# EFFICIENT, CONSISTENT, AND PERSISTENT VISUAL-INERTIAL NAVIGATION

by

Patrick F. Geneva

A dissertation submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science

Spring 2025

**EFFICIENT, CONSISTENT, AND PERSISTENT**

**VISUAL-INERTIAL NAVIGATION**

by

Patrick F. Geneva

Approved: _____

Weisong Shi, Ph.D.
Chair of the Department of Computer Science

Approved: _____

Jamie D. Phillips, Ph.D.
Interim Dean of the College of Engineering

Approved: _____

Louis F. Rossi, Ph.D.
Vice Provost for Graduate and Professional Education and
Dean of the Graduate College

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Guoquan (Paul) Huang, Ph.D.
Professor in charge of dissertation

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Herbert Tanner, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Xi Peng, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Christopher Rasmussen, Ph.D.
Member of dissertation committee

I certify that I have read this dissertation and that in my opinion it meets the academic and professional standard required by the University as a dissertation for the degree of Doctor of Philosophy.

Signed: _____

Chandra Kambhamettu, Ph.D.
Member of dissertation committee

## ACKNOWLEDGEMENTS

*To my family and friends, who have always supported me.*

None of my work presented in this thesis would be possible without the years of collaboration, reading group weekend discussions, and people who have supported my journey. I first wish to extend my eternal gratitude to my advisor Dr. Guoquan (Paul) Huang who has continued to nurture my skills as a researcher from my early undergraduate days almost 10 years ago. He has continued to be both an outstanding example researcher who I have continued to strive to become, but also a close collaborator and champion of my research to whom I owe my success to. I would like to especially thank the early lab members Kevin Eckenhoff and Yulin Yang who I have published so many works and have learned so much about both research and life, I am ever grateful and humbled by your skills. I would also like to thank my lab mates and collaborators Woosik Lee, Chuchu Chen, Xingxing Zuo, Pengxiang Zhu, Nathaniel Merrill, James Maley, Jesse Bloecker, Zheng Huai, Wenxuan (Owen) Li, Yuxiang Peng, Saimouli Katragadda, and Chinmay Burgul which I have enjoyed every interaction with and hope for your success in exploring the vast forest of knowledge. Many thanks to my thesis committee members Dr. Huang, Dr. Tanner, Dr. Peng, Dr. Rasmussen, and Dr. Kambhamettu for their flexibility, patience, and insight. I am also ever thankful to my family, friends, and fiance who have continued to support me throughout my degree. Thank you all.

## TABLE OF CONTENTS

## LIST OF TABLES

## LIST OF FIGURES

## ABSTRACT

The use of visual-inertial navigation systems (VINS) has become ubiquitous due to their ability to provide high quality 3D motion tracking and has continued to be at the center of simultaneous localization and mapping (SLAM) research. Deployment platforms continue to reduce in cost and miniaturize to further enable mass production to consumers (e.g., smartphones, virtual and augmented-reality headsets, and micro-aerial vehicles (MAV)). A key barrier that prevents the wider deployment of VINS is the accuracy and computational demands for long-term persistent state estimation (e.g., hours of continuous operation in a common global frame). Development of computationally efficient VINS which can efficiently incorporate loop-closure information to reduce estimator drift and increase accuracy over long-term estimation periods with persistent maps remains a crucial challenge, which this thesis looks to address.

We first introduce a state-of-the-art open-sourced filter-based VINS research framework, termed OpenVINS, which leverages cutting edge extended Kalman filter (EKF) estimator techniques and demonstrates accurate and consistent state estimation where both the mean and uncertainty of the state are recovered at each timestep. We then focus on how to improve this visual-inertial odometry (VIO) to include further loop-closure information by tracking large environmental plane geometric primitives in an efficient manner leveraging a novel minimal plane representation termed the Closest Point (CP) plane. We show that the inclusion of such CP planes, which can be tracked for significant periods due to their large spatial nature and the proposed novel tracking algorithm, reduces the long-term drift in both simulation and real-world experiments. We then focus on the visual-inertial simultaneous localization and mapping (VI-SLAM) task and how we can perform *consistent* long-term persistent localization without causing computational complexity to explode over time. We show that

the Schmidt-Kalman filter (SKF) methodology can be leveraged in conjunction with two different measurement models, including a novel 2D-to-2D method for indirect loop-closure to historical poses, to bound long-term drift which only increases complexity linearly in terms of size of the historical map. We then show that the proposed Schmidt-EKF for VI-SLAM (SEVIS) can be coupled with a secondary optimization thread, which enables relinearization, to perform large-scale estimation. We finally apply the learned loop-closure and measurement constraint techniques to the distributed multi-robot cooperative localization (CL) case. We show that covariance intersection (CI) can be efficiently leveraged for distributed VI-SLAM and we can limit long-term drift while also not requiring robots to simultaneously visit locations for cross-robot constraints. This novel distributed CL estimator shows state-of-the-art accurate, consistent, and efficient performance both in simulation and real-world experiments.

# Chapter 1

# INTRODUCTION

## 1.1 Introduction

The use of visual-inertial navigation systems (VINS) has become ubiquitous due to their ability to provide high quality 3D motion tracking and has continued to be at the center of simultaneous localization and mapping (SLAM) research [80]. Deployment platforms continue to reduce in cost and miniaturize to further enable mass production to consumers (e.g., smartphones, virtual and augmented-reality headsets, and micro-aerial vehicles (MAV)). A key barrier which prevents the wider deployment of VINS is the accuracy and computational demands for long-term persistent state estimation (e.g., hours of continuous operation in a common global frame). Many existing systems are unable to provide accurate drift-free state estimates for long periods or have significant growth in the computational cost due to the inclusion of loop-closure information and thus are limited by the memory and computational resources available. This is further compounded when single-user systems are expanded to multi-user localization which can naively cause exponential growth in complexity. Development of computationally efficient VINS which can efficiently incorporate loop-closure information to reduce estimator drift and increase accuracy over long-term estimation periods remains a crucial challenge, which this thesis looks to address.

One method, which is introduced in this thesis, to reduce long-term drift is the use of high-level geometric primitives, e.g. lines and planes, which have large spatial presences and can enable significant periods of re-detection and loop-closures. Many state-of-the-art VINS ignore high-level structural regularities such as planes even though they are common in man-made environments and can be exploited to further

constrain the estimated trajectory. This is typically due to requiring a stereo camera setup, an additional depth sensor, or a complex neural network which adds custom hardware and complexity. We introduce a new plane representation, termed the Closest Point (CP) plane, and demonstrate its use in a LiDAR-inertial planar SLAM system. We show that collections of 3D points can be compressed into a single CP plane measurement and uncertainty which can be efficiently fused and thus reduce the number of residuals used during optimization. We then look to the monocular-inertial case where we cannot actively extract these planes but wish the leverage point-on-plane constraints to provide loop-closures to planes which can be seen for significantly longer periods of time as compared to traditional point features. This is a particularly challenging open problem as without depth information we cannot directly recover surface normals which have been typically used to find planes when RGB-D [67, 75, 104, 209], 1D LRF [77, 175], or 3D LiDAR [106, 228, 230] are available. In contrast, we limit ourselves to a *single* monocular camera and IMU and present a novel method to track, initialize, estimate, and merge these environmental planes within an efficient filter-based VINS. We show in simulation and on real-world datasets that tracking planes for significant periods of time has large trajectory improvements.

We then look at how to consistently incorporate long-term loop-closure information without impacting the efficiency of our state estimator. Typically visual place recognition is first leveraged to recover correspondence information to historical states which can then be incorporated into VINS by constraining historical estimates states through different measurement models such as 2D-to-3D [34, 126, 127, 141, 146]. The major downside is that extra historical states are detrimental to estimator complexity, which is typically $O(n^3)$ in size of the state, since as more historical states are kept during environmental exploration, the estimator becomes more costly to run. To overcome this, light-weight visual-inertial odometry (VIO) [7, 13, 60, 71, 72, 78, 109, 112, 139, 153, 199, 200, 220] typically only estimates a small temporal window of states and is a small sub-system of the complete localization solution which has traditionally operated independently of the long-term visual-inertial simultaneous localization and

mapping (VI-SLAM) module. This VIO is relegated to only provide high frequency relative pose information to secondary *non-realtime* processing methods which can fuse loop-closure information and whose complexity is typically left to grow unbounded or some keyframing scheme is leveraged to bound complexity while slowly degrading constraint quality. This architecture has significant challenges when trying to deploy to resource constrained mobile platforms and additionally does not allow for the VIO to leverage historical information to constrain its state estimates since it is naturally decoupled in nature. Clearly, there is a need for a computationally efficient method which can incorporate historical information while retaining sufficient levels of accuracy and consistency.

A key focus throughout this work is to ensure we have a consistent estimator which estimates both an accurate *mean* and *covariance* both of which are crucial for practical use for downstream applications (e.g., control and planning). Additionally, we argue from a more fundamental level that the "consistency is necessary for filter optimality" [4, Section 5.4.1] and thus consistency is key to unlocking accurate estimation. This is of particular importance to VINS since it has been shown that through the linearization process, VINS estimators become *inconsistent* due to spurious information gain along the 4 degree-of-freedom (DoF) global yaw and position unobservable directions [71], and trajectory performance is directly tied to the consistency of the estimator. This has ultimately led to significant efforts within the class of visual-inertial estimators to perform "observability aware" state estimation and correct for these inaccurate information gains causing filter over-confidence [72, 84, 112]. There are many key works such as robocentric VINS [7, 78], invariant filters [12, 199, 206, 220], observability-constrained [72], and first-estimates Jacobian [22, 83, 84, 112] which have focused on this. Within the context of long-term VI-SLAM, there have been very limited works such as those by Mourikis et al. [140] which ensure consistent state estimation through pausing the VIO and performing a full VI-SLAM optimization to recompute the current state and covariance and Sartipi et al. [174] which tries to capture the untracked correlations through measurement inflation. In this thesis, we

show that it *is* possible to have both a consistent and efficient long-term VI-SLAM system. The proposed method leverages the efficiency and guaranteed consistency of the Schmidt-Kalman filter (SKF) [177], which can enable linear complexity in terms of the size of the map, to perform VI-SLAM with both traditional 2D-to-3D and a novel 2D-to-2D loop-closure constraint. Within the context of prior map localization, these two constraints are compared against naive inflation methods and their trade-offs are discussed in depth. This culminates in a novel hybrid estimator which couples the lightweight and consistent SKF-based VINS frontend with a secondary mapping thread which recovers an accuracy and *consistent* map with uncertainty via C-KLAM [147] which can then be leveraged.

Having addressed significant challenges within the single robot case, we then looked to see how these efficient and consistent long-term estimator methodologies can be applied to the challenging multi-robot case which is typically plagued by the same issues of complexity due to state size. Accurate and efficient cooperative localization (CL) that enables multi-user augmented reality (AR) experiences, multi-device cooperative mapping, and multi-vehicle formation control, is a key barrier to overcome due to the challenges of communication, distributed computation, and complex multi-robot asynchronous measurement constraints. The last part of this thesis builds upon the cooperative localization work by Zhu et al. [226] and proposes a fully distributed multi-robot visual-inertial covariance intersection (CI)-based estimator by delicately exploiting information contained in both environmental SLAM landmarks and loop-closures across robots and time. Specifically, a novel first-of-its-kind consistent and efficient CI-based distributed estimator is designed to include both SLAM features and incorporate loop-closure constraints to historical states of other robots leveraging our knowledge developed for the single-robot case in the previous thrust. As a result, the proposed distributed CL estimator does *not* require the *simultaneous* viewing of the same location due to leveraging of historical common features (e.g., a robot can gain information if another robot had previously explored the same location), while significantly improving the localization performance thanks to such common multi-robot

measurement information. We contrast the proposed method with the work by Sartipi et al. [174] which introduced a distributed method for multi-user AR experiences through the use of multi-map feature constraints. Within their estimator, common features between the current robot and another are found by performing a 2D-to-3D match to the other robot's environmental map, as compared to directly leveraging the other robot's feature measurements. They address the inconsistency of unknown cross-robot correlations by treating the feature in the other robot's map as true through measurement noise inflation and adding noise to the cross-robot map global frame map transform. As compared to this, the proposed distributed CL leverages CI that theoretically guarantees consistency in the case of unknown correlations and directly leverages the other user's common feature measurements avoiding limiting matching to only features with 3D position estimates. We show that this computationally efficient *distributed* estimator which only has each robot estimate its *own* state can leverage information from other robots through consistent CI can perform accurate and consistent localization in simulation and real-world datasets.

## 1.2 Related Works

Visual-inertial state estimation has over two decades of literature and continues to have significant innovations in recent years as it continues to become foundational to many robotic and commercial applications [80]. In this section, we will cover the key works which relate to the main thrusts of this thesis.

### 1.2.1 Higher Level Plane Primitives in VI-SLAM

Within the literature involving range based sensors, planar features have been shown to improve point cloud registration accuracy [150, 189, 194, 201]. In the field of LiDAR-based odometry, the state-of-the-art method is LOAM [218], which sequentially registers extracted planar and edge features to an incrementally built global map, and demonstrates impressive efficiency and accuracy. Proencca et al. [158] recently presented a planar odometry method that used a modified Hesse plane representation

but during plane matching they defined the cost function as the difference between two points residing on the planes. While this was not used as a planar representation, we will more formally present it as the "closest point" representation in Section 3.2.

One of the first uses of planes in SLAM was by Weingarten et al. [198] who extracted planes from incoming point clouds through a breadth-first region growing algorithm and fused similar planes using a Mahalanobis-distance test. Pathak et al. [151] extended this work by presenting a plane correspondence algorithm that maximized geometric scene consistency and allowed for real-time performance. Trevor et al. [187] combined lines, planes, and odometry measurements in a graph-based framework, but used the overparametrized Hesse form during optimization. Taguchi et al. [184] introduced a handheld RGBD point-plane SLAM system, provided analysis of degeneracy issues, and presented a RANSAC-based approach to the feature correspondence problem. Salas et al. [173] leveraged the creation of a dense planar map to allow for simple localization of incoming RGBD sensor readings through the direct projection of the dense planar map into the camera frame. Kaess [89] presented the unit quaternion plane representation, proposed a relative plane formulation for improved convergence in batch optimization, and demonstrated simple planar 3D mapping with a handheld RGBD sensor. Based on this work, Hsiao et al. [75, 76] performed keyframe-based dense planar SLAM and achieved higher estimation accuracy due to the additional plane constraints and recently incorporated discrete inertial preintegration. More recently, Zhang et al. [219] introduced a fast plane segmentation and map refinement step that improved real-time performance and constructed map quality. Ma et al. [128] used a RGBD camera to perform direct alignment to planar keyframes and optimized a global graph using an expectation-maximization framework. All these works have tried to solve the same basic question: *How does one optimally leverage high level plane primitives in SLAM?*

In the context of specifically LiDAR-inertial, conventionally IMUs have only really been used to provide 3D pose predictions for LiDAR registration methods. The work by Hesch et al. [73] first proposed a LiDAR-aided inertial EKF that used a 2D

LiDAR for indoor mapping where they assumed that all planes extracted were orthogonal in the environment and also performed frequent stops to prevent drift in the unobservable z-direction. Guo et al. [67] investigated the observability properties of an IMU-RGBD system and constructed an observability-constrained Kalman filter. They leveraged both point and planar features and only used the plane orientation as a measurement losing the information captured in the distance to the plane while avoiding singularity issues. In this thesis, we will present a new minimal plane representation called the Closest Point (CP) and demonstrate its efficient incorporation within a full LiDAR-inertial SLAM estimator.

We next consider the visual-inertial sensing case and the use of geometric planes within this context. Existing literature has primarily focused on explicitly detecting line and plane features with stereo or depth sensors [67, 204, 212, 223]. In particular, many methods have leveraged line features in conjunction with Manhattan [26] or Atlanta world [176] regularities, improving accuracy due to structured lines (e.g., aligned with building cardinal directions) that directly provide global attitude information [68, 69, 102, 117, 197, 227]. As environmental planes cannot be directly detected with a single monocular camera since the depth is unavailable, generic depth sensors that can directly measure environmental planes – such as RGB-D [67, 75, 104, 209], 1D LRF [77, 175], or 3D LiDAR [106, 228, 230] – have been fused with great success. Similarly to line features, planar Manhattan frames have been leveraged with success [99, 216]. Additionally, some works have enforced cross-plane orthogonality, parallelism [75, 117], or point-on-plane regularities [209], but require an additional sensor which increases cost, computation, calibration complexity, and data association challenges. Recently, deep-learning-based methods have become of interest due to their ability to perform single-shot detection of planar surfaces and normals [29, 118, 195, 214]. For example, RP-VIO [162] leverages a plane segmentation network [203] to separate planar surfaces which are assumed to be static within a dynamic environment and enforce point-on-plane camera homography constraints.

Closest to the method presented in this thesis, which leverages planar structural regularities, is that by Rosinol et al. [164–166]. They proposed a *stereo* VIO system that incrementally builds and estimates 3D meshes (planes) in which point-on-plane structural regularities are enforced during optimization. They have shown that the inclusion of planar regularities improves both state estimation and environmental mesh accuracy. This plane detection method was extended to include lines within the monocular VINS-Mono [160] framework in PLP-VIO [116], which additionally enforced point-to-line and line-to-plane regularities. Both *only* enforce structural regularities for vertical and horizontal planes (with respect to gravity), require the inclusion of planes in the state (increasing computation), and may experience significant computational spikes when the number of constraints grows. We will show that the inclusion of these planar regularities within an efficient filter-based VINS reduces trajectory drift due to the significant periods for which spatial planes can be tracked.

### 1.2.2 Leveraging Loop-closures and VI-SLAM

We now focus on long-term visual-inertial simultaneous localization and mapping (VI-SLAM). The related works have been summarized in Table 1.1. In the following subsections, we will first survey the literature for how loop-closures are leveraged when performing map-based localization. We will then consider the most common architectures, termed "decoupled", where a VIO is combined with a secondary process which then performs non-linear refinement and publishes a result. Importantly, the information only flows from the VIO to the backend sectionary optimization, not the other way. We then look at methods that close the "loop", and allow the VIO frontend to leverage the backend optimized result. Finally, we then focus on the few works which have tried to perform VI-SLAM in a consistent manner.

#### 1.2.2.1 Prior Map-based VINS

One of the earliest works which incorporated prior features within an efficient filter-based estimator was Mourikis et al. [141] which leveraged a prior surface map

Table 1.1: Overview of key works which build a map online (SLAM?), have real-time frontend leverage loop-closure-aided results (Feedback?), can perform relinerization to handle large loop-closures (Relinearize?), and provide consistent uncertainty estimation of the state (Consistent?).

| | SLAM? | Feedback? | Relinearize? | Consistent? |
|---|---|---|---|---|
| [126, 127, 138, 141, 178, 192] | | | | |
| [34, 62] | | | | * |
| [92, 119, 123, 160, 165, 190] | * | | * | |
| [16, 100, 108] | * | * | * | |
| [59, 65, 79, 93, 224] | * | * | | * |
| [140, 174], Proposed | * | * | * | * |

to aid a planetary terrain-relative estimation during entry, landing, and decent (EDL). Many later works by Ventura et al. [192] and Middelberg et al. [138] showed that client-server architecture could be leveraged for which a remote server provided the compute to perform global localization which could be communicated back to a client low-power device. These works, along with many others, treated the map as perfectly known with sufficient accuracy and thus didn't consider this source of error. A work which looked to address some of the inconsistencies is that by Lynen et al. [126, 127]. They investigated reduction techniques for visual descriptor and map summarization to ensure sufficient localization performance. The leveraged prior map feature positions were still treated as true, but loop-closure measurement noises were inflated to account for dropping the correlation between the current state and the map itself. There have been significant works which have looked at map summarization and sparsification [20, 35, 36, 69, 143], with the work by Dymczyk et al. [35, 36] showing that optimal removal of over 95% of map features had little impact on prior map localization performance. This work was released as a part of the maplab framework [27, 178], which additionally contains an extension of ROVIO [8], termed ROVIOLI, which provided prior map relocalization through pose updates, ignoring the consistency problem.

Closer to the work presented in this thesis are methods which try to address the consistency problem when leveraging an uncertain prior map and estimate the correlation of the current estimate with the prior map after loop-closure. The work by Dutoit

et al. [34] introduced the Cholesky-Schmidt-Kalman filter which explicitly considered the uncertainty of a given fixed prior map and enabled the tracking of correlations between the current state and map. This is typically expensive, but through leveraging a proposed Cholesky-SKF (C-SKF) update and a submap-based relaxation which split the map into independent submaps with duplicated features [69, 147], they showed impressive performance and consistency. While this allowed for consistency and $O(n)$ in terms of the map size complexity, the extension of the C-SKF to real-time simultaneous map building is unclear. In the later sections of this thesis we will compare and contrast different schemes in the context of map-based localization in terms of their complexity, consistency, and memory.

#### 1.2.2.2 SLAM without Feedback (decoupled)

A common approach to incorporate loop-closure information and allow for relinearization is to *decouple* the high-frequency odometry by leveraging the relative pose information generated to construct a backend secondary relative pose graph. This was seen in the earlier works by Engel et al. [46, 47] which leveraged a secondary pose thread, and direct sparse odometry [45] which was extended to include loop-closures [54] and inertial information [123, 124]. The works by Kasyanov et al. [92] and Qin et al. [160] further popularized this methodology. Specifically, they construct a visual-inertial sliding window bundle adjustment frontend which feeds pose estimates into a backend pose-graph backend which then inserts relative pose constraints between keyframes. This is decoupled by nature as the frontend cannot gain any corrections, and thus continuously drifts. Loop-closure constraints between keyframes are found through PnP of the landmark feature estimates provided by the frontend after performing place recognition. To recover a drift-free pose estimation, the most recent optimized keyframe estimate is used to "correct" the frontend drift by simple concatenation of the keyframe to the latest estimated relative pose from the frontend. This architecture is also leveraged in the work by Rosinol et al. [165], which additionally robustifies their secondary graph to loop-closure outliers with an incremental version

of PCM [129]. All these methods are unable to recover the uncertainty of the latest pose and leverage an unweighted relative pose graph.

The work by Liu et al. [119] and Usenko et al. [190] focused on recovering factors for use in the secondary optimization. Liu et al. [119] presented ICE-BA which optimized a sliding window relative to the last marginalized keyframe. This relative prior enables the backend drift to be corrected, but it is unclear if the frontend is able to leverage loop-closures. Usenko et al. [190] presented BASALT which recovers relative poses and global roll and pitch factors through non-linear factor recovery (NFR) [133], which were directly combined with traditional re-projection errors in the backend. Importantly, they showed that by leveraging the uncertainty of the frontend (as compared to uniformly weighting all edges) the backend optimization had accuracy gains.

#### 1.2.2.3 SLAM with Feedback (tightly-coupled)

The parallel tracking and mapping (PTAM) [100] methodologies continue to remain popular due to their ability to split the computational complexity across threads in a multi-core machine. Most notably, the ORB-SLAM [16, 144–146] family of works has expanded to include inertial information to further improve performance. ORB-SLAM 3 [16] maintains three threads which perform "tracking" of the most recent image to a *fixed* map alongside inertial information, "local mapping" which manages keyframes and optimizes a small window states around the current estimate, and "loop and map merging" which detects large scale loop-closures, optimizes a pose graph and propagates these loop-closures through the feature map. Leutenegger [108] recently presented the OKVIS 2 extension of OKVIS [110] with a focus on the marginalization of active states into relative pose factors, and loop-closures to prior locations. However, as common with the majority of all graph-based systems, all these methods are unable to recover their uncertainty, guarantee any sort of consistency, and are inherently inconsistent due to fixing of historical states and thus incorrectly conditions the current estimate on them.

More recently, there has been an effort to enable real-time consistent estimation which leverages incrementally built maps online but does not refine the map after its creation to reduce complexity. This was first shown by the conference publications of this thesis [59, 65] which leveraged the Schmidt-Kalman filter (SKF) to reduce the computational complexity of building a consistent map online and introduced the 2D-to-2D pose measurement model which showed impressive real-time performance with high accuracy. The work by Ke et al. [93] focused on a square-root information variant of the SKF to enable efficient estimation, but required state re-ordering between "exploration" and "relocalization" phases, which was later addressed in the work by Huai and Huang [79].

#### 1.2.2.4 Persistent Large-scale Consistent VI-SLAM with Feedback

Of the many works which have looked to incorporate loop-closure information, only a few allow for relinearization of states, feedback of backend estimates to the real-time frontend, and tried to address the inconsistency problems. The two works closest to the work in this thesis are the dual-layered estimation framework by Mourikis and Roumeliotis [140] and the method leveraged for multi-robot decentralized cooperative localization presented by Sartipi et al. [174]. The method by Mourikis and Roumeliotis [140] proposes to run a light-weight MSCKF until a loop-closure is detected, after which it is paused and a visual-inertial batch least-squares (VI-BLS) refines all states after which the sliding window covariance is recovered in its entirety. The key advantage is that this approach allows for the frontend covariance to incorporate relinearizations and loop-closures, at the cost of significant periods of high latency while it pauses for the BA to optimize. The method by Sartipi et al. [174] has each agent optimize a sub-set of their prior map and publish this to the other users which can then loop-close and localize against.

Both these methods perform full VI-BLS in their backend (i.e., they optimize both states and environmental features), which presents a significant computational challenge for both optimization and covariance recovery. Mourikis and Roumeliotis

[140] address this problem by selectively marginalizing historical states to bound the complexity but note that this limits the ability to loop-closure and prevents later relineraization. The work by Sartipi et al. [174] only optimizes the latest trajectory segment until the optimization process takes more than 6 seconds to complete. They additionally use an approximate marginal feature uncertain (which is leveraged by frontend) by conditioning on the camera poses they are anchored in, reducing complexity. To address the inconsistency of incremental VIO drift, they argue that the estimation and inflate of the relative robot to map transform is able to prevent the underlying inconsistencies.

As compared to these two works, in this thesis, we present a backend method which removes the need to estimate environmental features by leveraging a consistent relative pose graph formulation which reduces the complexity of the VI-BLS problem. Particular focus is spent on how we recover the relative uncertainty between keyframes, and how we are able to quantify the uncertainty of loop-closures provided through relative pose PnP. We show that this backend can be used to reduce VIO drift and corrected map feature estimates all while marginal uncertainties of environmental point features are efficiently and consistency recovered. We stress that we deliberately avoid performing a full VI-BLS to enable the use of the proposed architecture on more resource constrained platforms and keep a significant number of historical states to maximize the probability of loop-closures.

### 1.2.3 Distributed Cooperative Localization

In the last thrust of this thesis we look at the multi-robot case and how we can reduce the complexity through *distributed* estimation while remaining consistent. A naive extension of the single-robot estimation framework to the *multi-robot* case would be prohibitively costly and non-real-time. For example, one could communicate all measurements generated from itself to each other (or fusion center), where all measurements could be optimally fused and all states can be refined jointly. While this does allow for accurate estimation, both the requirement for constant communication

and the joint estimation of robot states requires $O((m*n)^3)$ computational complexity in terms of the number of $m$ robots. As such, a multi-robot *distributed* estimator is needed to address these shortcomings by relaxing communication requirements and distributing the computation cost across all robots resulting in $O(m*n^3)$ complexity.

Efficient 2D CL has focused on the fusion of relative measurements between robots (e.g., relative robot-to-robot bearing or distance range measurements). Roumeliotis et al. [169] proposed a decentralized algorithm that achieves performance equivalent to the centralized formulation, but requires communication between all robots and increases in computational cost due to its centralized nature as the number of robots grows. Other works such as [122] have investigated the approximation of the robot-to-robot cross-covariances that are not involved in a relative measurement update to reduce the computational cost, and while it performs close to its centralized, it is unable to guarantee consistency and thus can easily diverge. More recently, Jung et al. [88] extended this work to the 3D case, but inherits the same underlying issues and requires maintaining the approximated robot-to-robot cross-covariances. There exist other works aiming at estimating the relative poses between robots using relative measurements [131, 202]. Alternative approaches have leveraged CI [18, 225] to guarantee consistency and only require that each robot maintains its own state and auto-covariance (the correlations between robots are ignored). By contrast, in this thesis we specifically take advantage of the CI formulation for 3D multi-robot state estimation, enabling a consistent distributed algorithm which fuses inertial and visual sparse environmental feature information.

As compared to CL with relative distance, bearing, or poses between robots [18, 88, 105, 107, 122, 130, 131, 169, 202, 225], common sparse environmental features are used in [91, 135, 154, 174], which is appealing as getting relative robot information can be difficult with visual-inertial sensors in practice and requires both the detection and tracking of other robots (e.g., [42, 44]). For example, Melnyk et al. [135] introduced CL-MSCKF using common environmental feature constraints within a centralized formulation that jointly estimated all robot states. They required that robots

communicate all sensor data to a common fusion center and demonstrated its use for the two robot case in simulation. Karrer et al. [91] developed a graph-based centralized server which handled non-real-time computationally expensive loop closure detection and optimization of all robot maps to find the joint global optimal.

The closest work is that by Sartipi et al. [174] which introduced a distributed method for multi-user AR experiences through the use of multi-map feature constraints. Common features were detected in environmental maps received from other users and the transmitted feature position estimates were used to constrain the user's state directly. Instead of inflating measurement noise to compensate for the unknown correlations between the current user and the other user's map, we leverage CI that theoretically guarantees consistency to handle the unknown correlations. Also, instead of requiring that all common features must match sparse features in the other user's map, we leverage the other user's common feature measurements directly allowing for updates with additional measurements.

### 1.2.4 Era of Deep Learning

While the majority of the research presented in this thesis does not leverage deep learning techniques, it remains an ever-evolving field which has continued to develop interesting solutions which continue to become more and more applicable to robotics. In the following sections, we summarize some interesting works and discuss how they could be leveraged in the future.

#### 1.2.4.1 Bayesian Deep Networks

A key to incorporating deep neural networks into the more traditional probabilistically sound SLAM algorithms is modeling their prediction uncertainties. Gal et al. [51, 52] introduced the idea of using dropout layers as a Bayesian approximation to the underlying deep Gaussian process. They evaluated their proposed method for recovering model uncertainty on a variety of tasks and showed improvement over the

state-of-the-art. Within the computer vision field, Kendall et al. [95, 96, 98] investigated the use of *aleatoric* uncertainty, which captures noise inherent to the input data, and *epistemic* uncertainty, which captures the imperfect nature of the trained model parameters. Kendall et al. showed that by modeling both uncertainties the combined uncertainty better captures the predictive error. Additionally in multi-task learning, leveraging the uncertainties to normalize loss functions improves prediction performance as compared to each independent training of tasks. Gast and Roth [55] proposed to obtain predictive uncertainties efficiently through proposed probabilistic output layers, replacing intermediate activation with distributions, and used density filtering to propagate activation uncertainties through the network in a single pass. While these all aim to provide a predictive variance, none have looked at what *type* of distribution these predictions follow.

There have also been a few works that look to do end-to-end pose estimation from input images [94, 97, 196]. Notably, Kendall and Cipolla [94] fitted an unimodal Gaussian to sampled pose predictions and noted that a deeper network was needed to discriminate visually similar poses and thus remove the multi-modal (multi-hypothesis) nature of the uncertainty. This work stands out as being one that provides some preliminary investigation on what properties the predictive distribution has.

#### 1.2.4.2 Deep Networks and SLAM

Outside of quantifying the uncertainty of the networks, many have looked to incorporate networks into their SLAM systems through different heuristics or use cases. There have been quite a few works which look to leverage segmentation of the image to handle dynamic environmental objects. Yu et al. [215] used a segmentator to classify points, and then checked each point to see if it is dynamically moving or not and rejected dynamic points to prevent them from being included in the estimator. Brasch et al. [9] focused on highly dynamic scenes where most of the camera viewport is taken up by dynamic objects. They proposed using a network to predict an inlier ratio for all

3d features, thus allowing for features which are more likely to be dynamic to be down-weighted accordingly in the bundle adjustment. Esfahani et al. [48] proposed getting both pixel-level semantic and dense optical flow which allows for distinguishing of which semantic objects are moving or stationary (i.e., a car can be parked or moving). While these works were successful, none of these works directly modeled that the network predictions are imperfect.

There have been many SLAM systems which look to incorporate or leverage network uncertainty [19, 134, 183, 222], which typically rely on hand-picked values for the classification uncertainties or confidence scores. Other works have focused on estimating environmental semantic objects outside of the current camera state estimation [31, 49], preventing the current state estimate from improving from the additional semantic information. There have been quite a few works that leverage depth estimates in SLAM [120, 185, 203], but, to the best of our knowledge, none have addressed the aleatoric and epistemic uncertainties and instead rely on hand-tuned predictive uncertainties. Kopitkov and Indelman [101] introduced the idea of compressing images into a feature vector, and learning a measurement model offline for this feature vector including both a predictive mean and covariance given the current state estimates to incorporate into a batch-optimization SLAM problem as an additional factor. This contrasts traditional deep-learning state estimation approaches which typically predict a pose estimate. They argue that learning the conditional prediction on the current pose is more unimodal as compared to modeling the uncertainty of a predicted pose. They provide a few preliminary results with a limited investigation on if this unimodal nature is true. To the best of our knowledge, none of these works model that the network is imperfect and has limited applicability outside of their training data due to this. We argue that modeling the network as uncertain is key to robustly leveraging it in practical SLAM systems and is crucial to ensuring accurate online state estimation.

Clark et al. [25] looked at incorporating inertial measurements into an end-to-end network and showed that it outperformed SLAM systems which do not take into account the inaccuracy of camera-IMU calibration. Brossard et al. [11], recently

focused on learning IMU and odometry dead reckoning corrections, which could allow for more accurate SLAM systems due to corrected predictions. There have additionally been a few works that have looked into replacing the underlying Kalman filter equations with deep networks [74, 103, 163]. While still in their infancy, in the future better performance might be able to be achieved by leveraging a specifically trained estimator for fusion.

### 1.3 Research Objectives

The first objective of the research presented in this dissertation is to develop a comprehensive state-of-the-art visual-inertial navigation system, which can then enable further research of the aforementioned problems. The second objective is to leverage the proposed framework to address the challenges of long-term persistent single and multi-robot localization and its sub-challenges of geometric feature representation for persistent long-term loop-closure, efficient and consistent visual-inertial simultaneous localization and mapping (VI-SLAM), and finally efficient and consistent distributed multi-robot localization. A key focus throughout this work is to ensure proper consistency of the estimator which is crucial for practical use for downstream applications (e.g., control and planning). We argue that "consistency is necessary for filter optimality" [4, Section 5.4.1], and thus the estimation of the current mean alongside its uncertainty which captures the true error of the state is a fundamental goal. Within the context of long-term VI-SLAM, the consistency of the estimator has traditionally been a lower priority or ignored due to the significant computational cost penalty of tracking the correlations between the active state (e.g., current pose and velocity) and the historical map. In this thesis, we present multiple methodologies which can *efficiently* perform visual-inertial estimation in a *consistent* manner, demonstrating that it is possible and desirable to perform consistent persistent long-term VI-SLAM.

### 1.3.1 Research Framework for VINS

We present an open-sourced visual-inertial research framework which is leveraged throughout this thesis and also the broader research community. Termed OpenVINS, the codebase[1] contains an on-manifold sliding window Kalman filter, online camera intrinsic and extrinsic calibration, camera to inertial temporal calibration, in-state environmental "SLAM" landmarks with a variety of different representations, a modular type system which enables prototyping, an extendable visual-inertial simulator to verify trajectory and calibration accuracy along with the *consistency* of the estimator, static and dynamic state initialization, a groundtruthing pipeline[2] to recover groundtruth trajectories for evaluation, and finally an evaluation toolbox to evaluate different metrics to quantify performance. Significant effort has been taken to both document the code but also provide theoretic derivations which enable new graduate students to get started with visual-inertial estimation with only a minimal amount of linear systems background needed.[3] The proposed OpenVINS is compared against state-of-the-art open-sourced algorithms, showing its competing estimation performance.

### 1.3.2 Extending VINS to Higher Level Geometric Primitives

We then focus on how to incorporate higher-level geometric primitives such as planes within the context of VINS. A novel formalization of geometric planes, termed the Closest Point (CP) plane representation, is developed to represent planes with a minimal representation (3 degree of freedom (DoF)) and improved linearity which makes it suitable for use within simultaneous localization and mapping (SLAM) estimators. Within the context of a LiDAR-inertial SLAM system,[4] we show that CP planes can compress large amounts of environmental features into single 3 DoF plane

[1] https://github.com/rpng/open_vins

[2] https://github.com/rpng/vicon2gt

[3] https://docs.openvins.com/

[4] https://github.com/rpng/lips

and plane-to-plane constraints can then be leveraged with inertial preintegration [40] to perform accurate localization. We show that the CP representation can have more accurate localization performance as compared to the unit quaternion representation [89] while additionally simplifying the plane-to-plane constraint to simple efficient subtraction between transformed CP planes.

The CP representation is then applied to the filter-based OpenVINS monocular-inertial configuration to estimate environmental planes *without* requiring additional sensing modalities, e.g. stereo or depth camera, or a computationally expensive neural network to provide measurement and detection of such planes. Instead, these planes, which can be viewed by the camera for significant periods in man-made environments due to their large spatial presence, are used to constrain sparse point features over time through a point-on-plane constraint. Particular focus is spent on how to detect, extract, and track these planes from only monocular images efficiently and how CP planes are initialized, merged, and marginalized over time. The proposed approach is evaluated with extensive Monte-Carlo simulations and real-world experiments, including an author-collected AR scenario[5], and is shown to outperform the point-based VIO in structured environments. This work is open sourced[6] for the benefit of the research community.

### 1.3.3 Efficient Schmidt-Kalman Filtering for Persistent and Consistent Localization

We next focus on how to perform *consistent* long-term localization which has been typically plagued by the large amount of additional computational burden required to continuously track correlations with historical states and environmental features. While keeping historical states enables consistent loop-closure through visual feature matches, they are detrimental to estimator complexity, which is typically $O(n^3)$

[5] https://github.com/rpng/ar_table_dataset

[6] https://github.com/rpng/ov_plane

in size of the state, since more historical states are continuously added during environmental exploration and thus computational costs tend to explode over time. To solve this, we first proposed two estimators which leverage the computational properties of the Schmidt-Kalman filter (SKF) [177] to reduce the cost of keeping old states and still maintaining consistency through tracking of the correlations. A novel 2D-to-2D observation model is leveraged within the context of long-term loop-closures as compared to the more traditional 2D-to-3D model and is shown to also provide significant reductions in the long-term drift and computational costs. A novel dynamic Schmidting' method which allows features to be corrected when re-observed is then proposed and shown to balance computational efficiency and accuracy.

We then investigate how to leverage these efficient frontends in conjunction with a secondary thread which can enable re-linearization and thus correction of large loop-closure through relocalization to previously visited locations. Specifically, we have an efficient multi-state constraint Kalman filter (MSCKF) [139] Schmidt-Kalman filter (SKF) [177] frontend which contains a temporal map, e.g., a local map of recently viewed areas, which is then augmented by marginalized environmental features which have incorporated loop-closure information through relinearization. A particular focus is how to leverage these marginal features, which are optimized via a lightweight relative-pose graph optimization and can be incorporated via loop-closures, in a consistent manner which ensures global consistency and bounded localization errors over time. The hybrid extension of the MSCKF to map-based localization using different measurement techniques is investigated through a series of detailed numerical simulation experiments to demonstrate its real-time localization accuracy and efficiency. To the best of our knowledge, this is the first work to systematically study a hybrid filter-based estimator which can perform VI-SLAM in a consistent and efficient manner.

### 1.3.4 Distributed Cooperative Visual-Inertial Localization via Covariance Intersection

We then apply the developed loop-closure detection and constraint methodologies to the multi-robot / multi-user cooperative localization (CL) case. Accurate and efficient CL that enables multi-user augmented reality (AR) experiences, multi-device cooperative mapping, and multi-vehicle formation control, is a key barrier to overcome due to challenges of communication, distributed computation, and complexity of fusing multi-robot asynchronous measurement constraints. In particular, we focus on *distributed* CL VINS which has computational advantages through reducing the state size that each robot needs to estimate to just their own as compared to concurrent estimation of *all* robot states. We build on a consistent estimator distributed estimator design presented by Zhu et al. [226] which uses covariance intersection (CI) [87]. A novel and non-trivial extension to include persistent environmental "SLAM" features through two novel measurement constraints and a loop-closure method for historical locations which limits drift and enables *asynchronous* common views seen from other robots' historical poses to be fused is proposed. As a result, the proposed distributed CL estimator does *not* require *simultaneous* viewing of the same location due to leveraging of historical common features (e.g., a robot can gain information if another robot had previously explored the same location), while significantly improving the localization performance thanks to such common multi-robot measurement information and SLAM features. Particular focus is spent on validating the accuracy and consistency of the proposed CL CI-based VINS, with both Monte-Carlo simulations and real-world experiments which are directly compared to the more computationally expensive centralized CL estimator designs.

# Chapter 2

# OPENVINS: A RESEARCH PLATFORM FOR VISUAL-INERTIAL ESTIMATION

## 2.1 Introduction

In this section, we present the research framework, termed OpenVINS, which has enabled both the research in this thesis along with many works within the lab and external to our research group. This codebase has been the foundation of many of the recent visual-inertial estimation projects in our group at the University of Delaware, which include multi-camera [38], multi-IMU [41], visual-inertial moving object tracking [42, 44], Schmidt-based visual-inertial SLAM [59, 65], point-plane and point-line visual-inertial navigation [208, 209], among others [207, 228, 229]. The flexibility, plentiful documentation, and state-of-the-art performance are key to allowing this codebase to provide value to both the work done in this thesis and to others. We summarize the key functionality of the different components in OpenVINS as follows:

- *ov_core* – Contains 2D image sparse visual feature tracking; linear and Gauss-Newton feature triangulation methods; visual-inertial simulator for arbitrary number of cameras and frequencies; and fundamental manifold math operations and utilities.

- *ov_init* – Both stationary and dynamic initialization [32, 33, 64] which enable robust and accurate recovery of initial biases, velocity, and gravity direction. Special care is also taken to recover the initial uncertainty of the state.

- *ov_eval* – Contains trajectory alignment; plotting utilities for trajectory accuracy and consistency evaluation; Monte-Carlo evaluation of different accuracy metrics; and utility for recording ROS topics to file.

- *ov_msckf* – Contains the extendable modular Extended Kalman Filter (EKF)-based sliding window visual-inertial estimator with on-manifold type system for flexible state representation. Features include: First-Estimates Jacobains (FEJ) [82–84], IMU-camera time offset calibration [114], camera intrinsics and extrinsic online calibration [115], IMU intrinsic calibration [205, 210], standard MSCKF [139], and 3D SLAM landmarks of different representations [112].

In what follows we describe our generalized modular on-manifold EKF-based estimator which, in its simplest form, estimates the current pose and velocity of a camera-IMU pair. We then introduce the implemented features that provide the foundation for researchers to quickly build and extend on. Finally, we evaluate the proposed EKF-based solution in simulations and then on real-world datasets, demonstrating its competing performance against other open-sourced algorithms.

### 2.2 Visual-Inertial Estimation Formulation

The state vector of our visual-inertial system consists of the current inertial navigation state, a set of $c$ historical IMU pose clones, a set of $m$ environmental landmarks, and a set of $w$ cameras' extrinsic and intrinsic parameters. The clones will be used to update via multi-pose constraints via the Multi-State Constraint Kalman Filter (MSCKF) [139] methodology, while environmental landmarks, termed "SLAM" features, allow for direct estimation of features until their tracking is lost. More formally we have the following state:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_C^\top & \mathbf{x}_M^\top & \mathbf{x}_W^\top & {}^{C}t_I \end{bmatrix}^\top \tag{2.1}$$

$$\mathbf{x}_I = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_k}^\top & {}^{G}\mathbf{v}_{I_k}^\top & \mathbf{b}_{\omega_k}^\top & \mathbf{b}_{a_k}^\top \end{bmatrix}^\top \tag{2.2}$$

$$\mathbf{x}_C = \begin{bmatrix} {}^{I_{k-1}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{k-1}}^\top & \cdots & {}^{I_{k-c}}_{G}\bar{q}^\top & {}^{G}\mathbf{p}_{I_{k-c}}^\top \end{bmatrix}^\top \tag{2.3}$$

$$\mathbf{x}_M = \begin{bmatrix} {}^{G}\mathbf{p}_{f_1}^\top & \cdots & {}^{G}\mathbf{p}_{f_m}^\top \end{bmatrix}^\top \tag{2.4}$$

$$\mathbf{x}_W = \begin{bmatrix} {}^{I}_{C_1}\bar{q}^\top & {}^{C_1}\mathbf{p}_I^\top & \boldsymbol{\zeta}_0^\top \cdots & {}^{I}_{C_w}\bar{q}^\top & {}^{C_w}\mathbf{p}_I^\top & \boldsymbol{\zeta}_w^\top \end{bmatrix}^\top \tag{2.5}$$

where ${}^{I_k}_{G}\bar{q}$ is the unit quaternion parameterizing the rotation $\mathbf{R}({}^{I_k}_{G}\bar{q}) = {}^{I_k}_{G}\mathbf{R}$ from the global frame of reference $\{G\}$ to the IMU local frame $\{I_k\}$ at time $k$ [186], $\mathbf{b}_\omega$ and $\mathbf{b}_a$ are the gyroscope and accelerometer biases, and ${}^{G}\mathbf{v}_{I_k}$ and ${}^{G}\mathbf{p}_{I_k}$ are the velocity and position of the IMU expressed in the global frame, respectively. The inertial state $\mathbf{x}_I$ lies on the manifold defined by the product of the unit quaternions $\mathbb{H}$ with the vector space $\mathbb{R}^{12}$ (i.e. $\mathcal{M} = \mathbb{H} \times \mathbb{R}^{12}$) and has 15 total degrees of freedom (DoF).

For vector variables, the "boxplus" and "boxminus" operations, which map elements to and from a given manifold [70], equate to simple addition and subtraction of their vectors. For quaternions, we define the quaternion boxplus operation as:

$$\bar{q}_1 \boxplus \delta\boldsymbol{\theta} \triangleq \begin{bmatrix} \frac{\delta\boldsymbol{\theta}}{2} \\ 1 \end{bmatrix} \otimes \bar{q}_1 \simeq \bar{q}_2 \tag{2.6}$$

Note that although we have defined the orientations using the *left* quaternion error, it is not limited to this and any on-manifold representation in practice can be used (e.g., [199]). A set of $c$ historical stochastic pose clones [170], defined as $\mathbf{x}_{T_k} = [{}^{I_k}_{G}\bar{q}^\top \ {}^{G}\mathbf{p}_{I_k}^\top]^\top$, are kept in $\mathbf{x}_C$ to allow for historical features to be incorporated and to enable feature triangulation. The map of environmental landmarks $\mathbf{x}_M$ contains global 3D positions only for simplicity, while in practice we offer support for different representations (e.g. inverse MSCKF [139], full inverse depth [24], and anchored 3D position [153], see Appendix A.6).

The calibration vector $\mathbf{x}_W$ contains the camera intrinsics $\boldsymbol{\zeta}$, consisting of focal length, camera center, and distortion parameters, and the camera-IMU extrinsics, i.e., the spatial transformation (relative pose) from the IMU to each camera. Since we consider synchronized camera clocks, we include a single time offset ${}^{C}t_I$ between the IMU and the camera clock in the calibration vector.

### 2.2.1 Inertial Propagation

The inertial state $\mathbf{x}_I$ is propagated forward using incoming IMU measurements of linear accelerations ${}^{I}\mathbf{a}_m$ and angular velocities ${}^{I}\boldsymbol{\omega}_m$ based on the following generic

nonlinear IMU kinematics propagating the state from timestep $k-1$ to $k$ [21]:

$$
\begin{aligned}
{}^{I}_{G}\dot{\bar{q}}(t) &= \frac{1}{2}\begin{bmatrix} -\lfloor {}^{I}\boldsymbol{\omega}(t)\times\rfloor & {}^{I}\boldsymbol{\omega}(t) \\ -{}^{I}\boldsymbol{\omega}(t)^{\top} & 0 \end{bmatrix} {}^{I_t}_{G}\bar{q} && (2.7)\\
&\triangleq \frac{1}{2}\boldsymbol{\Omega}({}^{I}\boldsymbol{\omega}(t)){}^{I_t}_{G}\bar{q} && (2.8)\\
{}^{G}\dot{\mathbf{p}}_{I}(t) &= {}^{G}\mathbf{v}_{I}(t) && (2.9)\\
{}^{G}\dot{\mathbf{v}}_{I}(t) &= {}^{I_t}_{G}\mathbf{R}^{\top}{}^{I}\mathbf{a}(t) - {}^{G}\mathbf{g} && (2.10)\\
\dot{\mathbf{b}}_{\mathbf{g}}(t) &= \mathbf{n}_{wg}(t) && (2.11)\\
\dot{\mathbf{b}}_{\mathbf{a}}(t) &= \mathbf{n}_{wa}(t) && (2.12)
\end{aligned}
$$

where we have modeled the gyroscope and accelerometer biases as a random walk and thus their time derivatives are white Gaussian. Note that the above kinematics have been defined in terms of the *true* acceleration and angular velocities which can be computed as a function of the sensor measurements and state. The relation to the measurements are:

$$
\begin{aligned}
{}^{I}\boldsymbol{\omega}(t) &= {}^{I}_{w}\mathbf{R}\mathbf{D}_{w}\left({}^{w}\boldsymbol{\omega}_{m}(t) - \mathbf{T}_{g}{}^{I}\mathbf{a}(t) - \mathbf{b}_{g}(t) - \mathbf{n}_{g}(t)\right) && (2.13)\\
{}^{I}\mathbf{a}(t) &= {}^{I}_{a}\mathbf{R}\mathbf{D}_{a}\left({}^{a}\mathbf{a}_{m}(t) - \mathbf{b}_{a}(t) - \mathbf{n}_{a}(t)\right) && (2.14)
\end{aligned}
$$

where we have the following IMU intrinsic parameters: scale/axis correction for gyroscope $\mathbf{D}_w$ (6 parameters), scale/axis correction for accelerometer $\mathbf{D}_a$ (6 parameters), rotation from gyroscope to IMU frame ${}^{I}_{w}\mathbf{R}$, rotation from accelerometer to IMU frame ${}^{I}_{a}\mathbf{R}$ and gravity sensitivity $\mathbf{T}_g$ (9 parameters). To estimate these online, the propagation of the state and covariance become a function of these parameters and is covered in detail in [205, 210].

Given the continuous-time ${}^{I}\boldsymbol{\omega}(t)$ and ${}^{I}\mathbf{a}(t)$, in the time interval $t \in [t_k, t_{k+1}]$, and their estimates, i.e. after taking the expectation, ${}^{I}\hat{\boldsymbol{\omega}}(t)$ and ${}^{I}\hat{\boldsymbol{a}}(t)$, we can define the solutions to the above IMU kinematics differential equation.

$$
{}^{I_{k+1}}_{G}\mathbf{R} = \text{Exp}\left(-\int_{t_k}^{t_{k+1}} {}^{I}\boldsymbol{\omega}(t_\tau)\, d\tau\right) {}^{I_k}_{G}\mathbf{R} \tag{2.15}
$$

$$\triangleq \Delta\mathbf{R}_{k-1,k} \; {}^{I_k}_{G}\mathbf{R} = {}^{I_{k+1}}_{I_k}\mathbf{R} \; {}^{I_k}_{G}\mathbf{R} \tag{2.16}$$

$${}^{G}\mathbf{p}_{I_{k+1}} = {}^{G}\mathbf{p}_{I_k} + {}^{G}\mathbf{v}_{I_k}\Delta t_k + {}^{I_k}_{G}\mathbf{R}^\top \int_{t_k}^{t_{k+1}} \int_{t_k}^{s} {}^{I_k}_{I_\tau}\mathbf{R}\, {}^{I}\mathbf{a}(t_\tau)\; d\tau ds - \frac{1}{2}{}^{G}\mathbf{g}\Delta t_k^2 \tag{2.17}$$

$$\triangleq {}^{G}\mathbf{p}_{I_k} + {}^{G}\mathbf{v}_{I_k}\Delta t_k + {}^{I_k}_{G}\mathbf{R}^\top \Delta\mathbf{p}_{k-1,k} - \frac{1}{2}{}^{G}\mathbf{g}\Delta t_k^2 \tag{2.18}$$

$${}^{G}\mathbf{v}_{I_{k+1}} = {}^{G}\mathbf{v}_{I_k} + {}^{I_k}_{G}\mathbf{R}^\top \int_{t_k}^{t_{k+1}} {}^{I_k}_{I_\tau}\mathbf{R}\, {}^{I}\mathbf{a}(t_\tau)\; d\tau - {}^{G}\mathbf{g}\Delta t_k \tag{2.19}$$

$$\triangleq {}^{G}\mathbf{v}_{I_k} + {}^{I_k}_{G}\mathbf{R}^\top \Delta\mathbf{v}_{k-1,k} - {}^{G}\mathbf{g}\Delta t_k \tag{2.20}$$

$$\mathbf{b}_{g_{k+1}} = \mathbf{b}_{g_k} + \int_{t_k}^{t_{k+1}} \mathbf{n}_{wg}(t_\tau)\; d\tau \tag{2.21}$$

$$\mathbf{b}_{a_{k+1}} = \mathbf{b}_{a_k} + \int_{t_k}^{t_{k+1}} \mathbf{n}_{wa}(t_\tau)\; d\tau \tag{2.22}$$

where $\Delta t_k = t_{k+1} - t_k$, ${}^{I_k}_{I_\tau}\mathbf{R} = \text{Exp}(\int_{t_k}^{t_\tau} {}^{I}\boldsymbol{\omega}(t_u)\; du)$ see Eq. (2.31), and vectors are evaluated at their subscript timesteps (e.g., ${}^{G}\mathbf{v}_{I_k} = {}^{G}\mathbf{v}_I(t_k)$). The biases are corrupted by random walk noises $\mathbf{n}_{wg}$ and $\mathbf{n}_{wa}$ that are zero-mean white Gaussians. We have the following integration components:

$$\Delta\mathbf{R}_{k-1,k} \triangleq {}^{I_{k+1}}_{I_k}\mathbf{R} = \text{Exp}\left(-\int_{t_k}^{t_{k+1}} {}^{I}\boldsymbol{\omega}(t_\tau)\; d\tau\right) \tag{2.23}$$

$$\Delta\mathbf{p}_{k-1,k} \triangleq \int_{t_k}^{t_{k+1}} \int_{t_k}^{s} {}^{I_k}_{I_\tau}\mathbf{R}\, {}^{I}\mathbf{a}(t_\tau)\; d\tau ds \tag{2.24}$$

$$\Delta\mathbf{v}_{k-1,k} \triangleq \int_{t_k}^{t_{k+1}} {}^{I_k}_{I_\tau}\mathbf{R}\, {}^{I}\mathbf{a}(t_\tau)\; d\tau \tag{2.25}$$

More generally we have now recovered the following non-linear function:

$$\mathbf{x}_k = f(\mathbf{x}_{k-1}, {}^{I}\mathbf{a}_m, {}^{I}\boldsymbol{\omega}_m, \mathbf{n}_{k-1}) \tag{2.26}$$

where $\mathbf{n}_{k-1} = [\mathbf{n}_g^\top \; \mathbf{n}_a^\top \; \mathbf{n}_{wg}^\top \; \mathbf{n}_{wa}^\top]^\top$ contains the zero-mean white Gaussian noise of the IMU measurements along with random walk bias noise. This state estimate is evaluated at the current estimate:

$$\hat{\mathbf{x}}_{k|k-1} = f(\hat{\mathbf{x}}_{k-1|k-1}, {}^{I}\mathbf{a}_m, {}^{I}\boldsymbol{\omega}_m, \mathbf{0}) \tag{2.27}$$

where $\hat{\cdot}$ denotes the estimated value and the subscript $k|k-1$ denotes the predicted estimate at time $k$ given the measurements up to time $k-1$.

If we linearize this nonlinear model at the current estimate and assume the measurements are constant over our integration period, we can recover the discrete propagation equations which can be used to propagate the state uncertainty forward in time [132, Sec. 2.3-2.4]:

$$\mathbf{P}_{k|k-1} = \mathbf{\Phi}_{k-1}\mathbf{P}_{k-1|k-1}\mathbf{\Phi}_{k-1}^\top + \mathbf{G}_{k-1}\mathbf{Q}_{k-1}\mathbf{G}_{k-1}^\top \tag{2.28}$$

where $\mathbf{\Phi}_{k-1}$ and $\mathbf{G}_{k-1}$ are the state and noise Jacobians and $\mathbf{Q}_{k-1}$ is the continuous time noise matrix of $\mathbf{n}_{k-1}$. As compared to the 15 DoF inertial state $\mathbf{x}_I$ which has an evolution model, see Eq. (2.7)-(2.12), the clones $\mathbf{x}_C$, environmental features $\mathbf{x}_M$, and calibration $\mathbf{x}_W$ states do not evolve with time and thus the corresponding state Jacobian entries are identity with zero propagation noise and allow for exploitation of the sparsity for computational savings. For the $\mathbf{x}_I$ state we have the following:

$$\mathbf{\Phi}_{k|k-1} = \tag{2.29}$$
$$\begin{bmatrix} {}^{I_k}_{I_{k-1}}\hat{\mathbf{R}} & \mathbf{0}_3 & \mathbf{0}_3 & -{}^{I_k}_{I_{k-1}}\hat{\mathbf{R}}\mathbf{J}_r(\delta\hat{\boldsymbol{\theta}})\Delta t_k & \mathbf{0}_3 \\ -\frac{1}{2}{}^{I_{k-1}}_{G}\hat{\mathbf{R}}^\top\lfloor {}^I\hat{\mathbf{a}}(t_{k-1})\Delta t_k^2\times\rfloor & \mathbf{I}_3 & \Delta t_k\mathbf{I}_3 & \mathbf{0}_3 & -\frac{1}{2}{}^{I_{k-1}}_{G}\hat{\mathbf{R}}^\top\Delta t_k^2 \\ -{}^{I_{k-1}}_{G}\hat{\mathbf{R}}^\top\lfloor {}^I\hat{\mathbf{a}}(t_{k-1})\Delta t_k\times\rfloor & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & -{}^{I_{k-1}}_{G}\hat{\mathbf{R}}^\top\Delta t_k \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix}$$

$$\mathbf{G}_{k-1} = \begin{bmatrix} -{}^{I_k}_{I_{k-1}}\hat{\mathbf{R}}\mathbf{J}_r(\delta\hat{\boldsymbol{\theta}})\Delta t_k & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -\frac{1}{2}{}^{I_{k-1}}_{G}\hat{\mathbf{R}}^\top\Delta t_k^2 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & -{}^{I_{k-1}}_{G}\hat{\mathbf{R}}^\top\Delta t_k & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \tag{2.30}$$

where $\Delta t_k = t_k - t_{k-1}$, ${}^{I_k}_{I_{k-1}}\hat{\mathbf{R}} = \text{Exp}({}^I\hat{\boldsymbol{\omega}}(t_{k-1})\Delta t_k)$ , and $\delta\hat{\boldsymbol{\theta}} = {}^I\hat{\boldsymbol{\omega}}(t_{k-1})\Delta t_k$. The function $\text{Exp}(\cdot)$ is the matrix exponential of $\mathbb{SO}(3)$, and $\mathbf{J}_r$ is the right Jacobian of $\mathbb{SO}(3)$ and are defined by [5, 23]:

$$\text{Exp}(\mathbf{v}) = \mathbf{I} + \frac{\sin\theta}{\theta}\lfloor\mathbf{v}\times\rfloor + \frac{1-\cos\theta}{\theta^2}\lfloor\mathbf{v}\times\rfloor^2 \tag{2.31}$$

$$\text{where} \quad \theta^2 = \mathbf{v}^\top \mathbf{v}$$

$$\mathbf{J}_r(\phi) = \mathbf{I} - \frac{1 - \cos(\| \phi \|)}{\| \phi \|^2} \lfloor \phi \times \rfloor + \frac{\| \phi \| - \sin(\| \phi \|)}{\| \phi \|^3} \lfloor \phi \times \rfloor^2 \tag{2.32}$$

### 2.2.2 On-Manifold Measurement Update

Consider the following nonlinear measurement function:

$$\mathbf{z}_{m,k} = h(\mathbf{x}_k) + \mathbf{n}_{m,k} \tag{2.33}$$

where we have the measurement noise $\mathbf{n}_{m,k} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{m,k})$. For the standard EKF update, one linearizes the above equation at the current state estimate. In our case, as in the indirect EKF [186], we linearize Eq. (2.33) with respect to the current zero-mean error state (i.e. $\tilde{\mathbf{x}} = \mathbf{x} \boxminus \hat{\mathbf{x}} \sim \mathcal{N}(\mathbf{0}, \mathbf{P})$):

$$\mathbf{z}_{m,k} = h(\hat{\mathbf{x}}_{k|k-1} \boxplus \tilde{\mathbf{x}}_{k|k-1}) + \mathbf{n}_{m,k} \tag{2.34}$$

$$= h(\hat{\mathbf{x}}_{k|k-1}) + \mathbf{H}_k \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_{m,k} \tag{2.35}$$

$$\Rightarrow \quad \tilde{\mathbf{z}}_{m,k} = \mathbf{H}_k \tilde{\mathbf{x}}_{k|k-1} + \mathbf{n}_{m,k} \tag{2.36}$$

where $\mathbf{H}_k$ is the measurement Jacobian computed as follows:

$$\mathbf{H}_k = \left. \frac{\partial h(\hat{\mathbf{x}}_{k|k-1} \boxplus \tilde{\mathbf{x}}_{k|k-1})}{\partial \tilde{\mathbf{x}}_{k|k-1}} \right|_{\tilde{\mathbf{x}}_{k|k-1} = \mathbf{0}} \tag{2.37}$$

Using this linearized measurement model, we can now perform the following standard EKF update to ensure the updated states remain on-manifold:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} \boxplus \mathbf{K}_k (\mathbf{z}_{m,k} - h(\hat{\mathbf{x}}_{k|k-1})) \tag{2.38}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \mathbf{K}_k \mathbf{H}_k \mathbf{P}_{k|k-1} \tag{2.39}$$

$$\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^\top (\mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^\top + \mathbf{R}_{m,k})^{-1} \tag{2.40}$$

Here we have stressed that we perform an on-manifold update since our orientations are non-linear and thus we need to map from our linear error-state space to our non-linear state space. When we perform our linearization, it is important to address issues with incorrect information gain in unobservable directions (global yaw and position) through

an observability-aware update. We leverage First Estimates Jacobians (FEJ) [83, 84] in which the linearization points of all Jacobians use the same linearization point as the first update for that state variable. This methodology ensures that spurious information is not gained which could make the estimator overconfident (inconsistent) and thus inaccurate and less robust.

## 2.3 Estimator Functionality Improvements

In what follows, we will present the key improvements to the multi-state constraint Kalman filter (MSCKF) [139] originally introduced in 2007, and later improved to include in-state SLAM features in the MSCKF 2.0 [112] which leverages First Estimate Jacobians (FEJ) [83, 84] to ensure proper observability properties. We will present each key improvement and component which has been incorporated within the OpenVINS framework to provide fundamental functionality, improve accuracy to match or exceed the existing state-of-the-art, improve robustness to low-cost sensors or poor calibration, and enable extendability and usability for VINS researchers.

### 2.3.1 Type-based Index System

At the core of the OpenVINS library is the type-based index system. Inspired by graph-based optimization frameworks such as GTSAM [28], we abstract away from the user the need to directly manipulate the covariance and instead provide the tools to automatically manage the state and its covariance. This offers many benefits such as reduced implementation time and being less prone to development errors due to explicit state and covariance access.

Each state variable "type" has internally the location of where it is in the error state which is automatically updated during initialization, cloning, or marginalization operations which affect variable ordering. A type is defined by its covariance location, its current estimate and its error state size. The current value does not have to be a vector but could be a matrix in the case of an $\mathbb{SO}(3)$ rotation representation. The error

state for all types is a vector and thus a type will need to define the boxplus mapping between its error state and its manifold representation (i.e. the update function).

```
class Type {
    // Current best estimate
    Eigen::MatrixXd _value;
    // Index of error state in covariance
    int _id = -1;
    // Dimension of error state
    int _size = -1;
    // Vector correction, how to update
    void update(const Eigen::VectorXd dx);
};
```

One of the main advantages of this type system is that it reduces the complexity of adding new features by allowing the user to construct sparse Jacobians. Instead of constructing a Jacobian for all state elements, the "sparse" Jacobian needs to only include the state elements that the measurement is a function of. This saves computation in the cases where a measurement is a function of only a few state elements and allows for measurement functions to be state agnostic as long as their involved state variables are present.

### 2.3.2 State Variable Delayed Initialization

Based on a set of linearized measurement equations, Eq. (2.36), we aim to optimally compute the initial estimate of a new state variable and its covariance and correlations with the existing state variables. As a motivating example, we here describe how to initialize a new SLAM landmark ${}^{G}\mathbf{p}_{f}$, whose key logic can be used for any new state variable and is generalized to any type within the codebase. As in [111] we first perform QR decomposition (e.g., using computationally efficient in-place Givens

rotations) to separate the linear system in Eq. (2.36) into two subsystems: (i) one that depends on the new state (i.e., ${}^{G}\mathbf{p}_f$), and (ii) the other that does not.

$$\tilde{\mathbf{z}}_{m,k} = \begin{bmatrix} \mathbf{H}_x & \mathbf{H}_f \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_k \\ {}^{G}\tilde{\mathbf{p}}_f \end{bmatrix} + \mathbf{n}_{m,k} \tag{2.41}$$

$$\Rightarrow \begin{bmatrix} \tilde{\mathbf{z}}_{m1,k} \\ \tilde{\mathbf{z}}_{m2,k} \end{bmatrix} = \begin{bmatrix} \mathbf{H}_{x1} & \mathbf{H}_{f1} \\ \mathbf{H}_{x2} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_k \\ {}^{G}\tilde{\mathbf{p}}_f \end{bmatrix} + \begin{bmatrix} \mathbf{n}_{f1} \\ \mathbf{n}_{f2} \end{bmatrix} \tag{2.42}$$

where $\mathbf{n}_{fi} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{fi})$, $i \in \{1, 2\}$. Note that in the above expression $\tilde{\mathbf{z}}_{m1,k}$ and $\tilde{\mathbf{z}}_{m2,k}$ are orthonormally transformed measurement residuals, not the direct partitions of $\tilde{\mathbf{z}}_{m,k}$. With the *top* transformed linearized measurement residual $\tilde{\mathbf{z}}_{m1,k}$ in Eq. (2.42), we now initialize the state estimate of ${}^{G}\hat{\mathbf{p}}_f$, its covariance, and correlations to $\mathbf{x}_k$ [see Eq. (2.38)] by augmenting the current state and covariance matrix.

$${}^{G}\hat{\mathbf{p}}_f = {}^{G}\hat{\mathbf{p}}_f \boxplus \mathbf{H}_{f1}^{-1}\tilde{\mathbf{z}}_{m1,k} \tag{2.43}$$

$$\mathbf{P}_{xf} = -\mathbf{P}_k\mathbf{H}_{x1}^{\top}\mathbf{H}_{f1}^{-\top} \tag{2.44}$$

$$\mathbf{P}_{ff} = \mathbf{H}_{f1}^{-1}(\mathbf{H}_{x1}\mathbf{P}_k\mathbf{H}_{x1}^{\top} + \mathbf{R}_{f1})\mathbf{H}_{f1}^{-\top} \tag{2.45}$$

It should be noted that a full-rank $\mathbf{H}_{f1}$ is needed to perform the above initialization, which normally is the case if enough measurements are collected (i.e., delayed initialization). Note also that to utilize all available measurement information, we also perform EKF update using the *bottom* measurement residual $\tilde{\mathbf{z}}_{m2,k}$ in Eq. (2.42), which essentially is equivalent to the Multi-State Constraint Kalman Filter (MSCKF) [139] update with nullspace projection [213]. More details about equivalence and a version for when Covariance Intersection (CI) is leveraged can be found in Appendix G.

### 2.3.3 Feature Observation Model via Raw Camera Coordinates

We generalize the landmark measurement model as a series of nested functions to encompass different feature parameterizations such as 3D position and inverse depth and so on. Assuming a visual feature that has been tracked over the sliding window

of stochastic clones [170], we can write the visual-bearing measurements (i.e., pixel coordinates) as the following series of nested functions:

$$\mathbf{z}_{m,k} = \mathbf{h}(\mathbf{x}_k) + \mathbf{n}_{m,k} \tag{2.46}$$

$$= h_d(\mathbf{z}_{n,k},\ \boldsymbol{\zeta}) + \mathbf{n}_{m,k} \tag{2.47}$$

$$= h_d(h_p({}^{C_k}\mathbf{p}_f),\ \boldsymbol{\zeta}) + \mathbf{n}_{m,k} \tag{2.48}$$

$$= h_d(h_p(h_t({}^{G}\mathbf{p}_f,\ {}^{C_k}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{C_k})),\ \boldsymbol{\zeta}) + \mathbf{n}_{m,k} \tag{2.49}$$

where $\mathbf{z}_{m,k}$ is the raw uv pixel coordinate; $\mathbf{n}_{m,k}$ the raw pixel noise and typically assumed to be zero-mean white Gaussian; $\mathbf{z}_{n,k}$ is the normalized undistorted uv measurement; ${}^{C_k}\mathbf{p}_f$ is the landmark position in the current camera frame; ${}^{G}\mathbf{p}_f$ is the landmark position in the global frame and depending on its representation may also be a function of state elements; and $\{{}^{C_k}_{G}\mathbf{R}, {}^{G}\mathbf{p}_{C_k}\}$ denotes the current camera pose (position and orientation) in the global frame. The measurement functions $h_d$, $h_p$, and $h_t$ correspond to the intrinsic distortion, projection, and transformation functions, and the corresponding measurement Jacobians can be computed through a simple chain rule. Note that we compute the errors on the raw uv pixels to allow for calibration of the camera intrinsics $\boldsymbol{\zeta}$ and that the function $h_d$ can be changed to support any camera model (e.g., radial-tangential and equidistant). Details of each function can be found in Appendix A.

#### 2.3.3.1 Camera Observation Model Linearization

Looking at a simplified model where we only estimate the pose and feature position, we can linearize this non-linear measurement model and obtain the following residual:

$$\mathbf{r}_k = \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{T_k}, {}^{G}\hat{\mathbf{p}}_f) \tag{2.50}$$

$$\simeq \mathbf{H}_{T_k}\tilde{\mathbf{x}}_{T_k} + \mathbf{H}_{f_k}{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n}_k \tag{2.51}$$

where $\mathbf{H}_{T_k}$ and $\mathbf{H}_{f_k}$ are the measurement Jacobians, and $\tilde{\mathbf{x}}_{T_k}$ and ${}^{G}\tilde{\mathbf{p}}_f$ are the error states for the observation pose $\mathbf{x}_{T_k} = [{}^{I_k}_{G}\bar{q}^\top\ {}^{G}\mathbf{p}_{I_k}^\top]^\top$ at time $t_k$ and feature, respectively.

Throughout this thesis $\hat{\mathbf{x}}$ is used to denote the estimate of a random variable $\mathbf{x}$, while $\tilde{\mathbf{x}} = \mathbf{x} \boxminus \hat{\mathbf{x}}$ is the error in this estimate. The updated linearization point from a correction $\boldsymbol{\delta}\mathbf{x}$ is $\hat{\mathbf{x}}^{\oplus} = \hat{\mathbf{x}} \boxplus \boldsymbol{\delta}\mathbf{x}$.

After sufficient observations of the feature, we can "stack" them to get:

$$\mathbf{r} = \mathbf{H}_T \tilde{\mathbf{x}}_{T_{1..c}} + \mathbf{H}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{n}_f \tag{2.52}$$

where the measurement is a function of $c$ clone poses, $\tilde{\mathbf{x}}_{T_{1..c}} = [\tilde{\mathbf{x}}_{T_1}^\top \cdots \tilde{\mathbf{x}}_{T_c}^\top]^\top$, corresponding to each observation time the feature was seen, and $\mathbf{n}_f$ is the stacked noise. Appendix A contains the complete Jacobians with respect to all feature states necessary for estimation and online calibration.

#### 2.3.3.2 VIO Features: MSCKF Update

For those features that have lost active track in the current window (termed VIO features), we perform the standard MSCKF update [139]. In particular, we first perform BA to triangulate these features for computing the feature Jacobians $\mathbf{H}_f$ (see Eq. (2.52)), and then project $\mathbf{r}_k$ onto the left nullspace of $\mathbf{H}_f$ (i.e., $\mathbf{N}^\top \mathbf{H}_f = \mathbf{0}$) to yield the measurement residual independent of features:

$$\mathbf{N}^\top \mathbf{r}_f = \mathbf{N}^\top \mathbf{H}_T \tilde{\mathbf{x}}_{T_{1..c}} + \mathbf{N}^\top \mathbf{H}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{N}^\top \mathbf{n}_f \tag{2.53}$$

$$\Rightarrow \ \mathbf{r}'_f = \mathbf{H}'_T \tilde{\mathbf{x}}_{T_{1..c}} + \mathbf{n}'_f \tag{2.54}$$

where $\mathbf{H}_T$ is the stacked measurement Jacobians with respect to the navigation states in the current sliding window, $\mathbf{R}'_f = \mathbf{N}^\top \mathbf{R}_f \mathbf{N}$ is the inferred noise covariance [139].

#### 2.3.3.3 SLAM Features: EKF Update

Once the measurement Jacobian and residual are computed, see Eq. (2.52), we can apply the standard EKF update equations to update the state estimates and error covariance [132] through the MSCKF projection [139]. For those features that can be reliably tracked longer than the current sliding window, we will initialize them into the active state, see Section 2.3.2, and perform EKF updates as in the standard EKF-based VI-SLAM.

### 2.3.4 Online Spatiotemporal and Intrinsic Calibration

We perform online spatiotemporal calibration of the camera-IMU time offset and extrinsic transformation and camera intrinsics. Looking at the landmark measurement Eq. (2.49), one can simply take the derivative with respect to the desired variables that one wishes to calibrate online. In this case the function $h_t$ is a function of the $\{{}^{C}_{I}\mathbf{R}, {}^{C}\mathbf{p}_{I}\}$ extrinsics that are used to compute the global camera pose $\{{}^{C_k}_{G}\mathbf{R}, {}^{G}\mathbf{p}_{C_k}\}$ given the inertial pose. We also co-estimate the time offset between the camera and IMU, which can commonly exist in low-cost devices due to sensor latency, clock skew, or data transmission delays via the method of Li [114]. Consider the time ${}^{C}t$ as expressed in the camera clock is related to the same event in the IMU clock, ${}^{I}t$, by a time offset ${}^{C}t_{I}$:

$$ {}^{I}t = {}^{C}t + {}^{C}t_{I} \tag{2.55} $$

To calibrate the camera's intrinsic parameters we can see that we can take the derivative in respect to the intrinsic $\boldsymbol{\zeta}$ in $h_d$ to perform online calibration. We additionally have support for online calibration of the IMU inertial intrinsic parameters which are involved during inertial propagation which is the open-sourced implementation of our work in Yang et al. [205, 210]. As investigated, performing online estimation of IMU intrinsics typically isn't recommended due to the large amount of possible degenerate motions which can cause poor calibration and thus trajectory estimation. The details of the measurement Jacobians required for online calibration are detailed in Appendix A.

