For this reason, we name it the resource-aware inverse Schmidt estimator, or RISE.

Other Approximations

The second approximate algorithm is obtained by setting $\mathbf{R}_{22}$ to be its reduced form in Step 4 of the exact ISE algorithm (see Alg. 3). Specifically, if the $\mathbf{R}_{22}$ matrix is (block) diagonally dominant, one way would be to use a banded version of it, taking the original upper-triangular matrix but only up to a certain bandwidth. This bandwidth can be an adjustable parameter, which controls the balancing between accuracy and speed: In general, a larger bandwidth would lead to smaller discrepancy between the results of this approximate algorithm and those of the exact ISE, and hence more accurate estimates, but at a slower speed. Other approximate forms of the $\mathbf{R}_{22}$ matrix are also possible, depending on the specific characteristics of this matrix. The procedure of this proposed algorithm is omitted here, as it is very similar to Alg. 3, with the only change being that the $\mathbf{R}_{22}$ matrix in Step 4 is replaced by its approximate as described above.

To summarize, in this section, we presented several novel inverse Schmidt estimators. First, we derived the exact inverse Schmidt estimator (ISE), which is mathematically equivalent to the Schmidt-Kalman filter (SKF), but in the square-root information form. A detailed derivation of our proposed exact ISE algorithm was presented. Specifically, we started by proposing a generic framework, which shares some fundamental properties with the SKF, such as preserving some portion of the state and ensuring estimation consistency. Next, we proved several sufficient and necessary conditions, under which the generic framework becomes the desired exact ISE. Then, based on these conditions, a version of the exact ISE algorithm was obtained. This proposed algorithm consists of three QR factorizations for numerical stability, while the upper-triangular structure of the information factor is maintained throughout the entire process and fully exploited for efficiency. In addition, complexity analysis of the exact ISE algorithm is presented. Moreover, to further improve the computational efficiency, we proposed several approximate algorithms to the exact ISE. One such result is the resource-aware inverse Schmidt estimator (RISE), which provides a mechanism to trade estimation accuracy for computational efficiency, by adjusting the size of the window of the states to be updated.

Next, we employ these inverse Schmidt estimators (exact or approximate) to provide an efficient, accurate, and consistent solution to the problem of visual-inertial simultaneous localization and mapping (SLAM) in large areas.

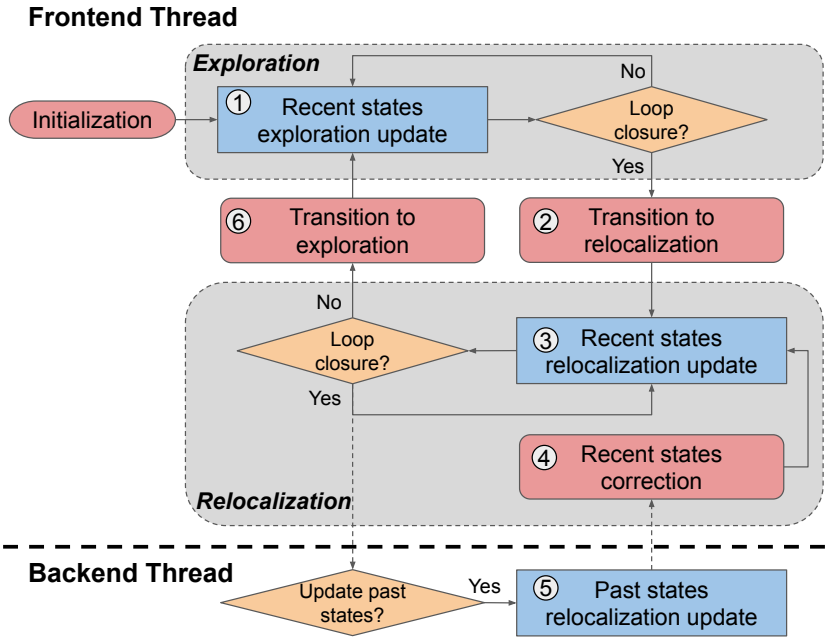


Figure 4.1: System overview. After initialization, the system starts in exploration mode (Sec. 4.3.3), and switches to relocalization mode when a set of loop closures is detected (Sec. 4.3.4). When in relocalization mode, besides the frontend thread (Sec. 4.3.4), the system may run a backend thread to perform global adjustment of past states (Sec. 4.3.4), while the frontend thread employs the backend's feedback (i.e., updated trajectory) to correct recent states (Sec. 4.3.4). Once no loop closures are detected, the system switches back to exploration mode (Sec. 4.3.3).

4.3 RISE-SLAM: A Resource-aware Inverse Schmidt Estimator for SLAM

4.3.1 System Overview

The proposed visual-inertial SLAM system, whose overview is depicted in Fig. 4.1, employs an incremental estimator comprising two key modes each addressing the particular needs of the corresponding phases of SLAM: Exploration and Relocalization. During exploration, the IMU-camera pair navigates through a new area. Thus, the feature observations available for processing span only a short window of recent poses. During relocalization, the IMU-camera pair enters areas it has previously visited, and acquires loop-closure measurements that relate recent camera poses with past ones, thus enabling to remove pose drift. Such reobservations

of past features, however, are typically expensive to process, and for this reason we explicitly distinguish between these two phases, and treat them differently.

Specifically, after initialization, the system begins in exploration mode. The estimator optimizes over a sliding window of recent states involved in the local feature-track measurements (① in Fig. 4.1), with *constant* cost (determined by the window size). Once loop-closure measurements are detected, the system enters the relocalization mode (② in Fig. 4.1), and spans two threads to process local feature-track measurements as well as loop-closure observations. In the frontend, the system estimates a sliding window of recent states using *both* types of visual measurements (③ in Fig. 4.1) with *constant* cost. This is a key novelty of our work and the process for accomplishing this in a consistent manner is detailed in Sec. 4.3.4. The optimization of other (past) states (⑤ in Fig. 4.1), which has approximately linear cost in their size, is assigned to the backend. Note that the two threads run independently so that the frontend can add new states and provide estimates even if the backend is still running. Once the backend finishes updating the past states, the frontend employs its feedback to correct the recent states (④ in Fig. 4.1). Once all the states are globally adjusted, we only need to run the frontend to update recent states (③ in Fig. 4.1). Though we could enable the backend optimization whenever the backend thread is idle, in order to save processing, in our implementation we choose to run the backend only once during each relocalization phase. Finally, when there are no more loop-closure measurements available, the system switches back to the exploration mode (⑥ in Fig. 4.1).

In what follows, we describe the key algorithmic components necessary for realizing the proposed estimation scheme. Specifically, we first provide an overview of the optimal square-root inverse estimator used during exploration, and then discuss its Schmidt-based approximation (derived in Sec. 4.2) employed later on for reducing the computational cost of relocalization.

4.3.2 Square-root Inverse Estimators for SLAM

In this section, we discuss SLAM estimators in the square-root inverse form. We start by denoting the state vector to be estimated as $\mathbf{x}$, which comprises IMU poses and feature positions, and extends with new states as time goes by. At every step, the estimator maintains a prior cost term of the current state estimate, $\|\mathbf{R}(\mathbf{x}-\hat{\mathbf{x}})\|^2$, where $\mathbf{R}$ is the upper-triangular information factor matrix (i.e., the Cholesky factor of the Hessian) and $\hat{\mathbf{x}}$ is the current estimate of $\mathbf{x}$. As new

visual or inertial measurements arrive, they contribute another cost term (after linearization), $\|\mathbf{H}(\mathbf{x}-\hat{\mathbf{x}})-\mathbf{r}\|^2$, where $\mathbf{H}$ and $\mathbf{r}$ are the measurement Jacobian and residual, respectively.[3] Then, the updated state estimate, $\hat{\mathbf{x}}^{\oplus}$, is found by minimizing the cost function:

$$\mathcal{C} = \|\mathbf{R}(\mathbf{x}-\hat{\mathbf{x}})\|^2 + \|\mathbf{H}(\mathbf{x}-\hat{\mathbf{x}})-\mathbf{r}\|^2 \tag{4.79}$$

$$\hat{\mathbf{x}}^{\oplus} = \arg\min_{\mathbf{x}} \mathcal{C} \tag{4.80}$$

Optimal Estimator

The optimal solution of (4.79) can be computed as:

$$\begin{aligned}
\mathcal{C} &= \left\| \begin{bmatrix} \mathbf{R} \\ \mathbf{H} \end{bmatrix} (\mathbf{x}-\hat{\mathbf{x}}) - \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \right\|^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}^{\oplus} \\ \mathbf{0} \end{bmatrix} (\mathbf{x}-\hat{\mathbf{x}}) - \begin{bmatrix} \mathbf{r}^{\oplus} \\ \mathbf{e} \end{bmatrix} \right\|^2 \\
&= \left\| \mathbf{R}^{\oplus}(\mathbf{x}-\hat{\mathbf{x}}-\mathbf{R}^{\oplus -1}\mathbf{r}^{\oplus}) \right\|^2 + \|\mathbf{e}\|^2 \qquad (4.81) \\
\Rightarrow \hat{\mathbf{x}}^{\oplus} &= \hat{\mathbf{x}} + \mathbf{R}^{\oplus -1}\mathbf{r}^{\oplus} \qquad (4.82)
\end{aligned}$$

where we have performed the following QR factorization:

$$\begin{bmatrix} \mathbf{R} \\ \mathbf{H} \end{bmatrix} = \mathbf{Q} \begin{bmatrix} \mathbf{R}^{\oplus} \\ \mathbf{0} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{r}^{\oplus} \\ \mathbf{e} \end{bmatrix} \triangleq \mathbf{Q}^T \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \tag{4.83}$$

The main advantage of this estimator is its optimality in minimizing the mean square error. Additionally, it is very efficient during the *exploration* phase, if the states in $\mathbf{x}$ follow a *chronological* order. Specially, when only local feature-track measurements are available, as described in [52], the QR factorization needs to involve only the bottom-right part of $\mathbf{R}$, which corresponds to recent states. Thus, the cost remains constant, irrespective of the size of the entire state vector $\mathbf{x}$. In contrast, during *relocalization*, this estimator becomes very inefficient for processing loop-closure measurements, which involve both recent and past states. In this case, the size of the submatrix of $\mathbf{R}$ involved in the QR factorization increases significantly, making the

[3] We follow [89] for the state parameterization, as well as for the visual-inertial measurement processing and cost term formulations.

cost at least linear in the size of $\mathbf{x}$. Since this becomes prohibitively expensive, especially when navigating in large areas, in what follows, we consider the inverse Schmidt approximations that reduce the computational cost, while preserving consistency.

Estimators Based on the Schmidt Approximation

As mentioned earlier, the Schmidt approximation, which was introduced originally for the Kalman filter [80], is consistent. The key idea behind it is to save processing cost by updating only a subset of the states while leaving the rest unaltered. In Sec. 4.2, we have derived its equivalent in the square-root inverse form. In brief, the Schmidt approximation starts by partitioning the state vector $\mathbf{x}$ into two parts: $\mathbf{x}_1$ and $\mathbf{x}_2$. Now the prior term can be written as

$$\|\mathbf{R}(\mathbf{x}-\hat{\mathbf{x}})\|^2 = \left\| \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} \right\|^2 \tag{4.84}$$

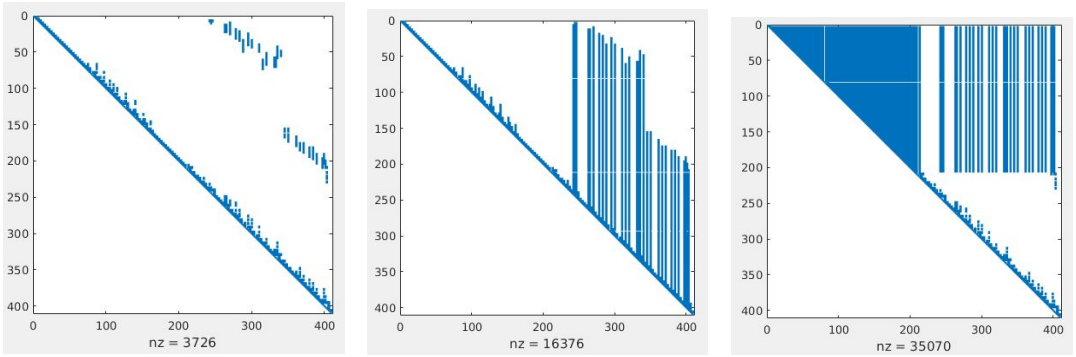
Employing the idea of Schmidt, we use the measurements to update the estimate of $\mathbf{x}_1$ to $\hat{\mathbf{x}}_1^{\oplus}$ but keep $\hat{\mathbf{x}}_2$ the same. By doing so, the posterior term should become

$$\left\| \begin{bmatrix} \mathbf{R}_{11}^{\oplus} & \mathbf{R}_{12}^{\oplus} \\ & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1^{\oplus} \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} \right\|^2 \tag{4.85}$$

A property of the Schmidt approximation in this square-root inverse form is that $\mathbf{R}_{22}$ remains the same, which does not hold if we change the state order (update $\mathbf{x}_2$ but not $\mathbf{x}_1$). For this reason, it is preferable to put the states to be updated on the upper part of $\mathbf{x}$. In practice, we typically focus more on recent states than past states, so the states must be organized in *reverse chronological* order in order to apply the Schmidt approximation. This is in stark contrast to the preferred state order for the case of the optimal estimator during exploration. The ramifications of this order switching will become evident when we discuss them in Sec. 4.3.3-4.3.4.

Exact Inverse Schmidt Estimator (ISE)

Among all Schmidt estimators, the exact Schmidt [80] yields the optimal solution for $\mathbf{x}_1$. Surprisingly (and quite unfortunately), its equivalent in the inverse form, which we derived in Sec. 4.2, called the exact inverse Schmidt estimator (ISE), has no speed advantage over the optimal estimator. In SLAM problems, the Cholesky factor of the Hessian matrix typically has a



(a) (b) (c)

Figure 4.2: Structures of the information factors: (a) Prior. (b) Posterior after the update using the optimal estimator. (c) Posterior after the update using the exact ISE with half of the state vector updated.

dense band on diagonal but is sparse off diagonal. Thus, the cost of the optimal estimator, which performs a Cholesky factorization on the Hessian matrix, is almost linear in the size of the state vector. According to our analysis in Sec. 4.2, when the Hessian matrix (or its Cholesky factor) is sparse, the exact ISE is also of linear cost in the size of the state vector if a major portion of the state vector is to be unaltered. Therefore, when applied to SLAM, the exact ISE shares the same order of computational cost as the optimal estimator even if it introduces approximation.

Moreover, the exact ISE introduces more fill-ins in the Cholesky factor of the Hessian matrix than the optimal estimator, which will significantly increase the processing and memory cost. As an example, Fig. 4.2(a) shows the Cholesky factor of the prior cost term at a step in a SLAM problem, and Fig. 4.2(b) depicts the structure of the posterior factor if the optimal estimator is employed. Instead, if the exact ISE is applied with updating half of the state vector, the structure becomes what is shown in Fig. 4.2(c), where the posterior Cholesky factor is much denser.

Resource-aware Inverse Schmidt Estimator (RISE)

Since the exact ISE is a consistent estimator in the inverse domain, in Sec. 4.2, we have used it as a starting point to investigate further approximation to it, which leads us to the *resource-aware inverse Schmidt estimator* (RISE). The RISE procedure is briefly summarized hereafter.

