Resource-Constrained State Estimation with Multi-Modal Sensing

John W. Yao

CMU-RI-TR-20-05
April 2020

Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Nathan Michael, *Chair*
William "Red" Whittaker
Michael Kaess
Hatem Alismail, Argo AI

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy in Robotics

Abstract

Accurate and reliable state estimation is essential for safe mobile robot operation in real-world environments because ego-motion estimates are required by many critical autonomy functions such as control, planning, and mapping. Computing accurate state estimates depends on the physical characteristics of the environment, the selection of suitable sensors to capture that information, and the availability of compute to perform inference on sensor observations. Because environment characteristics cannot be controlled in typical field operation scenarios, careful sensor selection and efficient fusion algorithms are required to achieve accurate, high-rate state estimation on mobile robots.

Visual-inertial odometry is well-suited to onboard state estimation in environments that have rich visual texture. As with other vision-based methods, it performs poorly when operating on images with non-ideal characteristics such as low visual texture, dynamic objects, or far away scenes. Common methods to mitigate these problems include introducing visual observations from different fields of view and using depth observations. However, processing additional sources and types of observations requires more computation, which is extremely limited on size, weight, and power constrained robots such as micro aerial vehicles. Therefore, there is a need to reduce the computational burden associated with using more observations to increase accuracy through efficient selection of useful observations from multiple heterogeneous sensors.

In this thesis, we propose an optimization-based state estimator that fuses observations from multiple heterogeneous sensors to achieve reliable performance in scenarios that are challenging for typical minimal sensing configurations. The foundation is a baseline monocular sparse visual-inertial odometry algorithm using fixed-lag smoothing augmented with improvements in covariance estimation and initialization from rest. We extend this formulation to improve state estimation performance in non-ideal sensing conditions by (1) modifying the visual-inertial odometry framework to support multiple asynchronous cameras with disjoint fields of view, (2) leveraging depth observations to boost performance in visually degraded environments, and (3) developing methods to allocate limited computational resources to process exteroceptive observations that are more informative for estimation. The proposed methods are evaluated both in real time onboard an aerial robot flying in a variety of challenging environments, as well as in postprocessing on datasets collected using the aerial robot.

Acknowledgments

First and foremost, I would like to thank my advisor, Prof. Nathan Michael, for his guidance, feedback, and insight throughout my time at CMU. He fostered an environment where I could learn to become a better roboticist through theory and practice, and repeatedly challenged me to go above and beyond my personal limits in pursuing that goal. I am also grateful to my thesis committee members, Prof. Red Whittaker, Prof. Michael Kaess, and Dr. Hatem Alismail, who provided valuable advice and encouragement throughout this process.

I also have to thank my friends and colleagues who have collaborated with me over the past years and helped improve my graduate school experience. I would like to thank my reliable senpais Vishnu Desaraju and Humphrey Hu for always being willing to listen to my problems and offer feedback on my research ideas. I am grateful to Erik Nelson for patiently helping me get up to speed with the practical side of aerial robotics and helping me survive the storm of demos and sponsor deliverables in the lab's early years. I would like to thank Curtis Boirum for designing, building, flying, and repairing our lab's aerial robots - without him none of the hardware experimental results in this thesis would have been possible. I am thankful for Wennie Tabib's leadership of the cave exploration project, as well for her help in encouraging me to see the big picture and not get lost in dead-end research endeavors. I am grateful to Kshitij Goel for his help in running many of the aerial robot experiments as well as working on the infrastructure that facilitates them. I would like to thank Aditya Dhawale and Cormac O'Meadhra for their help with depth-based odometry. I am grateful to past and present members of the Resilient Intelligent Systems Laboratory for making my time at CMU memorable, especially Kumar Shaurya Shankar, Ellen Cappo, Matt Collins, Micah Corah, Xuning Yang, Alex Spitzer, Arjav Desai, Mike Lee, and Mosam Dabhi. I would like to thank Chuck Whittaker, Karen Widmaier, Nora Kazour, Suzanne Lyons Muth, and Alison Day for their support and help with various things throughout my time at CMU.

Finally, I would like to thank my friends and family for their support and encouragement throughout the course of my Ph.D. I am grateful that they served as constant reminders, especially in the most trying of times, that there is more to life than graduate school.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Accurate and reliable navigation using onboard sensors is fundamental to the deployment of autonomous robots to execute tasks in real-world environments. State estimation is the most important building block of any intelligent mobile robot's autonomy framework because it provides information that is consumed by other system components such as planning, control, and mapping. For example, the controller compares state estimates with reference signals to compute actuator inputs, the planner uses state estimates to compute trajectories with desirable properties, and the mapper uses state estimates to stitch together range observations into consistent maps.

In the context of mobile robot navigation, a state estimator uses sensor observations to compute a best guess about the robot's current motion, including but not limited to 3D position, orientation, linear velocity, and angular velocity. Visual, inertial, and depth sensors are commonly used in mobile robot state estimation due to the complementary nature of their observations. Inertial measurement units (IMUs) have noisy, drift-prone, but outlier-free observations that are good for short-term tracking of dynamic motions, while cameras provide rich, high-dimensional image observations that are suitable for navigation at larger spatial scales but also susceptible to outliers. Depth sensors provide 3D information about the presence or absence of matter within the field of view and are less sensitive to visual texture and environmental lighting conditions than cameras. Observations from these sensors are fed into a probabilistic model of environment structure and robot motion, enabling the application of maximum likelihood approaches to estimate the hidden parameters.

State estimation algorithms for size, weight, and power constrained robots must reconcile the competing requirements of achieving high localization accuracy in a wide range of environments and keeping computational costs low. Localization accuracy depends on the information provided by sensor observations, the fidelity of sensor observation models, and the availability of compute. For exteroceptive sensors, the information that can be provided to state estimation is determined by sensor properties and the environment's physical characteristics. Examples of methods to increase robustness to varying environment characteristics include using different types of exteroceptive sensors and using higher performance sensors, but these come at the cost of higher size, weight, and power requirements. Another avenue of improving localization accuracy is through the use of high fidelity sensor models and probabilistic inference strategies that come at the cost of greater computation.

Many computationally constrained systems use visual-inertial odometry (VIO) for minimal sensor configurations consisting of only a single camera and IMU. These types of setups can achieve accurate motion estimation in environments with constant lighting and non-trivial visual texture due to the richness of visual information. However, their performance is highly sensitive to the quality of visual observations used to bound drift caused by inertial observations. Image sequences taken by cameras operating in visually degraded conditions such as underground, around abandoned buildings, under foliage, and outdoors at dawn

or dusk have non-constant lighting, low contrast, low brightness and repetitive texture. These characteristics are challenging for visual-inertial state estimation algorithms due to their violation of common simplifying assumptions about the environment. To overcome these limitations and enable estimation in a wider range of environmental conditions, depth sensors can be used to provide information in regions with low visual saliency but high geometric saliency. However, strategies for incorporating depth observations must be tailored to avoid excessive increases in required computation.

Computational tractability problems stem from the limited onboard processing power available on many mobile robots, coupled with the high dimensionality of visual and depth observations. Kinematic models used to incorporate inertial observations and geometric models used to incorporate exteroceptive observations are highly nonlinear, increasing the cost and complexity of probabilistic inference algorithms. To exacerbate the problem, other autonomy functions such as planning and control compete with state estimation for limited computational resources. Agile, small form factor systems such as micro aerial vehicles require high-rate control driven by high-rate state estimation, but have the most severe size, weight, and power constraints of all mobile robots. Consequently, improving the computational efficiency of multi-sensor state estimation is key to building robust autonomy frameworks for these systems.

This thesis seeks to address these challenges through the implementation and validation of an optimization-based state estimator that fuses information from vision, inertial, and depth sensors to improve reliability in situations that are challenging for typical minimal sensing configurations. We reduce state estimation optimization problem size by selecting a subset of highly informative observations across multiple sensors determined by a predictive model of anticipated motion, facilitating accurate odometry with a limited computational footprint. The contributions of this thesis are summarized below and detailed in subsequent chapters.

Monocular sliding window visual-inertial odometry: A baseline tightly-coupled optimization-based state estimator that fuses IMU and sparse feature observations from a single camera and an auxiliary estimator for takeoff and mid-air reinitialization. (Chapters 3, 4)

Multi-camera sliding window visual-inertial odometry: Extension of baseline state estimator to efficiently process sparse feature observations from multiple asynchronous cameras with disjoint fields of view. (Chapter 5)

Sensor resource allocation for computationally constrained state estimation: A method for selecting informative visual features that facilitate accurate state estimation while keeping optimization problem size small. (Chapter 6)

Depth-aided visual-inertial odometry: Methods for incorporating depth observations into sparse visual-inertial state estimation that make the latter more robust to environments with poor lighting and/or visual texture. (Chapter 7)

Implementation and experimental evaluation on an aerial robot: A set of flight experiments conducted to evaluate the performance of baseline and proposed state estimation methods in indoor and outdoor environments subject to different visual texture and lighting conditions. (Appendix G, Sections 3.9, 5.4, 6.4, 7.6)

Chapter 2

Background

This chapter reviews existing research on state estimation techniques for visual, inertial, and depth sensors. We cover not only the methods used in our proposed formulation, but also survey related works to provide context for our design decisions. Given that our baseline state estimation framework uses visual and inertial sensors, we cover visual-inertial odometry in Sections 2.1-2.3. Research relevant to the proposed work on feature selection and depth-based estimation is presented in Sections 2.3.4 and 2.4.

2.1 Camera Configuration

2.1.1 Stereo

Stereo camera setups consist of two cameras with overlapping fields of view separated by a known distance called the baseline. Assuming both cameras are calibrated, the 3D location of a real-world point observed in both cameras can be computed via triangulation. Using epipolar geometry, the problem of finding a point in the second camera's image that corresponds to a point observed in the first camera's image can be reduced to a search along a 1D epipolar line.

Although stereo cameras can provide metric information about the scene without moving, triangulation and motion estimation are extremely sensitive to the accuracy of the inter-camera extrinsic calibration. On robots undergoing non-trivial motions, shocks and vibration may cause the cameras to become misaligned, necessitating periodic re-calibration. Another drawback of stereo cameras is that large baselines are required to accurately estimate the depth of far-away objects, which is challenging to reliably perform online. Size constraints limit stereo cameras deployed on small mobile robots to small baselines, limiting their usefulness for handling wide open outdoor environments.