### 2.3.5 Accurate Groundtruth for Evaluation

We have created a groundtruthing pipeline, termed `vicon2gt`, which enables the fusion of inertial measurements and an external mocap system (e.g., an OptiTrack or Vicon) to generate temporally and spatially aligned high fidelity groundtruth trajectories. While a mocap system can directly provide 6 DoF poses, e.g. a frame $\{B\}$, to

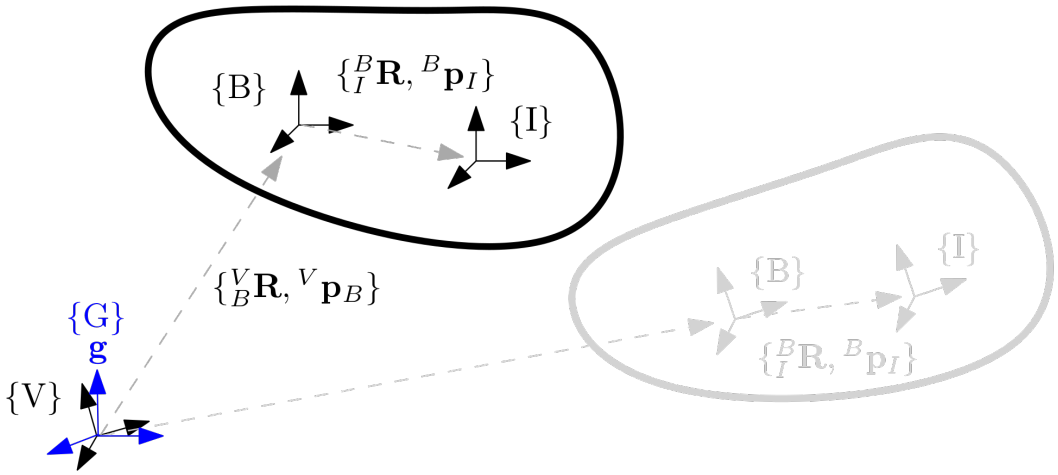



Figure 2.1: Sensor frames in the system. The motion capture frame $\{V\}$ which poses are captured in and is not gravity aligned along with the motion capture marker body frame $\{B\}$ and inertial IMU frame $\{I\}$ can be seen. Also seen is the gravity vector $\mathbf{g}$ which is perfectly along the z-axis in the global inertial frame $\{G\}$.

mocap markers placed on a robotic platform, the rigid extrinsic and temporal calibration between the mocap markers and the IMU sensor frame are unknown. Additionally, there can be an unknown global transformation between the VIO global frame $\{G\}$ which starts from zero yaw and position, and the arbitrarily placed global frame of the motion capture system $\{V\}$. See Figure 2.1 for an overview of the problem and the unknown transforms we need to be able to recover concurrently as we estimate the trajectory of the IMU. Specifically, we estimate the following states:

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_{I_0}^\top & \cdots & \mathbf{x}_{I_N}^\top & \mathbf{x}_J^\top & \mathbf{x}_G^\top \end{bmatrix}^\top \tag{2.56}$$

$$\mathbf{x}_{I_i} = \begin{bmatrix} {}^{I_i}_{V}\bar{q}^\top & {}^V\mathbf{p}_{I_i}^\top {}^V\mathbf{v}_{I_i}^\top & \mathbf{b}_{g,i}^\top & \mathbf{b}_{a,i}^\top \end{bmatrix}^\top \tag{2.57}$$

$$\mathbf{x}_J = \begin{bmatrix} {}^{I}_{B}\bar{q}^\top & {}^I\mathbf{p}_B^\top & {}^V t_I \end{bmatrix}^\top \tag{2.58}$$

$$\mathbf{x}_G = \begin{bmatrix} {}^{V}_{G}\theta_x & {}^{V}_{G}\theta_y \end{bmatrix}^\top \tag{2.59}$$

where we are estimating $N$ inertial states at an arbitrary frequency, along with a calibration state $\mathbf{x}_J$ containing the spatial-temporal parameters between the motion capture and IMU sensors, and the rotation between the motion capture frame and global inertial frame. ${}^{I_k}_{V}\bar{q}$ is the unit quaternion parameterizing the rotation $\mathbf{R}({}^{I_k}_{V}\bar{q}) =$

${}^{I_k}_{V}\mathbf{R}$ from the global frame of reference $\{V\}$ to the IMU local frame $\{I_k\}$ at time $t_k$ [186], and ${}^{V}\mathbf{v}_{I_k}$ and ${}^{V}\mathbf{p}_{I_k}$ are the velocity and position of the IMU expressed in the global frame, respectively. It is important to note here that we are estimating the IMU states at the *true* IMU clock time, meaning that the states that occur at time $t_k$ are in the IMU clock frame, ${}^{I}t_k$, and can be related a time in motion capture clock by:

$$ {}^{I}t = {}^{V}t + {}^{V}t_I \tag{2.60} $$

We additionally define our rotation in $\mathbf{x}_G$ from the gravity aligned inertial frame to the motion capture frame using the following roll-pitch rotation:

$$ {}^{V}_{G}\mathbf{R} = \mathbf{R}_y({}^{V}_{G}\theta_y)\mathbf{R}_x({}^{V}_{G}\theta_x) \tag{2.61} $$

$$ = \begin{bmatrix} \cos {}^{V}_{G}\theta_y & 0 & \sin {}^{V}_{G}\theta_y \\ 0 & 1 & 0 \\ -\sin {}^{V}_{G}\theta_y & 0 & \cos {}^{V}_{G}\theta_y \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos {}^{V}_{G}\theta_x & -\sin {}^{V}_{G}\theta_x \\ 0 & \sin {}^{V}_{G}\theta_x & \cos {}^{V}_{G}\theta_x \end{bmatrix} \tag{2.62} $$

Note that here we are fixing the yaw to be zero since the gravity aligned frame has arbitrary yaw and thus this rotation is only 2 DoF.

#### 2.3.5.1 Inertial Preintegrated Constraints

Shown in the factor graph in Figure 2.3, each state is constrained through propagation of the IMU using closed-form preintegration [40]. We can define the following integrations in the mocap frame of reference:

$$ {}^{k+1}_{V}\mathbf{R} = \Delta\mathbf{R}_{k-1,k}\ {}^{k}_{V}\mathbf{R} \tag{2.63} $$

$$ {}^{V}\mathbf{p}_{k+1} = {}^{V}\mathbf{p}_k + {}^{V}\mathbf{v}_k\Delta T - \frac{1}{2}{}^{V}_{G}\mathbf{R}\,{}^{G}\mathbf{g}\Delta T^2 + {}^{V}_{k}\mathbf{R}\Delta\mathbf{p}_{k-1,k} \tag{2.64} $$

$$ {}^{V}\mathbf{v}_{k+1} = {}^{V}\mathbf{v}_k - {}^{V}_{G}\mathbf{R}\,{}^{G}\mathbf{g}\Delta T + {}^{V}_{k}\mathbf{R}\Delta\mathbf{v}_{k-1,k} \tag{2.65} $$

During preintegration, the terms $\Delta\mathbf{R}_{k-1,k}$, $\Delta\mathbf{p}_{k-1,k}$, and $\Delta\mathbf{v}_{k-1,k}$ are only computed *once*, e.g. pre-integrated, and linearized at the current bias estimate $\bar{\mathbf{b}}_w$ and $\bar{\mathbf{b}}_a$. This

results in the following measurement residual after the bias relinearization

$$r_I(\mathbf{x}) = \begin{bmatrix} 2\text{vec}\left({}^{k+1}_{V}\bar{q} \otimes {}^{k}_{V}\bar{q}^{-1} \otimes {}^{k+1}_{k}\breve{q}^{-1} \otimes \bar{q}(\Delta\mathbf{b}_w)\right) \\ \begin{pmatrix} {}^{k}_{V}\mathbf{R}\left({}^{V}\mathbf{p}_{k+1} - {}^{V}\mathbf{p}_k - {}^{V}\mathbf{v}_k\Delta T + \frac{1}{2}{}^{V}_{G}\mathbf{R}\,{}^{G}\mathbf{g}\Delta T^2\right) \\ -\Delta\mathbf{p}_{k,k+1} - \frac{\partial\Delta\mathbf{p}}{\partial\mathbf{b}_a}\Big|_{\bar{\mathbf{b}}_a}\Delta\mathbf{b}_a - \frac{\partial\Delta\mathbf{p}}{\partial\mathbf{b}_w}\Big|_{\bar{\mathbf{b}}_w}\Delta\mathbf{b}_w \end{pmatrix} \\ \begin{pmatrix} {}^{k}_{V}\mathbf{R}\left({}^{V}\mathbf{v}_{k+1} - {}^{V}\mathbf{v}_k + {}^{V}_{G}\mathbf{R}\,{}^{G}\mathbf{g}\Delta T\right) \\ -\Delta\mathbf{v}_{k,k+1} - \frac{\partial\Delta\mathbf{v}}{\partial\mathbf{b}_a}\Big|_{\bar{\mathbf{b}}_a}\Delta\mathbf{b}_a - \frac{\partial\Delta\mathbf{v}}{\partial\mathbf{b}_w}\Big|_{\bar{\mathbf{b}}_w}\Delta\mathbf{b}_w \end{pmatrix} \\ \mathbf{b}_{w,k+1} - \mathbf{b}_{w,k} \\ \mathbf{b}_{a,k+1} - \mathbf{b}_{a,k} \end{bmatrix}$$

where $\Delta\mathbf{b}_w := \mathbf{b}_{w(k)} - \bar{\mathbf{b}}_w$ and $\Delta\mathbf{b}_a := \mathbf{b}_{a(k)} - \bar{\mathbf{b}}_a$ are the differences between the true biases and the current bias estimate used as the linearization point. We have also dropped the sensor frame $\{I\}$ have only specified the time index and used the quaternion orientation notation.

We can now linearize this non-linear measurement constraint and recover Jacobians needed for graph optimization. We refer the reader to the continuous preintegration technical report for the preintegration Jacobians [39]. There is an additional Jacobian which is needed since this measurement is now a function of the two angles which rotate the gravity vector into the motion capture frame. This can be found by directly taking the derivative in respect to the rotation matrix:

$$\frac{\partial r_I(\mathbf{x})}{\partial[{}^{V}_{G}\theta_x \;\; {}^{V}_{G}\theta_y]} = \begin{bmatrix} \mathbf{0}_{2\times3} & \left(\frac{1}{2}{}^{k}_{V}\mathbf{R}\Delta T^2\mathbf{H}_z\right)^\top & \left({}^{k}_{V}\mathbf{R}\Delta T\mathbf{H}_z\right)^\top & \mathbf{0}_{2\times3} & \mathbf{0}_{2\times3} \end{bmatrix}^\top \tag{2.66}$$

where we define the derivative in respect to ${}^{V}_{G}\mathbf{R}\,{}^{G}\mathbf{g}$ as (see Eq. (2.62)):

$$\mathbf{H}_z = 9.81 \begin{bmatrix} -\sin{}^{V}_{G}\theta_y \sin{}^{V}_{G}\theta_x & \cos{}^{V}_{G}\theta_y \cos{}^{V}_{G}\theta_x \\ -\cos{}^{V}_{G}\theta_x & 0 \\ -\cos{}^{V}_{G}\theta_y \sin{}^{V}_{G}\theta_x & -\sin{}^{V}_{G}\theta_y \cos{}^{V}_{G}\theta_x \end{bmatrix} \tag{2.67}$$

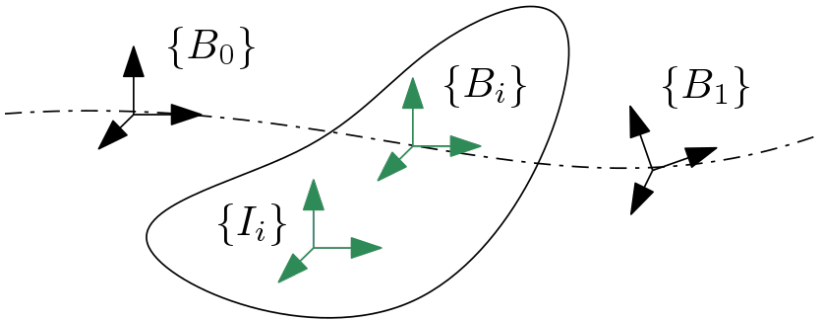



Figure 2.2: Example interpolation problem where two bounding motion capture poses $\{B_0\}$ and $\{B_1\}$. The motion capture pose is first interpolated to the pose time creating frame $\{B_i\}$, then the rigid extrinsic transformation can transform it into the IMU sensor frame $\{I_i\}$.

#### 2.3.5.2 Asynchronous 6 DoF Pose Factor

Our mocap provides pose estimates at an arbitrary rate which can be different than the groundtruth rate we wish to estimate and recover. Thus, we consider it as a sensor that is operating asynchronously from our states, and we need to define how we can relate it to the states we are estimating. To do so, we propose an efficient and novel method to synchronous the poses to the ones estimated in the state without requiring any additional states to be estimated which is key to reducing the $O(n^3)$ cost of graph optimization. The key idea, shown in Figure 2.2, is that we can take two mocap poses which bound a pose we are estimating and can interpolate between these two to recover the pose at the *exact* same time as the state that we are estimating. More general versions of this problem are presented in our early conference publication on asynchronous sensor fusion [57]. We show that this can be done in a rigorous manner and that both the uncertainty of the interpolated pose and spatiotemporal calibration between the two sensors can be properly estimated.

As shown in Figure 2.2, we construct a measurement by calculating an artificial pose $\{B_i\}$ which should occur at the state time. This can then be related to our state through the extrinsic transformation between the motion capture marker frame and the IMU sensor frame. This means that this measurement function is only a function of the state it is interpolated to and the calibration. We are estimating the state $\mathbf{x}_{I_i}$ which has occurred at time ${}^{I}t_i$ in the IMU clock frame. We can first calculate

a synthetic measurement which is of the motion capture marker frame at this time instance through the following:

$$\{{}^{B_i}_{V}\breve{\mathbf{R}}, {}^{V}\breve{\mathbf{p}}_{B_i}\} = \mathbf{g}\Big(\{{}^{B_0}_{V}\mathbf{R}, {}^{V}\mathbf{p}_{B_0}, \mathbf{P}_0\}, \{{}^{B_1}_{V}\mathbf{R}, {}^{V}\mathbf{p}_{B_1}, \mathbf{P}_1\}, {}^{V}t_I\Big) \tag{2.68}$$

This function $\mathbf{g}(\cdot)$ is not yet our measurement function to our state, but instead is transforming two mocap measurements into a new measurement at the desired state time in the IMU clock frame. Specifically, it is defined as:

$${}^{B_i}_{V}\breve{\mathbf{R}} = \text{Exp}\left(\lambda\ \text{Log}({}^{B_1}_{V}\mathbf{R}\,{}^{B_0}_{V}\mathbf{R}^{\top})\right){}^{B_0}_{V}\mathbf{R} \tag{2.69}$$

$${}^{V}\breve{\mathbf{p}}_{B_i} = (1-\lambda){}^{V}\mathbf{p}_{B_0} + \lambda {}^{V}\mathbf{p}_{B_1} \tag{2.70}$$

$$\lambda = \frac{({}^{V}t_i - {}^{V}t_{B_0})}{({}^{V}t_{B_1} - {}^{V}t_{B_0})} = \frac{({}^{I}t_i - {}^{V}t_I - {}^{V}t_{B_0})}{({}^{V}t_{B_1} - {}^{V}t_{B_0})} \tag{2.71}$$

where we have the bounding poses $\{B_0\}$ and $\{B_1\}$ which were collected at time ${}^{V}t_{B_0}$ and ${}^{V}t_{B_1}$ in the motion capture clock frame, and $\text{Log}(\cdot)$ is the $\mathbb{SO}(3)$ matrix logarithm [23]. We wish to interpolate to the state time ${}^{I}t_i$, thus we calculate the time in the motion capture clock as ${}^{V}t_i = {}^{I}t_i - {}^{V}t_I$. The measurement covariance is propagated through the following covariance propagation:

$$\boldsymbol{P}_i = \boldsymbol{H}_u \boldsymbol{P}_{1,2} \boldsymbol{H}_u{}^{\top} \tag{2.72}$$

where $\boldsymbol{P}_{1,2}$ is the joint covariance matrix from the bounding poses, and $\tilde{\boldsymbol{\theta}}$ and $\tilde{\boldsymbol{p}}$ are the error states of each angle and position measurement, respectively. The full covariance propagation is detailed in Appendix B.

Having calculated now the measurement $\{{}^{B_i}_{V}\breve{\mathbf{R}}, {}^{V}\breve{\mathbf{p}}_{B_i}\}$ and its measurement noise $\boldsymbol{P}_i$, we can formulate the measurement function which relates this measurement to our state estimates through our spatial calibration parameters:

$$\{{}^{B_i}_{V}\mathbf{R}, {}^{V}\mathbf{p}_{B_i}\} = \mathbf{h}\Big(\{{}^{I_i}_{V}\mathbf{R}, {}^{V}\mathbf{p}_{I_i}\}, \{{}^{I}_{B}\mathbf{R}, {}^{I}\mathbf{p}_{B}\}\Big) + \mathbf{n}_{pose} \tag{2.73}$$

$${}^{B_i}_{V}\mathbf{R} = {}^{I}_{B}\mathbf{R}^{\top}\,{}^{I_i}_{V}\mathbf{R} + \mathbf{n}_{ori} \tag{2.74}$$

$${}^{V}\mathbf{p}_{B_i} = {}^{V}\mathbf{p}_{I_i} + {}^{I_i}_{V}\mathbf{R}^{\top I}\mathbf{p}_B + \mathbf{n}_{pos} \tag{2.75}$$

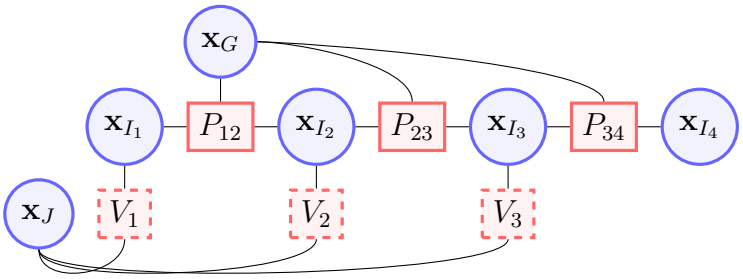



Figure 2.3: Example of a factor graph that our system created. States that will be estimated are denoted in circles and measurements are denoted in squares. Note that we differentiate asynchronous factors with dashed outlines.

From which we have the following residual:

$$r_V(\mathbf{x}) = \begin{bmatrix} \text{Log}\left({}^{I}_{B}\mathbf{R}\,{}^{B_i}_{V}\mathbf{R}\,{}^{B_i}_{V}\breve{\mathbf{R}}^{\top}\right) \\ {}^{V}\mathbf{p}_{I_i} + {}^{I_i}_{V}\mathbf{R}^{\top}\,{}^{I}\mathbf{p}_B - {}^{V}\breve{\mathbf{p}}_{B_i} \end{bmatrix} + \mathbf{n}_{pose} \tag{2.76}$$

The linearized measurement model is defined in Appendix B.

#### 2.3.5.3 Non-linear Problem Formulation

As shown in Figure 2.3, we have two key factors: (i) inertial preintegration between sequential states and (ii) interpolated pose factors. The specific optimization problem we optimize via the GTSAM [28] library is as follows:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\text{argmin}} \left[ \sum_{N-1} \left|\left|\mathbf{r}_{Ik}\left(\mathbf{x}_{I_{k-1}}, \mathbf{x}_{I_k}, \mathbf{x}_G\right)\right|\right|^2_{\mathbf{P}_{k-1,k}} + \sum_{N} \left|\left|\mathbf{r}_{Vk}\left(\mathbf{x}_{I_k}, \mathbf{x}_J\right)\right|\right|^2_{\mathbf{P}_i} \right] \tag{2.77}$$

#### 2.3.5.4 Experimental Validation of Groundtruth

We first validate in simulation using our simulator, see Section 2.3.7, where a 200 Hz IMU, 20 Hz camera, and 100 Hz 6 DoF motion capture system was simulated. For each simulation, the system started with identity spatial transforms and a time offset value of zero. A random motion capture to IMU to marker body orientation $\sigma = 0.1$ and position $\sigma = 0.2$, and time offset $\sigma = 0.05$ (we used the same distributions for all 3 vector dimensions) is generated for each Monte-Carlo run. We additionally select a random ${}^{V}_{G}\mathbf{R}$ with $\sigma = 0.1\pi \approx 10^{\circ}$ and simulate the system such that the

Table 2.1: Average absolute trajectory error (degrees / meters) over 10 Monte-Carlo runs. Each column is the motion capture orientation noises (deg), while each row is a different position motion capture noise (m).

| | **0.057** | **0.286** | **0.573** | **2.864** | **5.730** |
|---|---|---|---|---|---|
| **0.001** | 0.020 / 0.001 | 0.106 / 0.007 | 0.300 / 0.021 | 0.464 / 0.045 | 0.441 / 0.045 |
| **0.005** | 0.037 / 0.002 | 0.157 / 0.009 | 0.318 / 0.020 | 0.466 / 0.045 | 0.441 / 0.045 |
| **0.010** | 0.115 / 0.006 | 0.211 / 0.012 | 0.348 / 0.024 | 0.462 / 0.045 | 0.440 / 0.045 |
| **0.050** | 0.051 / 0.012 | 0.363 / 0.024 | 0.409 / 0.029 | 0.409 / 0.039 | 0.421 / 0.043 |
| **0.100** | 0.050 / 0.022 | 0.371 / 0.029 | 0.424 / 0.032 | 0.362 / 0.035 | 0.391 / 0.040 |

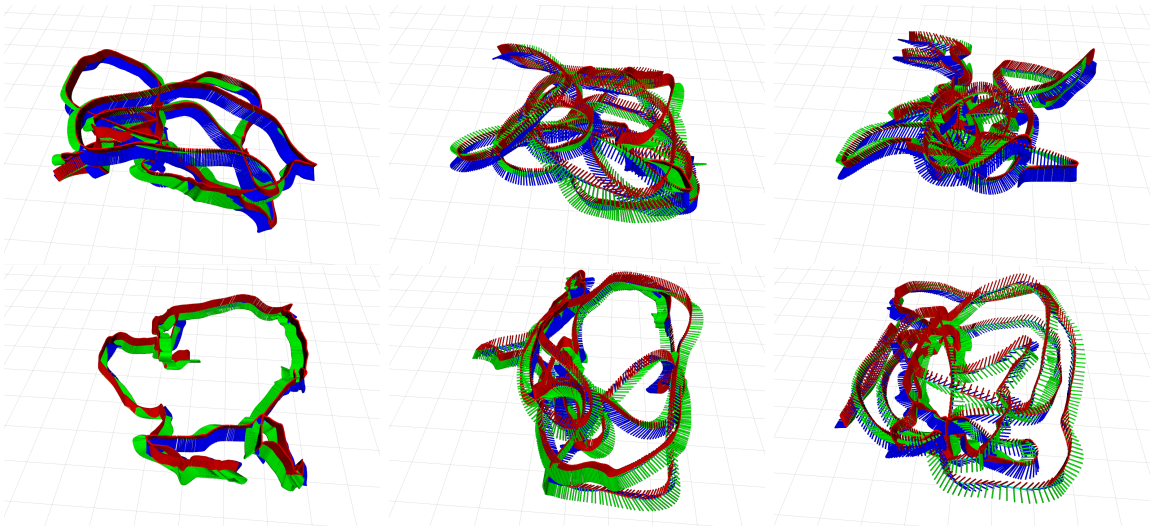


Figure 2.4: Example generated trajectories on the EuRoC MAV dataset.

Table 2.2: Average absolute trajectory error (degrees / meters) compared to the provided groundtruth of the EuRoC MAV dataset. Position and yaw alignment were performed.

| | **V1_01** | **V1_02** | **V1_03** | **V2_01** | **V2_02** | **V2_03** |
|---|---|---|---|---|---|---|
| Ori. (deg) | 5.789 | 1.969 | 2.269 | 5.779 | 0.853 | 0.791 |
| Std. Ori. (deg) | 0.161 | 0.170 | 0.144 | 0.978 | 0.406 | 0.279 |
| Pos. (m) | **0.036** | 0.009 | 0.006 | **0.068** | 0.015 | 0.018 |
| Std. Pos. (m) | 0.013 | 0.004 | 0.003 | 0.022 | 0.007 | 0.009 |

B-spline trajectory is in the fixed inertial frame (thus all groundtruths from multiple different motion capture frame transforms are the same). As per standard practice, the inertial measurement readings were corrupted using the random walk biases and corresponding white noises, while the motion capture poses were corrupted using an additive white noise. We observed that there was typically a small position offset between the groundtruth and optimized trajectory caused by errors in the marker to IMU transformation which can be due to the trajectory not fully exciting all axes enough to make these parameters observable. Thus as compared to not performing alignment, we perform position and yaw alignment between the simulated groundtruth and optimized trajectories.

Shown in Table 2.1, we can see that as both orientation and position errors increase, the orientation error plateaus at around half a degree even under 6 degrees, and 10cm position error. Position errors increase as motion capture measurements become noisier, but impressively never go above 5cm event with 10cm noise levels. This confirms that the system can reconstruct the trajectory in high noise level cases where the noise of the motion capture system and IMU sensor are known. Depending on the level of accuracy required, being conservative in the estimates might prove useful to ensure proper recovery.

Some example recovered trajectories from optimizing the IMU and raw motion capture messages in the EuRoC MAV datasets [14] bag files can be seen in Figure 2.4. Shown in Table 2.2, we compare the absolute trajectory error after position and yaw alignment of the generated groundtruth to that provided by the dataset authors. This dataset provided groundtruth has been time aligned to take into account the motion capture to IMU clocks and optimized the IMU sensor frame, temporal time offset, and spacial transform between the groundtruth and inertial frame. It is interesting that the V1_01 and V2_01 runs have large position and orientation errors suggesting that likely the dataset's groundtruth might have issues. We have found that for the V1_01 dataset, the orientation and bias estimates of the groundtruth are poor when compared to the output of a visual-inertial estimator (relative to the other datasets).

### 2.3.6 State Initialization

Next, we overview how we initialize our estimator. Since we are using a filter-based recursive estimator it is key to have a sufficiently accurate initial guess of the state parameters $\mathbf{x}_0$ to kickstart the estimator and prevent divergence.

#### 2.3.6.1 Static State Initialization

The simplest method requires that the camera-IMU pair are stationary, and thus we know that ${}^{G}\mathbf{v}_{I_0} = \mathbf{0}$. We can recover the gravity direction and biases via:

$$ {}^{I_0}\mathbf{a}^* = \frac{1}{N}\sum {}^{I}\mathbf{a}(t_i) \tag{2.78} $$

$$ {}^{I_0}\mathbf{g} = 9.81\ ({}^{I}\mathbf{a}^*/||{}^{I}\mathbf{a}^*||) \tag{2.79} $$

$$ {}^{I_0}\mathbf{b}_a = {}^{I_0}\mathbf{a}^* - {}^{I_0}\mathbf{g} \tag{2.80} $$

$$ {}^{I_0}\mathbf{b}_g = \frac{1}{N}\sum {}^{I}\boldsymbol{\omega}(t_i) \tag{2.81} $$

where we have used a known gravity magnitude, and can use ${}^{I_0}\mathbf{g}$ to recover the initial roll and pitch of ${}^{I_0}_{G}\mathbf{R}$ and arbitrarily assign the yaw along with global position to zero.

#### 2.3.6.2 Dynamic State Initialization

More interestingly, we wish to recover the initial state when we have non-zero velocity, which given image measurements should be observable. One of the key assumptions is that we have reasonably accurate biases which will allow us to integrate the IMU over the short initialization window (typically 1-2 seconds). As we will show, this is crucial as if we know the relative orientation, a linear problem which allows for direct recovery of the velocity and gravity direction can be created.

*Relative Inertial Integration:* The minimal state we wish to recover is [32, 33]:

$$ \mathbf{x} = \begin{bmatrix} {}^{I_0}\mathbf{p}_{f_1}^\top & \dots & {}^{I_0}\mathbf{p}_{f_M}^\top & {}^{I_0}\mathbf{v}_{I_0}^\top & {}^{I_0}\mathbf{g}^\top \end{bmatrix}^\top \tag{2.82} $$

Integrating Eq. (2.15)-(2.19) from $t_0$ to $t_k$ in the first frame IMU $\{I_0\}$ frame of reference will result in the following:

$$ {}^{I_k}_{I_0}\mathbf{R} \triangleq \Delta\mathbf{R}_{0,k} \tag{2.83} $$

$$
{}^{I_0}\mathbf{p}_{I_k} \triangleq {}^{I_0}\mathbf{v}_{I_0}\Delta T_k - \frac{1}{2}{}^{I_0}\mathbf{g}\Delta T_k^2 + \Delta\mathbf{p}_{0,k} \tag{2.84}
$$

$$
{}^{I_0}\mathbf{v}_{I_k} \triangleq {}^{I_0}\mathbf{v}_{I_0} - {}^{I_0}\mathbf{g}\Delta T_k + \Delta\mathbf{v}_{0,k} \tag{2.85}
$$

where $\Delta T_k = (t_k - t_0)$ is the time span for integration. These can be found by rotating the orientation and velocity with ${}^{I_0}_{G}\mathbf{R}$ and computing the relative position change ${}^{I_0}\mathbf{p}_{I_k}$ to define a *relative* IMU integration in the fixed $\{I_0\}$ frame:

$$
{}^{I_0}\mathbf{p}_{I_{k+1}} = {}^{I_0}_{G}\mathbf{R}\left({}^{G}\mathbf{p}_{I_{k+1}} - {}^{G}\mathbf{p}_{I_0}\right) \tag{2.86}
$$

$$
= {}^{I_0}_{G}\mathbf{R}\left({}^{G}\mathbf{p}_{I_k} + {}^{G}\mathbf{v}_{I_k}\Delta t_k - \frac{1}{2}{}^{G}\mathbf{g}\Delta T^2 + {}^{I_k}_{G}\mathbf{R}^\top\Delta\mathbf{p}_{k-1,k} - {}^{G}\mathbf{p}_{I_0}\right) \tag{2.87}
$$

$$
= {}^{I_0}_{G}\mathbf{R}({}^{G}\mathbf{p}_{I_k} - {}^{G}\mathbf{p}_{I_0}) + {}^{I_0}_{G}\mathbf{R}{}^{G}\mathbf{v}_{I_k}\Delta t_k - \frac{1}{2}{}^{I_0}_{G}\mathbf{R}{}^{G}\mathbf{g}\Delta T^2 + {}^{I_0}_{G}\mathbf{R}{}^{I_k}_{G}\mathbf{R}^\top\Delta\mathbf{p}_{k-1,k} \tag{2.88}
$$

$$
= {}^{I_0}\mathbf{p}_{I_k} + {}^{I_0}\mathbf{v}_{I_k}\Delta t_k - \frac{1}{2}{}^{I_0}\mathbf{g}\Delta {t_k}^2 + {}^{I_k}_{I_0}\mathbf{R}^\top\Delta\mathbf{p}_{k-1,k} \tag{2.89}
$$

If we then integrate from time $t_0$ we can arrive at our relative integration equation:

$$
{}^{I_0}\mathbf{p}_{I_{k+1}} = {}^{I_0}\mathbf{p}_{I_0} + {}^{I_0}\mathbf{v}_{I_0}\Delta T - \frac{1}{2}{}^{I_0}\mathbf{g}\Delta T^2 + {}^{I_0}_{I_0}\mathbf{R}^\top\mathbf{p}_{0,k} \tag{2.90}
$$

$$
\triangleq {}^{I_0}\mathbf{v}_{I_0}\Delta T - \frac{1}{2}{}^{I_0}\mathbf{g}\Delta T^2 + \Delta\mathbf{p}_{0,k} \tag{2.91}
$$

where $\Delta T_k = (t_k - t_0)$ is the time span for integration, and we have used that ${}^{I_0}\mathbf{p}_{I_0} = \mathbf{0}$ and ${}^{I_0}_{I_0}\mathbf{R} = \mathbf{I}$.

*Linear Normalized Bearing Observation:* Assuming a calibrated perspective camera, the bearing measurement of the $i$th feature at timestep $t_k$ can be related to the state by the following:

$$
\mathbf{z}_{i,k} := \mathbf{\Lambda}({}^{C_k}\mathbf{p}_{f_i}) + \mathbf{n}_i \tag{2.92}
$$

$$
{}^{C_k}\mathbf{p}_{f_i} = {}^{C}_{I}\mathbf{R}{}^{I_k}_{I_0}\mathbf{R}({}^{I_0}\mathbf{p}_{f_i} - {}^{I_0}\mathbf{p}_{I_k}) + {}^{C}\mathbf{p}_I \tag{2.93}
$$

where $\mathbf{\Lambda}([x\ y\ z]^\top) = [x/z\ y/z]^\top$ is the camera perspective projection model, $\mathbf{z}_{i,k} = [u_{i,k}, v_{i,k}]^\top$ is the normalized feature bearing measurement computed through the inverse of Eq. (2.47), along with its white Gaussian noise $\mathbf{n}_i \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_i)$, and $\{{}^{C}_{I}\mathbf{R}, {}^{C}\mathbf{p}_I\}$

are the known camera-IMU transformation. This perspective projection can be re-written as the following linear constraint [33]:

$$\begin{bmatrix} 1 & 0 & -u_{i,k} \\ 0 & 1 & -v_{i,k} \end{bmatrix} {}^{C_k}\mathbf{p}_{f_i} \triangleq \mathbf{\Gamma}_{i,k} {}^{C_k}\mathbf{p}_{f_i} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \tag{2.94}$$

*Linear Visual-Inertial System:* We can then substitute Eq. (2.84) and (2.93) to give:

$$\mathbf{\Gamma}_{i,k} \left( {}^{C}_{I}\mathbf{R}^{I_k}_{I_0}\mathbf{R} \left( {}^{I_0}\mathbf{p}_{f_i} - {}^{I_0}\mathbf{p}_{I_k} \right) + {}^{C}\mathbf{p}_{I} \right) = \mathbf{0}_2 \text{ Eq. (2.93)} \tag{2.95}$$

$$\mathbf{\Gamma}_{i,k} \left( {}^{C}_{I}\mathbf{R}^{I_k}_{I_0}\Delta\mathbf{R} \left( {}^{I_0}\mathbf{p}_{f_i} - {}^{I_0}\mathbf{v}_{I_0}\Delta T + \frac{1}{2}{}^{I_0}\mathbf{g}\Delta T^2 - \Delta\mathbf{p}_{0,k} \right) + {}^{C}\mathbf{p}_{I} \right) = \mathbf{0}_2 \text{ Eq. (2.84)} \tag{2.96}$$

$$\underbrace{\mathbf{\Gamma}_{i,k} \, {}^{C}_{I}\mathbf{R}^{I_k}_{I_0}\Delta\mathbf{R}}_{\mathbf{\Upsilon}_{i,k}} \left( {}^{I_0}\mathbf{p}_{f_i} - {}^{I_0}\mathbf{v}_{I_0}\Delta T + \frac{1}{2}{}^{I_0}\mathbf{g}\Delta T^2 \right) = \left( \mathbf{\Gamma}_{i,k} \, {}^{C}_{I}\mathbf{R}^{I_k}_{I_0}\Delta\mathbf{R}\Delta\mathbf{p}_{0,k} - \mathbf{\Gamma}_{i,k} \, {}^{C}\mathbf{p}_{I} \right) \tag{2.97}$$

For a single integration period (e.g. time $t_0$ to $t_k$) we will have the following linear system for our $M$ environmental features:

$$\underbrace{\begin{bmatrix} \mathbf{\Upsilon}_{0,k} & \mathbf{0} & \mathbf{0}_3 & -\mathbf{\Upsilon}_{0,k}\Delta T & \frac{1}{2}\mathbf{\Upsilon}_{0,k}\Delta T^2 \\ \mathbf{0} & \ddots & \mathbf{0} & \vdots & \vdots \\ \mathbf{0}_3 & \mathbf{0} & \mathbf{\Upsilon}_{M,k} & -\mathbf{\Upsilon}_{M,k}\Delta T & \frac{1}{2}\mathbf{\Upsilon}_{M,k}\Delta T^2 \end{bmatrix}}_{\mathbf{A}_k} \begin{bmatrix} {}^{I_0}\mathbf{p}_{f_0} \\ \vdots \\ {}^{I_0}\mathbf{p}_{f_M} \\ {}^{I_0}\mathbf{v}_{I_0} \\ {}^{I_0}\mathbf{g} \end{bmatrix} = \underbrace{\begin{bmatrix} \left( \mathbf{\Upsilon}_{0,k}\Delta\mathbf{p}_{0,k} - \mathbf{\Gamma}_{0,k} \, {}^{C}\mathbf{p}_{I} \right) \\ \vdots \\ \left( \mathbf{\Upsilon}_{M,k}\Delta\mathbf{p}_{M,k} - \mathbf{\Gamma}_{M,k} \, {}^{C}\mathbf{p}_{I} \right) \end{bmatrix}}_{\mathbf{b}_k}$$

This system can then be created for each sequential frame resulting in a stacked system. This system can be solved through a constrained linear least-squares which can remove the need to recover the scale of ${}^{I_0}\mathbf{g}$. Specifically we solve the following using Lagrange multiple method [32, 33]:

$$\text{minimize } \|\mathbf{A}\mathbf{x} - \mathbf{b}\|_2 = \left\| \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \end{bmatrix} \begin{bmatrix} \mathbf{x}_1 \\ {}^{I_0}\mathbf{g} \end{bmatrix} - \mathbf{b} \right\|_2 \tag{2.98}$$

$$\text{subject to } \left\| {}^{I_0}\mathbf{g} \right\|_2 = g = 9.81 \tag{2.99}$$

*Velocity Refinement and Covariance Recovery:* After solving the linear system we then wish to recover the initial inertial state $\mathbf{x}_I$ for use in estimation. Before doing so we will recover each frame's pose and velocity and perform a non-linear refinement to increase the accuracy of the initial velocity along with enabling the recovery of the covariance through inversion of the information matrix. We use inertial propagation we can recover the inertial state at each time as:

$$\begin{bmatrix} {}^{I_k}_{I_0}\mathbf{R} \\ {}^{I_0}\mathbf{p}_{I_k} \\ {}^{I_0}\mathbf{v}_{I_k} \end{bmatrix} = \begin{bmatrix} {}^{I_{k+1}}_{I_0}\Delta\mathbf{R} \\ {}^{I_0}\mathbf{v}_{I_0}\Delta T - \frac{1}{2}{}^{I_0}\mathbf{g}\Delta T^2 + {}^{0}\boldsymbol{\alpha}_{k+1} \\ {}^{I_0}\mathbf{v}_{I_0} - {}^{I_0}\mathbf{g}\Delta T + {}^{0}\boldsymbol{\beta}_{k+1} \end{bmatrix} \tag{2.100}$$

Where ${}^{I_0}\mathbf{g}$ and ${}^{I_0}\mathbf{v}_{I_0}$ have been recovered from our constrained least-squares and $\Delta T$ is from time $t_0$ to $t_k$. These state estimates can then be rotated into a gravity aligned frame after recovering the roll and pitch from ${}^{I_0}\mathbf{g}$ and setting the yaw to be zero to get ${}^{I_0}_{G}\mathbf{R}$. This can then be applied to transform such that gravity is now ${}^{G}\mathbf{g} = [0\ 0\ 9.81]^\top$:

$$\begin{bmatrix} {}^{I_k}_{G}\mathbf{R} \\ {}^{G}\mathbf{p}_{I_k} \\ {}^{G}\mathbf{v}_{I_k} \end{bmatrix} = \begin{bmatrix} {}^{I_k}_{I_0}\mathbf{R}\ {}^{I_0}_{G}\mathbf{R} \\ {}^{I_0}_{G}\mathbf{R}^\top {}^{I_0}\mathbf{p}_{I_k} \\ {}^{I_0}_{G}\mathbf{R}^\top {}^{I_0}\mathbf{v}_{I_k} \end{bmatrix} \tag{2.101}$$

$${}^{G}\mathbf{p}_{f_i} = {}^{I_0}_{G}\mathbf{R}^\top {}^{I_0}\mathbf{p}_{f_i}\ \forall i \tag{2.102}$$

These initial states are then refined through non-linear refinement optimization:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left[ \sum_N \left|\left|\mathbf{r}_{Ik}\left(\mathbf{x}_{I_{k-1}}, \mathbf{x}_{I_k}\right)\right|\right|^2_{\mathbf{P}_{k-1,k}} + \sum_M ||\mathbf{r}_{fj}(\mathbf{x})||^2_{\mathbf{R}_i} + ||\mathbf{x}_0 \boxminus \mathbf{x}_{lin}||^2_{\mathbf{P}_0} \right] \tag{2.103}$$

where $\mathbf{r}_{Ik}$ is a preintegrated inertial factor [39, 40] from time $t_{k-1}$ to $t_k$, $\mathbf{r}_{fj}$ is the visual feature reprojection cost from Eq. (2.49), and the third term is a prior factor which constrains the 4 DoF unobservable directions with a very large prior along with the gyroscope and accelerometer biases which are assumed to have known accuracy.

### 2.3.7 Extendable Simulator with Continue-time Representation

At the center of the simulator is an $\mathbb{SE}(3)$ B-spline which allows for the calculation of the pose, velocity, and accelerations at any given timestep along a given

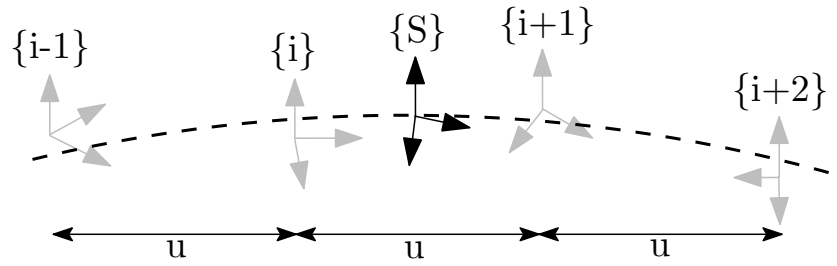



Figure 2.5: Illustration of the B-spline interpolation to a pose ${}^{G}_{I_S}\mathbf{T}$ which is bounded by four control poses which are separated by a constant time.

trajectory. We consider a series of temporally uniformly distributed "control point" poses, the pose $\{S\}$ at a given timestep $t_s$ can be interpolated by:[1]

$$
\begin{aligned}
{}^{G}_{I_s}\mathbf{T}(u(t_s)) &= \begin{bmatrix} {}^{G}_{I_s}\mathbf{R} & {}^{G}\mathbf{p}_{I_s} \\ \mathbf{0} & 1 \end{bmatrix} && (2.104)\\
&= {}^{G}_{i-1}\mathbf{T}\ \mathbf{A}_0\ \mathbf{A}_1\ \mathbf{A}_2 && (2.105)
\end{aligned}
$$

where we have the following intermediates:

$$
\begin{aligned}
\mathbf{A}_j &= \mathrm{Exp}\Big(B_j(u(t))\ {}^{i-1+j}_{i+j}\mathbf{\Omega}\Big) && (2.106)\\
{}^{i-1}_{i}\mathbf{\Omega} &= \mathrm{Log}\Big({}^{G}_{i-1}\mathbf{T}^{-1}\ {}^{G}_{i}\mathbf{T}\Big) && (2.107)\\
B_0(u(t)) &= \frac{1}{3!}\ (5 + 3u - 3u^2 + u^3) && (2.108)\\
B_1(u(t)) &= \frac{1}{3!}\ (1 + 3u + 3u^2 - 2u^3) && (2.109)\\
B_2(u(t)) &= \frac{1}{3!}\ (u^3) && (2.110)
\end{aligned}
$$

where $u(t_s) = (t_s - t_i)/(t_{i+1} - t_i)$ and $B_j(u(t))$ are our spline interpolation constants, and the frame notations are shown in Figure 2.5. Equation (2.105) can be interpreted as compounding the fraction portions of the bounding poses to the first $\mathbb{SE}(3)$ pose ${}^{G}_{i-1}\mathbf{T}$. The analytical derivative of this function can then be found to enable recovery of velocity and acceleration (see Appendix C)

[1] Some background on B-splines can be found in the works by Patron-Perez et al. [152], Mueggler et al. [142], and these notes [56].

The only needed input into the simulator is a pose trajectory which we uniformly sample to construct control points for the B-spline. To obtain the true measurements from our B-spline we can do:

$$ {}^{I}\boldsymbol{\omega}(t) = \text{Vee}\Big({}^{G}_{I}\mathbf{R}(u(t))^{\top}\,{}^{G}_{I}\dot{\mathbf{R}}(u(t))\Big) \tag{2.111} $$

$$ {}^{I}\mathbf{a}(t) = {}^{G}_{I}\mathbf{R}(u(t))^{\top}\,{}^{G}\ddot{\mathbf{p}}_{I}(u(t)) \tag{2.112} $$

where $\text{Vee}(\cdot)$ returns the vector portion of the skew-symmetric matrix. These are then corrupted using the random walk biases and corresponding white noises.

For feature observations, we first generate environmental landmarks along the trajectory at a fixed interval to ensure an average number of camera observations can be projected into synthetic camera frames during simulation. We generate landmarks' visual measurements by projecting them into the current pose provided by the spline, and then ensure that the 2D observations are within the field of view, in front, and close in distance to the camera. Pixel noise can be directly added to the true pixel values.

## 2.4 Visual-Inertial Odometry Simulation Results

With the proposed visual-inertial simulator, we evaluate the proposed OpenVINS with a 10Hz monocular camera and 400Hz IMU, a window size of 11, a maximum of 100 feature tracks per frame, and a maximum of 50 SLAM landmarks kept in the state, along with VIO feature tracks that are processed by the MSCKF update. We inject one pixel noise and the IMU noise characteristics of an ADIS16448 MEMS IMU. To simulate bad initial calibration values, we randomly initialize the calibration values using the prior distribution values of the estimator. This ensures that during the Monte-Carlo simulation we have both different measurement noises and initial calibration values for each run.

As summarized in Table 2.3, the average Absolute Trajectory Error (ATE) [221] and Normalized Estimation Error Squared (NEES) [4] for each different scenario shows that when performing online calibration, estimation accuracy does not degrade if we

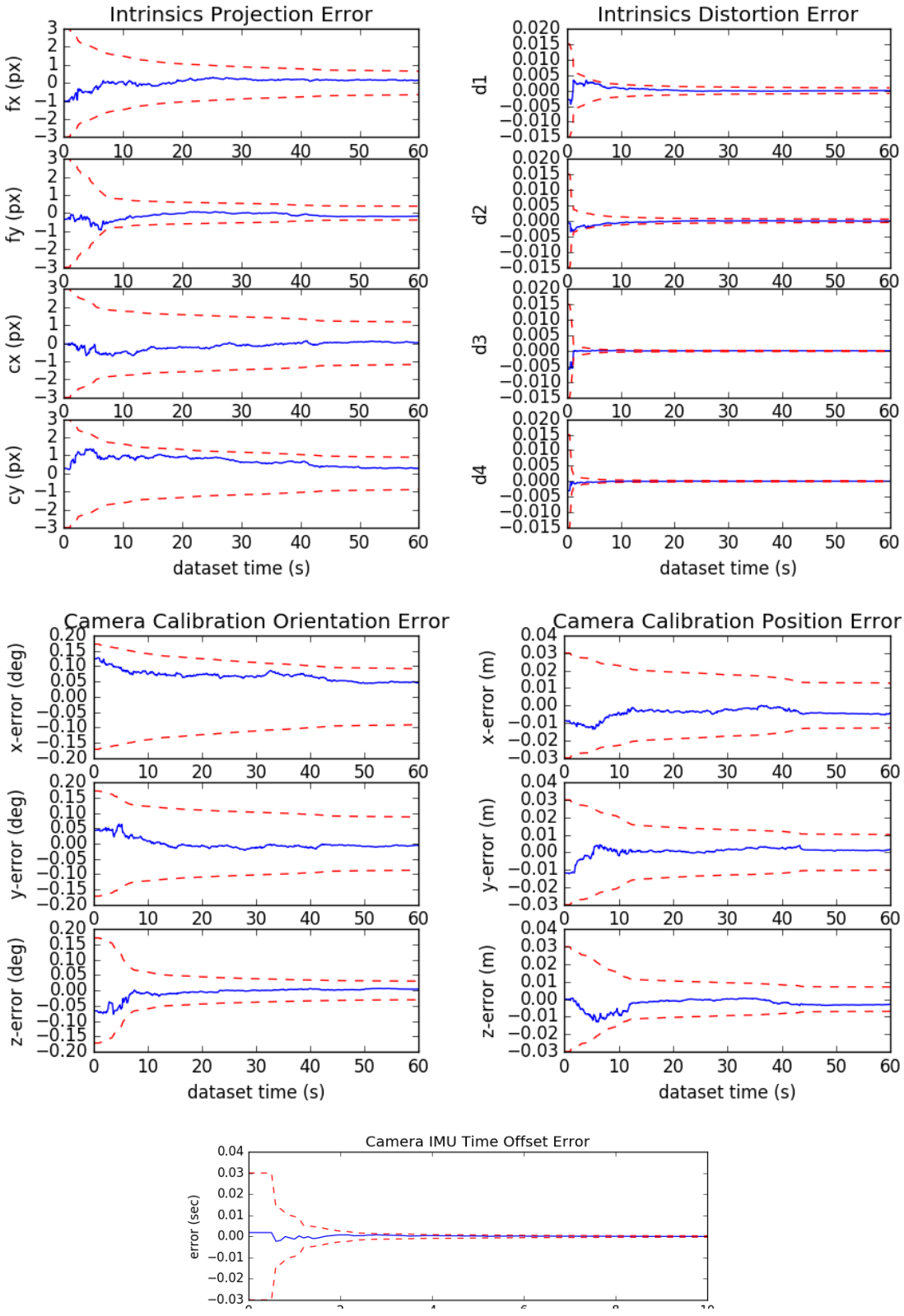



Figure 2.6: Estimated calibration parameter errors (blue-solid) and $3\sigma$ bounds (red-dashed) for a representative run. Note that we only plot the first sixty seconds of the dataset since the calibration parameters since they converge quickly.

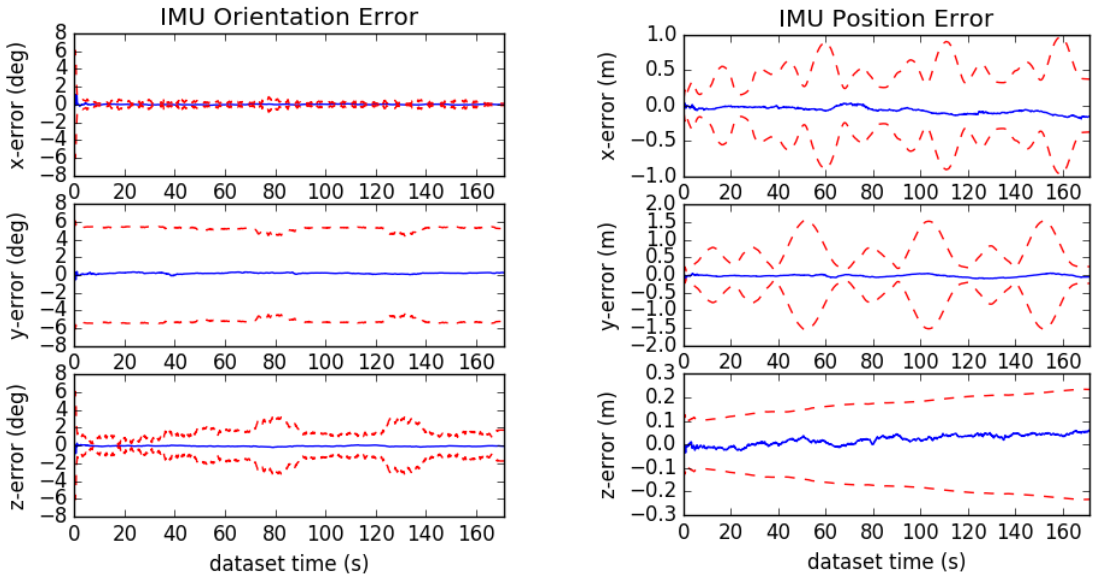



Figure 2.7: Global IMU pose errors (blue-solid) and $3\sigma$ bounds (red-dashed) for a representative run of the proposed method with SLAM landmarks and online calibration.

Table 2.3: Average ATE and NEES over twenty runs with true or bad calibration, with and without online calibration.

| | **ATE (deg)** | **ATE (m)** | **Ori. NEES** | **Pos. NEES** |
|---|---|---|---|---|
| true w/ calib | 0.212 | 0.134 | 2.203 | 1.880 |
| true w/o calib | 0.200 | 0.128 | 2.265 | 1.909 |
| bad w/ calib | 0.218 | 0.139 | 2.235 | 2.007 |
| bad w/o calib | 5.432 | 508.719 | 9.159 | 1045.174 |

are given the true calibration; while in the case that we have bad initial guesses, the estimator remains consistent and is able to estimate with reasonable accuracy. A representative run with uncertainty bounds is shown in Figure 2.7. When calibration is disabled and a bad initial guess is used, the NEES becomes large due to not modeling the uncertainty that these calibration parameters have, and in many cases, the estimate diverges. We also plot the first ten and sixty seconds of all calibration parameters of a representative run in Figure 2.6, showing that these parameters rapidly converge from their initially poor guesses.

Table 2.4: Ten runs mean absolute trajectory error (ATE) for each algorithm in units of degree/meters. Note that V2_03 dataset is excluded due the inability for some algorithms to run on it. Green denotes the best, while blue is second best.

| | **V1_01_easy** | **V1_02_medium** | **V1_03_difficult** | **V2_01_easy** | **V2_02_medium** | **Average** |
|---|---|---|---|---|---|---|
| mono_ov_slam | 0.699 / 0.058 | 1.675 / 0.076 | 2.542 / 0.063 | 0.773 / 0.124 | 1.538 / 0.074 | **1.445** / **0.079** |
| mono_ov_vio | 0.642 / 0.076 | 1.766 / 0.096 | 2.391 / 0.344 | 1.164 / 0.121 | 1.248 / 0.106 | **1.442** / 0.148 |
| mono_okvis | 0.823 / 0.090 | 2.082 / 0.146 | 4.122 / 0.222 | 0.826 / 0.117 | 1.704 / 0.197 | 1.911 / 0.154 |
| mono_rovioli | 2.249 / 0.153 | 1.635 / 0.131 | 3.253 / 0.158 | 1.455 / 0.106 | 1.678 / 0.153 | 2.054 / 0.140 |
| mono_rvio | 0.994 / 0.094 | 2.288 / 0.129 | 1.757 / 0.147 | 1.735 / 0.144 | 1.690 / 0.233 | 1.693 / 0.149 |
| mono_vinsfusion_vio | 1.199 / 0.064 | 3.542 / 0.103 | 5.934 / 0.202 | 1.585 / 0.073 | 2.370 / 0.079 | 2.926 / **0.104** |
| stereo_ov_slam | 0.856 / 0.061 | 1.813 / 0.047 | 2.764 / 0.059 | 1.037 / 0.056 | 1.292 / 0.047 | 1.552 / **0.054** |
| stereo_ov_vio | 0.905 / 0.061 | 1.767 / 0.056 | 2.339 / 0.057 | 1.106 / 0.053 | 1.151 / 0.048 | **1.454** / **0.055** |
| stereo_basalt | 0.654 / 0.035 | 2.067 / 0.059 | 2.017 / 0.085 | 0.981 / 0.046 | 0.888 / 0.059 | **1.321** / 0.057 |
| stereo_iceba | 0.909 / 0.059 | 2.574 / 0.120 | 3.206 / 0.137 | 1.819 / 0.128 | 1.212 / 0.116 | 1.944 / 0.112 |
| stereo_okvis | 0.603 / 0.039 | 1.963 / 0.079 | 4.117 / 0.122 | 0.834 / 0.075 | 1.201 / 0.092 | 1.744 / 0.081 |
| stereo_smsckf | 1.108 / 0.086 | 2.147 / 0.121 | 3.918 / 0.198 | 1.181 / 0.083 | 2.142 / 0.164 | 2.099 / 0.130 |
| stereo_vinsfusion_vio | 1.073 / 0.054 | 2.695 / 0.089 | 3.643 / 0.132 | 2.499 / 0.071 | 2.006 / 0.074 | 2.383 / 0.084 |

Table 2.5: Relative pose error (RPE) for different segment lengths for each algorithm variation over all datasets in units of degree/meters. Note that V2_03 dataset is excluded due the inability for some algorithms to run on it.

| | **8m** | **16m** | **24m** | **32m** | **40m** | **48m** |
|---|---|---|---|---|---|---|
| mono_ov_slam | **0.661** / **0.074** | **0.802** / **0.086** | **0.979** / **0.097** | **1.061** / **0.105** | **1.145** / **0.120** | **1.289** / **0.122** |
| mono_ov_vio | 0.826 / 0.094 | 1.039 / 0.106 | 1.215 / **0.111** | 1.283 / **0.132** | 1.342 / **0.151** | 1.425 / 0.184 |
| mono_okvis | **0.662** / 0.107 | **0.870** / 0.161 | 1.031 / 0.190 | 1.225 / 0.213 | 1.384 / 0.240 | 1.603 / 0.251 |
| mono_rovioli | 1.136 / 0.095 | 1.585 / 0.135 | 1.847 / 0.184 | 2.078 / 0.226 | 2.218 / 0.263 | 2.402 / 0.295 |
| mono_rvio | 0.705 / 0.130 | 0.902 / 0.160 | **1.029** / 0.183 | **1.074** / 0.213 | **0.991** / 0.227 | **1.077** / 0.232 |
| mono_vinsfusion_vio | 0.940 / **0.070** | 1.298 / **0.103** | 1.680 / 0.118 | 1.822 / 0.146 | 1.833 / 0.153 | 1.860 / **0.171** |
| stereo_ov_slam | 0.685 / 0.069 | 0.876 / 0.080 | 1.064 / **0.087** | 1.169 / **0.087** | 1.275 / **0.098** | 1.488 / **0.105** |
| stereo_ov_vio | 0.722 / 0.068 | 0.892 / **0.077** | 1.089 / **0.087** | 1.218 / 0.088 | 1.342 / 0.101 | 1.489 / **0.106** |
| stereo_basalt | **0.538** / **0.063** | **0.576** / **0.070** | **0.649** / **0.078** | **0.715** / **0.086** | **0.647** / **0.097** | **0.758** / 0.111 |
| stereo_iceba | 0.955 / 0.096 | 1.227 / 0.114 | 1.415 / 0.120 | 1.658 / 0.152 | 1.856 / 0.173 | 1.803 / 0.180 |
| stereo_okvis | **0.611** / 0.066 | **0.772** / 0.089 | **0.916** / 0.103 | **1.089** / 0.119 | **1.173** / 0.136 | **1.404** / 0.141 |
| stereo_smsckf | 1.084 / 0.098 | 1.462 / 0.136 | 1.578 / 0.159 | 1.667 / 0.187 | 1.901 / 0.200 | 2.134 / 0.217 |
| stereo_vinsfusion_vio | 0.946 / **0.057** | 1.357 / 0.079 | 1.721 / 0.097 | 1.928 / 0.111 | 1.935 / 0.125 | 1.805 / 0.132 |

## 2.5 Real-World Evaluations and Comparison to State-of-the-Art

We evaluate the proposed visual-inertial estimator OpenVINS with and without SLAM landmarks on the Vicon room scenarios from the EurocMav dataset [14] which provides both 20Hz stereo images, 200Hz ADIS16448 MEMS IMU measurements, and optimized groundtruth trajectories. It should be noted that we have recalculated the V1_01_easy groundtruth using Section 2.3.5 due to the original having inaccurate orientation values and have provided this corrected groundtruth trajectory to the community on our documentation website. All methods were run with the configuration files from their open-sourced repositories with each algorithm being run ten times on each dataset to compensate for some randomness inherent to the visual frontends. In this benchmarking test, we evaluate the following state-of-the-art visual-inertial estimation algorithms:

**OKVIS [109]** – Keyframe-based fixed-lag smoother which optimizes arbitrarily spaced keyframe poses connected with inertial measurement factors and environmental landmarks. Fixed window size was enforced to ensure computational feasibility with a focus on selective marginalization to allow for problem sparsity.

**VINS-Fusion VIO [161]** – Extension of the original VINS-Mono [160] sliding optimization-based method that leverages IMU preintegration which is then loosely coupled with a secondary pose-graph optimization. VINS-Fusion extends the original codebase to support stereo cameras.

**Basalt VIO [190]** – Stereo keyframe-based fix-lag smoother with custom feature tracking frontend with a focus on extracting relevant information from the VIO for later offline visual-inertial mapping.

**R-VIO [78]** – Robocentric MSCKF-based algorithm which estimates in a local frame and updates the global frame through a composition step. The direction of gravity is also estimated within the filter.

**ROVIO** [7] – We use the ROVIO implementation within maplab [178], which is a monocular iterative EKF-based approach that performs minimization on the direct image intensity patches allowing for tracking of non-corner features such as high gradient lines.

**ICE-BA** [119] – Stereo incremental bundle adjustment (BA) method which optimizes both a local siding window and global optimization problem in parallel. They exploited the sparseness of their formulation and introduced a relative marginalization procedure.

**S-MSCKF** [182] – An open-sourced implementation of original MSCKF [139] paper with stereo feature tracking and a focus on high-speed motion scenarios.

We evaluate *only* the VIO portion of these codebases (i.e., not the non-real-time back-end pose graph thread output of VINS-Fusion [161] and visual-inertial mapping of Basalt [190]), as the focus is the visual-inertial *odometry* performance.

Table 2.4 shows the average ATE of all methods for each dataset. It is clear that the addition of SLAM landmarks in our OpenVINS greatly reduces the drift in the monocular case, while it has a smaller impact on the stereo performance; and more importantly, OpenVINS can perform competitively to other methods. We additionally compared the Relative Pose Error (RPE) [221] of all methods over all datasets. Shown in Table 2.5, our monocular system clearly outperforms the current open-sourced codebases, with our stereo system being able to perform second to Basalt. In terms of computational cost, we found that Basalt outperformed all other algorithms, with our proposed method being limited by the visual-frontend implementation from OpenCV [149] and SLAM feature update equally. On the first EurocMav dataset we could process at 2.7x/4.3x and 1.2x/1.9x real-time for our monocular SLAM/VIO, and stereo SLAM/VIO, respectively, on an Intel(R) Xeon(R) CPU E3-1505M v6 @ 3.00GHz processor in single threaded execution.

## 2.6 Summary

In this chapter, we have presented a foundational framework that achieves state-of-the-art performance in an efficient, consistent, and robust manner, but also contains a significant amount of extensions which enables its practical use as both a platform for performing future research but also deployment onto real robotic systems. We first covered the basic visual-inertial framework followed by the key extensions which build OpenVINS. This includes the novel type-based covariance management system, state variable delayed initialization, complete feature observation model which incorporates the camera intrinsic and distortion parameters, concurrent spatiotemporal calibration of camera intrinsic, extrinsic, and time offset, accurate groundtruth generation through mocap IMU fusion, stationary and dynamic state initialization, flexible visual-inertial and simulator with continuous-time trajectory. Through simulation, the ability to perform robust and accurate calibration along with *consistent* trajectory estimation was shown, and real-world experiments demonstrated the ability to perform state-of-the-art visual-inertial estimation.

# Chapter 3

# EXTENDING VINS TO HIGHER LEVEL GEOMETRIC PRIMITIVES

## 3.1 Introduction

We next focus on how to incorporate higher-level geometric primitives such as planes within the context of VINS. First, we present the Closest Point (CP) plane representation and its use as a method for measurement compression to reduce the complexity within a LiDAR-inertial system. The novel CP plane representation is shown to have very appealing properties of being a minimal representation size (3 DoF) and improved linearity as compared to the quaternion representation [89]. While it does have a singularity when the plane intersects its frame of reference, we argue this will not be the case if we actively observe the plane since this would coincide with the sensor itself being inside the plane. We additionally demonstrate the ability to "compress" all points on a given plane to a single CP plane along with recovering its uncertainty to enable probabilistic fusion with an IMU. This compressed representation and the resulting plane-to-plane measurement model reduces the number of measurement residuals typically needed if using traditional point-on-plane constraints which can cause a large computational burden within a non-linear optimization which relinearizes each residual for the millions of LiDAR points. A LiDAR-inertial simulator is developed the validate the proposed CP plane compression method along with the accuracy of the CP representation versus the existing state-of-the-art. We finally demonstrate the system on a small-scale real-world dataset.

We will then explore the CP plane's use within the filter-based monocular-inertial estimator OpenVINS as a regularity to enable a significant number of loop-closures due to the large spatial nature of environmental planes in man-made environments. Specifically, environmental CP planes are both detected and tracked through

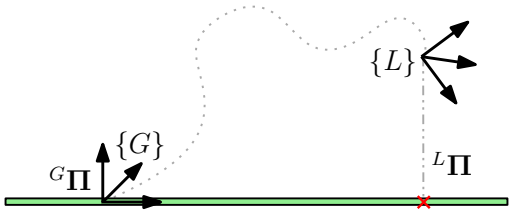



Figure 3.1: A visual representation of the closest point on the plane. Also shown is an example of a local plane parameter $^{L}\mathbf{\Pi}$ that is well defined, while the global plane representation $^{G}\mathbf{\Pi}$ is ill-defined.

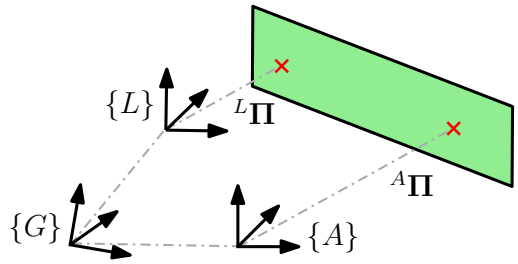



Figure 3.2: Pictorial view of a closest point plane representation seen in the local $\{L\}$ frame which can be transformed into its anchor frame $\{A\}$ and vice versa.

a novel algorithm which only leverages the monocular image feed without an active depth sensor or expensive neural network. Special care is taken to estimate these planes concurrently within the state in a consistent manner. Sparse 3D points are then regularized to these estimated planes via a point-on-plane measurement model and are shown to provide accuracy gains with only a minimal computational cost increase. Detailed simulations and real-world experiments on both existing datasets and a self-collected AR dataset show the impact of leveraging this regularization.

### 3.2 Closest Point Plane Representation

The "Closest Point (CP)" representation can be described as a 3D point that resides on the plane and is the *closest* to the current frame's origin. The benefit of this representation is that it is already in its minimal representation and is singularity free if special care is taken to select the frame of reference it is defined from. By representing the plane as a single 3D point, we also have a simple additive error model when updating the parameter during optimization. This CP representation can be described using the Hesse normal vector $^{G}\mathbf{n}$ and distance scalar $^{G}d$:

$$^{G}\mathbf{\Pi} = {}^{G}\mathbf{n}\ {}^{G}d \tag{3.1}$$

$$\begin{bmatrix} {}^{G}\mathbf{n} \\ {}^{G}d \end{bmatrix} = \begin{bmatrix} {}^{G}\mathbf{\Pi} / \left\| {}^{G}\mathbf{\Pi} \right\| \\ \left\| {}^{G}\mathbf{\Pi} \right\| \end{bmatrix} \tag{3.2}$$

It is important to point out that without special care, this representation still has a singularity when the value of ${}^{G}d$ approaches zero. Any plane ${}^{G}\mathbf{\Pi}$ that intersects our frame of reference ($\{G\}$ in this case) will be represented as the same zero vector regardless of the plane's orientation since the closest point on that plane is at the origin. Nevertheless, we argue that this singularity is well suited in the case of plane estimation using range-based sensors (e.g., LiDAR and RGBD cameras) since planes extracted from these sensors will not be ill-defined if they are represented in the frame they are extracted from. The singularity in practice would only arise if we transform a local CP plane, ${}^{L}\mathbf{\Pi}$, into a frame where the plane intersects its origin (see Figure 3.1). It was also noted in [218] that planes extracted from range-based sensors that are close to intersecting the sensor frame should be considered "unreliable" if found and discarded.

### 3.2.1 Anchored Plane-to-Plane Constraint

To overcome the aforementioned singularity issue of the CP representation, we parameterize the plane in the first observation frame, guaranteeing that the distance to the plane will be non-zero (from here forward this will be denoted the "anchor" frame/state). As seen in Figure 3.2, the transform of the plane representation from one frame to another is *not* a direct 3D point transformation, and instead requires the calculation of the CP in the new frame. Using the Hesse plane representation we can map a plane represented in the anchor frame $\{A\}$ into the local frame $\{L\}$ as:

$$\begin{bmatrix} {}^{L}\mathbf{n} \\ {}^{L}d \end{bmatrix} = \begin{bmatrix} {}^{L}_{A}\mathbf{R} & 0 \\ -{}^{A}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}\mathbf{n} \\ {}^{A}d \end{bmatrix} \tag{3.3}$$

where ${}^{L}_{A}\mathbf{R}$ is the relative rotation between the local and anchor LiDAR frames, ${}^{A}\mathbf{p}_{L}$ is the position of the local LiDAR frame seen from the anchor LiDAR frame.

The transform represents the rotation of the anchor plane normal vector into the local frame, and the subtraction of the distance between the two frames projected

onto the anchor plane normal. Using the definition of the CP representation (3.1), we can use the above $\{{}^{A}\mathbf{n}, {}^{A}d\}$ relation to obtain:

$$ {}^{L}\mathbf{\Pi}(\mathbf{x}) = \left({}^{L}_{A}\mathbf{R}\,{}^{A}\mathbf{n}\right)\left({}^{A}d - {}^{A}\mathbf{p}_{L}^{\top}\,{}^{A}\mathbf{n}\right) \tag{3.4} $$

For a given plane measurement, ${}^{L}\widehat{\mathbf{\Pi}}$, we have the following residual for use in graph optimization (see Eq. (3.12)):

$$ r_{\Pi}(\mathbf{x}) = {}^{L}\mathbf{\Pi}(\mathbf{x}) - {}^{L}\widehat{\mathbf{\Pi}} \tag{3.5} $$

The Jacobians needed for graph optimization can be found in Appendix D.1.

### 3.2.2 Point-to-Plane Measurement Compression

To get the local CP measurement, we fit planes to the incoming point clouds from the LiDAR sensor. To find subsets of the unordered point cloud that correspond to planes, RANSAC or other plane segmentation methods can be used. We model each point measurement ${}^{L}\mathbf{p}_{mi}$ as a true measurement ${}^{L}\mathbf{p}_{i}$ being corrupted by some zero mean Gaussian noise:

$$ {}^{L}\mathbf{p}_{mi} = {}^{L}\mathbf{p}_{i} + \mathbf{n}_{p}, \quad \mathbf{n}_{p} \sim \mathcal{N}(0, \mathbf{R}_{d}) \tag{3.6} $$

We look to first compress the extracted subset of points into a *local* CP primitive and matching covariance that can be used in optimization. We can start by formulating a weighted least squared optimization problem where we seek to minimize the point-to-plane distances between extracted points and the local CP measurement ${}^{L}\mathbf{\Pi}$:

$$ {}^{L}\mathbf{\Pi}^{*} = \underset{{}^{L}\mathbf{\Pi}}{\operatorname{argmin}} \sum_{i} \left\| \frac{{}^{L}\mathbf{\Pi}^{\top}}{\|{}^{L}\mathbf{\Pi}\|}\,{}^{L}\mathbf{p}_{mi} - \|{}^{L}\mathbf{\Pi}\| \right\|^{2}_{W_{i}^{-1}} \tag{3.7} $$

where $W_i$ is the inverse variance of the noise that corrupts the $i$th measurement. In practice, we also introduce a robust Huber loss to minimize the effect of outliers during optimization (see [37]). We minimize the above cost function using the Gauss-Newton method of iterative linearization of the residual about the current best estimate. Formally, we solve for the correction, ${}^{L}\widetilde{\mathbf{\Pi}}$, to our linearization point ${}^{L}\widehat{\mathbf{\Pi}}$:

$$ {}^{L}\widetilde{\mathbf{\Pi}} = -\left(\sum_{i} \mathbf{J}_{i}^{\top}\, W_{i}\, \mathbf{J}_{i}\right)^{-1}\left(\sum_{i} \mathbf{J}_{i}^{\top} W_{i}\, r_{i}({}^{L}\widehat{\mathbf{\Pi}})\right) $$

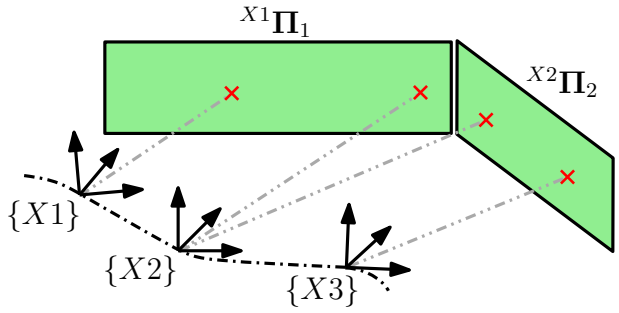



Figure 3.3: Pictorial view an example trajectory and measurements that are included in the proposed LIPS optimization. Two planes are being estimated in different anchor poses. In the case of $^{X1}\mathbf{\Pi}_1$, it was first seen from the $X1$ state so it will be estimated in the $\{X1\}$ frame of reference while $^{X2}\mathbf{\Pi}_2$ will be estimated in the $\{X2\}$ frame of reference.

where we have the following:

$$\mathbf{J}_i = \frac{^{L}\mathbf{p}_{mi}^{\top}}{\|^{L}\widehat{\mathbf{\Pi}}\|} - \left(^{L}\mathbf{p}_{mi}^{\top}\,{}^{L}\widehat{\mathbf{\Pi}}\right)\frac{^{L}\widehat{\mathbf{\Pi}}^{\top}}{\|^{L}\widehat{\mathbf{\Pi}}\|^3} - \frac{^{L}\widehat{\mathbf{\Pi}}^{\top}}{\|^{L}\widehat{\mathbf{\Pi}}\|} \tag{3.8}$$

$$W_i = \left(\frac{^{L}\widehat{\mathbf{\Pi}}^{\top}}{||^{L}\widehat{\mathbf{\Pi}}||}\,\mathbf{R}_d\,\frac{^{L}\widehat{\mathbf{\Pi}}}{||^{L}\widehat{\mathbf{\Pi}}||}\right)^{-1} \tag{3.9}$$

Additionally, we can calculate the covariance matrix of the final local CP $^{L}\widehat{\mathbf{\Pi}}$ as the following:

$$\mathbf{P}_{\Pi} = \left(\sum_i \mathbf{J}_i^{\top}\,W_i\,\mathbf{J}_i\right)^{-1} \tag{3.10}$$

In summary, we compress each of the extracted subsets of the point cloud into *local* CP plane, $^{L}\widehat{\mathbf{\Pi}}$, representations, which are then directly used to construct plane factors. We found that using a robust Huber loss function on the plane factors led to lower sensitivity to poor plane measurement compression performance.

### 3.3 LiDAR-Inertial 3D Plane SLAM (LIPS)

In the proposed LiDAR-Inertial 3D Plane SLAM (LIPS) the total state to be estimated, $\mathbf{x}$, consists of the $m$ historical IMU states and $k$ planar primitives:

$$\mathbf{x} = \begin{bmatrix}\mathbf{x}_{I_1}^{\top} & \cdots & \mathbf{x}_{I_N}^{\top} & ^{A}\mathbf{\Pi}_1^{\top} & \cdots & ^{A}\mathbf{\Pi}_M^{\top}\end{bmatrix}^{\top} \tag{3.11}$$

where $\mathbf{x}_{I_k}$ is the state of the IMU at timestep $t_k$ and ${}^{A}\mathbf{\Pi}_j$ refers to the $j$th plane, which is represented in the first LiDAR frame it was observed from $\{A\}$. To perform this estimation, we first determine the continuous IMU preintegration factors [40], and 3D plane factors from LiDAR measurements (see anchored CP in Section 3.2.1), thereby building a factor graph for optimization. In particular, planes are first extracted from the point cloud and then compressed into the CP representation (see Section 3.2.2). Figure 3.3 illustrates the overall LIPS system. These measurements are added to the graph and optimized using the iSAM2 [90] implementation available in the GTSAM [28] nonlinear optimization library. Note that, while not used in our small-scale experiments (see Section 4.3.4), an advantage of the relative information provided by the IMU preintegration is the ability to perform LiDAR cloud unwarping during high-speed maneuvers. The final cost function of the LIPS system can be described as:

$$\hat{\mathbf{x}} = \underset{\mathbf{x}}{\operatorname{argmin}} \left[ \sum_{N} \left|\left| \mathbf{r}_{Ik} \left( \mathbf{x}_{I_{k-1}}, \mathbf{x}_{I_k} \right) \right|\right|^2_{\mathbf{P}_{k-1,k}} + \sum_{M} \left|\left| \mathbf{r}_{\Pi j} \left( \mathbf{x} \right) \right|\right|^2_{\mathbf{P}_{\Pi j}} \right] \tag{3.12}$$

where $\mathbf{r}_{Ik}(\mathbf{x}_{I_{k-1}}, \mathbf{x}_{I_k})$ and $\mathbf{r}_{\Pi j}(\mathbf{x})$ are the zero mean residuals associated with the continuous preintegration and anchored CP planes measurements, respectively. While, $\mathbf{P}_{k-1,k}$ and $\mathbf{P}_{\Pi j}$ are the covariances of the continuous preintegration and anchor CP plane measurements, respectively.

### 3.3.1 LiDAR-Inertial Simulator

To evaluate the feasibility of the proposed system, a custom LiDAR IMU simulator was developed.[1] A 2D floorplan was created and extruded vertically to create a Manhattan world environment (we note, for clarity, that the CP representation can handle arbitrary plane orientations). A 3D B-Spline is leveraged to recover groundtruth poses and IMU measurements (see Figure 3.5 for the generated trajectory). At a given LiDAR sensing frequency, rays were generated using the intrinsic LiDAR sensor model defined by the angular resolution and vertical zenith angles. Generated rays are then intersected with all planes in the environment and all ray-plane intersections are found.