First, we rewrite the cost function in (4.79) as

$$\begin{aligned}
\mathcal{C} &= \left\| \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} \right\|^2 \\
&+ \left\| \begin{bmatrix} \mathbf{H}_1 & \mathbf{H}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} - \mathbf{r} \right\|^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}_{11} & \mathbf{R}_{12} \\ \mathbf{H}_1 & \mathbf{H}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \right\|^2 + \|\mathbf{R}_{22}(\mathbf{x}_2 - \hat{\mathbf{x}}_2)\|^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}_{11}^{\oplus} & \mathbf{R}_{12}^{\oplus} \\ & \mathbf{H}_2^{\oplus} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{r}_1^{\oplus} \\ \mathbf{e}_1 \end{bmatrix} \right\|^2 + \|\mathbf{R}_{22}(\mathbf{x}_2 - \hat{\mathbf{x}}_2)\|^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}_{11}^{\oplus} & \mathbf{R}_{12}^{\oplus} \\ & \mathbf{H}_2^{\oplus} \\ & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{r}_1^{\oplus} \\ \mathbf{e}_1 \\ \mathbf{0} \end{bmatrix} \right\|^2
\end{aligned} \tag{4.86}$$

where the following QR factorization was performed:

$$\begin{bmatrix} \mathbf{R}_{11} \\ \mathbf{H}_1 \end{bmatrix} = \mathbf{Q}_1 \begin{bmatrix} \mathbf{R}_{11}^{\oplus} \\ \mathbf{0} \end{bmatrix} \tag{4.87}$$

and

$$\begin{bmatrix} \mathbf{R}_{12}^{\oplus} \\ \mathbf{H}_2^{\oplus} \end{bmatrix} \triangleq \mathbf{Q}_1^T \begin{bmatrix} \mathbf{R}_{12} \\ \mathbf{H}_2 \end{bmatrix}, \quad \begin{bmatrix} \mathbf{r}_1^{\oplus} \\ \mathbf{e}_1 \end{bmatrix} \triangleq \mathbf{Q}_1^T \begin{bmatrix} \mathbf{0} \\ \mathbf{r} \end{bmatrix} \tag{4.88}$$

Next, instead of minimizing $\mathcal{C}$, we drop the cost term $\mathbf{x}_2$, $\left\|\mathbf{H}_2^{\oplus}\left(\mathbf{x}_2 - \hat{\mathbf{x}}_2\right) - \mathbf{e}_1\right\|^2$ in (4.86), and minimize (see Fig. 4.3):

$$\bar{\mathcal{C}} = \left\| \begin{bmatrix} \mathbf{R}_{11}^{\oplus} & \mathbf{R}_{12}^{\oplus} \\ & \mathbf{R}_{22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 - \hat{\mathbf{x}}_1 \\ \mathbf{x}_2 - \hat{\mathbf{x}}_2 \end{bmatrix} - \begin{bmatrix} \mathbf{r}_1^{\oplus} \\ \mathbf{0} \end{bmatrix} \right\|^2 \tag{4.89}$$

Finally, we update the estimate of $\mathbf{x}_1$ by setting

$$\hat{\mathbf{x}}_1^{\oplus} = \arg\min_{\mathbf{x}_1} \bar{\mathcal{C}} = \hat{\mathbf{x}}_1 + {\mathbf{R}_{11}^{\oplus}}^{-1} \mathbf{r}_1^{\oplus} \tag{4.90}$$

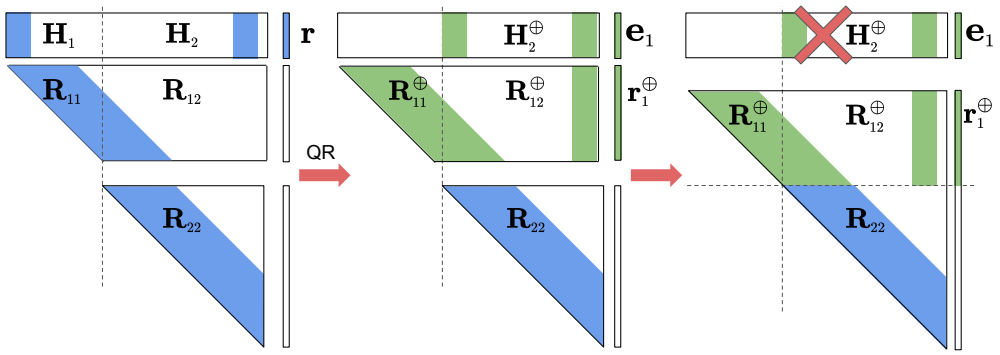


Figure 4.3: Structure of the information factor when applying RISE. The QR factorization does not involve $\mathbf{R}_{22}$. We drop the cost term $\|\mathbf{H}_2^{\oplus}(\mathbf{x}_2 - \hat{\mathbf{x}}_2) - \mathbf{e}_1\|^2$, and combine $\mathbf{R}_{22}$ with $\mathbf{R}_{11}^{\oplus}$ and $\mathbf{R}_{12}^{\oplus}$ to form the new cost function $\bar{\mathcal{C}}$ for updating $\mathbf{x}_1$, while $\mathbf{x}_2$ remains unchanged. Dropping this cost term is the key approximation of RISE. And by ignoring the information in it, we preserve the sparsity of the Cholesky factor.

while $\hat{\mathbf{x}}_2$ remains unchanged. As a Schmidt-type estimator, RISE is *consistent* since it does *not* assume any state as perfectly known. Instead, it only drops information (the term $\|\mathbf{H}_2^{\oplus}(\mathbf{x}_2 - \hat{\mathbf{x}}_2) - \mathbf{e}_1\|^2$), and correctly updates the cross term $\mathbf{R}_{12}$ between $\mathbf{x}_1$ and $\mathbf{x}_2$. As compared to the exact ISE, RISE computes an approximate estimate for $\mathbf{x}_1$, whose accuracy loss is negligible when the estimate of $\mathbf{x}_2$ is precise. In the extreme case, when the uncertainty of $\mathbf{x}_2$ goes to zero, RISE results in the optimal solution for $\mathbf{x}_1$ as ISE. For this reason, in practice, we set $\mathbf{x}_2$ to be states with low uncertainty. On the other hand, RISE is significantly more efficient than the exact ISE since the cost of the QR factorization is cubic in the size of $\mathbf{x}_1$ (instead of $\mathbf{x}$), and introduces no extra fill-ins, thus keeping $\mathbf{R}$ sparse. If we select $\mathbf{x}_1$ to contain a small number of states (e.g., a window of recent camera poses and features), the cost is O(1) in the size of $\mathbf{x}$. Although the column size of $\mathbf{R}_{12}^{\oplus}$ is the same as the size of $\mathbf{x}_2$ and can be comparable to that of $\mathbf{x}$, it is sparse in the context of SLAM. As a result, computing $\mathbf{R}_{12}^{\oplus}$ is also O(1).

A key advantage of RISE is that it can trade accuracy for speed by adjusting the size of $\mathbf{x}_1$. Specifically, during relocalization, we can set $\mathbf{x}_1 = \mathbf{x}$ to obtain an accurate global adjustment if it is the first loop-closure event, or we can use RISE with a small-size $\mathbf{x}_1$ for an approximate but efficient solution. Furthermore, this global adjustment can be split into two steps, where we first employ RISE with a small sized $\mathbf{x}_1$, and then we optimize over $\mathbf{x}_2$, which can run independently in the backend. Through this process, detailed in Sec. 4.3.4, the frontend maintains its real-time localization capability, while the backend allows taking advantage of loop-closure events after

long periods of exploration.

Note also that the optimal estimator for exploration in Sec. 4.3.2 can be considered as a special case of RISE, with $\mathbf{x}_1 = \mathbf{x}$ and a chronological state order. As it will become evident hereafter, by applying RISE, with different state orders/sizes of $\mathbf{x}_1$, in the two phases of SLAM, we can achieve real-time performance while maintaining consistency.

4.3.3 RISE-SLAM: Exploration

In this section, we describe how RISE-SLAM processes local feature tracks (see ① in Fig. 4.1) during exploration. We start with the first exploration, and then discuss the general case, where the system has just switched back to exploration from relocalization.

First Exploration

During the first exploration, we realize the efficiency of the optimal estimator (Sec. 4.3.2) by organizing the states in *chronological* order [52]. Moreover, we apply RISE with $\mathbf{x}_1 = \mathbf{x}$, which is equivalent to the optimal estimator. Denote the state vector as $\begin{bmatrix}\mathbf{x}_{E1}^T & \mathbf{x}_{E2}^T\end{bmatrix}^T$, where $\mathbf{x}_{E2}$ comprises a sliding window of *recent* states involved in the local feature-track measurements, while $\mathbf{x}_{E1}$ contains all other (previous) states. The cost function to minimize is (see Fig. 4.4)

$$\begin{aligned}\mathcal{C}_E = &\left\| \begin{bmatrix}\mathbf{R}_{E11} & \mathbf{R}_{E12} \\ & \mathbf{R}_{E22}\end{bmatrix} \begin{bmatrix}\mathbf{x}_{E1} - \hat{\mathbf{x}}_{E1} \\ \mathbf{x}_{E2} - \hat{\mathbf{x}}_{E2}\end{bmatrix} \right\|^2 \\ &+ \left\| \mathbf{H}_{E2}(\mathbf{x}_{E2} - \hat{\mathbf{x}}_{E2}) - \mathbf{r}_E \right\|^2 \end{aligned} \tag{4.91}$$

where the first term is the prior from the previous time step, while the second term corresponds to the new (IMU or feature) measurements. Then, the optimal solution is

$$\hat{\mathbf{x}}_{E2}^{\oplus} = \hat{\mathbf{x}}_{E2} + {\mathbf{R}_{E22}^{\oplus}}^{-1}\mathbf{r}_E^{\oplus} \tag{4.92}$$

$$\hat{\mathbf{x}}_{E1}^{\oplus} = \hat{\mathbf{x}}_{E1} - \mathbf{R}_{E11}^{-1}\mathbf{R}_{E12}{\mathbf{R}_{E22}^{\oplus}}^{-1}\mathbf{r}_E^{\oplus} \tag{4.93}$$

which results from rewriting (4.91) as

$$\begin{aligned}
\mathcal{C}_E &= \left\| \begin{bmatrix} \mathbf{R}_{E11} & \mathbf{R}_{E12} \end{bmatrix} (\mathbf{x}_E - \hat{\mathbf{x}}_E) \right\|^2 \\
&+ \left\| \begin{bmatrix} \mathbf{R}_{E22} \\ \mathbf{H}_{E2} \end{bmatrix} (\mathbf{x}_{E2} - \hat{\mathbf{x}}_{E2}) - \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_E \end{bmatrix} \right\|^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}_{E11} & \mathbf{R}_{E12} \end{bmatrix} (\mathbf{x}_E - \hat{\mathbf{x}}_E) \right\|^2 \\
&+ \left\| \begin{bmatrix} \mathbf{R}^{\oplus}_{E22} \\ \mathbf{0} \end{bmatrix} (\mathbf{x}_{E2} - \hat{\mathbf{x}}_{E2}) - \begin{bmatrix} \mathbf{r}^{\oplus}_E \\ \mathbf{e}_E \end{bmatrix} \right\|^2 \\
&= \left\| \begin{bmatrix} \mathbf{R}_{E11} & \mathbf{R}_{E12} \\ & \mathbf{R}^{\oplus}_{E22} \end{bmatrix} (\mathbf{x}_E - \hat{\mathbf{x}}_E) - \begin{bmatrix} \mathbf{0} \\ \mathbf{r}^{\oplus}_E \end{bmatrix} \right\|^2 + \|\mathbf{e}_E\|^2 \qquad (4.94) \\
&= \left\| \begin{bmatrix} \mathbf{R}_{E11} & \mathbf{R}_{E12} \\ & \mathbf{R}^{\oplus}_{E22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{E1} - \hat{\mathbf{x}}^{\oplus}_{E1} \\ \mathbf{x}_{E2} - \hat{\mathbf{x}}^{\oplus}_{E2} \end{bmatrix} \right\|^2 + \|\mathbf{e}_E\|^2 \qquad (4.95)
\end{aligned}$$

where $\mathbf{R}^{\oplus}_{E22}$ is computed by the following QR factorization:

$$\begin{bmatrix} \mathbf{R}_{E22} \\ \mathbf{H}_{E2} \end{bmatrix} = \mathbf{Q}_E \begin{bmatrix} \mathbf{R}^{\oplus}_{E22} \\ \mathbf{0} \end{bmatrix}, \quad \begin{bmatrix} \mathbf{r}^{\oplus}_E \\ \mathbf{e}_E \end{bmatrix} = \mathbf{Q}^T_E \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_E \end{bmatrix} \qquad (4.96)$$

The impact of this QR factorization on the terms appearing in the cost functions in (4.91)-(4.95) is depicted in Fig. 4.4. These steps are actually analogous to those described in [52], and similarly, the computational complexity is *constant* and only depends on the size of $\mathbf{x}_{E2}$. As compared to [52], we improve speed during exploration by limiting both the number of features processed at every step and the feature tracks' length (since longer tracks offer more accuracy yet diminishing returns, while increasing the processing cost cubically) based on a preselected size of $\mathbf{x}_{E2}$.

Transition from Relocalization to Exploration

Consider the case when the system is in the relocalization mode (Sec. 4.3.4) and is about to switch to the exploration mode (i.e., it receives no more loop-closure measurements) (see ⑥ in Fig. 4.1). Due to the opposite state orderings used in these two modes respectively (Sec. 4.3.2), we first need to change the order of the recent states from *reverse chronological*, as required

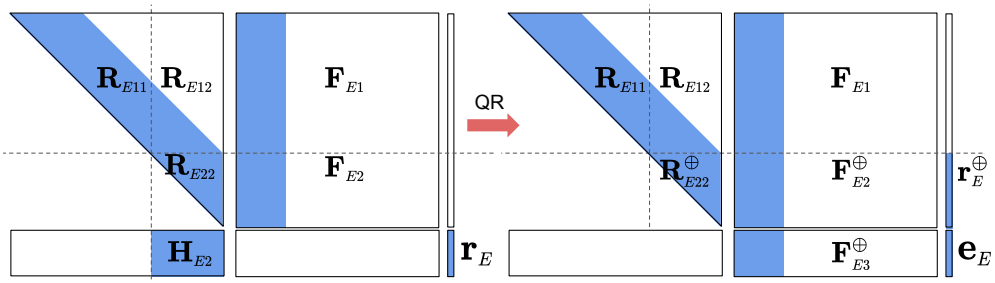


Figure 4.4: Structure of the information factor corresponding to the exploration cost terms before and after an update. $\mathbf{F}_E$ does not exist for the first exploration, while in the general case, it contains the cross information between the new and the old map, where the dense columns on the left correspond to some last states of the old map.

in relocalization (Sec. 4.3.4), to *chronological*, as for exploration (Sec. 4.3.3). Specifically, as what will become evident in Sec. 4.3.4, while in relocalization, the system has a prior term in the form of

$$\mathcal{C}_N = \left\| \begin{bmatrix} \mathbf{R}'_{N11} & \mathbf{R}'_{N12} \\ & \mathbf{R}'_{N22} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_N - \hat{\mathbf{x}}'_N \\ \mathbf{x}'_M - \hat{\mathbf{x}}'_M \end{bmatrix} \right\|^2 \tag{4.97}$$

where the state vector is divided into two parts: $\begin{bmatrix} \mathbf{x}'^T_N & \mathbf{x}'^T_M \end{bmatrix}^T$ (the superscript $'$ denotes reverse chronological order). $\mathbf{x}'_N$ contains the *recent* states where no loop-closure measurements are received, and they correspond to the beginning of the new exploration phase. Then, we change the state order of $\mathbf{x}'_N$ to chronological, by $\mathbf{x}_N \triangleq \mathbf{P}_N \mathbf{x}'_N$ ($\mathbf{P}_N$ is a permutation matrix). Subsequently, we perform a QR factorization to make the permuted Cholesky factor upper-triangular again:

$$\mathbf{Q}_N \begin{bmatrix} \mathbf{R}_{N11} & \mathbf{R}_{N12} \end{bmatrix} = \begin{bmatrix} \mathbf{R}'_{N11}\mathbf{P}_N^T & \mathbf{R}'_{N12} \end{bmatrix} \tag{4.98}$$

which is of *constant* cost regardless of the size of $\mathbf{x}'_M$ since $\mathbf{R}'_{N12}$ is sparse. After these operations, $\mathcal{C}_N$ can be written as

$$\begin{aligned}\mathcal{C}_N &= \left\| \begin{bmatrix} \mathbf{R}_{N11} & \mathbf{R}_{N12} \\ & \mathbf{R}'_{N22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_N - \hat{\mathbf{x}}_N \\ \mathbf{x}'_M - \hat{\mathbf{x}}'_M \end{bmatrix} \right\|^2 \\ &= \left\| \begin{bmatrix} \mathbf{R}_{N11} & \mathbf{R}_{N12} \end{bmatrix} \begin{bmatrix} \mathbf{x}_N - \hat{\mathbf{x}}_N \\ \mathbf{x}'_M - \hat{\mathbf{x}}'_M \end{bmatrix} \right\|^2 + \mathcal{C}_M \end{aligned} \tag{4.99}$$

where

$$\mathbf{R}_M \triangleq \mathbf{R}'_{N22} \tag{4.100}$$

$$\mathcal{C}_M \triangleq \left\| \mathbf{R}_M \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M \right) \right\|^2 \tag{4.101}$$

Note that the states in $\mathbf{x}'_M$ are considered as an old map, which we do not update during the new exploration, so $\mathcal{C}_M$ will be unchanged.