2.1.2 Monocular

A monocular camera setup uses a single camera to estimate motion and the 3D structure of the scene. Unlike the stereo setup, absolute scale is unobservable without extra information in the form of other sensor observations, knowledge about objects in the environment, or motion constraints. In many cases a monocular camera is combined with an IMU to provide scale observability, creating a minimal sensor suite for estimating rigid body motion [1]. The single camera and IMU setup provides superior accuracy during highly dynamic motion, as well as smaller size and processing requirements than the stereo setup.

An extension of the monocular camera setup is one comprised of multiple cameras with non-overlapping fields of view. Although the lack of overlap precludes the use of stereo techniques and hence depth recovery, gaining access to visual information from different parts of the scene improves the robustness of motion estimation to unfavorable visual conditions in any single camera.

2.2 Image Processing

The first task of a vision-based state estimation algorithm is to perform data association between different image observations in preparation for inferring motion. This is a non-trivial problem, given that camera images are high dimensional measurements on the order of 10^5 pixel values. The part of a vision-based state estimation algorithm that handles this task is called the *front-end*. Front-ends can be classified as indirect or direct, and as dense or sparse (see Table 2.1).

Table 2.1: Examples of image processing front-ends

	Sparse	**Dense**
Indirect	PTAM [2], ORB-SLAM [3], OKVIS [4], VINS-Mono [5]	Ranft et al. [6]
Direct	DSO [7], DSVIO [8]	DTAM [9], LSD-SLAM [10]

2.2.1 Indirect Methods

Indirect methods infer a camera's motion by first preprocessing images into an intermediate representation and then performing probabilistic inference with that representation. The preprocessing step typically consists of feature detection and feature tracking. Feature detection involves the selection of distinctive locations in the current image that are expected to be easily identifiable in subsequent images, while feature tracking seeks to find the locations of detected features in subsequent images. Feature correspondences are used to estimate the rigid body transform between image frames by minimizing geometric error associated with reprojecting features from one image frame to the other image frame.

The primary benefit of indirect methods is that features can provide robustness to geometric and photometric distortions that occur in low-cost cameras. Examples of these effects include vignetting, gamma correction, and auto-exposure changes. Increasing the reliability of feature correspondences between image frames can incur non-trivial computational cost, typically through the use of complex feature descriptors and/or outlier rejection schemes. On computationally constrained systems that can only support simpler feature descriptors, data association is degraded by scenes with repetitive visual texture. Another drawback of indirect methods is that they can fail when an insufficient number of features are tracked, such as when the camera's field of view changes too quickly.

Features can be classified as point-based or line-based.

Point Features

Point features, also known as corners, are locations within an image where intensity changes sharply in two directions. For example, the Shi-Tomasi corner [11] is used in the computationally efficient Kanade-Lucas-Tomasi (KLT) tracking algorithm [12]. KLT tracking across image frames seeks to minimize photometric error in small patches around the original image's Shi-Tomasi corner location and potential matching feature locations in the subsequent image. To improve feature matching reliability, patch photometric error can be replaced with descriptors that use other criteria to uniquely identify features across different images, such as SIFT [13], SURF [14], FAST [15], and BRISK [16].

Line Features

Line features, also known as edges, are regions with sharp intensity gradients in a single direction. They are more robust to lighting changes than point features but require more complicated detection and matching routines [17]. Line features are more robust to motion blur but provide less accuracy than point features in nominal tracking scenarios [18]. [19, 20] use line features in addition to point features, while [21] uses lines as sources of high-value point features.

2.2.2 Direct Methods

Direct methods use raw pixel intensity values as measurements in a probabilistic model for inferring a camera's rigid body transformation between images. While probabilistic inference for indirect methods operates on *geometric* errors, for direct methods it operates on *photometric* errors.

The main advantage of direct methods is that they do not require individual points to be recognizable, which provides robustness in low texture scenes where very few features can be detected. Another benefit is that direct methods can more fully exploit camera sensor model knowledge than indirect methods by virtue of incorporating more aspects of the image formulation process in probabilistic inference.

Standard direct methods rely on the *photo-consistency* assumption, which asserts that all parts of the scene have the same intensity value across a pair of frames. While serving as a convenient justification for using photometric error, this assumption is often violated in real-world conditions due to changes in camera exposure time, gain, view angle, and ambient lighting. To address this modeling gap, camera-controlled effects such as vignetting and exposure time variation can be accounted for through calibration [22]. However, capturing the effects of optical properties of surfaces in the environment in a photometric error model is a non-trivial task. One step towards a more realistic model is to replace the photo-consistency assumption with the *irradiance consistency* assumption [7]. This approach can be extended by modeling irradiance, illumination gain and illumination bias as random variables to be jointly estimated alongside camera motion and scene structure [23].

2.2.3 Sparse Methods

Sparse methods use only a selected subset of pixels within an image for motion estimation. The primary advantage of sparse methods over dense methods is the reduction in computation required to perform inference over image observations. For example, extracting 100 features from a 640×480 image reduces observation dimensionality by three orders of magnitude.

2.2.4 Dense Methods

Dense methods use all pixels within an image for motion estimation. While sparse methods assume parts of the scene are conditionally independent given camera poses and intrinsic parameters, dense methods can exploit geometry priors for increased local accuracy. However, the introduction of correlations between geometry parameters densifies the bundle adjustment Hessian, precluding efficient solutions via the Schur complement. Additionally, direct dense methods have a small basin of attraction and are not robust to large frame-to-frame motions [24].

2.3 Probabilistic Inference

After obtaining either direct pixel values or keypoints, a vision-based state estimator performs probabilistic inference to estimate the motion that is the most consistent with available observations. This component of the estimator is called the *back-end*. Back-ends can be classified as loosely-coupled or tightly-coupled depending when visual observations are fused with inertial observations, and as filter-based or optimization-based depending on the method of data fusion. Table 2.2 provides examples of back-ends used in recent works.

Table 2.2: Examples of motion inference back-ends

Filtering	EKF	OC-EKF [25], MSF [26], ROVIO [27]
	UKF	OC-UKF [28], Semi-Dense VIO [29]
	MSCKF	MSCKF [30], MSCKF 2.0 [25], Robocentric VIO [31]
Optimization	Fixed lag	OC-SWF [32], OKVIS [4], VINS-Mono [5]
	Incremental	iSAM [33], iSAM2 [34]

2.3.1 Loosely-Coupled vs. Tightly-Coupled

Loosely-coupled methods process visual measurements into pose observations before fusing them with inertial observations. They sacrifice accuracy for lower computational cost and greater ease of implementation. On the other hand, tightly-coupled formulations directly use visual measurements (either features or pixel intensity values) in motion estimation. Because tightly-coupled methods account for cross-correlations between visual and inertial sensor model parameters (e.g. feature depth, IMU biases, etc.), they can achieve greater accuracy than loosely-coupled methods. However, tightly-coupled methods are more computationally expensive and have higher complexity than loosely-coupled methods.

2.3.2 Filtering

Filtering imposes a Markov assumption on the probabilistic graphical model of visual and inertial sensor observations. The current hidden state, consisting of quantities such as pose, twist, IMU biases, and feature locations, is conditionally independent of all past states and observations given the immediately previous state. Filtering consists of a prediction step and a correction step. In the context of visual-inertial odometry systems, the prediction step uses IMU observations and the correction step uses visual observations. The Markov assumption enables computationally efficient inference but comes at the cost of lower accuracy due to its inability to model dependencies across longer time intervals.

Extended Kalman Filter

The Extended Kalman Filter (EKF) models the state as a Gaussian distribution and employs first-order linearizations for its prediction and correction models. While particle filters are more suited to handling non-Gaussian state distributions and transition models, their high computational overhead limits them to low-dimensional applications [35]. The main drawbacks of the EKF are its limited ability to handle nonlinearities as well as the implementation difficulty of computing the Jacobians of prediction and correction models.

Unscented Kalman Filter

The Unscented Kalman Filter (UKF) uses a weighted sum of sigma points to approximate the effect of passing Gaussian distributions through nonlinear functions [36]. For an n-dimensional state vector, $2n+1$ sigma points are chosen symmetrically about the mean and individually passed through the nonlinear function before being summed to compute the final mean and covariance. The UKF performs a statistical linearization of nonlinear process and correction model functions, as opposed to the EKF's analytical linearization. The unscented transform used by the UKF is accurate to third order for Gaussian inputs and second order for non-Gaussian inputs [37], giving it superior performance compared with the EKF. Another advantage of the UKF is that jacobian computation is not required.

Multi-State Constraint Kalman Filter

The Multi-State Constraint Kalman Filter's (MSCKF) state vector consists of the states at the times corresponding to a fixed number of the most recent image observations. While the process update is driven by IMU observations, the correction update applies geometric constraints between recent poses based on observations of all feature trails that have disappeared from view in the current image. Unlike standard EKF-SLAM approaches, the MSCKF does not include features in its state vector and thereby achieves greater computational efficiency [30]. The MSCKF's consistency and accuracy can be improved by computing jacobian matrices from the first estimates of each state and estimating IMU-to-camera extrinsic parameters online [38]. Although the MSCKF formulation is computationally cheaper than equivalent optimization-based approaches, it has lower accuracy and is harder to tune [39].

2.3.3 Optimization

Optimization-based approaches estimate motion by solving a minimization problem that involves visual and inertial observations. Because sensor measurement models are highly nonlinear, iterative minimization

techniques are used. Compared with filtering techniques, optimization-based techniques achieve higher accuracy at the cost of more computation. Optimization-based techniques can be classified as full batch smoothing, fixed lag smoothing, and incremental smoothing [40].

Full batch smoothing uses all visual and inertial observations to estimate the entire history of motion states. Although this results in the highest accuracy, problem size growth over time makes it unsuitable for online use on computationally constrained systems. *Fixed lag smoothing* estimates motion for a fixed number of states in a sliding window that extends a limited interval of time into the past. Computational complexity is kept constant by marginalizing old states as new states are added. Although marginalization can cause accuracy losses due to premature fixation of linearization points, fixed lag smoothing is a widely used compromise between the accuracy of full batch smoothing and the low cost of filtering-based methods. *Incremental smoothing* techniques solve for the entire history of states but bound growth in computational complexity by only updating variables affected by new observations. However, the need to store the entire history of states causes unbounded growth in memory usage. Additionally, as the factor graph connecting past states grows with time, new updates will tend to impact larger parts of the graph and drive up the cost of incremental updates.

Commonly used frameworks for optimization-based state estimation include iSAM2 [34], g2o [41], Ceres [42], and SLAM++ [43].