[1] https://github.com/rpng/lips

Table 3.1: Realistic parameters used in simulation.

| **Simulation Parameter** | **Value** |
|---|---|
| Gyroscope Noise Density | 0.005 $rad/s\sqrt{Hz}$ |
| Gyroscope Random Walk | 4.0e-06 $rad/s^2\sqrt{Hz}$ |
| Accelerometer Noise Density | 0.01 $m/s^2\sqrt{Hz}$ |
| Accelerometer Random Walk | 2.0e-04 $m/s^3\sqrt{Hz}$ |
| LiDAR Point Deviation | 1 and 3 $cm$ |
| LiDAR Angular Resolution | 0.25° |
| LiDAR Zenith Angles | 3.2°,0.0°,-3.2°,-6.4°,-9.5°,-12.5°,-15.4°,-18.3° |
| Rotation LiDAR to IMU | [-1,0,0;0,1,0;0,0,-1] |
| Position IMU in LiDAR | [0;0.04;-0.06] $m$ |
| Global Gravity | [0,0,9.81] $m/s^2$ |
| LiDAR / IMU Sensor Rate | 5 / 800 $Hz$ |

The final step performs invalidation of intersections that should not be generated due to occlusions by enforcing that each ray should *only* hit the plane that is *closest* to the LiDAR frame.

The simulation parameters shown in Table 3.1 are modeled after a Quanergy M8 sensor with an ADIS16448 IMU rigidly attached. Following the conventional inertial model, IMU measurements have additive discrete bias and white noise terms corrupting the true value of each measurement axis. The noise corrupting the generated 3D LiDAR points is modeled as an additive white noise to each measurement axis.

### 3.3.2 Monte-Carlo Simulations

A total of 80 Monte-Carlo simulations of the LIPS system were performed at varying LiDAR noise values, whose results are shown in Figure 3.4 and Table 3.2. The proposed CP representation and anchor plane factors were able to localize in the planar environment with high accuracy at different levels of LiDAR sensor noise. The simulations were done in real-time with the plane correspondences known and solved using iSAM2. The large non-zero orientation error towards the beginning is due to the sensors remaining stationary for a period after initialization with only a small number of faraway planes constraining the orientation.

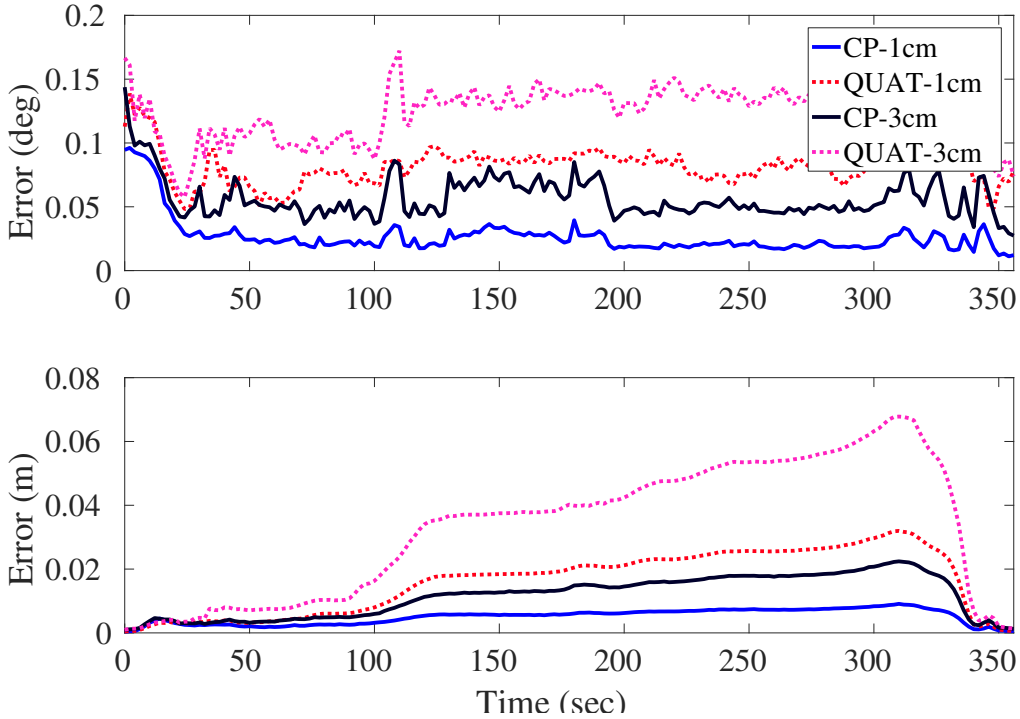



Figure 3.4: RMSE results from 80 Monte-Carlo simulations showing the achievable accuracy of the proposed LIPS system running in real-time. We show RMSE results for both CP and quaternion representations.

The beginning of the simulated trajectory has limited amounts of loop-closures due to the entering and leaving of rooms causing the estimation error to increase as one would normally see in odometry systems (see Figure 3.5). After 300 seconds the trajectory re-enters the long hallway and returns towards the starting position re-observing previously seen sections of the hallway. As seen in Figure 3.4, loop-closures with previous planes rapidly decrease the estimator error towards zero.

### 3.3.3 Plane Representation Comparison

To evaluate the effect of using the CP representation, we compare against the state-of-the-art that leverages the quaternion and its minimal error state [89]. We compressed the sets of point clouds into the quaternion representation using the same methodology used for CP, in which we perform a minimization on the point-to-plane distances. We implemented the "relative quaternion" factor proposed by Kaess, which

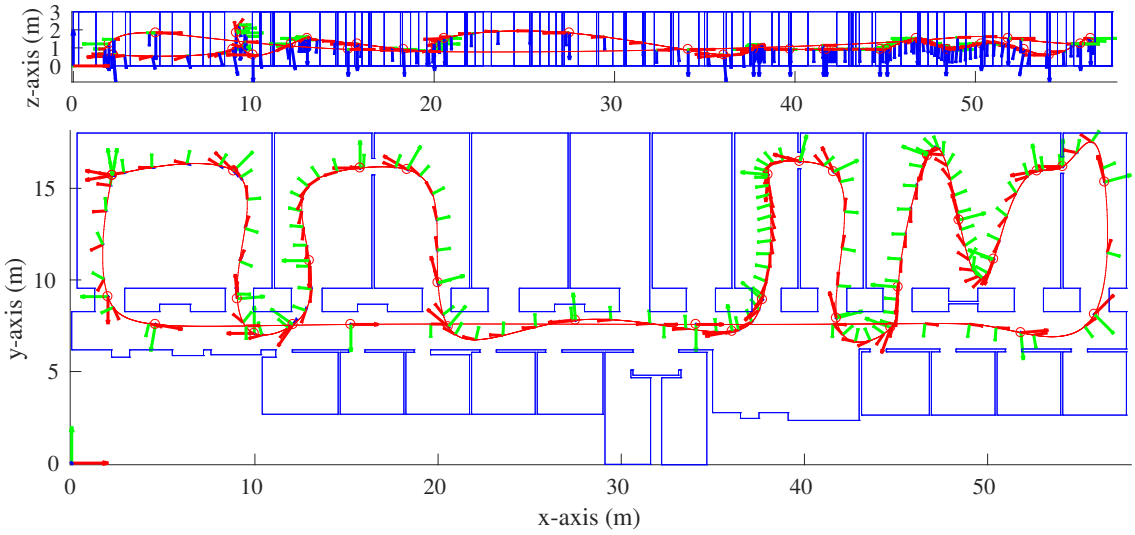



Figure 3.5: Generated 180 meter long simulation trajectory through a 3D environment. The original 2D floor plan (bottom) has been extruded, and a spline was fitted to control points to generate a complete trajectory. The trajectory starts in the top left corner and weaves in and out of rooms with varying heights from the floor before finally re-entering the hallway and returning back to the starting location.

Table 3.2: Average RMSE over 80 Monte-Carlo simulations at different LiDAR noise values.

| | **Closest Point (CP)** | | **Quaternion** [89] | |
|---|---|---|---|---|
| Units | m | deg | m | deg |
| 1 cm | 0.005 | 0.027 | 0.016 | 0.081 |
| 3 cm | 0.012 | 0.057 | 0.033 | 0.126 |

is directly comparable to our "anchored" CP representation (whose analytical Jacobians can be found in Appendix D.2). As shown in Table 3.2, the CP representation yielded improved accuracy over its quaternion counterpart. While the results presented here were generated using the iSAM2 solver, we found that during full batch optimization the quaternion representation converged slower compared to CP. Our conjecture for these results is that the CP-based measurement model has a better linear Gaussian approximation than the quaternion parametrization, and thus provides better numerical performance.

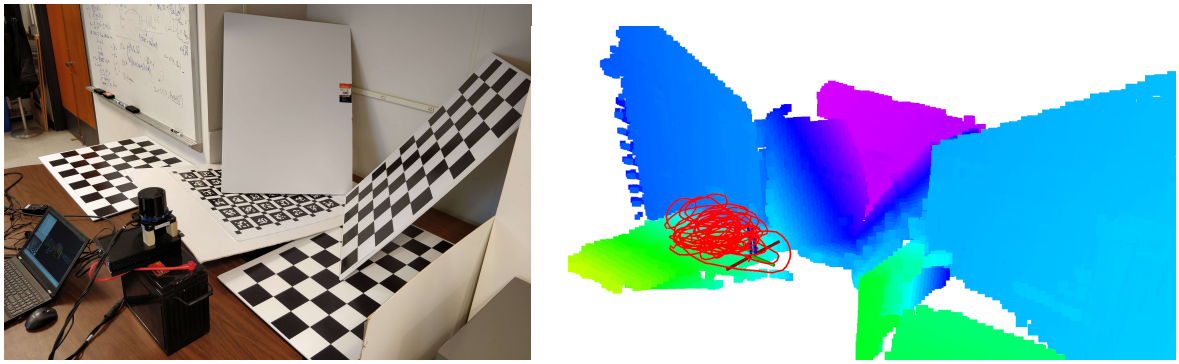


Figure 3.6: Experimental environment that the proposed localization operated in (left) and reconstructed depth map of the planar surfaces (right).

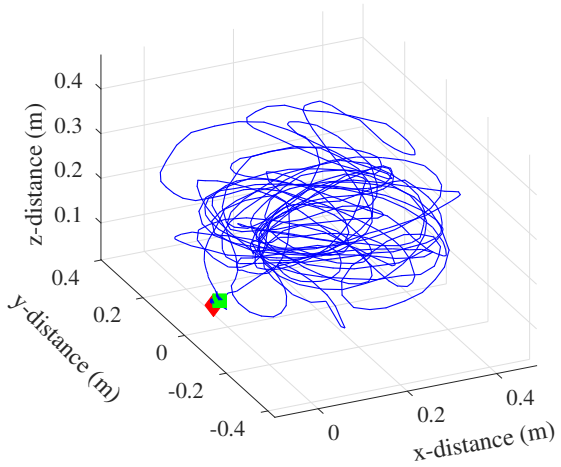



Figure 3.7: 3D trajectory of the sensor during the experiment. Total path length was 30 meters with the difference between the starting and end position being 1.5cm. The green square and red diamond denote the start and end of the trajectory, respectively.

### 3.3.4 Real-World Experiments

To further validate that the proposed LIPS system can be realized on physical sensors, the issue of plane correspondences needed to be addressed. While there have been works on matching planes in 3D space [76, 184], a simple Mahalanobis-distance test between incoming local CPs and existing planes was sufficient for our small scale experiments. Point clouds were first processed offline using RANSAC plane segmentation, available in the Point Cloud Library (PCL) [172], to find planar subsets. The measurement compression and estimator were able to run in real-time, but for real-world applications the RANSAC plane extraction will need to be substituted or have its execution time decreased to reach real-time performance.

In this test, planar objects were placed around the LiDAR sensor to allow for easy RANSAC extraction, to avoid degenerate motions [211], and to ensure that the LiDAR was fully constrained in all degrees of freedom (see Figure 3.6). An eight channel Quanergy M8 LiDAR operating at 10Hz was used with a Microstrain 3DM-GX3-25 IMU attached to the bottom of the LiDAR operating at 500Hz. We manually estimated the LiDAR to IMU extrinsic transformation but this could easily be added to the factor graph for online estimation. To evaluate the estimation drift, the sensor unit was moved in front of the planar surfaces and returned to the same starting location. As seen in Figure 3.7, after a 30 meter trajectory distance, the difference between the start and end poses was 1.5 cm corresponding to 0.05% error over the trajectory length.

## 3.4 Monocular Visual-Inertial Odometry with Planar Regularities

The state-of-the-art filter-based monocular VINS method OpenVINS directly relies on sparse point features in part due to their efficiency, robustness, and prevalence. However, additional information through high-level structural regularities such as planes that are common to man-made environments can be exploited to further constrain motion. Generally, planes can be observed by a camera for significant periods of time due to their large spatial presence and thus, are amenable for long-term navigation. We present an efficient and robust *monocular*-based plane detection algorithm,

which does *not* require additional sensing modalities such as a stereo or depth camera as commonly seen in the literature, while enabling real-time regularization of point features to environmental planes. These long-lived planes are maintained in the state vector, while shorter ones are marginalized after use for efficiency. Planar regularities via a point-on-plane constraint are applied to both in-state SLAM features and out-of-state MSCKF features, thus fully exploiting the environmental plane information to improve VIO performance. To the best of our knowledge, this is the first time that a monocular-VIO estimator, termed `ov_plane`, is able to rigorously enforce planar regularities within the MSCKF framework. An overview of the proposed approach is shown in Algorithm 1.

At time $t_k$, the system state $\mathbf{x}_k$ consists of the current navigation states $\mathbf{x}_{I_k}$, historical IMU pose clones $\mathbf{x}_C$, and a subset of 3D environmental (SLAM) point features, $\mathbf{x}_f$, and (SLAM) plane features, $\mathbf{x}_\pi$:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{I_k}^\top & \mathbf{x}_C^\top & \mathbf{x}_f^\top & \mathbf{x}_\pi^\top \end{bmatrix}^\top \tag{3.13}$$

$$\mathbf{x}_C = \begin{bmatrix} \mathbf{x}_{T_k}^\top & \dots & \mathbf{x}_{T_{k-c}}^\top \end{bmatrix}^\top \tag{3.14}$$

$$\mathbf{x}_f = \begin{bmatrix} {}^G\mathbf{p}_{f_1}^\top & \dots & {}^G\mathbf{p}_{f_g}^\top \end{bmatrix}^\top \tag{3.15}$$

$$\mathbf{x}_\pi = \begin{bmatrix} {}^G\mathbf{\Pi}_1^\top & \dots & {}^G\mathbf{\Pi}_h^\top \end{bmatrix}^\top \tag{3.16}$$

where we have the current inertial state $\mathbf{x}_{I_k}$, a collection of SLAM features ${}^G\mathbf{p}_{f_i}$, historical MSCKF clones $\mathbf{x}_{T_i} = [{}^{I_i}_G\bar{q}^\top \ {}^G\mathbf{p}_{I_i}^\top]^\top$, and estimated CP planes $\mathbf{x}_\pi$. We represent each plane, ${}^G\mathbf{\Pi}$, in the global frame with the minimal error state *closest point* (CP) representation (see Section 3.2). Note that we have dropped any calibration states OpenVINS additionally supports here for presentation clarity.

### 3.4.1 Regularization-Constrained Measurement

At the core of the proposed `ov_plane` system are the planar regularities. In the following, we explain how to perform MSCKF updates with planar regularities, while addressing practical challenges, which include efficient point-feature updates constrained by (in-state and out-of-state) planes, and robust initialization of plane features

**Algorithm 1** `ov_plane`

**Propagation:**

- Propagate the state vector and covariance with inertial readings [see Section 5.2.1]

**Feature Tracking**:

- Extract visual features from the image, then perform sparse KLT tracking and outlier rejection.
- Formulate a 2D Delaunay triangle mesh, detect, and match planes [Section 3.4.7.1]

**State Management:**

- Initialize SLAM point and plane features into the state if sufficient observations / features [Section 3.4.3]
- Merge planes if needed [Section 3.4.4]
- Marginalize SLAM point and plane features from the state when tracking is lost

**Update:**

- Update non-plane points [Eq. (2.52), (2.54)]
- Update MSCKF plane (out-of-state)
  - Recover points and plane, then jointly refine their estimates [Section 3.4.2]
  - Nullspace project $\mathbf{H}_\pi$ pre-update [Eq. (3.27), (3.29)]
- Update SLAM plane (in-state)
  - SLAM points directly update planes [Eq. (3.21)]
  - MSCKF points are projected onto their $\mathbf{H}_f$ nullspace before update [Eq. (3.24)]

that are augmented into the state. For clarity, we refer the reader to Section 3.4.7.1 for the proposed real-time extraction and robust matching of planes from sparse visual points.

The proposed VIO system enforces planar regularities through point-on-plane constraints. Consider a point feature ${}^{G}\mathbf{p}_f$ that lies on the plane ${}^{G}\mathbf{\Pi}$, we have:

$$z_d = \left({}^{G}\mathbf{p}_f^{\top}{}^{G}\mathbf{n} - {}^{G}d\right) + \sigma_d \tag{3.17}$$

where $\sigma_d$ is the noise that softens the constraint and should be zero in the ideal case [181]. We linearized Eq. (3.17) to get:

$$\tilde{z}_d = \mathbf{H}_f^{dG}\tilde{\mathbf{p}}_f + \mathbf{H}_\pi^{dG}\tilde{\mathbf{\Pi}} + \sigma_d \tag{3.18}$$

We then stack the point feature bearing observation model, Eq. (2.52), and point-on-plane constraint, Eq. (3.18):

$$\begin{bmatrix}\tilde{\mathbf{z}}_c \\ \tilde{z}_d\end{bmatrix} = \begin{bmatrix}\mathbf{H}_T^c \\ \mathbf{0}\end{bmatrix}\tilde{\mathbf{x}}_C + \begin{bmatrix}\mathbf{H}_f^c \\ \mathbf{H}_f^d\end{bmatrix}{}^{G}\tilde{\mathbf{p}}_f + \begin{bmatrix}\mathbf{0} \\ \mathbf{H}_\pi^d\end{bmatrix}{}^{G}\tilde{\mathbf{\Pi}} + \begin{bmatrix}\mathbf{n}_c \\ \sigma_d\end{bmatrix} \tag{3.19}$$

$$\Rightarrow \tilde{\mathbf{z}} = \mathbf{H}_T\tilde{\mathbf{x}}_C + \mathbf{H}_f{}^{G}\tilde{\mathbf{p}}_f + \mathbf{H}_\pi{}^{G}\tilde{\mathbf{\Pi}} + \mathbf{n} \tag{3.20}$$

$$= \mathbf{H}_x\tilde{\mathbf{x}}_k + \mathbf{H}_\pi{}^{G}\tilde{\mathbf{\Pi}} + \mathbf{n} \tag{3.21}$$

where $\mathbf{H}_T$, $\mathbf{H}_f$, and $\mathbf{H}_\pi$ are the Jacobians for the IMU poses, point feature, and plane feature, respectively; $\mathbf{H}_x = [\mathbf{H}_T\ \mathbf{H}_f]$ and $\tilde{\mathbf{x}}_k = [\tilde{\mathbf{x}}_C^{\top}\ \tilde{\mathbf{x}}_f^{\top}]^{\top}$; and $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ denotes the measurement noise after *whitening*.

### 3.4.2 Plane Recovery and Non-Linear Refinement

To enforce point-on-plane constraint, Eq. (3.18), we first robustly recover the initial guess for the plane by performing RANSAC [50] on a set of co-planar point features (details on how we extract co-planar sets are in Section 3.4.7.1 and Algorithm 2). A plane estimate can be solved from at least three points with the following linear system:

$$\begin{bmatrix}\cdots & {}^{G}\mathbf{p}_{f,i} & \cdots\end{bmatrix}^{\top}\boldsymbol{\pi} = \begin{bmatrix}\cdots & 1 & \cdots\end{bmatrix}^{\top} \tag{3.22}$$

After obtaining $\boldsymbol{\pi}$, the plane can be recovered by ${}^{G}\mathbf{n} = \boldsymbol{\pi}/||\boldsymbol{\pi}||$ and ${}^{G}d = 1/||\boldsymbol{\pi}||$. The RANSAC inlier set is selected based on the point-to-plane distance threshold, see Eq. (3.17), with the best-recovered plane having the most inliers and smallest average point-to-plane distance.

If a sufficient number of inliers are found, we perform a *joint* refinement of the point features and plane with fixed camera poses. SLAM points that lie on the plane are fixed during optimization but are included to further improve the plane estimate through their point-on-plane constraints. The non-linear optimization problem is formulated using Eq. (2.52) and (3.17) and is solved using ceres-solver [1] that takes 0.5-1.5 milliseconds (ms).

### 3.4.3 Plane Feature Initialization

We wish to initialize long-tracked planes into states, which offer dependable regularization information and constrain a large number of co-planar feature points. For an MSCKF planar point feature, in analogy to MSCKF feature marginalization, we project Eq. (3.20) onto the the left nullspace of $\mathbf{H}_f$ (i.e., $\mathbf{N}^\top\mathbf{H}_f = \mathbf{0}$) to get a residual function for plane ${}^{G}\boldsymbol{\Pi}$ that is independent to the point feature:

$$\mathbf{N}^\top\tilde{\mathbf{z}} = \mathbf{N}^\top\mathbf{H}_T\tilde{\mathbf{x}}_C + \mathbf{N}^\top\mathbf{H}_\pi{}^{G}\tilde{\boldsymbol{\Pi}} + \mathbf{N}^\top\mathbf{n} \tag{3.23}$$

$$\Rightarrow \quad \tilde{\mathbf{z}}^* = \mathbf{H}_x^*\tilde{\mathbf{x}}_k + \mathbf{H}_\pi^*{}^{G}\tilde{\boldsymbol{\Pi}} + \mathbf{n}^* \tag{3.24}$$

For a SLAM point feature, Eq. (3.21) can be directly used.

After collecting co-planar MSCKF [Eq. (3.24)] and SLAM [Eq. (3.21)] point feature measurements, we stack them into the following linear system:

$$\begin{matrix}\text{MSCKF}: \\ \text{SLAM}:\end{matrix} \begin{bmatrix}\tilde{\mathbf{z}}^* \\ \tilde{\mathbf{z}}\end{bmatrix} = \begin{bmatrix}\mathbf{H}_x^* \\ \mathbf{H}_x\end{bmatrix}\tilde{\mathbf{x}}_k + \begin{bmatrix}\mathbf{H}_\pi^* \\ \mathbf{H}_\pi\end{bmatrix}{}^{G}\tilde{\boldsymbol{\Pi}} + \begin{bmatrix}\mathbf{n}^* \\ \mathbf{n}\end{bmatrix} \tag{3.25}$$

where $\tilde{\mathbf{z}}^*$ and $\tilde{\mathbf{z}}$ represent the MSCKF and SLAM point feature measurement residuals, respectively. We then leverage the delayed initialization from Section 2.3.2 to initialize ${}^{G}\boldsymbol{\Pi}$ into the state.

### 3.4.4 Merging Common Plane

Planes are initialized into the state as soon as there are sufficient observations, and they are continually refined with new measurements. Over time, multiple planes may merge into a single common plane to reduce redundancy in the state. If more than one planes reside in the state, a pair-wise state constraint update is performed to enforce equality, then followed by the marginalization of all but one plane. For example, plane ${}^{G}\mathbf{\Pi}_1$ and ${}^{G}\mathbf{\Pi}_2$ have:

$$\mathbf{z}_p = \left({}^{G}\mathbf{\Pi}_2 - {}^{G}\mathbf{\Pi}_1\right) + \mathbf{n}_p \tag{3.26}$$

where $\mathbf{n}_p$ is the small noise that softens the constraint [181] (we used 1cm throughout the experiments). We are able to explicitly detect when we should check if two planes should be merged in the state since during feature tracking we can find this information. If a feature which has a unique plane id matches to a plane with an older plane id, then we will convert all points for the current plane to share the same plane id as the old plane. In addition to performing a conservative chi-squared check with the above constraint equation, we additionally check that the normal of the two planes are within 1 degree of each other. Both of these checks ensure that incorrect feature matching does not degrade the estimator's performance.

### 3.4.5 Planar Point Feature Update

Given the planar point feature linearized measurement function, Eq. (3.19), we will explain in detail how to process measurements with in-state or out-of-state features. We then consider the following update methods:

- *SLAM Plane + SLAM Point:* Standard EKF update
- *SLAM Plane + MSCKF Point:* Remove the point feature dependency through nullspace projection [see Eq. (3.23)].
- *MSCKF Plane + SLAM Point:* Remove the plane feature dependency by:

$$\mathbf{N}_\pi^\top \tilde{\mathbf{z}}_A = \mathbf{N}_\pi^\top \mathbf{H}_T \tilde{\mathbf{x}}_T + \mathbf{N}_\pi^\top \mathbf{H}_f {}^{G}\tilde{\mathbf{p}}_f + \mathbf{N}_\pi^\top \mathbf{n} \tag{3.27}$$

Table 3.3: Simulation parameters and prior standard deviations that perturbations of measurements were drawn from.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Gyro. White Noise | 1.6968e-04 | Gyro. Rand. Walk | 1.9393e-05 |
| Accel. White Noise | 2.0000e-3 | Accel. Rand. Walk | 3.0000e-3 |
| Cam Freq. (Hz) | 10 | IMU Freq. (Hz) | 400 |
| Num. Clones | 11 | Total Planes | 6 |
| Avg. Feats on Plane | 150 | Max SLAM Pts | 15 |

where $\mathbf{N}_\pi$ is the left nullspace of the stacked $\mathbf{H}_\pi$. This requires more than 3 planar point features.

- *MSCKF Plane + MSCKF Point:* Remove the plane and point feature dependency by:

$$\tilde{\mathbf{z}} = \mathbf{H}_T\tilde{\mathbf{x}}_T + \begin{bmatrix}\mathbf{H}_f & \mathbf{H}_\pi\end{bmatrix}\begin{bmatrix}{}^G\tilde{\mathbf{p}}_f \\ {}^G\tilde{\mathbf{\Pi}}\end{bmatrix} + \mathbf{n} \tag{3.28}$$

$$\Rightarrow \mathbf{N}_{f\pi}^\top\tilde{\mathbf{z}} = \mathbf{N}_{f\pi}^\top\mathbf{H}_T\tilde{\mathbf{x}}_T + \mathbf{N}_{f\pi}^\top\mathbf{n} \tag{3.29}$$

where $\mathbf{N}_{f\pi}$ is the left nullspace of $[\mathbf{H}_f \;\; \mathbf{H}_\pi]$. Observing a feature more than three times is necessary.

### 3.4.6 Monte-Carlo Simulations

The proposed `ov_plane` is built as an extension to OpenVINS. We generate a room surrounding the simulation trajectory and visual points lying on the planes, see Figure 3.8. Data associations between points and planes are assumed to be known. Table 5.1 contains the key sensor frequencies, sensor properties, and noise parameters used in the simulation. Errors are reported using the Normalized Estimation Error Squared (NEES), Relative Pose Error (RPE), and Absolute Trajectory Error (ATE) metrics throughout the different experiments (see [221] and [4]).

Results for a 20 run Monte-Carlo are shown in Table 3.4 with different estimator configurations. All algorithms remain consistent as their NEES values are close to three. The M-PT & M-PL, which adds MSCKF planes, has little improvement over

Table 3.4: Average 20 run RPE and NEES for different algorithm configurations. Units are in degrees / cm. A constraint noise of $\sigma_d = 0.001$ was used. M corresponds to MSCKF features (out-of-state), S for SLAM features (in-state), PT for point features, and PL represents plane features.

| **Algorithm** | **60m** | **80m** | **100m** | **120m** | **NEES(3)** |
|---|---|---|---|---|---|
| M-PT | 0.37 / 4.3 | 0.44 / 5.0 | 0.50 / 5.6 | 0.55 / 6.2 | 3.39 / 1.75 |
| M-PT & M-PL | 0.37 / 4.3 | 0.43 / 4.9 | 0.48 / 5.5 | 0.53 / 6.1 | 3.34 / 1.72 |
| M-PT & MS-PL | **0.36** / **3.6** | **0.42** / **4.1** | **0.48** / **4.6** | **0.53** / **5.1** | 3.99 / 1.44 |
| MS-PT | 0.30 / 3.6 | 0.35 / 4.1 | 0.40 / 4.6 | 0.43 / 5.1 | 3.45 / 1.63 |
| MS-PT & M-PL | 0.29 / 3.5 | **0.33** / 4.0 | **0.37** / 4.5 | **0.41** / 4.9 | 3.09 / 1.44 |
| MS-PT & MS-PL | **0.29** / **2.9** | 0.35 / **3.3** | 0.39 / **3.7** | 0.42 / **4.1** | 3.38 / 1.20 |

the baseline M-PT system. We attribute this to the MSCKF plane track length only being that of the sliding window size and the regularity does not improve MSCKF point linearizations by much. But, if the planes with sufficient observations are inserted into the state vector, M-PT & MS-PL, then a clear performance gain can be seen for all trajectory lengths. Within the simulation, the ceiling and floor can be tracked over significant portions of the trajectory allowing for improved feature triangulation and leveraging of the structural regularity information.

Next, we investigate the impact of co-estimating SLAM points and planes. The baseline MS-PT is more accurate than the M-PT as expected, but it is interesting to see that the M-PT & MS-PL is able to perform near the level of accuracy with only estimating, at maximum, six environmental planes alongside MSCKF point features. Again the MSCKF plane, MS-PT & M-PL, has little impact on accuracy over the point-only MS-PT, while the addition of SLAM plane estimation in MS-PT & MS-PL has the overall best accuracy. These simulation results demonstrate the improved VIO performance with planar regularities for both in-state SLAM and out-of-state MSCKF point features.

### 3.4.7 Real-World Experiments

We evaluate the proposed system on the Vicon room scenarios from the EuRoC MAV dataset [14] which provides 20Hz stereo images, 200Hz ADIS16448 MEMS IMU

measurements, and optimized groundtruth trajectories. We do not evaluate the machine hall scenarios due to their cluttered environment and lack of planar structures. An additional AR table dataset was collected as an example scenario in which a user walks around a central table.[2] An Intel Realsense D455[3] with 30Hz RGB-D (depth was not used) and 400Hz BMI055 IMU along with 100Hz OptiTrack poses were recorded in 1-2 minute segments. The groundtruth was recovered using the `vicon2gt` utility (see Section 2.3.5 and [61]). We extract 200 sparse point features and keep a maximum of 15 SLAM point features in MS-PL.[4] Two additional state-of-the-art visual-inertial systems, VINS-Fusion [161] and OKVIS [110], are evaluated in addition to OpenVINS [60], MS-PT, and the proposed `ov_plane` extensions.[5] All methods are run without loop-closure, with a monocular camera and IMU as input, and with spatial-temporal calibration if supported.

#### 3.4.7.1 Plane Detection and Tracking Performance

Details of the plane extraction are summarized in Algorithm 2, and an example extraction with recovered normals can be seen in the bottom of Figure 3.8. From a high level, we first perform sparse temporal point feature tracking which provides frame-to-frame matching knowledge. The 3D position of point features are recovered efficiently in the global frame by incrementing their information at each timestep. We then recover a sparse 3D geometric mesh of the environment which is used to recover per-feature normals. A pairwise comparison with a series of heuristics is used to finally cluster points into common planes.

---

[2] https://github.com/rpng/ar_table_dataset

[3] https://www.intelrealsense.com/depth-camera-d455/

[4] All computational results were performed in a single thread on an Intel(R) Xeon(R) E3-1505Mv6 @ 3.00GHz.

[5] Note that we have tried to reproduce the results of [164–166] for a fair comparison, but were unable to achieve sufficient accuracy on their 2019 v4.0 code release. The latest main branch no longer supports the use of structural regularities.

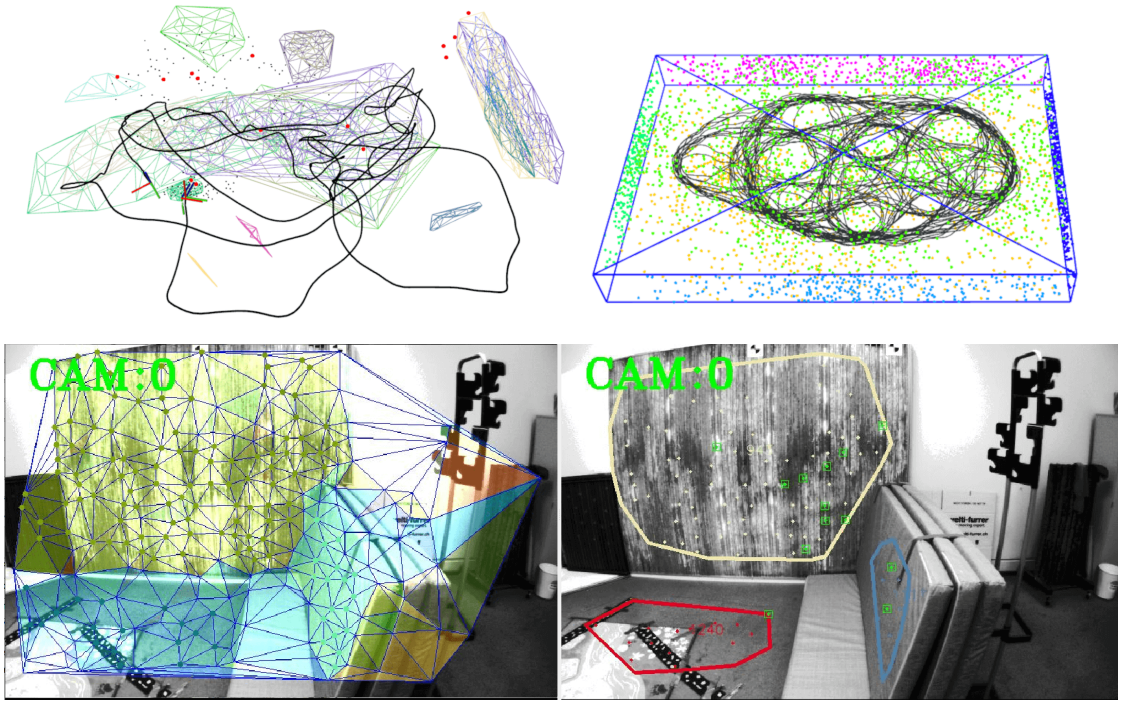



Figure 3.8: EuRoC MAV [14] with estimated planes shown as meshes (not all in the state vector, top left). Simulation environment (top, right) has a 1.2km trajectory in a 15.2 × 9.5 × 1.7m room (points are colored by plane). Bottom row shows V1_01 sparse tracking mesh with normals (left), and extracted planes (right).

**Algorithm 2** Plane Detection and Tracking

**Sparse Point Features:**

- FAST [167] detection with KLT optical flow [121]
- Robustified with 8-point RANSAC
- Provides frame-to-frame plane tracking

**Point Feature Preprocessing:**

- Point features are incrementally triangulated into 3D if sufficient observations
- Delaunay triangulation of valid features to determine spatial relationships [2, 3, 217]
- Each triangle's normal is computed using its three points:
  ${}^{G}\mathbf{v}_i = \text{normalize}({}^{G}\mathbf{p}_i - {}^{G}\mathbf{p}_0)$
  ${}^{G}\mathbf{n}_j = \text{normalize}(\lfloor {}^{G}\mathbf{v}_2 \times \rfloor \, {}^{G}\mathbf{v}_1)$

**Vertex Normals:**

- Vertex normals of connected triangles are collected
- Compute angle variance and max angle difference between normals
  $\theta = \text{acosd}({}^{G}\mathbf{n}_i^{\top} {}^{G}\mathbf{n}_j)$
- If either is above a threshold, reject this vertex as being on the "edge" of two planes
- Else average normal is computed and vertex is valid

**Vertex Matching Heuristics:**

- For each valid vertex, $i$, compare its neighbors
- Normal difference: $\text{acosd}({}^{G}\mathbf{n}_i^{\top} {}^{G}\mathbf{n}_j) < \Delta\theta$
- Point-to-plane distance: ${}^{G}\mathbf{n}_i^{\top} {}^{G}\mathbf{p}_j - {}^{G}d_i < \Delta d_z$
- Avg. distance $d_i$ of point ${}^{G}\mathbf{p}_i$ to $N$ closest points [15] passes plane Z-test:
  $(d_i - \bar{d})/\sigma_d < z$

**Plane Merging / ID Management:**

- For all vertexes matched, select the smallest (oldest) plane id and assign it to all
- If no feature has a plane id (from the previous frame or match), then assign a new id

Table 3.5: Tracking statistics and time to perform plane tracking (i.e., it does not include sparse point tracking). Statistics include: features per plane, average plane per frame, average plane tracking length, and active planes in the state per frame.

| **Dataset** | **Feat. / PL** | **PL / Frame** | **Track Len.** | **PL Active** | **Time (ms)** |
|---|---|---|---|---|---|
| V1_01 | 19.6 ± 13.3 | 2.9 ± 1.3 | 53.4 ± 74.0 | 0.9 ± 0.7 | 3.3 ± 0.7 |
| V1_02 | 13.7 ± 10.9 | 1.7 ± 1.3 | 20.0 ± 26.8 | 0.3 ± 0.5 | 2.5 ± 0.8 |
| V1_03 | 10.1 ± 9.4 | 0.7 ± 1.0 | 24.9 ± 26.0 | 0.0 ± 0.2 | 2.0 ± 0.7 |
| V2_01 | 8.0 ± 5.0 | 1.4 ± 1.3 | 39.9 ± 43.1 | 0.1 ± 0.3 | 2.5 ± 0.6 |
| V2_02 | 9.5 ± 8.1 | 1.0 ± 1.1 | 23.3 ± 22.8 | 0.0 ± 0.1 | 2.1 ± 0.6 |
| V2_03 | 6.3 ± 1.8 | 0.2 ± 0.4 | 14.4 ± 15.0 | 0.0 ± 0.0 | 1.4 ± 0.6 |
| table_01 | 27.3 ± 13.1 | 2.7 ± 1.1 | 61.1 ± 227.6 | 1.1 ± 0.5 | 3.5 ± 0.7 |
| table_02 | 82.0 ± 58.7 | 2.2 ± 1.3 | 49.1 ± 249.2 | 1.2 ± 0.6 | 4.1 ± 0.9 |
| table_03 | 33.9 ± 21.3 | 3.0 ± 1.2 | 88.5 ± 337.4 | 1.5 ± 0.6 | 4.0 ± 0.7 |
| table_04 | 35.3 ± 23.1 | 2.1 ± 0.9 | 68.6 ± 428.0 | 0.9 ± 0.4 | 4.2 ± 1.3 |
| table_05 | 38.6 ± 27.6 | 2.5 ± 1.0 | 119.2 ± 327.2 | 1.2 ± 0.7 | 3.5 ± 0.6 |
| table_06 | 43.5 ± 30.5 | 2.0 ± 0.9 | 69.3 ± 131.6 | 1.1 ± 0.8 | 3.2 ± 0.8 |
| table_07 | 16.6 ± 8.2 | 2.8 ± 0.9 | 106.8 ± 163.8 | 0.3 ± 0.5 | 3.0 ± 0.6 |
| table_08 | 20.7 ± 13.5 | 1.8 ± 1.0 | 54.1 ± 260.1 | 0.6 ± 0.5 | 2.7 ± 0.6 |

We ran the proposed plane tracker on a series of datasets and summarized its statistics in Table 3.5. In datasets with high dynamic motions, it can be challenging to extract planes because of poor sparse point feature tracking and dynamic movement preventing sufficient observations for feature triangulation. In particular, the V1_03, V2_02, and V2_03 datasets have very dynamic motions which are not amendable for uniform point feature extraction and a large number of sufficiently observed planar features. It can also be noticed that these datasets have a very low number of features per plane, limiting the number of possible SLAM planes (and on some datasets no planes are used in an update). The additional computational cost for plane detection and matching is around 2-4ms, which is similar to sparse point feature tracking (around 3-4ms).

On the self-collected AR table datasets, we observed that due to the larger planar surfaces and longer view time, planes can be sufficiently tracked for long periods of time with a high number of features per plane. This is amendable for leveraging structural regularities. In addition, it is not possible to take advantage of environmental

Table 3.6: EuRoC MAV ATE (degree / cm) along with average timing for the V1_01_easy dataset. $\sigma_d = 0.01$ was used.

| Algorithm | V1_01 | V1_02 | V1_03 | V2_01 | V2_02 | V2_03 | Time (ms) |
|---|---|---|---|---|---|---|---|
| M-PT | 0.83 / 8.6 | 1.57 / 9.1 | 2.50 / 15.5 | 1.73 / 12.1 | 1.34 / 9.4 | 1.61 / 15.6 | 8.3 ± 1.7 |
| M-PT & M-PL | 0.82 / 8.6 | 1.58 / 9.2 | **2.45** / **15.3** | 1.73 / 12.1 | **1.22** / 9.7 | 1.61 / 15.6 | 12.2 ± 2.7 |
| M-PT & MS-PL | **0.75** / **7.6** | **1.55** / **9.0** | 2.50 / 15.5 | 1.73 / 12.1 | 1.28 / **8.8** | 1.61 / 15.6 | 12.4 ± 2.7 |
| MS-PT | 1.32 / 8.4 | 1.58 / 7.0 | **2.20** / **12.2** | **0.80** / 11.3 | 1.96 / 8.3 | 1.77 / 16.9 | 9.0 ± 2.0 |
| MS-PT & M-PL | **0.61** / **5.3** | 1.58 / 7.5 | 2.32 / 12.5 | 0.89 / 12.5 | 1.93 / 7.4 | 1.77 / 16.9 | 13.9 ± 3.8 |
| MS-PT & MS-PL | 0.75 / 6.9 | **1.55** / **6.9** | 2.41 / 12.5 | 0.82 / **10.8** | **1.40** / **6.8** | 1.77 / 16.9 | 13.8 ± 3.4 |
| VINS-Fusion [161] | 1.24 / 5.8 | 2.61 / 11.5 | 3.61 / 20.5 | 1.99 / 8.0 | 3.13 / 8.7 | 3.54 / 19.7 | 31.9 ± 12.3* |
| OKVIS [110] | 0.72 / 8.3 | 2.01 / 14.5 | 10.47 / 107.4 | 0.94 / 13.4 | 1.17 / 19.1 | 2.37 / 23.3 | 59.9 ± 31.6* |

* Timing for VINS-Fusion [161] and OKVIS [110] *only* reports their optimization time (no feature tracking).

white walls since no visual point features are extracted to facilitate plane detection. Thus extraction of planes remains limited to regions with sufficient texture.

#### 3.4.7.2 EuRoC MAV Indoor Dataset Trajectory Accuracy

Table 3.6 shows the average ATE over each dataset for different configurations. Looking first at M-PT, on the V1_01 dataset there is a clear advantage to including SLAM plane features in the state (see Figure 3.8 for extracted planes). The use of MSCKF planes seems to show the same performance without planes, mirroring the simulation results. For most datasets with limited plane extraction, see Table 3.5 planes per frame, there is very little improvement over point-based VIO. In general, the OpenVINS-based systems demonstrate superior computational efficiency and outperform other state-of-the-art methods.

When SLAM point features are included, MS-PT, the performance gains between point-based and plane-aided become smaller. There can even be cases where the use of planes can hurt performance, which we equate to SLAM point features being more sensitive to incorrect data associations due to their length of time in the state. The system is able to perform well above the real-time threshold of 50ms, with the increase in computation mainly coming from plane detection and matching.

Table 3.7: Self-collected AR table ATE (degree / cm) and average timing for the table_01 dataset. $\sigma_d = 0.01$ was used.

| Algorithm | table_01 | table_02 | table_03 | table_04 | table_05 | table_06 | table_07 | table_08 | Time (ms) |
|---|---|---|---|---|---|---|---|---|---|
| M-PT | **0.45** / 6.8 | 0.85 / 2.4 | 1.37 / 5.6 | 0.83 / 7.5 | 0.78 / 5.0 | **0.66** / 4.9 | 0.94 / 4.8 | 2.00 / 12.5 | 8.7 ± 1.7 |
| M-PT & M-PL | 0.52 / 6.5 | 0.91 / 2.5 | 1.44 / 5.9 | 0.87 / 7.1 | 0.76 / 4.9 | 0.67 / 5.9 | **0.85** / 4.7 | 2.02 / 12.8 | 13.3 ± 3.2 |
| M-PT & MS-PL | 0.67 / **4.6** | **0.72** / **2.0** | **0.96** / **3.0** | **0.75** / **3.2** | **0.62** / **4.0** | 0.75 / **4.4** | 0.92 / **4.2** | **1.88** / **9.2** | 13.9 ± 2.9 |
| MS-PT | **1.15** / 5.7 | 1.79 / 4.1 | 2.41 / 6.9 | 1.28 / 5.7 | **0.56** / 2.7 | **0.78** / **3.6** | **1.00** / 4.8 | **0.68** / 11.2 | 9.4 ± 2.0 |
| MS-PT & M-PL | 1.32 / 5.5 | 0.89 / 2.5 | **1.03** / **4.5** | 1.10 / **4.7** | 1.01 / 4.4 | 1.81 / 6.0 | 1.06 / **4.6** | 1.29 / 11.2 | 15.0 ± 3.9 |
| MS-PT & MS-PL | 1.25 / **5.1** | **0.65** / **2.3** | 1.05 / 4.6 | **0.79** / 5.0 | 0.70 / **2.6** | 1.29 / 4.5 | 1.12 / 5.1 | 0.82 / **6.8** | 14.7 ± 3.2 |
| VINS-Fusion [161] | 1.62 / 5.8 | 1.32 / 3.0 | 1.47 / 7.6 | 1.75 / 5.6 | 1.12 / 3.4 | 0.98 / 5.3 | 1.67 / 9.3 | 5.03 / 23.3 | 35.6 ± 17.0* |
| OKVIS [110] | 2.48 / 9.0 | 2.01 / 7.7 | 3.94 / 15.3 | 2.05 / 16.2 | 0.77 / 24.5 | 0.74 / 10.2 | 2.07 / 13.8 | 1.54 / 19.8 | 85.5 ± 32.6* |

* Timing for VINS-Fusion [161] and OKVIS [110] *only* reports their optimization time (no feature tracking).

#### 3.4.7.3 AR Table Dataset Trajectory Accuracy

The ATE for the self-collected AR table dataset is shown in Table 3.7. Looking at M-PT, it is clear that there is a significant improvement of 1-3cm of accuracy when planar regularities are used. The table or floor planes are typically tracked over large segments of the trajectory, see Table 3.5 average track length, and thus provide a long-term loop-closure for all points. The planes' large spatial volume also allows for more accurate feature triangulation, possibly reducing linearization errors. When SLAM point features are added, there is still a gain of accuracy on most datasets, but there are a few where planar constraints can hurt performance. We additionally see that the use of MSCKF plane features has little impact both in real-world experiments and simulations thus we do not recommend their use as a regularization source.

### 3.5 Summary

In this thrust, we have presented the Closest Point (CP) plane representation and its application to an optimization-based LiDAR-inertial system and filter-based monocular VIO system. This novel plane representation was shown to have the desirable properties of being a minimal representation with simple vector difference residuals when performing plane-to-plane constraints. Additionally, a method for compressing point-on-plane constraints into this CP representation was developed which allowed for efficient estimation with only a single residual per-plane per-LiDAR scan. The CP representation, within the context of a LiDAR-inertial system, was shown to outperform

the existing minimal plane representation.

We then presented a method for performing planar regularization to sparse 3D points to environmental planes and showed that this provided benefits to VIO performance. A novel detection, extraction, and tracking method was developed to remove the need for an active depth sensor or expensive neural network. This was shown to both enable plane tracking, but also was efficient in nature. Finally, this planar regularity extension of OpenVINS, termed `ov_plane`, was evaluated on two real-world datasets including a self-collected AR demonstration, and showed clear gains in performance when long-lived planar features were available.

# Chapter 4

# EFFICIENT, CONSISTENT, AND PERSISTENT FILTER-BASED VISUAL-INERTIAL MAPPING

## 4.1 Introduction

As compared to the previously mentioned plane-aided structural VIO, we now focus on the goal of bounding trajectory drift by performing persistent filter-based VI-SLAM. It holds great potential to enable centimeter-accuracy positioning of VI-SLAM on smart phones, micro air vehicles (MAVs), augmented or virtual realtity (VR) to enable unique abilities and experiences. We additionally focus on how we can do this *consistency* and also provide an uncertainty metric, e.g. a covariance, of our current VI-SLAM state estimation while still remaining computationally efficient. At the core we leveraged the previously mentioned OpenVINS framework and extend it to incorporate historical states. We will show that a naive extension to keep additional historical states to loop-closure to, which bounds trajectory drift, is detrimental for estimator complexity due to the typically $O(n^3)$ cost in terms of the size of the state. To solve this, we proposed a methodology which leverages the Schmidt-Kalman filter (SKF) [177] to reduce the cost of maintaining these old states while still tracking the correlations enabling consistent estimation. We first present a system which builds and keeps a point-feature map and leverages 2D-to-3D loop-closure map constraints. We show that it can both simultaneously build the map and leverage it to bound the long-term drift in both simulation and real-world datasets. We then present a second system which instead keeps track of historical poses and leverages a 2D-to-2D loop-closure constraint between these historical poses and features tracked in the current sliding window. We show that this method is both able to constraint the long-term drift, but also achieve high efficiency.

Having presented two different methods for leveraging the SKF, we then focused on comparing and contrasting each of these in terms of their complexity, memory, and accuracy. The problem is simplified to the prior map-based estimator where the map and the features (or keyframes) mean and uncertainties are provided a priori. We additionally investigate a series of different *inconsistent* methods typically used in the literature which do not estimate the correlation with the prior map and simply perform inflation of the measurement noise to compensate. Armed with the insight that the SKF can enable consistent performance with high accuracy in small workspaces, the 2D-to-2D model can reduce complexity at a reduction in loop-closure constraint strength, and the inflation methods can be relatively invariant to their inflation parameters *if* the map covariance is *known*, we then focused on how to leverage the best of each. We finally look at how to scale the system to a larger scale multi-room.

As compared to the previously developed filter-based system, we next proposed coupling the filter-based system with a non-linear backend which supports *relinearization* which can correct for large map inconsistencies after loop-closure. To the best of our knowledge this is the first work to systematically study a hybrid filter-based estimator in conjunction with a lightweight relative-pose graph optimization which can perform relinearization in a consistent manner resulting in a map which can be directly leveraged through the inflation-based measurement model. Additionally, we present a novel method termed "dynamic Schmidt'ing" which enables moving the map state from the SKF state into the EKF during loop-closures re-observations to allow for their estimates to improve and uncertainties to decrease as compared to just fixing the estimates. This is shown to have a large accuracy improvement with very little computational cost penalty. The hybrid extension of the MSCKF to map-based localization using different techniques is investigated through a series of detailed numerical simulation experiments to demonstrate its real-time localization accuracy and efficiency.

## 4.2 Estimation Background

We now present some key background on the SKF and the common VIO measurement equations which will then be related to the map-based update in the later sections. A detailed complexity analysis is presented in Appendix E and our technical report [58] for all EKF operations, and shows that all are linear in terms of the number of map points.

### 4.2.1 Schmidt-Kalman State and Covariance Propagation

Consider the standard EKF which jointly estimates both the active and map states at timestep $t_{k-1}$.

$$\mathbf{x}_{k-1} = \begin{bmatrix} \mathbf{x}_{A_{k-1}} \\ \mathbf{x}_{S_{k-1}} \end{bmatrix}, \quad \mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{AA_{k-1}} & \mathbf{P}_{AS_{k-1}} \\ \mathbf{P}_{SA_{k-1}} & \mathbf{P}_{SS_{k-1}} \end{bmatrix} \tag{4.1}$$

The propagated state $\mathbf{x}_k$ can be directly computed while the covariance can be propagated forward in time via the following equations:

$$\mathbf{P}_{k|k-1} = \begin{bmatrix} \mathbf{\Phi}_{k-1}\mathbf{P}_{AA_{k-1|k-1}}\mathbf{\Phi}_{k-1}^\top & \mathbf{\Phi}_{k-1}\mathbf{P}_{AS_{k-1|k-1}} \\ \mathbf{P}_{SA_{k-1|k-1}}\mathbf{\Phi}_{k-1}^\top & \mathbf{P}_{SS_{k-1|k-1}} \end{bmatrix} + \begin{bmatrix} \mathbf{Q}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{4.2}$$

where one can see that the most computational expensive $\mathbf{\Phi}_{k-1}\mathbf{P}_{AS_{k-1|k-1}}$ will be linear in complexity in terms of $\mathbf{x}_M$ since the map states do not evolve with time.

### 4.2.2 Schmidt-Kalman State and Covariance Update

Given a non-linear measurement model Eq. (2.52), we will now consider how to perform an update given this stacked feature observation. We have the following linear system which is a function of the active and map state.

$$\mathbf{r}_f \simeq \mathbf{H}_{A_k}\tilde{\mathbf{x}}_{A_k} + \mathbf{H}_{S_k}\tilde{\mathbf{x}}_{S_k} + \mathbf{n}_f \tag{4.3}$$

where the measurement noise is $\mathbf{n}_f \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_f)$. This gives the Kalman gain as follows:

$$\mathbf{K}_k = \begin{bmatrix} \mathbf{K}_{A_k} \\ \mathbf{K}_{M_k} \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{AA_{k|k-1}}\mathbf{H}_{A_k}^\top + \mathbf{P}_{AS_{k|k-1}}\mathbf{H}_{S_k}^\top \\ \mathbf{P}_{SA_{k|k-1}}\mathbf{H}_{A_k}^\top + \mathbf{P}_{SS_{k|k-1}}\mathbf{H}_{S_k}^\top \end{bmatrix} \mathbf{S}_k^{-1} \tag{4.4}$$

where $\mathbf{H}_k = [\mathbf{H}_{A_k}\ \mathbf{H}_{S_k}]$ and $\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\top + \mathbf{R}_f$ is the measurement residual innovation. We can thus define the following standard EKF mean and covariance update equations:

$$\hat{\mathbf{x}}_{A_{k|k}} = \hat{\mathbf{x}}_{A_{k|k-1}} + \mathbf{K}_{A_k}\mathbf{r}_f \tag{4.5}$$

$$\hat{\mathbf{x}}_{S_{k|k}} = \hat{\mathbf{x}}_{S_{k|k-1}} + \mathbf{K}_{S_k}\mathbf{r}_f \tag{4.6}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \begin{bmatrix} \mathbf{K}_{A_k}\mathbf{S}_k\mathbf{K}_{A_k}^\top & \mathbf{K}_{A_k}\mathbf{H}_k \begin{bmatrix} \mathbf{P}_{AS_{k|k-1}} \\ \mathbf{P}_{SS_{k|k-1}} \end{bmatrix} \\ \begin{bmatrix} \mathbf{P}_{AS_{k|k-1}} \\ \mathbf{P}_{SS_{k|k-1}} \end{bmatrix}^\top \mathbf{H}_k{}^\top \mathbf{K}_{A_k}^\top & \mathbf{K}_{S_k}\mathbf{S}_k\mathbf{K}_{S_k}^\top \end{bmatrix} \tag{4.7}$$

We note that this process is $O(x^2)$ complexity in terms of the map if the number of measurements is far smaller than the state size (i.e., $\mathbf{S}_k{}^{-1}$ is cheap) due to the covariance update multiplication $\mathbf{K}_{S_k}\mathbf{S}_k\mathbf{K}_{S_k}^\top$, which is undesirable since the size of the map can grow unbounded.

We now look at a consistent alternative Schmidt-Kalman filter (SKF) [177] update which has been successfully used (along with different variations) to reduce the update complexity for map-based localization [34, 59, 65, 79, 93]. Using the same state definition as in Eq. (4.1), we set $\mathbf{K}_{S_k} = \mathbf{0}$ and get the following update equations (see Eq. (4.6) and (4.7)):

$$\hat{\mathbf{x}}_{A_{k|k}} = \hat{\mathbf{x}}_{A_{k|k-1}} + \mathbf{K}_{A_k}\mathbf{r}_f \tag{4.8}$$

$$\hat{\mathbf{x}}_{S_{k|k}} = \hat{\mathbf{x}}_{S_{k|k-1}} \tag{4.9}$$

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \begin{bmatrix} \mathbf{K}_{A_k}\mathbf{S}_k\mathbf{K}_{A_k}^\top & \mathbf{K}_{A_k}\mathbf{H}_k \begin{bmatrix} \mathbf{P}_{AS_{k|k-1}} \\ \mathbf{P}_{SS_{k|k-1}} \end{bmatrix} \\ \begin{bmatrix} \mathbf{P}_{AS_{k|k-1}} \\ \mathbf{P}_{SS_{k|k-1}} \end{bmatrix}^\top \mathbf{H}_k{}^\top \mathbf{K}_{A_k}^\top & \mathbf{0} \end{bmatrix} \tag{4.10}$$

This process is $O(x)$ and its memory requirements for landmark-based and keyframe based maps are $O((3m)^2)$ and $O((6n)^2)$, respectively. This is due to only updating the

cross-covariance terms. We note that this is a *consistent* approximation which ensures that the filter is never more confident than the original EKF (see [34]). Where we define consistency as an estimator is consistent when its errors are zero-mean (unbiased) and the covariance matrix is equal to that reported by the estimator [4, Section 5.4].

### 4.3 SEVIS-3D: Schmidt-EKF based VI-SLAM with 3D Points

It is known that the EKF update of state estimates and covariance has quadratic complexity in terms of the number of map features [155], making naive implementations of VI-SLAM too expensive to run in real-time. Leveraging the SKF [177], we propose a novel **S**chmidt-**E**KF for **VI**-**S**LAM with 3D map points (SEVIS-3D) algorithm which mitigates this quadratic complexity. The key idea is to selectively treat 3D map features as nuisance parameters in the state vector [i.e., Schmidt map state $\mathbf{x}_S$ Eq. (4.1)] whose mean and covariance will no longer be updated, while their cross-correlations with the active state $\mathbf{x}_A$ (e.g., current pose, calibration, etc.) are still utilized and updated. The specific state we estimate is:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_C^\top & \mathbf{x}_M^\top & | & {}^G\mathbf{p}_{f_1}^\top & \cdots & {}^G\mathbf{p}_{f_M}^\top \end{bmatrix}^\top =: \begin{bmatrix} \mathbf{x}_A^\top & \mathbf{x}_S^\top \end{bmatrix}^\top \tag{4.11}$$

$$\mathbf{x}_A = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_C^\top & \mathbf{x}_M^\top \end{bmatrix}^\top \tag{4.12}$$

Note we have omitted calibration states, see the OpenVINS state in Section 2.2 for the full definition of $\mathbf{x}_A$.

In what follows we primarily focus on the loop-closure details and update method for monocular images, which is at the core of our SEVIS-3D, but the approach is easily extendable to stereo systems. As the camera-IMU sensor pair moves through the environment, features are tracked using descriptor-based tracking. FAST features are first detected [168] and ORB descriptors [171] are extracted for each. The OpenCV [149] "BruteForce-Hamming" KNN descriptor matcher is used to find correspondences, after which we perform both a ratio test between the top two returns to ensure valid matches and 8-point RANSAC to reject any additional outliers. Once visual tracks are found, three types of tracked features are used to efficiently update

state estimates and covariance: (i) VIO features that are opportunistic and can only be tracked for a short period of time, (ii) SLAM features that are more stable than the above one and can be tracked beyond the current sliding window, and (iii) map features that are the matured and informative SLAM features which are kept in the Schmidt state for an indefinite period of time.

### 4.3.1 Loop-Closure to Historical 3D Points

To overcome the data association challenge, given 3D positions of map features already included in the state vector, one straightforward approach might be through 3D-to-2D projection (i.e., projecting the 3D map feature onto the current frame) to find the correspondence of current visual measurements to the mapped feature, which is often used in the literature (e.g., [34, 126]). However, in a typical SLAM scenario, estimating a map of 3D point features and matching them to current features is often sparse; for example, we found that it was common for a multi-floor indoor environment with up to 600 map features to only have about 10 features that can successfully project back into the active frame. Moreover, if there is any non-negligible drift in the state estimates (which is inevitable in practice), then projected features are likely to not correspond to the same spatial area as the current image is observing, thus preventing the utilization of map information to reduce navigation errors.

For these reasons, we advocate 2D-to-2D matching for data association with the aid of "keyframes" that observe previous areas in the environment, due to its ability to provide high quality estimates and not be affected by estimation drift. Each keyframe contains a subset of the extracted features that correspond to map features in the state vector, and thus, if we match active feature tracks to previous keyframes we can find the correspondence between the newly tracked features and the previously mapped features that reside in our state. Specifically, we first query the keyframe database to retrieve the closest keyframe to the current frame. To this end, different place recognition approaches such as DBoW2 [53] and CALC [136] can be used to find the best candidate. After retrieval, we perform an additional geometric check by

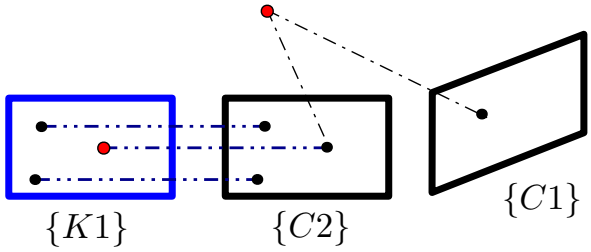



Figure 4.1: Illustration of the proposed keyframe-aided 2D-to-2D matching for data association from a 2D observation to a 3D point map feature. Assuming a cloned frame $\{C2\}$ matches to a keyframe $\{K1\}$ with all actively tracked features, and among these positive matches, one feature (red) corresponds to a map feature, the measurements in $\{C2\}$ and $\{C1\}$ will be used to update the active state by performing Schmidt-EKF update.

ensuring that the fundamental matrix can be calculated between the current frame and the proposed keyframe match, which we found provided extremely good matches to the best keyframe in the database. After retrieving a matching keyframe, we perform descriptor-based matching from features in the current frame to the keyframe with *all* extracted features from both frames followed by 8-point RANSAC to reject outliers. We now have the correspondence between the current frame feature tracks and keyframe map features. Figure 4.1 visualizes this process.

### 4.3.2 Map with 3D Features: Schmidt-EKF Update

As SLAM features are inserted into $\mathbf{x}_M$ they would typically remain until they have lost track. As compared to this, they will instead be moved to the Schmidt state $\mathbf{x}_S$ as nuisance parameters. The main steps of the proposed SEVIS-3D are outlined in Algorithm 3. Having matched a current set of feature observations to a historical map feature, we can then apply the Schmidt-Kalman update from Section 4.2 which will efficiently update our state. This update is identical to the SLAM feature update, but the position of the feature and its marginal uncertainty will not be updated. The cross-correlations will be updated such that the estimator remains consistent, thus balancing the trade-off of being unable to refine historical features with improved accuracy and robustness through consistent estimation.

**Algorithm 3** Schimdt-EKF Visual-Inertial SLAM with 3D Points (SEVIS-3D)

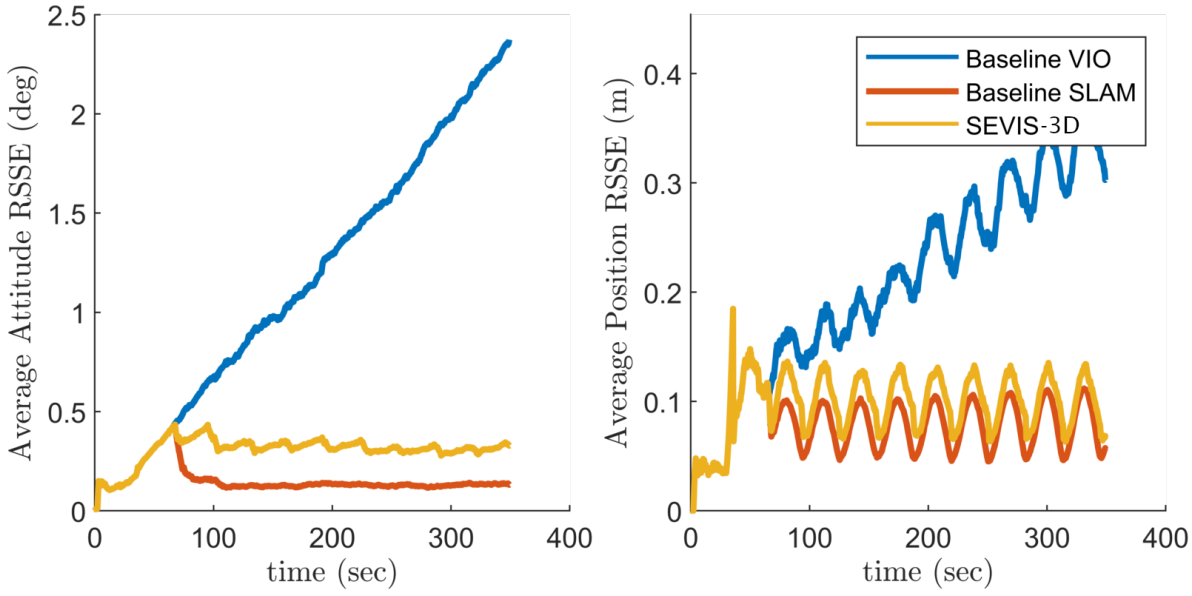
**Propagation**: Propagate the IMU navigation state estimate $\hat{\mathbf{x}}_{I_{k|k-1}}$ based on (2.26), the active state's covariance $\mathbf{P}_{AA_{k|k-1}}$ and cross-correlation $\mathbf{P}_{AS_{k|k-1}}$ based on (4.2).

**Update**: For an incoming image,

- Perform stochastic cloning [170] of current state.
- Track features into the newest frame.
- Perform keyframe-aided 2D-to-2D matching to find map feature correspondences:
  - Query keyframe database for a keyframe visually similar to the current frame.
  - Match currently active features to the features in the keyframe.
  - Associate those active features with mapped features in the keyframe.
- Perform MSCKF update for VIO features (i.e., those that have lost their tracks) as in Section 2.3.3.2.
- Initialize new SLAM features if needed and perform EKF update.
- Perform Schmidt-EKF update for map features as in Section 4.2.2.

**Management of Features and Keyframes**:

- Active SLAM features that have lost track are moved to the Schmidt state or marginalized out.
- Marginalize the oldest cloned pose from the sliding window state.
- Marginalize map features if exceeding the maximum map size.
- Insert a new keyframe into the database if we have many map features in the current view.
- Remove keyframes without map features in view.




(a) Orientation RSSE

(b) Position RSSE

Figure 4.2: Monte-Carlo simulation averaged RSSE of pose (position and orientation) estimates for the three considered VIO and VI-SLAM algorithms.

### 4.3.3 Monte-Carlo Simulation Results

To validate the back-end estimation engine of the proposed SEVIS-3D, we first perform Monte-Carlo simulations of visual-inertial SLAM with known measurement-feature correspondences, where a monocular-visual-inertial sensor platform is moving on a circular trajectory within a cylinder arena observing a series of environmental features. The simulation parameters are listed in Table 4.1.

In particular, we compare three VINS algorithms to reveal the benefits of the proposed SEVIS-3D: (i) The baseline VIO approach consists of the MSCKF augmented with 6 SLAM features (see [113]). These SLAM features are explicitly marginalized when they leave the field of view. (ii) The baseline SLAM method uses the same MSCKF window but is augmented with 90 SLAM features. Different from the above VIO, in this case the SLAM features are never marginalized so that they can be used for (implicit) loop-closures. (iii) The proposed SEVIS algorithm, which consists of the same MSCKF window and 6 SLAM features as in the baseline VIO, is augmented with a bank of 90 map features that are modeled as nuisance parameters. When the SLAM

Table 4.1: Monte-Carlo Simulation Parameters for SEVIS-3D

| Parameter | Value | Units |
|---|---|---|
| IMU Angle Random Walk Coeff. | 0.4 | deg/$\sqrt{\text{Hr}}$ |
| IMU Rate Random Walk Coeff. | 0.02 | deg/sec/$\sqrt{\text{Hr}}$ |
| IMU Velocity Random Walk Coeff. | 0.03 | m/sec/$\sqrt{\text{Hr}}$ |
| IMU Acceleration Random Walk Coeff. | 0.25 | milli-G/$\sqrt{\text{Hr}}$ |
| IMU Sample Rate | 100 | Hz |
| Image Processing Rate | 5 | Hz |
| Feature Point Error $1\sigma$ | 0.17 | deg |
| Number of MSCKF Poses | 15 | |
| Approximate Loop Period | 32 | sec |

features leave the field of view, they are moved into the Schmidt states, becoming the map features as described in Algorithm 3.

The average root sum squared error (RSSE) performance of 50 Monte-Carlo simulation runs are shown in Figure 4.2. As expected, the baseline VIO accumulates drift in both orientation and position over time while the baseline SLAM provides bounded error performance without long-term drift. It is interesting to point out that the position RSSE oscillates slightly depending on the location relative to the initial loop closure. This is because the EKF has limited ability to correct these errors as it cannot re-linearize past measurements unlike optimization-based approaches [188]. More importantly, it is clear that the proposed SEVIS algorithm also does not accumulate long-term drift, although it is slightly less accurate than the baseline SLAM. However, this degradation in accuracy is a small price to pay considering that the SEVIS-3D is of *linear* computational complexity with respect to the number of map features, while the baseline SLAM has *quadratic* complexity.

### 4.3.4 Real-World Experimental Results

We further evaluated the baseline MSCKF-based VIO (without map features), the baseline full VI-SLAM, and the proposed SEVIS-3D on real-world datasets. In what follows, we first examine the estimator accuracy and computational overhead,

Table 4.2: Relative trajectory error for different segment lengths along with the overall absolute trajectory error. Values were computed using [221].

| Dist. | Base. VIO | Base. SLAM | SEVIS-3D | VINS-Mono |
|---|---|---|---|---|
| 123m | 0.383 | 0.102 | 0.111 | 0.184 |
| 247m | 0.645 | 0.099 | 0.108 | 0.238 |
| 370m | 0.874 | 0.104 | 0.123 | 0.325 |
| 494m | 1.023 | 0.095 | 0.121 | 0.381 |
| 618m | 1.173 | 0.107 | 0.139 | 0.425 |
| ATE | 0.779 | 0.121 | 0.128 | 0.323 |

after which the systems are evaluated on a challenging nighttime multi-floor dataset, showing that the proposed SEVIS-3D can robustly be extended to realistic applications.

#### 4.3.4.1 Vicon Loops Dataset

We first validated the proposed system on the Vicon loops dataset [109] that spans **1.2km** in a single room over a 13 minute collection period. A hand-held VI-sensor [148] provides grayscale stereo image pairs and inertial information, while full 6DoF groundtruth is captured using a Vicon motion tracking system at 200 Hz. The maximum number of map features was set to 600 points to ensure real-time performance over the entire trajectory with images inserted into the query keyframe database at 0.5 Hz and a max of 5 SLAM features in the active state at a time. The results presented show three different configurations: (i) the baseline VIO augmented with 5 SLAM features, (ii) the baseline VI-SLAM with 600 SLAM/map features, and (iii) the proposed SEVIS-3D with 600 map features that leverages the Schmidt formulation for computational gains.

We evaluated the proposed method using two different error metrics: Absolute Trajectory Error (ATE) and Relative Error (RE). We point the reader to [221] for detailed definitions of these error metrics. Alongside our baseline and proposed methods, we additionally evaluated VINS-Mono [159, 160] to provide a comparison to a current state-of-the-art method that leverages loop closure information. Shown in Table 4.2 and Figure 4.4, the proposed SEVIS-3D is able to localize with high accuracy and

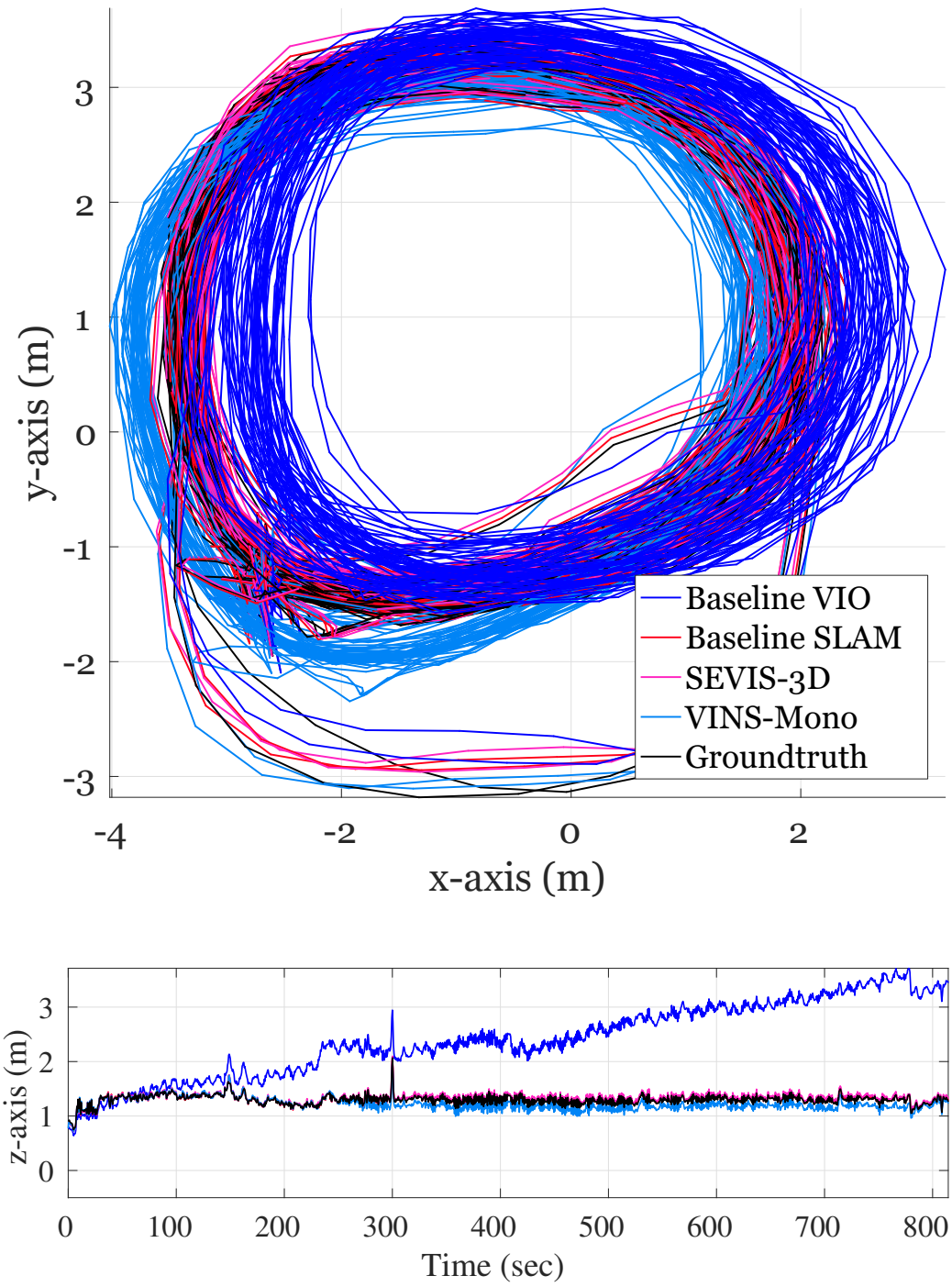



Figure 4.3: Trajectory of the baseline VIO, baseline SLAM with map features, proposed SEVIS-3D with Schmidt covariance update, and VINS-Mono [159, 160]. Clearly the inclusion of map features has limited the drift and allows for high accuracy.

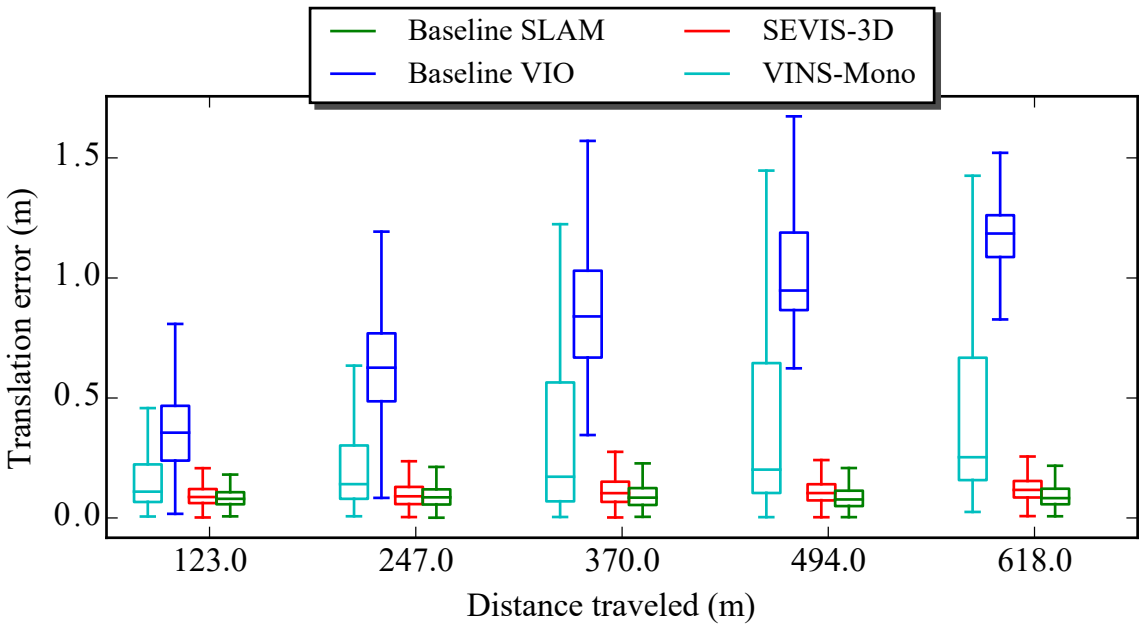



Figure 4.4: Boxplot of the relative trajectory error statistics. The middle box spans the first and third quartiles, while the whiskers are the upper and lower limits. Plot is best seen in color.