After this transition step, a new map (comprising new camera poses and features) begins with $\mathbf{x}_N$, while $\mathbf{R}_{N11}$ represents its information factor and $\mathbf{R}_{N12}$ the cross information between the two maps.

Exploration: General Case

In general, when in exploration mode, past map's states (from previous exploration and relocalization phases) do not need to be updated. Thus, we apply RISE with $\mathbf{x}_2 = \mathbf{x}'_M$ being the past map's states, while $\mathbf{x}_1 = \mathbf{x}_N$ comprises the new map's states which expand as the current exploration goes on. When in exploration mode, the system not only estimates the states of the new map built in the current exploration, but also maintains their correlations with the old map's states. Specifically, as compared to the first exploration, instead of (4.91), the cost function in the general case has the following form:

$$\begin{aligned}\mathcal{C}_E = &\left\| \begin{bmatrix} \mathbf{R}_{E11} & \mathbf{R}_{E12} \\ & \mathbf{R}_{E22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{E1} - \hat{\mathbf{x}}_{E1} \\ \mathbf{x}_{E2} - \hat{\mathbf{x}}_{E2} \end{bmatrix} + \mathbf{F}_E \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M \right) \right\|^2 \\ &+ \left\| \mathbf{H}_{E2} (\mathbf{x}_{E2} - \hat{\mathbf{x}}_{E2}) - \mathbf{r}_E \right\|^2 + \mathcal{C}_M \end{aligned} \tag{4.102}$$

In addition to terms in (4.91), (4.102) has a cost term corresponding to the old map from (4.99), $\mathcal{C}_M$, and a matrix $\mathbf{F}_E$ that keeps the correlation information between the two maps, which is an extension of $\mathbf{R}_{N12}$ in (4.99). In other words, (4.91) is a special case of (4.102) where the state vector of the old map, $\mathbf{x}'_M$, is empty and $\mathbf{F}_E$ does not exist. The estimates of the states of the new map are updated by the same operations as in the first exploration:

$$\begin{aligned}\mathcal{C}_E = &\left\| \begin{bmatrix} \mathbf{R}_{E11} & \mathbf{R}_{E12} \\ & \mathbf{R}^{\oplus}_{E22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{E1} - \hat{\mathbf{x}}^{\oplus}_{E1} \\ \mathbf{x}_{E2} - \hat{\mathbf{x}}^{\oplus}_{E2} \end{bmatrix} + \mathbf{F}^{\oplus}_E \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M \right) \right\|^2 \\ &+ \left\| \mathbf{F}^{\oplus}_{E3} \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M \right) - \mathbf{e}_E \right\|^2 + \mathcal{C}_M \end{aligned} \tag{4.103}$$

where $\hat{\mathbf{x}}^{\oplus}_{E2}$, $\hat{\mathbf{x}}^{\oplus}_{E1}$, and $\mathbf{R}^{\oplus}_{E22}$ are computed as described in (4.92)-(4.96), and we also need to update $\mathbf{F}_E$ to $\mathbf{F}^{\oplus}_E$ accordingly, as

$$\mathbf{F}_E = \begin{bmatrix} \mathbf{F}_{E1} \\ \mathbf{F}_{E2} \end{bmatrix} \tag{4.104}$$

$$\begin{bmatrix} \mathbf{F}^{\oplus}_{E2} \\ \mathbf{F}^{\oplus}_{E3} \end{bmatrix} = \mathbf{Q}^T_E \begin{bmatrix} \mathbf{F}_{E2} \\ \mathbf{0} \end{bmatrix} \tag{4.105}$$

$$\mathbf{F}^{\oplus}_E = \begin{bmatrix} \mathbf{F}_{E1} \\ \mathbf{F}^{\oplus}_{E2} \end{bmatrix} \tag{4.106}$$

where $\mathbf{F}_{E1}$ and $\mathbf{F}_{E2}$ have the same row size as $\mathbf{x}_{E1}$ and $\mathbf{x}_{E2}$ (see Fig. 4.4), respectively. Next, we apply RISE with updating the estimate of $\mathbf{x}_E$ but not $\mathbf{x}_M$ so as to save processing cost. Specifically, the term $\left\| \mathbf{F}^{\oplus}_{E3} \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M \right) - \mathbf{e}_E \right\|^2$ is dropped, and we have

$$\bar{\mathcal{C}}_E = \left\| \begin{bmatrix} \mathbf{R}_{E11} & \mathbf{R}_{E12} \\ & \mathbf{R}^{\oplus}_{E22} \end{bmatrix} \begin{bmatrix} \mathbf{x}_{E1} - \hat{\mathbf{x}}^{\oplus}_{E1} \\ \mathbf{x}_{E2} - \hat{\mathbf{x}}^{\oplus}_{E2} \end{bmatrix} + \mathbf{F}^{\oplus}_E \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M \right) \right\|^2 + \mathcal{C}_M \tag{4.107}$$

instead of $\mathcal{C}_E$ as the prior term for the next step.

Note that only the lower of part of $\mathbf{F}_E$, $\mathbf{F}_{E2}$, which has a bounded number of dense columns, needs updating. Thus, the computational complexity for the general case of exploration with RISE is also *constant*.

4.3.4 RISE-SLAM: Relocalization

We now consider the case, where the system switches from exploration to relocalization mode so as to use loop-closure measurements for global pose and map correction.

Transition from Exploration to Relocalization

Before entering relocalization, we need a transition step (see ② in Fig. 4.1) to switch the state order from *chronological* to *reverse chronological*. Specifically, during exploration, the state vector is in the form $\begin{bmatrix}\mathbf{x}_L^T & \mathbf{x}_M'^T\end{bmatrix}^T$, where $\mathbf{x}_L$ comprises all states in the current map, while $\mathbf{x}'_M$ corresponds to the old map [see (4.99) and Sec. 4.3.3]. We first change the state order of $\mathbf{x}_L$ to reverse chronological by defining $\mathbf{x}'_L \triangleq \mathbf{P}_L\mathbf{x}_L$ ($\mathbf{P}_L$ is a permutation matrix). Then, we split $\mathbf{x}'_L$ into two parts: $\mathbf{x}'_{L1}$ comprises the *recent* states to be maintained in the frontend, while the remaining states $\mathbf{x}'_{L2}$ are combined with the old map $\mathbf{x}'_M$ so as to be optimized by the backend ($\mathbf{x}'_B \triangleq \begin{bmatrix}\mathbf{x}_{L2}'^T & \mathbf{x}_M'^T\end{bmatrix}^T$). Accordingly, the prior term was in the form of

$$\mathcal{C}_L = \left\|\mathbf{R}_L(\mathbf{x}_L - \hat{\mathbf{x}}_L) + \mathbf{F}_L(\mathbf{x}'_M - \hat{\mathbf{x}}'_M)\right\|^2 + \mathcal{C}_M \tag{4.108}$$

and we perform a QR factorization to make the information factor upper-triangular after switching the state order:

$$\begin{aligned}
\mathcal{C}_L &= \left\| \begin{bmatrix}\mathbf{R}_L & \mathbf{F}_L \\ & \mathbf{R}'_M\end{bmatrix} \begin{bmatrix}\mathbf{x}_L - \hat{\mathbf{x}}_L \\ \mathbf{x}'_M - \hat{\mathbf{x}}'_M\end{bmatrix} \right\|^2 \\
&= \left\| \begin{bmatrix}\mathbf{R}'_L & \mathbf{F}'_L \\ & \mathbf{R}'_M\end{bmatrix} \begin{bmatrix}\mathbf{x}'_L - \hat{\mathbf{x}}'_L \\ \mathbf{x}'_M - \hat{\mathbf{x}}'_M\end{bmatrix} \right\|^2 \\
&= \left\| \begin{bmatrix}\mathbf{R}'_{L11} & \mathbf{R}'_{L12} \\ & \mathbf{R}'_{L22}\end{bmatrix} \begin{bmatrix}\mathbf{x}'_{L1} - \hat{\mathbf{x}}'_{L1} \\ \mathbf{x}'_{L2} - \hat{\mathbf{x}}'_{L2}\end{bmatrix} + \begin{bmatrix}\mathbf{F}'_{L1} \\ \mathbf{F}'_{L2}\end{bmatrix} \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M\right) \right\|^2 \\
&\quad + \left\|\mathbf{R}_M \left(\mathbf{x}'_M - \hat{\mathbf{x}}'_M\right)\right\|^2
\end{aligned} \tag{4.109}$$

where

$$\mathbf{Q}_L \begin{bmatrix}\mathbf{R}'_L & \mathbf{F}'_L\end{bmatrix} = \begin{bmatrix}\mathbf{R}_L\mathbf{P}_L^T & \mathbf{F}_L\end{bmatrix} \tag{4.110}$$

Then, we add the cost term from the new loop-closure measurement to the prior term and rearrange some of the matrix blocks, resulting in the total cost $\mathcal{C}_T$:

$$\begin{aligned}\mathcal{C}_T = &\left\| \begin{bmatrix} \mathbf{R}_{T11} & \mathbf{R}_{T12} \\ & \mathbf{R}_{T22} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{L1} - \hat{\mathbf{x}}'_{L1} \\ \mathbf{x}'_B - \hat{\mathbf{x}}'_B \end{bmatrix} \right\|^2 \\ &+ \left\| \begin{bmatrix} \mathbf{H}_{T1} & \mathbf{H}_{T2} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{L1} - \hat{\mathbf{x}}'_{L1} \\ \mathbf{x}'_B - \hat{\mathbf{x}}'_B \end{bmatrix} - \mathbf{r}_T \right\|^2 \end{aligned} \tag{4.111}$$

where

$$\mathbf{R}_{T11} \triangleq \mathbf{R}'_{L11},\ \mathbf{R}_{T12} \triangleq \begin{bmatrix} \mathbf{R}'_{L12} & \mathbf{F}'_{L1} \end{bmatrix} \tag{4.112}$$

$$\mathbf{R}_{T22} \triangleq \begin{bmatrix} \mathbf{R}'_{L22} & \mathbf{F}'_{L2} \\ & \mathbf{R}_M \end{bmatrix} \tag{4.113}$$

Then, we employ the QR factorization of RISE [see (4.86)-(4.88)] to split $\mathcal{C}_T$ into two parts:

$$\begin{aligned}\mathcal{C}_T = &\left\| \begin{bmatrix} \mathbf{R}_{T11} & \mathbf{R}_{T12} \\ & \mathbf{R}_{T22} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{L1} - \hat{\mathbf{x}}'_{L1} \\ \mathbf{x}'_B - \hat{\mathbf{x}}'_B \end{bmatrix} \right\|^2 \\ &+ \left\| \mathbf{H}_{T1}(\mathbf{x}'_{L1} - \hat{\mathbf{x}}'_{L1}) + \mathbf{H}_{T2}(\mathbf{x}'_B - \hat{\mathbf{x}}'_B) - \mathbf{r}_T \right\|^2 \end{aligned} \tag{4.114}$$

$$= \left\| \begin{bmatrix} \mathbf{R}^{\oplus}_{T11} & \mathbf{R}^{\oplus}_{T12} \\ & \mathbf{R}_{T22} \\ & \mathbf{H}^{\oplus}_{T2} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{L1} - \hat{\mathbf{x}}'_{L1} \\ \mathbf{x}'_B - \hat{\mathbf{x}}'_B \end{bmatrix} - \begin{bmatrix} \mathbf{r}^{\oplus}_T \\ \mathbf{0} \\ \mathbf{e}_T \end{bmatrix} \right\|^2$$

$$= \mathcal{C}_F + \mathcal{C}_B \tag{4.115}$$

where

$$\mathcal{C}_F \triangleq \left\| \mathbf{R}^{\oplus}_{T11}(\mathbf{x}'_{L1} - \hat{\mathbf{x}}'^{\oplus}_{L1}) + \mathbf{R}^{\oplus}_{T12}(\mathbf{x}'_B - \hat{\mathbf{x}}'_B) \right\|^2 \tag{4.116}$$

$$\hat{\mathbf{x}}'^{\oplus}_{L1} \triangleq \hat{\mathbf{x}}'_{L1} + \mathbf{R}^{\oplus -1}_{T11}\mathbf{r}^{\oplus}_T \tag{4.117}$$

$$\mathcal{C}_B \triangleq \left\| \begin{bmatrix} \mathbf{R}_{T22} \\ \mathbf{H}^{\oplus}_{T2} \end{bmatrix} (\mathbf{x}'_B - \hat{\mathbf{x}}'_B) - \begin{bmatrix} \mathbf{0} \\ \mathbf{e}_T \end{bmatrix} \right\|^2 \tag{4.118}$$

and the structure of the information factor is shown in Fig. 4.5.

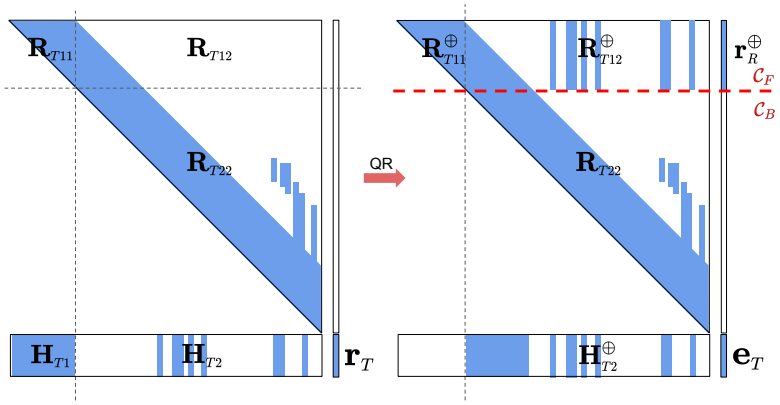


Figure 4.5: Structure of the information factor corresponding to the cost terms before and after the transition step from exploration to relocalization.

Among the operations required in this transition step, the frontend only needs a small-size matrix QR factorization to obtain $\mathcal{C}_F$ with *constant* cost, while all the (expensive) remaining work is done in the backend (see [53] for details).

After this transition step, in the *frontend* $\mathcal{C}_F$ is extended to include information from new poses and features and minimized to update a sliding window of recent states (starting from $\mathbf{x}'_{L1}$). Meanwhile, in the *backend*, $\mathcal{C}_B$ is minimized to provide global corrections for the past states $\mathbf{x}'_B$.