2.3.4 Feature Selection

When using sparse indirect methods on computationally constrained systems, the number of features that can be tracked and used for real-time motion inference is limited by onboard processing power. The fact that not all features are equally beneficial for improving estimation accuracy motivates the development of strategies to select the features that are most useful for reducing uncertainty. For example, far away features are useful for estimating rotation, while nearby features are useful for estimating depth [3, 44]. Despite this, many traditional sparse indirect methods select features to be uniformly distributed across the field of view and have a bias towards more distinctive features [45].

[46] selected features by computing an observability score for each feature and running a greedy selection algorithm. [47] simplified the feature selection problem for two cameras with disjoint fields of view into a problem of allocating a distribution of features between the two cameras. Their selection strategy sought to maximize expected information gain from the features predicted to be visible in each camera. Although their simplifications yielded a computationally tractable quartic equation zero-finding problem, their ad-hoc assumptions on feature distribution and the impact of velocity on information gain limit the method's general applicability. [48] incorporated the effect of predicted sensor motion on expected information gain and proposed a greedy feature selection algorithm. Their information gain model made less restrictive assumptions about the distribution of features in the environment and accounted for stochastic loss of feature trail tracking as well as IMU noise.

2.4 Depth-based Methods

Depth sensors provide 3D information about the environment in the form of range observations to non-occluded surfaces within their fields of view. Depth observations can aid vision-based state estimation by eliminating scale ambiguity. When vision-based techniques fail due to insufficient illumination or visual texture, depth-based techniques can leverage geometric information in the environment to estimate motion. However, depth-based techniques perform poorly in regions with few distinctive geometric features, such as long corridors. Additionally, depth sensors may not provide range observations for all parts of their field of view due to range limits, unfavorable surface reflectance properties, and direct sunlight. Due to these drawbacks, depth sensors are most commonly used alongside other sensors with complementary characteristics in state estimators.

State estimation techniques that utilize both depth and vision need to compensate for spatial and tem-

poral misalignment of image and range observations. For many consumer-grade RGB-D sensors, this task is performed within the sensor itself. In the case of a separate camera and depth sensors without hardware integration, extrinsic and temporal calibration are required to achieve spatial and temporal alignment. Alternatively, hardware triggering can be set up to ensure temporal alignment between the image and depth observations.

Techniques for incorporating depth observations in state estimation can be divided into sparse methods and dense methods.

2.4.1 Sparse Methods

The primary advantage of using sparse depth information is that it incurs lower computational cost than using all the depth information within the scene. One way of extracting sparse data from dense range data is to extract 3D features [49, 50]. However, 3D feature extraction is very computationally expensive.

Due to the ubiquity of RGB-D sensors with hardware-level alignment for RGB and depth images, a computationally efficient way to extract sparse depth information is to use only the depth values associated with features tracked in the RGB image. Given registered RGB (or monochrome) and depth images, sparse vision-based state estimation algorithms can trivially obtain the depth for each visual feature by reading off the pixel value in the corresponding depth image. Depth information fixes the scale of the scene, which makes the motion inference problem easier. This method is used by [51] to obtain two sets of 3D features in successive RGB-D frames, from which relative pose is estimated via minimizing reprojection error. [52] proposes a similar technique that replaces optimization with the Kabsch algorithm [53]. While both [51] and [52] only use features that have associated depth values, [54] also uses features without depth values for greater robustness to scenes with indistinctive geoemtry. [55] uses depth information to project points and lines to 3D space in a visual-inertial odometry framework designed for indoor and urban environments.

2.4.2 Dense Methods

By utilizing all valid range values produced by a depth sensor, dense methods can achieve highly accurate relative pose estimation but incur significant computational cost. Iterative Closest Point (ICP) [56] is an iterative algorithm that aligns point clouds by minimizing distances between corresponding points. Point-to-plane ICP [57] minimizes the distance between points and their corresponding tangent planes in the other point cloud to achieve improved robustness and accuracy. Generalized ICP [58] models point clouds as Gaussian distributions and uses maximum likelihood estimation for alignment. Although ICP-based algorithms are sensitive to initial conditions, they can be used with lower density lidar depth data. To improve convergence, [59] uses visual odometry as the initial guess for ICP.

An alternate depth-only dense method is KinectFusion [60], which represents occupied regions of the environment as a truncated signed distance field and estimates motion by registering incoming point clouds against the model. While KinectFusion only uses depth observations, [61] estimates motion by minimizing a combination of photometric and depth error between pairs of RGB-D frames. [62] also minimizes a similar cost function, but switches to a sparse method [51] when the environment geometry becomes degenerate. [24] models the photometric error with a Student's t-distribution instead of a Gaussian distribution to improve robustness to outliers. [63] uses an inverse depth parametrization to improve the modeling of range error. To improve robustness to lighting changes, [64] jointly estimates frame-to-frame relative pose and the parameters of an affine illumination model. Rather than using both RGB and depth data simultaneously, [65] estimates motion from downsampled depth images using the range change constraint equation and switches to using RGB images in environments with ill-conditioned geometry.

Chapter 3

Sliding Window Visual-Inertial Odometry

This chapter describes a tightly-coupled optimization-based fixed-lag smoother that utilizes IMU and image observations to estimate robot motion (Fig. 3.1). The system's front-end preprocesses IMU and image observations into features (Sect. 3.1) and relative motion constraints (Sect. 3.2), while the back-end performs inference on robot motion, IMU biases, and feature locations via *maximum a posteriori* (MAP) estimation on a factor graph model of preprocessed observations (Sect. 3.4, 3.5, 3.6). An initialization procedure (Sect. 3.7) that aligns IMU-derived and camera-derived motions is used to provide an accurate initial guess for the highly nonlinear optimization solved by the back-end. The state estimation system is evaluated in postprocessing on a publicly available dataset (Sect. 3.8) and in real time on hardware flight experiments (Sect. 3.9).

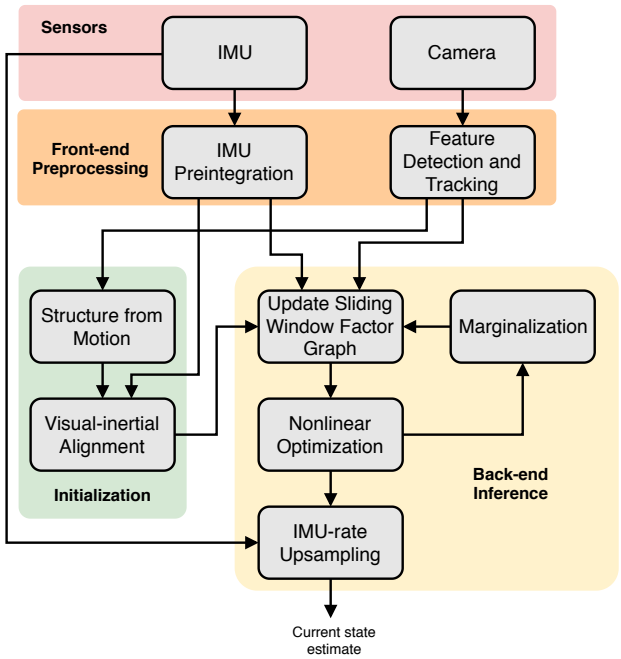


Figure 3.1: Sliding window visual-inertial odometry information flow diagram

3.1 Feature Detection and Tracking

Given a stream of grayscale images from a camera, the vision front-end detects features and tracks them over successive frames. We use the term *frame* both to refer to a camera image observation as well as the

set of feature observations extracted from that image.

3.1.1 Feature Detection

The feature detector finds new Shi-Tomasi corners [11] such that the total number of already tracked and new corner features meets a user-defined threshold. New corners are added in order of decreasing feature quality scores, which are indicative of visual distinctiveness, until the threshold is met. A uniform feature distribution over each image is achieved by enforcing a minimum separation distance between neighboring features.

3.1.2 Feature Tracking

Given a set of features in a previous image frame, the pyramidal Lucas-Kanade method [66] is used to estimate optical flow and find their correspondences in the current image frame. Outlier feature correspondences are discarded using a RANSAC scheme with a fundamental matrix model[1].

Although feature tracking occurs at camera frame rate (∼50 Hz), only a subset of frames are designated as *keyframes* and used in visual-inertial bundle adjustment. Keyframes are selected from incoming image frames in a manner that ensures that keyframe frequency matches optimization frequency (∼10 Hz) as closely as possible (Fig. 3.2). Keyframes that occur in motionless time intervals are managed by the marginalization strategy described in Sect. 3.6.3 to prevent estimation drift.

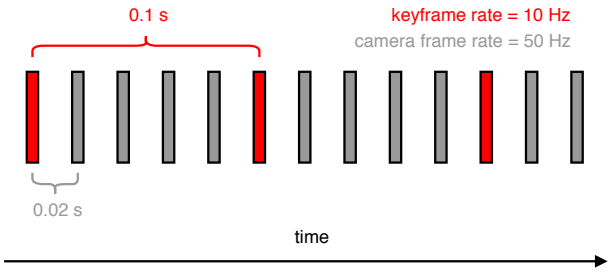


Figure 3.2: An incoming camera frame is selected to be a keyframe if the elapsed time since the previous keyframe is close to the desired time interval between keyframes.

3.1.3 Outlier Rejection

Even after using fundamental matrix RANSAC, feature tracking sometimes produces incorrect matches due to factors such as dynamic objects, shadows, highly repetitive texture, and rapid camera motion. To further improve tracking quality, we use relative rotation priors from the integration of high frequency IMU gyro observations over short time intervals to identify and reject wrongly matched features between two successive camera frames in a 2-point RANSAC scheme [67].

The 2-point RANSAC algorithm operates on a set of potential feature correspondences between frames k and $k+1$. At each iteration, two features are randomly selected and used to compute the up-to-scale translation $\mathbf{t}_{k+1,k}$ between the camera frames. Model quality is determined by the reprojection error of all features in frames k and $k+1$, where the 3D feature locations are triangulated using $\mathbf{t}_{k+1,k}$.

Given two features $\mathbf{p}_a$ and $\mathbf{p}_b$ whose observations appear in successive image frames k and $k+1$, let $\mathbf{u}_{a,k}, \mathbf{u}_{a,k+1}, \mathbf{u}_{b,k}, \mathbf{u}_{b,k+1}$ be the unit direction vectors associated with their observations in each camera frame (Figure 3.3). The rotation $\mathbf{R}_{k,k+1}$ takes vectors from the reference frame of the camera at time $k+1$ to the reference frame of the camera at time k.

$$\mathbf{R}_{k,k+1} = \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}_{\omega}(t_k, t_{k+1}) \mathbf{R}_{bc} \tag{3.1}$$

[1] If the camera's intrinsic parameters are known, the essential matrix model can be used in RANSAC to reduce the degrees of freedom from 7 to 5.