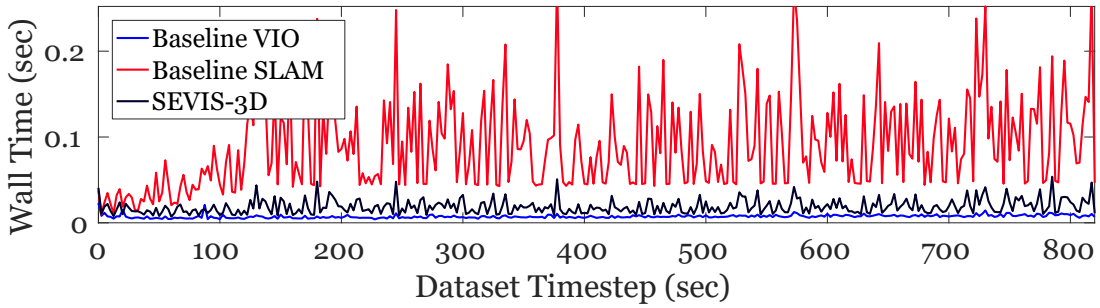



Figure 4.5: The wall clock execution time in seconds comparing the three methods can be seen. Plot is best seen in color.

Figure 4.6: Selected views during the night multi-floor trajectory show the high noise, poor lighting conditions, and motion blur that greatly challenge visual feature tracking.

perform on the level of the full baseline VI-SLAM system. Looking at the RE it is clear that the inclusion of map features prevents long-term drift and offers a greater accuracy shown by the almost constant RE as the trajectory segment length grows. The proposed SEVIS-3D provides a computationally feasible filter that has similar accuracy as full baseline VI-SLAM with competitive performance to that of VINS-Mono (although the VINS-Mono leverages batch optimization).

The primary advantage of the proposed SEVIS-3D algorithm over full-covariance SLAM is a decrease in computational complexity. The practical utility of this is evident in the run times of the different algorithms. As shown in Figure 4.5, we evaluated the three systems and collected timing statistics of our implementation.[1] The proposed SEVIS-3D is able to remain real-time (20 Hz camera means we need to be under 0.05 seconds total computation), while the full VI-SLAM method with 600 map features, has update spikes that reach magnitudes greater than four times the computational limit. This is due to the full covariance update being of order $O(n^2)$. Note that there is an additional overhead in the propagation stage as the symmetry of the covariance matrix needs to be enforced for the entire matrix instead of just the active elements to ensure numerical stability.

#### 4.3.4.2 Nighttime Multi-Floor Dataset

We further challenged the proposed system on a difficult indoor nighttime multi-floor dataset, which has multiple challenges including low light environments, long

[1] Single thread on an Intel(R) Xeon(R) E3-1505Mv6 @ 3.00GHz

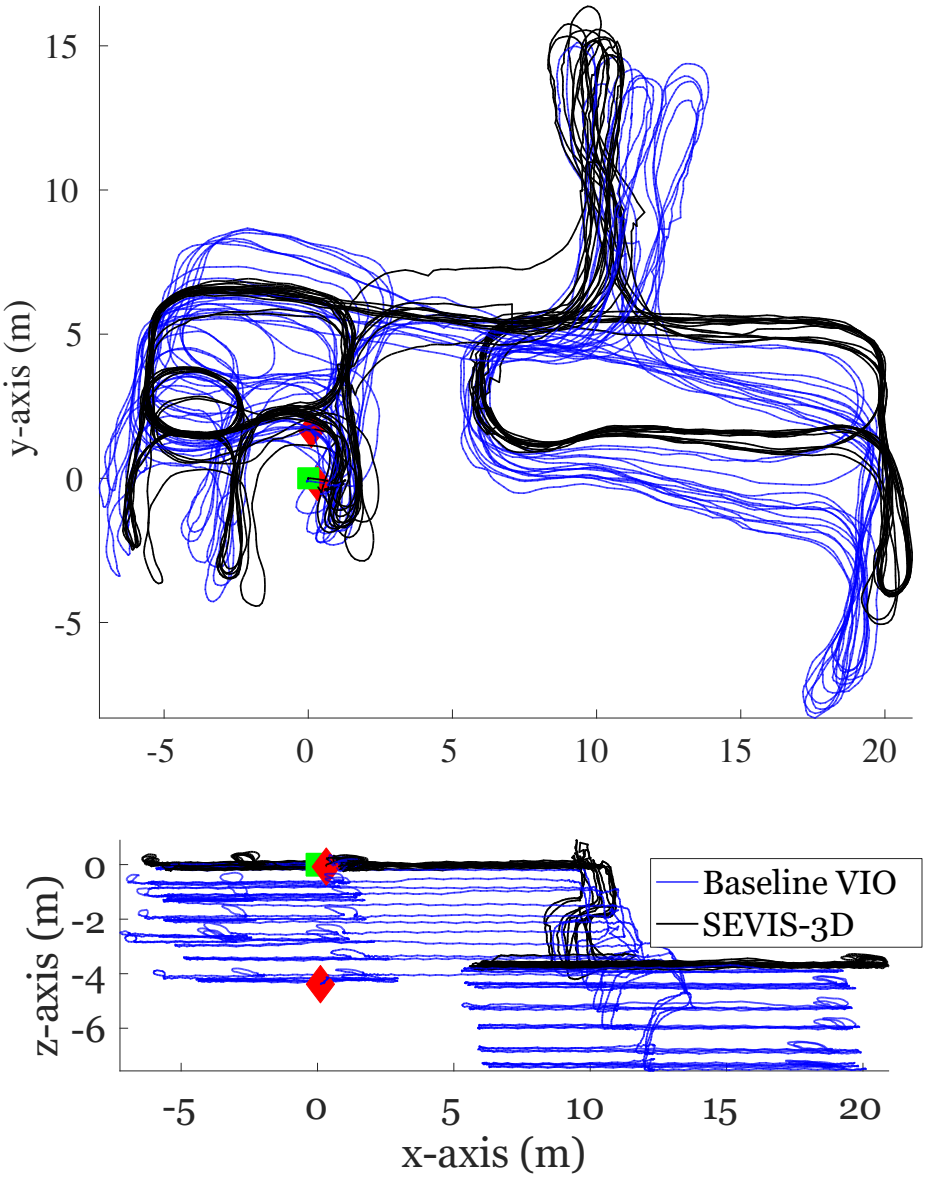



Figure 4.7: Estimated trajectories of the baseline VIO (blue) and SEVIS-3D (black) show improved performance due to the inclusion of map features. The start-end positions are denoted with a green square and a red diamond respectively.

exposure times, and low contrast images with motion blur unsuitable for proper feature extraction (see Figure 4.6). If features can be extracted, the resulting descriptor matching is poor due to the high noise and small gradients, and as compared to the Vicon Loops Dataset, more outliers are used during the update, causing large estimator jumps and incorrect corrections. We stress that the proposed SEVIS-3D can recover in these scenarios due to keyframe-aided 2D-to-2D matches which are invariant to poor estimator performance or drift and map feature updates correct and prevent incorrect drift.

A Realsense ZR300 sensor[2] was used to collect 20 minutes of grayscaled monocular fisheye images with inertial readings, with the 1.5km trajectory spanning two floors. We additionally performed online calibration of the camera to IMU extrinsic to further refine the transform provided by the manufacturer's driver. A max of 700 map points allowed for sufficient coverage of the mapping area, keyframes were inserted into the query database at 4Hz to ensure sufficient coverage of all map features, and 2 SLAM features in the active state at a time. The trajectory generated by the baseline VIO and the proposed SEVIS-3D are shown in Figure 4.7. Clearly, the inclusion of map features prevents long-term drift experienced by the baseline VIO which exhibits large errors in both the yaw and z-axis direction. Since no groundtruth was available for this dataset, as a common practice, we computed the start-end error of the trajectory which should ideally be equal to zero as the sensor platform was returned to the starting location. The baseline VIO had an error of 4.67m (0.31% of trajectory distance) while the proposed SEVIS-3D had an error of only 0.37m (0.02% of trajectory distance).

## 4.4 SEVIS-2D: Schmidt-EKF based VI-SLAM with 2D Observations

Next we will propose a novel **S**chmidt-**E**KF for **VI**-**S**LAM which leveraged 2D observations (SEVIS-2D) between active features and historical keyframes. As compared to keeping raw 3D points as in SEVIS-3D, here we look to try to reduce the

[2] https://software.intel.com/en-us/realsense/zr300

state size required to represent features by instead recording a keyframe with *multiple* feature observations which can each be leveraged. Specifically, we keep track of historical keyframes of our MSCKF stochastic clones which we can later perform loop-closure to get 2D-to-2D observation constraints to actively tracked features and include these additional measurements to indirectly constrain the state through the historical keyframes. To the best of the author's knowledge, this is the first time this type of indirect loop-closure has been leveraged to limit long-term estimation drift, while also improving computational efficiency through reduction of the state size. Consider the case that we continuously include the keyframe poses where loop closure events can be detected, into the state vector:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_C^\top & \mathbf{x}_M^\top & | & \mathbf{x}_{K_1}^\top & \cdots & \mathbf{x}_{K_n}^\top \end{bmatrix}^\top =: \begin{bmatrix} \mathbf{x}_A^\top & \mathbf{x}_S^\top \end{bmatrix}^\top \tag{4.13}$$

$$\mathbf{x}_A = \begin{bmatrix} \mathbf{x}_I^\top & \mathbf{x}_C^\top & \mathbf{x}_M^\top \end{bmatrix}^\top \tag{4.14}$$

where $\mathbf{x}_{K_i} = [{}^{I_i}_{G}\bar{q}^\top \ {}^{G}\mathbf{p}_{I_i}^\top]^\top$ is the $i$-th keyframe pose.

As keyframes are added over the trajectory length, the size of states that need to be estimated would grow over time, threatening the real-time VINS performance, although it grows much slower than adding keypoint features into the state vector. One approach that prevents the need to estimate the keyframes at later times, is to simply assume that keyframe poses are "true" and ignore the uncertainty associated with these estimates, which would cause an overconfident (inconsistent) filter. In contrast, we leverage the SKF to allow for efficient estimation while still tracking the uncertainty of all keyframes in the states.

Specifically, we carefully retain by stochastic cloning [170] a set of keyframe poses where loop-closures are likely to occur in the state vector (4.14) and consistently track their correlations with other state variables. We implicitly enforce loop closure constraints by adding additional observations from historical keyframe poses to actively tracked features in the sliding window of the MSCKF. It is important to note that this does not require estimating the 3D feature position, since these observations are only a function of the poses in the sliding window and the historical keyframes once

processed through the MSCKF update. This leads to substantial computational savings in additional gains from leveraging the SKF update, as the number of keyframes over a trajectory is typically *much* smaller than the number of features seen from those frames.

### 4.4.1 Visual Tracking and Loop Closing Methodology

As the IMU-camera sensor platform navigates in the environment, the sliding window sequentially shifts forward as a new image arrives. As in the standard MSCKF VIO (see Section 2.3.3.2), KLT-based visual tracking is employed to build feature tracks in the current sliding window. However, instead of marginalizing the old cloned camera poses as in the standard MSCKF, we select certain clones to be keyframes retained in the state vector for loop closure. While different heuristics may be used to select new keyframes, for example, based on the image parallax [160] or feature tracking quality [109], in this work, for proof-of-concept purposes, we simply add new keyframes at a fixed time interval. Once a cloned pose is chosen to be a keyframe, its corresponding state becomes part of the keyframe state [see (4.14)], whose cross-correlations (instead of autocovariance) will be updated at future times. This can be justified by the fact that when cloned poses reach the end of the sliding window and are selected as keyframes, their estimates are often accurate and can be assumed to have reached their steady states with matured but *non-zero* uncertainty, which will be properly and efficiently tracked in our estimator, instead of being naively assumed to be perfect with zero uncertainty, e.g., as in [160].

To perform keyframe-based loop closing, we leverage the state-of-the-art DBoW2 method [53] for finding loop closure candidates. When a new keyframe is inserted, the DBoW2 database is updated with the new keyframe image by extracting 300 FAST features [168] along with their ORB descriptors [171]. To detect loop-closures with the current camera image, we query the DBoW2 database to retrieve the top keyframes that are visually similar to it. After retrieval, a geometric check of the top candidate

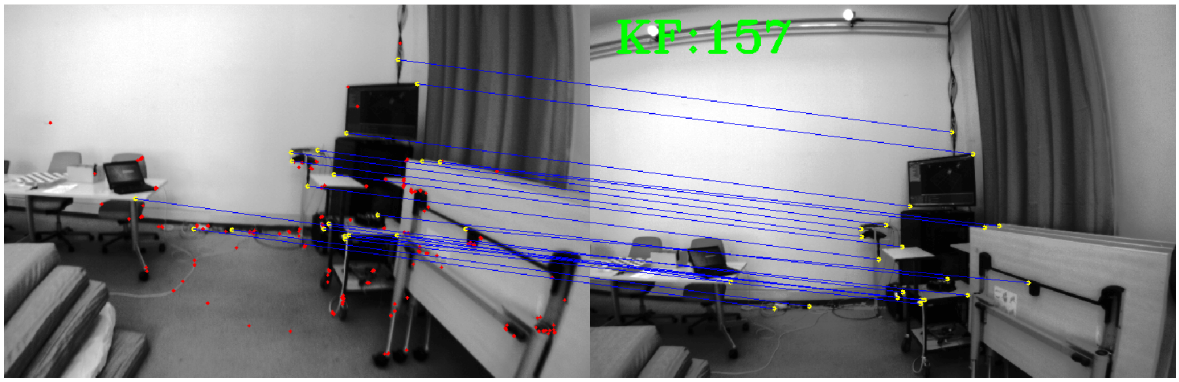



Figure 4.8: An example of feature matches between the current frame (left) and the keyframe (right). Active feature tracks are seen in red on the current image and their matches to the keyframe are visualized with blue correspondence lines.

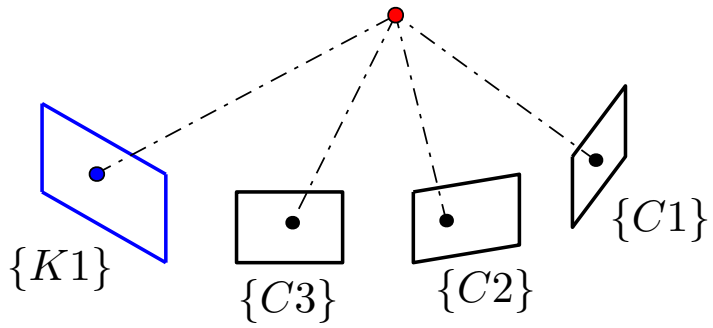



Figure 4.9: Illustration a keyframe-based loop closure scenario where an active feature tracked over three clones has matched to a keyframe $\{K1\}$. An additional feature measurement from the keyframe (blue) is added to the feature track such that an implicit loop closure constraint (by viewing the same scene) is formed and can be utilized in the EKF update.

keyframe from the database is performed by ensuring that the fundamental matrix can be calculated between the query image and candidate keyframe (see Figure 4.8).

Once a loop closure keyframe has been determined, we then incorporate the feature matches between the actively tracked features in the sliding window and those in the keyframe. For example, as illustrated in Figure 4.9, if a frame $\{C3\}$ is detected as matching a keyframe $\{K1\}$, then all extracted features in $\{C3\}$ will try to match with features extracted in $\{K1\}$. Specifically, when an actively tracked feature is matched to a keyframe feature, we add the additional keyframe observation to the feature track. Special care has to be taken that an active feature can only match to a keyframe once; that is, one feature measurement can only be involved in one feature track, in

order to prevent the reuse of information and thus ensure consistency. Note that for computational savings, the current image is only matched to a single keyframe while this can be easily modified to allow for more loop closure keyframes. We thus use the additional observations from the loop closure keyframe along with feature tracks from the current window to perform an MSCKF update once the active feature has lost track or reaches sliding window size. As explained in the following section, we update the active state estimate $\mathbf{x}_A$, its covariance, and the cross-correlations between the active state and keyframe state $\mathbf{x}_S$.

### 4.4.2 Map with Keyframes: Schmidt-EKF Update

Once feature tracks including both active feature measurements and (if any) loop closure constraints are ready for processing in the current sliding window, we perform an SKF update within the MSCKF framework. The main steps of the proposed SEVIS-2D approach are outlined in Algorithm 4. Specifically, as in the standard MSCKF, we first perform BA with all the feature measurements in the current window to triangulate the 3D feature positions ${}^G\mathbf{p}_f$ to linearize the measurement model in respect to all active clones $\mathbf{x}_C$ and any historical keyframe poses $\mathbf{x}_K$. For a given feature we perform linear marginalization of its position (i.e., null space operation) [211], and partition the active and Schmidt states as follows:

$$\mathbf{r}_f \simeq \mathbf{H}_x \tilde{\mathbf{x}}_{k|k-1} + \mathbf{H}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{n}_f \tag{4.15}$$

$$\simeq \mathbf{H}_{1\ldots c} \tilde{\mathbf{x}}_C + \mathbf{H}_K \tilde{\mathbf{x}}_K + \mathbf{H}_f {}^G\tilde{\mathbf{p}}_f + \mathbf{n}_f \tag{4.16}$$

We can then perform the MSCKF nullspace projection to remove the feature:

$$\mathbf{N}^\top \mathbf{r}_f \simeq \mathbf{N}^\top \mathbf{H}_{1\ldots c} \tilde{\mathbf{x}}_C + \mathbf{N}^\top \mathbf{H}_K \tilde{\mathbf{x}}_K + \mathbf{N}^\top \mathbf{n}_f \tag{4.17}$$

$$\mathbf{r}'_f \simeq \mathbf{H}'_{A_k} \tilde{\mathbf{x}}_{A_{k|k-1}} + \mathbf{H}'_{K_k} \tilde{\mathbf{x}}_{S_{k|k-1}} + \mathbf{n}'_f \tag{4.18}$$

where the residual is $\mathbf{r}'_f = \mathbf{N}^\top \mathbf{r}_f$, active clone Jacobian is $\mathbf{H}'_{A_k} = \mathbf{N}^\top \mathbf{H}_{1\ldots c}$, Schmidt'ed keyframe Jacobian is $\mathbf{H}'_{K_k} = \mathbf{N}^\top \mathbf{H}_K$, and projected noise $\mathbf{n}'_f = \mathbf{N}^\top \mathbf{n}_f$. We can then perform an SKF update with the above linear system, resulting in state corrections for the active state and cross-correlations with the map.

**Algorithm 4** Schmidt-EKF for VI-SLAM which Leveraged 2D Keyframe Observations (SEVIS-2D)

**Require:** Initial state estimate and covariance

1: **loop**
2: Propagate the state to the current image time.
3: Track features from the previous image into the current one.
4: Query the keyframe database for a loop closure and if there is a match, active features are appended with additional measurements from the loop closure keyframe.
5: Features that can be used for updates are collected and processed.
6: Oldest pose in the sliding window is either marginalized out or added to the keyframe state (nuisance parameters).
7: **end loop**

Table 4.3: RMSE position errors averaged over 10 runs of the Vicon loops dataset (units are in meters). SEVIS-2D (full) denotes running the proposed SEVIS-2D but allowing the covariance and keyframe estimates to update.

| | **MSCKF** | **SEVIS-2D (full)** | **SEVIS-2D** | **VINS-Mono [160]** |
|---|---|---|---|---|
| Monocular | 1.626 | 0.113 | 0.122 | 0.527 |
| Stereo | 1.555 | 0.093 | 0.172 | - |

### 4.4.3 Real-World Experimental Results

To validate the proposed SEVIS-2D, we have performed real-world experiments on different sensor platforms. In the following, we present two sets of results with hand-held sensors, demonstrating that the proposed approach achieves significantly better accuracy than the standard MSCKF (without loop-closures) while only incurring marginal computational overhead.

#### 4.4.3.1 Vicon Loops Dataset

We first test our VINS system on the Vicon loops dataset [109]. In this test, we select loop closure keyframes at a fixed rate of one every two seconds. For the results presented below, we have developed and validated both stereo and monocular (i.e., using only one of the stereo cameras) VINS algorithms. In particular, the estimators compared are (i) the standard MSCKF without loop-closures [139], (ii) the proposed

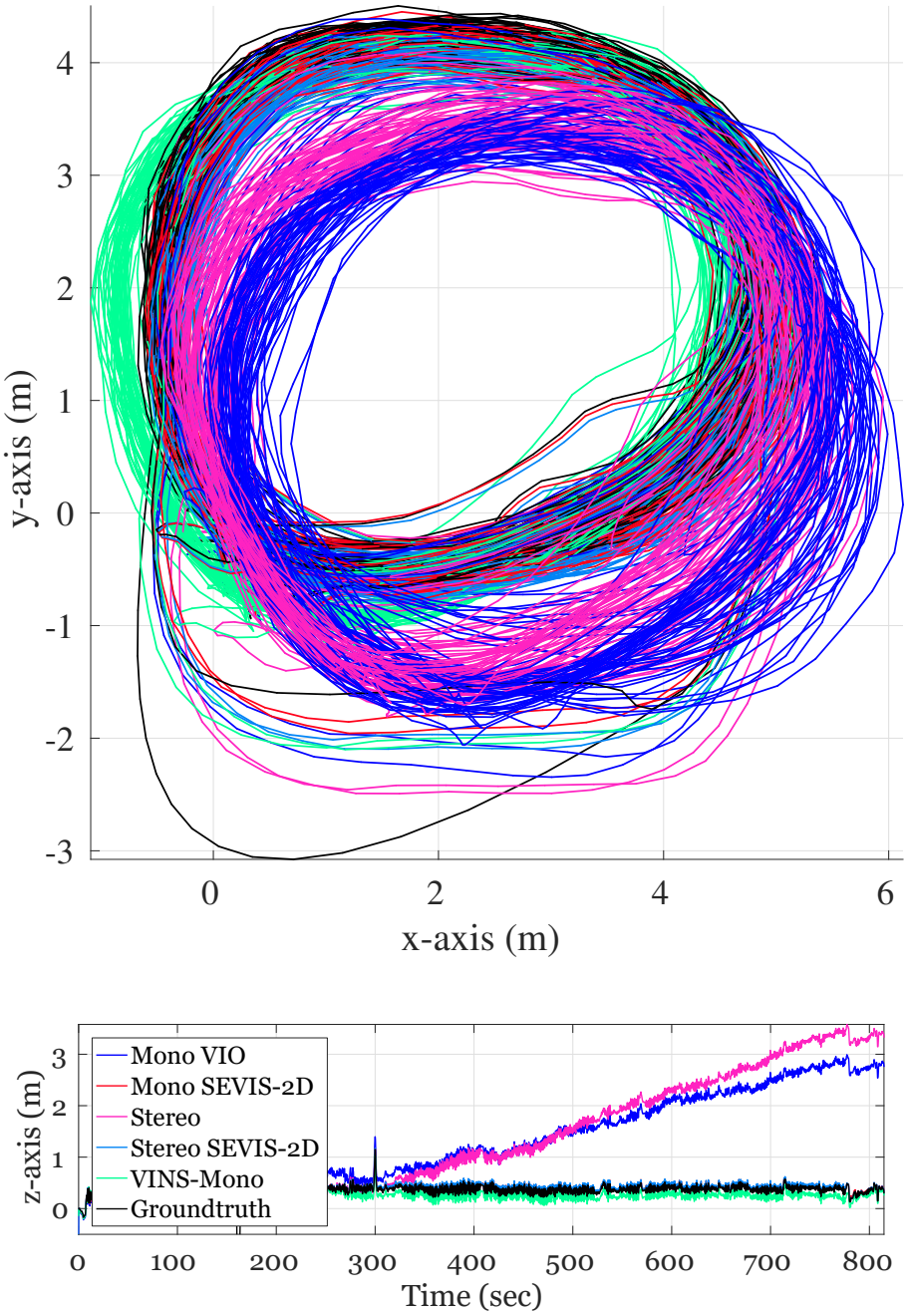



Figure 4.10: The estimated trajectories of the proposed SEVIS-2D, standard MSCKF [139], and VINS-Mono [159, 160]. Both monocular and stereo VINS results are presented. In particular, it is clear from the z-axis results (bottom) that the proposed approach is able to achieve bounded-error performance while the standard MSCKF has errors growing over time.

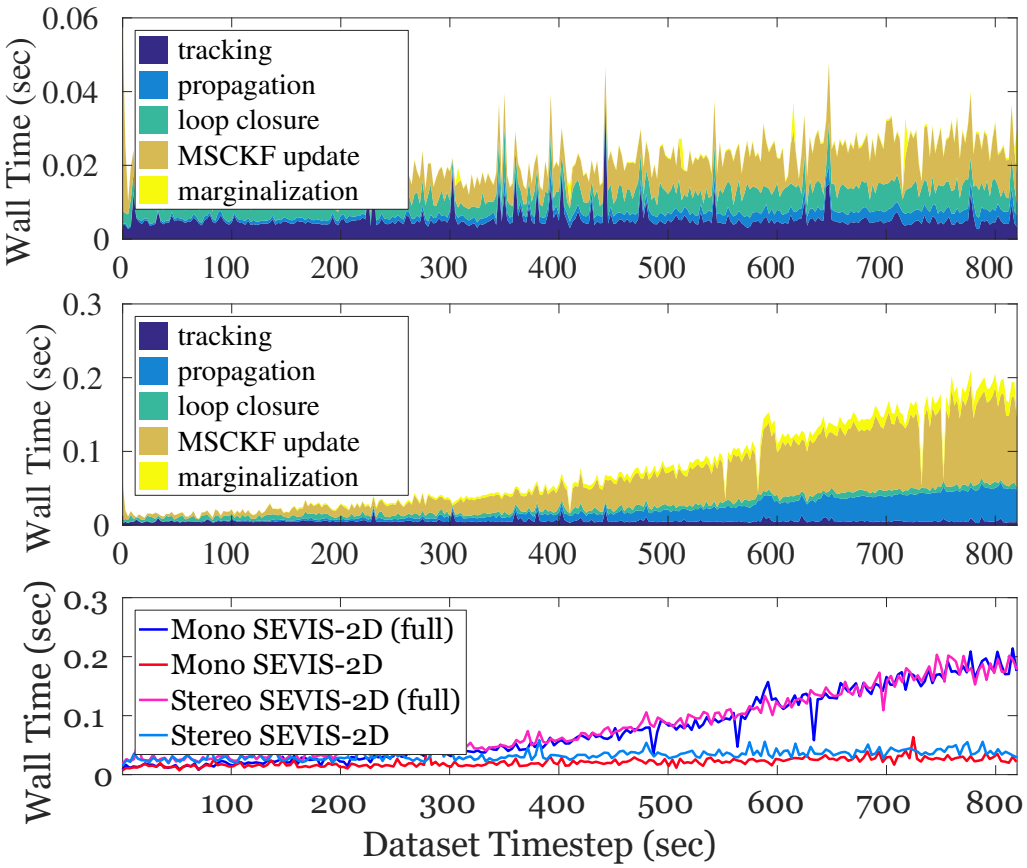



Figure 4.11: The CPU run time of the different components and the total execution time (bottom). The breakdown of the proposed monocular SEVIS-2D (top) and that of the SEVIS-2D with full covariance updates (middle).

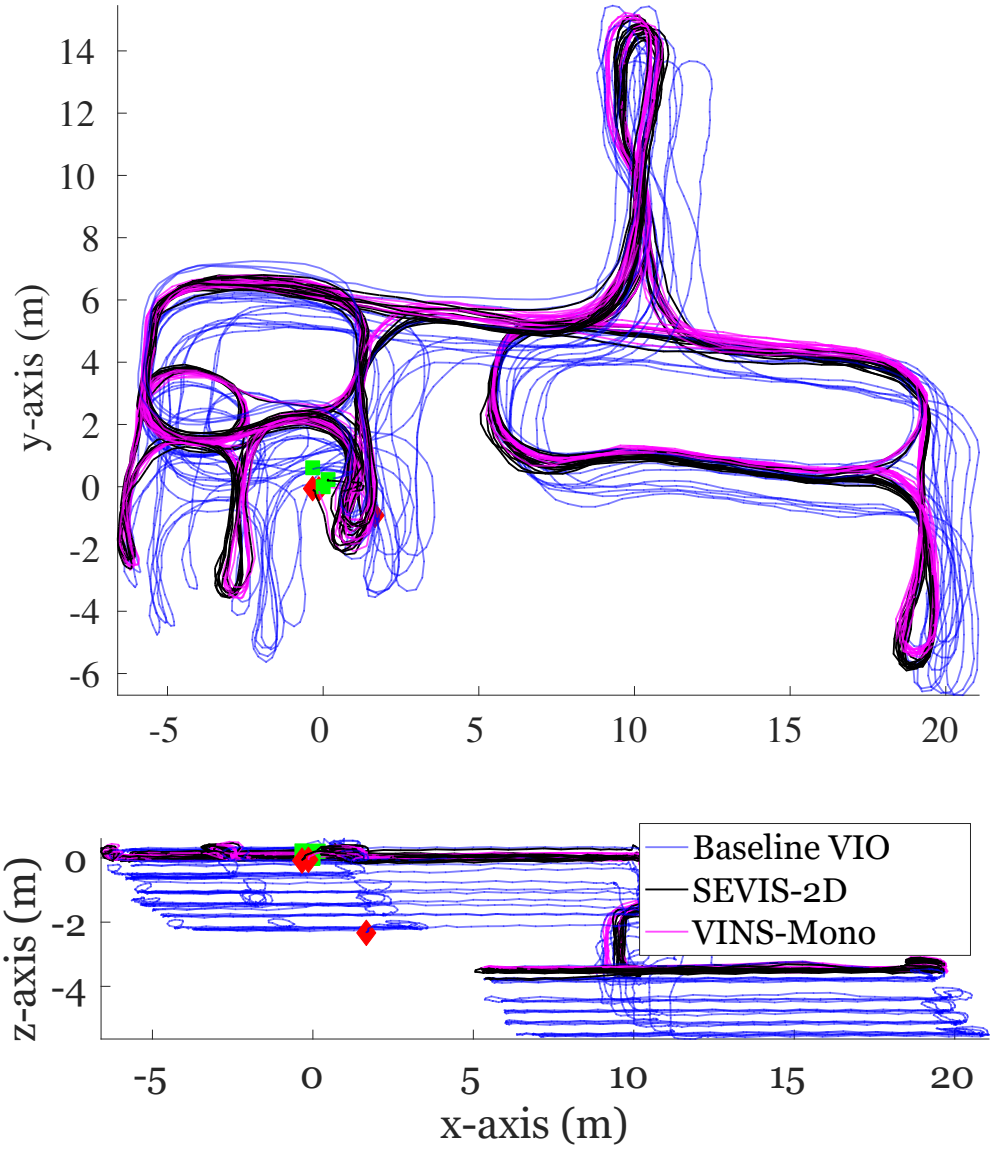



Figure 4.12: The trajectories of the standard MSCKF [139] (blue), proposed SEVIS-2D with loop-closures (black), and VINS-Mono [159, 160] (magenta). Clearly, without loop-closures, the standard MSCKF accumulates significant errors.

SEVIS-2D with full covariance update for the keyframes, (iii) the proposed SEVIS-2D and (iv) the open sourced VINS-Mono [159, 160]. The performance metrics used include (i) the root mean squared error (RMSE) that measures the estimation accuracy and (ii) the CPU run time that quantifies the computational cost.

Table 4.3 shows the RMSE values averaged over 10 runs (in order to consider the repeatability of the algorithms). Trajectories were aligned to the groundtruth using the first two minutes. Figure 4.10 depicts the estimated trajectories, which clearly shows that the information provided by the loop closure measurements significantly limits the drift of the proposed approach over time. Of the two MSCKF systems that utilize the loop closure constraints, the system that updates the full state estimate and covariance, as expected, achieves a slightly better performance; while the proposed SEVIS-2D closely follows it in accuracy but with a significant computational saving. Specifically, Figure 4.11 shows the CPU run time for a single representative run of the monocular VINS on the dataset.[3] As expected, the proposed SEVIS-2D (top) has only *linear* growth in the time it takes to perform an EKF update, while the standard MSCKF with full loop closure update (middle) shows *quadratic* growth in the computation time in both propagation and update. The proposed approach even stays below the real-time threshold of 0.05 sec (camera frame rate is 20Hz), processing 400 keyframes in real-time towards the end of the trajectory. In contrast, the VINS-Mono pose optimization backend (non-real-time thread) takes upwards of 2 seconds by the end of the trajectory, which significantly delays the inclusion of loop closure information in the current state estimate.

#### 4.4.3.2 Nighttime Multi-Floor Dataset

We further conducted a multi-floor indoor experiment at the University of Delaware (UD) Spencer Lab. Keyframes were inserted every 0.75 seconds, with a maximum of 886 keyframes in total, with 26% of all features containing at least one

[3] Single thread on an Intel(R) Xeon(R) E3-1505Mv6 @ 3.00GHz

keyframe feature observation during update. Figure 4.12 shows the estimated trajectories of the standard MSCKF, proposed monocular SEVIS-2D, and VINS-Mono. Clearly, the proposed method was able to localize with minimal drift over the trajectory while the standard MSCKF drifted significantly. As ground truth was not available for this dataset, we returned to the starting location and evaluated the start-end error to be 3.58m (0.2%), 0.64m (0.04%), 0.26m (0.02%), and 0.16m (0.01%) for the standard MSCKF, the proposed SEVIS-2D, SEVIS-2D with full covariance update, and VINS-Mono (with non-real-time optimization thread). Qualitatively, the trajectory of the proposed method is on par with that of VINS-Mono, while only requiring a single thread for real-time estimation.

## 4.5 SEVIS-PRIOR: Numerical Comparison of Loop-closure Constraints and Estimators

Next, the natural question is which of the two presented methodologies provides better performance in terms of accuracy and computational cost. Additionally, it is interesting to see how *inconsistent* methodologies which do not track correlations with the map perform. To simplify the problem and allow insight into the specific loop-closure constraint and estimator designs, we look at the case where an *a priori* map has been provided as compared to the previously presented methods which perform *simultaneous* localization and mapping. Specifically, we can define the following state of the system:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_A^\top & \mathbf{x}_F^\top & \mathbf{x}_K^\top \end{bmatrix}^\top \tag{4.19}$$

$$\mathbf{x}_F = \begin{bmatrix} {}^G\mathbf{p}_{f_1}^\top & \cdots & {}^G\mathbf{p}_{f_m}^\top \end{bmatrix}^\top \tag{4.20}$$

$$\mathbf{x}_K = \begin{bmatrix} \mathbf{x}_{K_1}^\top & \cdots & \mathbf{x}_{K_n}^\top \end{bmatrix}^\top \tag{4.21}$$

where we either have a map of features $\mathbf{x}_F$ or keyframes $\mathbf{x}_K$ in addition to the OpenVINS active state $\mathbf{x}_A$. For 2D-to-3D point loop-closures we will have the standard feature update equation:

$$\mathbf{r} = \mathbf{H}_T \tilde{\mathbf{x}}_{T_{1..c}} + \mathbf{H}_{f_i} {}^G\tilde{\mathbf{p}}_{f_i} + \mathbf{n} \tag{4.22}$$

where ${}^{G}\mathbf{p}_{f_i} \in \mathbf{x}_F$. When we have 2D-to-2D loop-closures via a historical keyframe observation will will have the following linear system after nullspace projection to remove the feature:

$$\mathbf{N}^\top \mathbf{r} = \mathbf{N}^\top \mathbf{H}_T \tilde{\mathbf{x}}_{T_{1..c}} + \mathbf{N}^\top \mathbf{H}_{T_k} \tilde{\mathbf{x}}_{T_k} + \mathbf{N}^\top \mathbf{H}_f {}^{G}\tilde{\mathbf{p}}_f + \mathbf{N}^\top \mathbf{n} \tag{4.23}$$

$$\Rightarrow \ \mathbf{r}' = \mathbf{H}'_T \tilde{\mathbf{x}}_{T_{1..c}} + \mathbf{H}'_{T_k} \tilde{\mathbf{x}}_{T_k} + \mathbf{n}' \tag{4.24}$$

where $\mathbf{n}' = \mathbf{N}^\top \mathbf{n}$ with covariance $\mathbf{R}' = \mathbf{N}^\top \mathbf{R} \mathbf{N}$.

### 4.5.1 Methods for Prior Map Updates

We now detail the different methods and techniques which enable the efficient incorporation of global information.

#### 4.5.1.1 Extended Kalman Filter

We first begin with the standard EKF which jointly estimates all variables. The equations for propagation and update are presented in Section 4.2. The memory requirements for landmark-based and keyframe-based maps are $O((3m)^2)$ and $O((6n)^2)$, respectively.

#### 4.5.1.2 Linear Schmidt-Kalman Filter

A consistent alternative to the standard EKF is the Schmidt-Kalman filter (SKF) [177]. This has been successfully used (along with different variations) to reduce the update complexity for map-based localization [34, 59, 65, 79, 93]. The equations for propagation and update are presented in Section 4.2. This process is $O(x)$ and its memory requirements for landmark-based and keyframe-based maps are $O((3m)^2)$ and $O((6n)^2)$, respectively. This is due to only updating the cross-covariance terms.

#### 4.5.1.3 Noise Inflation - Measurement

Another method for incorporating global information is to not explicitly estimate the map states (landmarks or keyframes). The downside is that this prevents the modeling of the correlation between the state and the map and thus is inconsistent.

Specifically, in Eq. (2.50) and (4.24) we treat the feature position and keyframe pose as known, and thus their Jacobians, $\mathbf{H}_{f_i}$ and $\mathbf{H}'_{T_k}$, become zero.

Naively the simplest way to solve this inconsistency due to an overconfident measurement is to inflate the measurement noise. For landmark-based or keyframe-based maps we can simply inflate the measurement observation noise as:

$$\mathbf{R} = (\gamma\sigma_{pix})^2\mathbf{I} \tag{4.25}$$

The key advantage of this method is that the computational cost is now constant $O(1)$ since only the inertial state, sliding window, and temporal SLAM map are tracked. The memory requirement for both landmark and keyframe-based maps is $O(0)$. This can have profound impacts on large maps and thus giving up the guarantee of consistency for this computational advantage is very alluring.

#### 4.5.1.4 Noise Inflation - Marginal Covariance Inflation

Many works have leveraged the marginal covariance of the prior map to both reduce the complexity and memory requirements of the system (e.g., [126]). The main advantage is that this allows for each landmark or keyframe to have different levels of uncertainty and the use of its Jacobian to map the additional error to the observed measurement. More concretely we have the following modified measurement noise for landmark-based and keyframe-based prior maps respectively [see Eq. (2.50) and (4.24)]:

$$\mathbf{R} = \mu\mathbf{H}_{f_i}\mathbf{P}_{ff_i}\mathbf{H}_{f_i}^\top + \sigma_{pix}^2\mathbf{I} \tag{4.26}$$

$$\mathbf{R} = \mu\mathbf{H}'_{T_k}\mathbf{P}_{TT_k}\mathbf{H}_{T_k}'^\top + \sigma_{pix}^2\mathbf{I} \tag{4.27}$$

where $\mathbf{P}_{ff_i}$ and $\mathbf{P}_{TT_k}$ are the 3x3 and 6x6 prior landmark and keyframe covariances, respectively. This process also ensures the computational cost is also now constant $O(1)$, with memory requirements of $O(9m)$ and $O(36n)$, respectively.

Table 4.4: Simulation parameters and priors that perturbations of measurements and initial states were drawn from.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Pixel Proj. (px) | 1 | Num. Camera | 1 |
| IMU Freq. (hz) | 400 | Cam Freq. (hz) | 10 |
| Avg. Feats | 15 | Num. SLAM | 10 |
| Num. Clones | 11 | Feat. Rep. | GLOBAL |
| Gyro. White Noise | 1.6968e-04 | Gyro. Rand. Walk | 1.9393e-05 |
| Accel. White Noise | 2.0000e-03 | Accel. Rand. Walk | 3.0000e-03 |
| Prior Key. Ori. (deg) | 1.0 | Prior Key. Pos. (cm) | 6 |
| Prior Feat. Pos. (cm) | 12 | % Feat. Lost Btw Key. | 75 |
| Max Dist. Btw Key. (m) | 1 | Max Deg. Btw Key. (deg) | 15 |
| Map PTS | 210 | Map KF | 86 |

#### 4.5.1.5 Noise Inflation - Alpha Beta Inflation

The final noise inflation variation investigated is the one presented in [174], which incorporates not only inflation due to the marginal prior map covariance but also the current state covariance (originally adopted by NASA's Apollo program [6] and used to "intentionally slow adaptation in linearized estimation problems"). Specifically, we have the following:

$$\mathbf{R} = \alpha \mathbf{H}_{f_i} \mathbf{P}_{f f_i} \mathbf{H}_{f_i}^\top + \beta \mathbf{H}_T \mathbf{P} \mathbf{H}_T^\top + \sigma_{pix}^2 \mathbf{I} \tag{4.28}$$

$$\mathbf{R} = \alpha \mathbf{H}'_{T_k} \mathbf{P}_{T T_k} \mathbf{H}_{T_k}'^\top + \beta \mathbf{H}'_T \mathbf{P} \mathbf{H}_T'^\top + \sigma_{pix}^2 \mathbf{I} \tag{4.29}$$

This process is constant $O(1)$ in terms of computational cost, with memory requirements of $O(9m)$ and $O(36n)$ for landmark and keyframe-based maps. We normally "whitten" the linearized measurement function with the now dense noise to regain an identity noise covariance form.

### 4.5.2 Numerical Study

To investigate and compare the different methods for global measurement inclusion we simulated a realistic indoor single room dataset which is approximately 15 minutes long and 1.2km in length (see Figure 4.13). We employ the OpenVINS simulator to generate realistic visual-bearing and inertial measurements from the trajectory

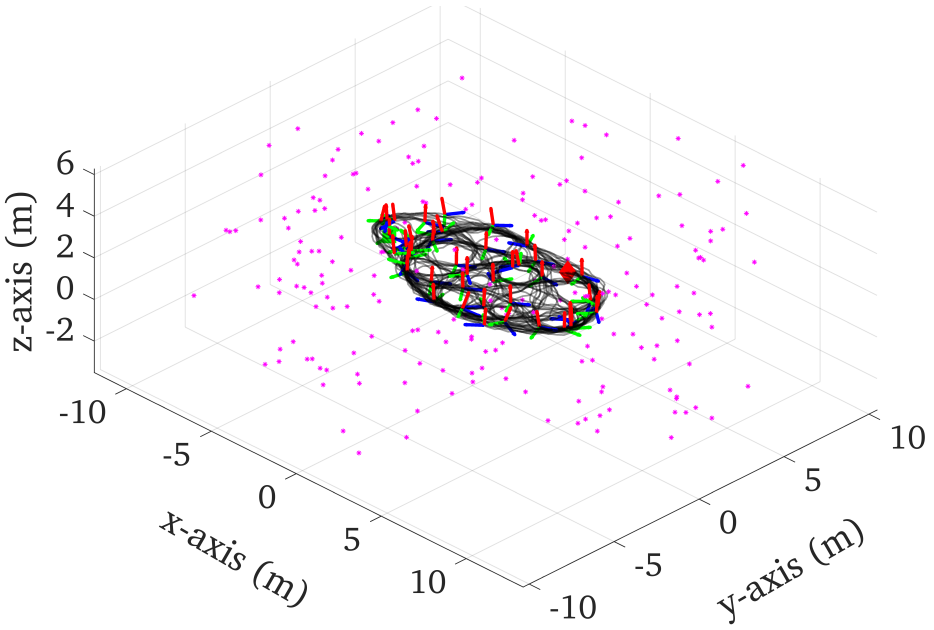



Figure 4.13: Simulated 1.2km hand-held Room trajectory, axes are in units of meters. Every other keyframe is shown to increase clarity. Feature depths (purple) are between 5 and 7 meters.

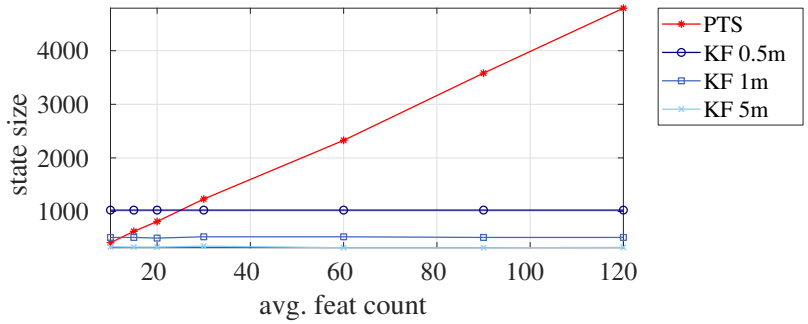



Figure 4.14: Relation between state size (number of variables) and the average number of features observed for both landmark-based (PTS) and keyframe-based (KF) maps in the Room dataset. Different maximum keyframe distance thresholds are also plotted.

generated by an existing VINS. Simulation parameters used are documented in Table 4.4, while details on how the prior map is generated are specified in the following section. First-estimates Jacobians (FEJ) [83, 84] were used to improve the estimator consistency as the use of environmental landmarks is known to introduce inconsistent information gains. For metrics we report the Absolute Trajectory Error (ATE), Normalized Estimation Error Squared (NEES), and Relative Pose Error (RPE) throughout the different experiments (see [221] and [4]). NEES's magnitude should match the 3 DoF orientation and position state sizes.

Feature matching to historical keyframes to gain additional feature observations was simulated by selecting the closest keyframe and using groundtruth labels, while for map features the groundtruth labels were directly used (thus perfect matching). In real-world experiments, where incorrect feature associations are prevalent, chi-squared thresholding can be leveraged before updating to reject outliers. Additional simulation results for different trajectories and noise perturbations can be found in the companion technical report [63].

#### 4.5.2.1 Prior Map Generation

We now describe the procedure for how we generate a prior map of environmental landmarks and keyframes (e.g., Figure 4.13). Starting at the beginning of the trajectory we move the camera forward in time at a rate of 4 Hz. At each timestep we project the current landmark map into the camera frame and if the number of seen features falls below our average feature tracking amount we generate new features. This is repeated until the end of the trajectory is reached and our prior landmark map is complete after applying perturbations.

To generate the keyframe map, we repeat this procedure. Specifically at each timestep the current camera must be near an existing keyframe and share a sufficient percentage of common overlapping features; otherwise, a new keyframe is created. After generating our keyframes, we project the landmark map into each to generate bearing observations, and both the keyframe poses and observations are perturbed.

Table 4.5: Average ATE and NEES over 5 Room dataset runs for different map priors and algorithms.

| | Prior | Algo. | ATE (deg / m) | NEES (3) |
|---|---|---|---|---|
| VIO | - | - | 2.603 / 0.271 | 3.524 / 1.591 |
| 2D-to-2D | 0.5°, 3cm | EKF | 0.324 / 0.090 | 2.933 / 3.327 |
| | | SKF | 0.374 / 0.099 | 2.758 / 3.248 |
| | 1.0°, 6cm | EKF | 0.442 / 0.105 | 3.236 / 3.698 |
| | | SKF | 0.518 / 0.130 | 2.806 / 3.466 |
| | 3.0°, 12cm | EKF | 0.629 / 0.127 | 4.353 / 5.335 |
| | | SKF | 0.941 / 0.167 | 3.009 / 3.585 |
| 2D-to-3D | 3cm | EKF | 0.051 / 0.010 | 5.975 / 6.586 |
| | | SKF | 0.064 / 0.021 | 2.898 / 3.188 |
| | 6cm | EKF | 0.068 / 0.014 | 8.224 / 9.292 |
| | | SKF | 0.087 / 0.036 | 2.863 / 3.210 |
| | 12cm | EKF | 0.079 / 0.015 | 9.321 / 9.472 |
| | | SKF | 0.122 / 0.065 | 2.761 / 3.175 |

Shown in Figure 4.14, we perform a small study on how the prior map state size changes with the average number of feature tracks. Landmark-based maps have a state of $3m$, where $m$ is the number of landmarks, and keyframe-based maps have $6n$, where $n$ is the number of keyframes. The landmark map has a very linear relationship with the average number of tracked features and grows to a very large size, which is expected. We additionally show three keyframe prior maps with different maximum distances between generated keyframes. For the keyframe-based maps there is a clear advantage in state size, as the average number of tracked features increases since more features just increase the number of observations in all keyframes. For the rest of the experiments we select a keyframe distance of 1 meter since the state size is close to the size of a point-based map when using 15 average features, and thus this advantage won't be shown.

Table 4.6: Average ATE and NEES over 5 Room runs for different inflation values.

| | $\gamma$ | ATE (deg / m) | NEES (3) | $\mu$ | ATE (deg / m) | NEES (3) | $\alpha$,$\beta$ | ATE (deg / m) | NEES (3) |
|---|---|---|---|---|---|---|---|---|---|
| VIO | - | 2.381 / 0.267 | 3.522 / 1.590 | - | 2.381 / 0.267 | 3.522 / 1.590 | - | 2.381 / 0.267 | 3.522 / 1.590 |
| 2D-to-2D | 1 | * / * | * / * | 1 | 0.853 / 0.187 | 4.219 / 6.928 | 1,1 | 0.883 / 0.187 | 3.796 / 6.070 |
| | 5 | 0.737 / 0.219 | 5.197 / 17.377 | 5 | 0.846 / 0.182 | 3.124 / 3.198 | 5,2 | 0.810 / 0.182 | 2.826 / 2.916 |
| | 10 | 0.931 / 0.181 | 3.960 / 6.099 | 10 | 0.787 / 0.180 | 2.699 / 2.385 | 10,5 | 0.899 / 0.192 | 2.688 / 2.275 |
| | 20 | 0.886 / 0.184 | 2.949 / 3.557 | 20 | 0.822 / 0.185 | 2.574 / 1.893 | 20,5 | 0.928 / 0.193 | 2.650 / 1.867 |
| 2D-to-3D | 1 | * / * | * / * | 1 | 0.132 / 0.045 | 12.438 / 18.407 | 1,1 | 0.131 / 0.045 | 12.184 / 17.957 |
| | 5 | 0.178 / 0.055 | 17.185 / 27.854 | 5 | 0.110 / 0.040 | 4.387 / 4.537 | 5,2 | 0.110 / 0.040 | 4.323 / 4.442 |
| | 10 | 0.163 / 0.054 | 7.584 / 10.841 | 10 | 0.109 / 0.041 | 3.308 / 2.731 | 10,5 | 0.109 / 0.041 | 3.233 / 2.611 |
| | 20 | 0.156 / 0.057 | 3.861 / 3.795 | 20 | 0.111 / 0.043 | 2.761 / 1.743 | 20,5 | 0.112 / 0.043 | 2.726 / 1.688 |

Table 4.7: Average RPE over the Room dataset for different prior map types and algorithms. Units are in degrees and meters. Additionally the NEES and total time to process each image is reported.

| | Algo. | 40m | 80m | 120m | 160m | 200m | 240m | NEES (ori / pos) | Time (ms) |
|---|---|---|---|---|---|---|---|---|---|
| VIO | - | 0.373 / 0.088 | 0.536 / 0.119 | 0.636 / 0.141 | 0.717 / 0.163 | 0.811 / 0.175 | 0.888 / 0.187 | 3.228 / 3.796 | 0.8 ± 0.3 |
| 2D-to-2D | EKF | 0.225 / 0.091 | 0.323 / 0.111 | 0.372 / 0.120 | 0.402 / 0.121 | 0.424 / 0.122 | 0.394 / 0.125 | 3.298 / 4.311 | 3.6 ± 1.8 |
| | SKF | 0.260 / 0.097 | 0.339 / 0.129 | 0.415 / 0.146 | 0.448 / 0.155 | 0.492 / 0.167 | 0.542 / 0.171 | 3.074 / 3.596 | 1.4 ± 0.7 |
| | Inf. Meas. | 0.276 / 0.099 | 0.353 / 0.134 | 0.449 / 0.152 | 0.518 / 0.163 | 0.531 / 0.173 | 0.562 / 0.180 | 3.016 / 3.647 | 0.9 ± 0.3 |
| | Inf. Marg. | 0.265 / 0.091 | 0.350 / 0.122 | 0.447 / 0.142 | 0.520 / 0.156 | 0.560 / 0.169 | 0.613 / 0.175 | 2.795 / 2.784 | 0.9 ± 0.3 |
| | Inf. $\alpha\beta$ | 0.269 / 0.091 | 0.353 / 0.122 | 0.456 / 0.142 | 0.546 / 0.156 | 0.599 / 0.168 | 0.656 / 0.173 | 2.781 / 2.689 | 0.9 ± 0.3 |
| 2D-to-3D | EKF | 0.041 / 0.009 | 0.041 / 0.009 | 0.041 / 0.009 | 0.041 / 0.009 | 0.041 / 0.009 | 0.041 / 0.009 | 9.612 / 7.792 | 5.8 ± 1.1 |
| | SKF | 0.090 / 0.040 | 0.092 / 0.038 | 0.091 / 0.040 | 0.090 / 0.038 | 0.092 / 0.039 | 0.091 / 0.039 | 3.051 / 2.963 | 1.4 ± 0.2 |
| | Inf. Meas. | 0.125 / 0.068 | 0.139 / 0.065 | 0.141 / 0.067 | 0.141 / 0.064 | 0.142 / 0.066 | 0.136 / 0.065 | 3.663 / 3.528 | 0.6 ± 0.1 |
| | Inf. Marg. | 0.102 / 0.046 | 0.103 / 0.045 | 0.102 / 0.046 | 0.098 / 0.044 | 0.103 / 0.046 | 0.100 / 0.045 | 3.201 / 2.546 | 0.6 ± 0.1 |
| | Inf. $\alpha\beta$ | 0.102 / 0.047 | 0.103 / 0.046 | 0.102 / 0.047 | 0.098 / 0.045 | 0.103 / 0.046 | 0.100 / 0.046 | 3.126 / 2.437 | 0.6 ± 0.1 |

#### 4.5.2.2 Map Prior Noise Sensitivity

A natural question is how will the "best" estimator perform with different prior map noises. We first investigate this using the standard EKF and SKF to see how the accuracy is affected by the quality and uncertainty levels of the prior map. Shown in Table 4.5, we report the VIO, which doesn't leverage any prior map, the landmark-based 2D-to-3D map, and the keyframe-based 2D-to-2D map. The simulator parameters used are reported in Table 4.4.

First, we can see that all the prior map methods are able to outperform the odometry VIO method. Additionally, even at large noise levels of 12cm, both the landmark and keyframe methods are still able to gain in both orientation and position accuracy. Additionally, we can see that the 2D-to-3D methods greatly outperform the 2D-to-2D method. This makes sense since the 2D-to-2D indirectly constrains the current pose of the system through additional feature observations, while the 2D-to-3D directly constrains *all* observations for a feature. It is also interesting to note that while the EKF 2D-to-3D has very good levels of accuracy the NEES increases with noise. We conjecture this is due to FEJ, which can introduce linearization errors at high noise levels (the SKF hides this due to its naturally conservative covariance, see [22] for a discussion). Given these results we pick our priors used during the rest of the simulations, Table 4.4, as 12cm for the landmark-based map and 1 degree and 6 centimeters for the keyframe-based map with 1 pixel observation noise.

#### 4.5.2.3 Inflation Tuning Sensitivity

A downside of the inflation methods is that their inflation multipliers need to be tuned. The results reported in Table 4.6 look to answer if they are sensitive to their value and determine what the optimal is. We can first see that the measurement-based inflation, $\gamma$, requires the largest amount of inflation levels to reach consistent estimation, and with an inflation value of 1 the estimator quickly diverged since it is equivalent to treating the feature position as true. The amount of inflation when using the marginal covariance $\mu$, alpha $\alpha$, and beta $\beta$ inflation does not have that large of

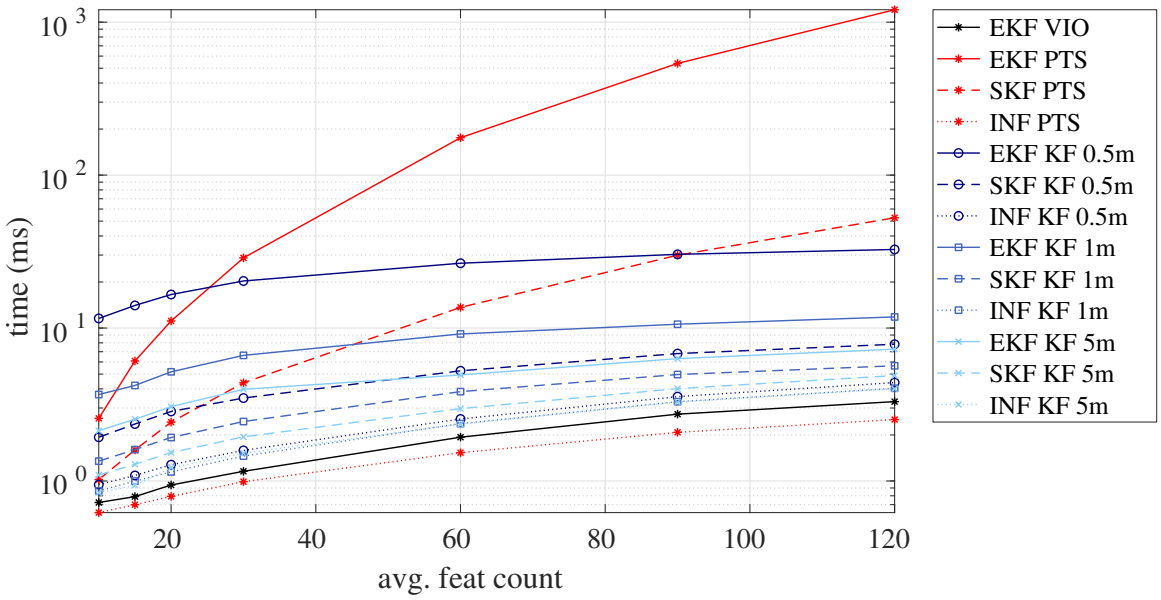



Figure 4.15: Runtime in milliseconds for both propagation and update without (VIO) and with both landmark-based (PTS) and keyframe-based (KF) maps for the Room dataset. Keyframe-based maps are reported for different max keyframe distances.

an effect on accuracy which is ideal. Additionally, there seems to be little difference between the two. We can therefore recommend inflating using the marginal or alpha beta covariance with a conservative (one order) multiplier (this does not *guarantee* consistency). It is also important to note that while these two methods do have some invariance to different prior map noises, the measurement inflation parameter $\gamma$ highly depends on the prior map quality. We select an inflation of $\gamma = 20$, $\mu = 10$, $\alpha = 10$, and $\beta = 5$ for the rest of the experiments.

#### 4.5.2.4 Map and Algorithm Comparison

We now look to compare the different prior map types and methods that incorporate global information. We report the results in Table 4.7. In general, we see that the 2D-to-3D landmark-based methods are able to achieve an order of magnitude better accuracy across all variants, with near constant error as the RPE segments grow in length. The 2D-to-2D method is able to halve the orientation error, but the position error has marginal improvements when compared to that of the 2D-to-3D method. The

majority of improvements are at longer trajectory lengths of 200-240m as compared to the shorter segments. This is likely due to the fact that it takes *many* historical 2D-to-2D observations to improve the state as compared to the "strong" constraint a 3D position of the feature in the 2D-to-3D method provides.

We additionally report on the right of Table 4.7 and in Figure 4.15 each method's average timing. The EKF takes the most time, the SKF second, and the inflation methods all around the same.[4] In Figure 4.15, we additionally show the computational cost as we increase the average number of features and for different keyframe distance thresholds. The 2D-to-2D (KF) methods have a near constant offset from the VIO time as the number of average features only marginally increases the computational cost due to more measurements. This is a clear advantage when the number of tracked features is large. The 2D-to-3D (PTS) method quickly increases an order of magnitude slower than VIO, which is expected as the state size dramatically grows (see Figure 4.14). The inflation methods (INF) for both landmark and keyframe prior maps perform as efficiently as VIO due to their near constant run-time and constant state vector size.

### 4.5.3 Findings and Discussions

In summary, we have investigated through simulation the: relation between state size and the average number of features, achievable accuracy given different map priors, sensitivity of inflation methods to their tuning parameters, and how all methods compare in terms of accuracy, consistency, and computational cost for both 2D-to-3D landmark and 2D-to-2D keyframe maps. We showed that even at extremely high noise levels, in general, the 2D-to-3D maps outperform the 2D-to-2D methods in accuracy. Keyframe maps have a computational advantage due to their state size when using a large number of features. The marginal and alpha beta covariance inflation methods are relatively invariant to their inflation parameters making them ideal for large environmental maps where EKF and SKF estimators become prohibitively expensive or

[4] All timings were run on an Intel(R) Xeon(R) CPU E3-1505M v6 @ 3.00GHz processor in single threaded execution.

the loss of consistency guarantees is acceptable.

Finally, we evaluated all methods against each other and make the following general recommendations: (1) the SKF should be used for small workspaces to ensure consistency and achieve high accuracy levels with low computational cost, (2) keyframe-based maps can be leveraged to reduce the computational cost while still reducing drift, (3) for large environments and map sizes, inflation methods can practically be leveraged with conservative inflation values.

## 4.6 SEVIS: Balancing Performance via Dynamic Schmidt'ing, Accuracy, Consistency, and Relinearization

Finally we look to address the long-standing issue of the above filter's ability to perform relinearization. While this does not affect the case when a prior map is being leveraged, when performing VI-SLAM odometry errors from significant periods of exploration can build up and corrupt the incrementally built SKF maps. To address this, here we propose a hybrid estimator which looks to combine this lightweight filter-based SEVIS with a secondary non-linear optimization which *can* perform relinearization and correct to ensure global consistency when a large loop closure is found. As compared to existing works which have leveraged a secondary thread to reduce the complexity of SLAM [92, 119, 159, 160], or treat the built map as true [100, 126, 127, 138, 141, 146, 192], we wish to still perform as consistent of estimation while balancing this computational complexity.

A system diagram and graphical overview of the proposed system are shown in Figure 4.16 and 4.17, respectively. At the core, we have a real-time filter-based frontend that slowly appends environmental features into a temporal map around it. Additionally, a novel "dynamic Schmidt'ing", Section 4.6.1.1, methodology is proposed which enables features to continuously be refined when re-observed but otherwise fixed to reduce the computational burden. This naturally allows for accuracy gains with minimal computational cost since the whole map isn't visible at each timestep. When

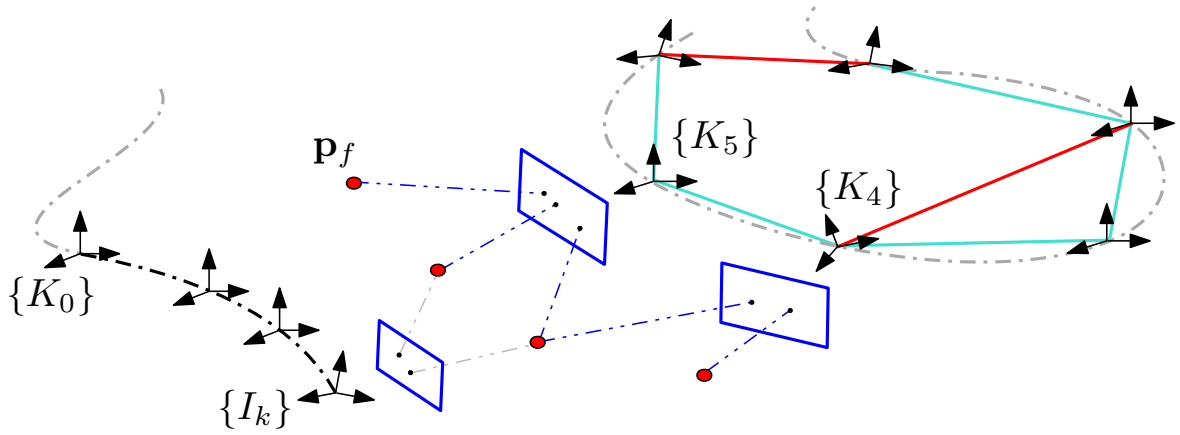



Figure 4.16: Overview of frontend odometry with a sliding window of clones and backend which maintains a sparse keyframe pose graph. The backend contains relative factors (cyan) computed relative to the last added keyframe $\{K_0\}$. Loop-closure between keyframes uses place recognition feature correspondences' from which the relative transformation and uncertainty can be found using PnP. The optimized keyframe states can then be leveraged to provide an optimized sparse feature map which can be directly leveraged by the real-time frontend odometry.

features have become lost they are retained for a period of time during which re-detection of them through place recognition can allow for future feature observations to update these historical points. This temporal map contains a mixture of features which have a correlation with the current frontend state and are being updated (e.g., via EKF update), ones which track the correlation but do not get updated (e.g., via SKF update), and ones which correlations are dropped and leverages the marginal covariance of the feature optimized by the backend (MARG).

The backend is a non-real-time thread which optimizes a relative pose graph containing relative optometry between sequential keyframes, and PnP loop-closure relatives which reduces the long-term VIO drift. A particular focus is paid to each relative factor to recover the odometry uncertainty in a consistent manner, Section 4.6.2.1, and quantify the PnP uncertainty, Section 4.6.2.2. The backend enables relinearization, and additionally allows for refined uncertainties of all keyframes to be recovered. The uncertainty of each keyframe is then used to recover the uncertainty of anchored features, which are *not* optimized to significantly reduce complexity, in the global frame which the frontend can directly leverage, Section 4.6.2.3.

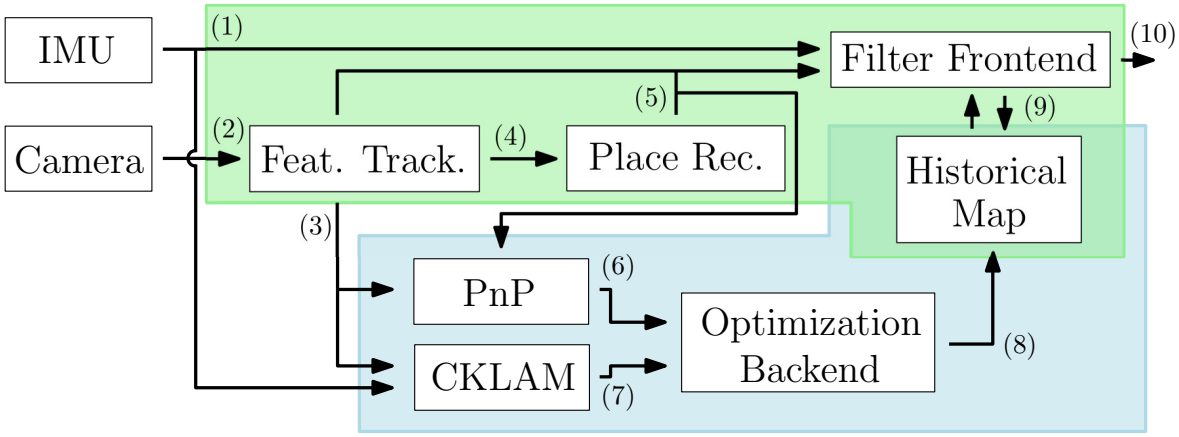



Figure 4.17: System diagram overview of the proposed real-time frontend (green) and consistent relative pose graph backend (blue). The data passed is: (1) IMU readings are both used to propagate the frontend state forward and recover consistent relative pose factors, (2) features are temporally tracked, (3) feature track measurements are provided to the backend, (4) feature descriptors are extracted and place recognition is performed to historical keyframes, (5) new features are merged with old features if matched, (6) relative PnP and its uncertainty is recovered, (7) relative keyframe odometry and uncertainty is recovered, (8) optimized keyframe poses and their uncertainties are used to update the map features, (9) the filter frontend is able to leverage optimized map features consistently, (10) low-latency and high frequency pose is provided to downstream applications.

**Remarks:** This hybrid combination of EKF, SKF, MARG, and backend optimization selects the strengths of each. The EKF and SKF allow for accurate estimation which tracks the correlations between inertial and map states preventing the need to introduce inflation. The use of "dynamic Schmidt'ing" additionally ensures that features, which have tracked correlations, gain maximum information and accuracy. MARG features are leveraged as a trade-off between accuracy and consistency with the frontend and enable using the drift-free estimates from the backend optimization process which supports relinearization and incorporates additional long-term loop-closures.

As compared to previous works [16, 92, 160] which either hand-tuned their secondary pose graph uncertainties, consider a uniformly weighted graph, or do not provide updated feature estimates and uncertainties to the frontend estimator, we leverage the uncertainty of relative odometry poses and modeling of PnP uncertainty to facilitate *feedback* of a optimized map to the lightweight odometry frontend. Our hybrid frontend additionally enables further accuracy by allowing for recent loop-closures to

a temporal map without employing approximations which can hurt consistency. As compared to the works closest to the proposed, [140, 174], we focus on providing low-latency estimates and additionally argue that the use of the computationally cheaper relative pose graph, as compared to full visual-inertial optimization, enables long-term estimation on more resource-constrained devices. We additionally, focus on not making assumptions, such as conditioning on the camera pose states in [174] when recovering the marginal feature uncertainties or in the design of our system architecture.

### 4.6.1 Frontend - SEVIS-3D with Dynamic Schmidt'ing

We build off MSCKF-based VIO [112, 139] and its extension to SEVIS-3D, Section 4.3 which incorporates prior landmark map via the SKF. Specifically, we can define the following state of the system:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_A^\top & \mathbf{x}_S^\top \end{bmatrix}^\top \tag{4.30}$$

$$\mathbf{x}_A = \begin{bmatrix} \mathbf{x}_{I_k}^\top & \mathbf{x}_C^\top & \mathbf{x}_M^\top & \mathbf{x}_F^\top \end{bmatrix}^\top \tag{4.31}$$

$$\mathbf{x}_M = \begin{bmatrix} {}^G\mathbf{p}_{f_1}^\top & \cdots & {}^G\mathbf{p}_{f_m}^\top \end{bmatrix}^\top \tag{4.32}$$

where:

$$\mathbf{x}_{I_k} = \begin{bmatrix} {}^{I_k}_{G}\bar{q}^\top & \mathbf{b}_{\omega_k}^\top & {}^G\mathbf{v}_{I_k}^\top & \mathbf{b}_{a_k}^\top & {}^G\mathbf{p}_{I_k}^\top \end{bmatrix}^\top \tag{4.33}$$

$$\mathbf{x}_C = \begin{bmatrix} \mathbf{x}_{T_{k-1}}^\top & \cdots & \mathbf{x}_{T_{k-c}}^\top \end{bmatrix}^\top \tag{4.34}$$

$$\mathbf{x}_L = \begin{bmatrix} {}^G\mathbf{p}_{f_1}^\top & \cdots & {}^G\mathbf{p}_{f_d}^\top \end{bmatrix}^\top \tag{4.35}$$

$$\mathbf{x}_{T_i} = \begin{bmatrix} {}^{I_i}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_i}^\top \end{bmatrix}^\top \tag{4.36}$$

where we define the "active" state $\mathbf{x}_A$ and a temporal map of $m$ historical features $\mathbf{x}_F$ for which we can loop-close to.

As compared to SEVIS-3D, some of these features will be out-of-state features for which only their marginal covariance will be tracked and updated through marginal inflation from Section 4.5.1.4. While we lose the guarantee of consistency and instead rely on the invariance of the inflation parameter, these features will be optimized in

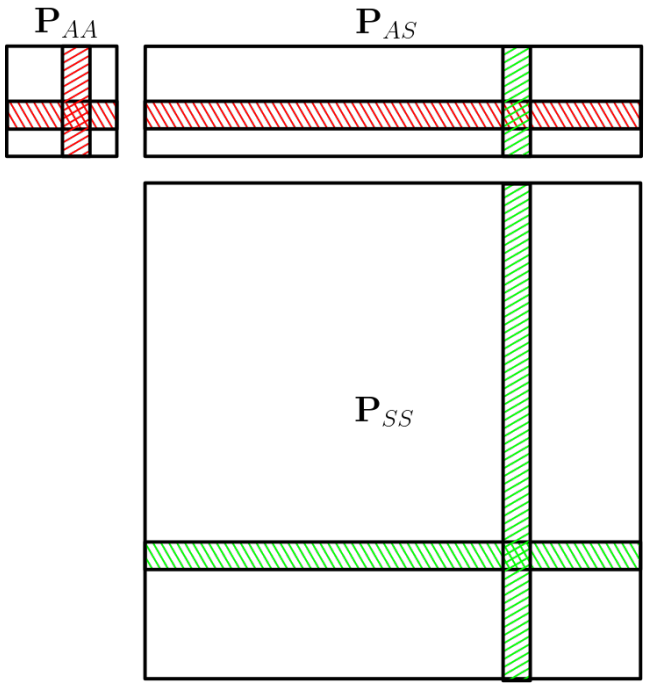



Figure 4.18: Example covariance reordering between an un-Schmidt'ed and Schmidt'ed state element. The two elements (green and red) are able to efficiently swap places through the use of a temporary variable. No covariance resizes or state reordering needs to be performed.

the secondary background thread and their uncertainties updated to properly capture the uncertainty of the prior map. A perspective is that the secondary thread performs optimization of a prior map of features which are uncorrelated with the current state due to the length of time which has passed and whose uncertainty is recovered to capture the true errors in such an optimized prior map.

#### 4.6.1.1 Dynamic Schmidt'ing via State Re-Ordering

A novel contribution is the use of dynamic Schmidt'ing of state elements allowing for information gain and optimization of observed map features. Map features which have been observed in the latest frame and have measurements that will be used for the update are un-Schmidt'ed and updated though the standard EKF Eq. (4.7). When map features are no longer observed, they are re-Schmidt'ed to improve the estimator's computational efficiency.

As shown in Figure 4.18, this is achieved through an efficient pair-wise swapping

between an un-Schmidt'ed which was previously observed, and Schmidt'ed map feature which has been recently re-observed. This efficient process enables a small local map of features to be updated (only a small subset of map features are observed in the current frame), while the correlations with other map features to still be tracked for future use through the efficient SKF update step. Specific details and full analysis can be found in Appendix E.5.

### 4.6.2 Backend – Secondary Optimization

Many long-term visual-inertial localization systems leverage the high frequency frontend and non-real-time backend design philosophy which prevents the computational complexity of the frontend from growing over time (it remains a small sliding window odometry) and allows for the backend to perform re-linearization and costly non-real-time graph optimization [10, 16, 92, 146, 159, 160]. The downside is that states corrected with global loop-closure information are either treated as true or are not leveraged by the frontend odometry to improve performance and limit drift, thus causing information to only flow from the frontend to the backend (e.g., a decoupled paradigm, Appendix F.3). We present a different method for secondary backend optimization which focuses on accurately modeling of relative pose uncertainties to enable feedback to the frontend estimator in a consistent manner.

To reduce the complexity and redundancies of the secondary pose graph, we perform keyframing. Keyframes are inserted based on a series of heuristics including: fixed frequency, max orientation and distance between, and a minimum number of common features. Specifically we minimize the keyframe pose set $\mathcal{K}$ that are constrained by the relative pose set $\mathcal{M}$:

$$\underset{\mathbf{x}_{T_i} \in \mathcal{K}}{\operatorname{argmin}} \sum_{(i,j) \in \mathcal{M}} \left|\left| \mathbf{r}_{ij}(\mathbf{x}_{T_i}, \mathbf{x}_{T_j}) \right|\right|^2_{\mathbf{P}_{r,ij}} \tag{4.37}$$

where $\mathbf{P}_{r,ij}$ is the relative 6 DoF uncertainty of the relative pose measurement. We can perform iterative optimization by linearizing at the current state estimates.

$$\Big( \sum \mathbf{H}^\top \mathbf{P}_{r,ij}^{-1} \mathbf{H} \Big) \delta\mathbf{x} = \sum \mathbf{H}^\top \mathbf{P}_{r,ij}^{-1} \mathbf{r}_{ij} \tag{4.38}$$

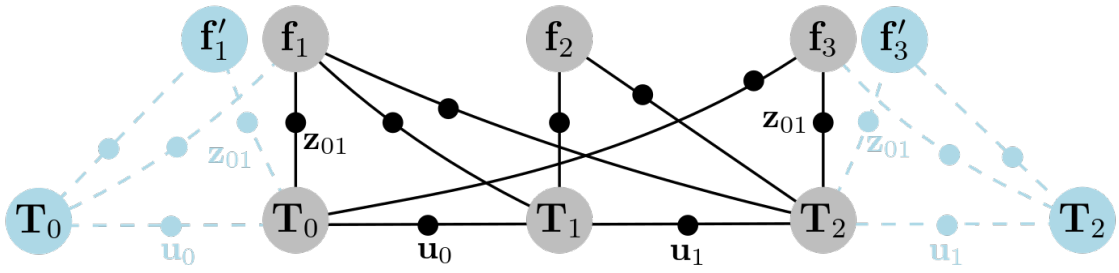



Figure 4.19: Measurements between the two keyframes are used to recover the relative information from $T_0$ to $T_2$. All inertial states and features are transformed into the $T_0$ frame, which allows for its pose to be fixed. All states besides the last pose are then marginalized to recover the relative marginal information.