Relocalization: Frontend Thread

The relocalization frontend employs RISE to process both local feature tracks and loop-closure measurements so as to update a sliding window of *recent* states (see ③ in Fig. 4.1). Specifically, the cost function $\mathcal{C}_R$, to be optimized in the frontend, extends from $\mathcal{C}_F$ in (4.116) and has the form

$$\begin{aligned}\mathcal{C}_R = &\left\| \begin{bmatrix} \mathbf{R}_{R11} & \mathbf{R}_{R12} \\ & \mathbf{R}_{R22} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{R1} - \hat{\mathbf{x}}'_{R1} \\ \mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2} \end{bmatrix} \right\|^2 \\ &+ \left\| \begin{bmatrix} \mathbf{H}_{R1} & \mathbf{H}_{R2} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{R1} - \hat{\mathbf{x}}'_{R1} \\ \mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2} \end{bmatrix} - \mathbf{r}_R \right\|^2 \end{aligned} \tag{4.119}$$

where $\mathbf{x}'_{R1}$ contains the window of recent states to be updated at the current step, while $\mathbf{x}'_{R2}$ represents all previous states (including $\mathbf{x}'_B$) whose estimates are kept unchanged in the frontend. The measurement Jacobians $\mathbf{H}_{R1}$ and $\mathbf{H}_{R2}$ are both nonzero (see Fig. 4.6) since here we consider both local feature tracks and loop-closure measurements. Then, the frontend employs RISE to update only $\mathbf{x}'_{R1}$ by first performing the QR factorization to transform $\mathcal{C}_R$ from (4.119) into

$$\begin{aligned}\mathcal{C}_R &= \left\| \begin{bmatrix} \mathbf{R}_{R11} & \mathbf{R}_{R12} \\ \mathbf{H}_{R1} & \mathbf{H}_{R2} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{R1} - \hat{\mathbf{x}}'_{R1} \\ \mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2} \end{bmatrix} - \begin{bmatrix} \mathbf{0} \\ \mathbf{r}_R \end{bmatrix} \right\|^2 \\ &\quad + \left\| \mathbf{R}_{R22}(\mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2}) \right\|^2 \qquad (4.120) \\ &= \left\| \begin{bmatrix} \mathbf{R}^{\oplus}_{R11} & \mathbf{R}^{\oplus}_{R12} \\ & \mathbf{H}^{\oplus}_{R2} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{R1} - \hat{\mathbf{x}}'_{R1} \\ \mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2} \end{bmatrix} - \begin{bmatrix} \mathbf{r}^{\oplus}_R \\ \mathbf{e}_R \end{bmatrix} \right\|^2 \\ &\quad + \left\| \mathbf{R}_{R22}(\mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2}) \right\|^2 \qquad (4.121)\end{aligned}$$

and then dropping the term $\left\| \mathbf{H}^{\oplus}_{R2}(\mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2}) - \mathbf{e}_R \right\|^2$ (since we are not updating $\mathbf{x}'_{R2}$) to get $\bar{\mathcal{C}}_R$

$$\bar{\mathcal{C}}_R = \left\| \begin{bmatrix} \mathbf{R}^{\oplus}_{R11} & \mathbf{R}^{\oplus}_{R12} \\ & \mathbf{R}_{R22} \end{bmatrix} \begin{bmatrix} \mathbf{x}'_{R1} - \hat{\mathbf{x}}'^{\oplus}_{R1} \\ \mathbf{x}'_{R2} - \hat{\mathbf{x}}'_{R2} \end{bmatrix} \right\|^2 \qquad (4.122)$$

$$\hat{\mathbf{x}}'^{\oplus}_{R1} \triangleq \hat{\mathbf{x}}'_{R1} + \mathbf{R}^{\oplus -1}_{R11}\mathbf{r}^{\oplus}_R \qquad (4.123)$$

where $\hat{\mathbf{x}}'^{\oplus}_{R1}$ is the updated state estimate. The structural changes of the factors appearing in (4.119)-(4.121) are shown in Fig. 4.6. The processing cost of this update is defined by the QR factorization, and is *constant* since the size of $\mathbf{x}'_{R1}$ and the number of dense columns in $\mathbf{R}^{\oplus}_{R12}$ are bounded. Due to its low processing cost, the frontend thread is able to update the states at high frequency in real time.

Relocalization: Backend Thread

The backend performs global adjustment so as to accurately update a large number of *past* states, while running in parallel with the frontend to avoid blocking it (see ⑤ in Fig. 4.1). Within RISE, the backend can run whenever the thread is idle, to increase accuracy. In our current design, however, in order to save processing, we choose to run it only once during each

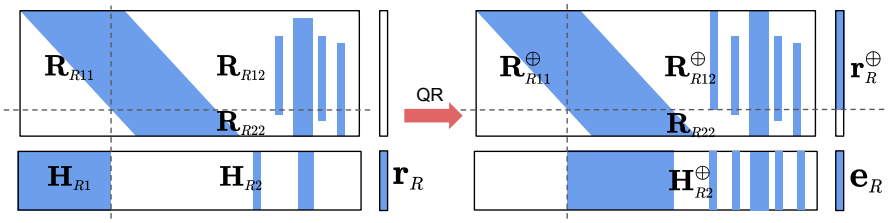


Figure 4.6: Structure of the information factor corresponding to the relocalization cost terms before and after a RISE update in the frontend.

relocalization phase, right after the transition step described in Sec. 4.3.4, and update all the past states to obtain the optimal solution. Specifically, from (4.118), we compute the optimal estimate of the past states $\mathbf{x}'_B$ by minimizing the cost function $\mathcal{C}_B$, which is a batch least-squares problem and can be solved by employing a sparse QR factorization. Once the new estimate $\hat{\mathbf{x}}'^{\oplus}_B$ is obtained, $\mathcal{C}_B$ can be rewritten as (after ignoring a constant term)

$$\mathcal{C}_B = \left\|\mathbf{R}_B(\mathbf{x}'_B - \hat{\mathbf{x}}'^{\oplus}_B)\right\|^2 \tag{4.124}$$

The computational cost is approximately *linear* in the (large) size of $\mathbf{x}'_B$, but it does not prevent the frontend from performing real-time estimation since they run in parallel.

Feedback from Backend to Frontend Thread

After the backend finishes a global update on the *past* states $\mathbf{x}'_B$, the frontend employs this result to update the *recent* states so that they benefit from the global correction (see ④ in Fig. 4.1). Specifically, if we denote all the recent states accumulated in the frontend since the backend started as $\mathbf{x}'_F$, then from (4.122) and Fig. 4.6, the frontend maintains a cost term of the form $\|\mathbf{R}_F(\mathbf{x}'_F - \hat{\mathbf{x}}'_F) + \mathbf{R}_{FB}(\mathbf{x}'_B - \hat{\mathbf{x}}'_B)\|^2$, where the information factor $\mathbf{R}_F$ is upper-triangular and

$\mathbf{R}_{FB}$ represents the cross information between the states in the frontend and the backend. Combining this frontend's cost term with that of the backend [$\mathcal{C}_B$ in (4.124)] yields

$$\begin{aligned}\mathcal{C}_{FB} &= \left\|\mathbf{R}_F(\mathbf{x}'_F - \hat{\mathbf{x}}'_F) + \mathbf{R}_{FB}(\mathbf{x}'_B - \hat{\mathbf{x}}'_B)\right\|^2 \\ &\quad + \left\|\mathbf{R}_B(\mathbf{x}'_B - \hat{\mathbf{x}}'^{\oplus}_B)\right\|^2 \\ &= \left\| \begin{bmatrix} \mathbf{R}_F & \mathbf{R}_{FB} \\ & \mathbf{R}_B \end{bmatrix} \begin{bmatrix} \mathbf{x}'_F - \hat{\mathbf{x}}'^{\oplus}_F \\ \mathbf{x}'_B - \hat{\mathbf{x}}'^{\oplus}_B \end{bmatrix} \right\|^2 \end{aligned} \tag{4.125}$$

$$\hat{\mathbf{x}}'^{\oplus}_F \triangleq \hat{\mathbf{x}}'_F + \mathbf{R}_F^{-1}\mathbf{R}_{FB}\left(\hat{\mathbf{x}}'_B - \hat{\mathbf{x}}'^{\oplus}_B\right) \tag{4.126}$$

where $\hat{\mathbf{x}}'^{\oplus}_F$ is the globally corrected estimate for the recent states in the frontend. The operations required in (4.126) are sparse matrix-vector multiplications and back substitutions, which are faster than the operations of the frontend in practice.

As a result of this step, all current states in the frontend are *immediately* corrected using the globally adjusted estimates from the backend. Note that, this is a key difference between our algorithm and existing multi-thread SLAM systems. Specifically, [63, 71, 77] solve *separate* optimization problems independently in different threads; hence their frontend has to rely on map reobservations *after* the backend's global adjustment finishes in order to obtain corrections. This is due to the fact that the two sets of states involved in their frontend's and backend's (*separate*) optimization problems are considered *uncorrelated*. In contrast, our algorithm always solves a *single* optimization problem over two *correlated* sets of states [see (4.125)], carried out in two separate threads. Consequently, as soon as the corrections from the backend become available, they immediately affect the frontend's states, even if the map is not reobserved.

4.3.5 Experimental Results

To evaluate the performance of the proposed RISE-SLAM, we compared it against state-of-the-art visual-inertial SLAM estimators, including a visual-inertial odometry (VIO) system without loop closures (OKVIS [58]), and SLAM systems with loop closures (VINS-Mono [77], ICE-BA [63]), using the EuRoC [18] datasets. Since our implementation focuses on visual-inertial SLAM, we did not compare to vision-only systems, such as [71]. The datasets contain stereo images (only the left-camera images are used) from global shutter cameras (20 Hz) and IMU

measurements (200 Hz), along with ground-truth poses from VICON. The code of each compared system is downloaded from their Github repositories and run with the provided configuration files for the EuRoC datasets.

System Setup

In our implementation, we extract 300 ORB [79] features per image and match them based on their descriptors against previous images to generate feature tracks. Loop-closure measurements are provided by a vocabulary tree [75], and the system switches to relocalization mode when detecting features that have not been observed for more than 15 sec. We model the noise of all visual observations as zero-mean white Gaussian with $\sigma = 1.5$ pixels. During both exploration and relocalization, the frontend updates a sliding window corresponding to 10 recent poses and their observed features. To improve efficiency, feature tracks longer than 20 are split into multiple shorter ones while the number of feature tracks being processed per time step is limited to 40.

Localization Accuracy

We compute the root-mean-square error (RMSE) of the estimated positions as compared to the ground truth, to evaluate the tracking accuracy of all estimators considered. Each estimated trajectory is aligned with the ground-truth coordinate frame by a 3D-to-3D matching using [43].

For fairness, we first run the proposed RISE-SLAM without using any loop-closure measurements [RISE (vio)] so as to compare it with the VIO system OKVIS. Then, we add loop-closure measurements [RISE (lc)] to compare it with the SLAM systems VINS-Mono and ICE-BA. The position RMSE results from all EuRoC datasets are shown in Table 4.1. As shown, our RISE (vio) outperforms OKVIS on all datasets. This is due to the fact that RISE optimally processes all available measurements during exploration (see Sec. 4.3.3). Among the SLAM systems, ICE-BA performs the worst. This is probably because loop closure is not implemented yet in their public code, which makes their system a VIO with a backend that performs global smoothing. Regardless, ICE-BA does not achieve better accuracy even when compared to RISE (vio), while requiring more CPU resources for running a backend thread. Meanwhile, our RISE (lc) is the best on most of the datasets, and outperforms VINS-Mono.

Table 4.1: Position RMSE (cm) on EuRoC datasets

Dataset	RISE (vio)	OKVIS	RISE (lc)	VINS-Mono	ICE-BA
MH_01_easy	**16.1**	34.6	**5.1**	8.8	16.4
MH_02_easy	**19.3**	40.9	12.0	**6.3**	8.9
MH_03_medium	**21.9**	22.1	**8.4**	8.5	17.2
MH_04_difficult	**24.1**	33.7	**16.1**	17.5	31.6
MH_05_difficult	**30.5**	42.5	25.6	**14.5**	31.4
V1_01_easy	**5.1**	10.6	**4.1**	4.5	5.2
V1_02_medium	**7.0**	11.6	**3.3**	6.5	11.7
V1_03_difficult	**11.6**	22.7	**6.7**	29.9	11.1
V2_01_easy	**6.6**	16.2	**5.5**	6.4	9.1
V2_02_medium	**10.4**	18.1	**4.0**	13.3	11.5
V2_03_difficult	**10.1**	28.5	**11.7**	18.0	12.8
Average	**14.8**	25.6	**9.3**	12.2	15.2

Table 4.2: Average running time (msec) of one estimator run

RISE exploration	RISE relocalization	RISE overall	OKVIS	VINS-Mono	ICE-BA
10	13	11	26	52	11

Computational Efficiency

We run all algorithms on a laptop with Intel Core i7-6700HQ 2.60GHz x 8 CPU to compare the efficiency of each estimator. Since we focus on estimation speed, we do not consider the cost of image processing. Average times of one run are shown in Table 4.2. Note that since RISE-SLAM has two modes with different costs, we report the averages of exploration and relocalization (the time of transition steps is also included), respectively, as well as the overall average across all runs. As expected, the exploration is typically faster than relocalization since it only processes local feature tracks. As evident from Table 4.2, RISE-SLAM is significantly faster than VINS-Mono and OKVIS. Only ICE-BA runs as fast as RISE-SLAM, but has lower tracking accuracy.[4]

Estimation Consistency

To assess the consistency improvement of RISE as compared to estimators that assume some past states to be perfectly known, we employ the normalized estimation error squared

[4]Although we did not compare to ORB-SLAM as they do not use IMU, it is worth noting that as reported in [63] ORB-SLAM is more than 10 times slower than [63] whose average processing time is almost the same as ours.

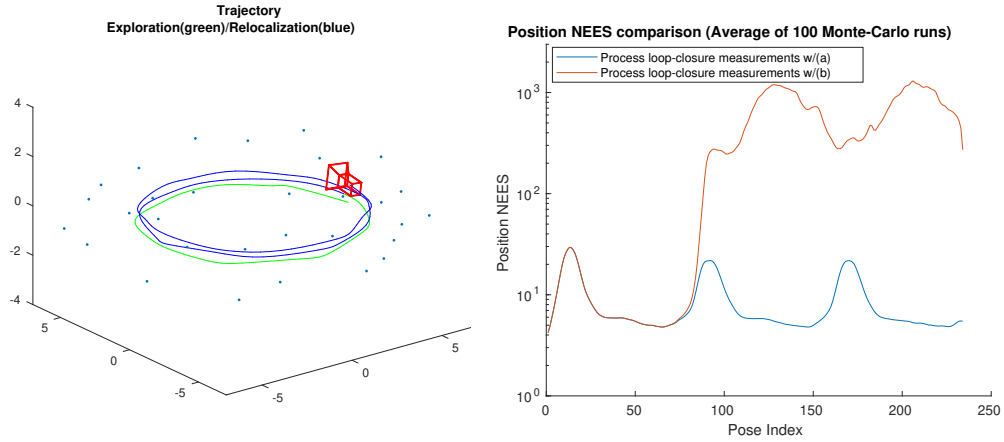


Figure 4.7: NEES comparison in simulation: We run RISE-SLAM exploration mode (i.e., the optimal estimator) during the first loop, and for the second and third loops where there are loop-closure measurements, we employ either: (a) RISE-SLAM relocalization mode, or (b) assuming past poses and features as perfectly known (i.e., zeroing out Jacobians w.r.t. them).

(NEES) [11]. Typically more overconfident (inconsistent) estimates yield larger NEES. To isolate the effect of other factors affecting consistency (e.g., local minima, outliers), we first evaluate NEES in Monte Carlo simulations where the camera follows a circular path three times (see Fig. 4.7). In this case, the RISE-SLAM's NEES fluctuates around 10, while if the map is assumed perfectly-known the NEES increases to the level of 10^3. Furthermore, on EuRoC datasets, VINS-Mono, which assumes perfectly-known keyframe poses, has a median position NEES of over 100, while that of RISE-SLAM is 38.7. The medians of the NEES are shown in Table 4.3, while the NEES distribution on each dataset is shown in Fig. 4.8.