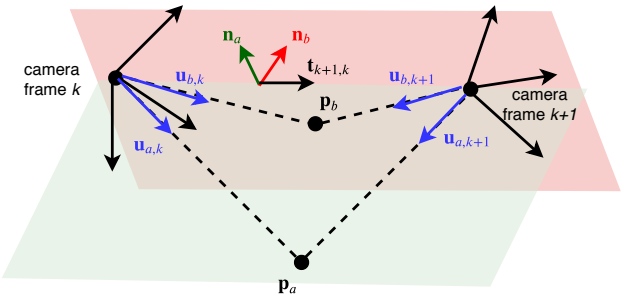


Figure 3.3: Geometry for determining up-to-scale translation between camera frames k and $k+1$

$\mathbf{R}_{bc}$ is the extrinsic rotation matrix that takes vectors from the camera frame to the body frame (assumed without loss of generality to be coincident with the IMU frame). $\mathbf{R}_\omega(t_k, t_{k+1})$ is the rotation that takes vectors from the IMU frame at time t_{k+1} to the IMU frame at t_k. Let $\{t_i, \boldsymbol{\omega}_i\}_{i=1}^n$, $t_k = t_1$, $t_{k+1} = t_n$ be a set of timestamped IMU gyro observations taken during this time interval. The quaternion representation of $\mathbf{R}_\omega(t_k, t_{k+1})$ is approximated as

$$\mathbf{q}_{k,k+1} \approx \begin{bmatrix} 1 \\ \frac{1}{2}(t_2 - t_1)\boldsymbol{\omega}_2 \end{bmatrix} \otimes \cdots \otimes \begin{bmatrix} 1 \\ \frac{1}{2}(t_n - t_{n-1})\boldsymbol{\omega}_n \end{bmatrix} \tag{3.2}$$

using Euler integration.

The normal vectors of the epipolar planes associated with the features are given by

$$\mathbf{n}_a = \mathbf{u}_{a,k} \times \mathbf{R}_{k,k+1}\mathbf{u}_{a,k+1} \tag{3.3}$$

$$\mathbf{n}_b = \mathbf{u}_{b,k} \times \mathbf{R}_{k,k+1}\mathbf{u}_{b,k+1} \tag{3.4}$$

The intersection of the pair of epipolar planes yields the up-to-scale translation between camera frames.

$$\mathbf{t}_{k+1,k} = \mathbf{n}_a \times \mathbf{n}_b \tag{3.5}$$

Figure 3.4 depicts the application of 2-point RANSAC to reject outliers caused by moving shadows on downward images taken from an aerial robot during takeoff.

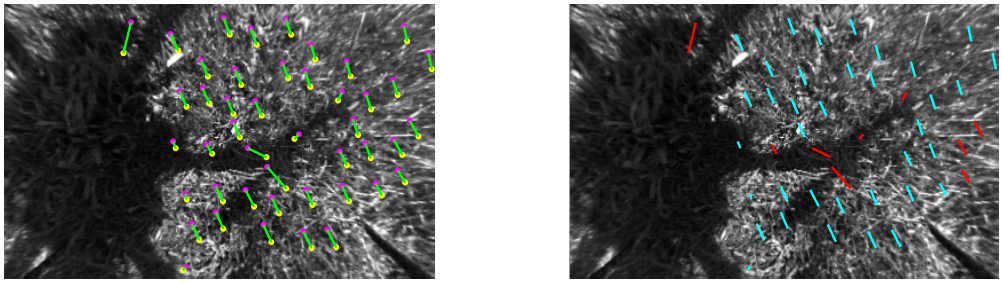


Figure 3.4: 2 point RANSAC can detect outlier feature correspondences caused by moving shadows. On the left image, magenta dots represent features from the previous image frame, yellow dots represent their tracked locations in the current image frame, and green lines represent correspondences. On the right image, 2 point RANSAC has been used to classify the set of all correspondences into a set of inliers (cyan) and outliers (red).

3.1.4 Implementation Details

- Contrast limited adaptive histogram equalization [68] is applied to grayscale images to improve their contrast prior to being used in feature detection or tracking.
- Optical flow pyramids are pre-built to speed up Lucas-Kanade tracking.
- A mask is used to prevent features from being detected or tracked in regions of the image that are occluded by parts of the same rigid body the camera is mounted on (e.g. landing gear or propellers in the case of an aerial robot).
- When two tracked features are closer than a user-defined minimum separation distance, the one that has been tracked for more frames is retained while the one that has been tracked for fewer frames is discarded.

3.2 IMU Preintegration

IMU preintegration is a method of summarizing multiple linear acceleration and angular velocity observations into relative motion constraints between robot states at different points in time. If IMU observations are directly incorporated into a factor graph, each new observation adds a new node and edge. Since IMU observations are typically available at frequencies of greater than 100 Hz, such a strategy would result in a large factor graph and a costly optimization.

IMU preintegration accumulates successive IMU observations into a single pseudo-observation, reducing the number of nodes and edges required to represent observations within the same temporal interval. Instead of adding a new node whenever a new IMU observation arrives, IMU preintegration makes it possible for a new node to be added only when a new keyframe arrives from the feature tracker.

IMU preintegration was first introduced in [69], which used Euler angles to parameterize relative rotations. [70] reformulated IMU preintegration on the $SO(3)$ manifold to remove singularities. Building on this foundation, rotation was parametrized by quaternions instead of rotation matrices in [5]. Discrete-time integration of IMU measurements was replaced with closed-form continuous time integration in [71], but this change only makes a performance impact for highly dynamic motions.

3.2.1 IMU Measurement Model

Let $\{w\}$ be the world frame and $\{b\}$ be the body frame. We assume that the IMU is located at the body frame origin and has the same orientation as the body frame[2]. We use a simplified version of the body frame IMU measurement model described in [72].

$$\hat{\mathbf{a}}_b = \text{diag}(\mathbf{s}_a)\, \mathbf{M}_a \mathbf{R}_{wb}^{\mathrm{T}} \left(\mathbf{a}_w - \mathbf{g}_w\right) + \mathbf{b}_a + \mathbf{n}_a \tag{3.6}$$

$$\hat{\boldsymbol{\omega}}_b = \text{diag}(\mathbf{s}_\omega)\, \mathbf{M}_\omega \boldsymbol{\omega}_b + \mathbf{b}_\omega + \mathbf{n}_\omega \tag{3.7}$$

The $\text{diag}(\cdot)$ function turns a 3×1 vector into a 3×3 diagonal matrix. The measured body frame linear acceleration ($\hat{\mathbf{a}}_b$) captures the sum of true world frame linear acceleration ($\mathbf{a}_w$) and negative gravity (-$\mathbf{g}_w$) rotated into the body frame. $\mathbf{s}_a \in \mathbb{R}^3$ is a vector of accelerometer axes scale factors, while $\mathbf{M}_a \in \mathbb{R}^{3\times3}$ is a lower triangular matrix that represents accelerometer axis misalignment and coupling. The measured body frame linear acceleration is corrupted by additive bias ($\mathbf{b}_a$) and noise ($\mathbf{n}_a$). The measured body frame angular velocity ($\hat{\boldsymbol{\omega}}_b$) is the sum of true body angular velocity ($\boldsymbol{\omega}_b$), bias ($\mathbf{b}_\omega$), and noise ($\mathbf{n}_a$). $\mathbf{s}_\omega \in \mathbb{R}^3$ is a vector of gyroscope axes scale factors, while $\mathbf{M}_\omega \in \mathbb{R}^{3\times3}$ is a matrix that represents gyroscope axis misalignment and coupling. Note that $\mathbf{M}_a$ is lower triangular because we assume that the accelerometer's x-axis is aligned with the IMU's input reference axes. On the other hand, $\mathbf{M}_\omega$ is a full matrix in order to capture the full misalignment between the gyroscopes and accelerometers.

[2]If this is not the case, Sect. 3.2.4 describes how to compensate for the offset

Unlike [72], we do not model separate displacements between each accelerometer and gyroscope axes and the body frame. We do not model g-sensitivity, which captures the impact of accelerometer observations on gyroscope observations, because its effects are small relative to axis scaling and misalignment.

To minimize the number of parameters in the IMU model, we define combined scale and misalignment matrices

$$\mathbf{L}_a = \left(\text{diag}(\mathbf{s}_a)\ \mathbf{M}_a\right)^{-1} \tag{3.8}$$

$$\mathbf{L}_\omega = \left(\text{diag}(\mathbf{s}_\omega)\ \mathbf{M}_\omega\right)^{-1} \tag{3.9}$$

where $\mathbf{L}_a$ is lower diagonal and $\mathbf{L}_\omega$ is a full matrix.

$$\mathbf{L} = \begin{bmatrix} L_{a,xx} & 0 & 0 \\ L_{a,yx} & L_{a,yy} & 0 \\ L_{a,zx} & L_{a,zy} & L_{a,zz} \end{bmatrix}, \quad \mathbf{L}_\omega = \begin{bmatrix} L_{\omega,xx} & L_{\omega,xy} & L_{\omega,xz} \\ L_{\omega,yx} & L_{\omega,yy} & L_{\omega,yz} \\ L_{\omega,zx} & L_{\omega,yz} & L_{\omega,zz} \end{bmatrix} \tag{3.10}$$

Note that IMUs without scale or misalignment errors can be modeled with $\mathbf{L}_a = \mathbf{L}_\omega = \mathbf{I}$. Substitute (3.8)-(3.9) into (3.6)-(3.7) and isolate world frame coordinate acceleration and body frame angular velocity.

$$\mathbf{a}_w = \mathbf{R}_{wb}\mathbf{L}_a\left(\hat{\mathbf{a}}_b - \mathbf{b}_a - \mathbf{n}_a\right) + \mathbf{g}_w \tag{3.11}$$

$$\boldsymbol{\omega}_b = \mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_b - \mathbf{b}_\omega - \mathbf{n}_\omega\right) \tag{3.12}$$

Accelerometer and gyroscope additive noise terms are modeled as isotropic Gaussian white noise

$$\mathbf{n}_a \sim \mathcal{N}\left(\mathbf{0}, \sigma_{na}^2\mathbf{I}\right) \tag{3.13}$$

$$\mathbf{n}_\omega \sim \mathcal{N}\left(\mathbf{0}, \sigma_{n\omega}^2\mathbf{I}\right) \tag{3.14}$$

Linear acceleration and angular velocity biases are modeled as random walks whose derivatives are isotropic Gaussian white noise

$$\mathbf{n}_{ba} \sim \mathcal{N}\left(\mathbf{0}, \sigma_{ba}^2\mathbf{I}\right) \tag{3.15}$$

$$\mathbf{n}_{b\omega} \sim \mathcal{N}\left(\mathbf{0}, \sigma_{b\omega}^2\mathbf{I}\right) \tag{3.16}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{ba} \tag{3.17}$$

$$\dot{\mathbf{b}}_\omega = \mathbf{n}_{b\omega} \tag{3.18}$$

The additive noise intensities σ_{na} and $\sigma_{n\omega}$ can be found empirically by taking the standard deviation of detrended in-flight IMU observations. The random walk noise intensities σ_{ba} and $\sigma_{b\omega}$ can be found empirically by performing Allan variance analysis on in-flight IMU observations.