The relative measurement residual cost and its Jacobian in respect to its two poses are defined as:

$$\mathbf{r}_{ij}(\mathbf{x}_{T_i}, \mathbf{x}_{T_j}) = \begin{bmatrix} -\log({}^{I_j}_{G}\mathbf{R}\,{}^{I_i}_{G}\mathbf{R}^\top) \\ {}^{I_j}_{G}\mathbf{R}({}^{G}\mathbf{p}_{I_i} - {}^{G}\mathbf{p}_{I_j}) \end{bmatrix} \tag{4.39}$$

$$\mathbf{H}_{ij} = \begin{bmatrix} -{}^{I_j}_{I_i}\mathbf{R} & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^{I_j}_{G}\mathbf{R} & \lfloor {}^{I_j}\mathbf{p}_{I_i} \times \rfloor & -{}^{I_j}_{G}\mathbf{R} \end{bmatrix} \tag{4.40}$$

The gauge freedom of a relative pose graph is 6 DoF, which can be addressed by fixing the first pose of the system. Alternatively, optimization of just a 4 DoF graph could be performed to take into account the observability of roll and pitch in VINS [160], but this introduces further errors due to treating the roll and pitch of the platform as true with zero error. We optimize using Ceres Solver [1]. Another key advantage to not estimating features, which can lose rank under insufficient observations, is we directly leverage the sparse QR covariance recovery method within Ceres as compared to a dense SVD which is prohibitively expensive.

#### 4.6.2.1 Frontend Odometry Relative Recovery

Each keyframe is connected with a relative odometry factor based on the frontend relative clone estimates. These factors thus contain the visual and inertial information used to update the frontend state, and we drop the correlations between the frontend estimates and backend relative measurements. To construct a relative,

the two global poses from the last added keyframe, $\mathbf{x}_{T_1}$, and to-be-added keyframe, $\mathbf{x}_{T_N} \in \mathbf{x}_C$, need to be transformed into $\{I_N\}$ relative frame.

$$\begin{bmatrix} {}^{I_N}_{I_1}\mathbf{R} \\ {}^{I_1}\mathbf{p}_{I_N} \end{bmatrix} = \begin{bmatrix} {}^{I_N}_{G}\mathbf{R}\,{}^{I_1}_{G}\mathbf{R}^\top \\ {}^{I_1}_{G}\mathbf{R}({}^{G}\mathbf{p}_{I_N} - {}^{G}\mathbf{p}_{I_1}) \end{bmatrix} \tag{4.41}$$

We now wish to compute the uncertainty of this relative transformation. One could directly leverage the covariance estimated by the frontend, but as shown in the later Section 4.6.3.3 simulation experiments, this directly leads to the backend becoming overconfident due to the non-independent nature of sequential relative poses. Details on how this marginal covariance could be recovered from the frontend covariance is explained in Appendix F.2. This method additionally would couple the frontend to the backend probabilistically and the sequential relative pose factors, which are highly correlated, can cause the backend keyframes to be overconfident. An alternative is to leverage a NFR [133, 190] method which enforces consistency [43, 81, 193]. While these methods can work, they solve an additional optimization problem, increasing computation, and we found their covariances to be overly conservative (they do not necessarily "tightly" bound the true uncertainty causing degradation of backend accuracy).

To address the aforementioned issues, we finally converged on leveraging a mythology similar in spirit to C-KLAM [147]. C-KLAM decouples sequential marginal information between keyframes by dropping common feature correspondences (loss of information) to retain sparsity in their optimization. Importantly, the dropping of information remains consistent in nature. To apply this to our problem, we first collect all IMU and feature observations between the two desired keyframes, thus causing a drop of information as features are considered independent from observations outside of these two keyframes (see Figure 4.19). Next, the frontend state estimates are transformed into the first keyframe frame of reference such the last inertial state is the desired relative pose between the two keyframes and the first pose at $\mathbf{x}^r_{T_1}$ can be fixed

since it is a known identity. Specifically, the $i$'th inertial state is:

$$\mathbf{x}^r_{I_i} = \begin{bmatrix} {}^{I_i}_{I_1}\bar{q} \\ {}^{I_1}\mathbf{p}_{I_i} \\ {}^{I_1}\mathbf{v}_{I_i} \\ \mathbf{b}_{g,i} \\ \mathbf{b}_{a,i} \end{bmatrix} = \begin{bmatrix} {}^{I_i}_{G}\bar{q} \otimes {}^{I_1}_{G}\bar{q}^{-1} \\ {}^{I_1}_{G}\mathbf{R}({}^{G}\mathbf{p}_{I_i} - {}^{G}\mathbf{p}_{I_1}) \\ {}^{I_1}_{G}\mathbf{R}\,{}^{G}\mathbf{v}_{I_i} \\ \mathbf{b}_{g,i} \\ \mathbf{b}_{a,i} \end{bmatrix} \tag{4.42}$$

and the gravity required for relative preintegration [40] and $j$'th feature are:

$${}^{I_1}\mathbf{g} = {}^{I_1}_{G}\mathbf{R}\,{}^{G}\mathbf{g} \tag{4.43}$$

$${}^{I_1}\mathbf{p}_{f_j} = {}^{I_1}_{G}\mathbf{R}({}^{G}\mathbf{p}_{f_j} - {}^{G}\mathbf{p}_{I_1}) \tag{4.44}$$

Then a classical batch-MLE visual-inertial problem is formulated using the inertial and feature observations:

$$\mathcal{C} = \sum_{i=1}^{N} \frac{1}{2} \left|\left| \mathbf{x}^r_{I_{i+1}} \boxminus \mathbf{f}(\mathbf{x}^r_{I_i}, {}^{I_1}\mathbf{g}, \mathbf{a}_k, \boldsymbol{\omega}_k) \right|\right|^2_{\mathbf{Q}_i} + \sum_{\mathbf{z}_{i,j} \in \mathcal{Z}_{1:N}} \frac{1}{2} \left|\left| \mathbf{z}_{ij} \boxminus \mathbf{h}(\mathbf{x}^r_{I_i}, {}^{I_1}\mathbf{p}_{f_j}) \right|\right|^2_{\mathbf{R}_{ij}} \tag{4.45}$$

where the set $\mathcal{Z}_{1:N}$ denotes all measurements that occurred between the two keyframes. The information hessian of this problem is then recovered after linearization and marginalization of states are performed.

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{NN} & \mathbf{A}_{NX} \\ \mathbf{A}_{XN} & \mathbf{A}_{XX} \end{bmatrix} \tag{4.46}$$

where we have denoted all other states as the "X" sub-script and the desired $\mathbf{x}^r_{T_N} = [{}^{I_N}_{I_1}\bar{q}^\top \ {}^{I_1}\mathbf{p}^\top_{I_N}]^\top$ as "N". The block-diagonal structure of the environmental features is leverage to efficiently marginalize them [66], which is followed by the marginalization of all states (including the relative velocity and biases at $\{I_1\}$) besides the last frame's pose $\mathbf{x}^r_{T_N}$.

$$\mathbf{P}^{-1}_{r,1N} := \mathbf{A}_{r,1N} = \mathbf{A}_{NN} - \mathbf{A}_{NX}\mathbf{A}^{-1}_{XX}\mathbf{A}_{XN} \tag{4.47}$$

This recovered marginal information can now be directly used to weight the relative transformation, see Eq. (4.37), between each keyframe.

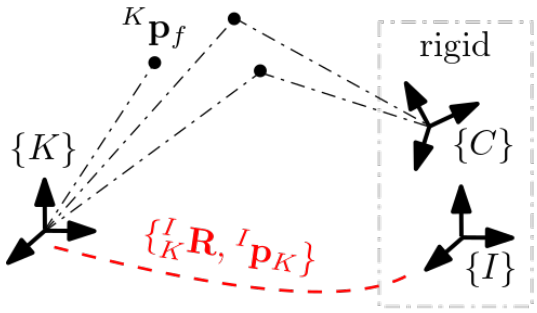



Figure 4.20: Frames involved in the PnP problem. The camera $\{C\}$ to keyframe $\{K\}$ pose is recovered through PnP. We wish to have the uncertainty between the keyframe and inertial frame $\{I\}$ which is rigidly connected to the camera.

#### 4.6.2.2 PnP-based Relative Loop-closures

When keyframes are added to the secondary pose graph, their loop-closure information with prior keyframes is leveraged (e.g., DBoW2 [53], CALC [136, 137], or placeless [125] could be leveraged, see [180]). Given a match between the current keyframe and a historical keyframe, we wish to construct a relative pose measurement between the two and also recover the uncertainty of this relative transformation. To find the relative transform we use RANSAC Lambda twist PnP solver [156] which allows for recovery of the relative pose between the historical keyframe 3d feature pointcloud and the matched feature 2d coordinates of the newest keyframe (one could leverage an uncertainty aware PnP [191]).

Now we can derive how to recover the uncertainty of this PnP transformation. This process is similar in mythology to the closed-form recovery of 3D ICP covariance [157]. The frame of references involved are overview in Figure 4.20. We wish to recover the uncertainty of the keyframe to inertial frame transformation using the inlier set of features. There are two sources of error: (1) the feature observations from $\{C\}$ and (2) the 3d position of the features in the keyframe. We can define these as:

$$\mathbf{z}_k = \mathbf{h}({}^{I}_{K}\mathbf{R}, {}^{I}\mathbf{p}_K, {}^{K}\mathbf{p}_f) + \mathbf{n}_k \tag{4.48}$$

$${}^{K}\mathbf{p}_{f,m} = {}^{K}\mathbf{p}_f + \mathbf{n}_p \tag{4.49}$$

The uncertainty of the feature in the keyframe, $\mathbf{n}_p \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_p)$, is provided by the frontend if the feature is a SLAM feature or can be set to a fixed inflation value

otherwise (e.g., 0.5cm position noise). The visual feature measurement model is defined similarly to Eq. (2.49) as follows:

$$\mathbf{z}_k = \boldsymbol{\Lambda}({}^{C}\mathbf{p}_f) + \mathbf{n}_k \tag{4.50}$$

$${}^{C}\mathbf{p}_f = {}^{C}_{I}\mathbf{R}^{I}_{K}\mathbf{R}\,{}^{K}\mathbf{p}_f + {}^{C}_{I}\mathbf{R}\,{}^{I}\mathbf{p}_K + {}^{C}\mathbf{p}_I \tag{4.51}$$

We can formulate a weighted minimization problem as:

$$\underset{({}^{I}_{K}\mathbf{R}, {}^{I}\mathbf{p}_K, {}^{K}\mathbf{p}_f)}{\operatorname{argmin}} \sum \left|\left| \mathbf{z}_k - \mathbf{h}({}^{I}_{K}\mathbf{R}, {}^{I}\mathbf{p}_K, {}^{K}\mathbf{p}_f) \right|\right|^2_{\sigma^2_{pix}\mathbf{I}_2} + \left|\left| {}^{K}\mathbf{p}_{f,m} - {}^{K}\mathbf{p}_f \right|\right|^2_{\mathbf{R}_p} \tag{4.52}$$

The uncertainty of the state transform after marginalization of the feature ${}^{K}\mathbf{p}_f$ can be shown to be the following:

$$\mathbf{P}_{r,KI} = \left( \sum \mathbf{H}_x^\top \left( \sigma^2_{pix}\mathbf{I} + \mathbf{H}_f \mathbf{R}_p \mathbf{H}_f^\top \right)^{-1} \mathbf{H}_x \right)^{-1} \tag{4.53}$$

where we have defined the following Jacobians:

$$\mathbf{H}_x = \mathbf{H}_\pi \begin{bmatrix} {}^{C}_{I}\mathbf{R} \left\lfloor {}^{I}_{K}\mathbf{R}\,{}^{K}\mathbf{p}_f \times \right\rfloor & {}^{C}_{I}\mathbf{R} \end{bmatrix}$$

$$\mathbf{H}_f = \mathbf{H}_\pi {}^{C}_{I}\mathbf{R}^{I}_{K}\mathbf{R}\ ,\ \mathbf{H}_\pi = \frac{1}{z}\begin{bmatrix} 1 & 0 & -x/z \\ 0 & 1 & -y/z \end{bmatrix}$$

This derivation is specific to the single-view problem. This uncertainty was verified in simulation to be consistent in nature. After the covariance of the PnP transform is recovered, the factor is appended to the relative pose graph and optimized.

#### 4.6.2.3 Frontend Feedback

Marginalized features are not estimated to reduce the complexity of the estimation problem. To allow for corrections in the keyframe states to improve the feature map, we anchor features in their closest keyframe state. One can argue that the estimate of the feature in a keyframe should have sufficient accuracy as compared to their global estimate which can be biased by odometry drift. To recover the current best estimate for a feature anchored in the keyframe $\{K\}$ we can do the following:

$${}^{G}\mathbf{p}_{f,opt} = {}^{K}_{G}\mathbf{R}^\top_{opt}\,{}^{K}\mathbf{p}_{f,vio} + {}^{G}\mathbf{p}_{K,opt} \tag{4.54}$$

To recover the uncertainty of the 3D features, we leverage a marginal estimate of the feature in the last seen keyframe $\mathbf{P}_{f,A}$ computed by the frontend before marginalization of the feature from the state. This uncertainty, along with the uncertainty of the keyframe in the secondary graph can then the approximately propagated as:

$$\mathbf{P}_{f,G} \simeq \mathbf{H}_{f,GA} \begin{bmatrix} \mathbf{P}_{KK} & \mathbf{0}_{6\times 3} \\ \mathbf{0}_{3\times 6} & \mathbf{P}_{f,A} \end{bmatrix} \mathbf{H}_{f,GA}^{\top} \tag{4.55}$$

$$\mathbf{H}_{f,GA} = \begin{bmatrix} -{}^{K}_{G}\mathbf{R}_{opt}^{\top} \lfloor {}^{K}\mathbf{p}_{f,vio} \times \rfloor & \mathbf{I}_3 & {}^{K}_{G}\mathbf{R}_{opt}^{\top} \end{bmatrix}$$

The covariance of the optimized keyframes, $\mathbf{P}_{KK}$, can be recovered using Ceres Solver [1] and is the most costly part of the secondary optimization process.

**Remarks:** We contrast this against the methods by Mourikis and Roumeliotis [140] and Sartipi et al. [174] which perform a full VI-BLS which optimizes both keyframes and feature estimates. This optimization problem quickly grows to become computationally infeasible, especially with a large amount of features. Sartipi et al. [174] address this by optimizing a sub-section of the trajectory and fixing it after the computational cost exceeds 6 seconds, but then requires trajectory segments to be aligned introducing errors. Additionally, to reduce the complexity of recovering the covariance of environmental features, they treat the observing camera pose as true and recover the conditional uncertainty of the feature given its pose, possibly introducing large errors if the poses are not sufficiently accurate.

The proposed relative pose graph is more computationally efficient (can be solved through sparse methods due to being well-constrained and is smaller in size) compared to VI-BLS, and additionally recovers the uncertainty of the keyframe pose, which is then leveraged to recover the uncertainty of the feature in the global. As shown in the later experiments, Section 4.6.3, this backend is both consistent and efficient and enables the reconstruction of a consistent sparse point map online. While we allow or pose graph to grow over time due to its efficiency, in the future we will investigate approaches which perform marginalization and sparsification on this relative pose graph to bound its complexity [17, 30, 85], but the main focus of this manuscript is

Table 4.8: Simulation parameters and prior standard deviations that perturbations of measurements were drawn from.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Gyro. White Noise | 1.6968e-04 | Gyro. Rand. Walk | 1.9393e-05 |
| Accel. White Noise | 2.0000e-3 | Accel. Rand. Walk | 3.0000e-3 |
| Avg. Feats | 40 | Avg. Feats Map | 20 |
| Num. SLAM | 15 | Camera Noise (px) | 1 |
| Cam Freq. (hz) | 10 | IMU Freq. (hz) | 400 |
| KF Max Dist (m) | 1.0 | KF Max Ori (deg) | 50.0 |
| KF Com. Feat. (%) | 50.0 | Inflation $\mu$ | 500 |
| Num. Clones | 11 | Feat. Rep. | GLOBAL |

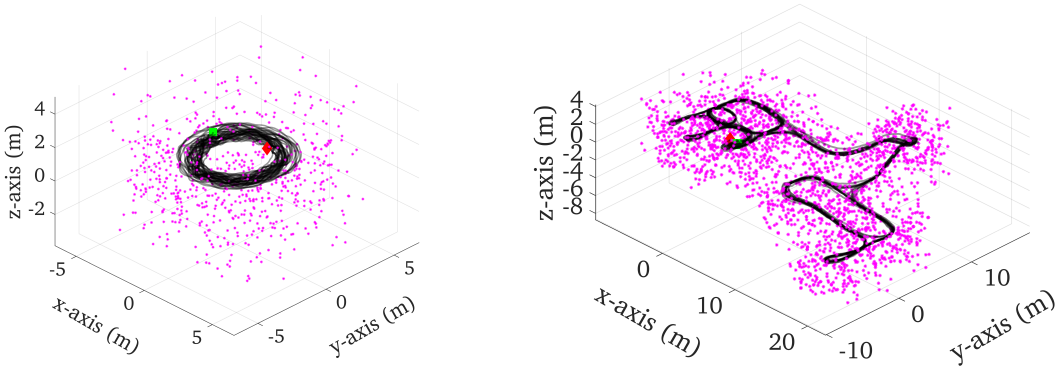



Figure 4.21: Small-scale long-term `Room` trajectory (left) and two floor `Spencer` lab trajectory (right). Both were generated from a visual-inertial odometry method, and thus provide realistic hand-held motion. The Room dataset has 669 map points, is 748 meters in length, and has an average velocity of 1.37 m/s. The Spencer dataset has 2093 map points, is 1612 meters in length, and has an average velocity of 1.35 m/s.

to investigate how feedback can be leveraged. This can be done since marginalization and sparsification can be done consistently [43, 81, 193].

### 4.6.3 Simulation Results

We performed a series of detailed simulations to validate the design decisions of the proposed system. We employ the OpenVINS simulator to generate realistic visual-bearing and inertial measurements from the real-world trajectory generated by an existing VINS (see Figure 4.21). We consider two trajectories, one small-scale, the other a much larger multi-floor dataset, to contrast the inherited odometry drift

which can affect the consistency of the inflation-based measurement update, see Section 4.5.1.4, and thus motivating the need to leverage secondary optimization to ensure global consistency of the estimator. Simulation parameters used are documented in Table 4.8. First-estimates Jacobians (FEJ) [83, 84] were used to improve the estimator consistency as the use of environmental landmarks is known to introduce inconsistent information gains. For metrics we report the Absolute Trajectory Error (ATE), Normalized Estimation Error Squared (NEES), and Relative Pose Error (RPE) throughout the different experiments (see [221] and [4]). NEES's magnitude should match the 3 DoF orientation and position state sizes. All timings were run on an Intel(R) Xeon(R) CPU E3-1505M v6 @ 3.00GHz processor with single-threaded frontend estimation. The backend optimization was optimized using Ceres Solver [1] with a max of 6 threads. Place recognition (loop-closure) to historical keyframes through feature matching was simulated by selecting the closest keyframe and finding correspondences using groundtruth labels (thus perfect matching). Note that a subset of the active feature tracks, see Table 4.8, are considered map points to ensure that there are always non-map features available.

In simulation we compare against two different baselines: a Covariance Intersection (CI)-based frontend, and decoupled backend thread which does not perform any feedback of information to the frontend when used. The details of the CI-based EKF is discussed in Appendix F.1, while the decoupled backend is described in Appendix F.3. The CI method was hand-tuned to have a weight of $w_0 = 0.993$ for the non-map states while the map states were equally distributed with the remaining weight such that all weights together sum to one. The CI feature map is not updated by the frontend and remains fixed.

#### 4.6.3.1 Experiment 1: Impact of Dynamic Schmidt'ing

The first investigation is how the proposed dynamic Schmidt'ing can impact the estimation performance and map quality. To isolate the impact of this process from the incremental map-building SLAM problem, we provide the map a priori to

the estimator with known uncertainty bounds. If the estimator performs dynamic Schmidt'ing then the prior map can be improved, otherwise its mean and uncertainty will remain fixed through the standard SKF methodology. Details on how the simulated map was generated can be found in Section 4.5 and technical report [62, 63]. From a high level, a series of keyframes are generated with a random set of features, from which the feature bearing and depth are perturbed from the given feature distribution. The feature means and covariance are then transformed into the global frame and provided as a global map with uncertainty to the estimator.

The results have been tabulated in Table 4.9 for different prior map noise levels. First we can see that the EKF method which optimizes the map is able to have the most accurate trajectory and map. Additionally, we can see that all methods are able to outperform the VIO method which does not leverage any loop-closure information. As the maps get more uncertain, the accuracy of methods which do not allow for the map to be updated are hurt significantly as compared to ones that do. We can conclude that the trajectory accuracy performance of the methods which do not improve the given a priori map (e.g. Fixed-SKF, CI, Inf. Marg) is limited by the quality of the map. EKF and Dynamic-SKF-based methods are able to improve the trajectory and map accuracy but have a very small increase in trajectory error which we equate to poor linearization points leveraged via FEJ (over $\sim 20cm$ of linearization errors are likely introduced just from the features) [22].

It can be seen that the use of dynamic Schmidt'ing has improved performance which can nearly reach the level of full EKF state estimation. The accuracy doesn't reach the EKF exactly, as features are only un-Schmidt'ed when observed, thus correlations from other feature updates will not be able to correct feature estimates when they are not observed. We can also see that for the proposed hybrid method SKF+Marg. the inclusion of more SKF features at 25% has a clear improvement of performance while having more features in the Marg. state, which are updated through the measurement inflation method presented in Section 4.5.1.4, has an accuracy drop but large computational advantages. A mix between these two can provide the best of both and

is our aim.

Looking at the baseline CI map-based method, detailed in Appendix F.1, it is able to perform with accuracy near that of the inflation method with an overly conservative covariance (small NEES). We can see that when CI is combined with a mixture of SKF features, the performance of the estimator actually becomes worse than CI itself. This is counter-intuitive. One can think about how the CI update will affect SKF feature uncertainties. It will increase the uncertainty of the SKF map features, which are unable to reduce their uncertainty due to the SKF update, see Eq. (4.10). This means that all SKF features become more and more uncertain as CI updates are performed, losing significant amounts of information. This causes the system with high amounts of SKF features, +CI (25%) with 75% of the map being SKF map features, to perform far worse than a method with only a few SKF features, +CI (25%) where only 25% of the map is SKF. Thus we can't recommend the combined hybrid use of the SKF and CI within an estimator due to this effect.

#### 4.6.3.2 Experiment 2: Hybrid Estimators' Complexity-Accuracy Trade-off

We now focus on the proposed *hybrid* estimator which combines the SKF and Marg. feature inflation updates. The key question to ask is if this combination allows for the benefit of higher accuracy due to tracking of correlations of some features, while a reduction in computation with a small accuracy loss due to using the Marg. feature inflation. The proposed hybrid method was tested with different ratios of SKF to Marg. with a higher percentage representing more features being in the Marg. map state. The results shown in Table 4.10 are for different numbers of max map points allowed for each method. The map was built online for all methods, but the Fixed-SKF and Marg. methods are unable to improve the feature estimates once inserted into the map (i.e. incremental map building). When the map reaches its max capacity, a random feature from the map is removed to make room for the to-be-inserted newly inserted SLAM feature.

Table 4.9: Average ATE over 5 dataset runs of the Room dataset for different map priors and algorithms.

| | | Algorithm | ATE (deg / m) | NEES (3) | Map Pts (cm) | Time (s) |
|---|---|---|---|---|---|---|
| | | VIO | 1.256 / 0.154 | 2.500 / 3.630 | - | 0.0037 ± 0.0010 |
| 0.5°, 12cm Feat. Prior | | EKF | 0.027 / 0.003 | 3.153 / 3.514 | 0.792 ± 2.640 | 0.0878 ± 0.0030 |
| | | CI | 0.221 / 0.033 | 0.167 / 0.117 | 10.580 ± 6.567 | 0.0022 ± 0.0003 |
| | | Inf. Marg. | 0.191 / 0.034 | 2.088 / 0.643 | 10.580 ± 6.567 | 0.0019 ± 0.0003 |
| | Fixed SKF | SKF | 0.073 / 0.012 | 3.067 / 3.484 | 10.580 ± 6.567 | 0.0118 ± 0.0011 |
| | | +CI (25%) | 0.340 / 0.058 | 0.468 / 0.281 | 10.580 ± 6.567 | 0.0081 ± 0.0009 |
| | | +CI (50%) | 0.262 / 0.044 | 0.264 / 0.194 | 10.580 ± 6.567 | 0.0048 ± 0.0006 |
| | | +CI (75%) | 0.216 / 0.034 | 0.203 / 0.146 | 10.580 ± 6.567 | 0.0026 ± 0.0004 |
| | | +Marg. (25%) | 0.079 / 0.013 | 3.039 / 3.508 | 10.580 ± 6.567 | 0.0078 ± 0.0009 |
| | | +Marg. (50%) | 0.092 / 0.016 | 3.158 / 3.655 | 10.580 ± 6.567 | 0.0045 ± 0.0006 |
| | | +Marg. (75%) | 0.106 / 0.021 | 2.900 / 3.388 | 10.580 ± 6.567 | 0.0023 ± 0.0003 |
| | Dyn. SKF | SKF | 0.033 / 0.003 | 3.535 / 3.753 | 0.852 ± 2.620 | 0.0257 ± 0.0083 |
| | | +Marg. (25%) | 0.034 / 0.004 | 3.142 / 3.944 | 3.339 ± 5.878 | 0.0141 ± 0.0042 |
| | | +Marg. (50%) | 0.040 / 0.004 | 3.285 / 3.447 | 5.752 ± 7.013 | 0.0069 ± 0.0016 |
| | | +Marg. (75%) | 0.046 / 0.005 | 2.860 / 3.227 | 8.232 ± 7.262 | 0.0028 ± 0.0005 |
| 1.0°, 24cm Feat. Prior | | EKF | 0.033 / 0.003 | 3.615 / 4.127 | 1.583 ± 5.589 | 0.2426 ± 0.0063 |
| | | CI | 0.345 / 0.052 | 0.120 / 0.087 | 21.160 ± 13.136 | 0.0037 ± 0.0004 |
| | | Inf. Marg. | 0.316 / 0.059 | 2.107 / 0.677 | 21.160 ± 13.136 | 0.0033 ± 0.0003 |
| | Fixed SKF | SKF | 0.129 / 0.022 | 3.635 / 3.849 | 21.160 ± 13.136 | 0.0291 ± 0.0014 |
| | | +CI (25%) | 0.538 / 0.090 | 0.390 / 0.238 | 21.160 ± 13.136 | 0.0203 ± 0.0012 |
| | | +CI (50%) | 0.416 / 0.070 | 0.210 / 0.159 | 21.160 ± 13.136 | 0.0117 ± 0.0011 |
| | | +CI (75%) | 0.339 / 0.052 | 0.153 / 0.115 | 21.160 ± 13.136 | 0.0062 ± 0.0007 |
| | | +Marg. (25%) | 0.137 / 0.024 | 3.518 / 3.830 | 21.160 ± 13.136 | 0.0189 ± 0.0011 |
| | | +Marg. (50%) | 0.162 / 0.030 | 3.649 / 3.870 | 21.160 ± 13.136 | 0.0111 ± 0.0011 |
| | | +Marg. (75%) | 0.175 / 0.038 | 3.063 / 3.529 | 21.160 ± 13.136 | 0.0059 ± 0.0007 |
| | Dyn. SKF | SKF | 0.041 / 0.004 | 4.020 / 4.625 | 1.658 ± 5.533 | 0.0378 ± 0.0092 |
| | | +Marg. (25%) | 0.046 / 0.006 | 3.597 / 5.499 | 6.626 ± 11.780 | 0.0273 ± 0.0063 |
| | | +Marg. (50%) | 0.052 / 0.006 | 3.551 / 4.099 | 11.452 ± 14.102 | 0.0148 ± 0.0032 |
| | | +Marg. (75%) | 0.059 / 0.007 | 2.915 / 3.239 | 16.440 ± 14.572 | 0.0069 ± 0.0010 |

Table 4.10: Average RPE over 5 runs of the `Room` dataset for different map sizes lengths. RPE units are in degrees and meters. The map is built incrementally and not improved unless Dynamic-SKF is leveraged.

| | Map Size | 40m | 80m | 120m | 160m | 200m | 240m | NEES (ori / pos) | Map Pts (cm) | Time (s) |
|---|---|---|---|---|---|---|---|---|---|---|
| VIO | - | 0.445 / 0.044 | 0.607 / 0.062 | 0.701 / 0.074 | 0.820 / 0.086 | 0.919 / 0.094 | 1.006 / 0.102 | 2.407 / 2.723 | - | 0.0040 ± 0.0011 |
| Inf. Marg. | 100pts | 0.132 / 0.030 | 0.185 / 0.035 | 0.232 / 0.038 | 0.273 / 0.042 | 0.302 / 0.043 | 0.325 / 0.044 | 5.435 / 7.820 | 7.185 ± 3.782 | 0.0046 ± 0.0010 |
| | 200pts | 0.130 / 0.033 | 0.163 / 0.035 | 0.189 / 0.036 | 0.210 / 0.037 | 0.219 / 0.037 | 0.224 / 0.037 | 5.405 / 4.846 | 5.843 ± 3.147 | 0.0043 ± 0.0008 |
| | 300pts | 0.129 / 0.035 | 0.152 / 0.036 | 0.165 / 0.036 | 0.176 / 0.037 | 0.181 / 0.037 | 0.187 / 0.036 | 5.496 / 4.114 | 5.675 ± 3.036 | 0.0044 ± 0.0007 |
| | 400pts | 0.137 / 0.038 | 0.169 / 0.039 | 0.188 / 0.039 | 0.198 / 0.040 | 0.199 / 0.040 | 0.202 / 0.039 | 5.316 / 3.205 | 5.527 ± 3.222 | 0.0043 ± 0.0006 |
| Fixed SKF | 100pts | 0.078 / 0.011 | 0.084 / 0.011 | 0.085 / 0.011 | 0.088 / 0.011 | 0.088 / 0.011 | 0.089 / 0.011 | 1.951 / 0.704 | 2.509 ± 2.061 | 0.0052 ± 0.0012 |
| | 200pts | 0.072 / 0.011 | 0.076 / 0.011 | 0.078 / 0.011 | 0.080 / 0.011 | 0.081 / 0.012 | 0.079 / 0.012 | 2.028 / 0.679 | 2.709 ± 2.123 | 0.0063 ± 0.0015 |
| | 300pts | 0.073 / 0.011 | 0.074 / 0.011 | 0.076 / 0.011 | 0.080 / 0.011 | 0.082 / 0.012 | 0.085 / 0.012 | 2.156 / 0.702 | 2.893 ± 2.242 | 0.0073 ± 0.0020 |
| | 400pts | 0.076 / 0.012 | 0.080 / 0.012 | 0.081 / 0.012 | 0.083 / 0.012 | 0.085 / 0.013 | 0.087 / 0.013 | 2.110 / 0.765 | 3.096 ± 2.459 | 0.0086 ± 0.0026 |
| Fixed Hybrid | 200pts (25%) | 0.082 / 0.013 | 0.098 / 0.013 | 0.112 / 0.014 | 0.124 / 0.015 | 0.133 / 0.015 | 0.141 / 0.016 | 5.151 / 2.558 | 3.234 ± 2.550 | 0.0063 ± 0.0014 |
| | 200pts (50%) | 0.078 / 0.013 | 0.086 / 0.013 | 0.094 / 0.013 | 0.102 / 0.014 | 0.105 / 0.014 | 0.106 / 0.015 | 3.305 / 2.746 | 3.381 ± 2.612 | 0.0056 ± 0.0011 |
| | 200pts (75%) | 0.091 / 0.016 | 0.100 / 0.015 | 0.110 / 0.016 | 0.114 / 0.016 | 0.115 / 0.016 | 0.117 / 0.017 | 4.711 / 3.728 | 3.340 ± 2.656 | 0.0049 ± 0.0009 |
| Dyn. SKF | 100pts | 0.059 / 0.007 | 0.064 / 0.007 | 0.065 / 0.008 | 0.067 / 0.008 | 0.069 / 0.008 | 0.068 / 0.008 | 2.058 / 0.465 | 1.639 ± 1.511 | 0.0067 ± 0.0017 |
| | 200pts | 0.049 / 0.006 | 0.052 / 0.006 | 0.052 / 0.006 | 0.055 / 0.006 | 0.058 / 0.006 | 0.057 / 0.007 | 2.027 / 0.380 | 1.210 ± 0.973 | 0.0088 ± 0.0024 |
| | 300pts | 0.048 / 0.006 | 0.050 / 0.005 | 0.051 / 0.006 | 0.053 / 0.006 | 0.054 / 0.006 | 0.054 / 0.006 | 2.126 / 0.373 | 1.000 ± 0.578 | 0.0108 ± 0.0034 |
| | 400pts | 0.046 / 0.005 | 0.048 / 0.005 | 0.049 / 0.005 | 0.051 / 0.005 | 0.053 / 0.006 | 0.053 / 0.006 | 2.099 / 0.357 | 0.944 ± 0.533 | 0.0127 ± 0.0048 |
| Dyn. Hybrid | 200pts (25%) | 0.061 / 0.007 | 0.066 / 0.007 | 0.071 / 0.007 | 0.075 / 0.007 | 0.077 / 0.008 | 0.079 / 0.008 | 3.814 / 3.026 | 1.565 ± 1.132 | 0.0085 ± 0.0022 |
| | 200pts (50%) | 0.062 / 0.007 | 0.069 / 0.007 | 0.072 / 0.007 | 0.075 / 0.008 | 0.075 / 0.008 | 0.074 / 0.008 | 5.807 / 9.440 | 1.814 ± 1.234 | 0.0073 ± 0.0018 |
| | 200pts (75%) | 0.068 / 0.010 | 0.074 / 0.010 | 0.076 / 0.010 | 0.079 / 0.010 | 0.078 / 0.010 | 0.077 / 0.011 | 6.949 / 16.017 | 2.518 ± 1.719 | 0.0059 ± 0.0012 |

Table 4.11: Average factor and state errors of a representative run on the `Spencer` dataset for different relative pose frequencies. ATE is in units of degrees and meters with the pose NEES also being reported. The frontend does not leverage a map (e.g. it is VIO). The backend is the proposed relative pose graph with only relative pose uncertainty from VIO (MARG), equal weighted relative (EQUAL), or proposed relative C-KLAM method (C-KLAM).

| | Algo. | Rel. ATE | NEES(6) | # | PnP ATE | NEES(6) | # | Kfs ATE | NEES(6) | ${}^{A}\mathbf{p}_f$ | NEES(3) | ${}^{G}\mathbf{p}_f$ | NEES(3) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Without LC | MARG-0.3hz | 0.030 / 0.010 | 6.430 | 538 | - | - | - | 0.585 / 0.284 | 13.382 | 0.029 | 3.037 | 0.203 | 3.075 |
| | MARG-0.5hz | 0.026 / 0.009 | 6.530 | 672 | - | - | - | 0.599 / 0.290 | 15.213 | 0.029 | 3.037 | 0.211 | 3.595 |
| | MARG-1.0hz | 0.019 / 0.006 | 6.339 | 1094 | - | - | - | 0.629 / 0.316 | 19.415 | 0.029 | 3.037 | 0.223 | 4.696 |
| | C-KLAM-0.3hz | 0.030 / 0.010 | 6.430 | 538 | - | - | - | 0.585 / 0.284 | 1.416 | 0.029 | 3.037 | 0.203 | 0.360 |
| | C-KLAM-0.5hz | 0.026 / 0.009 | 6.530 | 672 | - | - | - | 0.599 / 0.290 | 1.160 | 0.029 | 3.037 | 0.211 | 0.326 |
| | C-KLAM-1.0hz | 0.019 / 0.006 | 6.339 | 1094 | - | - | - | 0.629 / 0.316 | 0.638 | 0.029 | 3.037 | 0.223 | 0.182 |
| With PnP LC | MARG-0.3hz | 0.030 / 0.010 | 6.430 | 538 | 0.324 / 0.018 | 9.572 | 63 | 0.397 / 0.130 | 9.752 | 0.029 | 3.037 | 0.135 | 6.158 |
| | MARG-0.5hz | 0.026 / 0.009 | 6.530 | 672 | 0.336 / 0.020 | 11.109 | 92 | 0.260 / 0.056 | 9.219 | 0.029 | 3.037 | 0.070 | 6.104 |
| | MARG-1.0hz | 0.019 / 0.006 | 6.339 | 1094 | 0.316 / 0.018 | 9.428 | 162 | 0.225 / 0.042 | 11.366 | 0.029 | 3.037 | 0.059 | 8.307 |
| | C-KLAM-0.3hz | 0.030 / 0.010 | 6.430 | 538 | 0.324 / 0.018 | 9.572 | 63 | 0.274 / 0.108 | 1.170 | 0.029 | 3.037 | 0.108 | 0.769 |
| | C-KLAM-0.5hz | 0.026 / 0.009 | 6.530 | 672 | 0.336 / 0.020 | 11.109 | 92 | 0.188 / 0.048 | 2.738 | 0.029 | 3.037 | 0.063 | 2.532 |
| | C-KLAM-1.0hz | 0.019 / 0.006 | 6.339 | 1094 | 0.316 / 0.018 | 9.428 | 162 | 0.209 / 0.037 | 1.664 | 0.029 | 3.037 | 0.055 | 1.751 |
| | EQUAL-0.3hz | 0.030 / 0.010 | - | 538 | 0.324 / 0.018 | - | - | 0.356 / 0.139 | - | 0.029 | - | 0.125 | - |
| | EQUAL-0.5hz | 0.026 / 0.009 | - | 672 | 0.336 / 0.020 | - | - | 0.437 / 0.096 | - | 0.029 | - | 0.114 | - |
| | EQUAL-1.0hz | 0.019 / 0.006 | - | 1094 | 0.316 / 0.018 | - | - | 0.492 / 0.093 | - | 0.029 | - | 0.106 | - |

Table 4.12: Average ATE over 5 runs of the `Spencer` dataset for different relative pose frequencies. The frontend does not leverage a map (e.g. it is VIO). The non-VIO pose and uncertainty is of the backend reported using the decoupled pose recovery (see Appendix F.3).

| | **Algo.** | **ATE (deg/m)** | **NEES(3)** | **Time (s)** |
|---|---|---|---|---|
| VIO | - | 1.349 / 0.362 | 2.826 / 5.450 | 0.0083 ± 0.0020 |
| Without LC | MARG-0.3hz | 1.425 / 0.471 | 16.851 / 1.673 | 0.0413 ± 0.0276 |
| | MARG-0.5hz | 1.429 / 0.500 | 20.449 / 1.863 | 0.0525 ± 0.0368 |
| | MARG-1.0hz | 1.427 / 0.509 | 26.954 / 2.028 | 0.1010 ± 0.0775 |
| | C-KLAM-0.3hz | 1.425 / 0.471 | 1.305 / 0.755 | 0.0964 ± 0.0426 |
| | C-KLAM-0.5hz | 1.305 / 0.490 | 1.466 / 0.753 | 0.0876 ± 0.0389 |
| | C-KLAM-1.0hz | 1.427 / 0.509 | 0.630 / 0.524 | 0.1181 ± 0.0777 |
| With PnP LC | MARG-0.3hz | 0.961 / 0.245 | 18.681 / 0.556 | 0.0569 ± 0.0445 |
| | MARG-0.5hz | 0.633 / 0.153 | 18.761 / 0.272 | 0.0836 ± 0.0693 |
| | MARG-1.0hz | 0.732 / 0.175 | 26.424 / 0.425 | 0.1899 ± 0.1694 |
| | C-KLAM-0.3hz | 0.872 / 0.228 | 1.892 / 0.320 | 0.1129 ± 0.0559 |
| | C-KLAM-0.5hz | 0.450 / 0.119 | 1.826 / 0.242 | 0.1189 ± 0.0713 |
| | C-KLAM-1.0hz | 0.563 / 0.140 | 0.978 / 0.164 | 0.2087 ± 0.1716 |
| | EQUAL-0.3hz | 0.968 / 0.245 | - | 0.0567 ± 0.0445 |
| | EQUAL-0.5hz | 0.760 / 0.168 | - | 0.0831 ± 0.0690 |
| | EQUAL-1.0hz | 0.740 / 0.158 | - | 0.1912 ± 0.1714 |

Table 4.13: Average RPE over 5 runs of the Spencer dataset for different algorithms. RPE units are in degrees and meters, with a max total of 400 map features being leveraged. Dynamic Schmidt'ing is used for all methods which have SKF features. The backend relative pose optimization uses a clone rate of 0.5Hz and the proposed relative C-KLAM for all methods. We show the configurations for frontend estimation without feedback (F), backend without feedback (B), frontend leveraging the optimized backend map (F+B(F)), and backend leveraging the frontend which leverages the optimized backend map (F+B(B)).

| Algo. | Config | 40m | 80m | 120m | 160m | 200m | 240m | NEES (ori/pos) | Time (s) |
|---|---|---|---|---|---|---|---|---|---|
| VIO | F | 0.278 / 0.067 | 0.404 / 0.101 | 0.476 / 0.125 | 0.536 / 0.144 | 0.596 / 0.164 | 0.645 / 0.181 | 2.962 / 4.968 | 0.0084 ± 0.0020 |
| | B | 0.254 / 0.069 | 0.304 / 0.089 | 0.329 / 0.100 | 0.344 / 0.096 | 0.350 / 0.086 | 0.351 / 0.080 | 1.827 / 0.242 | 0.1046 ± 0.0559 |
| EKF | F | 0.075 / 0.020 | 0.079 / 0.023 | 0.078 / 0.024 | 0.080 / 0.025 | 0.080 / 0.025 | 0.080 / 0.025 | 2.175 / 1.095 | 0.0789 ± 0.0296 |
| | B | 0.126 / 0.023 | 0.132 / 0.030 | 0.136 / 0.033 | 0.139 / 0.033 | 0.136 / 0.028 | 0.133 / 0.025 | 1.139 / 0.288 | 0.3713 ± 0.2561 |
| SKF | F | 0.087 / 0.022 | 0.094 / 0.026 | 0.094 / 0.028 | 0.100 / 0.029 | 0.097 / 0.029 | 0.099 / 0.028 | 2.128 / 1.274 | 0.0285 ± 0.0085 |
| | B | 0.141 / 0.028 | 0.151 / 0.035 | 0.155 / 0.038 | 0.155 / 0.038 | 0.150 / 0.033 | 0.148 / 0.029 | 2.155 / 0.375 | 0.3739 ± 0.2569 |
| Inf. Marg. | F | 0.152 / 0.043 | 0.213 / 0.057 | 0.265 / 0.069 | 0.309 / 0.082 | 0.352 / 0.092 | 0.394 / 0.102 | 11.434 / 64.794 | 0.0085 ± 0.0020 |
| | F+B (F) | 0.183 / 0.058 | 0.237 / 0.077 | 0.263 / 0.084 | 0.285 / 0.084 | 0.293 / 0.078 | 0.305 / 0.070 | 3.804 / 6.512 | 0.0101 ± 0.0023 |
| | B | 0.244 / 0.069 | 0.283 / 0.085 | 0.302 / 0.091 | 0.310 / 0.090 | 0.312 / 0.080 | 0.319 / 0.077 | 2.051 / 1.202 | 0.1010 ± 0.0526 |
| | F+B (B) | 0.239 / 0.071 | 0.286 / 0.088 | 0.307 / 0.097 | 0.319 / 0.095 | 0.321 / 0.085 | 0.321 / 0.079 | 1.688 / 1.002 | 0.1006 ± 0.0519 |
| SKF+Marg. (50%) | F | 0.112 / 0.028 | 0.140 / 0.037 | 0.164 / 0.045 | 0.185 / 0.048 | 0.203 / 0.052 | 0.227 / 0.055 | 4.868 / 15.249 | 0.0197 ± 0.0046 |
| | F+B (F) | 0.126 / 0.033 | 0.154 / 0.042 | 0.165 / 0.047 | 0.174 / 0.048 | 0.179 / 0.048 | 0.185 / 0.047 | 3.311 / 3.644 | 0.0173 ± 0.0043 |
| | B | 0.147 / 0.032 | 0.158 / 0.040 | 0.165 / 0.043 | 0.166 / 0.043 | 0.158 / 0.038 | 0.160 / 0.034 | 1.329 / 0.647 | 0.2696 ± 0.1837 |
| | F+B (B) | 0.148 / 0.032 | 0.157 / 0.040 | 0.166 / 0.043 | 0.164 / 0.043 | 0.160 / 0.038 | 0.160 / 0.034 | 1.634 / 0.600 | 0.2704 ± 0.1850 |
| SKF+Marg. (75%) | F | 0.117 / 0.030 | 0.149 / 0.040 | 0.172 / 0.047 | 0.186 / 0.053 | 0.200 / 0.058 | 0.214 / 0.062 | 9.155 / 51.779 | 0.0150 ± 0.0032 |
| | F+B (F) | 0.138 / 0.037 | 0.169 / 0.047 | 0.177 / 0.051 | 0.186 / 0.052 | 0.187 / 0.049 | 0.183 / 0.046 | 3.510 / 4.491 | 0.0128 ± 0.0029 |
| | B | 0.159 / 0.035 | 0.166 / 0.042 | 0.174 / 0.045 | 0.174 / 0.044 | 0.174 / 0.040 | 0.175 / 0.037 | 1.771 / 0.809 | 0.1977 ± 0.1314 |
| | F+B (B) | 0.162 / 0.036 | 0.172 / 0.043 | 0.181 / 0.047 | 0.180 / 0.046 | 0.180 / 0.042 | 0.181 / 0.039 | 1.210 / 0.660 | 0.1991 ± 0.1340 |
| SKF+Marg. (90%) | F | 0.135 / 0.034 | 0.181 / 0.048 | 0.226 / 0.060 | 0.261 / 0.069 | 0.298 / 0.076 | 0.331 / 0.082 | 4.586 / 31.880 | 0.0124 ± 0.0027 |
| | F+B (F) | 0.151 / 0.042 | 0.187 / 0.056 | 0.203 / 0.061 | 0.211 / 0.061 | 0.219 / 0.057 | 0.219 / 0.053 | 3.488 / 5.361 | 0.0105 ± 0.0024 |
| | B | 0.174 / 0.041 | 0.181 / 0.049 | 0.195 / 0.054 | 0.196 / 0.054 | 0.198 / 0.049 | 0.194 / 0.045 | 1.457 / 0.935 | 0.1602 ± 0.1065 |
| | F+B (B) | 0.172 / 0.042 | 0.181 / 0.051 | 0.195 / 0.055 | 0.198 / 0.054 | 0.198 / 0.049 | 0.195 / 0.045 | 1.547 / 0.886 | 0.1569 ± 0.1030 |

Table 4.14: Average RPE over 5 runs of the Spencer dataset for different algorithms. RPE units are in degrees and meters, with a max total of 200 map features being leveraged.

| Algo. | Config | 40m | 80m | 120m | 160m | 200m | 240m | NEES (ori/pos) | Time (s) |
|---|---|---|---|---|---|---|---|---|---|
| SKF+Marg. (50%) | F | 0.160 / 0.039 | 0.234 / 0.056 | 0.296 / 0.072 | 0.345 / 0.083 | 0.383 / 0.092 | 0.423 / 0.102 | 4.494 / 18.620 | 0.0126 ± 0.0030 |
| | F+B (F) | 0.168 / 0.045 | 0.232 / 0.063 | 0.269 / 0.073 | 0.281 / 0.072 | 0.272 / 0.065 | 0.257 / 0.060 | 3.037 / 3.667 | 0.0144 ± 0.0033 |
| | B | 0.148 / 0.035 | 0.157 / 0.041 | 0.166 / 0.044 | 0.164 / 0.043 | 0.165 / 0.039 | 0.165 / 0.036 | 1.357 / 0.424 | 0.2231 ± 0.1568 |
| | F+B (B) | 0.150 / 0.034 | 0.160 / 0.040 | 0.168 / 0.044 | 0.168 / 0.043 | 0.167 / 0.038 | 0.166 / 0.035 | 1.759 / 0.474 | 0.2220 ± 0.1565 |
| SKF+Marg. (75%) | F | 0.150 / 0.037 | 0.222 / 0.054 | 0.283 / 0.070 | 0.332 / 0.081 | 0.370 / 0.089 | 0.398 / 0.097 | 4.848 / 26.821 | 0.0109 ± 0.0025 |
| | F+B (F) | 0.161 / 0.042 | 0.209 / 0.056 | 0.229 / 0.063 | 0.239 / 0.062 | 0.246 / 0.058 | 0.247 / 0.054 | 3.635 / 4.758 | 0.0126 ± 0.0028 |
| | B | 0.159 / 0.036 | 0.170 / 0.043 | 0.178 / 0.046 | 0.179 / 0.046 | 0.176 / 0.042 | 0.172 / 0.039 | 1.517 / 0.514 | 0.1850 ± 0.1311 |
| | F+B (B) | 0.159 / 0.035 | 0.167 / 0.042 | 0.173 / 0.045 | 0.174 / 0.045 | 0.172 / 0.041 | 0.171 / 0.038 | 2.115 / 0.635 | 0.1871 ± 0.1331 |
| SKF+Marg. (90%) | F | 0.157 / 0.041 | 0.237 / 0.060 | 0.303 / 0.077 | 0.362 / 0.092 | 0.413 / 0.106 | 0.470 / 0.118 | 12.327 / 59.804 | 0.0097 ± 0.0023 |
| | F+B (F) | 0.169 / 0.048 | 0.222 / 0.066 | 0.248 / 0.076 | 0.270 / 0.076 | 0.282 / 0.072 | 0.283 / 0.066 | 3.289 / 4.242 | 0.0113 ± 0.0026 |
| | B | 0.180 / 0.045 | 0.197 / 0.052 | 0.210 / 0.058 | 0.215 / 0.058 | 0.220 / 0.054 | 0.222 / 0.050 | 1.561 / 0.698 | 0.1578 ± 0.1065 |
| | F+B (B) | 0.181 / 0.045 | 0.198 / 0.054 | 0.212 / 0.060 | 0.220 / 0.059 | 0.219 / 0.054 | 0.221 / 0.050 | 1.852 / 0.740 | 0.1570 ± 0.1067 |

First we can see that the inclusion of map features allows for all methods to improve in accuracy over VIO. Additionally, we can see that the inclusion of more map features in the Dynamic-SKF allows for further accuracy gains, but plateaus after 300 points. The inflation baseline is also, in general, able to improve in accuracy levels, and becomes more consistent as more features are included in the map. The key difference for the inflation Marg. feature method is that the feature's uncertainty is built online, which can be inaccurate due to long-term VIO drift. This clearly causes the inflation Marg. method to perform much worse as compared to when a consistent prior covariance is provided as in Section 4.6.3.1. In fact, as the scale of the map becomes bigger, such as the multi-room dataset, the inconsistencies of the feature uncertainties become much larger and need to be addressed.

As compared to when a prior map was provided, one can see that the Hybrid use of Marg. features with the SKF tell a much different story due to their inconsistent nature. The Dynamic-SKF and Marg. features are able to obtain high levels of accuracy near the full SKF, but with an overconfident covariance matrix. This is due to the Marg. feature inconsistencies "corrupting" the SKF feature estimates and clone states during the update, which was prevented when the SKF marginal covariance was fixed. We equate the Fixed-SKF with marginal features better consistency, to be an artifact of the conservative nature of the SKF "hiding" the inconsistencies of the Marg. features.

As demonstrated, the use of Marg. features directly will provide computational gains, but will hurt estimation accuracy and consistency. The proposed mixture of Dynamic-SKF and Marg. features are able to provide significant computational gains with minimal impact on accuracy. A natural remedy to this issue is to ensure that Marg. features and their covariances are globally self-consistent, without inertial drift, nor overconfident from their marginal nature. This is exactly solved by the proposed backend optimization framework, which aims to efficiently recover an accurate, drift-free, and consistent feature map that can be leveraged to remove this source of error.

#### 4.6.3.3 Experiment 3: Consistent Backend Optimization

Next, we look to investigate the performance of the backend optimization which allows for *relinearization* and global consistency in the map through the inclusion of loop-closures. Standard VIO is run, using the configuration described in Table 4.8, without any online map building to provide relative poses to the backend optimization. The use of the marginal relative covariance from Appendix F.2 (MARG), the proposed relative C-KLAM covariance from Section 4.6.2.1 (C-KLAM), and equally weighting all edges with a fixed identity covariance (EQUAL) is reported. We first investigate the consistency of the relative pose factors provided by the frontend, after which the PnP relative poses recovered and full optimization is investigated.

Shown in Table 4.11, we can see that the relative MARG VIO factors are accurate in nature, and that the covariance recovered is as consistent as expected (the frontend provides consistent relative information). If the previously appended keyframe is not kept in the state as proposed, see Appendix F.2, the correlations between sequentially added keyframe relatives are dropped, and the NEES is extremely conservative with an average NEES of 0.25. More importantly, one can see that the recovered covariance of the keyframe poses in the global are highly inconsistent, and become more so as the frequency of relative factor creation is increased. This clearly shows that recovering the relative pose covariance directly front the frontend prohibits the backend from achieving consistency since sequential relative measurements are highly correlated, and thus ignoring this correlation is highly inconsistent. Once PnP loop-closures are included to reduce the estimation error and VIO drive, the poses become a bit more consistent but the feature uncertainties which we aim to recover and provide to the frontend become overconfident. One can also see that using EQUAL weights significantly impacts the accuracy of the keyframe poses and features, prevents the recovery of the feature covariance, and thus is not usable.

In contrast, the proposed relative C-KLAM methodology provides a consistent relative factor uncertainty and the global keyframe poses are conservative in nature.

When PnP relative factors are included, the features remain consistent and have reduced errors, confirming the benefit of a secondary optimization. It can also be seen that the proposed PnP method is able to recover a slightly overconfident pose uncertainty which we equate to not modeling the correlations between the anchored features leveraged. We also note, that while PnP has a much larger error of 0.5deg and 2cm, it is drift-free, thus is still able to provide large improvements and enforce global consistency.

Next we look at the pose trajectory accuracy and NEES of the backend when leveraging the decoupled pose recovery method (see Appendix F.3). Shown in Table 4.12, the decoupled method with C-KLAM and PnP loop-closures is able to recover an accurate pose and uncertainty, with superior performance when compared to just VIO. Both the MARG and EQUAL methods perform poorly, are inconsistent, or are unable to recover their uncertainty. Furthermore, the average computational cost of each method to optimize at each timestep is reported. As more states and factors are included the computational cost remains real-time. The proposed relative C-KLAM does have some overhead in computing the marginal information, especially seen in the lower clone rates, but this increase is minimal.

#### 4.6.3.4 Experiment 4: Frontend-Backend Feedback Analysis

The final question we look to answer is if providing the backend's consistent and optimized map to the frontend solves the inconsistencies introduced by VIO drift, and improves state estimation performance. Shown in Table 4.13, it is clear that all methods that leveraged a map in some form are able to improve their accuracy over VIO. It is also clear that both EKF and SKF frontends have nearly the same level of accuracy over all trajectory lengths. The Inf. Marg and hybrid SKF+Marg. methods which *do not* leverage an optimized map through feedback, do not retain this property and instead have trajectory errors increase as the trajectory length grows. This makes sense as the incrementally built maps in large spaces are unable to improve their map estimates and thus are bound by the accuracy during the exploration and creation of

the map. The methods that do leverage the optimized backend map, F+B, are able both able to have trajectory error which is constant over the trajectory lengths, but also have improved consistency.

The Inf. Marg. method has the worst consistency and is only partially addressed through feedback of the optimized map. The proposed hybrid SKF+Marg. also suffers from the inaccuracies of the marginal features which contain the VIO drift and while are able to have improved accuracy through leveraging SKF features, are unable to limit accuracy lost as the trajectory length increases. The proposed hybrid SKF+Marg. which have feedback is able to outperform in terms of accuracy and consistency of the Inf. Marg method even with just 40 SKF features (SKF+Marg. with 90%). While the hybrid is unable to outperform the SKF, it is able to have more than half the computational cost and still be able to remain drift-free. This clearly shows that both the proposed system architecture choices, and the hybrid estimator provide a way to trade off efficiency for accuracy while maintaining consistency.

We further look to the case where the proposed SKF+Marg. is only allowed to keep 200 features (there are over 4k points in the map in the `Spencer` dataset). Shown

## 4.7 Summary

In this thesis thrust, we have presented three novel estimators for efficient and consistent VI-SLAM. The first, termed SEVIS-3D leveraged the Schmidt-Kalman update to efficiently update when tracking a historical map of 3D points which were loop-closed through a visual keyframe-based feature matching methodology. We then proposed SEVIS-2D a novel 2D-to-2D loop-closure constraint which instead kept a historical map of keyframes from which historical feature observations could be matched to active features and included in the feature update. This loop-closure indirectly constrained the active state through the Schmidt'ed keyframe pose as compared to the more direct 2D-to-3D loop-closure to a historical 3D feature in SEVIS-3D. We then,

within the context of a prior localization, compared the different measurement constraints and estimators, such as the EKF, SKF and feature inflation methods, in terms of their efficiency, accuracy, and consistency. This brought us to the conclusion that the SKF is well suited for smaller workspaces and features that are highly correlated with the state and thus are properly tracked to ensure consistency. The 2D-to-2D measurement model, while not achieving the same level of accuracy as the 2D-to-3D, was able to be more efficient as the number of features grew making it a compelling alternative. Finally, we showed that for larger environments, inflation methods do work if we relax the need to have *guaranteed* consistent estimation and that within the context of prior map localization the performance was relativity invariant to the inflation parameters chosen.

Then the complete SEVIS was proposed which aimed to solve the larger scale persistent localization problem and address the previous methods' inability to perform relinearization of their state estimates due to their filter-based formulation after significant periods of odometry drift and loop-closure. The novel architecture first focused on the idea of dynamic Schmidt'ing which allows for map features to be updated and gain information after re-observation, enabling refinement of the map and reduction in state uncertainty and was shown to improve performance to almost the level of full covariance estimation with only a fraction of the computational cost. We then presented an efficient and consistent backend optimization framework which leveraged a relative pose graph whose PnP uncertainty was properly captured and relative odometry uncertainty was recovered via the C-KLAM [147] methodology. This backend was shown to recover an accurate and consistent 3D point map which could then be leveraged by the visual frontend to address the relinearization problem and demonstrated efficient, accurate, and consistent performance.

# Chapter 5

# EFFICIENT VISUAL-INERTIAL COOPERATIVE LOCALIZATION WITH TEMPORAL LOOP-CLOSURE CONSTRAINTS

## 5.1 Introduction

Having improved the efficiency of long-term estimation for the single robot case, we now look at how these loop-closure methodologies can be applied to the cooperative localization (CL) multi-robot case to enable long-term efficient, consistent, and accurate state estimation. Multi-robot localization inherently comes with many challenges which are absent from the single robot case: exponential increase in state size, limited communication frequency and data rates, and an inherent asynchronicity as robots can move independently through an environment which means that two robots may never be in the same location at the same time instance. In this thrust, we look to address these problems through the use of an efficient *distributed* estimator which only requires each robot to estimate its own state, and only leverage the other robot's state when fusing cross-robot constraint information. We advocate for the use of covariance intersection (CI) [87] to compensate for the unknown correlations between robots in a consistent nature, and show that while it is very conservative in its covariance estimation, it enables high quality localization performance. We present a novel method for leveraging historical poses and features within the context of a CI estimator and extend the previously created single-robot loop-closure detection and measurement methodology to this multi-robot case. The system is evaluated extensively in simulation along with on two real-world datasets clearly showing the efficiency, consistency, and accuracy of the proposed distributed VINS.

This work builds on the work by Zhu et al. [226] which only leveraged cross-robot MSCKF features within a distributed CI estimator framework and required *simultaneous* viewing of the scene to perform cross-robot updates. In contrast, we present two novel methods on how to leverage SLAM within the multi-robot case and incorporate loop-closure constraints to historical states of other robots and thus we do *not* require *simultaneous* viewing of the same location (e.g., a robot can gain information if another robot had previously explored the same location), while significantly improving the localization performance thanks to such common multi-robot measurement information. Another close work is that by Sartipi et al. [174] which introduced a distributed method for multi-user AR experiences through the use of multi-map feature constraints. Common features were detected in environmental maps received from other users and the transmitted feature position estimates were used to constrain the user's state directly. Instead of inflating measurement noise and the robot-to-robot map transform to compensate for the unknown correlations between the current user and the other user's map, we leverage CI that theoretically guarantees consistency to handle the unknown correlations and also do not require that all common features must match to each robot's environmental maps thus leveraging additional information.

## 5.2 Distributed Localization

In this section, we briefly describe the cooperative visual-inertial system that serves as the basis for the proposed distributed CI-based estimator. The state vector for the $i$'th robot contains its current IMU navigation state $\mathbf{x}_{I_i}$, sliding window of cloned IMU poses $\mathbf{x}_{C_i}$, spatial-temporal calibration parameters $\mathbf{x}_{W_i}$, along with a small temporal map (i.e., SLAM features) $\mathbf{x}_{M_i}$ at time $t_k$ (see [60, 226]).

$$\mathbf{x}_{i,k} = \begin{bmatrix} \mathbf{x}_{I_i}^\top & \mathbf{x}_{W_i}^\top & \mathbf{x}_{C_i}^\top & \mathbf{x}_{M_i}^\top \end{bmatrix}^\top \tag{5.1}$$

$$\mathbf{x}_{I_i} = \begin{bmatrix} {}^{I_{i,k}}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_{i,k}}^\top & {}^G\mathbf{v}_{I_{i,k}}^\top & \mathbf{b}_{\omega_{i,k}}^\top & \mathbf{b}_{a_{i,k}}^\top \end{bmatrix}^\top \tag{5.2}$$

$$\mathbf{x}_{W_i} = \begin{bmatrix} {}^{C_i}t_{I_i} & {}^{C_i}_{I_i}\bar{q}^\top & {}^{C_i}\mathbf{p}_{I_i}^\top & \boldsymbol{\zeta}_i^\top \end{bmatrix}^\top \tag{5.3}$$

$$\mathbf{x}_{C_i} = \begin{bmatrix} {}^{I_{i,k-1}}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_{i,k-1}}^\top & \cdots & {}^{I_{i,k-c}}_{G}\bar{q}^\top & {}^G\mathbf{p}_{I_{i,k-c}}^\top \end{bmatrix}^\top \tag{5.4}$$

$$\mathbf{x}_{M_i} = \begin{bmatrix} {}^{G}\mathbf{p}_{f1}^{\top} & \cdots & {}^{G}\mathbf{p}_{fm}^{\top} \end{bmatrix}^{\top} \tag{5.5}$$

where we have the typical clone state $\mathbf{x}_C$ which contains $c$ historical IMU poses in a sliding window and a temporal map state $\mathbf{x}_M$ has $m$ features. Each robot additionally calibrates its camera intrinsics $\boldsymbol{\zeta}_i$, camera-IMU extrinsics, and camera-IMU temporal offset ${}^{C_i}t_{I_i}$, see Section 2.2. Note that the subscript now specifies which robot the state corresponds to, e.g. the $i$'th, as compared to time, e.g. the $k$'th time at $t_k$, which is specified after the robot index. Finally, given a group of $n$ robots, we have the following combined state and covariance matrix decomposition:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_{1,k}^{\top} & \cdots & \mathbf{x}_{n,k}^{\top} \end{bmatrix}^{\top} \tag{5.6}$$

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{11_k} & \cdots & \mathbf{P}_{N1_k} \\ \vdots & \ddots & \vdots \\ \mathbf{P}_{1N_k} & \cdots & \mathbf{P}_{NN_k} \end{bmatrix} \tag{5.7}$$

Here we note that in the centralized formulation, this is the state that we jointly estimate along with the cross-covariance terms, while in the distributed case each robot only estimates a sub-set of the total state and correlations between robots are dropped (e.g., robot $i$ only tracks $\mathbf{x}_{i,k}$ and $\mathbf{P}_{ii_k}$).

### 5.2.1 Distributed Inertial Propagation

Similar to the single robot case presented in Section 2.2.1, we can propagate each robot's inertial state forward *independently* using their own internal IMU information. This can be seen when we inspect the linearized error state propagation model for our states which do not have any correlations with each other:

$$\mathbf{P}_{k|k-1} = \boldsymbol{\Phi}_{k-1}\mathbf{P}_{k-1|k-1}\boldsymbol{\Phi}_{k-1}^{\top} + \mathbf{Q}_{k-1} \tag{5.8}$$

$$\boldsymbol{\Phi}_{k-1} = \mathbf{Diag}\left(\boldsymbol{\Phi}_{1,k-1}, \ldots, \boldsymbol{\Phi}_{N,k-1}\right) \tag{5.9}$$

$$\mathbf{Q}_{k-1} = \mathbf{Diag}\left(\mathbf{Q}_{1,k-1}, \ldots, \mathbf{Q}_{N,k-1}\right) \tag{5.10}$$

where $\boldsymbol{\Phi}_{i,k}$ and $\mathbf{Q}_{i,k}$ are respectively the system Jacobian and discrete noise covariance for the $i$'th robot (Section 2.2.1 and [139]), and $\mathbf{Diag}(\cdots)$ creates a block diagonal

matrix from the specified values. It can be seen that in the distributed case, all states can be propagated independently since cross-covariance is not tracked. Thus robots can continue to propagate independently without having to perform communication as in the case of the centralized estimator, leading to computational efficiency and reductions in communication.

### 5.3 Distributed Visual-Inertial Cooperative Localization

As it is known that the standard EKF in the worst case has cubic computation complexity due to its covariance update, a naive implementation of the multi-robot visual-inertial CL can become prohibitively expensive as the number of robots grows in size. Note also that due to communication constraints, the robots might not be able to communicate with all the other robots or a common fusion center. To address these issues, the key idea of our CL approach is to leverage CI [87] to reduce the estimation cost, by only updating the state and error covariance of the current robot (i.e., robot $i$ only updates $\mathbf{x}_{i,k}$ and $\mathbf{P}_{ii_k}$) while ensuring consistency.

In particular, each robot independently propagates its own state and updates with measurements that are only a function of its own state. When updating with measurements of features observed from multiple robots, CI is employed to consistently handle the unknown and untracked cross-covariance terms between the involved robots. This means that robots need to communicate their state and covariance, along with visual feature information to the other robots. Each robot tracks a set of visual features using KLT optical flow [121], and communicates its latest tracks and extracted ORB descriptors [171] to the other robots in communication range. A robot then performs descriptor-based feature matching and loop-closure detection to find correspondences between its most recent features and other robots' feature tracks. After tracking and matching, feature tracks are categorized as follows:

(A) VIO features which have only been tracked for a short period of time.

(B) Temporal SLAM features which have been tracked beyond the current sliding window.

(C) Common VIO features which have been matched to features in another robot and tracked for only a short period of time.

(D) Common SLAM features which have been matched to features in another robot. Note that this feature might be either a VIO or SLAM feature in the other robot.

In the following, we present in detail how we update our state with these different feature variants. Note that for the centralized case independent features update the full state and covariance since cross-covariances are tracked, while in the distributed case only the $i$'th robot state and covariance is updated thus allowing for computational savings.

### 5.3.1 Independent VIO Feature: MSCKF Update

For VIO features that have lost active track in the current window, we perform MSCKF update [139]. In particular, we first triangulate these features for computing the feature Jacobians $\mathbf{H}_{f_k}$, and then project $\mathbf{r}_{f_k}$ (see Eq. (2.52)) onto the left nullspace of $\mathbf{H}_{f_k}$ (i.e., $\mathbf{Q}_2^\top \mathbf{H}_{f_k} = \mathbf{0}$) to yield the measurement noise independent of state:

$$\mathbf{Q}_2^\top \mathbf{r}_{f_k} = \mathbf{Q}_2^\top \mathbf{H}_{x_{i,k}} \tilde{\mathbf{x}}_{i,k} + \mathbf{Q}_2^\top \mathbf{H}_{f_k}{}^G\tilde{\mathbf{p}}_f + \mathbf{Q}_2^\top \mathbf{n}_{f_k} \tag{5.11}$$

$$\Rightarrow \ \mathbf{r}'_{f_k} = \mathbf{H}'_x \tilde{\mathbf{x}}_k + \mathbf{n}'_{f_k} \tag{5.12}$$

where $\mathbf{H}_{x_{i,k}}$ is the stacked measurement Jacobians with respect to the navigation states in the current robot's window.

### 5.3.2 Independent SLAM Feature: FEJ-EKF Update

SLAM features which a robot is able to reliably track longer than its sliding window in length, will be initialized into the SLAM map state vector $\mathbf{x}_{M_i}$. These features are directly updated using the linearized system (2.52) and will remain in the state until they have lost track. To improve consistency, we employ First Estimate

Jacobians (FEJ) [83, 84] ensuring Jacobians are evaluated at the same linearization points to prevent spurious information gain.

### 5.3.3 Common VIO Feature: CI-EKF Update

Consider we find a feature which has been seen from multiple robots and want to use this information to update the state. In the centralized case, we would directly update our state with all available measurements (2.52) through the standard EKF since we track the cross-covariance (e.g., $\mathbf{P}_{iN_k}$). In the distributed case, a robot only tracks its own state and autocovariance to ensure computational efficiency and scalability with respect to the robot team size. This presents two key challenges: (i) how to efficiently and consistently fuse multiple robots' autocovariances, and (ii) how to find the data association between different features, which motivates us to leverage CI to fuse estimates and covariances transmitted from other robots.

#### 5.3.3.1 CI-EKF Update

Consider the $i$'th robot has a measurement which is a function of $L$ other robot states. The linearized measurement model can be computed as:

$$\mathbf{r}_{f_k} = \mathbf{H}_{x_{i,k}}\widetilde{\mathbf{x}}_{i,k} + \mathbf{H}_{x_{1..L,k}}\widetilde{\mathbf{x}}_{1..L,k} + \mathbf{H}_{f_k}{}^{G}\widetilde{\mathbf{p}}_{f} + \mathbf{n}_{f_k} \tag{5.13}$$

where $\mathbf{H}_{x_{i,k}}$ is the Jacobian in respect to the $i$'th robot state using the $k$'th estimates, and $\mathbf{H}_{x_{1..L,k}}$ is the stacked Jacobian with respect to all other robots the measurement is a function of. To guarantee consistency when updating with this measurement, we adopt the CI-EKF update [87] to construct a prior covariance such that:

$$\mathbf{Diag}\left(\frac{1}{\omega_i}\mathbf{P}_{ii_k}, \frac{1}{\omega_1}\mathbf{P}_{11_k}, \cdots, \frac{1}{\omega_L}\mathbf{P}_{LL_k}\right) \geq \mathbf{P}_k \tag{5.14}$$

where the left side is the CI covariance with zero off-diagonal elements and the right-hand side is the unknown true covariance of the state with cross-covariances (see Eq. (5.7)). The weights $\omega_l > 0$ and $\sum_l \omega_l = 1$, for $l \in \{i, 1..L\}$, can be found optimally [87].

Substituting (5.14) into the standard EKF equations and only selecting the portion that updates the current robot's state (say robot $i$) yields:

$$\delta\mathbf{x}_{i,k} = \frac{1}{\omega_i}\mathbf{P}_{ii,k|k-1}\mathbf{H}_{x_{i,k}}^\top\mathbf{S}_k^{-1}\mathbf{r}'_{f_k} \tag{5.15}$$

$$\mathbf{P}_{ii,k|k} = \frac{1}{\omega_i}\mathbf{P}_{ii,k|k-1} - \frac{1}{\omega_i^2}\mathbf{P}_{ii,k|k-1}\mathbf{H}_{x_{i,k}}^\top\mathbf{S}_k^{-1}\mathbf{H}_{x_{i,k}}\mathbf{P}_{ii,k|k-1} \tag{5.16}$$

$$\mathbf{S}_k = \sum_{o\in\{i,1..L\}}\frac{1}{\omega_o}\mathbf{H}_{x_{o,k}}\mathbf{P}_{oo,k|k-1}\mathbf{H}_{x_{o,k}}^\top + \mathbf{R}_{f_k} \tag{5.17}$$

where $\delta\mathbf{x}_{i,k}$ is the correction to the state estimate $\hat{\mathbf{x}}_{i,k}$.

#### 5.3.3.2 Common VIO Feature: Efficient Nullspace Projection

To process common features which are short in length, we leverage the similar logic as in Sec. 5.3.1. For example, we have multiple measurements from two different robots and wish to update our state:

$$\begin{bmatrix}\mathbf{r}_{f_{i,k}}\\ \mathbf{r}_{f_{2,k}}\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{x_{i,k}} & \mathbf{0} & \mathbf{H}_{f_{i,k}}\\ \mathbf{0} & \mathbf{H}_{x_{2,k}} & \mathbf{H}_{f_{2,k}}\end{bmatrix}\begin{bmatrix}\widetilde{\mathbf{x}}_{I_i}\\ \widetilde{\mathbf{x}}_{I_2}\\ {}^G\widetilde{\mathbf{p}}_f\end{bmatrix} + \begin{bmatrix}\mathbf{n}_{f_{i,k}}\\ \mathbf{n}_{f_{2,k}}\end{bmatrix} \tag{5.18}$$

We can then project both equations onto their left range and nullspace (e.g., $\mathbf{H}_{f_{i,k}} = [\mathbf{Q}_{i,1}\ \mathbf{Q}_{i,2}][\mathbf{U}_i\ \mathbf{0}]^\top$):

$$\begin{bmatrix}\mathbf{r}^1_{f_{i,k}}\\ \mathbf{r}^2_{f_{i,k}}\\ \mathbf{r}^1_{f_{2,k}}\\ \mathbf{r}^2_{f_{2,k}}\end{bmatrix} = \begin{bmatrix}\mathbf{Q}_{i,1}^\top\mathbf{H}_{x_{i,k}} & \mathbf{0} & \mathbf{U}_i\\ \mathbf{Q}_{i,2}^\top\mathbf{H}_{x_{i,k}} & \mathbf{0} & \mathbf{0}\\ \mathbf{0} & \mathbf{Q}_{2,1}^\top\mathbf{H}_{x_{2,k}} & \mathbf{U}_2\\ \mathbf{0} & \mathbf{Q}_{2,2}^\top\mathbf{H}_{x_{2,k}} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\widetilde{\mathbf{x}}_{I_i}\\ \widetilde{\mathbf{x}}_{I_2}\\ {}^G\widetilde{\mathbf{p}}_f\end{bmatrix} + \begin{bmatrix}\mathbf{n}^1_{f_{i,k}}\\ \mathbf{n}^2_{f_{i,k}}\\ \mathbf{n}^1_{f_{2,k}}\\ \mathbf{n}^2_{f_{2,k}}\end{bmatrix}$$

where we have defined that $\mathbf{r}^1_{f_{i,k}} = \mathbf{Q}_{i,1}^\top\mathbf{r}_{f_{i,k}}$ and $\mathbf{n}^1_{f_{i,k}} = \mathbf{Q}_{i,1}^\top\mathbf{n}_{f_{i,k}}$. Note that the last row is no longer dependent on the current robot's state, $\mathbf{x}_{I_i}$, and thus, this can be discarded since it will not update the state or covariance due to the lack of tracked cross-covariances. This directly reduces the number of measurements involved during

the update and makes the computation of $\mathbf{S}_k^{-1}$ substantially cheaper (see Eq. (5.17)). We then have the following linear systems:

$$\begin{bmatrix} \mathbf{r}^1_{f_{i,k}} \\ \mathbf{r}^1_{f_{2,k}} \end{bmatrix} = \begin{bmatrix} \mathbf{Q}^\top_{i,1}\mathbf{H}_{x_{i,k}} & \mathbf{0} & \mathbf{U}_i \\ \mathbf{0} & \mathbf{Q}^\top_{2,1}\mathbf{H}_{x_{2,k}} & \mathbf{U}_2 \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{x}}_{I_i} \\ \widetilde{\mathbf{x}}_{I_2} \\ {}^G\widetilde{\mathbf{p}}_f \end{bmatrix} + \begin{bmatrix} \mathbf{n}^1_{f_{i,k}} \\ \mathbf{n}^1_{f_{2,k}} \end{bmatrix}$$

$$\mathbf{r}^2_{f_{i,k}} = \mathbf{Q}^\top_{i,2}\mathbf{H}_{x_{i,k}}\widetilde{\mathbf{x}}_{I_i} + \mathbf{n}^2_{f_{i,k}} \tag{5.19}$$

A second nullspace projection onto the left nullspace of $\mathbf{H}_f = [\mathbf{U}_i \ \mathbf{U}_2]^\top$ is performed to create a linear system which is only a function of the $\mathbf{x}_{I_i}$ and $\mathbf{x}_{I_2}$ states. The CI-EKF update (see Eq. (5.15) and (5.16)) is then used to update the state $\mathbf{x}_{I_i}$. The second equation (see Eq. (5.19)) can update the current robot state without CI through the standard EKF equations since it is only a function of the current robot state. This update contains the same information as in the case that we performed a "large" nullspace projection using the full feature Jacobians in (5.18), but results in a much smaller measurement size since we can drop measurement residuals which are not a function of the $i$'th robot's state.