4.4 Summary

Motivated by the consistency guarantees and the linear processing cost of the Schmidt-Kalman filter (SKF), as well as the linear memory requirements of the Hessian's Cholesky factor of maps computed online, in this chapter, we derived the exact inverse Schmidt estimator (ISE), as an equivalent of the SKF but in the information form. When applied to the visual-inertial SLAM problem, however, no computational saving is achieved by using the exact ISE. Therefore, we further introduced the resource-aware inverse Schmidt estimator (RISE), as an approximation to the exact ISE, that maintains the sparsity of the Hessian's Cholesky factor and improves

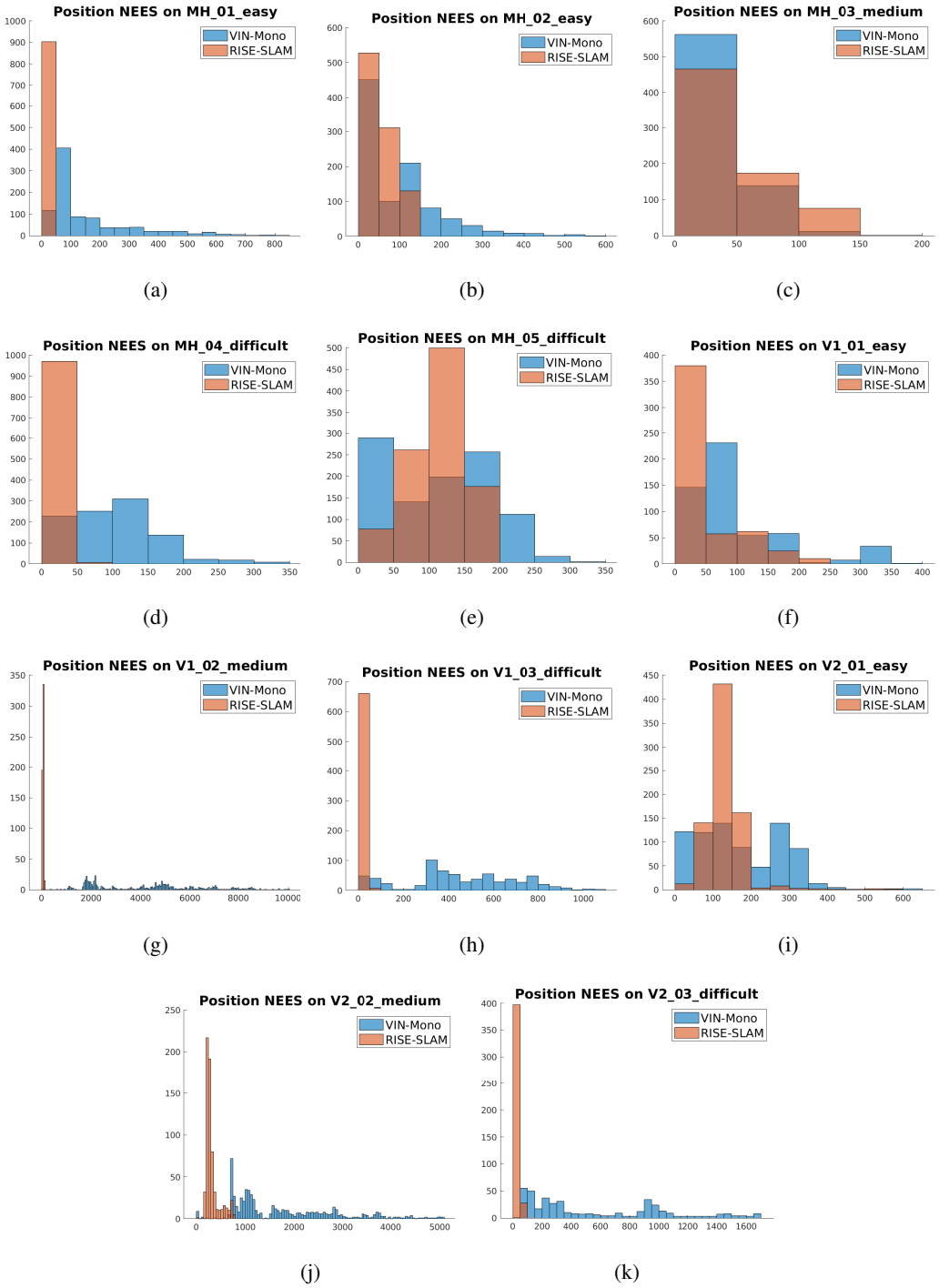


Figure 4.8: Histograms of position NEES on EuRoC datasets (VINS-Mono vs. RISE-SLAM)

Table 4.3: Medians of position NEES on EuRoC datasets

Dataset	RISE-SLAM	VINS-Mono
MH_01_easy	**4.7**	87.0
MH_02_easy	**47.9**	70.4
MH_03_medium	32.7	**25.1**
MH_04_difficult	**18.2**	101.8
MH_05_difficult	**115.7**	119.4
V1_01_easy	**36.6**	65.8
V1_02_medium	**58.7**	4405.8
V1_03_difficult	**6.8**	420.8
V2_01_easy	**128.4**	152.8
V2_02_medium	**278.4**	1323.0
V2_03_difficult	**20.4**	337.4
Overall	**38.7**	129.1

computational efficiency. We employed different (in terms of the order and number of states involved) configurations of the RISE to form the frontend and backend of a visual-inertial system, RISE-SLAM, which appropriately treats the exploration versus the relocalization phases of SLAM to achieve real-time operation. As demonstrated by the experimental evaluation, RISE-SLAM achieves position accuracy typically better than alternative state-of-the-art visual-inertial estimators, at lower processing cost while improving estimation consistency.

Chapter 5

Concluding Remarks

5.1 Summary of Contributions

The work presented in this dissertation has focused on resolving key challenges of localization and mapping using an IMU and a camera. In particular, we focused on improving the efficiency of short-term VI-SLAM, the accuracy of planar VI-SLAM, and the efficiency and consistency of long-term VI-SLAM. The main contributions of this work can be summarized as follows:

- **Efficient Short-Term VI-SLAM**

  In Chapter 2, we studied the problem of designing a state estimator for visual-inertial odometry (VIO), and for the first time ever, presented a VIO estimator in the square-root information domain, the SR-ISWF, which enables single-precision numerical representation and arithmetic for implementation. This leads to significant speedups of the program, especially on mobile devices. Additionally, as a theoretical foundation, we established the equivalence and correspondence between the filtering-based and optimization-based methods, especially on their (re-)linearization behaviors. Extensions of the basic filters were discussed, for performing the same relinearization processes as in the optimization-based methods. Based on these analysis, we introduced a hybrid visual-information management and processing scheme, that is suitable for any sparse-feature-based sliding-window VIO system, for balancing between estimation accuracy and computational efficiency. Under this information processing scheme, we derived in detail all steps of our estimation algorithm in the square-root inverse form. In particular, we showed the

specific problem structures (i.e., the nonzero patterns of the information factor and Jacobian matrices) and exploited them to obtain an efficient implementation of the SR-ISWF algorithm. Moreover, some of the most important structure findings and our approach for handling them, to achieve significant computational savings, are applicable to other popular VIO estimators, such as the MSC-KF and its extensions. Finally, a complete VIO system with our SR-ISWF algorithm was implemented and tested. Our system outperforms alternative state-of-the-art VIO systems on public datasets, providing better pose tracking accuracy, with significantly reduced processing time. Experiments on smart phones demonstrate our system's capability for faster than real-time operations on resource-constrained mobile devices.

- **Accurate Planar VI-SLAM**

  In Chapter 3, we discovered the fundamental cause of the large localization error of any visual-inertial navigation system (VINS), when deployed on ground robots moving primarily on a plane. In particular, for the first time ever, we proved that the VINS scale, or 2 additional dof (i.e., roll and pitch) of its global orientation, become unobservable when the robot moves with constant acceleration, or it is not rotating, respectively. For this reason, directly employing VINS on a ground robot results in inaccurate pose estimates. Moreover, as compared to existing observability analysis of VINS under special motion profiles [67], we considered the most general case of unknown gyroscope biases and determined these unobservable directions *analytically*. Through experiments, we validated these key findings of our theoretical analysis, where results showed that special motions, even approximately as in practice, indeed lead to larger positioning errors when using VINS on a ground robot.

  To address this issue and achieve accurate localization results for wheeled robots, we extended the VINS algorithm to incorporate two extra sources of information: wheel-encoder data and planar-motion constraints. Specifically, in order to process the noisy odometer data in a robust manner, we first integrated them into 2D displacement estimates, and then fused these inferred measurements with the 3D VINS. As a result, these odometer measurements ensure that the scale is always observable, even under special motions. Additionally, we introduced the manifold-(m)VINS that properly models the ground robot's almost-planar motion (as stochastic constraints) and directly employs this

information in the VINS estimator. Finally, built upon our efficient VINS (i.e., the SR-ISWF presented in Chapter 2), a complete system using our extended VINS algorithm for localizing wheeled robots was implemented and tested. Experimental results show that our system achieves significantly-increased localization accuracy when deployed on a tablet onboard a wheeled robot that navigates within a large-scale building.

- **Efficient and Consistent Long-Term VI-SLAM**

  In Chapter 4, we provided an efficient algorithm for long-term VI-SLAM, that improves estimation consistency as compared to existing approaches, by leveraging the idea of the Schmidt-Kalman filter (SKF). Specifically, motivated by the consistency guarantees and the linear processing cost of the SKF, as well as the linear memory requirements of the Hessian's Cholesky factor of maps computed online, for the first time ever, we derived the exact inverse Schmidt estimator (ISE), as an equivalent of the SKF but in the information form. We applied the exact ISE to the long-term VI-SLAM problem, and found out its limitations: The exact ISE provides no computational saving as compared to the optimal estimator, while introducing a large number of fill-ins to the Hessian's Cholesky factor. Therefore, to address this issue, we further derived the resource-aware inverse Schmidt estimator (RISE), which approximates the exact ISE and has adjustable processing cost, while preserving sparsity and ensuring consistency. By adjusting the size of the window of the states to be updated, the RISE provides a mechanism to trade estimation accuracy for computational efficiency.

  By employing the RISE with different configurations, in terms of the order and number of states involved, we developed an accurate and efficient algorithm for long-term VI-SLAM, the RISE-SLAM. In particular, when navigating through a new area, the RISE-SLAM optimally processes visual-inertial measurements to build a 3D map of the scene; Meanwhile, when revisiting a previously-mapped area, the RISE-SLAM is able to process loop-closure measurements for relocalizing the pose in a consistent manner with constant cost. Finally, a complete system using our RISE-SLAM algorithm for long-term VI-SLAM was implemented and tested. Experimental results show that, as compared to state-of-the-art approaches, our system performs better in terms of localization accuracy and processing time, while improving estimation consistency.

With this work, we provide theoretical and algorithmic foundations for improving the efficiency, accuracy, and consistency of visual-inertial based 3D localization and mapping systems, that will benefit a wide range of robotics applications, such as indoor navigation, self-driving vehicles, and augmented or virtual reality.

5.2 Future Research Directions

In our work of the short-term VI-SLAM, one important problem to be addressed is information selection. Given the large number of feature observations per image, we need to select a portion of all available measurements for processing, in order to bound the computational cost of the VIO algorithm, while ensuring the loss of accuracy is minimal. To obtain the optimal solution of this selection problem, however, one will end up spending more computations in the selection algorithm itself than simply processing all available measurements. Therefore, heuristics based on prior knowledge or assumption of the VIO problem needs to be used. In our current work, we follow two guidelines when choosing feature measurements: using features with long track lengths and enforcing a uniform distribution of the features in the image. While these heuristics are shown to work well in nominal cases, they provide no guarantee on the performance of the VIO algorithm, and may not generalize for other scenarios. As a future work, it would be interesting to develop a visual measurement selection algorithm for VIO that has certain guarantees on the output accuracy, e.g., based on the posterior covariance. Moreover, the system can be extended to use multiple cameras for increased accuracy and robustness, where these cameras may point at different directions. In this case, the measurement selection problem becomes more complicated, as we need allocate the total processing resource among all these cameras, considering their geometric configurations and scene structures.

In the presented observability analysis of VI-SLAM under special motions, we have made the assumption that all intrinsic and extrinsic parameters are known as prior knowledge. In practice, however, due to the lack of precise calibration or to compensate for changes over time, it is common that some of these parameters are also under estimation. In this case, a fundamental problem to be answered is that, under which motion profiles are these parameters observable? In other words, we need to: i) Determine the singular motions that make a certain parameter unobservable; ii) Be able to detect these singular cases when they occur; iii) Have a mechanism to deal with such cases to ensure the accuracy and consistency of the estimates.

Lastly, as for our work of the long-term VI-SLAM, there are several potential directions for improving the current RISE-SLAM algorithm. One major drawback of our RISE-SLAM system is the low speed of the backend thread. Specifically, since we keep all the pose and feature states in the estimation problem, the size of the entire state vector increases linearly with time. Although the frontend thread guarantees constant processing time by design, the backend thread, which optimizes over the entire state history, has an increasing computational cost at least linearly with time. This may cause a significant delay on the feedback from the backend thread to the frontend thread, which will lead to degraded accuracy of the system. To resolve this issue, one potential approach would be to employ keyframing for the backend, i.e., to select a subset of poses and features that provide sufficient information for building a map of the scene, while significantly reducing the number of states to be optimized in the backend. Another remedy would be to apply the RISE again in the backend, so that it only updates the states up to some point back in the history (e.g., till the last loop-closure event), instead of all the states.

References

[1] Apple, ARKit, https://developer.apple.com/augmented-reality/.

[2] Facebook, Oculus VR, https://www.oculus.com/.

[3] GM, Cruise, https://www.getcruise.com/.

[4] Google, ARCore, https://developers.google.com/ar.

[5] Google, Waymo, https://waymo.com/.

[6] Microsoft, HoloLens, https://www.microsoft.com/en-us/hololens.

[7] Project Tango, https://developers.google.com/tango/.

[8] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org.

[9] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*, ser. Information and System Sciences Series. Englewood Cliffs, NJ: Prentice-Hall Inc., 1979.

[10] T. Bailey and H. Durrant-Whyte, "Simultaneous localization and mapping (slam): part ii," *IEEE Robotics Automation Magazine*, vol. 13, no. 3, pp. 108–117, Sept 2006.

[11] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[12] D. Bernstein, *Matrix Mathematics*. Princeton University Press, 2005.

[13] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena Scientific, 1999.

[14] G. J. Bierman, *Factorization Methods for Discrete Sequential Estimation*, ser. Mathematics in Science and Engineering. New York, NY: Academic Press, 1977, vol. 128.

[15] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Hamburg, Germany, Sep. 28 – Oct. 2 2015, pp. 298–304.

[16] M. Bloesch, M. Burri, S. Omari, M. Hutter, and R. Siegwart, "Iterated extended Kalman filter based visual-inertial odometry using direct photometric feedback," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1053–1072, 2017.

[17] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, "Simultaneous localization and mapping: A survey of current trends in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 2, pp. 194–220, 2017.

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

[19] C. Cadena, L. Carlone, H. Carrillo, Y. Latif, D. Scaramuzza, J. Neira, I. Reid, and J. J. Leonard, "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age," *IEEE Transactions on Robotics*, vol. 32, no. 6, pp. 1309–1332, 2016.

[20] F. Capezio, F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "Robot-assisted surveillance in large environments," *Journal of Computing and Information Technology*, vol. 17, no. 1, pp. 95–108, mar 2009.

[21] J. Casper and R. R. Murphy, "Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 33, no. 3, pp. 367–385, June 2003.

[22] Z. Chen, K. Jiang, and J. C. Hung, "Local observability matrix and its application to observability analyses," in *Proc. of 16th Annual Conference IEEE Industrial Electronics Society*, Pacific Grove, CA, Nov. 27–30 1990, pp. 100–103.