3.2.2 Motion Integration

Global Frame

World frame linear acceleration and body frame angular velocity are related to world frame position, velocity, and attitude by the following kinematic model:

$$\dot{\mathbf{p}}_w = \mathbf{v}_w \tag{3.19}$$

$$\dot{\mathbf{v}}_w = \mathbf{a}_w \tag{3.20}$$

$$\dot{\mathbf{q}}_{wb} = \frac{1}{2}\mathbf{q}_{wb} \otimes \begin{bmatrix} 0 \\ \boldsymbol{\omega}_b \end{bmatrix} \tag{3.21}$$

The quaternion time derivative (3.21) is derived from first principles in equations 195-199 of [73]. Euler integration of (3.19)-(3.21) over a time interval Δt yields

$$\mathbf{p}_w(t+\Delta t) = \mathbf{p}_w(t) + \mathbf{v}_w(t)\Delta t + \frac{1}{2}\mathbf{a}_w(t)\Delta t^2 \tag{3.22}$$

$$\mathbf{v}_w(t+\Delta t) = \mathbf{v}_w(t) + \mathbf{a}_w(t)\Delta t \tag{3.23}$$

$$\mathbf{q}_{wb}(t+\Delta t) = \mathbf{q}_{wb}(t) \otimes \text{Exp}\left(\boldsymbol{\omega}_b \Delta t\right) \tag{3.24}$$

Note that (3.24) uses zeroth order forward quaternion integration, which is derived from (3.21) in equations 210-214 of [73].

The equations used to update position, velocity, and attitude with IMU observations are obtained by substituting (3.11) -(3.12) into (3.22)-(3.24).

$$\mathbf{p}_w(t+\Delta t) = \mathbf{p}_w(t) + \mathbf{v}_w(t)\Delta t + \frac{1}{2}\left[\mathbf{R}(\mathbf{q}_{wb}(t))\mathbf{L}_a\left(\hat{\mathbf{a}}_b(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)\right) + \mathbf{g}_w\right]\Delta t^2 \tag{3.25}$$

$$\mathbf{v}_w(t+\Delta t) = \mathbf{v}_w(t) + \left[\mathbf{R}(\mathbf{q}_{wb}(t))\mathbf{L}_a\left(\hat{\mathbf{a}}_b(t) - \mathbf{b}_a(t) - \mathbf{n}_a(t)\right) + \mathbf{g}_w\right]\Delta t \tag{3.26}$$

$$\mathbf{q}_{wb}(t+\Delta t) = \mathbf{q}_{wb}(t) \otimes \text{Exp}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_b(t) - \mathbf{b}_\omega(t) - \mathbf{n}_\omega(t)\right)\Delta t\right] \tag{3.27}$$

Suppose we have a set of IMU observations $\{\mathbf{a}_i, \boldsymbol{\omega}_i\}_{i=0}^n$ corresponding to times $t_0, t_1, ..., t_n$. We apply (3.25)-(3.27) n times to an initial motion state $(\mathbf{p}_0, \mathbf{v}_0, \mathbf{q}_0)$ to find the final motion state.

$$\mathbf{p}_n = \mathbf{p}_0 + \frac{1}{2}\mathbf{g}_w\Delta T^2 + \sum_{i=0}^{n-1}\left[\mathbf{v}_i\Delta t_i + \frac{1}{2}\mathbf{R}(\mathbf{q}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \mathbf{b}_{a,i} - \mathbf{n}_{a,i}\right)\Delta t_i^2\right] \tag{3.28}$$

$$\mathbf{v}_n = \mathbf{v}_0 + \mathbf{g}_w\Delta T + \sum_{i=0}^{n-1}\mathbf{R}(\mathbf{q}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \mathbf{b}_{a,i} - \mathbf{n}_{a,i}\right)\Delta t_i \tag{3.29}$$

$$\mathbf{q}_n = \mathbf{q}_0 \otimes \text{Exp}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_0 - \mathbf{b}_{\omega,0} - \mathbf{n}_{\omega,0}\right)\Delta t_0\right] \otimes \ldots \otimes \text{Exp}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_{n-1} - \mathbf{b}_{\omega,n-1} - \mathbf{n}_{\omega,n-1}\right)\Delta t_{n-1}\right] \tag{3.30}$$

Note that in the above motion state propagation equations we drop the frame subscripts because they are clear from context and replace them with time index subscripts, where $(\cdot)_i \equiv (\cdot)(t_i)$. Also, we define $\Delta t_i \equiv t_{i+1} - t_i$ and $\Delta T \equiv t_n - t_0$.

Local Frame

Although equations (3.28)-(3.30) can be used in optimization, they must be reevaluated whenever the initial motion state changes. To avoid this computational cost, define *motion deltas* by rearranging (3.28)-(3.30) to separate motion states and gravity from quantities that only depend on IMU observations, biases, noise, and time increments.

$$\boldsymbol{\gamma}_n \equiv \mathbf{q}_0^{-1} \otimes \mathbf{q}_n = \text{Exp}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_0 - \mathbf{b}_{\omega,0} - \mathbf{n}_{\omega,0}\right)\Delta t_0\right] \otimes \ldots \otimes \text{Exp}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_{n-1} - \mathbf{b}_{\omega,n-1} - \mathbf{n}_{\omega,n-1}\right)\Delta t_{n-1}\right] \tag{3.31}$$

$$\boldsymbol{\beta}_n \equiv \mathbf{R}(\mathbf{q}_0)^{\text{T}}\left(\mathbf{v}_n - \mathbf{v}_0 - \mathbf{g}_w\Delta T\right) = \sum_{i=0}^{n-1}\mathbf{R}(\boldsymbol{\gamma}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \mathbf{b}_{a,i} - \mathbf{n}_{a,i}\right)\Delta t_i \tag{3.32}$$

$$\boldsymbol{\alpha}_n \equiv \mathbf{R}(\mathbf{q}_0)^{\text{T}}\left(\mathbf{p}_n - \mathbf{p}_0 - \mathbf{v}_0\Delta T - \frac{1}{2}\mathbf{g}_w\Delta T^2\right) = \sum_{i=0}^{n}\left[\boldsymbol{\beta}_i\Delta t_i + \frac{1}{2}\mathbf{R}(\boldsymbol{\gamma}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \mathbf{b}_{a,i} - \mathbf{n}_{a,i}\right)\Delta t_i^2\right] \tag{3.33}$$

The motion deltas can be updated recursively with IMU observations starting from $\boldsymbol{\alpha}_0 = \boldsymbol{\beta}_0 = \mathbf{0}$ and $\boldsymbol{\gamma}_0 = [1\ 0\ 0\ 0]^{\mathrm{T}}$.

$$\boldsymbol{\alpha}_{i+1} = \boldsymbol{\alpha}_i + \boldsymbol{\beta}_i \Delta t_i + \frac{1}{2}\mathbf{R}(\boldsymbol{\gamma}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \mathbf{b}_{a,i} - \mathbf{n}_{a,i}\right)\Delta t_i^2 \tag{3.34}$$

$$\boldsymbol{\beta}_{i+1} = \boldsymbol{\beta}_i + \mathbf{R}(\boldsymbol{\gamma}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \mathbf{b}_{a,i} - \mathbf{n}_{a,i}\right)\Delta t_i \tag{3.35}$$

$$\boldsymbol{\gamma}_{i+1} = \boldsymbol{\gamma}_i \otimes \mathrm{Exp}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_i - \mathbf{b}_{\omega,i} - \mathbf{n}_{\omega,i}\right)\Delta t_i\right] \tag{3.36}$$

Although the motion deltas no longer depend on motion states, they still depend on the IMU biases. To avoid repropagating (3.34)-(3.36) when IMU bias estimates change during optimization, we approximate the effect of IMU bias changes on the motion deltas to first order. First order approximation requires the Jacobian of motion deltas with respect to IMU bias perturbations, which can be derived using a motion delta error dynamics model.

3.2.3 Motion Delta Error Dynamics Model

The motion delta error dynamics model is an integral part of an error state Kalman filter (ESKF). The ESKF state is called the true state and can be divided into nominal and error states. The nominal state incorporates raw observations, while the error state accounts for perturbations and noise. The nominal state is a large signal that evolves in a nonlinear manner, while the error state is a small signal that evolves as a time-varying linear dynamic system suitable for Kalman filtering.