### 5.3.4 Common SLAM Feature: CI-EKF Update

There are two different cases for temporal SLAM features: (i) a SLAM feature in the current robot state matches to a feature that is not a SLAM feature in another robot, and (ii) a SLAM feature matches to another robot's SLAM feature. For example as in Figure 5.1, in the first case we collect the measurements from the other robot ($\mathbf{z}_{1..N}$) and directly apply (5.13) and update both the current robot's poses and its estimate of the SLAM feature. In the second case, we can follow this same logic (i.e., grab the measurements from the other robot and update the current robot's estimate) or we can leverage the knowledge that the 3D position of these two features should be equal. This SLAM feature constraint model is similar to the one introduced in [69] for cooperative mapping. Consider we have the following two robots:

$$\mathbf{x}_{i,k} = \begin{bmatrix} \mathbf{x}^\top_{I_i} & \mathbf{x}^\top_{W_i} & \mathbf{x}^\top_{C_i} & {}^G\mathbf{p}^\top_{fa} \end{bmatrix}^\top \tag{5.20}$$

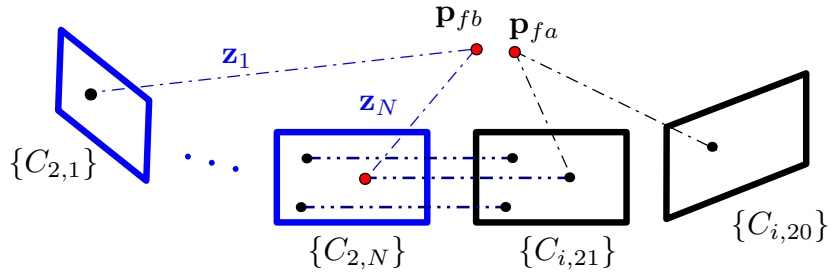



Figure 5.1: Illustration of the keyframe-aided 2D-to-2D matching for data association. Assuming robot $i$'s 21st frame $\{C_{i,21}\}$ matches to the 2nd robot's $N$'th frame $\{C_{2,N}\}$. We are able to find all feature correspondences between the features the robot's observer, namely $\mathbf{z}_{1..N}$.

$$\mathbf{x}_{2,k} = \begin{bmatrix} \mathbf{x}_{I_2}^\top & \mathbf{x}_{W_2}^\top & \mathbf{x}_{C_2}^\top & {}^G\mathbf{p}_{fb}^\top \end{bmatrix}^\top \tag{5.21}$$

If we have matched feature ${}^G\mathbf{p}_{fa}$ in the current $i$'th robot to the ${}^G\mathbf{p}_{fb}$ in the other robot, then we can construct the following feature constraint (see Figure 5.1):

$${}^G\mathbf{p}_{fa} - {}^G\mathbf{p}_{fb} = \mathbf{0} \Rightarrow \ \mathbf{r}_c\left(\mathbf{x}_{i,k}, \mathbf{x}_{2,k}\right) = \mathbf{0} \tag{5.22}$$

which can be linearized to yield:

$$\mathbf{r}_c\left(\hat{\mathbf{x}}_{i,k}, \hat{\mathbf{x}}_{2,k}\right) + \mathbf{H}_{fa}{}^G\tilde{\mathbf{p}}_{fa} + \mathbf{H}_{fb}{}^G\tilde{\mathbf{p}}_{fb} \approx \mathbf{0} \tag{5.23}$$

$$\Rightarrow \mathbf{0} - \mathbf{r}_c\left(\hat{\mathbf{x}}_{i,k}, \hat{\mathbf{x}}_{2,k}\right) \approx \mathbf{H}_{fa}{}^G\tilde{\mathbf{p}}_{fa} + \mathbf{H}_{fb}{}^G\tilde{\mathbf{p}}_{fb} \tag{5.24}$$

This linearized system can then update the $i$'th robot state estimate using the CI-EKF update (see Eq. (5.15) and (5.16)). Note that this is a very efficient update, as it is only a function of the two estimated feature positions.

### 5.3.5 Historical Features: CI-EKF Update

We now explain how to leverage loop-closure constraints to previous robot states. First, to find the feature correspondences between robots, as previously presented in Chapter 4, each robot creates DBoW2 [53] databases for all other robots. When a robot receives feature tracks and descriptors from other robots they are appended to their corresponding DBoW2 database. The current image can then be queried against the other robots' databases to see if any other robots are or have been at the current

location. If a loop closure is detected and verified using a fundamental matrix geometric check, then we assume that we have detected that another robot has been at our current location. After matching descriptors, we know the correspondences between a feature in the current robot, and that of the features in the other robot (see Figure 5.1). We can then grab the history of measurements and formulate a common feature update.

To incorporate these measurements from historical states, each robot records the measurement and previous states received from the other agents.[1] Outside of the most recent sliding window, these historical states can provide loop-closure information if we are able to generate measurement constraints to them. Specifically, we store the following historical states and covariances in addition to their most recent states published:

$$\mathbf{x}_i = \{\mathbf{x}_{i,0}, \cdots, \mathbf{x}_{i,k-1}\} \tag{5.25}$$

$$\mathbf{P}_i = \left\{\mathbf{P}_{ii_0}, \cdots, \mathbf{P}_{ii_{k-1}}\right\} \tag{5.26}$$

Since each one of these historical states contains a sliding window of poses and SLAM features, we only store non-overlapping sliding windows. To accelerate lookup we only store historical descriptor information at a fixed rate (normally 1Hz) since recent frames in the same sliding window contain redundant loop-closure information. A more ideal heuristic could be leveraged here to increase match rates. Once loop-closure is detected, we know old historical feature correspondences which we can then use to retrieve measurements and update our current robot state. This update is identical to the CI-EKF update as in Sec. 5.3.3–5.3.4, which only needs to involve the historical windows that contain the historical measurements, and thus is efficient since historical states are not updated.

Table 5.1: Simulation parameters and prior standard deviations that perturbations of measurements and initial states were drawn from.

| Parameter | Value | Parameter | Value |
|---|---|---|---|
| Gyro. White Noise | 1.6968e-04 | Gyro. Rand. Walk | 1.9393e-05 |
| Accel. White Noise | 2.0000e-3 | Accel. Rand. Walk | 3.0000e-3 |
| Pixel Proj. (px) | 1 | Robot Num. | 3 |
| IMU Freq. (Hz) | 400 | Cam Freq. (Hz) | 10 |
| AR Avg. Feats | 25 | AR Num. SLAM | 3 |
| ETH Avg. Feats | 50 | ETH Num. SLAM | 5 |
| Num. Clones | 11 | Feat. Rep. | GLOBAL |

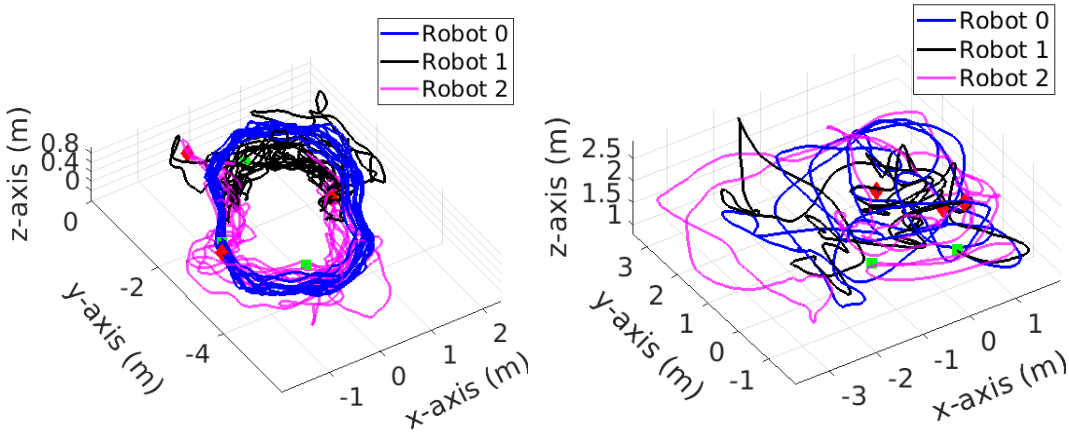



Figure 5.2: Simulated trajectories, axes are in units of meters. General hand-held AR dataset (left) are 147, 93, and 100 meters long, while ETH EuRoC MAV Vicon room datasets (right) are 70, 58, and 59 meters long for each robot. Green square denotes the start and red diamond denotes the end.

## 5.4 Numerical Results

To validate the proposed method, we have simulated two realistic scenarios both with three robots (see Figure 5.2). The first is a hand-held mobile AR dataset which has a series of users look and move around a central table, while the second is a series of trajectories from the ETH EuRoC MAV dataset [14]. We employ the OpenVINS simulator, see Section 2.3.7, to generate realistic visual-bearing and inertial measurements from these supplied trajectories. On average each robot is able to find common features on, respectively, 79.0% and 83.5% (43.7% and 62.7%) of the frames without or with loop-closure in AR datasets (ETH dataset). This clearly shows the advantage of historical loop-closure on datasets which have limited temporal view overlaps between robots. Simulation parameters used are documented in Table 5.1. We fix the weight of other robots' covariance in the CI-EKF update as $\omega_o = 0.001$. For the constraint measurement update presented in Sec. 5.3.4, we use the value $\omega_o = 0.005$ and a synthetic measurement noise of 2cm. Note that while these weights can be found by minimizing the trace or determinant of $\mathbf{P}_{ii,k|k}$ [87], we have empirically found that using fixed weights still ensures consistent performance. For fair and thorough comparison, we define the following variations of the centralized and proposed distributed CL estimators:

**indp** – No common features are found between robots and all measurements are processed as independent features which only relate to the current robot.

**indp-slam** – Same as *indp*, but temporal SLAM features are included in each robot to show the relative improvement.

**ce-cmsckf** – The centralized estimator uses the common VIO features over the sliding window.

[1] In the future we plan to investigate the latency introduced due to communication constraints, but historical matching ensures that the robot will leverage *all* available information at the current time including delayed information recently communicated.

**ce-cmsckf-cslam** – The centralized estimator uses the common VIO and SLAM features over the sliding window.

**dc-cmsckf** [226] – The distributed estimator using the common VIO features over the sliding window.

**dc-cmsckf-cslam** – The distributed estimator uses the common VIO and SLAM features over the sliding window without enforcing the same feature constraint. For example, even if a common SLAM feature is a SLAM feature in another robot's state, we grab the measurements from the other robot and update as the first case in Sec. 5.3.4.

**dc-full-window** – The distributed estimator uses the common VIO and SLAM features over the sliding window while enforcing the same feature constraint.

**dc-full-history** – The distributed estimator uses both the common VIO and SLAM features over the sliding window and from historical matching.

Note that the observed independent VIO features and SLAM features are used in all these estimators. To ensure a fair comparison, the same parameters reported in Table 5.1 are used for all algorithms and for all robots.

### 5.4.1 Accuracy and Consistency Evaluation

We performed 20 Monte-Carlo simulations on each dataset. The average Absolute Trajectory Error (ATE) [221] can be found in Table 5.2 and 5.3. It is clear from the top two rows that the additional SLAM features improve `indp`. In the cooperative case, when using the common VIO features, both `ce-msckf` and `dc-msckf` outperform the `indp-slam`, and when including common SLAM features, the accuracy is further improved. It is worth noting that the efficient `dc-full-window` with feature constraint has close accuracy to its counterpart `dc-cmsckf-cslam`. Moreover, when including the historical common features, the distributed estimator becomes

Table 5.2: ATE on simulated AR datasets in degrees / meters for each algorithm variation. Green denotes the best, while blue is second best.

| Algorithm | Robot 0 | Robot 1 | Robot 2 | Average |
|---|---|---|---|---|
| indp | 1.957 / 0.072 | 0.811 / 0.041 | 0.742 / 0.039 | 1.170 / 0.051 |
| indp-slam | 1.396 / 0.046 | 0.602 / 0.029 | 0.557 / 0.022 | 0.852 / 0.032 |
| ce-cmsckf | 0.364 / 0.017 | 0.323 / 0.015 | 0.355 / 0.015 | 0.347 / 0.016 |
| ce-cmsckf-cslam | **0.232** / **0.011** | **0.228** / **0.011** | **0.220** / **0.010** | **0.227** / **0.011** |
| dc-cmsckf | 0.759 / 0.029 | 0.540 / 0.025 | 0.553 / 0.020 | 0.617 / 0.025 |
| dc-cmsckf-cslam | 0.643 / 0.025 | 0.496 / 0.022 | 0.478 / 0.017 | 0.539 / 0.022 |
| dc-full-window | 0.644 / 0.024 | 0.547 / 0.022 | 0.480 / 0.017 | 0.557 / 0.021 |
| dc-full-history | **0.356** / **0.017** | **0.299** / **0.014** | **0.319** / **0.013** | **0.325** / **0.014** |

Table 5.3: ATE on simulated ETH datasets in degrees / meters for each algorithm variation. Green denotes the best, while blue is second best.

| Algorithm | Robot 0 | Robot 1 | Robot 2 | Average |
|---|---|---|---|---|
| indp | 0.569 / 0.088 | 0.578 / 0.092 | 0.560 / 0.093 | 0.569 / 0.091 |
| indp-slam | 0.371 / 0.070 | 0.406 / 0.069 | 0.444 / 0.075 | 0.407 / 0.071 |
| ce-cmsckf | 0.221 / 0.052 | 0.221 / 0.049 | 0.221 / 0.051 | 0.221 / 0.050 |
| ce-cmsckf-cslam | **0.151** / **0.042** | **0.143** / **0.038** | **0.144** / **0.040** | **0.146** / **0.040** |
| dc-cmsckf | 0.329 / 0.064 | 0.342 / 0.061 | 0.319 / 0.062 | 0.330 / 0.062 |
| dc-cmsckf-cslam | 0.298 / 0.054 | 0.325 / 0.050 | 0.290 / 0.052 | 0.304 / 0.052 |
| dc-full-window | 0.285 / 0.052 | 0.287 / 0.047 | 0.268 / 0.047 | 0.280 / 0.049 |
| dc-full-history | **0.211** / **0.029** | **0.207** / **0.031** | **0.218** / **0.030** | **0.212** / **0.030** |

more accurate as expected. Interestingly, with only the common features over the sliding window, the `ce-cmsckf-cslam` can achieve the best performance on the AR dataset even without loop-closure. This is likely due to the fact that over the whole dataset all robots look in the same general location thus negating any benefit of loop-closure detection. As shown in Table 5.3 and in the following real-world experiments, when robots do not have many overlapping views, the historical information plays an important role.

We additionally show the average Root Mean Square Error (RMSE) [221] and Normalized Estimation Error Squared (NEES) [4] of the distributed algorithms for

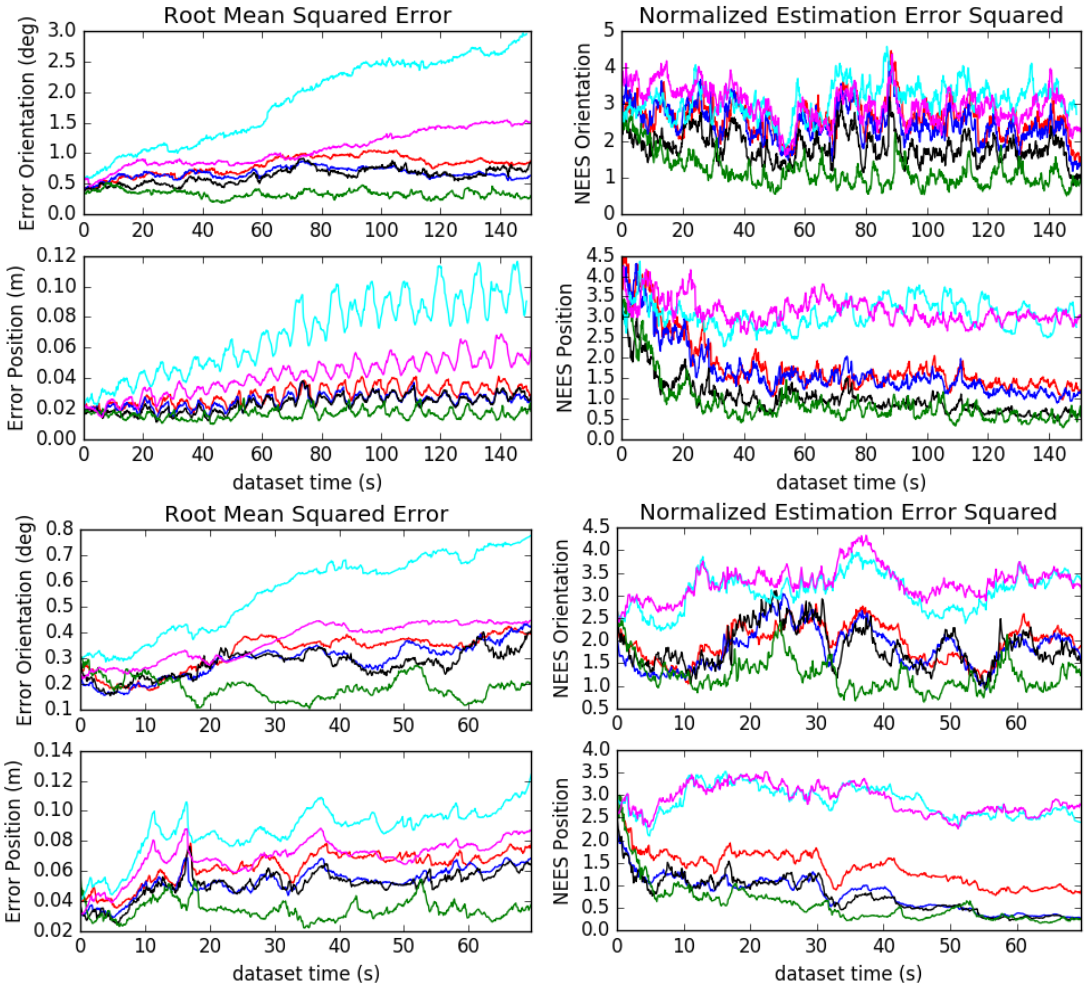



Figure 5.3: Robot 0's average RMSE (left) and NEES (right) results in the simulated AR (top) and ETH datasets (bottom). Cyan represents indp, magenta represents indp-slam, red represents dc-msckf, blue represents dc-cmsckf-cslam, green represents dc-full-window and green represents dc-full-history. Please refer to the color figure.

Robot 0 in Figure 5.3. The results for the other two robots are similar and are omitted here for space. The `indp` has the largest drift that can be reduced as shown by `indp-slam` and leveraging common features. The `dc-cmsckf-cslam` and `dc-full-window` have almost the same performance while the `dc-full-history` achieves the best accuracy. It is clear that all the distributed algorithms are conservative in nature (NEES is smaller than three) and have smaller NEES than the centralized ones.

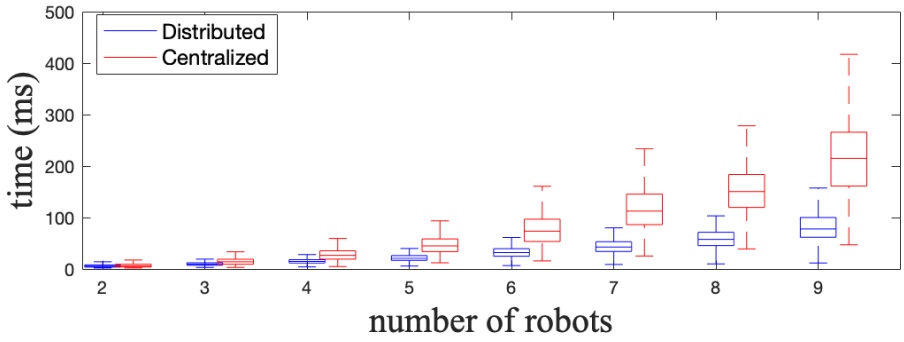



Figure 5.4: Sequential propagation and update time (ms). Note that while decentralized can update in parallel, here we report its sequential timings.

Table 5.4: Timing for AR dataset. Millisecond mean and deviation.

| Algorithm | Proposed | Combined |
|---|---|---|
| MSCKF update (window) | 1.20 ± 0.94 | 2.88 ± 3.90 |
| MSCKF update (hist) | 4.11 ± 5.52 | 22.75 ± 159.65 |
| **Algorithm** | **Constraint** | **No Constraint** |
| SLAM update (window) | 0.10 ± 0.03 | 0.15 ± 0.16 |
| SLAM update (hist) | 0.17 ± 0.06 | 27.11 ± 160.95 |

### 5.4.2 Timing Analysis

#### 5.4.2.1 Multiple Robots

We now investigate the computational efficiency of the proposed work in comparison to the centralized estimator using only common features over the sliding window. We compare the timing results of `dc-full-window` and `ce-cmsckf-cslam` while processing the same amount of measurements. We first investigate the performance as more robots are added to show the efficiency gains from the distributed formulation. The results in Figure 5.4 show that as more robots are added, the centralized estimator quickly becomes computationally expensive while the distributed one is able to remain efficient since each robot only needs to propagate and update its own state and auto-covariance. Additionally, if one robot does not find common features in a given frame, the robot can update the estimator independently in the distributed case. On the contrary, the centralized estimator needs to collect all data, propagate, and update

the whole state even if there are no common features. The distributed algorithm does have a slight increase in cost, which is due to the increase of common measurements from the additional robots.

#### 5.4.2.2 Common VIO Features

We next investigate the efficiency of the common VIO feature nullspace projection and subsequent CI-EKF update introduced in Sec. 5.3.3.2. We report the update time for `dc-full-window` (window) and `dc-full-history` (history) without common SLAM features. The results presented in Table 5.4 show that if we use the proposed method to first perform nullspace projection and separate each robot's systems into two systems (Proposed) we are able to outperform the naive way of performing nullspace projection on a "stacked" Jacobian containing all robot feature Jacobians (Combined). It is clear that in both algorithms, the proposed method is able to have less computational cost, especially in the historical case due to the proposed system reducing the number of measurements in the update. We also note that there is a high level of variance in the historical case due to loop-closure introducing large amounts of measurements in short intervals.

#### 5.4.2.3 SLAM Constraint Update

Now we investigate the efficiency of the common SLAM feature update introduced in Sec. 5.3.4. Only common SLAM features that can be matched to another robot's SLAM feature are used to ensure that both variants have the same number of measurements in the update. When we match features in the current window, the constraint update (Constraint) is slight more efficient than the naive way of grabbing all the measurements from the other robots (No Constraint) since all robots only have the most recent measurements (in most cases just one). During historical SLAM matching, by definition SLAM features are long feature tracks, and thus many measurements and clones states are associated with a historical SLAM feature. This means that after loop-closure in the naive case (No Constraint) we will process all measurements ever

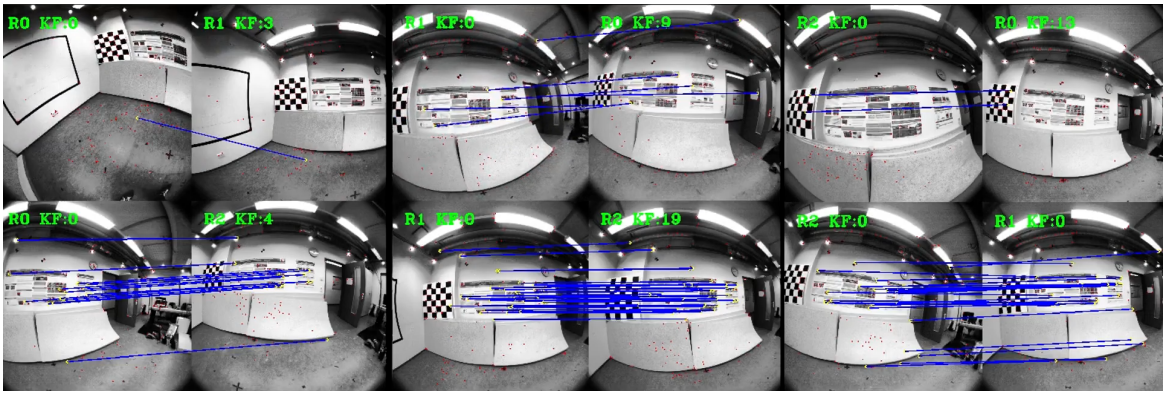



Figure 5.5: Example feature matching for each robot to the other two robot keyframes descriptors. Images are just shown for visualization, only descriptors and point coordinates need to be transmitted.

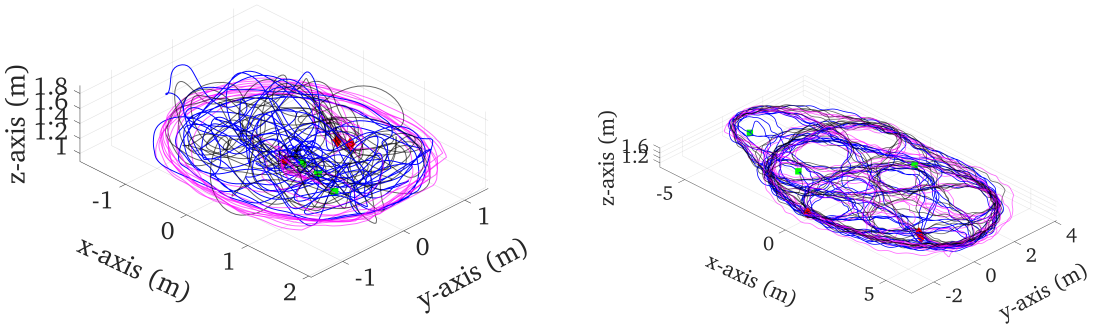



Figure 5.6: TUM-VI groundtruth (left) and Vicon room groundtruth trajectories (right) TUM-VI trajectories are 146, 131, and 134 meters long, while the Vicon room datasets are 507, 509, and 501 meters long.

recorded for a SLAM feature which can easily reach many sliding windows in length. If instead we use the constraint update, only the two feature positions are involved, thus the update is extremely efficient in nature (bottom Table 5.4).

## 5.5 Real-World Results

We have also evaluated the proposed distributed CL estimators on the TUM-VI dataset [179] and a hand collected 10 minute long Vicon room dataset (see Figure 5.6).[2] Both datasets provide monochrome stereo images at 20Hz and IMU readings at 200Hz. We only leverage the left camera and initialize all robots based on the

[2] A video demo https://youtu.be/boHBcVoMKk8

Table 5.5: Relative pose error (RPE) on TUM-VI datasets in degrees / meters averaged over all robots for the dataset.

| Algorithm | 40m | 60m | 80m | 100m | 120m |
|---|---|---|---|---|---|
| indp-slam | 1.818 / 0.093 | 2.833 / 0.126 | 2.604 / 0.154 | 2.774 / 0.185 | 2.716 / 0.215 |
| ce-cmsckf | **1.358** / 0.071 | **1.321** / 0.091 | 1.357 / 0.108 | 0.843 / 0.128 | 0.932 / 0.140 |
| ce-cmsckf-cslam | 1.758 / **0.069** | 1.350 /**0.079** | **1.027** / **0.100** | **0.718** / 0.119 | 0.938 / **0.130** |
| dc-cmsckf | 1.662 / 0.075 | 2.005 / 0.104 | 1.605 / 0.129 | 1.142 / 0.141 | 1.531 / 0.170 |
| dc-cmsckf-cslam | 1.800 / 0.080 | 2.642 / 0.093 | 2.233 / 0.106 | 1.544 / **0.114** | 0.934 / 0.157 |
| dc-full-window | 1.768 / 0.075 | 2.218 / 0.091 | 1.788 / 0.109 | 1.257 / 0.123 | **0.854** / 0.159 |
| dc-full-history | **1.213** / **0.067** | **1.232** / **0.061** | **1.029** / **0.065** | **1.004** / **0.068** | **0.784** / **0.072** |

Table 5.6: Relative pose error (RPE) on Vicon room dataset in degrees / meters averaged over all robots.

| Algorithm | 80m | 100m | 200m | 300m | 420m |
|---|---|---|---|---|---|
| indp-slam | **2.022** / **0.276** | 2.416 / 0.334 | 3.872 / 0.613 | 5.222 / 0.870 | 8.045 / 1.189 |
| ce-cmsckf-cslam | 2.180 / 0.288 | 2.603 / 0.333 | **2.771** / **0.548** | **3.050** / **0.770** | **3.557** / **1.044** |
| dc-full-window | 2.197 / 0.281 | **2.340** / **0.332** | 3.322 / 0.580 | 3.670 / 0.804 | 5.977 / 1.102 |
| dc-full-history | **1.271** / **0.145** | **1.307** / **0.151** | **1.346** / **0.158** | **1.267** / **0.157** | **1.343** / **0.160** |

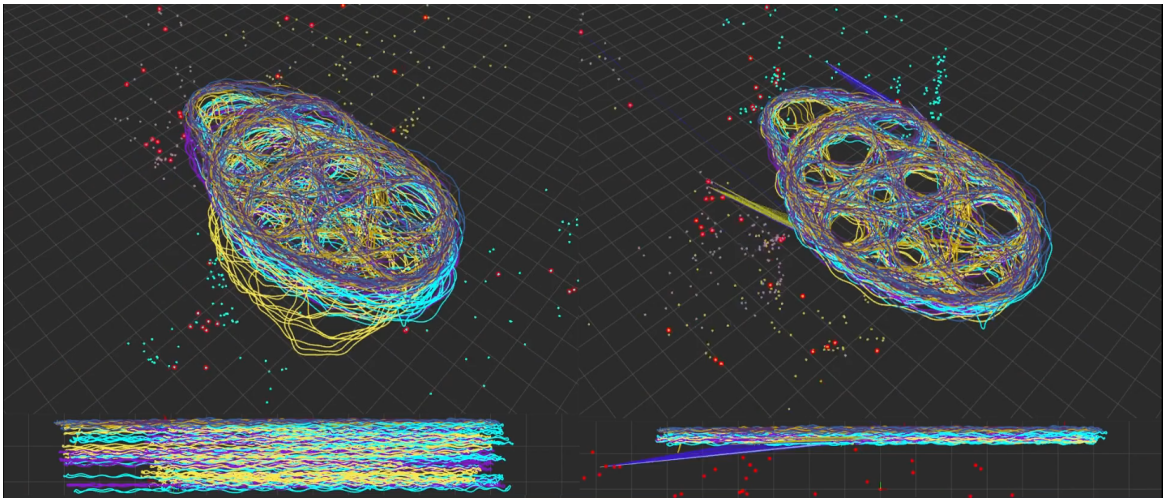


Figure 5.7: Example trajectory of `indp-slam` (left) and proposed `dc-full-history` (right). The benefit of leveraging cross-robot loop-closure constraints can be clearly seen by the minimal drift of the proposed. Robot 1, 2, and 3 are shown in different colors along with their feature maps.

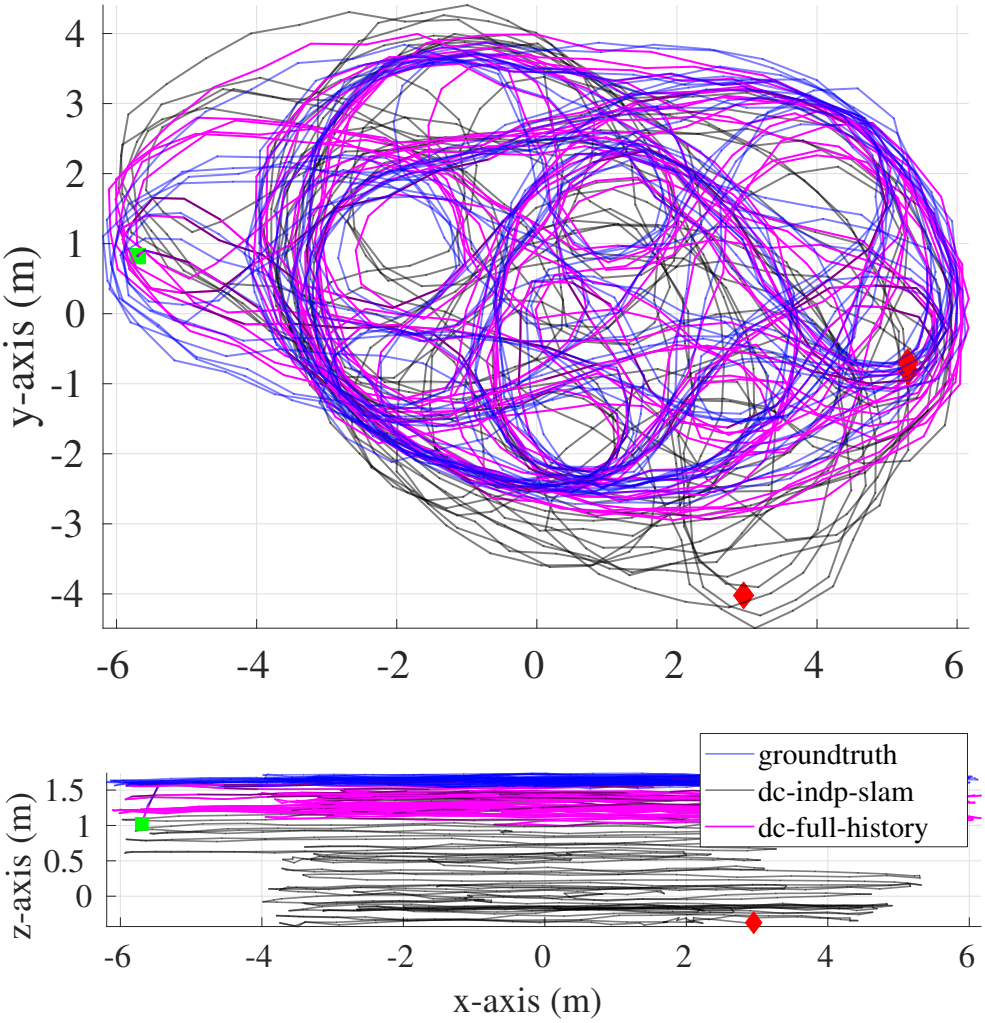



Figure 5.8: Trajectory of the groundtruth, independent, and distributed historical trajectory for Robot 0 in the Vicon room dataset. It can be seen that through the use of common historical features the drift in the z-axis direction along with improvements in the x-y accuracy can be seen. Please refer to the color figure.

groundtruth orientation and position with zero velocity. The specific datasets we run on for the TUM-VI are the room1, room3, and room5. For the Vicon room dataset, the groundtruth has been generated using the vicon2gt utility presented in Section 2.3.5. The shorter TUM-VI dataset has more time periods where multiple robots are looking at the same environmental location (26.7% and 41.8% of the frames detected common features without and with loop-closure), thus providing a good insight into an expected performance in a multi-user AR case where many users are observing the same environment at the *same time*. On the other hand, the Vicon room dataset has near-zero time periods where we are able to detect common features between robots by matching the most recent features. Thus, we use the Vicon room dataset to show the accuracy gain from leveraging historical loop-closure information by matching it to historical states (28.8% of the frames detected common loop-closure features).

### 5.5.1 TUM-VI Dataset

We use a sliding window of 11, a max of 5 SLAM features, max 30 VIO features per update, 300 active tracks, and perform online calibration of all parameters. For the historical method, we insert keyframes into our database at 5Hz and detect and match to historical keyframes at each timestep. Example feature matching performance is shown in Figure 5.5. We used a static weight of $\omega_i = 0.99$ and distributed the remaining weight to all other robot covariances used in the CI-EKF update, and for constraint measurement updates (see Eq. (5.24)), we used a value of $\omega_i = 0.995$ and injected a synthetic measurement noise of 2cm to relax the hard constraint.

The Relative Pose Error (RPE) [221] results are shown in Table 5.5 solidify the performance gains due to leveraging common features from other robot agents. The independent methods which leverage only independent VIO and SLAM feature updates have about three times the error compared to the distributed method which leverages loop-closure information. Additionally, we can see that all variations which leverage common features are able to reduce errors due to the additional information. It is also important to note that even though the distributed variants do not track

the cross-covariances between robotic states, the use of CI allows the accuracy to be near the same level as that of the centralized algorithm, and in the case where we leverage historical information (which the centralized algorithm is unable to do), we can slightly outperform for longer trajectory length. The `dc-full-history` method, which leverages loop-closure information, has a relatively constant error as the trajectory lengths increase as expected (showing its drift-free nature).

### 5.5.2 Vicon Room Dataset

We now present results on the longer hand-held, approximately 500 meter and 10 minute trajectory. We use a sliding window of 11, a max of 20 SLAM features, max 30 VIO features per update, 200 active tracks, and perform online calibration of all parameters. The RPE results for different segment lengths can be found in Table 5.6 and give the same conclusion as the previous TUM-VI dataset. It is also important to note that there is very similar performance of the `indp-slam` and `ce-cmsckf-cslam` methods (and their distributed equivalents). This is expected as there are no time periods in any of the robotic trajectories where robots are looking at the same location at the *same* time. Compared to these cases, we have huge accuracy gains due to the inclusion of common feature measurement constraints in the historical case, with halved orientation errors and a quarter of the position error at long trajectory lengths. This can be clearly seen in Figure 5.7 where the proposed `dc-full-history` is able to have little long-term drift. We also plot the groundtruth, `indp-slam`, and `dc-full-history` Robot 0 trajectories in Figure 5.8, which reinforces that by leveraging historical information we are able to prevent inherent drift in the loop-closure-free case.

## 5.6 Summary

In this thesis thrust, we have presented a distributed visual-inertial cooperative CL estimator that efficiently fuses constraints between robots and leverages temporal SLAM and loop-closure information. We have introduced two different ways to

incorporate temporal SLAM features: (i) directly update using the other robot's measurements, and (ii) if both robots are estimating the SLAM feature, a constraint between the two feature positions is leveraged. We have adapted CI to ensure consistent fusion of loop-closure constraints to other agents' historical poses and SLAM features whose cross-correlations are unknown. Extensive simulation and real-world evaluations have demonstrated the performance of the proposed method in realistic scenarios and showed impressive accuracy gains over the single robot case.

# Chapter 6

# CONCLUSION AND FUTURE WORK

## 6.1 Conclusion

In this thesis, we first introduced the state-of-the-art filter-based visual-inertial estimator and research framework OpenVINS. OpenVINS was shown to provide accurate, efficient, and consistent state estimation in both simulation and real-world experiments. Then a novel minimal plane representation, termed Closest Point (CP) plane, was introduced and shown to outperform the existing plane representation within a LiDAR-inertial estimation framework. The monocular visual-inertial OpenVINS was then extended with a novel plane tracking method and feature update which enabled estimation of environmental planar primitives. It was shown that these planes which could be tracked for significant periods of time improve the state estimation accuracy due to their ability to provide high-quality loop-closure information.

We then focused on the long-term persistent SLAM problem, where we looked to perform *consistent* and efficient state estimation. We first proposed SEVIS-3D and SEVIS-2D which leveraged a point feature map with 2D-to-3D loop-closures and a historical keyframe map with a novel 2D-to-2D loop-closure constraint, respectively. The complexity of keeping historical states was addressed through the application of the Schmidt-Kalman filter (SKF), which reduced the complexity to linear in terms of the size of the map. We additionally proposed a methodology, termed dynamic Schmidt'ing, which enabled the refinement of map features instead of fixing them as in the traditional SKF. Both SEVIS methods were evaluated on real-world experiments, and then compared against each other in a detailed numerical study within the context of prior map localization. Finally we proposed Schmidt-EKF for VI-SLAM (SEVIS)

which looked to address the large-scale localization problem through use of an efficient secondary thread which can perform relinearization and recover a sparse point feature map with *consistent* marginal feature uncertainties.

Finally, we moved to the cooperative localization (CL) problem. We proposed an efficient distributed covariance intersection (CI)-based multi-robot VI-SLAM system which leveraged the previously developed loop-closure technique and measurement models. We showed that we could limit long-term drift without requiring simultaneous viewing of common locations, leveraging cross-time historical constraints maximizing information gain and elegantly addressing the inherit asynchronousity of multi-robot systems. This OpenVINS extension to CL through CI fusion was shown to achieve state-of-the-art accuracy, consistency, and efficiency in both simulation and real-world experiments.

## 6.2 Future Work

A natural extension of the research presented in this thesis is to apply the novel Closest Point (CP) planar primitive to the explored Schmidt and feature matching methodologies to create planar environmental maps which can be efficiently leveraged for long-term persistent loop-closure. Each plane can contain a set of sparse features which can be matched to using keyframes as explored in the SEVIS systems, elegantly merging the planar feature tracking and historical matching. Instead of estimating point features, a planar map naturally reduces the state size (and thus complexity) through the compression of environmental structures into this higher-level primitive. For larger environments, the complexity of this planar map can still be reduced through the application of the SKF and whose estimate can continue to be refined through the proposed dynamic Schmidt'ing. These planes are also memory efficient large planar structures which can be leveraged during cooperative localization to reduce communication bandwidth. Outside of planes, reduced state representations either numerically with fixed-point numerical representation or reduced state representations such as 1 DoF point features holds great potential to maximize the number of retained states for

which correlations are tracked (and thus information) while decreasing computational cost.

An unexplored direction, which has seen some research [20, 35, 36, 143], is how to optimally select what map points (or planes) and/or keyframes to keep given a computational budget. While the complexity of the proposed SEVIS is able to be on $O(n)$ in terms of the map size, this is not constant, thus at some point, given a large enough space, sparsification will need to occur. A natural desire is to keep the most informative features which can benefit us both in the present and in the future when we revisit locations. A key question is how can we capture this desire in a meaningful way to rank our map features and how to select such features.

## Bibliography


[1] Sameer Agarwal, Keir Mierle, and Others. Ceres solver. http://ceres-solver.org.

[2] Artem Amirkhanov, Karl Äkerblom, and Others. Constrained delaunay triangulation. https://github.com/artem-ogre/CDT, 2019.

[3] Marc Vigo Anglada. An improved incremental algorithm for constructing restricted delaunay triangulations. *Computers & Graphics*, 21(2):215–223, 1997.

[4] Yaakov Bar-Shalom, X Rong Li, and Thiagalingam Kirubarajan. *Estimation with applications to tracking and navigation: theory algorithms and software*. John Wiley & Sons, 2004.

[5] Timothy D Barfoot. *State estimation for robotics*. Cambridge University Press, 2017.

[6] David S Bayard and Paul B Brugarolas. An estimation algorithm for vision-based exploration of small bodies in space. In *Proceedings of the 2005, American Control Conference, 2005.*, pages 4589–4595. IEEE, 2005.

[7] Michael Bloesch, Michael Burri, Sammy Omari, Marco Hutter, and Roland Siegwart. Iterated extended kalman filter based visual-inertial odometry using direct photometric feedback. *The International Journal of Robotics Research*, 36(10):1053–1072, 2017.

[8] Michael Bloesch, Sammy Omari, Marco Hutter, and Roland Siegwart. Robust visual inertial odometry using a direct ekf-based approach. In *2015 IEEE/RSJ*

*international conference on intelligent robots and systems (IROS)*, pages 298–304. IEEE, 2015.

[9] Nikolas Brasch, Aljaz Bozic, Joe Lallemand, and Federico Tombari. Semantic monocular slam for highly dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 393–400. IEEE, 2018.

[10] Guillaume Bresson, Romuald Aufrère, and Roland Chapuis. Making visual slam consistent with geo-referenced landmarks. In *2013 IEEE Intelligent Vehicles Symposium (IV)*, pages 553–558. IEEE, 2013.

[11] Martin Brossard, Axel Barrau, and Silvère Bonnabel. Ai-imu dead-reckoning. *arXiv preprint arXiv:1904.06064*, 2019.

[12] Martin Brossard, Silvère Bonnabel, and Axel Barrau. Invariant kalman filtering for visual inertial slam. In *2018 21st International Conference on Information Fusion (FUSION)*, pages 2021–2028. IEEE, 2018.

[13] Martin Brossard, Silvere Bonnabel, and Jean-Philippe Condomines. Unscented kalman filtering on lie groups. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2485–2491. IEEE, 2017.

[14] Michael Burri, Janosch Nikolic, Pascal Gohl, Thomas Schneider, Joern Rehder, Sammy Omari, Markus W Achtelik, and Roland Siegwart. The euroc micro aerial vehicle datasets. *The International Journal of Robotics Research*, 35(10):1157–1163, 2016.

[15] Yixi Cai, Wei Xu, and Fu Zhang. ikd-tree: An incremental kd tree for robotic applications. *arXiv preprint arXiv:2102.10808*, 2021.

[16] Carlos Campos, Richard Elvira, Juan J Gómez Rodríguez, José MM Montiel, and Juan D Tardós. ORB-SLAM3: An accurate open-source library for visual, visual–inertial, and multimap slam. *IEEE Transactions on Robotics*, 2021.

[17] Luca Carlone, Zsolt Kira, Chris Beall, Vadim Indelman, and Frank Dellaert. Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4290–4297. IEEE, 2014.

[18] Luis C Carrillo-Arce, Esha D Nerurkar, José L Gordillo, and Stergios I Roumeliotis. Decentralized multi-robot cooperative localization using covariance intersection. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1412–1417. IEEE, 2013.

[19] Tommaso Cavallari and Luigi Di Stefano. On-line large scale semantic fusion. In *European Conference on Computer Vision*, pages 83–99. Springer, 2016.

[20] Ming-Fang Chang, Yipu Zhao, Rajvi Shah, Jakob J Engel, Michael Kaess, and Simon Lucey. Long-term visual map sparsification with heterogeneous gnn. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2406–2415, 2022.

[21] Averil B. Chatfield. *Fundamentals of High Accuracy Inertial Navigation.* American Institute of Aeronautics and Astronautics, Inc., Reston, VA, 1997.

[22] Chuchu Chen, Yulin Yang, Patrick Geneva, and Guoquan Huang. FEJ2: A consistent visual-inertial state estimator design. In *Proc. International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, May 2022.

[23] Gregory S Chirikjian. *Stochastic models, information theory, and Lie groups, volume 2: Analytic methods and modern applications*, volume 2. Springer Science & Business Media, 2011.

[24] Javier Civera, Andrew J Davison, and JM Martinez Montiel. Inverse depth parametrization for monocular slam. *IEEE transactions on robotics*, 24(5):932–945, 2008.

[25] Ronald Clark, Sen Wang, Hongkai Wen, Andrew Markham, and Niki Trigoni. Vinet: Visual-inertial odometry as a sequence-to-sequence learning problem. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[26] James M Coughlan and Alan L Yuille. Manhattan world: Compass direction from a single image by bayesian inference. In *IEEE International Conference on Computer Vision*, volume 2, pages 941–947. IEEE, 1999.

[27] Andrei Cramariuc, Lukas Bernreiter, Florian Tschopp, Marius Fehr, Victor Reijgwart, Juan Nieto, Roland Siegwart, and Cesar Cadena. maplab 2.0–a modular and multi-modal mapping framework. *IEEE Robotics and Automation Letters*, 2022.

[28] Frank Dellaert. Factor graphs and gtsam: A hands-on introduction. Technical report, Georgia Institute of Technology, 2012.

[29] Tien Do, Khiem Vuong, Stergios I Roumeliotis, and Hyun Soo Park. Surface normal estimation of tilted images via spatial rectifier. In *European Conference on Computer Vision*, pages 265–280. Springer, 2020.

[30] Kevin J Doherty, David M Rosen, and John J Leonard. Spectral measurement sparsification for pose-graph slam. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 01–08. IEEE, 2022.

[31] Jingming Dong, Xiaohan Fei, and Stefano Soatto. Visual-inertial-semantic scene representation for 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 960–970, 2017.

[32] Tue-Cuong Dong-Si and Anastasios I Mourikis. Closed-form solutions for vision-aided inertial navigation. Technical report, Dept. of Electrical Engineering, University of California, Riverside, 2011.

[33] Tue-Cuong Dong-Si and Anastasios I Mourikis. Estimator initialization in vision-aided inertial navigation with unknown camera-imu calibration. In *2012*

*IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1064–1071. IEEE, 2012.

[34] Ryan C DuToit, Joel A Hesch, Esha D Nerurkar, and Stergios I Roumeliotis. Consistent map-based 3d localization on mobile devices. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 6253–6260. IEEE, 2017.

[35] Marcin Dymczyk, Simon Lynen, Michael Bosse, and Roland Siegwart. Keep it brief: Scalable creation of compressed localization maps. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2536–2542. IEEE, 2015.

[36] Marcin Dymczyk, Simon Lynen, Titus Cieslewski, Michael Bosse, Roland Siegwart, and Paul Furgale. The gist of maps-summarizing experience for lifelong localization. In *2015 IEEE international conference on robotics and automation (ICRA)*, pages 2767–2773. IEEE, 2015.

[37] Ethan Eade. Gauss-newton / levenberg-marquardt optimization. *Technical Report*, 2013. Available: https://www.ethaneade.org/optimization.pdf.

[38] Kevin Eckenhoff, Patrick Geneva, Jesse Bloecker, and Guoquan Huang. Multi-camera visual-inertial navigation with online intrinsic and extrinsic calibration. In *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[39] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Continuous preintegration theory for visual-inertial navigation. Technical Report RPNG-2018-CPI, University of Delaware, 2018. Available: http://udel.edu/~ghuang/papers/tr_cpi.pdf.

[40] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Closed-form preintegration methods for graph-based visual-inertial navigation. *International Journal of Robotics Research*, 38(5):563–586, 2019.

[41] Kevin Eckenhoff, Patrick Geneva, and Guoquan Huang. Sensor-failure-resilient multi-imu visual-inertial navigation. In *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[42] Kevin Eckenhoff, Patrick Geneva, Nathaniel Merrill, and Guoquan Huang. Schmidt-ekf-based visual-inertial moving object tracking. In *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020.

[43] Kevin Eckenhoff, Liam Paull, and Guoquan Huang. Decoupled, consistent node removal and edge sparsification for graph-based SLAM. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3275–3282, Daejeon, Korea, October 2016.

[44] Kevin Eckenhoff, Yulin Yang, Patrick Geneva, and Guoquan Huang. Tightly-coupled visual-inertial localization and 3D rigid-body target tracking. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):1541–1548, 2019.

[45] Jakob Engel, Vladlen Koltun, and Daniel Cremers. Direct sparse odometry. *IEEE transactions on pattern analysis and machine intelligence*, 40(3):611–625, 2017.

[46] Jakob Engel, Thomas Schöps, and Daniel Cremers. Lsd-slam: Large-scale direct monocular slam. In *European conference on computer vision*, pages 834–849. Springer, 2014.

[47] Jakob Engel, Jörg Stückler, and Daniel Cremers. Large-scale direct slam with stereo cameras. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1935–1942. IEEE, 2015.

[48] Mahdi Abolfazli Esfahani, Keyu Wu, Shenghai Yuan, and Han Wang. Towards utilizing deep uncertainty in traditional slam. In *2019 IEEE 15th International Conference on Control and Automation (ICCA)*, pages 344–349. IEEE, 2019.

[49] Xiaohan Fei and Stefano Soatto. Visual-inertial object detection and mapping. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 301–317, 2018.

[50] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[51] Yarin Gal. *Uncertainty in deep learning*. PhD thesis, PhD thesis, University of Cambridge, 2016.

[52] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

[53] Dorian Gálvez-López and J. D. Tardós. Bags of binary words for fast place recognition in image sequences. *IEEE Transactions on Robotics*, 28(5):1188–1197, 2012.

[54] Xiang Gao, Rui Wang, Nikolaus Demmel, and Daniel Cremers. Ldso: Direct sparse odometry with loop closure. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2198–2204. IEEE, 2018.

[55] Jochen Gast and Stefan Roth. Lightweight probabilistic deep networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3369–3378, 2018.

[56] Patrick Geneva. Research notes and jacobians: Continuous-time visual-inertial trajectory estimation with event cameras, 2018.

[57] Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Asynchronous multi-sensor fusion for 3d mapping and localization. In *Proc. of the IEEE International Conference on Robotics and Automation*, Brisbane, Australia, May 2018.

[58] Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Complexity analysis: A linear-complexity ekf for visual-inertial navigation with loop closures. Technical Report RPNG-2019-LOOP, University of Delaware, 2019. Available: http://udel.edu/~ghuang/papers/tr_loop.pdf.

[59] Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. A linear-complexity EKF for visual-inertial navigation with loop closures. In *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[60] Patrick Geneva, Kevin Eckenhoff, Woosik Lee, Yulin Yang, and Guoquan Huang. OpenVINS: a research platform for visual-inertial estimation. In *Proc. of the IEEE International Conference on Robotics and Automation*, Paris, France, 2020.

[61] Patrick Geneva and Guoquan Huang. vicon2gt: Derivations and analysis. Technical Report RPNG-2020-VICON2GT, University of Delaware, 2020. Available: http://udel.edu/~ghuang/papers/tr_vicon2gt.pdf.

[62] Patrick Geneva and Guoquan Huang. Map-based visual-inertial localization: A numerical study. In *Proc. International Conference on Robotics and Automation (ICRA)*, Philadelphia, PA, May 2022.

[63] Patrick Geneva and Guoquan Huang. Map-based visual-inertial localization: A numerical study. Technical Report RPNG-2022-MAPPING, University of Delaware, 2022.

[64] Patrick Geneva and Guoquan Huang. Openvins state initialization: Details and derivations. Technical Report RPNG-2022-INIT, University of Delaware, 2022. Available: https://pgeneva.com/downloads/reports/tr_init.pdf.

[65] Patrick Geneva, James Maley, and Guoquan Huang. An efficient schmidt-ekf for 3D visual-inertial SLAM. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR)*, Long Beach, CA, June 2019.

[66] Giorgio Grisetti, Rainer Kümmerle, Hauke Strasdat, and Kurt Konolige. g2o: A general framework for (hyper) graph optimization. In *International conference on robotics and automation (ICRA)*, pages 9–13, 2011.

[67] Chao X Guo and Stergios I Roumeliotis. IMU-RGBD camera navigation using point and plane features. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3164–3171. IEEE, 2013.

[68] Chao X Guo, Kourosh Sartipi, Ryan C DuToit, Georgios A Georgiou, Ruipeng Li, John O'Leary, Esha D Nerurkar, Joel A Hesch, and Stergios I Roumeliotis. Large-scale cooperative 3d visual-inertial mapping in a manhattan world. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1071–1078. IEEE, 2016.

[69] Chao X. Guo, Kourosh Sartipi, Ryan C. DuToit, Georgios A. Georgiou, Ruipeng Li, John O'Leary, Esha D. Nerurkar, Joel A. Hesch, and Stergios I. Roumeliotis. Resource-aware large-scale cooperative three-dimensional mapping using multiple mobile devices. *IEEE Transactions on Robotics*, 34(5):1349–1369, 2018.

[70] Christoph Hertzberg, René Wagner, Udo Frese, and Lutz Schröder. Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds. *Information Fusion*, 14(1):57–77, 2013.

[71] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Consistency analysis and improvement of vision-aided inertial navigation. *IEEE Transactions on Robotics*, 30(1):158–176, 2013.

[72] Joel A Hesch, Dimitrios G Kottas, Sean L Bowman, and Stergios I Roumeliotis. Camera-imu-based localization: Observability analysis and consistency improvement. *The International Journal of Robotics Research*, 33(1):182–201, 2014.

[73] Joel A Hesch, Faraz M Mirzaei, Gian Luca Mariottini, and Stergios I Roumeliotis. A laser-aided inertial navigation system (l-ins) for human localization in unknown indoor environments. In *2010 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5376–5382. IEEE, 2010.

[74] Blanka Hovart, Anastasis Kratsios, Yannick Limmer, and Xuwei Yang. Deep kalman filters can filter. *arXiv preprint arXiv:2310.19603*, 2023.

[75] Ming Hsiao, Eric Westman, and Michael Kaess. Dense planar-inertial slam with structural constraints. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6521–6528. IEEE, 2018.

[76] Ming Hsiao, Eric Westman, Guofeng Zhang, and Michael Kaess. Keyframe-based dense planar slam. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5110–5117. IEEE, 2017.

[77] Jiaxin Hu, Jun Hu, Y-J. Shen, Xiaomin Lang, Bo Zang, Guoquan Huang, and Yinian Mao. 1d-lrf aided visual-inertial odometry for high-altitude mav flight. In *Proc. of the IEEE International Conference on Robotics and Automation*, Philadelphia, PA, May 2022.

[78] Zheng Huai and Guoquan Huang. Robocentric visual-inertial odometry. *International Journal of Robotics Research*, April 2019.

[79] Zheng Huai and Guoquan Huang. Markov parallel tracking and mapping for probabilistic slam. In *Proc. of the IEEE International Conference on Robotics and Automation*, Xi'an, China, 2021.

[80] Guoquan Huang. Visual-inertial navigation: A concise review. In *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[81] Guoquan Huang, Michael Kaess, and John Leonard. Consistent sparsification for graph optimization. In *Proc. of the European Conference on Mobile Robots*, pages 150–157, Barcelona, Spain, September 2013.

[82] Guoquan Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. Analysis and improvement of the consistency of extended Kalman filter-based SLAM. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 473–479, Pasadena, CA, May 2008.

[83] Guoquan Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. A first-estimates Jacobian EKF for improving SLAM consistency. In *Proc. of the 11th International Symposium on Experimental Robotics*, Athens, Greece, July 2008.

[84] Guoquan Huang, Anastasios I. Mourikis, and Stergios I. Roumeliotis. Observability-based rules for designing consistent EKF SLAM estimators. *International Journal of Robotics Research*, 29(5):502–528, April 2010.

[85] Hordur Johannsson, Michael Kaess, Maurice Fallon, and John J Leonard. Temporally scalable visual slam using a reduced pose graph. In *2013 IEEE International Conference on Robotics and Automation*, pages 54–61. IEEE, 2013.

[86] Simon J Julier and Jeffrey K Uhlmann. A non-divergent estimation algorithm in the presence of unknown correlations. In *Proceedings of the 1997 American Control Conference (Cat. No. 97CH36041)*, volume 4, pages 2369–2373. IEEE, 1997.

[87] SJ Julier and Jeffrey K Uhlmann. General decentralized data fusion with covariance intersection. *Handbook of multisensor data fusion: theory and practice*, pages 319–344, 2009.

[88] Roland Jung, Christian Brommer, and Stephan Weiss. Decentralized collaborative state estimation for aided inertial navigation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4673–4679. IEEE, 2020.

[89] Michael Kaess. Simultaneous localization and mapping with infinite planes. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4605–4611. IEEE, 2015.

[90] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. iSAM2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.

[91] Marco Karrer, Patrik Schmuck, and Margarita Chli. Cvi-slam—collaborative visual-inertial slam. *IEEE Robotics and Automation Letters*, 3(4):2762–2769, 2018.

[92] Anton Kasyanov, Francis Engelmann, Jörg Stückler, and Bastian Leibe. Keyframe-based visual-inertial online slam with relocalization. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 6662–6669. IEEE, 2017.

[93] Tong Ke, Kejian J Wu, and Stergios I Roumeliotis. Rise-slam: A resource-aware inverse schmidt estimator for slam. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 354–361. IEEE, 2019.

[94] Alex Kendall and Roberto Cipolla. Modelling uncertainty in deep learning for camera relocalization. In *2016 IEEE international conference on Robotics and Automation (ICRA)*, pages 4762–4769. IEEE, 2016.

[95] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in neural information processing systems*, pages 5574–5584, 2017.

[96] Alex Kendall, Yarin Gal, and Roberto Cipolla. Multi-task learning using uncertainty to weigh losses for scene geometry and semantics. In *Proceedings of*

*the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7482–7491, 2018.

[97] Alex Kendall, Matthew Grimes, and Roberto Cipolla. Posenet: A convolutional network for real-time 6-dof camera relocalization. In *Proceedings of the IEEE international conference on computer vision*, pages 2938–2946, 2015.

[98] Alex Guy Kendall. *Geometry and uncertainty in deep learning for computer vision*. PhD thesis, University of Cambridge, 2019.

[99] Pyojin Kim, Brian Coltin, and H Jin Kim. Linear rgb-d slam for planar environments. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 333–348, 2018.

[100] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pages 225–234. IEEE, 2007.

[101] Dmitry Kopitkov and Vadim Indelman. Bayesian information recovery from cnn for probabilistic inference. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7795–7802. IEEE, 2018.

[102] Dimitrios G Kottas and Stergios I Roumeliotis. Exploiting urban scenes for vision-aided inertial navigation. In *Robotics: Science and Systems*, 2013.

[103] Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1511.05121*, 2015.

[104] Tristan Laidlow, Michael Bloesch, Wenbin Li, and Stefan Leutenegger. Dense rgb-d-inertial slam with map deformations. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 6741–6748. IEEE, 2017.

[105] Pierre-Yves Lajoie, Benjamin Ramtoula, Yun Chang, Luca Carlone, and Giovanni Beltrame. Door-slam: Distributed, online, and outlier resilient slam for robotic teams. *IEEE Robotics and Automation Letters*, 5(2):1656–1663, 2020.

[106] Woosik Lee, Yulin Yang, and Guoquan Huang. Efficient multi-sensor aided inertial navigation with online calibration. In *Proc. of the IEEE International Conference on Robotics and Automation*, Xi'an, China, 2021.

[107] Keith YK Leung, Yoni Halpern, Timothy D Barfoot, and Hugh HT Liu. The utias multi-robot cooperative localization and mapping dataset. *The International Journal of Robotics Research*, 30(8):969–974, 2011.

[108] Stefan Leutenegger. OKVIS2: Realtime scalable visual-inertial slam with loop closure. *arXiv preprint arXiv:2202.09199*, 2022.

[109] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[110] Stefan Leutenegger, Simon Lynen, Michael Bosse, Roland Siegwart, and Paul Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *The International Journal of Robotics Research*, 34(3):314–334, 2015.

[111] Mingyang Li. *Visual-Inertial Odometry on Resource-Constrained Systems*. PhD thesis, UC Riverside, 2014.

[112] Mingyang Li and Anastasios I Mourikis. High-precision, consistent ekf-based visual-inertial odometry. *The International Journal of Robotics Research*, 32(6):690–711, 2013.

[113] Mingyang Li and Anastasios I Mourikis. Optimization-based estimator design for vision-aided inertial navigation. In *Robotics: Science and Systems*, pages 241–248. Berlin Germany, 2013.

[114] Mingyang Li and Anastasios I Mourikis. Online temporal calibration for Camera–IMU systems: Theory and algorithms. *The International Journal of Robotics Research*, 33(7):947–964, 2014.

[115] Mingyang Li, Hongsheng Yu, Xing Zheng, and Anastasios I Mourikis. High-fidelity sensor modeling and self-calibration in vision-aided inertial navigation. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 409–416. IEEE, 2014.

[116] Xin Li, Yijia He, Jinlong Lin, and Xiao Liu. Leveraging planar regularities for point line visual-inertial odometry. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5120–5127. IEEE, 2020.

[117] Yanyan Li, Raza Yunus, Nikolas Brasch, Nassir Navab, and Federico Tombari. RGB-D SLAM with structural regularities. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11581–11587. IEEE, 2021.

[118] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. PlaneRCNN: 3d plane detection and reconstruction from a single image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4450–4459, 2019.

[119] Haomin Liu, Mingyu Chen, Guofeng Zhang, Hujun Bao, and Yingze Bao. Ice-ba: Incremental, consistent and efficient bundle adjustment for visual-inertial slam. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1974–1982, 2018.

[120] Shing Yan Loo, Ali Jahani Amiri, Syamsiah Mashohor, Sai Hong Tang, and Hong Zhang. Cnn-svo: Improving the mapping in semi-direct visual odometry using single-image depth prediction. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 5218–5223. IEEE, 2019.

[121] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, BC, August 1981.

[122] Lukas Luft, Tobias Schubert, Stergios I Roumeliotis, and Wolfram Burgard. Recursive decentralized localization for multi-robot systems with asynchronous pairwise communication. *The International Journal of Robotics Research*, 37(10):1152–1167, 2018.

[123] Lukas Lukas Von Stumberg and Daniel Cremers. Dm-vio: Delayed marginalization visual-inertial odometry. *IEEE Robotics and Automation Letters*, 7(2):1408–1415, 2022.

[124] Lukas Lukas Von Stumberg, Vladyslav Usenko, and Daniel Cremers. Direct sparse visual-inertial odometry using dynamic marginalization. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2510–2517. IEEE, 2018.

[125] Simon Lynen, Michael Bosse, Paul Furgale, and Roland Siegwart. Placeless place-recognition. In *2014 2nd International Conference on 3D Vision*, volume 1, pages 303–310. IEEE, 2014.

[126] Simon Lynen, Torsten Sattler, Michael Bosse, Joel A Hesch, Marc Pollefeys, and Roland Siegwart. Get out of my lab: Large-scale, real-time visual-inertial localization. In *Robotics: Science and Systems*, volume 1, page 1, 2015.

[127] Simon Lynen, Bernhard Zeisl, Dror Aiger, Michael Bosse, Joel Hesch, Marc Pollefeys, Roland Siegwart, and Torsten Sattler. Large-scale, real-time visual–inertial localization revisited. *The International Journal of Robotics Research*, 39(9):1061–1084, 2020.

[128] Lingni Ma, Christian Kerl, Jörg Stückler, and Daniel Cremers. CPA-SLAM: Consistent plane-model alignment for direct rgb-d slam. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1285–1291. IEEE, 2016.

[129] Joshua G Mangelson, Derrick Dominic, Ryan M Eustice, and Ram Vasudevan. Pairwise consistent measurement set maximization for robust multi-robot map merging. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2916–2923. IEEE, 2018.

[130] Agostino Martinelli. Cooperative visual-inertial odometry: Analysis of singularities, degeneracies and minimal cases. *IEEE Robotics and Automation Letters*, 5(2):668–675, 2020.

[131] Agostino Martinelli, Alexander Oliva, and Bernard Mourrain. Cooperative visual-inertial sensor fusion: The analytic solution. *IEEE Robotics and Automation Letters*, 4(2):453–460, 2019.

[132] Peter S. Maybeck. *Stochastic Models, Estimation, and Control*, volume 1. Academic Press, London, 1979.

[133] Mladen Mazuran, Wolfram Burgard, and Gian Diego Tipaldi. Nonlinear factor recovery for long-term slam. *The International Journal of Robotics Research*, 35(1-3):50–72, 2016.

[134] John McCormac, Ankur Handa, Andrew Davison, and Stefan Leutenegger. Semanticfusion: Dense 3d semantic mapping with convolutional neural networks. In *2017 IEEE International Conference on Robotics and automation (ICRA)*, pages 4628–4635. IEEE, 2017.

[135] Igor V Melnyk, Joel A Hesch, and Stergios I Roumeliotis. Cooperative vision-aided inertial navigation using overlapping views. In *2012 IEEE International Conference on Robotics and Automation*, pages 936–943. IEEE, 2012.

[136] Nathaniel Merrill and Guoquan Huang. Lightweight unsupervised deep loop closure. In *Proc. of Robotics: Science and Systems (RSS)*, Pittsburgh, PA, June 2018.

[137] Nathaniel Merrill and Guoquan Huang. CALC2.0: Combining appearance, semantic and geometric information for robust and efficient visual loop closure. In *2019 International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, November 2019.

[138] Sven Middelberg, Torsten Sattler, Ole Untzelmann, and Leif Kobbelt. Scalable 6-dof localization on mobile devices. In *European conference on computer vision*, pages 268–283. Springer, 2014.

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

[140] Anastasios I Mourikis and Stergios I Roumeliotis. A dual-layer estimator architecture for long-term localization. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8. IEEE, 2008.

[141] Anastasios I Mourikis, Nikolas Trawny, Stergios I Roumeliotis, Andrew E Johnson, Adnan Ansar, and Larry Matthies. Vision-aided inertial navigation for spacecraft entry, descent, and landing. *IEEE Transactions on Robotics*, 25(2):264–280, 2009.

[142] Elias Mueggler, Guillermo Gallego, Henri Rebecq, and Davide Scaramuzza. Continuous-time visual-inertial odometry for event cameras. *IEEE Transactions on Robotics*, 34(6):1425–1440, 2018.

[143] Peter Mühlfellner, Mathias Bürki, Michael Bosse, Wojciech Derendarz, Roland Philippsen, and Paul Furgale. Summary maps for lifelong visual localization. *Journal of Field Robotics*, 33(5):561–590, 2016.

[144] Raul Mur-Artal, Jose Maria Martinez Montiel, and Juan D Tardos. ORB-SLAM: a versatile and accurate monocular slam system. *IEEE transactions on robotics*, 31(5):1147–1163, 2015.

[145] Raul Mur-Artal and Juan D Tardós. ORB-SLAM2: An open-source slam system for monocular, stereo, and rgb-d cameras. *IEEE transactions on robotics*, 33(5):1255–1262, 2017.

[146] Raúl Mur-Artal and Juan D Tardós. Visual-inertial monocular slam with map reuse. *IEEE Robotics and Automation Letters*, 2(2):796–803, 2017.

[147] Esha D Nerurkar, Kejian J Wu, and Stergios I Roumeliotis. C-KLAM: Constrained keyframe-based localization and mapping. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 3638–3643. IEEE, 2014.

[148] Janosch Nikolic, Joern Rehder, Michael Burri, Pascal Gohl, Stefan Leutenegger, Paul T Furgale, and Roland Siegwart. A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam. In *2014 IEEE international conference on robotics and automation (ICRA)*, pages 431–437. IEEE, 2014.

[149] OpenCV Developers Team. Open source computer vision (OpenCV) library. Available: http://opencv.org.

[150] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, Max Pfingsthorn, Sören Schwertfeger, and Jann Poppinga. Online three-dimensional slam by registration of large planar surface segments and closed-form pose-graph relaxation. *Journal of Field Robotics*, 27(1):52–84, 2010.

[151] Kaustubh Pathak, Andreas Birk, Narunas Vaskevicius, and Jann Poppinga. Fast registration based on noisy planes with unknown correspondences for 3-d mapping. *IEEE Transactions on Robotics*, 26(3):424–441, 2010.

[152] Alonso Patron-Perez, Steven Lovegrove, and Gabe Sibley. A spline-based trajectory representation for sensor fusion and rolling shutter cameras. *International Journal of Computer Vision*, 113(3):208–219, 2015.

[153] Mrinal K Paul, Kejian Wu, Joel A Hesch, Esha D Nerurkar, and Stergios I Roumeliotis. A comparative analysis of tightly-coupled monocular, binocular, and stereo vins. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 165–172. IEEE, 2017.

[154] Liam Paull, Guoquan Huang, Mae Seto, and John Leonard. Communication-constrained multi-auv cooperative slam. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 509–516, Seattle, WA, May 2015.

[155] Lina M Paz, Juan D Tardós, and José Neira. Divide and conquer: Ekf slam in $o(n)$. *IEEE Transactions on Robotics*, 24(5):1107–1120, 2008.

[156] Mikael Persson and Klas Nordberg. Lambda twist: An accurate fast robust perspective three point (p3p) solver. In *Proceedings of the European conference on computer vision (ECCV)*, pages 318–332, 2018.

[157] Sai Manoj Prakhya, Liu Bingbing, Yan Rui, and Weisi Lin. A closed-form estimate of 3d icp covariance. In *2015 14th IAPR International Conference on Machine Vision Applications (MVA)*, pages 526–529. IEEE, 2015.

[158] Pedro F Proença and Yang Gao. Probabilistic combination of noisy points and planes for rgb-d odometry. In *Conference Towards Autonomous Robotic Systems*, pages 340–350. Springer, 2017.

[159] Tong Qin, Peiliang Li, and Shaojie Shen. Relocalization, global optimization and map merging for monocular visual-inertial slam. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1197–1204. IEEE, 2018.

[160] Tong Qin, Peiliang Li, and Shaojie Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Transactions on Robotics*, 34(4):1004–1020, 2018.

[161] Tong Qin, Jie Pan, Shaozu Cao, and Shaojie Shen. A general optimization-based framework for local odometry estimation with multiple sensors. *arXiv preprint arXiv:1901.03638*, 2019.

[162] Karnik Ram, Chaitanya Kharyal, Sudarshan S Harithas, and K Madhava Krishna. RP-VIO: Robust plane-based visual-inertial odometry for dynamic environments. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 9198–9205. IEEE, 2021.

[163] Guy Revach, Nir Shlezinger, Xiaoyong Ni, Adria Lopez Escoriza, Ruud JG Van Sloun, and Yonina C Eldar. Kalmannet: Neural network aided kalman filtering for partially known dynamics. *IEEE Transactions on Signal Processing*, 70:1532–1547, 2022.

[164] Antoni Rosinol. Densifying sparse vio: a mesh-based approach using structural regularities. Master's thesis, ETH Zurich; Massachusetts Institute of Technology (MIT), 2018.

[165] Antoni Rosinol, Marcus Abate, Yun Chang, and Luca Carlone. Kimera: an open-source library for real-time metric-semantic localization and mapping. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1689–1696. IEEE, 2020.

[166] Antoni Rosinol, Torsten Sattler, Marc Pollefeys, and Luca Carlone. Incremental visual-inertial 3d mesh generation with structural regularities. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8220–8226. IEEE, 2019.

[167] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European conference on computer vision*, pages 430–443. Springer, 2006.

[168] Edward Rosten, Reid Porter, and Tom Drummond. Faster and better: A machine learning approach to corner detection. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):105–119, 2010.

[169] Stergios I Roumeliotis and George A Bekey. Distributed multirobot localization. *IEEE transactions on robotics and automation*, 18(5):781–795, 2002.

[170] Stergios I Roumeliotis and Joel W Burdick. Stochastic cloning: A generalized framework for processing relative state measurements. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1788–1795. IEEE, 2002.

[171] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In *2011 IEEE international conference on Computer Vision (ICCV)*, pages 2564–2571. IEEE, 2011.

[172] Radu Bogdan Rusu and Steve Cousins. 3d is here: Point cloud library (pcl). In *2011 IEEE International Conference on Robotics and automation (ICRA)*, pages 1–4. IEEE, 2011.

[173] Renato F Salas-Moreno, Ben Glocken, Paul HJ Kelly, and Andrew J Davison. Dense planar slam. In *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 157–164. IEEE, 2014.

[174] Kourosh Sartipi, Ryan C DuToit, Christopher B Cobar, and Stergios I Roumeliotis. Decentralized visual-inertial localization and mapping on mobile devices for augmented reality. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2145–2152. IEEE, 2019.

[175] Martin Scheiber, Jeff Delaune, Stephan Weiss, and Roland Brockers. Mid-air range-visual-inertial estimator initialization for micro air vehicles. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7613–7619. IEEE, 2021.

[176] Grant Schindler and Frank Dellaert. Atlanta world: An expectation maximization framework for simultaneous low-level edge grouping and camera calibration in complex man-made environments. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–I. IEEE, 2004.

[177] Stanley F Schmidt. Application of state-space methods to navigation problems. In *Advances in control systems*, volume 3, pages 293–340. Elsevier, 1966.

[178] Thomas Schneider, Marcin Dymczyk, Marius Fehr, Kevin Egger, Simon Lynen, Igor Gilitschenski, and Roland Siegwart. maplab: An open framework for research in visual-inertial mapping and localization. *IEEE Robotics and Automation Letters*, 3(3):1418–1425, 2018.

[179] David Schubert, Thore Goll, Nikolaus Demmel, Vladyslav Usenko, Jörg Stückler, and Daniel Cremers. The tum vi benchmark for evaluating visual-inertial odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1680–1687. IEEE, 2018.

[180] Stefan Schubert, Peer Neubert, Sourav Garg, Michael Milford, and Tobias Fischer. Visual place recognition: A tutorial. *arXiv preprint arXiv:2303.03281*, 2023.

[181] Dan Simon. Kalman filtering with state constraints: a survey of linear and nonlinear algorithms. *IET Control Theory & Applications*, 4(8):1303–1318, 2010.

[182] Ke Sun, Kartik Mohta, Bernd Pfrommer, Michael Watterson, Sikang Liu, Yash Mulgaonkar, Camillo J Taylor, and Vijay Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters*, 3(2):965–972, 2018.