[23] J. Civera, A. J. Davison, and J. M. M. Montiel, "Inverse depth parametrization for monocular slam," *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 932–945, Oct. 2008.

[24] F. Dellaert and M. Kaess, "Square Root SAM: Simultaneous localization and mapping via square root information smoothing," *International Journal of Robotics Research*, vol. 25, no. 12, pp. 1181–1203, Dec. 2006.

[25] T.-C. Dong-Si and A. I. Mourikis, "Motion tracking with fixed-lag smoothing: Algorithm and consistency analysis," in *Proc. of the IEEE International Conference on Robotics and Automation*, Shanghai, China, May 9 – 13 2011, pp. 5655–5662.

[26] H. Durrant-Whyte and T. Bailey, "Simultaneous localisation and mapping (slam): Part i the essential algorithms," *IEEE Robotics and Automation Magazine*, vol. 13, no. 2, pp. 99–110, june 2006.

[27] R. C. DuToit, J. A. Hesch, E. D. Nerurkar, and S. I. Roumeliotis, "Consistent map-based 3D localization on mobile devices," in *Proc. of the IEEE International Conference on Robotics and Automation*, 2017, pp. 6253–6260.

[28] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, March 2018.

[29] J. Engel, J. Sturm, and D. Cremers, "Semi-dense visual odometry for a monocular camera," in *Proc. of the IEEE International Conference on Computer Vision*, Sydney, Australia, Dec.1–8 2013, pp. 1449–1456.

[30] J. Engel, T. Schöps, and D. Cremers, "LSD-SLAM: Large-scale direct monocular SLAM," in *Proc. of the European Conference on Computer Vision*, Zurich, Switzerland, Sep.6–12 2014, pp. 834–849.

[31] J. K. Erickson, "Living the dream - an overview of the mars exploration project," *IEEE Robotics Automation Magazine*, vol. 13, no. 2, pp. 12–18, June 2006.

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

[33] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual-inertial odometry," *IEEE Trans. on Robotics*, vol. 33, no. 1, pp. 1–21, Feb 2017.

[34] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "Imu preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation," in *Proc. of Robotics: Science and Systems*, Rome, Italy, Jul. 13-17 2015.

[35] C. Forster, Z. Zhang, M. Gassner, M. Werlberger, and D. Scaramuzza, "SVO: Semidirect visual odometry for monocular and multicamera systems," *IEEE Trans. on Robotics*, vol. 33, no. 2, pp. 249–265, April 2017.

[36] P. Geneva, K. Eckenhoff, W. Lee, Y. Yang, and G. Huang, "OpenVINS: A research platform for visual-inertial estimation," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, China, Nov. 4–8 2019.

[37] G. Golub and C. Van Loan, *Matrix Computations*, 4th ed. Johns Hopkins University Press, 2013.

[38] J. E. Guivant and E. M. Nebot, "Optimization of the simultaneous localization and map-building algorithm for real-time implementation," *IEEE Trans. on Robotics and Automation*, vol. 17, no. 3, pp. 242–257, 2001.

[39] C. Guo, D. G. Kottas, R. DuToit, A. Ahmed, R. Li, and S. I. Roumeliotis, "Efficient visual-inertial navigation using a rolling-shutter camera with inaccurate timestamps," in *Proc. of Robotics: Science and Systems*, Berkeley, CA, July 12 – 16 2014.

[40] C. X. Guo, F. M. Mirzaei, and S. I. Roumeliotis, "An analytical least-squares solution to the odometer-camera extrinsic calibration problem," in *Proc. of the IEEE International Conference on Robotics and Automation*, Saint Paul, Minnesota, May 14–18 2012, pp. 3962–3968.

[41] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. Cambridge University Press, March 2004.

[42] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis, "Consistency analysis and improvement of vision-aided inertial navigation," *IEEE Trans. on Robotics*, vol. 30, no. 1, pp. 158–176, Feb. 2014.

[43] B. K. Horn, "Closed-form solution of absolute orientation using unit quaternions," *JOSA A*, vol. 4, no. 4, pp. 629–642, 1987.

[44] Z. Huai and G. Huang, "Robocentric visual-inertial odometry," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Madrid, Spain, Oct. 1–5 2018, pp. 6319–6326.

[45] G. Huang, "Visual-inertial navigation: A concise review," in *Proc. of the IEEE International Conference on Robotics and Automation*, Montreal, Canada, May 20–24 2019, pp. 9572–9582.

[46] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "An observability-constrained sliding window filter for slam," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, Sep. 25–30 2011, pp. 65 – 72.

[47] ——, "Observability-based rules for designing consistent ekf slam estimators," *International Journal of Robotics Research*, vol. 29, no. 5, pp. 502–528, Apr. 2010.

[48] A. H. Jazwinski, *Stochastic Processes and Filtering Theory*. New York, NY: Academic Press, 1970.

[49] S. J. Julier and J. K. Uhlmann, "A non-divergent estimation algorithm in the presence of unknown correlations," in *Proc. of the American Control Conference*, vol. 4, Albuquerque, NM, Jun. 4–6 1997, pp. 2369–2373.

[50] S. J. Julier, "A sparse weight Kalman filter approach to simultaneous localisation and map building," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, Maui, HI, Oct. 29 – Nov. 3 2001, pp. 1251–1256.

[51] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard, and F. Dellaert, "iSAM2: Incremental smoothing and mapping using the bayes tree," *International Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, February 2012.

[52] M. Kaess, A. Ranganathan, and F. Dellaert, "iSAM: Incremental smoothing and mapping," *IEEE Trans. on Robotics*, vol. 24, no. 6, pp. 1365–1378, December 2008.

[53] T. Ke, K. J. Wu, and S. I. Roumeliotis, "Visual-inertial SLAM with inverse Schmidt estimators," http://mars.cs.umn.edu/tr/risetechreport.pdf.

[54] G. Klein and D. Murray, "Parallel tracking and mapping for small AR workspaces," in *Proc. of the IEEE International Symposium on Mixed and Augmented Reality*, Nara, Japan, Nov.11–16 2007, pp. 225–234.

[55] L. Kneip, M. Chli, and R. Siegwart, "Robust real-time visual odometry with a single camera and an IMU," in *Proc. of The British Machine Vision Conference*, Dundee, Scotland, August 29 - September 2 2011, pp. 1–11.

[56] K. Konolige and M. Agrawal, "Frameslam: From bundle adjustment to real-time visual mapping," *IEEE Trans. on Robotics*, vol. 24, no. 5, pp. 1066–1077, Oct 2008.

[57] D. G. Kottas and S. I. Roumeliotis, "An iterative Kalman smoother for robust 3D localization on mobile and wearable devices," in *Proc. of the IEEE International Conference on Robotics and Automation*, Seattle, Washington, May 26 – 30 2015, pp. 6336–6343.

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

[59] M. Li, B. H. Kim, and A. I. Mourikis, "Real-time motion tracking on a cellphone using inertial sensing and a rolling-shutter camera," in *Proc. of the IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6-10 2013, pp. 4697–4704.

[60] M. Li and A. I. Mourikis, "Optimization-based estimator design for vision-aided inertial navigation," in *Proc. of Robotics: Science and Systems*, Sydney, Australia, July 9 –13 2012, pp. 241–248.

[61] ——, "High-precision, consistent EKF-based visual-inertial odometry," *International Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, June 2013.

[62] M. Li, H. Yu, X. Zheng, and A. I. Mourikis, "High-fidelity sensor modeling and calibration in vision-aided inertial navigation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Hong Kong, China, May 31 - June 7 2014, pp. 409–416.

[63] H. Liu, M. Chen, G. Zhang, H. Bao, and Y. Bao, "ICE-BA: Incremental, consistent and efficient bundle adjustment for visual-inertial SLAM," in *Proc. of the IEEE Conference*

on Computer Vision and Pattern Recognition, Salt Lake City, UT, Jun.18-22 2018, pp. 1974–1982.

[64] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Proc. of the International Joint Conference on Artificial Intelligence*, Vancouver, British Columbia, Aug. 24–28 1981, pp. 674–679.

[65] S. Lynen, T. Sattler, M. Bosse, J. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization," in *Proc. of Robotics: Science and Systems*, Rome, Italy, Jul. 13-17 2015.

[66] A. Martinelli, "Vision and imu data fusion: Closed-form solutions for attitude, speed, absolute scale, and bias determination," *IEEE Trans. on Robotics*, vol. 28, no. 1, pp. 44–60, Feb. 2012.

[67] ——, "Closed-form solution of visual-inertial structure from motion," *International Journal of Computer Vision*, vol. 106, no. 2, pp. 138–152, 2013.

[68] P. S. Maybeck, *Stochastic Models, Estimation and Control*. New York, NY: Academic Press, 1979, vol. 1.

[69] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint Kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Rome, Italy, Apr. 10–14 2007, pp. 3482–3489.

[70] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, A. E. Johson, A. Ansar, and L. Matthies, "Vision-aided inertial navigation for spacecraft entry, descent, and landing," *IEEE Trans. on Robotics*, vol. 25, no. 2, pp. 264–280, Apr. 2009.

[71] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "ORB-SLAM: A versatile and accurate monocular slam system," *IEEE Trans. on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[72] R. Mur-Artal and J. D. Tardós, "Visual-inertial monocular SLAM with map reuse," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 796–803, 2017.

[73] E. D. Nerurkar and S. I. Roumeliotis, "Power-SLAM: A linear-complexity, anytime algorithm for SLAM," *The International Journal of Robotics Research*, vol. 30, no. 6, pp. 772–788, 2011.

[74] E. D. Nerurkar, K. J. Wu, and S. I. Roumeliotis, "C-KLAM: Constrained keyframe-based localization and mapping for long-term navigation," in *Proc. of the IEEE International Conference on Robotics and Automation*, Hong Kong, China, May 31 – Jun. 7 2014, pp. 3638–3643.

[75] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, New York, NY, Jun.17–22 2006, pp. 2161–2168.

[76] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. Springer, 2006.

[77] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. on Robotics*, vol. 34, no. 99, pp. 1–17, 2018.

[78] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. of the 9th European Conference on Computer Vision*, vol. 3951, Graz, Austria, May 7–13 2006, pp. 430–443.

[79] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "ORB: An efficient alternative to SIFT or SURF," in *Proc. of the IEEE International Conference on Computer Vision*, Barcelona, Spain, Nov. 6–13 2011, pp. 2564–2571.

[80] S. F. Schmidt, "Application of state-space methods to navigation problems," in *Advances in control systems*. Elsevier, 1966, vol. 3, pp. 293–340.

[81] S. Shen, N. Michael, and V. Kumar, "Autonomous multi-floor indoor navigation with a computationally constrained mav," in *2011 IEEE International Conference on Robotics and Automation*, Shanghai, China, May 2011, pp. 20–25.

[82] G. Sibley, L. Matthies, and G. Sukhatme, "Sliding window filter with application to planetary landing," *Journal of Field Robotics*, vol. 27, no. 5, pp. 587–608, Aug. 2010.

[83] S. Thrun, Y. Liu, D. Koller, A. Y. Ng, Z. Ghahramani, and H. Durrant-Whyte, "Simultaneous localization and mapping with sparse extended information filters," *International Journal of Robotics Research*, vol. 23, no. 7-8, pp. 693–716, August 2004.

[84] S. Thrun and M. Montemerlo, "The graph SLAM algorithm with applications to large-scale mapping of urban structures," *The International Journal of Robotics Research*, vol. 25, no. 5-6, pp. 403–429, 2006.

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

[86] N. Trawny and S. I. Roumeliotis, "Indirect kalman filter for 3d attitude estimation," University of Minnesota, Dept. of Comp. Sci. & Eng., Tech. Rep., March 2005.

[87] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment - a modern synthesis," in *Proc. of the International Workshop on Vision Algorithms*, Springer, Verlag, 1999, pp. 298–372.

[88] M. R. Walter, R. M. Eustice, and J. J. Leonard, "Exactly sparse extended information filters for feature-based SLAM," *International Journal of Robotics Research*, vol. 26, no. 4, pp. 335–359, April 2007.

[89] K. J. Wu, A. Ahmed, G. Georgiou, and S. I. Roumeliotis, "A square root inverse filter for efficient vision-aided inertial navigation on mobile devices," in *Proc. of Robotics: Science and Systems*, Rome, Italy, Jul. 13-17 2015.

[90] K. J. Wu and S. I. Roumeliotis, "Inverse Schmidt estimators," http://mars.cs.umn.edu/tr/inverseschmidt.pdf.

Appendix A

Appendices for Chapter 3

A.1 Proof of Theorem 1

In this section, we prove that the scale in (3.6) is an unobservable direction of the VINS model, if and only if the platform is moving with constant *local* linear acceleration [see (3.5)]. We follow the approach presented in [42], that examines the right null space of the observability matrix of the corresponding linearized VINS model. As is the case in [42], and for clarity of presentation, we include only one feature in the state vector (the extension to multiple features is straightforward).

As previously shown (see (51) in [42]), any block row, $\mathbf{M}_k$, of the observability matrix has the following structure:

$$\mathbf{M}_k = \mathbf{H}_k \mathbf{\Phi}_{k,1} = \mathbf{\Gamma}_1 \Big[\mathbf{\Gamma}_2 \;\; \mathbf{\Gamma}_3 \;\; -\delta t_k \mathbf{I}_3 \;\; \mathbf{\Gamma}_4 \;\; -\mathbf{I}_3 \;\; \mathbf{I}_3 \Big] \tag{A.1}$$

for any time $t_k \geq t_0$, with the matrices $\mathbf{\Gamma}_i, i = 1, \ldots, 4$, defined by (52)-(55) in [42]. From the property of the observability matrix, the scale direction, $\mathbf{N}_s$, is unobservable, if and only if, $\mathbf{M}_k \mathbf{N}_s = \mathbf{0}$ [22]. From (A.1) and (3.6), together with the definition of the matrices $\mathbf{\Gamma}_i$, we

obtain:

$$\mathbf{M}_k\mathbf{N}_s = \mathbf{\Gamma}_1(-{}^G\mathbf{v}_{I_0}\delta t_k - \mathbf{\Gamma}_4{}^I\mathbf{a} - {}^G\mathbf{p}_{I_0} + {}^G\mathbf{f}) \tag{A.2}$$

$$\text{with } \mathbf{\Gamma}_4{}^I\mathbf{a} = \int_{t_0}^{t_k}\int_{t_0}^{s} {}^G_{I_\tau}\mathbf{C}\, d\tau ds \cdot {}^I\mathbf{a} \tag{A.3}$$

$$= \int_{t_0}^{t_k}\int_{t_0}^{s} {}^G_{I_\tau}\mathbf{C}{}^I\mathbf{a}\, d\tau ds \tag{A.4}$$

$$= \int_{t_0}^{t_k}\int_{t_0}^{s} {}^G_{I_\tau}\mathbf{C}{}^{I_\tau}\mathbf{a}(\tau)\, d\tau ds \tag{A.5}$$

$$= \int_{t_0}^{t_k}\int_{t_0}^{s} {}^G\mathbf{a}(\tau)\, d\tau ds \tag{A.6}$$

$$= \int_{t_0}^{t_k} ({}^G\mathbf{v}_{I_s} - {}^G\mathbf{v}_{I_0})\, ds \tag{A.7}$$

$$= {}^G\mathbf{p}_{I_k} - {}^G\mathbf{p}_{I_0} - {}^G\mathbf{v}_{I_0}\delta t_k \tag{A.8}$$

where the equality from (A.4) to (A.5) holds if and only if the constant acceleration assumption in (3.5) is satisfied. Substituting (A.8) into (A.2) yields:

$$\begin{aligned}\mathbf{M}_k\mathbf{N}_s &= \mathbf{\Gamma}_1({}^G\mathbf{f} - {}^G\mathbf{p}_{I_k}) = \mathbf{H}_{c,k}{}^{I_k}_G\mathbf{C}({}^G\mathbf{f} - {}^G\mathbf{p}_{I_k}) \\ &= \mathbf{H}_{c,k}{}^{I_k}\mathbf{f} = \mathbf{0}\end{aligned} \tag{A.9}$$

where the last equality holds since the camera perspective-projection Jacobian matrix, $\mathbf{H}_{c,k}$, has as its right null space the feature position in the IMU frame (see (30) in [42]).