In the context of IMU preintegration, the ESKF's true state vector is $\mathbf{x} = [\boldsymbol{\alpha}^{\mathrm{T}}\ \boldsymbol{\beta}^{\mathrm{T}}\ \boldsymbol{\gamma}^{\mathrm{T}}\ \mathbf{b}_a^{\mathrm{T}}\ \mathbf{b}_\omega^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{16}$ and its nominal state vector is $\bar{\mathbf{x}} = [\bar{\boldsymbol{\alpha}}^{\mathrm{T}}\ \bar{\boldsymbol{\beta}}^{\mathrm{T}}\ \bar{\boldsymbol{\gamma}}^{\mathrm{T}}\ \bar{\mathbf{b}}_a^{\mathrm{T}}\ \bar{\mathbf{b}}_\omega^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{16}$. Each parameter block in $\mathbf{x}$ is expressed as the composition of a nominal state $\bar{(\cdot)}$ and a perturbation $\delta(\cdot)$.

$$\boldsymbol{\alpha} = \bar{\boldsymbol{\alpha}} + \delta\boldsymbol{\alpha} \tag{3.37}$$

$$\boldsymbol{\beta} = \bar{\boldsymbol{\beta}} + \delta\boldsymbol{\beta} \tag{3.38}$$

$$\boldsymbol{\gamma} = \bar{\boldsymbol{\gamma}} \otimes \delta\boldsymbol{\gamma} \tag{3.39}$$

$$\mathbf{b}_a = \bar{\mathbf{b}}_a + \delta\mathbf{b}_a \tag{3.40}$$

$$\mathbf{b}_\omega = \bar{\mathbf{b}}_\omega + \delta\mathbf{b}_\omega \tag{3.41}$$

Define the rotational perturbation $\delta\boldsymbol{\theta} \in \mathbb{R}^3$ via the capitalized quaternion logarithmic map (A.29) and use it in place of the quaternion $\delta\boldsymbol{\gamma}$ to ensure that the error state has a minimal parameterization.

$$\delta\boldsymbol{\theta} = \mathrm{Log}\,\delta\boldsymbol{\gamma} \tag{3.42}$$

The IMU preintegration ESKF's error state vector is $\delta\mathbf{x} = [\delta\boldsymbol{\alpha}^{\mathrm{T}}\ \delta\boldsymbol{\beta}^{\mathrm{T}}\ \delta\boldsymbol{\theta}^{\mathrm{T}}\ \delta\mathbf{b}_a^{\mathrm{T}}\ \delta\mathbf{b}_\omega^{\mathrm{T}}]^{\mathrm{T}} \in \mathbb{R}^{15}$.

True State Continuous Time Dynamics

The continuous time dynamics of the motion deltas are given by the continuous time differential forms of (3.34)-(3.36), while the IMU bias dynamics are restated from (3.17)-(3.18).

$$\dot{\boldsymbol{\alpha}} = \boldsymbol{\beta} \tag{3.43}$$

$$\dot{\boldsymbol{\beta}} = \mathbf{R}(\boldsymbol{\gamma})\mathbf{L}_a\left(\hat{\mathbf{a}} - \mathbf{b}_a - \mathbf{n}_a\right) \tag{3.44}$$

$$\dot{\boldsymbol{\gamma}} = \frac{1}{2}\boldsymbol{\gamma} \otimes \begin{bmatrix} 0 \\ \mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \mathbf{b}_\omega - \mathbf{n}_\omega\right) \end{bmatrix} \tag{3.45}$$

$$\dot{\mathbf{b}}_a = \mathbf{n}_{ba} \tag{3.46}$$

$$\dot{\mathbf{b}}_\omega = \mathbf{n}_{b\omega} \tag{3.47}$$

Nominal State Continuous Time Dynamics

The nominal state dynamics are obtained by substituting (3.37)-(3.42) into (3.43)-(3.47) and setting all error state and noise terms to zero.

$$\dot{\bar{\boldsymbol{\alpha}}} = \bar{\boldsymbol{\beta}} \tag{3.48}$$

$$\dot{\bar{\boldsymbol{\beta}}} = \mathbf{R}(\bar{\boldsymbol{\gamma}})\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a\right) \tag{3.49}$$

$$\dot{\bar{\boldsymbol{\gamma}}} = \frac{1}{2}\bar{\boldsymbol{\gamma}} \otimes \begin{bmatrix} 0 \\ \mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right) \end{bmatrix} \tag{3.50}$$

$$\dot{\bar{\mathbf{b}}}_a = \mathbf{0} \tag{3.51}$$

$$\dot{\bar{\mathbf{b}}}_\omega = \mathbf{0} \tag{3.52}$$

Nominal State Discrete Time Dynamics

The nominal motion delta states can be updated recursively with IMU observations starting from $\bar{\boldsymbol{\alpha}}_0 = \bar{\boldsymbol{\beta}}_0 = \mathbf{0}$ and $\bar{\boldsymbol{\gamma}}_0 = [1\ 0\ 0\ 0]^{\mathrm{T}}$.

$$\bar{\boldsymbol{\alpha}}_{i+1} = \bar{\boldsymbol{\alpha}}_i + \bar{\boldsymbol{\beta}}_i \Delta t_i + \frac{1}{2}\mathbf{R}(\bar{\boldsymbol{\gamma}}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \bar{\mathbf{b}}_{a,i}\right)\Delta t_i^2 \tag{3.53}$$

$$\bar{\boldsymbol{\beta}}_{i+1} = \bar{\boldsymbol{\beta}}_i + \mathbf{R}(\bar{\boldsymbol{\gamma}}_i)\mathbf{L}_a\left(\hat{\mathbf{a}}_i - \bar{\mathbf{b}}_{a,i}\right)\Delta t_i \tag{3.54}$$

$$\bar{\boldsymbol{\gamma}}_{i+1} = \bar{\boldsymbol{\gamma}}_i \otimes \mathrm{Exp}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}_i - \bar{\mathbf{b}}_{\omega,i}\right)\Delta t_i\right] \tag{3.55}$$

Error State Continuous Time Dynamics

Substitute (3.37)-(3.42) into (3.43)-(3.47), subtract the nominal dynamics equations (3.48)-(3.52), linearize rotations, and ignore second-order terms to obtain

$$\delta\dot{\mathbf{x}} = \mathbf{F}_c\delta\mathbf{x} + \mathbf{G}_c\mathbf{n} \tag{3.56}$$

$$\mathbf{F}_c = \begin{bmatrix} \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\mathbf{R}(\bar{\boldsymbol{\gamma}})\left[\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a\right)\right]_\times & -\mathbf{R}(\bar{\boldsymbol{\gamma}})\mathbf{L}_a & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & -\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times & \mathbf{0} & -\mathbf{L}_\omega \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \end{bmatrix} \tag{3.57}$$

$$\mathbf{G}_c = \begin{bmatrix} \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ -\mathbf{R}(\bar{\boldsymbol{\gamma}})\mathbf{L}_a & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & -\mathbf{L}_\omega & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \tag{3.58}$$

where $\mathbf{n} = \begin{bmatrix} \mathbf{n}_a^{\mathrm{T}} & \mathbf{n}_\omega^{\mathrm{T}} & \mathbf{n}_{ba}^{\mathrm{T}} & \mathbf{n}_{b\omega}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{12}$ is a concatenated vector of noise terms defined in (3.13)-(3.16). See Appendix B.1 for a full derivation of the continuous time error state dynamics.

Error State Discrete Time Dynamics

Obtain an approximation of the discrete time error dynamics by applying Euler integration[3] to the continuous time error dynamics (3.56)-(3.58).

$$\delta\mathbf{x}_{k+1} = \mathbf{F}_k\delta\mathbf{x}_k + \mathbf{G}_k\mathbf{n}_k \tag{3.59}$$

[3] See Appendix B.2.2 for an approximation of the discrete time error dynamics using midpoint intergration.

$$\mathbf{F}_k = \mathbf{I} + \Delta t_k \mathbf{F}_c(t_k) \tag{3.60}$$
$$\mathbf{G}_k = \Delta t_k \mathbf{G}_c(t_k) \tag{3.61}$$

Rewrite (3.59) by expressing $\{\delta\mathbf{x}_i\}_{i=1}^{k}$ in terms of $\delta\mathbf{x}_0$ to identify the Jacobian ($\mathbf{J}_k$) of the motion delta error states with respect to their original values.

$$\delta\mathbf{x}_{k+1} = \left(\prod_{i=0}^{k} \mathbf{F}_i\right) \delta\mathbf{x}_0 + \dots \tag{3.62}$$
$$\mathbf{J}_k = \prod_{i=0}^{k} \mathbf{F}_i \tag{3.63}$$

The discrete time covariance is updated recursively:

$$\mathbf{P}_{k+1} = \mathbf{F}_k \mathbf{P}_k \mathbf{F}_k^{\mathrm{T}} + \mathbf{G}_k \mathbf{Q} \mathbf{G}_k^{\mathrm{T}} \tag{3.64}$$
$$\mathbf{Q} = \text{blkdiag}\{\sigma_{na}^2 \mathbf{I}, \sigma_{n\omega}^2 \mathbf{I}, \sigma_{ba}^2 \mathbf{I}, \sigma_{b\omega}^2 \mathbf{I}\} \tag{3.65}$$
$$\mathbf{P}_0 = \mathbf{0} \tag{3.66}$$

IMU Bias Perturbation Model

The first order approximation of the perturbed delta motion states at the end of the preintegration interval with respect to perturbations in IMU bias states at the beginning of the preintegration interval can be obtained by substituting (3.63) into (3.62) and extracting relevant sub-blocks of $\mathbf{J}_n$.

$$\delta\mathbf{x}_n = \mathbf{J}_n \delta\mathbf{x}_0 + \dots \tag{3.67}$$
$$\begin{bmatrix} \delta\boldsymbol{\alpha}_n \\ \delta\boldsymbol{\beta}_n \\ \delta\boldsymbol{\theta}_n \\ \delta\mathbf{b}_{a,n} \\ \delta\mathbf{b}_{\omega,n} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{\alpha\alpha} & \mathbf{J}_{\alpha\beta} & \mathbf{J}_{\alpha\theta} & \mathbf{J}_{\alpha b_a} & \mathbf{J}_{\alpha b_\omega} \\ \mathbf{J}_{\beta\alpha} & \mathbf{J}_{\beta\beta} & \mathbf{J}_{\beta\theta} & \mathbf{J}_{\beta b_a} & \mathbf{J}_{\beta b_\omega} \\ \mathbf{J}_{\theta\alpha} & \mathbf{J}_{\theta\beta} & \mathbf{J}_{\theta\theta} & \mathbf{J}_{\theta b_a} & \mathbf{J}_{\theta b_\omega} \\ \mathbf{J}_{b_a\alpha} & \mathbf{J}_{b_a\beta} & \mathbf{J}_{b_a\theta} & \mathbf{J}_{b_a b_a} & \mathbf{J}_{b_a b_\omega} \\ \mathbf{J}_{b_\omega\alpha} & \mathbf{J}_{b_\omega\beta} & \mathbf{J}_{b_\omega\theta} & \mathbf{J}_{b_\omega b_a} & \mathbf{J}_{b_\omega b_\omega} \end{bmatrix} \begin{bmatrix} \delta\boldsymbol{\alpha}_0 \\ \delta\boldsymbol{\beta}_0 \\ \delta\boldsymbol{\theta}_0 \\ \delta\mathbf{b}_{a,0} \\ \delta\mathbf{b}_{\omega,0} \end{bmatrix} + \dots \tag{3.68}$$
$$\delta\boldsymbol{\alpha}_n \approx \mathbf{J}_{\alpha b_a} \delta\mathbf{b}_{a,0} + \mathbf{J}_{\alpha b_\omega} \delta\mathbf{b}_{w,0} + \dots \tag{3.69}$$
$$\delta\boldsymbol{\beta}_n \approx \mathbf{J}_{\beta b_a} \delta\mathbf{b}_{a,0} + \mathbf{J}_{\beta b_\omega} \delta\mathbf{b}_{w,0} + \dots \tag{3.70}$$
$$\delta\boldsymbol{\theta}_n \approx \mathbf{J}_{\theta b_\omega} \delta\mathbf{b}_{\omega,0} + \dots \tag{3.71}$$