[183] Niko Sünderhauf, Trung T Pham, Yasir Latif, Michael Milford, and Ian Reid. Meaningful maps with object-oriented semantic mapping. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5079–5085. IEEE, 2017.

[184] Yuichi Taguchi, Yong-Dian Jian, Srikumar Ramalingam, and Chen Feng. Point-plane slam for hand-held 3d sensors. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5182–5189. IEEE, 2013.

[185] Keisuke Tateno, Federico Tombari, Iro Laina, and Nassir Navab. Cnn-slam: Real-time dense monocular slam with learned depth prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6243–6252, 2017.

[186] Nikolas Trawny and Stergios I. Roumeliotis. Indirect Kalman filter for 3D attitude estimation. Technical report, University of Minnesota, Dept. of Comp. Sci. & Eng., March 2005.

[187] Alexander JB Trevor, John G Rogers, and Henrik I Christensen. Planar surface slam with 3d and 2d sensors. In *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3041–3048. IEEE, 2012.

[188] Bill Triggs, Philip F McLauchlan, Richard I Hartley, and Andrew W Fitzgibbon. Bundle adjustment—a modern synthesis. In *Vision Algorithms: Theory and*

*Practice: International Workshop on Vision Algorithms Corfu, Greece, September 21–22, 1999 Proceedings*, pages 298–372. Springer, 2000.

[189] Cihan Ulaş and Hakan Temeltaş. A fast and robust scan matching algorithm based on ml-ndt and feature extraction. In *2011 International Conference on Mechatronics and Automation (ICMA)*, pages 1751–1756. IEEE, 2011.

[190] Vladyslav Usenko, Nikolaus Demmel, David Schubert, Jörg Stückler, and Daniel Cremers. Visual-inertial mapping with non-linear factor recovery. *IEEE Robotics and Automation Letters*, 5(2):422–429, 2019.

[191] Alexander Vakhitov, Luis Ferraz, Antonio Agudo, and Francesc Moreno-Noguer. Uncertainty-aware camera pose estimation from points and lines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4659–4668, 2021.

[192] Jonathan Ventura, Clemens Arth, Gerhard Reitmayr, and Dieter Schmalstieg. Global localization from monocular slam on a mobile phone. *IEEE transactions on visualization and computer graphics*, 20(4):531–539, 2014.

[193] John Vial, Hugh Durrant-Whyte, and Tim Bailey. Conservative sparsification for efficient and consistent approximate estimation. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 886–893. IEEE, 2011.

[194] Diego Viejo and Miguel Cazorla. 3d plane-based egomotion for slam on semi-structured environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2761–2766. IEEE, 2007.

[195] Rui Wang, David Geraghty, Kevin Matzen, Richard Szeliski, and Jan-Michael Frahm. VPLNet: Deep single view normal estimation with vanishing points and lines. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 689–698, 2020.

[196] Sen Wang, Ronald Clark, Hongkai Wen, and Niki Trigoni. End-to-end, sequence-to-sequence probabilistic visual odometry through deep neural networks. *The International Journal of Robotics Research*, 37(4-5):513–542, 2018.

[197] Hao Wei, Fulin Tang, Zewen Xu, and Yihong Wu. Structural regularity aided visual-inertial odometry with novel coordinate alignment and line triangulation. *IEEE Robotics and Automation Letters*, 7(4):10613–10620, 2022.

[198] Jan Weingarten and Roland Siegwart. 3d slam using planar segments. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3062–3067. IEEE, 2006.

[199] Kanzhi Wu, Teng Zhang, Daobilige Su, Shoudong Huang, and Gamini Dissanayake. An invariant-ekf vins algorithm for improving consistency. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 1578–1585. IEEE, 2017.

[200] Kejian Wu, Ahmed M Ahmed, Georgios A Georgiou, and Stergios I Roumeliotis. A square root inverse filter for efficient vision-aided inertial navigation on mobile devices. In *Robotics: Science and Systems*, volume 2. Rome, Italy, 2015.

[201] Junhao Xiao, Bejamin Adler, and Houxiang Zhang. 3d point cloud registration based on planar surfaces. In *2012 IEEE Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI)*, pages 40–45. IEEE, 2012.

[202] Hao Xu, Luqi Wang, Yichen Zhang, Kejie Qiu, and Shaojie Shen. Decentralized visual-inertial-UWB fusion for relative state estimation of aerial swarm. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8776–8782. IEEE, 2020.

[203] Fengting Yang and Zihan Zhou. Recovering 3d planes from a single image via convolutional neural networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 85–100, 2018.

[204] Shichao Yang and Sebastian Scherer. Monocular object and plane slam in structured environments. *IEEE Robotics and Automation Letters*, 4(4):3145–3152, 2019.

[205] Yulin Yang. *Aided Inertial Navigation System: Analysis and Algorithms*. PhD thesis, Unviersity of Delaware, 2024.

[206] Yulin Yang, Chuchu Chen, Woosik Lee, and Guoquan Paul Huang. Decoupled right invariant error states for consistent visual-inertial navigation. *IEEE Robotics and Automation Letters*, January 2022.

[207] Yulin Yang, Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Degenerate motion analysis for aided INS with online spatial and temporal calibration. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):2070–2077, 2019.

[208] Yulin Yang, Patrick Geneva, Kevin Eckenhoff, and Guoquan Huang. Visual-inertial navigation with point and line features. In *Proc. International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, November 2019.

[209] Yulin Yang, Patrick Geneva, Xingxing Zuo, Kevin Eckenhoff, Yong Liu, and Guoquan Huang. Tightly-coupled aided inertial navigation with point and plane features. In *Proc. International Conference on Robotics and Automation*, Montreal, Canada, May 2019.

[210] Yulin Yang, Patrick Geneva, Xingxing Zuo, and Guoquan Huang. Online self-calibration for visual-inertial navigation systems: Models, analysis and degeneracy. *IEEE Transactions on Robotics*, May 2023.

[211] Yulin Yang and Guoquan Huang. Aided inertial navigation with geometric features: Observability analysis. In *Proc. of the IEEE International Conference on Robotics and Automation*, Brisbane, Australia, May 2018.

[212] Yulin Yang and Guoquan Huang. Observability analysis of aided ins with heterogeneous features of points, lines and planes. *IEEE Transactions on Robotics*, 35(6):399–1418, December 2019.

[213] Yulin Yang, James Maley, and Guoquan Huang. Null-space-based marginalization: Analysis and algorithm. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 6749–6755, Vancouver, Canada, September 2017.

[214] Wei Yin, Yifan Liu, and Chunhua Shen. Virtual normal: Enforcing geometric constraints for accurate and robust depth prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[215] Chao Yu, Zuxin Liu, Xin-Jun Liu, Fugui Xie, Yi Yang, Qi Wei, and Qiao Fei. Ds-slam: A semantic visual slam towards dynamic environments. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1168–1174. IEEE, 2018.

[216] Raza Yunus, Yanyan Li, and Federico Tombari. Manhattanslam: Robust planar tracking and mapping leveraging mixture of manhattan frames. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6687–6693. IEEE, 2021.

[217] Borut Žalik and Ivana Kolingerová. An incremental construction algorithm for delaunay triangulation using the nearest-point paradigm. *International Journal of Geographical Information Science*, 17(2):119–138, 2003.

[218] Ji Zhang and Sanjiv Singh. LOAM: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, 2014.

[219] Lizhi Zhang, Diansheng Chen, and Weihui Liu. Point-plane slam based on line-based plane segmentation approach. In *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1287–1292. IEEE, 2016.

[220] Teng Zhang, Kanzhi Wu, Jingwei Song, Shoudong Huang, and Gamini Dissanayake. Convergence and consistency analysis for a 3-d invariant-ekf slam. *IEEE Robotics and Automation Letters*, 2(2):733–740, 2017.

[221] Zichao Zhang and Davide Scaramuzza. A tutorial on quantitative trajectory evaluation for visual (-inertial) odometry. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7244–7251. IEEE, 2018.

[222] Zhe Zhao and Xiaoping Chen. Building 3d semantic maps for mobile robots using rgb-d camera. *Intelligent Service Robotics*, 9(4):297–309, 2016.

[223] Lipu Zhou, Shengze Wang, Yu Jincheng, Guoquan Huang, and Michael Kaess. Plc-lislam: Lidar slam with planes, lines and cylinders. *IEEE Robotics and Automation Letters (RA-L)*, May 2022.

[224] Pengxiang Zhu, Patrick Geneva, Wei Ren, and Guoquan Huang. Distributed visual-inertial cooperative localization. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8714–8721. IEEE, 2021.

[225] Pengxiang Zhu and Wei Ren. Fully distributed joint localization and target tracking with mobile robot networks. *IEEE Transactions on Control Systems Technology*, 2020.

[226] Pengxiang Zhu, Yulin Yang, Wei Ren, and Guoquan Huang. Cooperative visual-inertial odometry. In *Proc. of the IEEE International Conference on Robotics and Automation*, pages 13135–13141, Xi'an, China, 2021. IEEE.

[227] Danping Zou, Yuanxin Wu, Ling Pei, Haibin Ling, and Wenxian Yu. Structvio: visual-inertial odometry with structural regularity of man-made environments. *IEEE Transactions on Robotics*, 35(4):999–1013, 2019.

[228] Xingxing Zuo, Patrick Geneva, Woosik Lee, Yong Liu, and Guoquan Huang. LIC-Fusion: Lidar-inertial-camera odometry. In *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, Macau, China, November 2019.

[229] Xingxing Zuo, Patrick Geneva, Yulin Yang, Wenlong Ye, Yong Liu, and Guoquan Huang. Visual-inertial localization with prior lidar map constraints. *IEEE Robotics and Automation Letters (RA-L)*, 2019.

[230] Xingxing Zuo, Yulin Yang, Patrick Geneva, Jiajun Lv, Yong Liu, Guoquan Huang, and Marc Pollefeys. Lic-fusion 2.0: Lidar-inertial-camera odometry with sliding-window plane-feature tracking. In *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Las Vegas, NV, 2020.

# Appendix A

# VISUAL MEASUREMENT MODEL DERIVATIONS

## A.1 Camera Measurement Modeling

Consider a 3D feature is detected from the camera image at time $k$, whose $uv$ measurement (i.e., the corresponding pixel coordinates) on the image plane is given by:

$$\begin{aligned}
\mathbf{z}_{m,k} &= \mathbf{h}(\mathbf{x}_k) + \mathbf{n}_k \\
&= \mathbf{h}_d(\mathbf{z}_{n,k},\ \boldsymbol{\zeta}) + \mathbf{n}_k \\
&= \mathbf{h}_d(\mathbf{h}_p({}^{C_k}\mathbf{p}_f),\ \boldsymbol{\zeta}) + \mathbf{n}_k \\
&= \mathbf{h}_d(\mathbf{h}_p(\mathbf{h}_t({}^{G}\mathbf{p}_f,\ {}^{C_k}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{C_k})),\ \boldsymbol{\zeta}) + \mathbf{n}_k \\
&= \mathbf{h}_d(\mathbf{h}_p(\mathbf{h}_t(\mathbf{h}_r(\boldsymbol{\lambda}, \cdots),\ {}^{C_k}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{C_k})),\ \boldsymbol{\zeta}) + \mathbf{n}_k
\end{aligned}$$

where $\mathbf{n}_k$ is the measurement noise and typically assumed to be zero-mean white Gaussian; $\mathbf{z}_{n,k}$ is the normalized undistorted uv measurement; $\boldsymbol{\zeta}$ is the camera intrinsic parameters such as focal length and distortion parameters; ${}^{C_k}\mathbf{p}_f$ is the feature position in the current camera frame $\{C_k\}$; ${}^{G}\mathbf{p}_f$ is the feature position in the global frame $\{G\}$; $\{{}^{C_k}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{C_k}\}$ denotes the current camera pose (position and orientation) in the global frame (or camera extrinsics); and $\boldsymbol{\lambda}$ is the feature's parameters of different representations (other than position) such as simply a xyz position or an inverse depth with bearing.

In the above expression, we decompose the measurement function into multiple concatenated functions corresponding to different operations, which map the states into the raw uv measurement on the image plane. It should be noted that as we will perform intrinsic calibration along with extrinsic with different feature representations,

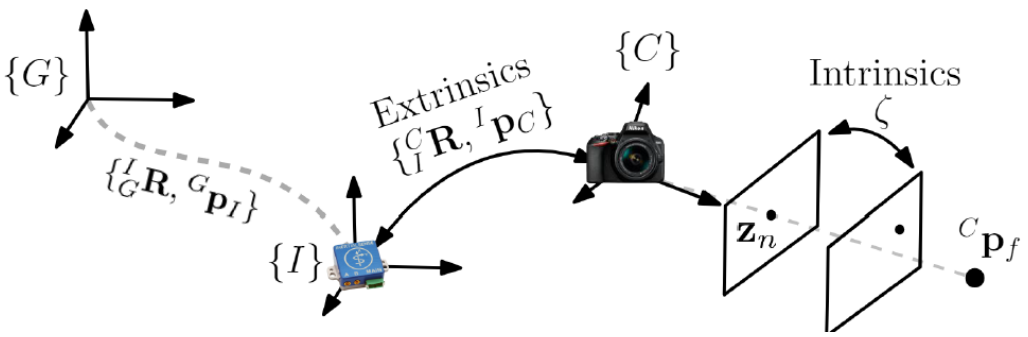



Figure A.1: Overview of the different transformations needed to relate an estimated feature and transform it into the observation on the camera image plane.

the above camera measurement model is general. The high-level description of each function is:

- $\mathbf{z}_k = \mathbf{h}_d(\mathbf{z}_{n,k},\ \boldsymbol{\zeta})$: The distortion function that takes normalized coordinates and maps it into distorted uv coordinates.
- $\mathbf{z}_{n,k} = \mathbf{h}_p({}^{C_k}\mathbf{p}_f)$: The projection function that takes a 3D point in the image and converts it into the normalized uv coordinates.
- ${}^{C_k}\mathbf{p}_f = \mathbf{h}_t({}^{G}\mathbf{p}_f,\ {}^{C_k}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{C_k})$: Transforming a feature's position in the global frame into the current camera frame.
- ${}^{G}\mathbf{p}_f = \mathbf{h}_r(\boldsymbol{\lambda}, \cdots)$: Converting from a feature representation to a 3D feature in the global frame.

## A.2 Jacobian Computation

Given the above nested functions, we can leverage the chainrule to find the total state Jacobian. Since our feature representation function $\mathbf{h}_r(\cdots)$ might also depend on the state, i.e. an anchoring pose, we need to carefully consider its additional derivatives. Consider the following example of our measurement in respect to a state $\mathbf{x}$ Jacobian:

$$\frac{\partial \mathbf{z}_k}{\partial \mathbf{x}} = \frac{\partial \mathbf{h}_d(\cdot)}{\partial \mathbf{z}_{n,k}} \frac{\partial \mathbf{h}_p(\cdot)}{\partial {}^{C_k}\mathbf{p}_f} \frac{\partial \mathbf{h}_t(\cdot)}{\partial \mathbf{x}} + \frac{\partial \mathbf{h}_d(\cdot)}{\partial \mathbf{z}_{n,k}} \frac{\partial \mathbf{h}_p(\cdot)}{\partial {}^{C_k}\mathbf{p}_f} \frac{\partial \mathbf{h}_t(\cdot)}{\partial {}^{G}\mathbf{p}_f} \frac{\partial \mathbf{h}_r(\cdot)}{\partial \mathbf{x}}$$

In the global feature representations the second term will be zero while for the anchored representations it will need to be computed.

## A.3 Distortion Function

### A.3.1 Radial model

To calibrate camera intrinsics, we need to know how to map our normalized coordinates into the raw pixel coordinates on the image plane. We first employ the radial distortion as in OpenCV model:

$$\begin{bmatrix} u \\ v \end{bmatrix} := \mathbf{z}_k = \mathbf{h}_d(\mathbf{z}_{n,k},\ \boldsymbol{\zeta}) = \begin{bmatrix} f_x * x + c_x \\ f_y * y + c_y \end{bmatrix}$$

$$\begin{aligned} \text{where} \quad x &= x_n(1 + k_1 r^2 + k_2 r^4) + 2p_1 x_n y_n + p_2(r^2 + 2x_n^2) \\ y &= y_n(1 + k_1 r^2 + k_2 r^4) + p_1(r^2 + 2y_n^2) + 2p_2 x_n y_n \end{aligned}$$

$$r^2 = x_n^2 + y_n^2$$

where $\mathbf{z}_{n,k} = [x_n \ y_n]^\top$ are the normalized coordinates of the 3D feature and u and v are the distorted image coordinates on the image plane. The following distortion and camera intrinsic (focal length and image center) parameters are involved in the above distortion model, which can be estimated online:

$$\boldsymbol{\zeta} = \begin{bmatrix} f_x & f_y & c_x & c_y & k_1 & k_2 & p_1 & p_2 \end{bmatrix}^\top$$

Note that we do not estimate the higher order (i.e., higher than fourth order) terms as in most offline calibration methods such as Kalibr. To estimate these intrinsic parameters (including the distortion parameters), the following Jacobian for these parameters is needed:

$$\frac{\partial \mathbf{h}_d(\cdot)}{\partial \boldsymbol{\zeta}} = \begin{bmatrix} x & 0 & 1 & 0 & f_x * (x_n r^2) & f_x * (x_n r^4) & f_x * (2x_n y_n) & f_x * (r^2 + 2x_n^2) \\ 0 & y & 0 & 1 & f_y * (y_n r^2) & f_y * (y_n r^4) & f_y * (r^2 + 2y_n^2) & f_y * (2x_n y_n) \end{bmatrix}$$

Similarly, the Jacobian with respect to the normalized coordinates can be obtained as follows:

$$\frac{\partial \mathbf{h}_d(\cdot)}{\partial \mathbf{z}_{n,k}} = \begin{bmatrix} H_{11} & H_{12} \\ H_{21} & H_{22} \end{bmatrix}$$

$$H_{11} = f_x * ((1 + k_1 r^2 + k_2 r^4) + (2k_1 x_n^2 + 4k_2 x_n^2 (x_n^2 + y_n^2)) + 2p_1 y_n + (2p_2 x_n + 4p_2 x_n))$$

$$H_{12} = f_x * (2k_1 x_n y_n + 4k_2 x_n y_n (x_n^2 + y_n^2) + 2p_1 x_n + 2p_2 y_n)$$

$$H_{21} = f_y * (2k_1 x_n y_n + 4k_2 x_n y_n (x_n^2 + y_n^2) + 2p_1 x_n + 2p_2 y_n)$$

$$H_{22} = f_y * ((1 + k_1 r^2 + k_2 r^4) + (2k_1 y_n^2 + 4k_2 y_n^2 (x_n^2 + y_n^2)) + (2p_1 y_n + 4p_1 y_n) + 2p_2 x_n)$$

### A.3.2 Fisheye model

As fisheye or wide-angle lenses are widely used in practice, we here provide mathematical derivations of such distortion model as in OpenCV fisheye.

$$\begin{bmatrix} u \\ v \end{bmatrix} := \mathbf{z}_k = \mathbf{h}_d(\mathbf{z}_{n,k},\ \boldsymbol{\zeta}) = \begin{bmatrix} f_x * x + c_x \\ f_y * y + c_y \end{bmatrix}$$

$$\begin{aligned} \text{where} \quad x &= \frac{x_n}{r} * \theta_d \\ y &= \frac{y_n}{r} * \theta_d \\ \theta_d &= \theta(1 + k_1\theta^2 + k_2\theta^4 + k_3\theta^6 + k_4\theta^8) \\ r^2 &= x_n^2 + y_n^2 \\ \theta &= atan(r) \end{aligned}$$

where $\mathbf{z}_{n,k} = [x_n \ y_n]^\top$ are the normalized coordinates of the 3D feature and u and v are the distorted image coordinates on the image plane.

Clearly, the following distortion intrinsic parameters are used in the above model:

$$\boldsymbol{\zeta} = \begin{bmatrix} f_x & f_y & c_x & c_y & k_1 & k_2 & k_3 & k_4 \end{bmatrix}^\top$$

In analogy to the previous radial distortion case, the following Jacobian for these parameters is needed for intrinsic calibration:

$$\frac{\partial \mathbf{h}_d(\cdot)}{\partial \boldsymbol{\zeta}} = \begin{bmatrix} x_n & 0 & 1 & 0 & f_x * (\frac{x_n}{r}\theta^3) & f_x * (\frac{x_n}{r}\theta^5) & f_x * (\frac{x_n}{r}\theta^7) & f_x * (\frac{x_n}{r}\theta^9) \\ 0 & y_n & 0 & 1 & f_y * (\frac{y_n}{r}\theta^3) & f_y * (\frac{y_n}{r}\theta^5) & f_y * (\frac{y_n}{r}\theta^7) & f_y * (\frac{y_n}{r}\theta^9) \end{bmatrix}$$

Similarly, with the chain rule of differentiation, we can compute the following Jacobian with respect to the normalized coordinates:

$$\frac{\partial \mathbf{h}_d(\cdot)}{\partial \mathbf{z}_{n,k}} = \frac{\partial uv}{\partial xy}\frac{\partial xy}{\partial x_n y_n} + \frac{\partial uv}{\partial xy}\frac{\partial xy}{\partial r}\frac{\partial r}{\partial x_n y_n} + \frac{\partial uv}{\partial xy}\frac{\partial xy}{\partial \theta_d}\frac{\partial \theta_d}{\partial \theta}\frac{\partial \theta}{\partial r}\frac{\partial r}{\partial x_n y_n}$$

$$\begin{aligned}
\text{where} \quad \frac{\partial uv}{\partial xy} &= \begin{bmatrix} f_x & 0 \\ 0 & f_y \end{bmatrix} \\
\frac{\partial xy}{\partial x_n y_n} &= \begin{bmatrix} \theta_d/r & 0 \\ 0 & \theta_d/r \end{bmatrix} \\
\frac{\partial xy}{\partial r} &= \begin{bmatrix} -\frac{x_n}{r^2}\theta_d \\ -\frac{y_n}{r^2}\theta_d \end{bmatrix} \\
\frac{\partial r}{\partial x_n y_n} &= \begin{bmatrix} \frac{x_n}{r} & \frac{y_n}{r} \end{bmatrix} \\
\frac{\partial xy}{\partial \theta_d} &= \begin{bmatrix} \frac{x_n}{r} \\ \frac{y_n}{r} \end{bmatrix} \\
\frac{\partial \theta_d}{\partial \theta} &= \begin{bmatrix} 1 + 3k_1\theta^2 + 5k_2\theta^4 + 7k_3\theta^6 + 9k_4\theta^8 \end{bmatrix} \\
\frac{\partial \theta}{\partial r} &= \begin{bmatrix} \frac{1}{r^2+1} \end{bmatrix}
\end{aligned}$$

### A.4 Perspective Projection Function

The standard pinhole camera model is used to project a 3D point in the camera frame into the normalized image plane (with unit depth):

$$\mathbf{z}_{n,k} = \mathbf{h}_p({}^{C_k}\mathbf{p}_f) = \begin{bmatrix} {}^{C}x/{}^{C}z \\ {}^{C}y/{}^{C}z \end{bmatrix}$$

$$\text{where} \quad {}^{C_k}\mathbf{p}_f = \begin{bmatrix} {}^{C}x \\ {}^{C}y \\ {}^{C}z \end{bmatrix}$$

whose Jacobian matrix is computed as follows:

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

## A.5 Euclidean Transformation

We employ the 6DoF rigid-body Euclidean transformation to transform the 3D feature position in the global frame $\{G\}$ to the current camera frame $\{C_k\}$ based on the current global camera pose:

$$^{C_k}\mathbf{p}_f = \mathbf{h}_t(^G\mathbf{p}_f,\ ^{C_k}_{G}\mathbf{R},\ ^G\mathbf{p}_{C_k}) = ^{C_k}_{G}\mathbf{R}(^G\mathbf{p}_f - ^G\mathbf{p}_{C_k})$$

Note that in visual-inertial navigation systems, we often keep the IMU, instead of camera, state in the state vector. So, we need to further transform the above geometry using the time-invariant IMU-camera extrinsic parameters $\{^C_I\mathbf{R},\ ^C\mathbf{p}_I\}$ as follows:

$$\begin{aligned}
^G\mathbf{p}_{C_k} &= ^G\mathbf{p}_{I_k} + ^G_I\mathbf{R}\,^I\mathbf{p}_{C_k} = ^G\mathbf{p}_{I_k} + ^G_I\mathbf{R}\,^I\mathbf{p}_C \\
^{C_k}_{G}\mathbf{R} &= ^{C_k}_{I}\mathbf{R}\,^{I_k}_{G}\mathbf{R} = ^C_I\mathbf{R}\,^{I_k}_{G}\mathbf{R}
\end{aligned}$$

Substituting these quantities into the equation of $^{C_k}\mathbf{p}_f$ yields:

$$^{C_k}\mathbf{p}_f = ^C_I\mathbf{R}\,^{I_k}_{G}\mathbf{R}(^G\mathbf{p}_f - ^G\mathbf{p}_{I_k}) + ^C\mathbf{p}_I$$

We now can compute the following Jacobian with respect to the pertinent states:

$$\begin{aligned}
\frac{\partial\mathbf{h}_t(\cdot)}{\partial^G\mathbf{p}_f} &= ^C_I\mathbf{R}\,^{I_k}_{G}\mathbf{R} \\
\frac{\partial\mathbf{h}_t(\cdot)}{\partial^{I_k}_{G}\mathbf{R}} &= ^C_I\mathbf{R}\left\lfloor ^{I_k}_{G}\mathbf{R}(^G\mathbf{p}_f - ^G\mathbf{p}_{I_k})\times\right\rfloor \\
\frac{\partial\mathbf{h}_t(\cdot)}{\partial^G\mathbf{p}_{I_k}} &= -^C_I\mathbf{R}\,^{I_k}_{G}\mathbf{R}
\end{aligned}$$

where $\lfloor\mathbf{a}\times\rfloor$ denotes the skew symmetric matrix of a vector $\mathbf{a}$ (see Quaternion TR [186]). Note also that in the above expression (as well as in ensuing derivations), there is a little abuse of notation; that is, the Jacobian with respect to the rotation matrix is not the direct differentiation with respect to the 3x3 rotation matrix, instead with respect to the corresponding 3x1 rotation angle vector. Moreover, if performing online extrinsic calibration, the Jacobian with respect to the IMU-camera extrinsics is needed:

$$\begin{aligned}
\frac{\partial\mathbf{h}_t(\cdot)}{\partial^C_I\mathbf{R}} &= \left\lfloor ^C_I\mathbf{R}\,^{I_k}_{G}\mathbf{R}(^G\mathbf{p}_f - ^G\mathbf{p}_{I_k})\times\right\rfloor \\
\frac{\partial\mathbf{h}_t(\cdot)}{\partial^C\mathbf{p}_I} &= \mathbf{I}_{3\times3}
\end{aligned}$$

### A.6 Point Feature Representations

There are two main parameterizations of a 3D point feature: 3D position (xyz) and inverse depth with bearing. Both of these can either be represented in the global frame or in an anchor frame of reference which adds a dependency on having an "anchor" pose where the feature is observed. To allow for a unified treatment of different feature parameterizations $\boldsymbol{\lambda}$ in our codebase, we derive in detail the generic function ${}^{G}\mathbf{p}_f = \mathbf{f}(\cdot)$ that maps different representations into global position.

#### A.6.1 Global XYZ

As the canonical parameterization, the global position of a 3D point feature is simply given by its xyz coordinates in the global frame of reference:

$$\begin{aligned} {}^{G}\mathbf{p}_f &= \mathbf{f}(\boldsymbol{\lambda}) \\ &= \begin{bmatrix} {}^{G}x \\ {}^{G}y \\ {}^{G}z \end{bmatrix} \\ \text{where} \quad \boldsymbol{\lambda} &= {}^{G}\mathbf{p}_f = \begin{bmatrix} {}^{G}x & {}^{G}y & {}^{G}z \end{bmatrix}^\top \end{aligned}$$

It is clear that the Jacobian with respect to the feature parameters is:

$$\frac{\partial \mathbf{f}(\cdot)}{\partial \boldsymbol{\lambda}} = \mathbf{I}_{3\times 3}$$

#### A.6.2 Global Inverse Depth

The global inverse-depth representation of a 3D point feature is given by (akin to spherical coordinates):

$$\begin{aligned} {}^{G}\mathbf{p}_f &= \mathbf{f}(\boldsymbol{\lambda}) \\ &= \frac{1}{\rho} \begin{bmatrix} \cos(\theta)\sin(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\phi) \end{bmatrix} \\ \text{where} \quad \boldsymbol{\lambda} &= \begin{bmatrix} \theta & \phi & \rho \end{bmatrix}^\top \end{aligned}$$

The Jacobian with respect to the feature parameters can be computed as:

$$\frac{\partial \mathbf{f}(\cdot)}{\partial \boldsymbol{\lambda}} = \begin{bmatrix} -\frac{1}{\rho}\sin(\theta)\sin(\phi) & \frac{1}{\rho}\cos(\theta)\cos(\phi) & -\frac{1}{\rho^2}\cos(\theta)\sin(\phi) \\ \frac{1}{\rho}\cos(\theta)\sin(\phi) & \frac{1}{\rho}\sin(\theta)\cos(\phi) & -\frac{1}{\rho^2}\sin(\theta)\sin(\phi) \\ 0 & -\frac{1}{\rho}\sin(\phi) & -\frac{1}{\rho^2}\cos(\phi) \end{bmatrix}$$

### A.6.3 Anchored XYZ

We can represent a 3D point feature in some "anchor" frame (say some IMU local frame, $\{{}^{I_a}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{I_a}\}$), which would normally be the IMU pose corresponding to the first camera frame where the feature was detected.

$$\begin{aligned} {}^{G}\mathbf{p}_f &= \mathbf{f}(\boldsymbol{\lambda},\ {}^{I_a}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{I_a},\ {}^{C}_{I}\mathbf{R},\ {}^{C}\mathbf{p}_I) \\ &= {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top(\boldsymbol{\lambda} - {}^{C}\mathbf{p}_I) + {}^{G}\mathbf{p}_{I_a} \\ \text{where} \quad \boldsymbol{\lambda} &= {}^{C_a}\mathbf{p}_f = \begin{bmatrix} {}^{C_a}x & {}^{C_a}y & {}^{C_a}z \end{bmatrix}^\top \end{aligned}$$

The Jacobian with respect to the feature state is given by:

$$\frac{\partial \mathbf{f}(\cdot)}{\partial \boldsymbol{\lambda}} = {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top$$

As the anchor pose is involved in this representation, its Jacobians are computed as:

$$\begin{aligned} \frac{\partial \mathbf{f}(\cdot)}{\partial {}^{I_a}_{G}\mathbf{R}} &= -{}^{I_a}_{G}\mathbf{R}^\top \left\lfloor {}^{C}_{I}\mathbf{R}^\top({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I)\times \right\rfloor \\ \frac{\partial \mathbf{f}(\cdot)}{\partial {}^{G}\mathbf{p}_{I_a}} &= \mathbf{I}_{3\times 3} \end{aligned}$$

Moreover, if performing extrinsic calibration, the following Jacobians with respect to the IMU-camera extrinsics are also needed:

$$\begin{aligned} \frac{\partial \mathbf{f}(\cdot)}{\partial {}^{C}_{I}\mathbf{R}} &= -{}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \left\lfloor ({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I)\times \right\rfloor \\ \frac{\partial \mathbf{f}(\cdot)}{\partial {}^{C}\mathbf{p}_I} &= -{}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \end{aligned}$$

### A.6.4 Anchored Inverse Depth

In analogy to the global inverse depth case, we can employ the inverse depth with bearing (akin to spherical coordinates) in the anchor frame, $\{{}^{I_a}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{I_a}\}$, to represent a 3D point feature:

$$\begin{aligned}
{}^{G}\mathbf{p}_f &= \mathbf{f}(\boldsymbol{\lambda},\ {}^{I_a}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{I_a},\ {}^{C}_{I}\mathbf{R},\ {}^{C}\mathbf{p}_I) \\
&= {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I) + {}^{G}\mathbf{p}_{I_a} \\
&= {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \left( \frac{1}{\rho} \begin{bmatrix} \cos(\theta)\sin(\phi) \\ \sin(\theta)\sin(\phi) \\ \cos(\phi) \end{bmatrix} - {}^{C}\mathbf{p}_I \right) + {}^{G}\mathbf{p}_{I_a} \\
\text{where} \quad \boldsymbol{\lambda} &= \begin{bmatrix} \theta & \phi & \rho \end{bmatrix}^\top
\end{aligned}$$

The Jacobian with respect to the feature state is given by:

$$\frac{\partial \mathbf{f}(\cdot)}{\partial \boldsymbol{\lambda}} = {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \begin{bmatrix} -\frac{1}{\rho}\sin(\theta)\sin(\phi) & \frac{1}{\rho}\cos(\theta)\cos(\phi) & -\frac{1}{\rho^2}\cos(\theta)\sin(\phi) \\ \frac{1}{\rho}\cos(\theta)\sin(\phi) & \frac{1}{\rho}\sin(\theta)\cos(\phi) & -\frac{1}{\rho^2}\sin(\theta)\sin(\phi) \\ 0 & -\frac{1}{\rho}\sin(\phi) & -\frac{1}{\rho^2}\cos(\phi) \end{bmatrix}$$

The Jacobians with respect to the anchor pose are:

$$\begin{aligned}
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{I_a}_{G}\mathbf{R}} &= -{}^{I_a}_{G}\mathbf{R}^\top \left\lfloor {}^{C}_{I}\mathbf{R}^\top({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I)\times \right\rfloor \\
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{G}\mathbf{p}_{I_a}} &= \mathbf{I}_{3\times 3}
\end{aligned}$$

The Jacobians with respect to the IMU-camera extrinsics are:

$$\begin{aligned}
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{C}_{I}\mathbf{R}} &= -{}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \left\lfloor ({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I)\times \right\rfloor \\
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{C}\mathbf{p}_I} &= -{}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top
\end{aligned}$$

### A.6.5 Anchored Inverse Depth (MSCKF Version)

Note that a simpler version of inverse depth was used in the original MSCKF paper [139]. This representation does not have the singularity if it is represented in a camera frame the feature was measured.

$$ {}^{G}\mathbf{p}_f = \mathbf{f}(\boldsymbol{\lambda},\ {}^{I_a}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{I_a},\ {}^{C}_{I}\mathbf{R},\ {}^{C}\mathbf{p}_I)$$

$$
\begin{aligned}
&= {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top ({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I) + {}^{G}\mathbf{p}_{I_a} \\
&= {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \left( \frac{1}{\rho} \begin{bmatrix} \alpha \\ \beta \\ 1 \end{bmatrix} - {}^{C}\mathbf{p}_I \right) + {}^{G}\mathbf{p}_{I_a} \\
\text{where} \quad & \boldsymbol{\lambda} = \begin{bmatrix} \alpha & \beta & \rho \end{bmatrix}^\top
\end{aligned}
$$

The Jacobian with respect to the feature state is:

$$
\frac{\partial \mathbf{f}(\cdot)}{\partial \boldsymbol{\lambda}} = {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \begin{bmatrix} \frac{1}{\rho} & 0 & -\frac{1}{\rho^2}\alpha \\ 0 & \frac{1}{\rho} & -\frac{1}{\rho^2}\beta \\ 0 & 0 & -\frac{1}{\rho^2} \end{bmatrix}
$$

The Jacobians with respect to the anchor state are:

$$
\begin{aligned}
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{I_a}_{G}\mathbf{R}} &= -{}^{I_a}_{G}\mathbf{R}^\top \left\lfloor {}^{C}_{I}\mathbf{R}^\top ({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I) \times \right\rfloor \\
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{G}\mathbf{p}_{I_a}} &= \mathbf{I}_{3\times 3}
\end{aligned}
$$

The Jacobians with respect to the IMU-camera extrinsics are:

$$
\begin{aligned}
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{C}_{I}\mathbf{R}} &= -{}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \left\lfloor ({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I) \times \right\rfloor \\
\frac{\partial \mathbf{f}(\cdot)}{\partial {}^{C}\mathbf{p}_I} &= -{}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top
\end{aligned}
$$

### A.6.6 Anchored Inverse Depth (MSCKF Single Depth Version)

This feature representation is based on the MSCKF representation [139], and the the single depth from VINS-Mono [160]. As compared to the implementation in [160], we are careful about how we handle treating of the bearing of the feature. During initialization we initialize a full 3D feature and then follow that by marginalize the bearing portion of it leaving the depth in the state vector. The marginalized bearing is then fixed for all future linearizations.

Then during the update, we perform nullspace projection at every timestep to remove the feature dependence on this bearing. To do so, we need at least *two*

sets of UV measurements to perform this bearing nullspace operation since we lose two dimensions of the feature in the process. We can define the feature measurement function as follows:

$$
\begin{aligned}
{}^{G}\mathbf{p}_f &= \mathbf{f}(\boldsymbol{\lambda},\ {}^{I_a}_{G}\mathbf{R},\ {}^{G}\mathbf{p}_{I_a},\ {}^{C}_{I}\mathbf{R},\ {}^{C}\mathbf{p}_I) \\
&= {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top ({}^{C_a}\mathbf{p}_f - {}^{C}\mathbf{p}_I) + {}^{G}\mathbf{p}_{I_a} \\
&= {}^{I_a}_{G}\mathbf{R}^\top {}^{C}_{I}\mathbf{R}^\top \Big(\frac{1}{\rho}\hat{\mathbf{b}} - {}^{C}\mathbf{p}_I\Big) + {}^{G}\mathbf{p}_{I_a} \\
\text{where} \quad \boldsymbol{\lambda} &= [\rho]
\end{aligned}
$$

In the above case we have defined a bearing $\hat{\mathbf{b}}$ which is the marginalized bearing of the feature after initialization. After collecting two measurements, we can nullspace project to remove the Jacobian in respect to this bearing variable.

# Appendix B

# ASYNCHRONOUS 6 DOF POSE FACTOR JACOBIANS

## B.1 Measurement Noise Jacobian

The measurement covariance is propagated through the following covariance propagation:

$$\boldsymbol{P}_i = \boldsymbol{H}_u \boldsymbol{P}_{1,2} \boldsymbol{H}_u{}^\top \tag{B.1}$$

$$= \begin{bmatrix} \frac{\delta^i_G \tilde{\boldsymbol{\theta}}}{\delta^1_G \tilde{\boldsymbol{\theta}}} & \mathbf{0}_{3\times3} & \frac{\delta^i_G \tilde{\boldsymbol{\theta}}}{\delta^2_G \tilde{\boldsymbol{\theta}}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \frac{\delta^G \tilde{\boldsymbol{p}}_i}{\delta^G \tilde{\boldsymbol{p}}_1} & \mathbf{0}_{3\times3} & \frac{\delta^G \tilde{\boldsymbol{p}}_i}{\delta^G \tilde{\boldsymbol{p}}_2} \end{bmatrix} \begin{bmatrix} \boldsymbol{P}_1 & \mathbf{0}_{6\times6} \\ \mathbf{0}_{6\times6} & \boldsymbol{P}_2 \end{bmatrix} \begin{bmatrix} \frac{\delta^i_G \tilde{\boldsymbol{\theta}}}{\delta^1_G \tilde{\boldsymbol{\theta}}} & \mathbf{0}_{3\times3} & \frac{\delta^i_G \tilde{\boldsymbol{\theta}}}{\delta^2_G \tilde{\boldsymbol{\theta}}} & \mathbf{0}_{3\times3} \\ \mathbf{0}_{3\times3} & \frac{\delta^G \tilde{\boldsymbol{p}}_i}{\delta^G \tilde{\boldsymbol{p}}_1} & \mathbf{0}_{3\times3} & \frac{\delta^G \tilde{\boldsymbol{p}}_i}{\delta^G \tilde{\boldsymbol{p}}_2} \end{bmatrix}^\top \tag{B.2}$$

where $\boldsymbol{P}_{1,2}$ is the joint covariance matrix from the bounding poses, and $\tilde{\boldsymbol{\theta}}$ and $\tilde{\boldsymbol{p}}$ are the error states of each angle and position measurement, respectively. The resulting Jacobian matrix $\boldsymbol{H}_u$ is defined as the following:

$$\boldsymbol{H}_u = \begin{bmatrix} -{}^i_1\hat{\boldsymbol{R}} \left( \mathbf{J_r}(\lambda \operatorname{Log}({}^2_1\hat{\boldsymbol{R}})) \right. & \mathbf{0}_{3\times3} & {}^i_1\hat{\boldsymbol{R}} \, \mathbf{J_r}\Big( -\lambda \operatorname{Log}({}^2_1\hat{\mathbf{R}}^\top)) & \mathbf{0}_{3\times3} \\ \left. \lambda \, \mathbf{J_r}{}^{-1}(\operatorname{Log}({}^2_1\hat{\boldsymbol{R}})) - I \right) & & \lambda \, \mathbf{J_r}{}^{-1}(\operatorname{Log}({}^2_1\hat{\mathbf{R}}^\top)) \Big) & \\ \mathbf{0}_{3\times3} & (1-\lambda)\,\mathbf{I} & \mathbf{0}_{3\times3} & \lambda\,\mathbf{I} \end{bmatrix}$$

## B.2 Model Linearization

We have the following residual:

$$r_V(\mathbf{x}) = \begin{bmatrix} \operatorname{Log}\left( {}^I_B\mathbf{R}^{B_i}_V \mathbf{R}^{B_i}_V \breve{\mathbf{R}}^\top \right) \\ {}^V\mathbf{p}_{I_i} + {}^{I_i}_V\mathbf{R}^\top {}^I\mathbf{p}_B - {}^V\breve{\mathbf{p}}_{B_i} \end{bmatrix} + \mathbf{n}_{pose} \tag{B.3}$$

We can then define the following linearized residual as:

$$\tilde{r}_V(\mathbf{x}) = \frac{\partial \mathbf{h}}{\partial \mathbf{x}_{I_i}} \tilde{\mathbf{x}}_{I_i} + \begin{bmatrix} \frac{\partial \mathbf{h}}{\partial [{}^I_B\delta\boldsymbol{\theta} \;\; {}^I\tilde{\mathbf{p}}_B]} & -\frac{\partial \mathbf{g}}{\partial {}^V t_I} \end{bmatrix} \tilde{\mathbf{x}}_J + \mathbf{n}_{pose} \tag{B.4}$$

Thus, the measurement Jacobians are as follows:

$$\frac{\partial \mathbf{h}}{\partial \mathbf{x}_{I_i}} = \begin{bmatrix} {}^{I}_{B}\mathbf{R}^{\top} & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ -{}^{I_i}_{V}\mathbf{R}^{\top} \lfloor {}^{I}\mathbf{p}_B \times \rfloor & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \end{bmatrix} \tag{B.5}$$

$$\frac{\partial \mathbf{h}}{\partial [{}^{I}_{B}\delta\boldsymbol{\theta} \ {}^{I}\tilde{\mathbf{p}}_B]} = \begin{bmatrix} -{}^{I}_{B}\mathbf{R}^{\top} & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^{I_i}_{V}\mathbf{R}^{\top} \end{bmatrix} \tag{B.6}$$

$$\frac{\partial \mathbf{g}}{\partial {}^{V}t_I} = \frac{\partial \mathbf{g}}{\partial \lambda}\frac{\partial \lambda}{\partial {}^{V}t_I} = \begin{bmatrix} -{}^{B_i}_{B_0}\mathbf{R} \ \mathbf{J}_{\mathbf{r}}\left(\lambda \ {}^{B_1}_{B_0}\boldsymbol{\phi}\right) {}^{B_1}_{B_0}\boldsymbol{\phi} \\ {}^{V}\mathbf{p}_{B_1} - {}^{V}\mathbf{p}_{B_0} \end{bmatrix} \frac{-1}{({}^{V}t_{B_1} - {}^{V}t_{B_0})} \tag{B.7}$$

where we define the following:

$${}^{B_i}_{B_0}\mathbf{R} = \mathrm{Exp}\left(\lambda \ \mathrm{Log}({}^{B_1}_{V}\mathbf{R}\,{}^{B_0}_{V}\mathbf{R}^{\top})\right) \ , \ {}^{B_1}_{B_0}\boldsymbol{\phi} = \mathrm{Log}({}^{B_i}_{B_0}\mathbf{R}) \tag{B.8}$$

# Appendix C

# $\mathbb{SE}(3)$ B-SPLINE STATE TIME DERIVATIVE

A key foundation of our simulator is the use of a $\mathbb{SE}(3)$ B-Spline. The derivations are based on the works [142] and [152]. The key properties we care about are:

- Local control, allowing the system to function online as well as in batch
- $C^2$-continuity to enable inertial predictions and calculations
- Good approximation of minimal torque trajectories
- A parameterization of rigid-body motion devoid of singularities

The key idea is to convert a set of trajectory points into a continuous-time uniform cubic cumulative b-spline. As compared to standard b-spline representations, the cumulative form ensures local continuity which is needed for on-manifold interpolation. We leverage the cubic b-spline to ensure $C^2$-continuity to ensure that we can calculate accelerations at any point along the trajectory.

## C.1 General Form

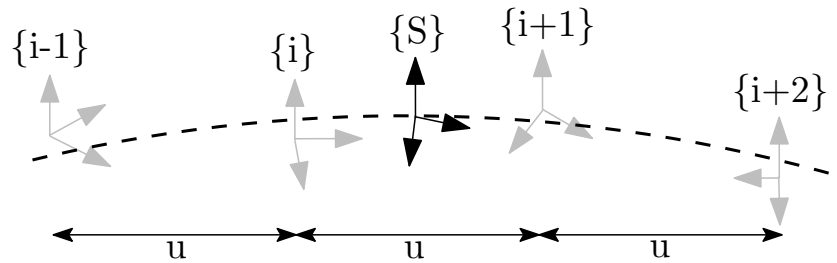



Figure C.1: Illustration of the B-spline interpolation to a pose ${}^{G}_{I_S}\mathbf{T}$ which is bounded by four control poses which are separated by a constant time.

We look at the special case where we have a uniform distribution that can be described as if the variable $t$ describes the distance to any point along the b-spline and each control point is separated by some time $\Delta t$. We can define the time fraction:

$$u(t) = \frac{t - t_i}{t_{i+1} - t_i} = \frac{t - t_i}{\Delta t}, \ u \in [0, 1) \tag{C.1}$$

where $t_i$ and $t_{i+1}$ are the first and second control pose positions along the spline (can also be denoted as $t_i = i\Delta t$ if uniform time between each control pose), respectively. The equations for a uniform cumulative b-spline supported by 4 bounding control poses are:

$$\begin{aligned} {}^{w}_{s}\mathbf{T}(u(t)) = &\exp\Big(\frac{1}{6}(6)\ \log\big({}^{w}_{i-1}\mathbf{T}\big)\Big)\exp\Big(\frac{1}{6}(5 + 3u - 3u^2 + u^3)\ \log\big({}^{w}_{i-1}\mathbf{T}^{-1}\ {}^{w}_{i}\mathbf{T}\big)\Big) \\ &\exp\Big(\frac{1}{6}(1 + 3u + 3u^2 - 2u^3)\ \log\big({}^{w}_{i}\mathbf{T}^{-1}\ {}^{w}_{i+1}\mathbf{T}\big)\Big) \\ &\exp\Big(\frac{1}{6}u^3\ \log\big({}^{w}_{i+1}\mathbf{T}^{-1}\ {}^{w}_{i+2}\mathbf{T}\big)\Big) \end{aligned} \tag{C.2}$$

$$\begin{aligned} = &\exp\Big(\log\big({}^{w}_{i-1}\mathbf{T}\big)\Big)\exp\Big(\frac{1}{6}(5 + 3u - 3u^2 + u^3)\ \log\big({}^{i-1}_{i}\mathbf{T}\big)\Big) \\ &\exp\Big(\frac{1}{6}(1 + 3u + 3u^2 - 2u^3)\ \log\big({}^{i}_{i+1}\mathbf{T}\big)\Big)\exp\Big(\frac{1}{6}u^3\ \log\big({}^{i+1}_{i+2}\mathbf{T}\big)\Big) \end{aligned} \tag{C.3}$$

where $\exp(\cdot)$ is the matrix exponential, $\log(\cdot)$ is the matrix logarithm [5].

## C.2 Time Derivatives

From here, we can look at taking the derivative of our state in respect to time to get these values. Our state is made of control points, so we look at taking the derivative of equation (C.3) in respect to time. First we can simplify equation (C.3) as follows:

$$\begin{aligned} {}^{w}_{s}\mathbf{T}(u(t)) = {}&{}^{w}_{i-1}\mathbf{T}\ \exp\Big(B_0(u(t))\ {}^{i-1}_{i}\mathbf{\Omega}\Big) \\ &\exp\Big(B_1(u(t))\ {}^{i}_{i+1}\mathbf{\Omega}\Big)\exp\Big(B_2(u(t))\ {}^{i+1}_{i+2}\mathbf{\Omega}\Big) \end{aligned} \tag{C.4}$$

$$= {}^{w}_{i-1}\mathbf{T}\ \mathbf{A}_0\ \mathbf{A}_1\ \mathbf{A}_2 \tag{C.5}$$

We can apply the product derivative rule to get the following:

$$\frac{\partial}{\partial t}{}^{w}_{s}\mathbf{T}(u(t)) = {}^{w}_{i-1}\mathbf{T}\Big(\dot{\mathbf{A}}_0\ \mathbf{A}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \dot{\mathbf{A}}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \mathbf{A}_1\ \dot{\mathbf{A}}_2\Big) \tag{C.6}$$

$$\frac{\partial}{\partial t^2}{}^{w}_{s}\mathbf{T}(u(t)) = {}^{w}_{i-1}\mathbf{T}\Big(\ddot{\mathbf{A}}_0\ \mathbf{A}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \ddot{\mathbf{A}}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \mathbf{A}_1\ \ddot{\mathbf{A}}_2 + 2\dot{\mathbf{A}}_0\dot{\mathbf{A}}_1\mathbf{A}_2 + 2\mathbf{A}_0\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2 + 2\dot{\mathbf{A}}_0\mathbf{A}_1\dot{\mathbf{A}}_2\Big) \tag{C.7}$$

To find the derivatives of our matrix exponential we can look at the series definition and take the derivative from there. For the first matrix exponential, we can write the following:

$$\mathbf{A}_0 = \mathbf{I} + B_0(u(t))\ {}^{i-1}_{i}\mathbf{\Omega}^{\wedge} + \frac{1}{2}B_0(u(t))^2\ \left({}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\right)^2 + \frac{1}{6}B_0(u(t))^3\ \left({}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\right)^3 + \cdots \tag{C.8}$$

Taking the derivative in respect to time, we can get the following:

$$\dot{\mathbf{A}}_0 = \dot{B}_0(u(t))\ {}^{i-1}_{i}\mathbf{\Omega}^{\wedge} + \dot{B}_0(u(t))B_0(u(t))\ \left({}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\right)^2 + \frac{1}{2}\dot{B}_0(u(t))B_0(u(t))^2\ \left({}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\right)^3 + \cdots \tag{C.9}$$

$$\dot{\mathbf{A}}_0 = \dot{B}_0(u(t))\ {}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\Big(\mathbf{I} + B_0(u(t))\ \left({}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\right) + \frac{1}{2}\dot{B}_0(u(t))B_0(u(t))^2\ \left({}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\right)^2 + \cdots\Big) \tag{C.10}$$

$$\dot{\mathbf{A}}_0 = \dot{B}_0(u(t))\ {}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\ \mathbf{A}_0 \tag{C.11}$$

where $(\cdot)^{\wedge}$ is the matrix "hat" operation which converts vectors into skew-symmetric matrices [5]. We can see that this is true for all $\mathbf{A}_i$ because the $\Omega$'s control point values do not vary with time. From the above equation, it is simple to apply the product rule to find the second derivative as follows:

$$\ddot{\mathbf{A}}_0 = \dot{B}_0(u(t))\ {}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\ \dot{\mathbf{A}}_0 + \ddot{B}_0(u(t))\ {}^{i-1}_{i}\mathbf{\Omega}^{\wedge}\ \mathbf{A}_0 \tag{C.12}$$

The final piece is to find the derivative of our basis function. We can take the derivative in respect to our variable $u$ and then chain rule this with the derivative of $u$ in respect to time. We can perform that on the matrix form as follows:

$$\tilde{\mathbf{B}}(u(t)) = \frac{1}{3!}\begin{bmatrix}6 & (5+3u-3u^2+u^3) & (1+3u+3u^2-2u^3) & u^3\end{bmatrix}_{1\times 4} \tag{C.13}$$

$$\dot{\mathbf{B}}(u(t)) = \frac{1}{3!}\begin{bmatrix}0 & (3-6u+3u^2) & (3+6u-6u^2) & 3u^2\end{bmatrix}_{1\times 4} \times \frac{1}{\Delta t} \tag{C.14}$$

$$\ddot{\mathbf{B}}(u(t)) = \frac{1}{3!}\begin{bmatrix} 0 & (-6+6u) & (6-12u) & 6u \end{bmatrix}_{1\times 4} \times \frac{1}{\Delta t^2} \tag{C.15}$$

This gives us the following final equation set:

$$ {}^{w}_{s}\mathbf{T}(u(t)) = {}^{w}_{i-1}\mathbf{T}\ \mathbf{A}_0\ \mathbf{A}_1\ \mathbf{A}_2 \tag{C.16}$$

$$ {}^{w}_{s}\dot{\mathbf{T}}(u(t)) = {}^{w}_{i-1}\mathbf{T}\Big(\dot{\mathbf{A}}_0\ \mathbf{A}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \dot{\mathbf{A}}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \mathbf{A}_1\ \dot{\mathbf{A}}_2\Big) \tag{C.17}$$

$$ \begin{aligned} {}^{w}_{s}\ddot{\mathbf{T}}(u(t)) = {}^{w}_{i-1}\mathbf{T}\Big(&\ddot{\mathbf{A}}_0\ \mathbf{A}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \ddot{\mathbf{A}}_1\ \mathbf{A}_2 + \mathbf{A}_0\ \mathbf{A}_1\ \ddot{\mathbf{A}}_2 \\ &+ 2\dot{\mathbf{A}}_0\dot{\mathbf{A}}_1\mathbf{A}_2 + 2\mathbf{A}_0\dot{\mathbf{A}}_1\dot{\mathbf{A}}_2 + 2\dot{\mathbf{A}}_0\mathbf{A}_1\dot{\mathbf{A}}_2\Big) \end{aligned} \tag{C.18}$$

$$ {}^{i-1}_{i}\mathbf{\Omega} = \log\big({}^{w}_{i-1}\mathbf{T}^{-1}\ {}^{w}_{i}\mathbf{T}\big) \tag{C.19}$$

$$ \mathbf{A}_j = \exp\Big(B_j(u(t))\ {}^{i-1+j}_{i+j}\mathbf{\Omega}\Big) \tag{C.20}$$

$$ \dot{\mathbf{A}}_j = \dot{B}_j(u(t))\ {}^{i-1+j}_{i+j}\mathbf{\Omega}^{\wedge}\ \mathbf{A}_j \tag{C.21}$$

$$ \ddot{\mathbf{A}}_j = \dot{B}_j(u(t))\ {}^{i-1+j}_{i+j}\mathbf{\Omega}^{\wedge}\ \dot{\mathbf{A}}_j + \ddot{B}_j(u(t))\ {}^{i-1+j}_{i+j}\mathbf{\Omega}^{\wedge}\ \mathbf{A}_j \tag{C.22}$$

$$ B_0(u(t)) = \frac{1}{3!}\ (5 + 3u - 3u^2 + u^3) \tag{C.23}$$

$$ B_1(u(t)) = \frac{1}{3!}\ (1 + 3u + 3u^2 - 2u^3) \tag{C.24}$$

$$ B_2(u(t)) = \frac{1}{3!}\ (u^3) \tag{C.25}$$

$$ \dot{B}_0(u(t)) = \frac{1}{3!}\ \frac{1}{\Delta t}\ (3 - 6u + 3u^2) \tag{C.26}$$

$$ \dot{B}_1(u(t)) = \frac{1}{3!}\ \frac{1}{\Delta t}\ (3 + 6u - 6u^2) \tag{C.27}$$

$$ \dot{B}_2(u(t)) = \frac{1}{3!}\ \frac{1}{\Delta t}\ (3u^2) \tag{C.28}$$

$$ \ddot{B}_0(u(t)) = \frac{1}{3!}\ \frac{1}{\Delta t^2}\ (-6 + 6u) \tag{C.29}$$

$$ \ddot{B}_1(u(t)) = \frac{1}{3!}\ \frac{1}{\Delta t^2}\ (6 - 12u) \tag{C.30}$$

$$ \ddot{B}_2(u(t)) = \frac{1}{3!}\ \frac{1}{\Delta t^2}\ (6u) \tag{C.31}$$

# Appendix D

# PLANE REPRESENTATION DERIVATIONS

## D.1 Closest Point (CP) Anchor Plane Factor Jacobians

### D.1.1 $\mathbf{H}_A$ Jacobian

$$ {}^{L}\mathbf{\Pi} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}\,{}^{A}d - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{L}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}\,{}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}\,{}^{A}\mathbf{n}\Big) \tag{D.1} $$

We perturb the rotation as follows:

$$ \begin{aligned} {}^{L}\hat{\mathbf{\Pi}} + {}^{L}\tilde{\mathbf{\Pi}} &= {}^{L}_{G}\mathbf{R}\ \left(\Big(\mathbf{I} - \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big)\ {}^{A}_{G}\hat{\mathbf{R}}\right)^{\top} \\ &\quad \left({}^{A}\mathbf{n}\,{}^{A}d - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{L}^{\top}\ \left(\Big(\mathbf{I} - \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big)\ {}^{A}_{G}\hat{\mathbf{R}}\right)^{\top}{}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{A}^{\top}\ \left(\Big(\mathbf{I} - \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big)\ {}^{A}_{G}\hat{\mathbf{R}}\right)^{\top}{}^{A}\mathbf{n}\right) \end{aligned} \tag{D.2} $$

$$ \begin{aligned} &= {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\Big(\mathbf{I} + \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big) \\ &\qquad \left({}^{A}\mathbf{n}\,{}^{A}d - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{L}^{\top}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\Big(\mathbf{I} + \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big){}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\Big(\mathbf{I} + \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big){}^{A}\mathbf{n}\right) \end{aligned} \tag{D.3} $$

$$ \begin{aligned} &= {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\Big(\mathbf{I} + \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big) \\ &\qquad \Bigg({}^{A}\mathbf{n}\,{}^{A}d - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{L}^{\top}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\,{}^{A}\mathbf{n} - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{L}^{\top}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\ \lfloor\delta\boldsymbol{\theta}\times\rfloor\,{}^{A}\mathbf{n} \\ &\qquad\qquad + {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\,{}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top}\ \lfloor\delta\boldsymbol{\theta}\times\rfloor\,{}^{A}\mathbf{n}\Bigg) \end{aligned} \tag{D.4} $$

$$ = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\hat{\mathbf{R}}^{\top} $$

$$\left( {}^{A}\mathbf{n}\,{}^{A}d - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n} \right.$$
$$+ {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n}$$
$$\left. + \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n}\,{}^{A}d - \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} + \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} \right) \tag{D.5}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R} \, {}^{A}_{G}\hat{\mathbf{R}}^\top \left( - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n} \right.$$
$$\left. + \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n}\,{}^{A}d - \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} + \lfloor \delta\boldsymbol{\theta} \times \rfloor \, {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} \right) \tag{D.6}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R} \, {}^{A}_{G}\hat{\mathbf{R}}^\top \left( {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \left\lfloor {}^{A}\mathbf{n} \times \right\rfloor - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \left\lfloor {}^{A}\mathbf{n} \times \right\rfloor \right.$$
$$\left. - \left\lfloor {}^{A}\mathbf{n}\,{}^{A}d \times \right\rfloor + \left\lfloor {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} \times \right\rfloor - \left\lfloor {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} \times \right\rfloor \right) \delta\boldsymbol{\theta} \tag{D.7}$$

Thus, we have the following:

$$\boxed{\begin{aligned} \frac{\partial \, h}{\partial \, {}^{A}_{G}\tilde{\boldsymbol{\delta\theta}}} = {}^{L}_{G}\mathbf{R} \, {}^{A}_{G}\hat{\mathbf{R}}^\top \Big( & {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \left\lfloor {}^{A}\mathbf{n} \times \right\rfloor - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, \left\lfloor {}^{A}\mathbf{n} \times \right\rfloor \\ & - \left\lfloor {}^{A}\mathbf{n}\,{}^{A}d \times \right\rfloor + \left\lfloor {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} \times \right\rfloor - \left\lfloor {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_A^\top \, {}^{A}_{G}\hat{\mathbf{R}}^\top \, {}^{A}\mathbf{n} \times \right\rfloor \Big) \end{aligned}} \tag{D.8}$$

We perturb the position as follows:

$${}^{L}\hat{\mathbf{\Pi}} + {}^{L}\tilde{\mathbf{\Pi}} = {}^{L}_{G}\mathbf{R} \, {}^{A}_{G}\mathbf{R}^\top \Big( {}^{A}\mathbf{n}\,{}^{A}d - {}^{A}\mathbf{n} \, {}^{G}\mathbf{p}_L^\top {}^{A}_{G}\mathbf{R}^\top \, {}^{A}\mathbf{n} + {}^{A}\mathbf{n}({}^{G}\hat{\mathbf{p}}_A + {}^{G}\tilde{\mathbf{p}}_A)^\top \, {}^{A}_{G}\mathbf{R}^\top \, {}^{A}\mathbf{n} \Big) \tag{D.9}$$

$$= {}^{L}_{G}\mathbf{R} \, {}^{A}_{G}\mathbf{R}^\top \Big( {}^{A}\mathbf{n}\,{}^{A}d - {}^{A}\mathbf{n}\,{}^{G}\mathbf{p}_L^\top \, {}^{A}_{G}\mathbf{R}^\top \, {}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\hat{\mathbf{p}}_A^\top \, {}^{A}_{G}\mathbf{R}^\top \, {}^{A}\mathbf{n} + {}^{A}\mathbf{n}\,{}^{G}\tilde{\mathbf{p}}_A^\top \, {}^{A}_{G}\mathbf{R}^\top \, {}^{A}\mathbf{n} \Big) \tag{D.10}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R} \, {}^{A}_{G}\mathbf{R}^\top \Big( {}^{A}\mathbf{n}\,{}^{G}\tilde{\mathbf{p}}_A^\top \, {}^{A}_{G}\mathbf{R}^\top \, {}^{A}\mathbf{n} \Big) \tag{D.11}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}\ ({}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n})^{\top}\Big){}^{G}\tilde{\mathbf{p}}_{A} \tag{D.12}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + \Big({}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\ {}^{A}\mathbf{n}^{\top A}_{G}\mathbf{R}\Big){}^{G}\tilde{\mathbf{p}}_{A} \tag{D.13}$$

Thus, we have the following:

$$\boxed{\frac{\partial\ h}{\partial\ {}^{G}\tilde{\mathbf{p}}_{A}} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\ {}^{A}\mathbf{n}^{\top A}_{G}\mathbf{R}} \tag{D.14}$$

### D.1.2 $\mathbf{H}_L$ Jacobian

$${}^{L}\mathbf{\Pi} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}^{A}d - {}^{A}\mathbf{n}^{G}\mathbf{p}_{L}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n} + {}^{A}\mathbf{n}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\Big) \tag{D.15}$$

We perturb the rotation as follows:

$${}^{L}\hat{\mathbf{\Pi}} + {}^{L}\tilde{\mathbf{\Pi}} = \Big(\mathbf{I} - \lfloor\delta\boldsymbol{\theta}\times\rfloor\Big)\ {}^{L}_{G}\hat{\mathbf{R}}\ \ {}^{A}_{G}\mathbf{R}^{\top}\Big(\cdots\Big) \tag{D.16}$$

$$= {}^{L}\hat{\mathbf{\Pi}} - \lfloor\delta\boldsymbol{\theta}\times\rfloor\ {}^{L}_{G}\hat{\mathbf{R}}\ \ {}^{A}_{G}\mathbf{R}^{\top}\Big(\cdots\Big) \tag{D.17}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + \Big\lfloor\ {}^{L}_{G}\hat{\mathbf{R}}\ \ {}^{A}_{G}\mathbf{R}^{\top}\Big(\cdots\Big)\times\Big\rfloor\,\delta\boldsymbol{\theta} \tag{D.18}$$

Thus, we have the following:

$$\boxed{\frac{\partial\ h}{\partial\ {}^{L}_{G}\tilde{\boldsymbol{\delta\theta}}} = \Big\lfloor\ {}^{L}_{G}\hat{\mathbf{R}}\ \ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}^{A}d - {}^{A}\mathbf{n}^{G}\mathbf{p}_{L}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n} + {}^{A}\mathbf{n}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\Big)\times\Big\rfloor} \tag{D.19}$$

We perturb the position as follows:

$${}^{L}\hat{\mathbf{\Pi}} + {}^{L}\tilde{\mathbf{\Pi}} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}^{A}d - {}^{A}\mathbf{n}({}^{G}\hat{\mathbf{p}}_{L} + {}^{G}\tilde{\mathbf{p}}_{L})^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n} + {}^{A}\mathbf{n}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\Big) \tag{D.20}$$

$$= {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}^{A}d - {}^{A}\mathbf{n}^{G}\hat{\mathbf{p}}_{L}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n} - {}^{A}\mathbf{n}^{G}\tilde{\mathbf{p}}_{L}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n} + {}^{A}\mathbf{n}^{G}\mathbf{p}_{A}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\Big) \tag{D.21}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big(-{}^{A}\mathbf{n}^{G}\tilde{\mathbf{p}}_{L}^{\top}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\Big) \tag{D.22}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big(-{}^{A}\mathbf{n}\ ({}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n})^{\top}\Big){}^{G}\tilde{\mathbf{p}}_{L} \tag{D.23}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + \Big(-{}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top A}\mathbf{n}\ {}^{A}\mathbf{n}^{\top A}_{G}\mathbf{R}\Big){}^{G}\tilde{\mathbf{p}}_{L} \tag{D.24}$$

Thus, we have the following:

$$\boxed{\frac{\partial\, h}{\partial\, {}^{G}\tilde{\mathbf{p}}_L} = -{}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\mathbf{n}\ {}^{A}\mathbf{n}^{\top}{}^{A}_{G}\mathbf{R}} \tag{D.25}$$

### D.1.3 $\mathbf{H}_{nd}$ Jacobian

$${}^{L}\mathbf{\Pi} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}{}^{A}d - {}^{A}\mathbf{n}{}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\mathbf{n} + {}^{A}\mathbf{n}{}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\mathbf{n}\Big) \tag{D.26}$$

We perturb the normal vector ${}^{A}\mathbf{n}$ as follows:

$$\begin{aligned}
{}^{L}\hat{\mathbf{\Pi}} + {}^{L}\tilde{\mathbf{\Pi}} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big(&({}^{A}\hat{\mathbf{n}} + {}^{A}\tilde{\mathbf{n}}){}^{A}d - ({}^{A}\hat{\mathbf{n}} + {}^{A}\tilde{\mathbf{n}}){}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}({}^{A}\hat{\mathbf{n}} + {}^{A}\tilde{\mathbf{n}}) \\
&+ ({}^{A}\hat{\mathbf{n}} + {}^{A}\tilde{\mathbf{n}}){}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}({}^{A}\hat{\mathbf{n}} + {}^{A}\tilde{\mathbf{n}})\Big) \qquad \text{(D.27)}\\
= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big(&{}^{A}\tilde{\mathbf{n}}{}^{A}d - {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\tilde{\mathbf{n}} - {}^{A}\tilde{\mathbf{n}}{}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}} \\
&+ {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\tilde{\mathbf{n}} + {}^{A}\tilde{\mathbf{n}}{}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}}\Big) \qquad \text{(D.28)}\\
= {}^{L}\hat{\mathbf{\Pi}} + {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big(&{}^{A}d - {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top} - {}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}} \\
&+ {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top} + {}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}}\Big){}^{A}\tilde{\mathbf{n}} \qquad \text{(D.29)}
\end{aligned}$$

Thus, we have the following:

$$\boxed{\begin{aligned}
\frac{\partial\, h}{\partial\, {}^{A}\tilde{\mathbf{n}}} &= {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}d - {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top} - {}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}} + {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top} + {}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}}\Big) \\
&= {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}d - {}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}} + {}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\hat{\mathbf{n}}\Big) \\
&\qquad + {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big( - {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top} + {}^{A}\hat{\mathbf{n}}{}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}\Big)
\end{aligned}} \tag{D.30}$$

$$\tag{D.31}$$

We perturb the distance ${}^{A}d$ as follows:

$${}^{L}\hat{\mathbf{\Pi}} + {}^{L}\tilde{\mathbf{\Pi}} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\Big({}^{A}\mathbf{n}({}^{A}\hat{d} + {}^{A}\tilde{d}) - {}^{A}\mathbf{n}{}^{G}\mathbf{p}_L^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\mathbf{n} + {}^{A}\mathbf{n}{}^{G}\mathbf{p}_A^{\top}\ {}^{A}_{G}\mathbf{R}^{\top}{}^{A}\mathbf{n}\Big) \tag{D.32}$$

$$= {}^{L}\hat{\mathbf{\Pi}} + \left({}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\,{}^{A}\mathbf{n}\right){}^{A}\tilde{d} \tag{D.33}$$

$$\tag{D.34}$$

Thus, we have the following:

$$\boxed{\frac{\partial\, h}{\partial\, {}^{A}\tilde{d}} = {}^{L}_{G}\mathbf{R}\ {}^{A}_{G}\mathbf{R}^{\top}\,{}^{A}\mathbf{n}} \tag{D.35}$$

### D.1.4 $\mathbf{H}_{\Pi}$ Jacobian

$$\begin{bmatrix} {}^{A}\mathbf{n} \\ {}^{A}d \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{{}^{A}\Pi(x)^2+{}^{A}\Pi(y)^2+{}^{A}\Pi(z)^2}}\ {}^{A}\mathbf{\Pi} \\ \sqrt{{}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2} \end{bmatrix} \tag{D.36}$$

We can take the element wise derivative to get the following:

$$\frac{\partial\, {}^{A}\mathbf{n}}{\partial\, {}^{A}\widetilde{\mathbf{\Pi}}} = \begin{bmatrix} \frac{{}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} & -\frac{{}^{A}\Pi(x){}^{A}\Pi(y)}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} & -\frac{{}^{A}\Pi(x){}^{A}\Pi(z)}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} \\ -\frac{{}^{A}\Pi(y){}^{A}\Pi(x)}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} & \frac{{}^{A}\Pi(x)^2 + {}^{A}\Pi(z)^2}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} & -\frac{{}^{A}\Pi(y){}^{A}\Pi(z)}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} \\ -\frac{{}^{A}\Pi(z){}^{A}\Pi(x)}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} & -\frac{{}^{A}\Pi(z){}^{A}\Pi(y)}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} & \frac{{}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2}{({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2)^{\frac{3}{2}}} \end{bmatrix}$$

$$\frac{\partial\, {}^{A}\mathbf{n}}{\partial\, {}^{A}\widetilde{\mathbf{\Pi}}} = \frac{1}{\left({}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2\right)^{\frac{3}{2}}} \begin{bmatrix} {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2 & -{}^{A}\Pi(x){}^{A}\Pi(y) & -{}^{A}\Pi(x){}^{A}\Pi(z) \\ -{}^{A}\Pi(y){}^{A}\Pi(x) & {}^{A}\Pi(x)^2 + {}^{A}\Pi(z)^2 & -{}^{A}\Pi(y){}^{A}\Pi(z) \\ -{}^{A}\Pi(z){}^{A}\Pi(x) & -{}^{A}\Pi(z){}^{A}\Pi(y) & {}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 \end{bmatrix}$$

$$= \frac{1}{{}^{A}\hat{d}^3} \left( \begin{bmatrix} {}^{A}\hat{d}^2 & 0 & 0 \\ 0 & {}^{A}\hat{d}^2 & 0 \\ 0 & 0 & {}^{A}\hat{d}^2 \end{bmatrix} - \begin{bmatrix} {}^{A}\Pi(x)^2 & {}^{A}\Pi(x){}^{A}\Pi(y) & {}^{A}\Pi(x){}^{A}\Pi(z) \\ {}^{A}\Pi(y){}^{A}\Pi(x) & {}^{A}\Pi(y)^2 & {}^{A}\Pi(y){}^{A}\Pi(z) \\ {}^{A}\Pi(z){}^{A}\Pi(x) & {}^{A}\Pi(z){}^{A}\Pi(y) & {}^{A}\Pi(z)^2 \end{bmatrix} \right)$$

$$= \frac{1}{{}^{A}\hat{d}} \left( \mathbf{I}_{3\times 3} - {}^{A}\hat{\mathbf{n}}{}^{A}\hat{\mathbf{n}}^{\top} \right)$$

$$\boxed{\frac{\partial\, {}^{A}\mathbf{n}}{\partial\, {}^{A}\widetilde{\mathbf{\Pi}}} = \frac{1}{{}^{A}\hat{d}} \left( \mathbf{I}_{3\times 3} - {}^{A}\hat{\mathbf{n}}\ {}^{A}\hat{\mathbf{n}}^{\top} \right)} \tag{D.37}$$

We can then take the element wise derivatives in respect to the distance scalar.

$$\frac{\partial\, {}^{A}d}{\partial\, {}^{A}\widetilde{\mathbf{\Pi}}} = \frac{1}{\sqrt{{}^{A}\Pi(x)^2 + {}^{A}\Pi(y)^2 + {}^{A}\Pi(z)^2}} \begin{bmatrix} {}^{A}\Pi(x) & {}^{A}\Pi(y) & {}^{A}\Pi(z) \end{bmatrix} \tag{D.38}$$

$$= {}^{A}\hat{\mathbf{n}}^{\top} \tag{D.39}$$

$$\boxed{\frac{\partial\, {}^{A}d}{\partial\, {}^{A}\widetilde{\mathbf{\Pi}}} = {}^{A}\hat{\mathbf{n}}^{\top}} \tag{D.40}$$

## D.2 Quaternion (CP) Anchor Plane Factor Jacobians

### D.2.1 Quaternion Representation

The plane can also be represented by the a quaternion $\bar{q}_\pi$, and the relation with the plane normal direction $\mathbf{n}_\pi$ and plane distance $d_\pi$ as:

$$\bar{q}_\pi = \begin{bmatrix} \mathbf{q}_v \\ q_4 \end{bmatrix} = \frac{1}{\sqrt{1 + d_\pi^2}} \begin{bmatrix} \mathbf{n}_\pi \\ d_\pi \end{bmatrix} \tag{D.41}$$

Therefore, we can use the error states $\delta\boldsymbol{\theta}_\pi$ for quaternion to represent the plane, that is:

$$\bar{q}_\pi = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_\pi \\ 1 \end{bmatrix} \otimes \hat{\bar{q}}_\pi \tag{D.42}$$

Note that the Jacobians w.r.t. the quaternion error states can be written as:

$$\bar{q}_\pi = \hat{\bar{q}}_\pi + \tilde{\bar{q}}_\pi = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_\pi \\ 1 \end{bmatrix} \otimes \hat{\bar{q}}_\pi \tag{D.43}$$

$$\Rightarrow \tilde{\bar{q}}_\pi = \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_\pi \\ 0 \end{bmatrix} \otimes \hat{\bar{q}}_\pi = \begin{bmatrix} \hat{q}_4\mathbf{I}_3 + \lfloor \hat{\mathbf{q}}_v \rfloor & \hat{\mathbf{q}}_v \\ -\hat{\mathbf{q}}_v^\top & \hat{q}_4 \end{bmatrix} \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_\pi \\ 0 \end{bmatrix} \tag{D.44}$$

$$\Rightarrow \begin{bmatrix} \frac{1}{2}\delta\boldsymbol{\theta}_\pi \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ 1 \end{bmatrix} + \tilde{\bar{q}}_\pi \otimes \hat{\bar{q}}_\pi^{-1} = \begin{bmatrix} \mathbf{0}_{3\times 1} \\ 1 \end{bmatrix} + \begin{bmatrix} \hat{q}_4\mathbf{I}_3 - \lfloor \hat{\mathbf{q}}_v \rfloor & -\hat{\mathbf{q}}_v \\ \hat{\mathbf{q}}_v^\top & \hat{q}_4 \end{bmatrix} \tilde{\bar{q}}_\pi \tag{D.45}$$

### D.2.2 Measurement Noise Covariance

Consider a point $\mathbf{p_f}$ on the plane, then we write the plane measurement as the following:

$$\mathbf{z} = \mathbf{h}\left(\bar{q}_\pi, \mathbf{n}_p\right) \tag{D.46}$$

$$= \mathbf{q}_v^\top \left(\mathbf{p_f} + \mathbf{n}_p\right) - q_4 \tag{D.47}$$