Lastly, this new unobservable direction is in addition to the four directions corresponding to global translation and yaw, i.e., $\mathbf{N}_s$ and $\mathbf{N}_1$ in (57) of [42] are independent, since the 4th block element of $\mathbf{N}_1$ is zero while that of $\mathbf{N}_s$ is not.

A.2 Proof of Theorem 2

In this section, we prove that the 3-dof global orientation in (3.8) is an unobservable direction of the VINS model, if and only if the platform does not rotate [see (3.7)]. Similarly to the proof presented in Appendix A.1, in this case, we need to show that $\mathbf{M}_k\mathbf{N}_o = \mathbf{0}$. From (A.1)

and (3.8), together with the definition of the matrices $\mathbf{\Gamma}_i, i = 1, \ldots, 4$, we obtain:

$$\begin{aligned}
\mathbf{M}_k\mathbf{N}_o &= \mathbf{\Gamma}_1(\mathbf{\Gamma}_4 \, {}^{I_0}_{G}\mathbf{C} - \frac{1}{2}\delta t_k^2 \, \mathbf{I}_3)\lfloor {}^{G}\mathbf{g} \rfloor && \text{(A.10)} \\
&= \mathbf{\Gamma}_1(\int_{t_0}^{t_k}\int_{t_0}^{s} {}^{G}_{I_\tau}\mathbf{C} \, d\tau ds \cdot {}^{I_0}_{G}\mathbf{C} - \frac{1}{2}\delta t_k^2 \, \mathbf{I}_3)\lfloor {}^{G}\mathbf{g} \rfloor && \text{(A.11)} \\
&= \mathbf{\Gamma}_1(\int_{t_0}^{t_k}\int_{t_0}^{s} {}^{G}_{I_0}\mathbf{C} \, d\tau ds \cdot {}^{I_0}_{G}\mathbf{C} - \frac{1}{2}\delta t_k^2 \, \mathbf{I}_3)\lfloor {}^{G}\mathbf{g} \rfloor && \text{(A.12)} \\
&= \mathbf{\Gamma}_1(\int_{t_0}^{t_k}\int_{t_0}^{s} 1 \, d\tau ds \cdot {}^{G}_{I_0}\mathbf{C}{}^{I_0}_{G}\mathbf{C} - \frac{1}{2}\delta t_k^2 \, \mathbf{I}_3)\lfloor {}^{G}\mathbf{g} \rfloor \\
&= \mathbf{\Gamma}_1(\frac{1}{2}\delta t_k^2 \, \mathbf{I}_3 - \frac{1}{2}\delta t_k^2 \, \mathbf{I}_3)\lfloor {}^{G}\mathbf{g} \rfloor = \mathbf{0} && \text{(A.13)}
\end{aligned}$$

where the equality from (A.11) to (A.12) holds if and only if the no rotation (i.e., constant orientation) assumption in (3.7) is satisfied.

Lastly, these new unobservable directions are in addition to the three directions corresponding to global translation, i.e., $\mathbf{N}_o$ and $\mathbf{N}_{t,1}$ in (57) of [42] are independent, since the first block element of $\mathbf{N}_{t,1}$ is zero while that of $\mathbf{N}_o$ is a (full-rank) rotational matrix.

A.3 Proof of Theorem 3

In this section, we prove that the scale in (3.6) is observable for the VINS model when an odometer is present. Specifically, the odometer provides measurements of the 2-dof planar component of the robot's linear velocity:

$$\mathbf{v}_k = \Lambda^{O_k}\mathbf{v}_{O_k} = \Lambda^{O}_{I}\mathbf{C}({}^{I_k}_{G}\mathbf{C}{}^{G}\mathbf{v}_{I_k} + \lfloor \boldsymbol{\omega}_m(t_k) - \mathbf{b}_g(t_k) \rfloor {}^{I}\mathbf{p}_O)$$

from which we obtain the following measurement Jacobians with respect to the states involved:

$$\begin{aligned}
\mathbf{H}^{O}_{\delta\theta} &= \Lambda^{O}_{I}\mathbf{C}\lfloor {}^{I_k}_{G}\mathbf{C}{}^{G}\mathbf{v}_{I_k} \rfloor \, , \;\; \mathbf{H}^{O}_{b_g} = \Lambda^{O}_{I}\mathbf{C}\lfloor {}^{I}\mathbf{p}_O \rfloor \\
\mathbf{H}^{O}_{v} &= \Lambda^{O}_{I}\mathbf{C}{}^{I_k}_{G}\mathbf{C} && \text{(A.14)}
\end{aligned}$$

The odometry measurements provide extra block rows in the observability matrix, in addition to the ones corresponding to the camera observations [see (A.1)]. From (A.14) and the analytical form of the state transition matrix, $\mathbf{\Phi}_{k,1}$ (see (44) in [42]), it can be verified that these extra

block rows have the following structure:

$$\begin{aligned}
\mathbf{M}_k^O &= \mathbf{H}_k^O \boldsymbol{\Phi}_{k,1} \\
&= \boldsymbol{\Gamma}_1^O \begin{bmatrix} \boldsymbol{\Gamma}_2^O & \boldsymbol{\Gamma}_3^O & {}^{I_k}_{G}\mathbf{C} & {}^{I_k}_{G}\mathbf{C}\boldsymbol{\Phi}_{k,1}^{(3,4)} & \mathbf{0}_{3\times 3} & \mathbf{0}_{3\times 3} \end{bmatrix} \\
\text{with } \boldsymbol{\Gamma}_1^O &= \Lambda_I^O \mathbf{C} \\
\boldsymbol{\Gamma}_2^O &= \lfloor {}^{I_k}_{G}\mathbf{C}\,{}^{G}\mathbf{v}_{I_k} \rfloor \boldsymbol{\Phi}_{k,1}^{(1,1)} + {}^{I_k}_{G}\mathbf{C}\boldsymbol{\Phi}_{k,1}^{(3,1)} \\
\boldsymbol{\Gamma}_3^O &= \lfloor {}^{I_k}_{G}\mathbf{C}\,{}^{G}\mathbf{v}_{I_k} \rfloor \boldsymbol{\Phi}_{k,1}^{(1,2)} + \lfloor {}^{I}\mathbf{p}_O \rfloor + {}^{I_k}_{G}\mathbf{C}\boldsymbol{\Phi}_{k,1}^{(3,2)}
\end{aligned} \tag{A.15}$$

for any time $t_k \geq t_0$, with $\boldsymbol{\Phi}_{k,1}^{(i,j)}$ denoting the (i,j)-th block element of the state transition matrix $\boldsymbol{\Phi}_{k,1}$. From Thm. 1, the scale becomes unobservable if and only if the acceleration is constant. Therefore, it suffices to show that $\mathbf{M}_k^O \mathbf{N}_s \neq \mathbf{0}$ when (3.5) is satisfied. Specifically:

$$\mathbf{M}_k^O \mathbf{N}_s = \boldsymbol{\Gamma}_1^O ({}^{I_k}_{G}\mathbf{C}\,{}^{G}\mathbf{v}_{I_0} - {}^{I_k}_{G}\mathbf{C}\boldsymbol{\Phi}_{k,1}^{(3,4)}\,{}^{I}\mathbf{a}) \tag{A.16}$$

$$= \Lambda_I^O \mathbf{C}_G^{I_k} \mathbf{C} ({}^{G}\mathbf{v}_{I_0} + \int_{t_0}^{t_k} {}^{G}_{I_\tau}\mathbf{C}\, d\tau \cdot {}^{I}\mathbf{a}) \tag{A.17}$$

$$= \Lambda_I^O \mathbf{C}_G^{I_k} \mathbf{C}\,{}^{G}\mathbf{v}_{I_k} = \Lambda^{O_k} \mathbf{v}_{I_k} \tag{A.18}$$

where we have followed the same reasoning as in (A.3)-(A.7). The quantity in (A.18) is non-zero, if the velocity of the IMU frame, expressed in the odometer frame, does not vanish along the $x-y$ directions, i.e., if the platform has translational motion along the horizontal plane. Under this condition, which is satisfied in practice as long as the vehicle does not stay static forever, the odometer measurements make the scale observable.

A.4 The Unobservable Direction of Scale

In this section, we show that the unobservable direction in (3.6) corresponds to the scale. Assume that there exists a VINS state vector $\mathbf{x}$ and the corresponding measurements from the IMU [see (3.2)] and the camera [see (3.4)]. Consider the case where both the entire trajectory of the platform and the scene are "scaled up" by a factor of α, or equivalently, the global coordinate system $\{G\}$ is "shrunken down" by the factor α. This corresponds to a change of the original state $\mathbf{x}$ into a new state $\mathbf{x}'$. Specifically, as for the IMU's position, $\mathbf{p}_I$, and the feature's position,

$\mathbf{f}_j$, with respect to $\{G\}$, the scale change can be written as:[1]

$$^G\mathbf{p}'_I = \alpha\, ^G\mathbf{p}_I \tag{A.19}$$

$$^G\mathbf{f}'_j = \alpha\, ^G\mathbf{f}_j, \quad j = 1, \ldots, N \tag{A.20}$$

where $^G\mathbf{p}'_I$ and $^G\mathbf{f}'_j$ are the new positions after the scaling. By taking the first and second-order time derivative on both sides of (A.19), we obtain the scaled velocity and body acceleration of the IMU as:

$$^G\mathbf{v}'_I = \alpha\, ^G\mathbf{v}_I \tag{A.21}$$

$$^G\mathbf{a}'_I = \alpha\, ^G\mathbf{a}_I \tag{A.22}$$

Note that, on the other hand, the scale change does not affect the IMU's orientation with respect to the global frame (as the "scale" referred here is with respect to translation only), i.e.:

$$^I_G\mathbf{C}' = {}^I_G\mathbf{C} \tag{A.23}$$

and hence the rotational velocity remains the same as well:

$$^I\boldsymbol{\omega}' = {}^I\boldsymbol{\omega} \tag{A.24}$$

Moreover, to ensure that this scale change is unobservable, the measurements from the IMU and the camera need to be unchanged. As for the camera observations, for each feature j,

[1] Note that the presented analysis holds true for any time $t \geq t_0$. Hence we omit the time index for the clarity of presentation.

from (3.4) (A.19) (A.20) (A.23) we have:

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \triangleq {}^{I}\mathbf{f}'_j = {}^{I}_{G}\mathbf{C}'({}^{G}\mathbf{f}'_j - {}^{G}\mathbf{p}'_I) = {}^{I}_{G}\mathbf{C}(\alpha^{G}\mathbf{f}_j - \alpha^{G}\mathbf{p}_I) = \alpha^{I}_{G}\mathbf{C}({}^{G}\mathbf{f}_j - {}^{G}\mathbf{p}_I)$$

$$= \alpha^{I}\mathbf{f}_j = \alpha \begin{bmatrix} x \\ y \\ z \end{bmatrix} \tag{A.25}$$

$$\Rightarrow \quad \mathbf{z}'_j = \frac{1}{z'} \begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{1}{\alpha z} \begin{bmatrix} \alpha x \\ \alpha y \end{bmatrix} = \frac{1}{z} \begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{z}_j \tag{A.26}$$

Hence, after scaling, the camera measurements do not change due to the perspective projection model. This result is to be expected, since a camera's observation is scale invariant, therefore it's insensitive to any scale change. As for the IMU measurements, we first examine the rotational velocity measured by the gyroscope. Since the gyroscope measurements [see (3.2)] need to stay the same before and after the scaling, i.e.:

$$\boldsymbol{\omega}_m = {}^{I}\boldsymbol{\omega} + \mathbf{b}_g = {}^{I}\boldsymbol{\omega}' + \mathbf{b}'_g \tag{A.27}$$

by substituting (A.24), we obtain:

$$\mathbf{b}'_g = \mathbf{b}_g \tag{A.28}$$

Similarly, for the linear acceleration measurements from the accelerometer, from (3.2) and (A.22) (A.23), we obtain:

$$\mathbf{a}_m = {}^{I}_{G}\mathbf{C}({}^{G}\mathbf{a}_I - {}^{G}\mathbf{g}) + \mathbf{b}_a = {}^{I}_{G}\mathbf{C}'({}^{G}\mathbf{a}'_I - {}^{G}\mathbf{g}) + \mathbf{b}'_a = {}^{I}_{G}\mathbf{C}(\alpha^{G}\mathbf{a}_I - {}^{G}\mathbf{g}) + \mathbf{b}'_a \tag{A.29}$$

$$\Rightarrow \quad \mathbf{b}'_a = \mathbf{b}_a - (\alpha - 1)^{I}_{G}\mathbf{C}^{G}\mathbf{a}_I = \mathbf{b}_a - (\alpha - 1)^{I}\mathbf{a} \tag{A.30}$$

Collecting the equations (A.23) (A.28) (A.21) (A.30) (A.19) (A.20), we put together the

VINS state element changes due to the scaling, by a factor of α, as:

$$\begin{aligned}
{}^I_G\mathbf{C}' &= {}^I_G\mathbf{C} \\
\mathbf{b}'_g &= \mathbf{b}_g \\
{}^G\mathbf{v}'_I &= \alpha\,{}^G\mathbf{v}_I = {}^G\mathbf{v}_I + (\alpha-1)\,{}^G\mathbf{v}_I \\
\mathbf{b}'_a &= \mathbf{b}_a - (\alpha-1)\,{}^I\mathbf{a} \\
{}^G\mathbf{p}'_I &= \alpha\,{}^G\mathbf{p}_I = {}^G\mathbf{p}_I + (\alpha-1)\,{}^G\mathbf{p}_I \\
{}^G\mathbf{f}'_j &= \alpha\,{}^G\mathbf{f}_j = {}^G\mathbf{f}_j + (\alpha-1)\,{}^G\mathbf{f}_j, \quad j=1,\ldots,N
\end{aligned} \tag{A.31}$$

If we define the original and the new error state as $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$, corresponding to the original and the new VINS state $\mathbf{x}$ and $\mathbf{x}'$ (see [42] for the definition of the VINS error state), respectively, then (A.31) can be rewritten in the error-state form as:

$$\begin{bmatrix}
{}^I\delta\boldsymbol{\theta}'_G \\ \tilde{\mathbf{b}}'_g \\ {}^G\tilde{\mathbf{v}}'_I \\ \tilde{\mathbf{b}}'_a \\ {}^G\tilde{\mathbf{p}}'_I \\ {}^G\tilde{\mathbf{f}}'_1 \\ \vdots \\ {}^G\tilde{\mathbf{f}}'_N
\end{bmatrix} =
\begin{bmatrix}
{}^I\delta\boldsymbol{\theta}_G \\ \tilde{\mathbf{b}}_g \\ {}^G\tilde{\mathbf{v}}_I + (\alpha-1)\,{}^G\mathbf{v}_I \\ \tilde{\mathbf{b}}_a - (\alpha-1)\,{}^I\mathbf{a} \\ {}^G\tilde{\mathbf{p}}_I + (\alpha-1)\,{}^G\mathbf{p}_I \\ {}^G\tilde{\mathbf{f}}_1 + (\alpha-1)\,{}^G\mathbf{f}_1 \\ \vdots \\ {}^G\tilde{\mathbf{f}}_N + (\alpha-1)\,{}^G\mathbf{f}_N
\end{bmatrix} =
\begin{bmatrix}
{}^I\delta\boldsymbol{\theta}_G \\ \tilde{\mathbf{b}}_g \\ {}^G\tilde{\mathbf{v}}_I \\ \tilde{\mathbf{b}}_a \\ {}^G\tilde{\mathbf{p}}_I \\ {}^G\tilde{\mathbf{f}}_1 \\ \vdots \\ {}^G\tilde{\mathbf{f}}_N
\end{bmatrix} + (\alpha-1)
\begin{bmatrix}
\mathbf{0}_{3\times1} \\ \mathbf{0}_{3\times1} \\ {}^G\mathbf{v}_I \\ -{}^I\mathbf{a} \\ {}^G\mathbf{p}_I \\ {}^G\mathbf{f}_1 \\ \vdots \\ {}^G\mathbf{f}_N
\end{bmatrix} \tag{A.32}$$

where we see that the right-most vector is exactly the same as in (3.6), hence

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{x}} + (\alpha-1)\mathbf{N}_s \tag{A.33}$$

To summarize, if the entire trajectory of the platform and the scene are scaled by a factor of α (as the starting point of this analysis), then the VINS error state (and hence the state) will be changed along the direction of $\mathbf{N}_s$ by a factor of $\alpha-1$ [see (A.33)], *without* changing the measurements from the camera [see (A.26)] or the IMU [see (A.27) and (A.29)]. Moreover, it is obvious that the reverse statement holds true as well. Therefore, we conclude that the direction $\mathbf{N}_s$ in (3.6) is unobservable, and it corresponds to the scale change in terms of its physical

meaning.