Note that $\mathbf{J}_{\theta b_a} = \mathbf{0}$ because the linear acceleration bias does not affect rotation. Let the IMU bias perturbations represent the difference between the current IMU bias estimate at the start of the preintegration time interval $(\mathbf{b}_{a,0}, \mathbf{b}_{\omega,0})$ and the initial IMU bias estimate at the start of the preintegration time interval $(\bar{\mathbf{b}}_{a,0}, \bar{\mathbf{b}}_{\omega,0})$ used to compute the nominal motion delta states $\bar{\boldsymbol{\alpha}}_n$, $\bar{\boldsymbol{\beta}}_n$, and $\bar{\boldsymbol{\gamma}}_n$.

$$\delta\mathbf{b}_{a,0} = \mathbf{b}_{a,0} - \bar{\mathbf{b}}_{a,0} \tag{3.72}$$
$$\delta\mathbf{b}_{\omega,0} = \mathbf{b}_{\omega,0} - \bar{\mathbf{b}}_{\omega,0} \tag{3.73}$$

Let the final motion delta perturbations represent the difference between the motion deltas evaluated at the current IMU bias estimate and the motion deltas evaluated at the initial IMU bias estimate.

$$\delta\boldsymbol{\alpha}_n = \boldsymbol{\alpha}'_n\left(\mathbf{b}_{a,0}, \mathbf{b}_{\omega,0}\right) - \bar{\boldsymbol{\alpha}}_n\left(\bar{\mathbf{b}}_{a,0}, \bar{\mathbf{b}}_{\omega,0}\right) \tag{3.74}$$
$$\delta\boldsymbol{\beta}_n = \boldsymbol{\beta}'_n\left(\mathbf{b}_{a,0}, \mathbf{b}_{\omega,0}\right) - \bar{\boldsymbol{\beta}}_n\left(\bar{\mathbf{b}}_{a,0}, \bar{\mathbf{b}}_{\omega,0}\right) \tag{3.75}$$
$$\delta\boldsymbol{\theta}_n = \text{Log}\left[\bar{\boldsymbol{\gamma}}_n\left(\bar{\mathbf{b}}_{\omega,0}\right)^{-1} \otimes \boldsymbol{\gamma}'_n\left(\mathbf{b}_{\omega,0}\right)\right] \tag{3.76}$$

The first order approximation of changes in motion deltas due to changes in initial IMU bias estimates is obtained by substituting (3.72)-(3.76) into (3.69)-(3.71).

$$\boldsymbol{\alpha}'_n\left(\mathbf{b}_{a,0}, \mathbf{b}_{\omega,0}\right) \approx \bar{\boldsymbol{\alpha}}_n\left(\bar{\mathbf{b}}_{a,0}, \bar{\mathbf{b}}_{\omega,0}\right) + \mathbf{J}_{\alpha b_a}\left(\mathbf{b}_{a,0} - \bar{\mathbf{b}}_{a,0}\right) + \mathbf{J}_{\alpha b_\omega}\left(\mathbf{b}_{\omega,0} - \bar{\mathbf{b}}_{\omega,0}\right) \tag{3.77}$$

$$\boldsymbol{\beta}'_n\left(\mathbf{b}_{a,0}, \mathbf{b}_{\omega,0}\right) \approx \bar{\boldsymbol{\beta}}_n\left(\bar{\mathbf{b}}_{a,0}, \bar{\mathbf{b}}_{\omega,0}\right) + \mathbf{J}_{\beta b_a}\left(\mathbf{b}_{a,0} - \bar{\mathbf{b}}_{a,0}\right) + \mathbf{J}_{\beta b_\omega}\left(\mathbf{b}_{\omega,0} - \bar{\mathbf{b}}_{\omega,0}\right) \tag{3.78}$$

$$\boldsymbol{\gamma}'_n\left(\mathbf{b}_{\omega,0}\right) \approx \bar{\boldsymbol{\gamma}}_n\left(\bar{\mathbf{b}}_{\omega,0}\right) \otimes \mathrm{Exp}\left[\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{\omega,0} - \bar{\mathbf{b}}_{\omega,0}\right)\right] \tag{3.79}$$

3.2.4 Implementation Details

Body to IMU Frame Offset Compensation

An IMU observation consists of linear acceleration ($\hat{\mathbf{a}}_i$) and angular velocity ($\hat{\boldsymbol{\omega}}_i$) measured in the IMU frame $\{i\}$. If the IMU frame is located at a known rigid offset from the body frame $\{b\}$, we use the observation that would be measured by a virtual IMU coincident with the body frame for state estimation. Let $\mathbf{p}_{bi}$ be the IMU's location in the body frame and $\mathbf{R}_{bi}$ be the IMU frame's orientation with respect to the body frame[4]. According to rigid body kinematics, the virtual IMU body frame linear acceleration is

$$\hat{\mathbf{a}}_b = \mathbf{R}_{bi}\left[\hat{\mathbf{a}}_i + \dot{\hat{\boldsymbol{\omega}}}_i \times \mathbf{p}_{ib} + \hat{\boldsymbol{\omega}}_i \times (\hat{\boldsymbol{\omega}}_i \times \mathbf{p}_{ib})\right] \tag{3.80}$$

where $\mathbf{p}_{ib} = -\mathbf{R}_{bi}^{\mathrm{T}}\mathbf{p}_{bi}$. Note that IMU frame angular acceleration ($\dot{\hat{\boldsymbol{\omega}}}_i$) required by (3.80) can be obtained by numerical differentiation of IMU frame angular velocity, but care must be taken to mitigate noise amplification. The virtual IMU body frame angular velocity is computed as

$$\hat{\boldsymbol{\omega}}_b = \mathbf{R}_{bi}\hat{\boldsymbol{\omega}}_i \tag{3.81}$$

Although virtual body frame IMU observations (3.80) and (3.81) can be substituted for true body frame IMU observations, linear acceleration additive noise and bias rate noise intensities should be increased to capture the noise introduced by numerical differentiation. If IMU noise is not isotropic, both linear acceleration and angular velocity noise covariance matrices should be transformed using $\mathbf{R}_{bi}$.

Approximations

We assume that the time interval Δt between successive IMU observations is small, which causes delta rotations $\boldsymbol{\omega}_b \Delta t$ to be small. Also, we assume that rotation perturbations in optimization are always small because the optimization will always start with a good initial guess from either initialization or the previous optimization. To reduce computational cost, we apply the approximations (A.28) and (A.30) to all capitalized quaternion exponential and logarithmic maps in preintegration-related equations.

Midpoint Integration

Although (3.34)-(3.36) and (3.59) are discretized using Euler integration, the implementation uses midpoint integration (see Appendix B.2) to improve accuracy without significantly increasing computational cost.

3.3 IMU Factor

A set of consecutive IMU observations over a time interval between a pair of images establishes a motion constraint between the IMU poses at those times. The IMU factor encodes this information as a cost term in a visual-inertial optimization problem's factor graph via IMU preintegration techniques.

[4]These quantities can be obtained using a calibration that aligns motion capture and IMU-derived accelerations (see Appendix D.1)

3.3.1 Residual

The IMU preintegration residual represents a soft constraint between the motion state and IMU bias estimates of two successive keyframes i and $j = i + 1$. Starting from this section, we replace IMU observation indices 0 and n from the preceding derivation with keyframe indices i and j, respectively. Accordingly, $(\cdot)'_{i,j} = (\cdot)'_n$ for $(\cdot) = \{\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}\}$ and $\Delta T = t_j - t_i = t_n - t_0$. The position, orientation, and velocity components of the IMU preintegration residual are formed by taking the manifold difference of observation-independent relative motion (left hand sides of (3.31)-(3.33)) and observation-dependent relative motion (3.77)-(3.79).

$$\mathbf{r}_\alpha = \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \left(\mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i \Delta T - \frac{1}{2} \mathbf{g}_w \Delta T^2 \right) - \boldsymbol{\alpha}'_{i,j} \left(\mathbf{b}_{a,i}, \mathbf{b}_{\omega,i} \right) \tag{3.82}$$

$$\mathbf{r}_\beta = \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}_w \Delta T \right) - \boldsymbol{\beta}'_{i,j} \left(\mathbf{b}_{a,i}, \mathbf{b}_{\omega,i} \right) \tag{3.83}$$

$$\mathbf{r}_\theta = \mathrm{Log} \left[\boldsymbol{\gamma}'_{i,j} \left(\mathbf{b}_{\omega,i} \right)^{-1} \otimes \mathbf{q}_i^{-1} \otimes \mathbf{q}_j \right] \tag{3.84}$$

The bias components of the IMU preintegration residual encode the assumption that IMU biases remain constant over the preintegration time interval[5].

$$\mathbf{r}_{ba} = \mathbf{b}_{a,j} - \mathbf{b}_{a,i} \tag{3.85}$$

$$\mathbf{r}_{b\omega} = \mathbf{b}_{\omega,j} - \mathbf{b}_{a,i} \tag{3.86}$$

The IMU preintegration residual is formed by stacking (3.82)-(3.86).

$$\mathbf{r}_{\text{imu}}(\mathbf{x}) = \begin{bmatrix} \mathbf{r}_\alpha \left(\mathbf{p}_i, \mathbf{v}_i, \mathbf{q}_i, \mathbf{b}_{a,i}, \mathbf{b}_{\omega,i}, \mathbf{p}_j \right) \\ \mathbf{r}_\beta \left(\mathbf{v}_i, \mathbf{q}_i, \mathbf{b}_{a,i}, \mathbf{b}_{\omega,i}, \mathbf{v}_j \right) \\ \mathbf{r}_\theta \left(\mathbf{q}_i, \mathbf{b}_{\omega,j}, \mathbf{q}_j \right) \\ \mathbf{r}_{ba} \left(\mathbf{b}_{a,i}, \mathbf{b}_{a,j} \right) \\ \mathbf{r}_{b\omega} \left(\mathbf{b}_{\omega,i}, \mathbf{b}_{\omega,j} \right) \end{bmatrix} \in \mathbb{R}^{15} \tag{3.87}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_i^{\mathrm{T}} & \mathbf{v}_i^{\mathrm{T}} & \mathbf{q}_i^{\mathrm{T}} & \mathbf{b}_{a,i}^{\mathrm{T}} & \mathbf{b}_{\omega,i}^{\mathrm{T}} & \mathbf{p}_j^{\mathrm{T}} & \mathbf{v}_i^{\mathrm{T}} & \mathbf{q}_j^{\mathrm{T}} & \mathbf{b}_{a,j}^{\mathrm{T}} & \mathbf{b}_{\omega,j}^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{32} \tag{3.88}$$

3.3.2 Covariance

The IMU factor residual's covariance is the covariance of the motion delta error dynamics (3.64) at the end of preintegration time interval.