We linearize the above equation and get:

$$\tilde{\mathbf{z}} \simeq \mathbf{H}_\pi \delta\boldsymbol{\theta}_\pi + \mathbf{H}_n \mathbf{n}_p \tag{D.48}$$

$$\mathbf{H}_\pi = \frac{\partial \tilde{\mathbf{z}}}{\partial \delta\boldsymbol{\theta}_\pi} = \frac{\partial \tilde{\mathbf{z}}}{\partial \tilde{\bar{q}}_\pi} \frac{\partial \tilde{\bar{q}}_\pi}{\partial \delta\boldsymbol{\theta}_\pi} \tag{D.49}$$

$$\mathbf{H}_n = \frac{\partial \tilde{\mathbf{z}}}{\partial \mathbf{n}_p} \tag{D.50}$$

where:

$$\frac{\partial \tilde{\mathbf{z}}}{\partial \tilde{\bar{q}}_\pi} = \begin{bmatrix} \hat{\mathbf{p}}_\mathbf{f}^\top & -1 \end{bmatrix} \tag{D.51}$$

$$\frac{\partial \tilde{\bar{q}}_\pi}{\partial \delta\boldsymbol{\theta}_\pi} = \frac{1}{2} \begin{bmatrix} \hat{q}_4\mathbf{I}_3 + \lfloor \hat{\mathbf{q}}_v \rfloor \\ -\hat{\mathbf{q}}_v^\top \end{bmatrix} \tag{D.52}$$

$$\frac{\partial \tilde{\mathbf{z}}}{\partial \mathbf{n}_p} = \hat{\mathbf{q}}_v^\top \tag{D.53}$$

We can minimize the difference between each point and the quaternion representation to get the optimal plane parameters. After optimization, we can get the measurement covariance, $\mathbf{R}_\pi$, by looping over all measurements to compute the following:

$$\mathbf{R}_\pi = \left( \sum_i \mathbf{H}_{\pi i}^\top \left( \mathbf{H}_{ni} W_i \mathbf{H}_{ni}^\top \right)^{-1} \mathbf{H}_{\pi i} \right)^{-1} \tag{D.54}$$

### D.2.3 Jacobians for Quaternion Plan Anchor Factor

Having compressed the point cloud into the quaternion plane representation, we can add it to our factor graph. In order to optimize we need the Jacobians of the measurement in respect to the states that it depends on. We define the following frame of references: $\{L\}$ current LiDAR frame, $\{I\}$ current IMU frame, $\{A\}$ anchored IMU frame and $\{L_a\}$ the anchored LiDAR frame. The measurement function of the anchored plane projected into the current local frame can be summarized as follows:

$$\mathbf{z} = \mathbf{h}\left( {}^{La}\bar{q}_\pi, \mathbf{n}_\mathbf{R} \right) \tag{D.55}$$

$$= \mathbf{h}\left( {}^{I}_{G}\mathbf{R}, {}^{G}\mathbf{p}_I, {}^{A}_{G}\mathbf{R}, {}^{G}\mathbf{p}_A, {}^{La}\bar{q}_\pi, \mathbf{n}_\mathbf{R} \right) \tag{D.56}$$

where ${}^{I}_{G}\mathbf{R}$, ${}^{G}\mathbf{p}_I$ is the current IMU pose, ${}^{A}_{G}\mathbf{R}$, ${}^{G}\mathbf{p}_A$ is the current anchored IMU pose, ${}^{La}\bar{q}_\pi$ is the plane in the anchor LiDAR frame, and $\mathbf{n}_\mathbf{R}$ is noise corrupting the quaternion plane measurement, whose covariance is $\mathbf{R}_\pi$. Therefore, we can have:

$$\mathbf{H}_x = \frac{\partial \tilde{\mathbf{z}}}{\partial \tilde{\mathbf{x}}} = \begin{bmatrix} \frac{\partial \delta^L \boldsymbol{\theta}_\pi}{\partial \delta \boldsymbol{\theta}_I} & \frac{\partial \delta^L \boldsymbol{\theta}_\pi}{\partial {}^G\tilde{\mathbf{p}}_I} & \frac{\partial \delta^L \boldsymbol{\theta}_\pi}{\partial \delta \boldsymbol{\theta}_A} & \frac{\partial \delta^L \boldsymbol{\theta}_\pi}{\partial {}^G\tilde{\mathbf{p}}_A} & \frac{\partial \delta^L \boldsymbol{\theta}_\pi}{\partial {}^{La}\boldsymbol{\theta}_\pi} \end{bmatrix} \tag{D.57}$$

where:

$$\begin{bmatrix} {}^L\mathbf{n}_\pi \\ {}^L d_\pi \end{bmatrix} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^I\mathbf{p}_L^\top & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{G}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^G\mathbf{p}_I^\top & 1 \end{bmatrix} \begin{bmatrix} {}^{G}_{A}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^A\mathbf{p}_G^\top & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{L}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^L\mathbf{p}_I^\top & 1 \end{bmatrix} \begin{bmatrix} {}^{La}\mathbf{n}_\pi \\ {}^{La} d_\pi \end{bmatrix} \tag{D.58}$$

We first compute the Jacobians w.r.t. the current IMU pose ${}^{I}_{G}\mathbf{R}$ and ${}^{G}\mathbf{p}_I$ as:

$$\frac{\partial \delta^L \boldsymbol{\theta}_\pi}{\partial \delta \boldsymbol{\theta}_I} = \frac{\partial \delta^L \boldsymbol{\theta}_\pi}{\partial^L \tilde{\bar{q}}_\pi} \frac{\partial^L \tilde{\bar{q}}_\pi}{\partial \begin{bmatrix} {}^L\tilde{\mathbf{n}}_\pi \\ {}^L\tilde{d}_\pi \end{bmatrix}} \frac{\partial \begin{bmatrix} {}^L\tilde{\mathbf{n}}_\pi \\ {}^L\tilde{d}_\pi \end{bmatrix}}{\partial \delta \boldsymbol{\theta}_I} \tag{D.59}$$

$$\frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{G}\tilde{\mathbf{p}}_{I}} = \frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{L}\tilde{\bar{q}}_{\pi}} \frac{\partial {}^{L}\tilde{\bar{q}}_{\pi}}{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}} \frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial {}^{G}\tilde{\mathbf{p}}_{I}} \tag{D.60}$$

The measurement relationship can be described as:

$$\begin{bmatrix} {}^{L}\mathbf{n}_{\pi} \\ {}^{L}d_{\pi} \end{bmatrix} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^{I}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{G}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^{G}\mathbf{p}_{I}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{G}\mathbf{n}_{\pi} \\ {}^{G}d_{\pi} \end{bmatrix} \tag{D.61}$$

where:

$$\frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{L}\tilde{\bar{q}}_{\pi}} = 2 \begin{bmatrix} {}^{L}\hat{q}_{4}\mathbf{I}_{3} - \lfloor {}^{L}\hat{\mathbf{q}}_{v} \rfloor & -{}^{L}\hat{\mathbf{q}}_{v} \end{bmatrix} \tag{D.62}$$

$$\frac{\partial {}^{L}\tilde{\bar{q}}_{\pi}}{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}} = \frac{1}{\left[1 + {}^{L}d_{\pi}^{2}\right]^{\frac{3}{2}}} \begin{bmatrix} \left(1 + {}^{L}d_{\pi}^{2}\right)\mathbf{I}_{3} & -{}^{L}d_{\pi}{}^{L}\mathbf{n}_{\pi} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{D.63}$$

$$\frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial \delta\boldsymbol{\theta}_{I}} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^{I}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} \lfloor {}^{I}_{G}\mathbf{R}{}^{G}\mathbf{n}_{\pi} \rfloor \\ \mathbf{0}_{1\times 3} \end{bmatrix} \tag{D.64}$$

$$\frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial {}^{G}\tilde{\mathbf{p}}_{I}} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times 1} \\ -{}^{I}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3} \\ -{}^{G}\mathbf{n}_{\pi}^{\top} \end{bmatrix} \tag{D.65}$$

We then compute the Jacobians w.r.t. the current IMU pose ${}^{A}_{G}\mathbf{R}$ and ${}^{G}\mathbf{p}_{A}$ as:

$$\frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial \delta\boldsymbol{\theta}_{A}} = \frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{L}\tilde{\bar{q}}_{\pi}} \frac{\partial {}^{L}\tilde{\bar{q}}_{\pi}}{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}} \frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial \delta\boldsymbol{\theta}_{A}} \tag{D.66}$$

$$\frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{G}\tilde{\mathbf{p}}_{A}} = \frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{L}\tilde{\bar{q}}_{\pi}} \frac{\partial {}^{L}\tilde{\bar{q}}_{\pi}}{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}} \frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial {}^{G}\tilde{\mathbf{p}}_{A}} \tag{D.67}$$

The measurement relationship can be described as:

$$\begin{bmatrix} {}^{L}\mathbf{n}_{\pi} \\ {}^{L}d_{\pi} \end{bmatrix} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{I}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{G}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{G}\mathbf{p}_{I}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{G}_{A}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{A}\mathbf{p}_{G}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{A}\mathbf{n}_{\pi} \\ {}^{A}d_{\pi} \end{bmatrix} \tag{D.68}$$

where:

$$\frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{L}\tilde{\bar{q}}_{\pi}} = 2\begin{bmatrix} {}^{L}\hat{q}_{4}\mathbf{I}_{3} - \lfloor {}^{L}\hat{\mathbf{q}}_{v} \rfloor & -{}^{L}\hat{\mathbf{q}}_{v} \end{bmatrix} \tag{D.69}$$

$$\frac{\partial {}^{L}\tilde{\bar{q}}_{\pi}}{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}} = \frac{1}{\left[1 + {}^{L}d_{\pi}^{2}\right]^{\frac{3}{2}}} \begin{bmatrix} \left(1 + {}^{L}d_{\pi}^{2}\right)\mathbf{I}_{3} & -{}^{L}d_{\pi}{}^{L}\mathbf{n}_{\pi} \\ \mathbf{0}_{1\times3} & 1 \end{bmatrix} \tag{D.70}$$

$$\frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial \delta\boldsymbol{\theta}_{A}} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{I}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{G}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{G}\mathbf{p}_{I}^{\top} & 1 \end{bmatrix} \begin{bmatrix} -{}^{G}_{A}\mathbf{R}\lfloor {}^{A}\mathbf{n}_{\pi} \rfloor \\ {}^{A}\mathbf{n}_{\pi}^{\top}\lfloor {}^{A}_{G}\mathbf{R}\,{}^{G}\mathbf{p}_{A} \rfloor \end{bmatrix} \tag{D.71}$$

$$\frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial {}^{G}\tilde{\mathbf{p}}_{A}} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{I}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{G}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{G}\mathbf{p}_{I}^{\top} & 1 \end{bmatrix} \begin{bmatrix} \mathbf{0}_{3} \\ {}^{A}\mathbf{n}_{\pi}^{\top}\,{}^{A}_{G}\mathbf{R} \end{bmatrix} \tag{D.72}$$

Finally, we compute the Jacobians w.r.t. the plane state in the anchored LiDAR frame ${}^{La}\bar{q}_{\pi}$ as:

$$\frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial \delta^{G}\boldsymbol{\theta}_{\pi}} = \frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{L}\tilde{\bar{q}}_{\pi}} \frac{\partial {}^{C}\tilde{\bar{q}}_{\pi}}{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}} \frac{\partial \begin{bmatrix} {}^{L}\tilde{\mathbf{n}}_{\pi} \\ {}^{L}\tilde{d}_{\pi} \end{bmatrix}}{\partial \begin{bmatrix} {}^{La}\tilde{\mathbf{n}}_{\pi} \\ {}^{La}\tilde{d}_{\pi} \end{bmatrix}} \frac{\partial \begin{bmatrix} {}^{La}\tilde{\mathbf{n}}_{\pi} \\ {}^{La}\tilde{d}_{\pi} \end{bmatrix}}{\partial {}^{La}\tilde{\bar{q}}_{\pi}} \frac{\partial {}^{La}\tilde{\bar{q}}_{\pi}}{\partial \delta^{La}\boldsymbol{\theta}_{\pi}} \tag{D.73}$$

The measurement relationship can be described as:

$$\begin{bmatrix} {}^{L}\mathbf{n}_{\pi} \\ {}^{L}d_{\pi} \end{bmatrix} = \begin{bmatrix} {}^{L}_{I}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{I}\mathbf{p}_{L}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{G}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{G}\mathbf{p}_{I}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{G}_{A}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{A}\mathbf{p}_{G}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{I}_{L}\mathbf{R} & \mathbf{0}_{3\times1} \\ -{}^{L}\mathbf{p}_{I}^{\top} & 1 \end{bmatrix} \begin{bmatrix} {}^{La}\mathbf{n}_{\pi} \\ {}^{La}d_{\pi} \end{bmatrix} \tag{D.74}$$

where:

$$\frac{\partial \delta^{L}\boldsymbol{\theta}_{\pi}}{\partial {}^{L}\tilde{\bar{q}}_{\pi}} = 2\begin{bmatrix} {}^{L}\hat{q}_{4}\mathbf{I}_{3} - \lfloor {}^{L}\hat{\mathbf{q}}_{v} \rfloor & -{}^{L}\hat{\mathbf{q}}_{v} \end{bmatrix} \tag{D.75}$$

$$\frac{\partial^{L}\tilde{\tilde{q}}_{\pi}}{\partial\begin{bmatrix}{}^{L}\tilde{\mathbf{n}}_{\pi}\\ {}^{L}\tilde{d}_{\pi}\end{bmatrix}}=\frac{1}{\left[1+{}^{L}d_{\pi}^{2}\right]^{\frac{3}{2}}}\begin{bmatrix}\left(1+{}^{L}d_{\pi}^{2}\right)\mathbf{I}_{3} & -{}^{L}d_{\pi}{}^{L}\mathbf{n}_{\pi}\\ \mathbf{0}_{1\times3} & 1\end{bmatrix} \tag{D.76}$$

$$\frac{\partial\begin{bmatrix}{}^{L}\tilde{\mathbf{n}}_{\pi}\\ {}^{L}\tilde{d}_{\pi}\end{bmatrix}}{\partial\begin{bmatrix}{}^{La}\tilde{\mathbf{n}}_{\pi}\\ {}^{La}\tilde{d}_{\pi}\end{bmatrix}}=\begin{bmatrix}{}_{I}^{L}\mathbf{R} & \mathbf{0}_{3\times1}\\ -{}^{I}\mathbf{p}_{L}^{\top} & 1\end{bmatrix}\begin{bmatrix}{}_{G}^{I}\mathbf{R} & \mathbf{0}_{3\times1}\\ -{}^{G}\mathbf{p}_{I}^{\top} & 1\end{bmatrix}\begin{bmatrix}{}_{A}^{G}\mathbf{R} & \mathbf{0}_{3\times1}\\ -{}^{A}\mathbf{p}_{G}^{\top} & 1\end{bmatrix}\begin{bmatrix}{}_{L}^{I}\mathbf{R} & \mathbf{0}_{3\times1}\\ -{}^{L}\mathbf{p}_{I}^{\top} & 1\end{bmatrix} \tag{D.77}$$

$$\frac{\partial\begin{bmatrix}{}^{La}\tilde{\mathbf{n}}_{\pi}\\ {}^{La}\tilde{d}_{\pi}\end{bmatrix}}{\partial^{La}\tilde{\tilde{q}}_{\pi}}=\frac{1}{\left[{}^{La}\mathbf{q}_{v}^{\top}{}^{La}\mathbf{q}_{v}\right]^{\frac{3}{2}}}\begin{bmatrix}-\left\lfloor{}^{La}\mathbf{q}_{v}\right\rfloor^{2} & \mathbf{0}_{3\times1}\\ -{}^{La}q_{4}{}^{La}\mathbf{q}_{v}^{\top} & {}^{La}\mathbf{q}_{v}^{\top}{}^{La}\mathbf{q}_{v}\end{bmatrix} \tag{D.78}$$

$$\frac{\partial^{La}\tilde{\tilde{q}}_{\pi}}{\partial\delta^{La}\boldsymbol{\theta}_{\pi}}=\frac{1}{2}\begin{bmatrix}{}^{La}\hat{q}_{4}\mathbf{I}_{3}+\left\lfloor{}^{La}\hat{\mathbf{q}}_{v}\right\rfloor\\ -{}^{La}\hat{\mathbf{q}}_{v}^{\top}\end{bmatrix} \tag{D.79}$$

# Appendix E

# COMPLEXITY OF SCHMIDT-KALMAN OPERATIONS

## E.1 State Propagation

### E.1.1 Problem Formulation

During propagation we process a set of inertial measurements to move the state mean and covariance forward in time. To propagate the state covariance matrix forward, we use the discrete-time state transition matrix as follows:

$$\mathbf{x}_{k+1} = \mathbf{f}(\mathbf{x}_k, \mathbf{a}_{m_k} - \mathbf{n}_{a_k}, \boldsymbol{\omega}_{m_k} - \mathbf{n}_{\omega_k}) \tag{E.1}$$

$$\mathbf{P}_{k|k-1} = \begin{bmatrix} \boldsymbol{\Phi}_{k-1}\mathbf{P}_{AA_{k-1|k-1}}\boldsymbol{\Phi}_{k-1}^\top & \boldsymbol{\Phi}_{k-1}\mathbf{P}_{AS_{k-1|k-1}} \\ \mathbf{P}_{SA_{k-1|k-1}}\boldsymbol{\Phi}_{k-1}^\top & \mathbf{P}_{SS_{k-1|k-1}} \end{bmatrix} + \begin{bmatrix} \mathbf{Q}_{k-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{E.2}$$

### E.1.2 Complexity Analysis

We consider that the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size $a$ and the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size $n$. The mean propagation only affects the active state variables, thus we only focus on the state covariance propagation. We have the following algorithm and computational costs for a set of IMU measurements $\mathcal{I}$ that are of some size $q$:

**Algorithm 5** Schmidt-MSCKF Covariance Propagation

| | | cost | times |
|---|---|---|---|
| 1: | **procedure** COV_PROP($\mathbf{P}_{AA_{k-1\|k-1}}$, $\mathbf{P}_{AS_{k-1\|k-1}}$, $\mathcal{I}$) | | |
| 2: | // Set initial values for state transition and noise | | |
| 3: | $\mathbf{\Phi} = \mathbf{I}_{a\times a}$ | $a^2$ | 1 |
| 4: | $\mathbf{Q} = \mathbf{0}_{a\times a}$ | $a^2$ | 1 |
| 5: | // Compound all inertial measurements | | |
| 6: | **for** $\omega_i, \mathbf{a}_i \in \mathcal{I}$ **do** | 1 | $q$ |
| 7: | $\mathbf{\Phi} = \mathbf{\Phi}(i+1,i)\mathbf{\Phi}$ | $a^3 + a^2$ | $q$ |
| 8: | $\mathbf{Q} = \mathbf{\Phi}(i+1,i)\mathbf{Q}\mathbf{\Phi}(i+1,i)^\top + \mathbf{Q}_i$ | $2a^3 + 2a^2$ | $q$ |
| 9: | **end for** | | |
| 10: | // Update active covariance, and Schmidt cross-terms | | |
| 11: | $\mathbf{P}_{AA_{k\|k-1}} = \mathbf{\Phi}\mathbf{P}_{AA_{k-1\|k-1}}\mathbf{\Phi}^\top + \mathbf{Q}$ | $2a^3 + 2a^2$ | 1 |
| 12: | $\mathbf{P}_{AS_{k\|k-1}} = \mathbf{\Phi}\mathbf{P}_{AS_{k-1\|k-1}}$ | $a^2n + an$ | 1 |
| 13: | **end procedure** | | |

It is clear to see that the most expensive computation is the final multiplication of the Schmidt cross-term covariance. We have the following computational cost for propagation if we take the size of the active state to remain constant over time:

$$\boxed{\text{mean propagation} : O(1)} \tag{E.3}$$

$$\boxed{\text{covariance propagation} : O(n)} \tag{E.4}$$

## E.2 Clone Marginalization

### E.2.1 Problem Formulation

When a clone leaves the sliding window, we choose if we should add that clone to the Schmidt state, or marginalize it. Here we look at what happens when we marginalize a clone from the end of our state.

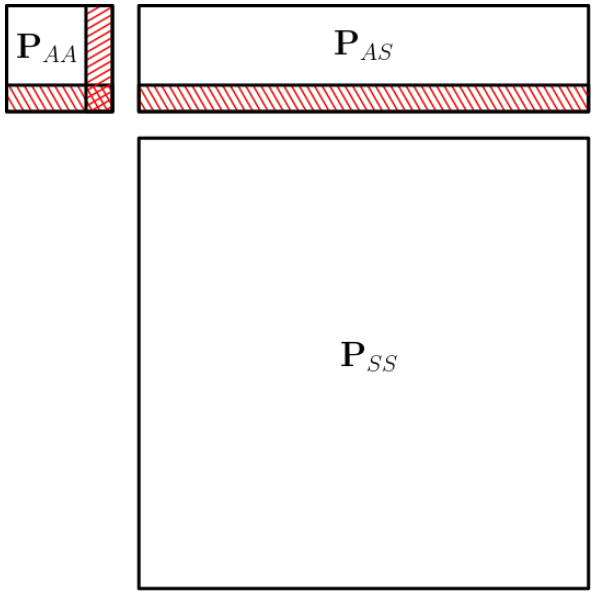



Figure E.1: Illustrate of what is deleted upon marginalization of a clone. The rows shown in red will be deleted after the process is finished.

### E.2.2 Complexity Analysis

We consider that the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size $a$ and the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size $n$. We assume that the clone that will be marginalized is at the *end* of the active variable state/covariance.

**Algorithm 6** Schmidt-MSCKF Covariance Clone Marginalization

| | | cost | times |
|---|---|---|---|
| 1: | **procedure** COVARIANCE_CLONE_MARG($\mathbf{P}_{AA}$, $\mathbf{P}_{AS}$, $\mathbf{P}_{SS}$) | | |
| 2: | $\mathbf{P}_{AA}$.resize($P_{AA}.r-6, P_{AA}.c-6$) | $c_1$ | 1 |
| 3: | $\mathbf{P}_{AS}$.resize($P_{AS}.r-6, P_{AS}.c$) | $c_2$ | 1 |
| 4: | **end procedure** | | |

From the above, we can see that the time to resize the matrices should dominate the actual cost. In practice we don't deallocate memory and simply keep track of the current size of the covariance, thus we don't have a cost to resize the matrix allowing

for constant computation cost. Considering the active state remains constant over time we have the following costs:

$$\boxed{\text{mean clone marginalization} : O(1)} \tag{E.5}$$

$$\boxed{\text{covariance clone marginalization} : O(c)} \tag{E.6}$$

## E.3 Keyframe/Point Augmentation

### E.3.1 Problem Formulation

When a clone leaves the sliding window, we choose if we should add that clone to the Schmidt state, or marginalize it. If we are going to add it to the Schmidt state, we call this operation "keyframe augmentation" as we are adding a new keyframe into our Schmidt state.

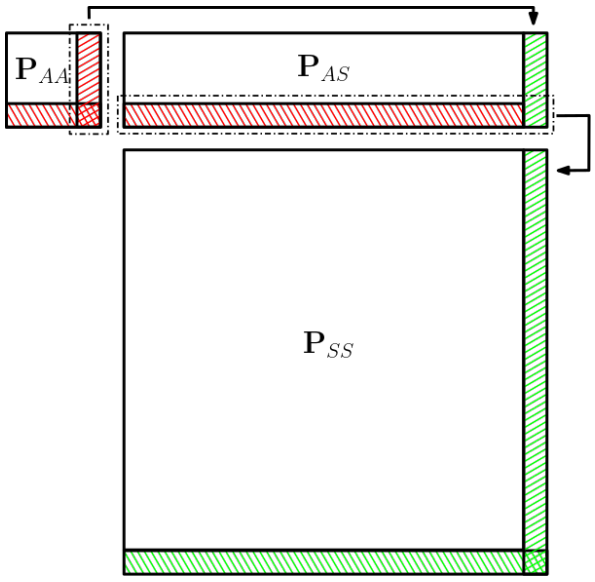



Figure E.2: Illustrate of what is added and deleted upon keyframe augmentation. The rows shown in red will be deleted after the process is finished, while the rows shown in green have been added by copying the cross-terms from the $\mathbf{P}_{AA}$ and $\mathbf{P}_{AS}$ matrices.

### E.3.2 Complexity Analysis

We consider that the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size $a$ and the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size $n$. For the mean, we simply need to append the new $6 \times 1$ (or $3 \times 1$ for a point) to the end of our Schmidt state vector and remove it from our active state vector. Thus, we focus on how to re-order the covariance such that it has the new Schmidt variable at the end of it. We assume that the clone that will be moved to the Schmidt state is at the *end* of the active variable state/covariance.

**Algorithm 7** Schmidt-MSCKF Covariance Keyframe Augmentation

| | | cost | times |
|---|---|---|---|
| 1: | **procedure** COVARIANCE_KEY_AUG($\mathbf{P}_{AA}$, $\mathbf{P}_{AS}$, $\mathbf{P}_{SS}$) | | |
| 2: | // Resize our covariance matrix | | |
| 3: | $\mathbf{P}_{AS}$.resize($P_{AS}.r, P_{AS}.c$+6) | $c_1$ | 1 |
| 4: | $\mathbf{P}_{SS}$.resize($P_{SS}.r$+6, $P_{SS}.c$+6) | $c_2$ | 1 |
| 5: | // Copy from active state to cross-terms | | |
| 6: | $\mathbf{P}_{AS}(0, P_{AS}.c-6, P_{AS}.r, 6) = \mathbf{P}_{AA}(0, P_{AA}.c-6, P_{AA}.r, 6)$ | $6a$ | 1 |
| 7: | // Copy from cross-terms to Schmidt | | |
| 8: | $\mathbf{P}_{SS}(P_{SS}.r-6, 0, 6, P_{SS}.c) = \mathbf{P}_{AS}(P_{AS}.r-6, 0, 6, P_{AS}.c)$ | $6n$ | 1 |
| 9: | $\mathbf{P}_{SS}(0, P_{SS}.c-6, P_{SS}.r, 6) = \mathbf{P}_{AS}(P_{AS}.r-6, 0, 6, P_{AS}.c)^\top$ | $6n$ | 1 |
| 10: | // Finally, reduce size of covariances | | |
| 11: | $\mathbf{P}_{AA}$.resize($P_{AA}.r-6, P_{AA}.c-6$) | $c_3$ | 1 |
| 12: | $\mathbf{P}_{AS}$.resize($P_{AS}.r-6, P_{AS}.c$) | $c_4$ | 1 |
| 13: | **end procedure** | | |

From the above, we can see that the time to resize the matrices should dominate the actual cost of copying the covariance elements. In practice we preallocate memory and simply keep track of the current size of the covariance, thus we don't have a cost to resize the matrix. Considering the active state remains constant over time we have

the following costs:

$$\boxed{\begin{array}{c}\text{mean augmentation} : O(1)\end{array}} \tag{E.7}$$

$$\boxed{\begin{array}{c}\text{covariance augmentation} : O(n+c)\end{array}} \tag{E.8}$$

## E.4 EKF Update

### E.4.1 Problem Formulation

Having propagated the state, we can update the state estimate means and covariance as follows:

$$\hat{\mathbf{x}}_{A_{k|k}} = \hat{\mathbf{x}}_{A_{k|k-1}} + \mathbf{K}_{A_k}\tilde{\mathbf{z}}'_k \tag{E.9}$$

$$\hat{\mathbf{x}}_{S_{k|k}} = \hat{\mathbf{x}}_{S_{k|k-1}} \tag{E.10}$$

With the Schmidt Kalman gain $\mathbf{K}_{S_k} = \mathbf{0}$, we immediately have the covariance update as follows:

$$\mathbf{P}_{k|k} = \mathbf{P}_{k|k-1} - \begin{bmatrix} \mathbf{K}_{A_k}\mathbf{S}_k\mathbf{K}_{A_k}^\top & \mathbf{K}_{A_k}\mathbf{H}'_k \begin{bmatrix}\mathbf{P}_{AS_{k|k-1}} \\ \mathbf{P}_{SS_{k|k-1}}\end{bmatrix} \\ \begin{bmatrix}\mathbf{P}_{AS_{k|k-1}} \\ \mathbf{P}_{SS_{k|k-1}}\end{bmatrix}^\top \mathbf{H}'^\top_k \mathbf{K}_{A_k}^\top & \mathbf{0} \end{bmatrix} \tag{E.11}$$

$$= \mathbf{P}_{k|k-1} - \begin{bmatrix} \mathbf{L}_{A_k}\mathbf{S}_k^{-1}\mathbf{L}_{A_k}^\top & \mathbf{L}_{A_k}\mathbf{S}_k^{-1}\mathbf{L}_{S_k}^\top \\ \mathbf{L}_{S_k}\mathbf{S}_k^{-1}\mathbf{L}_{A_k}^\top & \mathbf{0} \end{bmatrix} \tag{E.12}$$

where the standard Kalman gain can be defined as:

$$\begin{aligned}\mathbf{K}_k = \begin{bmatrix}\mathbf{K}_{A_k} \\ \mathbf{K}_{S_k}\end{bmatrix} &= \begin{bmatrix}\mathbf{P}_{AA_{k|k-1}}\mathbf{H}_{A_k}^\top + \mathbf{P}_{AS_{k|k-1}}\mathbf{H}_{S_k}^\top \\ \mathbf{P}_{SA_{k|k-1}}\mathbf{H}_{A_k}^\top + \mathbf{P}_{SS_{k|k-1}}\mathbf{H}_{S_k}^\top\end{bmatrix}\mathbf{S}_k^{-1} \\ &=: \begin{bmatrix}\mathbf{L}_{A_k} \\ \mathbf{L}_{S_k}\end{bmatrix}\mathbf{S}_k^{-1}\end{aligned} \tag{E.13}$$

where:

$$\mathbf{S}_k = \mathbf{H}'_k\mathbf{P}_{k|k-1}\mathbf{H}'^\top_k + \mathbf{R}'_k \tag{E.14}$$

### E.4.2 Complexity Analysis

We consider the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size $a$, the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size $n$, and the size of the measurement residuals to be of size $q$. We define the set of variables that the update involves as $\mathcal{K}$ which is sparse and has $|\mathcal{K}|$ non-zero elements (i.e. $\mathcal{K} \ll n$).

For the state mean during update, once the Kalman gain $\mathbf{K}_{A_k}$ is computed, it can be clearly seen that it is only of the complexity of the active state and the number of measurements contained in $\tilde{\mathbf{z}}'_k$. This Kalman gain $\mathbf{K}_{A_k}$ will be found through the computation of the updated covariance, and is the most expensive computation during the mean update.

**Algorithm 8** Schmidt-MSCKF Covariance Update

| | | cost | times |
|---|---|---|---|
| 1: | **procedure** COV_UPDATE($\mathbf{P}_{AA}$, $\mathbf{P}_{AS}$, $\mathbf{P}_{SS}$, $\mathbf{H}_k$) | | |
| 2: | // First, compute the sub-matrices $\mathbf{L}_{A_k}$ and $\mathbf{L}_{S_k}$ | | |
| 3: | $\mathbf{L}_{A_k} = \mathbf{0}_{a \times q}$ | $aq$ | 1 |
| 4: | $\mathbf{L}_{S_k} = \mathbf{0}_{n \times q}$ | $nq$ | 1 |
| 5: | // Loop through all the sparse $\mathbf{H}_k$ for each state variable scalar (whose *id* is its location in the covariance matrix) | | |
| 6: | **for all** active state variables **do** | 1 | $a$ |
| 7: | $\mathbf{L}_{A_k}(id, 0, 1, q) = \sum_{k \in \mathcal{K}} P_{ik} \mathbf{H}_k^\top$ | $2\lvert\mathcal{K}\rvert q$ | $a$ |
| 8: | **end for** | | |
| 9: | **for all** schmidt state variables **do** | 1 | $n$ |
| 10: | $\mathbf{L}_{S_k}(id, 0, 1, q) = \sum_{k \in \mathcal{K}} P_{ik} \mathbf{H}_k^\top$ | $2\lvert\mathcal{K}\rvert q$ | $n$ |
| 11: | **end for** | | |
| 12: | // Compute the smaller covariance matrix by *element*, only involving the variables that the Jacobian $\mathbf{H}_k$ involves. | | |
| 13: | $\mathbf{H}\mathbf{P}\mathbf{H}^\top = \sum_{(i,j) \in \mathcal{K} \times \mathcal{K}} \mathbf{H}_i P_{ij} \mathbf{H}_j^\top$ | $2\lvert\mathcal{K}\rvert^2 q^2$ | 1 |
| 14: | // Compute $\mathbf{S}_k$ and its inverse | | |
| 15: | $\mathbf{S}_k = \mathbf{H}\mathbf{P}\mathbf{H}^\top + \mathbf{R}'_k$ | $2q^2$ | 1 |
| 16: | $\mathbf{S}_k^{-1} = \text{inv}(\mathbf{S}_k)$ | $q^3 + q^2$ | 1 |
| 17: | // Finally, do the covariance update | | |
| 18: | $\mathbf{P}_{AA} = \mathbf{P}_{AA} - \mathbf{L}_{A_k} \mathbf{S}_k^{-1} \mathbf{L}_{A_k}^\top$ | $aq^2 + a^2 q + 2a^2$ | 1 |
| 19: | $\mathbf{P}_{AS} = \mathbf{P}_{AS} - \mathbf{L}_{A_k} \mathbf{S}_k^{-1} \mathbf{L}_{S_k}^\top$ | $aq^2 + aqn + 2an$ | 1 |
| 20: | **end procedure** | | |

It is important to note that the size non-zero entries in the sparse Jacobian $\mathbf{H}_k$ which involves $\mathcal{K}$ elements (which we have defined as size $|\mathcal{K}|$), directly impacts the

complexity of the update. To ensure that update is linear in time, we enforce that at each clone time (i.e., image time) we only match to a single keyframe. This ensures that the maximum keyframes that we will ever match to is the size of the sliding window (i.e., no more than $a$ in size). The impact this design decision has on the number of non-zero elements in $\mathbf{H}_k$ is that it is never more than the number of state elements, thus we can consider it constant in terms of big-O. Thus, we have the following results:

$$\boxed{\text{mean update} : O(n)} \tag{E.15}$$

$$\boxed{\text{covariance update} : O(n)} \tag{E.16}$$

## E.5 Dynamic Schmidt'ing

### E.5.1 Problem Formulation

Dynamic Schmidt'ing is the problem of being able to un-Schmidt or Schmidt a variable during online operation. This can be arbitrarily done online without hurting consistency. Specifically Section E.3 can be followed directly to Schmidt a variable, or in reverse to un-Schmidt one. If there are many variables that are being simultaneously un-Schmidt'ed and Schmidt'ed we can further improve performance by reducing the number of matrix operations needed. Specifically, the resize operations in Algorithm 9 can be completely avoided by performing an in-place switching of two variables that are of equal error state dimension (which is common).

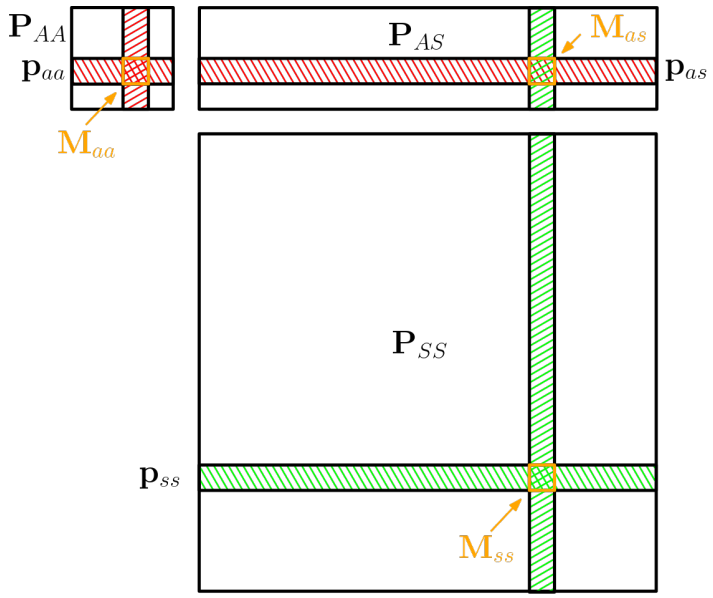



Figure E.3: Illustrate of how to perform in-place switching of two variables.

### E.5.2 Complexity Analysis

We consider that the current active state variables $\mathbf{x}_{A_{k|k}}$ to have an error state of size $a$ and the Schmidt state variables $\mathbf{x}_{S_{k|k}}$ to have an error state of size $n$. We have two variables we wish to switch places, one in the $\mathbf{x}_{A_{k|k}}$ and the other in $\mathbf{x}_{S_{k|k}}$. Both are the same error state size, consider it is a pose and thus is a $6 \times 1$ ($3 \times 1$ for a point) error state. For the mean, we simply swap the two from each state vector. Thus, we focus on how to re-order the covariance such that they have switched places.

**Algorithm 9** Schmidt-MSCKF In-Place Dynamic Schmidt'ing

| | | cost | times |
|---|---|---|---|
| 1: | **procedure** COV_INPLACE_SWAP($\mathbf{P}_{AA}$, $\mathbf{P}_{AS}$, $\mathbf{P}_{SS}$) | | |
| 2: | // Store temp variables | | |
| 3: | $\mathbf{p}_{aa} = \mathbf{P}_{AA}(a, 0, 6, P_{AA}.c)$ | $6a + c_1$ | 1 |
| 4: | $\mathbf{p}_{as} = \mathbf{P}_{AS}(a, 0, 6, P_{SS}.c)$ | $6n + c_2$ | 1 |
| 5: | $\mathbf{M}_{aa} = \mathbf{P}_{AA}(a, a, 6, 6)$ | $36 + c_3$ | 1 |
| 6: | $\mathbf{M}_{as} = \mathbf{P}_{AS}(a, s, 6, 6)$ | $36 + c_4$ | 1 |
| 7: | $\mathbf{M}_{ss} = \mathbf{P}_{SS}(s, s, 6, 6)$ | $36 + c_5$ | 1 |
| 8: | // Copy from Schmidt to active state | | |
| 9: | $\mathbf{P}_{AA}(a, 0, 6, P_{AA}.c) = \mathbf{P}_{AS}(0, s, P_{AA}.r, 6)$ | $6a$ | 1 |
| 10: | $\mathbf{P}_{AA}(0, a, P_{AA}.r, 6) = \mathbf{P}_{AS}(0, s, P_{AA}.r, 6)^\top$ | $6a$ | 1 |
| 11: | $\mathbf{P}_{AS}(a, 0, 0, P_{AS}.c) = \mathbf{P}_{SS}(s, 0, 6, P_{SS}.c)$ | $6n$ | 1 |
| 12: | // Copy from temp active to Schmidt | | |
| 13: | $\mathbf{P}_{SS}(s, 0, 6, P_{SS}.c) = \mathbf{p}_{as}$ | $6n$ | 1 |
| 14: | $\mathbf{P}_{SS}(0, s, P_{SS}.r, 6) = \mathbf{p}_{as}^\top$ | $6n$ | 1 |
| 15: | $\mathbf{P}_{AS}(0, s, P_{AA}.r, 6) = \mathbf{p}_{aa}^\top$ | $6a$ | 1 |
| 16: | // Finally, flip the marginal covariance ordering | | |
| 17: | $\mathbf{P}_{AA}(a, a, 6, 6) = \mathbf{M}_{ss}$ | 36 | 1 |
| 18: | $\mathbf{P}_{AS}(a, s, 6, 6) = \mathbf{M}_{as}^\top$ | 36 | 1 |
| 19: | $\mathbf{P}_{SS}(s, s, 6, 6) = \mathbf{M}_{aa}$ | 36 | 1 |
| 20: | **end procedure** | | |

where we have defined $a$ and $s$ as the index into the covariance of the originally active and the Schmidt'ed variables, respectively. It can be seen that this operation only requires a small memory allocation for the temporary variables. In practice these can be pre-allocated to save computation, but are significantly smaller than the matrix resize operations required in Algorithm 9. Considering the active state remains constant over

time we have the following costs:

$$\boxed{\text{mean switching} : O(1)} \tag{E.17}$$

$$\boxed{\text{covariance switching} : O(n)} \tag{E.18}$$

# Appendix F

# SEVIS VARIANTS

## F.1 CI-EKF Estimator

Another alternative method which provides consistent state estimation is covariance intersection (CI) [86, 87], which only tracks the marginal covariances of the states. For example, we can track the full dense covariance for the active state, $\mathbf{x}_A$, as $\mathbf{P}_{AA_k}$, along with the marginal covariance $\mathbf{P}^{(c)}_{MM_k}$ for each map feature or keyframe pose:

$$\mathbf{Diag}\left(\frac{1}{\omega_0}\mathbf{P}_{AA_k}, \frac{1}{\omega_1}\mathbf{P}^{(1)}_{MM_k}, \cdots, \frac{1}{\omega_C}\mathbf{P}^{(C)}_{MM_k}\right) \geq \mathbf{P}_k \tag{F.1}$$

The left portion of the above equation is the CI covariance zero off-diagonal elements and is conservative, and thus *consistent*, when compared to the true covariance $\mathbf{P}_k$. The weights $\omega_c > 0$ and $\sum \omega_c = 1$, for $c \in \{0, 1..C\}$, can be found optimally or found empirically [87]. Substituting Eq. (F.1) into the standard EKF equations, Eq. (4.6)-(4.7), and only selecting the portion that updates the active state $\mathbf{x}_A$ yields:

$$\hat{\mathbf{x}}^{\oplus}_{A_k} = \hat{\mathbf{x}}_{A_k} + \frac{1}{\omega_0}\mathbf{P}_{AA_k}\mathbf{H}^{\top}_{A_k}\mathbf{S}^{-1}_k\ \mathbf{r} \tag{F.2}$$

$$\mathbf{P}^{\oplus}_{AA_k} = \frac{1}{\omega_0}\mathbf{P}_{AA_k} - \frac{1}{\omega_0^2}\mathbf{P}_{AA_k}\mathbf{H}^{\top}_{A_k}\mathbf{S}^{-1}_k\mathbf{H}_{A_k}\mathbf{P}_{AA_k} \tag{F.3}$$

$$\mathbf{S}_k = \frac{1}{\omega_0}\mathbf{H}_{A_k}\mathbf{P}_{AA_k}\mathbf{H}^{\top}_{A_k} + \mathbf{R} + \sum_{c\in\{1..C\}}\left(\frac{1}{\omega_c}\mathbf{H}^{(c)}_{M_k}\mathbf{P}^{(c)}_{MM_k}\mathbf{H}^{(c)\ \top}_{M_k}\right) \tag{F.4}$$

This process is constant $O(1)$ in terms of computational cost since we leveraged fixed weights and only update the active states, with a memory requirement of $O(9m)$. Details on how to perform delayed state initialization using this method can be found Appendix G.

**Remark:** While this method is conservative in nature, in general the CI process will inflate the covariance of all marginal covariances by a large amount since it will always bound the fused distributions. As shown in our later simulations, Section 4.6.3, this causes the system to become very under-confident (e.g., NEES goes towards zero) impacting the achievable accuracy of the hybrid system and the ability to robustly reject outliers.

### F.2 Relative Marginal Covariance Recovery

Each keyframe is connected with a relative odometry factor based on the frontend relative clone estimates. These factors thus contain the visual and inertial information used to update the frontend state, and we drop the correlations between the frontend estimates and backend relative measurements. To construct a relative, the two global poses from the last added keyframe, $\mathbf{x}_{T_1}$, and to-be-added keyframe, $\mathbf{x}_{I_N} \in \mathbf{x}_C$, need to be transformed into $\{I_N\}$ relative frame.

$$\begin{bmatrix} {}^{I_N}_{I_1}\mathbf{R} \\ {}^{I_2}\mathbf{p}_{I_1} \end{bmatrix} = \begin{bmatrix} {}^{I_N}_{G}\mathbf{R}\,{}^{I_1}_{G}\mathbf{R}^\top \\ {}^{I_N}_{G}\mathbf{R}({}^{G}\mathbf{p}_{I_N} - {}^{G}\mathbf{p}_{I_1}) \end{bmatrix} \tag{F.5}$$

We now wish to compute the uncertainty of this relative transformation. This can be done by propagating the uncertainty from the global frame into the relative:

$$\mathbf{P}_{r,1N} = \mathbf{H}_{1N} \begin{bmatrix} \mathbf{P}_{11} & \mathbf{P}_{1N} \\ \mathbf{P}_{1N}^\top & \mathbf{P}_{NN} \end{bmatrix} \mathbf{H}_{1N}^\top \tag{F.6}$$

where $\mathbf{P}_{11}$, $\mathbf{P}_{1N}$, and $\mathbf{P}_{NN}$ are the frontend covariance of the global clone pose, and $\mathbf{H}_{1N}$ is the same as in Eq. (4.40).

In a naive implementation, the use of Eq. (F.6) is not possible since typically only the marginal keyframe covariances $\mathbf{P}_{11}$ and $\mathbf{P}_{NN}$ are known at keyframe creation. These by themselves are *insufficient* and if $\mathbf{P}_{1N}$ is naively set to zero, then the resulting measurement uncertainty will be highly *underconfident* due to the highly correlated nature of odometry poses within small time horizons. To address this, we modify the state, Eq. (5.1), of the frontend as follows:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_A^\top & \mathbf{x}_{T_1}^\top & \mathbf{x}_M^\top \end{bmatrix}^\top \tag{F.7}$$

where $\mathbf{x}_{T_1}$ is the previously selected historical keyframe clone for insertion into the backend. When the next new keyframe is selected to be added, we first can recover the relative uncertainty, Eq. (F.6), will full correlations since the previous keyframe is still in the state. After, we can marginalize the old keyframe state, $\mathbf{x}_{T_1}$, and retain the newly selected keyframe in the frontend state for future use.

### F.3 Decoupled Estimation

An alternative to passing data back to the frontend is to fully separate the secondary thread and only pass information from the frontend odometry to the backend [92, 159, 160]. Since the backend only maintains a set of keyframe poses, the frontend pose is used to recover a pose at every timestep. For example, we can compute the following:

$$
{}^{I_k}_{K}\mathbf{R}_{vio} = {}^{I_k}_{G}\mathbf{R}_{vio}{}^{K}_{G}\mathbf{R}^{\top}_{vio} \tag{F.8}
$$

$$
{}^{K}\mathbf{p}_{I_k,vio} = {}^{K}_{G}\mathbf{R}_{vio}({}^{G}\mathbf{p}_{I_k,vio} - {}^{G}\mathbf{p}_{K,vio}) \tag{F.9}
$$

This relative transform should be accurate since it is normally over a short time period. We can then append the optimized pose to recover the decoupled optimized pose:

$$
{}^{I_k}_{G}\mathbf{R}_{opt} = {}^{I_k}_{K}\mathbf{R}_{vio}{}^{K}_{G}\mathbf{R}_{opt} \tag{F.10}
$$

$$
{}^{G}\mathbf{p}_{I_k,opt} = {}^{G}\mathbf{p}_{K,opt} + {}^{K}_{G}\mathbf{R}^{\top}_{opt}\,{}^{K}\mathbf{p}_{I_k,vio} \tag{F.11}
$$

Next we wish to recover the uncertainty of this estimate pose. Unfortunately, we cannot directly recover the covariance since correlations between the frontend and backend keyframes are not tracked (keyframes are inserted at a much lower frequency than the required frontend pose frequency). We can recover an approximate as follows:

$$
\mathbf{P}_{r,GI} \simeq \mathbf{H}_{ro}\begin{bmatrix}\mathbf{P}_{rel} & \mathbf{0}_6 \\ \mathbf{0}_6 & \mathbf{P}_{opt}\end{bmatrix}\mathbf{H}^{\top}_{ro} \tag{F.12}
$$

$$
\mathbf{H}_{ro} = \begin{bmatrix}\mathbf{I}_3 & \mathbf{0}_3 & {}^{I_k}_{K}\mathbf{R}_{vio} & \mathbf{0}_3 \\ \mathbf{0}_3 & {}^{K}_{G}\mathbf{R}^{\top}_{opt} & -{}^{K}_{G}\mathbf{R}^{\top}_{opt}\lfloor{}^{K}\mathbf{p}_{I_k,vio}\times\rfloor & \mathbf{I}_3\end{bmatrix} \tag{F.13}
$$

where $\mathbf{P}_{rel}$ is the covariance of the relative $\{{}^{I_k}_{K}\mathbf{R}_{vio}, {}^{K}\mathbf{p}_{I_k,vio}\}$ calculated via Eq. (F.6).

# Appendix G

# COVARIANCE INTERSECTION DELAYED FEATURE INITIALIZATION

## G.1 Problem Statement

In an extended Kalman filter [139], we wish to initialize previously unseen state variables using feature measurements. This process is called *delayed initialization* since typically the process delayed to collect enough measurement observations to fully recover the to-be initialized state variable. In what follows we will first introduce two methods for performing delayed initialization, after which we will introduce the covariance intersection (CI) [87] update. We then re-derive the delayed initialization procedure when covariance intersection is leveraged.

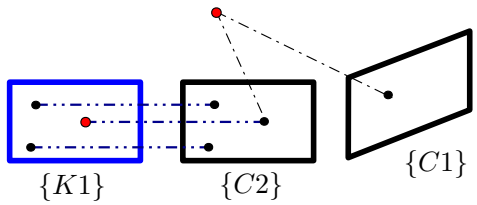



Figure G.1: Illustration of the considered visual feature observation scenario. In this case, a historical keyframe $\{K1\}$ has been matched to an actively tracked feature (red). We wish to initialize this feature estimate into our state using all three measurements from the keyframe and poses $\{C1\}$ and $\{C2\}$.

We now consider a bit more concrete measurement model as shown in Figure G.1. We consider we observe a 3d environmental feature with a camera from two camera clone poses, $\{C1\}$ and $\{C2\}$, along with a loop-closure measurement from a historical keyframe state $\{K1\}$. In the case of delayed initialization we do not have the

feature estimate ${}^{G}\mathbf{p}_f$ mean, uncertainty, or correlation with the rest of the state yet. Consider we have this linearized measurement model and obtain the following residual:

$$\mathbf{r}_i = \mathbf{z}_i - \mathbf{h}(\hat{\mathbf{x}}_{T_i}, {}^{G}\hat{\mathbf{p}}_f) \tag{G.1}$$

$$\simeq \mathbf{H}_{T_i}\tilde{\mathbf{x}}_{T_i} + \mathbf{H}_{f_i}{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n}_i \tag{G.2}$$

where $\mathbf{H}_{T_i}$ and $\mathbf{H}_{f_i}$ are the measurement Jacobians, and $\tilde{\mathbf{x}}_{T_i}$ and ${}^{G}\tilde{\mathbf{p}}_f$ are the error states for the observation pose and feature, respectively. After sufficient observations of the feature, we can "stack" them as:

$$\mathbf{r} = \mathbf{H}_T\tilde{\mathbf{x}}_{T_{1..c}} + \mathbf{H}_{T_k}\tilde{\mathbf{x}}_{T_k} + \mathbf{H}_f{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n} \tag{G.3}$$

$$= \mathbf{H}_a\tilde{\mathbf{x}}_A + \mathbf{H}_k\tilde{\mathbf{x}}_K + \mathbf{H}_f{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n} \tag{G.4}$$

$$= \underbrace{\begin{bmatrix}\mathbf{H}_a & \mathbf{H}_k\end{bmatrix}}_{\mathbf{H}_x}\begin{bmatrix}\tilde{\mathbf{x}}_A \\ \tilde{\mathbf{x}}_K\end{bmatrix} + \mathbf{H}_f{}^{G}\tilde{\mathbf{p}}_f + \mathbf{n} \tag{G.5}$$

where the measurement is a function of $c$ clone poses, $\tilde{\mathbf{x}}_{T_{1..c}} = [\tilde{\mathbf{x}}_{T_1}^\top \cdots \tilde{\mathbf{x}}_{T_c}^\top]^\top$, corresponding to each non-keyframe observation time the feature was seen, and the stacked measurement noise is $\mathbf{n} \sim \mathcal{N}(\mathbf{0}, \mathbf{R})$ where $\mathbf{R} = \sigma_{pix}^2\mathbf{I}$.

## G.2 EKF-based Delayed Initialization

### G.2.1 Method 1: Two System Invertible

Based on the stacked linearized measurement equation, Eq. (G.5), we aim to optimally compute the initial estimate of a new state variable and its covariance and correlations with the existing state variables. As derived by Mingyang Li [111] we first perform QR decomposition (e.g., using computationally efficient in-place Givens rotations) to separate the linear system into two subsystems: (i) one that depends on the new state (i.e., ${}^{G}\mathbf{p}_f$), and (ii) the other that does not.

$$\mathbf{r} = \begin{bmatrix}\mathbf{H}_x & \mathbf{H}_f\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_k \\ {}^{G}\tilde{\mathbf{p}}_f\end{bmatrix} + \mathbf{n} \tag{G.6}$$

$$\Rightarrow \begin{bmatrix}\mathbf{r}_1 \\ \mathbf{r}_2\end{bmatrix} = \begin{bmatrix}\mathbf{H}_{x1} & \mathbf{H}_{f1} \\ \mathbf{H}_{x2} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\tilde{\mathbf{x}}_k \\ {}^{G}\tilde{\mathbf{p}}_f\end{bmatrix} + \begin{bmatrix}\mathbf{n}_{f1} \\ \mathbf{n}_{f2}\end{bmatrix} \tag{G.7}$$

where $\mathbf{n}_{fi} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}_{fi})$, $i \in \{1, 2\}$. Note that in the above expression $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormally transformed measurement residuals, not the direct partitions of $\mathbf{r}$. With the *top* transformed linearized measurement residual $\mathbf{r}_1$ in Eq. (G.7), we now initialize the state estimate of ${}^G\mathbf{p}_f$ and its covariance and correlations to $\mathbf{x}_k$ [see Eq. (5.1)], which will then be augmented to the current state and covariance matrix.

$$
\begin{aligned}
{}^G\tilde{\mathbf{p}}_f &= \mathbf{H}_{f1}^{-1}(\mathbf{r}_1 - \mathbf{n}_1 - \mathbf{H}_x\tilde{\mathbf{x}}) && \text{(G.8)}\\
\Rightarrow\ \mathbb{E}[{}^G\tilde{\mathbf{p}}_f] &= \mathbf{H}_{f1}^{-1}(\mathbf{r}_1) && \text{(G.9)}\\
\mathbf{P}_{ff} &= \mathbb{E}\Big[({}^G\tilde{\mathbf{p}}_f - \mathbb{E}[{}^G\tilde{\mathbf{p}}_f])({}^G\tilde{\mathbf{p}}_f - \mathbb{E}[{}^G\tilde{\mathbf{p}}_f])^\top\Big] && \text{(G.10)}\\
&= \mathbb{E}\Big[(\mathbf{H}_{f1}^{-1}(-\mathbf{n}_1 - \mathbf{H}_{x1}\tilde{\mathbf{x}}))(\mathbf{H}_{f1}^{-1}(-\mathbf{n}_1 - \mathbf{H}_{x1}\tilde{\mathbf{x}}))^\top\Big] && \text{(G.11)}\\
&= \mathbf{H}_{f1}^{-1}(\mathbf{H}_{x1}\mathbf{P}_{xx}\mathbf{H}_{x1}^\top + \mathbf{R}_1)\mathbf{H}_{f1}^{-\top} && \text{(G.12)}\\
\mathbf{P}_{xf} &= \mathbb{E}\Big[(\tilde{\mathbf{x}})({}^G\tilde{\mathbf{p}}_f - \mathbb{E}[{}^G\tilde{\mathbf{p}}_f])^\top\Big] && \text{(G.13)}\\
&= \mathbb{E}\Big[(\tilde{\mathbf{x}})(\mathbf{H}_{f1}^{-1}(-\mathbf{n}_1 - \mathbf{H}_{x1}\tilde{\mathbf{x}}))^\top\Big] && \text{(G.14)}\\
&= -\mathbf{P}_{xx}\mathbf{H}_{x1}^\top\mathbf{H}_{f1}^{-\top} && \text{(G.15)}
\end{aligned}
$$

where $\mathbb{E}[\cdot]$ is the expectation operator. These derivations can be summarized as follows:

$$
\begin{aligned}
{}^G\mathbf{p}_f^\oplus &= {}^G\mathbf{p}_f + \mathbf{H}_{f1}^{-1}\mathbf{r}_1 && \text{(G.16)}\\
\mathbf{P}_{xx}^\oplus &= \mathbf{P}_{xx} && \text{(G.17)}\\
\mathbf{P}_{ff}^\oplus &= \mathbf{H}_{f1}^{-1}(\mathbf{H}_{x1}\mathbf{P}_{xx}\mathbf{H}_{x1}^\top + \mathbf{R}_{f1})\mathbf{H}_{f1}^{-\top} && \text{(G.18)}\\
\mathbf{P}_{xf}^\oplus &= -\mathbf{P}_{xx}\mathbf{H}_{x1}^\top\mathbf{H}_{f1}^{-\top} && \text{(G.19)}\\
\mathbf{P}_{fx}^\oplus &= (\mathbf{P}_{xf}^\oplus)^\top && \text{(G.20)}
\end{aligned}
$$

It should be noted that a full-rank $\mathbf{H}_{f1}$ is needed to perform the above initialization, which normally is the case if enough measurements are collected (i.e., delayed initialization). Note also that to utilize all available measurement information, we also perform EKF update using the *bottom* measurement residual $\mathbf{r}_2$ in Eq. (2.42).

### G.2.2 Method 2: Infinite Uncertainty with Update

We now look at an alternate formulation for delayed initialization. In this method, we consider the case where the state already has a prior covariance of the state-to-be-initialized but its uncertainty is at infinity and has not been correlated with the current state through a measurement yet. More concretely, we define the following covariance:

$$\mathbf{P}_k = \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{0} \\ \mathbf{0} & \mu\mathbf{I} \end{bmatrix} \tag{G.21}$$

where we have defined $\mathbf{P}_{ff} = \mu\mathbf{I}$ with $\mu \to \infty$ since we have no prior knowledge of the feature's state. We now wish to perform an EKF update using the measurement information collected. We define the stacked measurements as:

$$\mathbf{r} = \mathbf{H}_k\tilde{\mathbf{x}}_k + \mathbf{n} \tag{G.22}$$

$$= \begin{bmatrix} \mathbf{H}_x & \mathbf{H}_f \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{x}}_A \\ {}^G\tilde{\mathbf{p}}_f \end{bmatrix} + \mathbf{n} \tag{G.23}$$

This gives us the following update equations:

$$\mathbf{x}_A^{\oplus} = \mathbf{x}_A + \mathbf{K}_x\mathbf{r} \tag{G.24}$$

$${}^G\mathbf{p}_f^{\oplus} = {}^G\mathbf{p}_f + \mathbf{K}_f\mathbf{r} \tag{G.25}$$

$$\mathbf{P}_k^{\oplus} = \mathbf{P}_k - \begin{bmatrix} \mathbf{K}_x\mathbf{S}_k\mathbf{K}_x^\top & \mathbf{K}_x\mathbf{H}_k \begin{bmatrix} \mathbf{P}_{xf} \\ \mathbf{P}_{ff} \end{bmatrix} \\ \begin{bmatrix} \mathbf{P}_{xf} \\ \mathbf{P}_{ff} \end{bmatrix}^\top \mathbf{H}_k{}^\top\mathbf{K}_x^\top & \mathbf{K}_f\mathbf{S}_k\mathbf{K}_f^\top \end{bmatrix} \tag{G.26}$$

$$= \begin{bmatrix} \mathbf{P}_{xx} & \mathbf{0} \\ \mathbf{0} & \mathbf{P}_{ff} \end{bmatrix} - \begin{bmatrix} \mathbf{K}_x\mathbf{S}_k\mathbf{K}_x^\top & \mathbf{K}_x\mathbf{H}_f\mathbf{P}_{ff} \\ \mathbf{P}_{ff}\mathbf{H}_f{}^\top\mathbf{K}_x^\top & \mathbf{K}_f\mathbf{S}_k\mathbf{K}_f^\top \end{bmatrix} \tag{G.27}$$

where we have used that the initial feature is uncorrelated with the state (i.e., $\mathbf{P}_{xf} = \mathbf{P}_{xf}^\top = \mathbf{0}$) and we have defined the following Kalman gains:

$$\mathbf{K}_k = \begin{bmatrix} \mathbf{K}_x \\ \mathbf{K}_f \end{bmatrix} = \begin{bmatrix} \mathbf{P}_{xx}\mathbf{H}_x^\top + \mathbf{P}_{xf}\mathbf{H}_f^\top \\ \mathbf{P}_{fx}\mathbf{H}_x^\top + \mathbf{P}_{ff}\mathbf{H}_f^\top \end{bmatrix} \mathbf{S}_k^{-1} := \begin{bmatrix} \mathbf{P}_{xx}\mathbf{H}_x^\top \\ \mathbf{P}_{ff}\mathbf{H}_f^\top \end{bmatrix} \mathbf{S}_k^{-1} \tag{G.28}$$

We now first look at how to calculate the measurement innovation term. It is as follows:

$$\mathbf{S}_k^{-1} = \left(\mathbf{H}_k \mathbf{P}_k \mathbf{H}_k^\top + \mathbf{R}_m\right)^{-1} \tag{G.29}$$

$$= \left(\mathbf{H}_x \mathbf{P}_{xx} \mathbf{H}_x^\top + \mathbf{H}_f \mathbf{P}_{ff} \mathbf{H}_f^\top + \mathbf{R}_m\right)^{-1} \tag{G.30}$$

$$= \left(\mathbf{A} + \mathbf{H}_f \mathbf{P}_{ff} \mathbf{H}_f^\top\right)^{-1} \tag{G.31}$$

$$= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}_f \left(\mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f + \cancel{\mathbf{P}_{ff}^{-1}}\right)^{-1} \mathbf{H}_f^\top \mathbf{A}^{-1} \tag{G.32}$$

$$= \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}_f \left(\mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f\right)^{-1} \mathbf{H}_f^\top \mathbf{A}^{-1} \tag{G.33}$$

where we have defined $\mathbf{A} = \mathbf{H}_x \mathbf{P}_{xx} \mathbf{H}_x^\top + \mathbf{R}_m$, and $\mathbf{P}_{ff}^{-1} = (\mu \mathbf{I})^{-1} \to \mathbf{0}$ when $\mu \to \infty$, and the matrix inversion lemma as:

$$(\mathbf{A} + \mathbf{UCV})^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}\left(\mathbf{VA}^{-1}\mathbf{U} + \mathbf{C}^{-1}\right)^{-1}\mathbf{VA}^{-1} \tag{G.34}$$

This leads the following conclusion for $\mathbf{P}_{xx}$:

$$\mathbf{P}_{xx}^{\oplus} = \mathbf{P}_{xx} - \mathbf{P}_{xx}\mathbf{H}_x^\top \mathbf{S}_k^{-1} \mathbf{H}_x \mathbf{P}_{xx} \tag{G.35}$$

$$= \mathbf{P}_{xx} - \mathbf{P}_{xx}\mathbf{H}_x^\top \left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}_f \left(\mathbf{H}_f^\top \mathbf{A}^{-1}\mathbf{H}_f\right)^{-1} \mathbf{H}_f^\top \mathbf{A}^{-1}\right) \mathbf{H}_x \mathbf{P}_{xx} \tag{G.36}$$

Now we look at how to compute the feature's uncertainty $\mathbf{P}_{ff}$.

$$\mathbf{P}_{ff}^{\oplus} = \mathbf{P}_{ff} - \mathbf{K}_f \mathbf{S}_k \mathbf{K}_f^\top \tag{G.37}$$

$$= \mathbf{P}_{ff} - \mathbf{P}_{ff}\mathbf{H}_f^\top \mathbf{S}_k^{-1} \mathbf{H}_f \mathbf{P}_{ff} \tag{G.38}$$

$$= \mathbf{P}_{ff} + \mathbf{P}_{ff}\mathbf{H}_f^\top (-\mathbf{S}_k^{-1}) \mathbf{H}_f \mathbf{P}_{ff} \tag{G.39}$$

$$= \left(\cancel{\mathbf{P}_{ff}^{-1}} - \cancel{\mathbf{P}_{ff}^{-1}\mathbf{P}_{ff}}\mathbf{H}_f^\top \left((-\mathbf{S}_k) + \mathbf{H}_f \mathbf{P}_{ff} \cancel{\mathbf{P}_{ff}^{-1}\mathbf{P}_{ff}} \mathbf{H}_f^\top\right)^{-1} \mathbf{H}_f \mathbf{P}_{ff} \cancel{\mathbf{P}_{ff}^{-1}}\right)^{-1} \tag{G.40}$$

$$= \left(-\mathbf{H}_f^\top \left((-\mathbf{S}_k) + \mathbf{H}_f \mathbf{P}_{ff} \mathbf{H}_f^\top\right)^{-1} \mathbf{H}_f\right)^{-1} \tag{G.41}$$

$$= \left(-\mathbf{H}_f^\top \left((-\mathbf{A} - \mathbf{H}_f \mathbf{P}_{ff} \mathbf{H}_f^\top) + \mathbf{H}_f \mathbf{P}_{ff} \mathbf{H}_f^\top\right)^{-1} \mathbf{H}_f\right)^{-1} \tag{G.42}$$

$$= \left(\mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f\right)^{-1} \tag{G.43}$$

$$\tag{G.44}$$

This leads the following conclusion for $\mathbf{P}_{ff}$:

$$\mathbf{P}_{ff}^{\oplus} = \left(\mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f\right)^{-1} \tag{G.45}$$

Now we look at how to compute the feature's correlation with the state $\mathbf{P}_{xf}$.

$$\mathbf{P}_{xf}^{\oplus} = \cancel{\mathbf{P}_{xf}} - \mathbf{K}_x \mathbf{H}_f \mathbf{P}_{ff} \tag{G.46}$$

$$= -\mathbf{P}_{xx} \mathbf{H}_x^\top \mathbf{S}_k^{-1} \mathbf{H}_f \mathbf{P}_{ff} \tag{G.47}$$

Looking at the last three terms and substituting in the equality from Eq. (G.32) (the only part that is a function of $\mathbf{P}_{ff}$) we have:

$$\mathbf{S}_k^{-1} \mathbf{H}_f \mathbf{P}_{ff} = \left( \mathbf{A}^{-1} - \mathbf{A}^{-1} \mathbf{H}_f \left( \mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f + \mathbf{P}_{ff}^{-1} \right)^{-1} \mathbf{H}_f^\top \mathbf{A}^{-1} \right) \mathbf{H}_f \mathbf{P}_{ff} \tag{G.48}$$

$$= \mathbf{A}^{-1} \mathbf{H}_f \left( \mathbf{I} - \left( \mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f + \mathbf{P}_{ff}^{-1} \right)^{-1} \mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f \right) \mathbf{P}_{ff} \tag{G.49}$$

$$= \mathbf{A}^{-1} \mathbf{H}_f \left( \mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f + \mathbf{P}_{ff}^{-1} \right)^{-1} \left[ \left( \cancel{\mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f} + \mathbf{P}_{ff}^{-1} \right) - \cancel{\mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f} \right] \mathbf{P}_{ff} \tag{G.50}$$

$$= \mathbf{A}^{-1} \mathbf{H}_f \left( \mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f + \cancel{\mathbf{P}_{ff}^{-1}} \right)^{-1} \tag{G.51}$$

$$= \mathbf{A}^{-1} \mathbf{H}_f \left( \mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f \right)^{-1} \tag{G.52}$$

This leads the following conclusion for $\mathbf{P}_{xf}$:

$$\mathbf{P}_{xf}^{\oplus} = -\mathbf{P}_{xx} \mathbf{H}_x^\top \mathbf{A}^{-1} \mathbf{H}_f \left( \mathbf{H}_f^\top \mathbf{A}^{-1} \mathbf{H}_f \right)^{-1} \tag{G.53}$$

The state means can be updated similarly.

### G.2.3 Method Equivalence

A natural question is the equivalence between these two methods. Just by looking at Method's 1 Eq. (G.17) and Method's 2 Eq. (G.36) one can see that Method 2 updates the original covariance while the first method does not! At first glance this would mean that the two methods are not doing the exact same thing and that one is better than the other. There is a subtle difference between the two: Method 1 first initializes with a sub-system of the full measurement, while the Method 2 initializes the prior information with all measurements. We can show that Method 2 is exactly the same as the first by considering we have a square measurement Jacobian that is

invertible $\mathbf{H}_f\mathbf{H}_f^{-1} = \mathbf{H}_f^{-1}\mathbf{H}_f = \mathbf{I}$ (thus there is no second update using $\mathbf{r}_2$ in method 1). We get:

$$\mathbf{P}_{xx}^{\oplus} = \mathbf{P}_{xx} - \mathbf{P}_{xx}\mathbf{H}_x^\top \left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}_f \left(\mathbf{H}_f^\top \mathbf{A}^{-1}\mathbf{H}_f\right)^{-1} \mathbf{H}_f^\top \mathbf{A}^{-1}\right) \mathbf{H}_x\mathbf{P}_{xx} \tag{G.54}$$

$$= \mathbf{P}_{xx} - \mathbf{P}_{xx}\mathbf{H}_x^\top \left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}_f\mathbf{H}_f^{-1}\mathbf{A}\mathbf{H}_f^{-\top}\mathbf{H}_f^\top \mathbf{A}^{-1}\right) \mathbf{H}_x\mathbf{P}_{xx} \tag{G.55}$$

$$= \mathbf{P}_{xx} - \mathbf{P}_{xx}\mathbf{H}_x^\top \left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{A}\mathbf{A}^{-1}\right) \mathbf{H}_x\mathbf{P}_{xx} \tag{G.56}$$

$$= \mathbf{P}_{xx} - \mathbf{P}_{xx}\mathbf{H}_x^\top \left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\right) \mathbf{H}_x\mathbf{P}_{xx} \tag{G.57}$$

$$= \mathbf{P}_{xx} \tag{G.58}$$

$$\mathbf{P}_{ff}^{\oplus} = \left(\mathbf{H}_f^\top \mathbf{A}^{-1}\mathbf{H}_f\right)^{-1} \tag{G.59}$$

$$= \mathbf{H}_f^{-1} \left(\mathbf{H}_x\mathbf{P}_{xx}\mathbf{H}_x^\top + \mathbf{R}_m\right) \mathbf{H}_f^{-\top} \tag{G.60}$$

$$\mathbf{P}_{xf}^{\oplus} = -\mathbf{P}_{xx}\mathbf{H}_x^\top \mathbf{A}^{-1}\mathbf{H}_f \left(\mathbf{H}_f^\top \mathbf{A}^{-1}\mathbf{H}_f\right)^{-1} \tag{G.61}$$

$$= -\mathbf{P}_{xx}\mathbf{H}_x^\top \mathbf{A}^{-1}\mathbf{H}_f\mathbf{H}_f^{-1}\mathbf{A}\mathbf{H}_f^{-\top} \tag{G.62}$$

$$= -\mathbf{P}_{xx}\mathbf{H}_x^\top \mathbf{A}^{-1}\mathbf{A}\mathbf{H}_f^{-\top} \tag{G.63}$$

$$= -\mathbf{P}_{xx}\mathbf{H}_x^\top \mathbf{H}_f^{-\top} \tag{G.64}$$

To explain it in an intuitive way, if you have a square matrix, there is enough measurement information to recover the state you wish to initialize. But just having this information does not allow you to improve your state estimate (decrease $\mathbf{P}_{xx}$). Once you have more measurements than what is required to initialize it then you can improve it (e.g., non-empty secondary system in Method 1).

## G.3 Covariance Intersection-based Delayed Initialization

### G.3.1 Covariance Intersection State Update

To guarantee consistency when updating with this measurement, we adopt the CI-EKF update [87] to construct a prior covariance such that:

$$\mathbf{Diag}\left(\frac{1}{\omega_a}\mathbf{P}_{aa}, \frac{1}{\omega_1}\mathbf{P}_1, \cdots, \frac{1}{\omega_n}\mathbf{P}_n\right) \geq \mathbf{P}_k \tag{G.65}$$

where the left side is the CI covariance with zero off-diagonal elements and the right-hand side is the unknown true covariance of the state with cross-covariances. The

weights $\omega_l > 0$ and $\sum_l \omega_l = 1$, for $l \in \{a, 1...n\}$, can be found optimally [87]. The first weight corresponds to the "active" covariance, while the remainder corresponds to each keyframe for which we only keep their marginal covariance and do not track their correlations with the active state elements.

Substituting Eq. (G.65) into the standard EKF equations and only selecting the portion that updates active state yields (that is, we do not update keyframe states in the prior map):

$$\mathbf{x}_A^{\oplus} = \mathbf{x}_A + \frac{1}{\omega_a}\mathbf{P}_{aa}\mathbf{H}_a^{\top}\mathbf{S}_k^{-1}\mathbf{r} \tag{G.66}$$

$$\mathbf{P}_{aa}^{\oplus} = \frac{1}{\omega_a}\mathbf{P}_{aa} - \frac{1}{\omega_a^2}\mathbf{P}_{aa}\mathbf{H}_a^{\top}\mathbf{S}_k^{-1}\mathbf{H}_a\mathbf{P}_{aa} \tag{G.67}$$

$$\mathbf{S}_k = \sum_{o\in\{a,1...n\}} \left(\frac{1}{\omega_o}\mathbf{H}_o\mathbf{P}_{oo}\mathbf{H}_o^{\top}\right) + \mathbf{R}_m \tag{G.68}$$

### G.3.2 Delayed Initialization

We can follow the logic presented in the previous segments. Specifically, we can start with the following covariance matrix:

$$\mathbf{P}_k = \begin{bmatrix} \frac{1}{\omega_a}\mathbf{P}_{aa} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \ddots & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \frac{1}{\omega_o}\mathbf{P}_{oo} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \ddots & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mu\mathbf{I} \end{bmatrix} \tag{G.69}$$

We can see that the CI variables only show up in the $\mathbf{S}_k$ term and can be grouped into the value $\mathbf{A}$ as before (see Eq. (G.30)).

$$\mathbf{A} = \frac{1}{\omega_a}\mathbf{H}_a\mathbf{P}_{aa}\mathbf{H}_a^{\top} + \sum_{o\in\{1...n\}} \left(\frac{1}{\omega_o}\mathbf{H}_o\mathbf{P}_{oo}\mathbf{H}_o^{\top}\right) + \mathbf{R}_m \tag{G.70}$$

We can then perform an update using Eq. (G.66)-(G.68) to get the following:

$$\mathbf{P}_{xx}^{\oplus} = \frac{1}{\omega_a}\mathbf{P}_{aa} - \frac{1}{\omega_a^2}\mathbf{P}_{aa}\mathbf{H}_a^{\top}\left(\mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{H}_f\left(\mathbf{H}_f^{\top}\mathbf{A}^{-1}\mathbf{H}_f\right)^{-1}\mathbf{H}_f^{\top}\mathbf{A}^{-1}\right)\mathbf{H}_a\mathbf{P}_{aa} \tag{G.71}$$

$$\mathbf{P}_{ff}^{\oplus} = \left(\mathbf{H}_f^{\top}\mathbf{A}^{-1}\mathbf{H}_f\right)^{-1} \tag{G.72}$$

$$\mathbf{P}_{af}^{\oplus} = -\frac{1}{\omega_a}\mathbf{P}_{aa}\mathbf{H}_a^{\top}\mathbf{A}^{-1}\mathbf{H}_f\left(\mathbf{H}_f^{\top}\mathbf{A}^{-1}\mathbf{H}_f\right)^{-1} \tag{G.73}$$

We can then equate this result to Method's 1 structure to get:

$${}^{G}\mathbf{p}_f^{\oplus} = {}^{G}\mathbf{p}_f + \mathbf{H}_f^{-1}\mathbf{r}_1 \tag{G.74}$$

$$\mathbf{P}_{aa}^{\oplus} = \frac{1}{\omega_a}\mathbf{P}_{aa} \tag{G.75}$$

$$\mathbf{P}_{ff}^{\oplus} = \mathbf{H}_f^{-1}\left[\frac{1}{\omega_a}\mathbf{H}_a\mathbf{P}_{aa}\mathbf{H}_a^{\top} + \sum_{o\in\{1\ldots n\}}\left(\frac{1}{\omega_o}\mathbf{H}_o\mathbf{P}_{oo}\mathbf{H}_o^{\top}\right) + \mathbf{R}_m\right]\mathbf{H}_f^{-\top} \tag{G.76}$$

$$\mathbf{P}_{af}^{\oplus} = -\frac{1}{\omega_a}\mathbf{P}_{aa}\mathbf{H}_a^{\top}\mathbf{H}_f^{-\top} \tag{G.77}$$

$$\mathbf{P}_{fa}^{\oplus} = (\mathbf{P}_{af}^{\oplus})^{\top} \tag{G.78}$$

# Appendix H

# PERMISSIONS