A.5 The Unobservable Directions of Orientation

In this section, we show that the unobservable directions in (3.8) correspond to the 3-dof global orientation. Assume that there exists a VINS state vector $\mathbf{x}$ and the corresponding measurements from the IMU [see (3.2)] and the camera [see (3.4)]. Consider the case where the global frame $\{G\}$ is rotated by a *small* angle $\delta\boldsymbol{\phi}$ into a new global frame $\{G'\}$, where $\delta\boldsymbol{\phi}$ is a 3×1 vector whose direction and magnitude represent the axis and angle of the rotation, respectively. Hence,

$$ {}^{G}_{G'}\mathbf{C} = \mathbf{C}(\delta\boldsymbol{\phi}) \simeq \mathbf{I}_3 - \lfloor \delta\boldsymbol{\phi} \rfloor \tag{A.34} $$

by the small-angle approximation of the rotational matrix based on the assumption that the amount of rotation is small. Due to this change of the global frame (from $\{G\}$ to $\{G'\}$), the original VINS state, $\mathbf{x}$, which is expressed with respect to $\{G\}$, is now changed to a new state, $\mathbf{x}'$, which is expressed with respect to $\{G'\}$. Specifically, as for the orientation of the IMU:[2]

$$ {}^{I}_{G'}\mathbf{C} = {}^{I}_{G}\mathbf{C}\,{}^{G}_{G'}\mathbf{C} = {}^{I}_{G}\mathbf{C}(\mathbf{I}_3 - \lfloor \delta\boldsymbol{\phi} \rfloor) = (\mathbf{I}_3 - \lfloor {}^{I}_{G}\mathbf{C}\,\delta\boldsymbol{\phi} \rfloor){}^{I}_{G}\mathbf{C} \tag{A.35} $$

Since the transformation involves only rotation, the new position state of the IMU can be obtained as:

$$ \begin{aligned} {}^{G'}\mathbf{p}_I &= {}^{G'}_{G}\mathbf{C}\,{}^{G}\mathbf{p}_I = {}^{G}_{G'}\mathbf{C}^T{}^{G}\mathbf{p}_I = (\mathbf{I}_3 - \lfloor \delta\boldsymbol{\phi} \rfloor)^T{}^{G}\mathbf{p}_I = (\mathbf{I}_3 + \lfloor \delta\boldsymbol{\phi} \rfloor){}^{G}\mathbf{p}_I \\ &= {}^{G}\mathbf{p}_I + \lfloor \delta\boldsymbol{\phi} \rfloor {}^{G}\mathbf{p}_I = {}^{G}\mathbf{p}_I - \lfloor {}^{G}\mathbf{p}_I \rfloor \delta\boldsymbol{\phi} \end{aligned} \tag{A.36} $$

and similarly for the feature's position:

$$ {}^{G'}\mathbf{f}_j = {}^{G'}_{G}\mathbf{C}\,{}^{G}\mathbf{f}_j = {}^{G}\mathbf{f}_j - \lfloor {}^{G}\mathbf{f}_j \rfloor \delta\boldsymbol{\phi}, \quad j = 1, \ldots, N \tag{A.37} $$

By taking the first-order time derivative on both sides of (A.36), we obtain the new velocity state of the IMU:

$$ {}^{G'}\mathbf{v}_I = {}^{G}\mathbf{v}_I - \lfloor {}^{G}\mathbf{v}_I \rfloor \delta\boldsymbol{\phi} \tag{A.38} $$

[2] Note that the presented analysis holds true for any time $t \geq t_0$. Hence we omit the time index for the clarity of presentation.

Note that, on the other hand, the transformation of the *global* frame does not affect the trajectory, and hence the motion, of the IMU when expressed in the IMU's *local* frame of reference $\{I\}$. Therefore, the *local* rotational velocity and linear acceleration of the IMU are the same before and after the transformation of the global frame, i.e.:

$$^{I}\boldsymbol{\omega}' = {}^{I}\boldsymbol{\omega} \tag{A.39}$$

$$^{I}\mathbf{a}' = {}^{I}\mathbf{a} \tag{A.40}$$

Moreover, to ensure that the change of the global frame's orientation is unobservable, the measurements from the IMU and the camera need to be unchanged. As for the camera observations, for each feature j, from (3.4) (A.35) (A.36) (A.37) we have:

$$\begin{aligned}\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \triangleq {}^{I}\mathbf{f}'_j &= {}^{I}_{G'}\mathbf{C}({}^{G'}\mathbf{f}_j - {}^{G'}\mathbf{p}_I) = {}^{I}_{G}\mathbf{C}\,{}^{G}_{G'}\mathbf{C}({}^{G'}_{G}\mathbf{C}\,{}^{G}\mathbf{f}_j - {}^{G'}_{G}\mathbf{C}\,{}^{G}\mathbf{p}_I) \\ &= {}^{I}_{G}\mathbf{C}({}^{G}\mathbf{f}_j - {}^{G}\mathbf{p}_I) = {}^{I}\mathbf{f}_j = \begin{bmatrix} x \\ y \\ z \end{bmatrix}\end{aligned} \tag{A.41}$$

$$\Rightarrow \quad \mathbf{z}'_j = \frac{1}{z'}\begin{bmatrix} x' \\ y' \end{bmatrix} = \frac{1}{z}\begin{bmatrix} x \\ y \end{bmatrix} = \mathbf{z}_j \tag{A.42}$$

Hence, after the transformation of the global frame, the camera measurements do not change. This result is to be expected, since a camera's observation depends only on the relative position of the feature with respect to the camera's frame, therefore it's insensitive to any change in the global frame itself. As for the IMU measurements, we first examine the rotational velocity measured by the gyroscope. Since the gyroscope measurements [see (3.2)] need to stay the same before and after the transformation of the global frame, i.e.:

$$\boldsymbol{\omega}_m = {}^{I}\boldsymbol{\omega} + \mathbf{b}_g = {}^{I}\boldsymbol{\omega}' + \mathbf{b}'_g \tag{A.43}$$

by substituting (A.39), we obtain:

$$\mathbf{b}'_g = \mathbf{b}_g \tag{A.44}$$

Similarly, for the linear acceleration measurements from the accelerometer, from (3.2) and the definition that ${}^{I}\mathbf{a} = {}^{I}_{G}\mathbf{C}\,{}^{G}\mathbf{a}_I$, we obtain:

$$\mathbf{a}_m = {}^{I}_{G}\mathbf{C}({}^{G}\mathbf{a}_I - {}^{G}\mathbf{g}) + \mathbf{b}_a = {}^{I}\mathbf{a} - {}^{I}_{G}\mathbf{C}\,{}^{G}\mathbf{g} + \mathbf{b}_a = {}^{I}\mathbf{a}' - {}^{I}_{G'}\mathbf{C}\,{}^{G'}\mathbf{g}' + \mathbf{b}'_a \tag{A.45}$$

Substituting (A.40) yields:

$$\mathbf{b}'_a = \mathbf{b}_a + {}^{I}_{G'}\mathbf{C}\,{}^{G'}\mathbf{g}' - {}^{I}_{G}\mathbf{C}\,{}^{G}\mathbf{g} \tag{A.46}$$

Note that according to the definition, the gravity vector, $\mathbf{g}$, is a known *constant* in the corresponding global frame, i.e., $\mathbf{g}$ is *fixed* with respect to the global frame. Hence, as the global frame rotates from $\{G\}$ to $\{G'\}$, the gravity vector rotates simultaneously from $\mathbf{g}$ to $\mathbf{g}'$, such that:

$${}^{G'}\mathbf{g}' = {}^{G}\mathbf{g} \tag{A.47}$$

Substituting (A.47) and (A.35) into (A.46), we obtain:

$$\begin{aligned}\mathbf{b}'_a &= \mathbf{b}_a + {}^{I}_{G'}\mathbf{C}\,{}^{G'}\mathbf{g}' - {}^{I}_{G}\mathbf{C}\,{}^{G}\mathbf{g} = \mathbf{b}_a + {}^{I}_{G'}\mathbf{C}\,{}^{G}\mathbf{g} - {}^{I}_{G}\mathbf{C}\,{}^{G}\mathbf{g} = \mathbf{b}_a + ({}^{I}_{G'}\mathbf{C} - {}^{I}_{G}\mathbf{C})^{G}\mathbf{g} \\ &= \mathbf{b}_a + ({}^{I}_{G}\mathbf{C} - {}^{I}_{G}\mathbf{C}\lfloor\delta\boldsymbol{\phi}\rfloor - {}^{I}_{G}\mathbf{C})^{G}\mathbf{g} = \mathbf{b}_a - {}^{I}_{G}\mathbf{C}\lfloor\delta\boldsymbol{\phi}\rfloor{}^{G}\mathbf{g} = \mathbf{b}_a + {}^{I}_{G}\mathbf{C}\lfloor{}^{G}\mathbf{g}\rfloor\delta\boldsymbol{\phi}\end{aligned} \tag{A.48}$$

Collecting the equations (A.35) (A.44) (A.38) (A.48) (A.36) (A.37), we put together the VINS state element changes due to the rotation of the global frame, by a small angle $\delta\boldsymbol{\phi}$, as:

$$\begin{aligned}{}^{I}_{G'}\mathbf{C} &= (\mathbf{I}_3 - \lfloor{}^{I}_{G}\mathbf{C}\,\delta\boldsymbol{\phi}\rfloor){}^{I}_{G}\mathbf{C} \\ \mathbf{b}'_g &= \mathbf{b}_g \\ {}^{G'}\mathbf{v}_I &= {}^{G}\mathbf{v}_I - \lfloor{}^{G}\mathbf{v}_I\rfloor\delta\boldsymbol{\phi} \\ \mathbf{b}'_a &= \mathbf{b}_a + {}^{I}_{G}\mathbf{C}\lfloor{}^{G}\mathbf{g}\rfloor\delta\boldsymbol{\phi} \\ {}^{G'}\mathbf{p}_I &= {}^{G}\mathbf{p}_I - \lfloor{}^{G}\mathbf{p}_I\rfloor\delta\boldsymbol{\phi} \\ {}^{G'}\mathbf{f}_j &= {}^{G}\mathbf{f}_j - \lfloor{}^{G}\mathbf{f}_j\rfloor\delta\boldsymbol{\phi}, \quad j = 1, \ldots, N\end{aligned} \tag{A.49}$$

If we define the original and the new error state as $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{x}}'$, corresponding to the original and the new VINS state $\mathbf{x}$ and $\mathbf{x}'$ (see [42] for the definition of the VINS error state), respectively,

then (A.49) can be rewritten in the error-state form as:

$$\begin{bmatrix} {}^{I}\delta\boldsymbol{\theta}_{G'} \\ \tilde{\mathbf{b}}'_g \\ {}^{G'}\tilde{\mathbf{v}}_I \\ \tilde{\mathbf{b}}'_a \\ {}^{G'}\tilde{\mathbf{p}}_I \\ {}^{G'}\tilde{\mathbf{f}}_1 \\ \vdots \\ {}^{G'}\tilde{\mathbf{f}}_N \end{bmatrix} = \begin{bmatrix} {}^{I}\delta\boldsymbol{\theta}_{G} + {}^{I}_{G}\mathbf{C}\,\delta\boldsymbol{\phi} \\ \tilde{\mathbf{b}}_g \\ {}^{G}\tilde{\mathbf{v}}_I - \lfloor {}^{G}\mathbf{v}_I \rfloor \delta\boldsymbol{\phi} \\ \tilde{\mathbf{b}}_a + {}^{I}_{G}\mathbf{C}\lfloor {}^{G}\mathbf{g} \rfloor \delta\boldsymbol{\phi} \\ {}^{G}\tilde{\mathbf{p}}_I - \lfloor {}^{G}\mathbf{p}_I \rfloor \delta\boldsymbol{\phi} \\ {}^{G}\tilde{\mathbf{f}}_1 - \lfloor {}^{G}\mathbf{f}_1 \rfloor \delta\boldsymbol{\phi} \\ \vdots \\ {}^{G}\tilde{\mathbf{f}}_N - \lfloor {}^{G}\mathbf{f}_N \rfloor \delta\boldsymbol{\phi} \end{bmatrix} = \begin{bmatrix} {}^{I}\delta\boldsymbol{\theta}_{G} \\ \tilde{\mathbf{b}}_g \\ {}^{G}\tilde{\mathbf{v}}_I \\ \tilde{\mathbf{b}}_a \\ {}^{G}\tilde{\mathbf{p}}_I \\ {}^{G}\tilde{\mathbf{f}}_1 \\ \vdots \\ {}^{G}\tilde{\mathbf{f}}_N \end{bmatrix} + \begin{bmatrix} {}^{I}_{G}\mathbf{C} \\ \mathbf{0}_{3\times 3} \\ -\lfloor {}^{G}\mathbf{v}_I \rfloor \\ {}^{I}_{G}\mathbf{C}\lfloor {}^{G}\mathbf{g} \rfloor \\ -\lfloor {}^{G}\mathbf{p}_I \rfloor \\ -\lfloor {}^{G}\mathbf{f}_1 \rfloor \\ \vdots \\ -\lfloor {}^{G}\mathbf{f}_N \rfloor \end{bmatrix} \delta\boldsymbol{\phi} \tag{A.50}$$

where we see that the matrix multiplied with $\delta\boldsymbol{\phi}$ on the right-hand side is exactly the same as in (3.8), hence

$$\tilde{\mathbf{x}}' = \tilde{\mathbf{x}} + \mathbf{N}_o \delta\boldsymbol{\phi} \tag{A.51}$$

To summarize, if the global frame is rotated by a small angle $\delta\boldsymbol{\phi}$ (as the starting point of this analysis), or equivalently, the entire trajectory of the platform and the scene are rotated by $-\delta\boldsymbol{\phi}$ with respect to the global frame, then the VINS error state (and hence the state) will be changed along the direction as a linear combination of the columns of $\mathbf{N}_o$ [see (A.51)], *without* changing the measurements from the camera [see (A.42)] or the IMU [see (A.43) and (A.45)]. Moreover, it is obvious that the reverse statement holds true as well. Therefore, we conclude that the directions $\mathbf{N}_o$ in (3.8) are unobservable, and they correspond to the change of the 3-dof global orientation in terms of the physical meaning. In particular, if $\delta\boldsymbol{\phi} = ||\delta\boldsymbol{\phi}||\mathbf{e}_1$, it would correspond to a rotation about the global frame's x-axis, i.e., a change in the roll angle. Similarly for the pitch and yaw angles. Hence, the three columns of $\mathbf{N}_o$ correspond to the roll, pitch, and yaw angle change, respectively, of the orientation of the IMU's frame with respect to the global frame.