3.3.3 Jacobians

This section derives the nonzero Jacobians of the IMU factor's residuals. All Jacobians presented in this section are 3×3 matrices.

Vector Space Jacobians

The Jacobians of the residuals $\mathbf{r}_\alpha$, $\mathbf{r}_\beta$, $\mathbf{r}_{ba}$, and $\mathbf{r}_{b\omega}$ with respect to vector space parameter blocks $\mathbf{p}_i$, $\mathbf{v}_i$, $\mathbf{b}_{a,i}$, $\mathbf{b}_{\omega,i}$, $\mathbf{p}_j$, $\mathbf{v}_j$, $\mathbf{b}_{a,j}$, $\mathbf{b}_{\omega,j}$ are obtained by isolating the term in the relevant residual that involves each parameter block and taking the derivative. The Jacobians of $\mathbf{r}_\alpha$ with respect to $\mathbf{p}_i$, $\mathbf{v}_i$, $\mathbf{b}_{a,i}$, $\mathbf{b}_{\omega,i}$, and $\mathbf{p}_j$ are

$$\frac{\partial \mathbf{r}_\alpha}{\partial \mathbf{p}_i} = -\mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \tag{3.89}$$

$$\frac{\partial \mathbf{r}_\alpha}{\partial \mathbf{v}_i} = -\Delta T \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \tag{3.90}$$

[5] Given typical time intervals of < 0.1 s between successive keyframes, this is a reasonable assumption

$$\frac{\partial \mathbf{r}_\alpha}{\partial \mathbf{b}_{a,i}} = -\mathbf{J}_{\alpha b_a} \tag{3.91}$$

$$\frac{\partial \mathbf{r}_\alpha}{\partial \mathbf{b}_{\omega,i}} = -\mathbf{J}_{\alpha b_\omega} \tag{3.92}$$

$$\frac{\partial \mathbf{r}_\alpha}{\partial \mathbf{p}_j} = \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \tag{3.93}$$

The Jacobians of $\mathbf{r}_\beta$ with respect to $\mathbf{v}_i$, $\mathbf{b}_{a,i}$, $\mathbf{b}_{\omega,i}$, and $\mathbf{v}_j$ are

$$\frac{\partial \mathbf{r}_\beta}{\partial \mathbf{v}_i} = -\mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \tag{3.94}$$

$$\frac{\partial \mathbf{r}_\beta}{\partial \mathbf{b}_{a,i}} = -\mathbf{J}_{\beta b_a} \tag{3.95}$$

$$\frac{\partial \mathbf{r}_\beta}{\partial \mathbf{b}_{\omega,i}} = -\mathbf{J}_{\beta b_\omega} \tag{3.96}$$

$$\frac{\partial \mathbf{r}_\beta}{\partial \mathbf{v}_j} = \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \tag{3.97}$$

The Jacobians of $\mathbf{r}_{ba}$ and $\mathbf{r}_{b\omega}$ with respect to the IMU bias states are

$$\frac{\partial \mathbf{r}_{ba}}{\partial \mathbf{b}_{a,i}} = -\mathbf{I} \tag{3.98}$$

$$\frac{\partial \mathbf{r}_{ba}}{\partial \mathbf{b}_{a,j}} = \mathbf{I} \tag{3.99}$$

$$\frac{\partial \mathbf{r}_{b\omega}}{\partial \mathbf{b}_{\omega,i}} = -\mathbf{I} \tag{3.100}$$

$$\frac{\partial \mathbf{r}_{b\omega}}{\partial \mathbf{b}_{\omega,j}} = \mathbf{I} \tag{3.101}$$

Manifold Jacobians

Instead of taking Jacobians with respect to the quaternions $\mathbf{q}_i$ and $\mathbf{q}_j$, we take Jacobians with respect to minimal parameterizations of their perturbations, $\delta\boldsymbol{\theta}_i$ and $\delta\boldsymbol{\theta}_j$.

We use the perturbation method to obtain the Jacobian of $\mathbf{r}_\alpha$ with respect to $\delta\boldsymbol{\theta}_i$. First rewrite (3.82) with the temporary variable $\Delta\mathbf{p}_{w,ij}$.

$$\mathbf{r}_\alpha = \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \Delta\mathbf{p}_{w,ij} - \boldsymbol{\alpha}'_{i,j} \tag{3.102}$$

$$\Delta\mathbf{p}_{w,ij} = \mathbf{p}_j - \mathbf{p}_i - \mathbf{v}_i \Delta T - \frac{1}{2}\mathbf{g}_w \Delta T^2 \tag{3.103}$$

Apply an additive perturbation to the left hand side of (3.102) and a rotational perturbation to the base rotation $\mathbf{R}(\mathbf{q}_i)$ on the right hand side of (3.102).

$$\mathbf{r}_\alpha + \delta\mathbf{r}_\alpha = \left[\mathbf{R}(\mathbf{q}_i)\,\mathrm{Exp}(\delta\boldsymbol{\theta}_i)\right]^{\mathrm{T}} \Delta\mathbf{p}_{w,ij} - \boldsymbol{\alpha}'_{i,j} \tag{3.104}$$

Assuming the rotational perturbation $\delta\boldsymbol{\theta}_i$ is small, use the approximation (A.14).

$$\mathbf{r}_\alpha + \delta\mathbf{r}_\alpha \approx \left[\mathbf{R}(\mathbf{q}_i)\left(\mathbf{I} + [\delta\boldsymbol{\theta}_i]_\times\right)\right]^{\mathrm{T}} \Delta\mathbf{p}_{w,ij} - \boldsymbol{\alpha}'_{i,j} \tag{3.105}$$

$$\mathbf{r}_\alpha + \delta\mathbf{r}_\alpha \approx \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \Delta\mathbf{p}_{w,ij} - \boldsymbol{\alpha}'_{i,j} + [\delta\boldsymbol{\theta}_i]_\times^{\mathrm{T}}\, \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \Delta\mathbf{p}_{w,ij} \tag{3.106}$$

$$\delta\mathbf{r}_\alpha \overset{(3.102)}{\approx} [\delta\boldsymbol{\theta}_i]_\times^{\mathrm{T}}\, \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \Delta\mathbf{p}_{w,ij} \tag{3.107}$$

$$\delta\mathbf{r}_\alpha \overset{(A.4)}{\approx} -[\delta\boldsymbol{\theta}_i]_\times\, \mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \Delta\mathbf{p}_{w,ij} \tag{3.108}$$

$$\delta\mathbf{r}_\alpha \overset{(A.3)}{\approx} \left[\mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \Delta\mathbf{p}_{w,ij}\right]_\times \delta\boldsymbol{\theta}_i \tag{3.109}$$

We obtain the approximate Jacobian of $\mathbf{r}_\alpha$ with respect to $\delta\boldsymbol{\theta}_i$ by substituting (3.103) into (3.109).

$$\frac{\partial \mathbf{r}_\alpha}{\partial \delta\boldsymbol{\theta}_i} \approx \left[\mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \left(\mathbf{p}_j - \mathbf{p}_i - \Delta T \mathbf{v}_i - \frac{1}{2}\mathbf{g}\Delta T^2\right)\right]_\times \tag{3.110}$$

Using a similar method, we obtain the approximate Jacobian of $\mathbf{r}_\beta$ with respect to $\delta\boldsymbol{\theta}_i$ as

$$\frac{\partial \mathbf{r}_\beta}{\partial \delta\boldsymbol{\theta}_i} \approx \left[\mathbf{R}(\mathbf{q}_i)^{\mathrm{T}} \left(\mathbf{v}_j - \mathbf{v}_i - \mathbf{g}\Delta T\right)\right]_\times \tag{3.111}$$

We use the perturbation method to derive the Jacobians of $\mathbf{r}_\theta$ with respect to $\delta\boldsymbol{\theta}_i$, $\mathbf{b}_{\omega,i}$, and $\delta\boldsymbol{\theta}_j$. First, rewrite (3.79) in terms of rotation matrices instead of quaternions

$$\mathbf{R}_{\gamma'_{ij}}(\mathbf{b}_{\omega,i}) \approx \mathbf{R}_{\bar{\gamma}_{ij}}(\bar{\mathbf{b}}_{\omega,i}) \operatorname{Exp}\left[\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i}\right)\right] \tag{3.112}$$

$$\mathbf{R}_{\gamma'_{ij}} \equiv \mathbf{R}_{\gamma'_{ij}}(\mathbf{b}_{\omega,i}) \equiv \mathbf{R}\left(\boldsymbol{\gamma}'_{ij}(\mathbf{b}_{\omega,i})\right) \tag{3.113}$$

$$\mathbf{R}_{\bar{\gamma}_{ij}} \equiv \mathbf{R}_{\bar{\gamma}_{ij}}(\bar{\mathbf{b}}_{\omega,i}) \equiv \mathbf{R}\left(\bar{\boldsymbol{\gamma}}_{ij}(\bar{\mathbf{b}}_{\omega,i})\right) \tag{3.114}$$

Next, rewrite (3.84) in terms of rotation matrices instead of quaternions

$$\mathbf{r}_\theta = \operatorname{Log}\left[\mathbf{R}_{\gamma'_{ij}}(\mathbf{b}_{\omega,i})^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_j\right] \tag{3.115}$$

$$\mathbf{R}_i \equiv \mathbf{R}(\mathbf{q}_i) \tag{3.116}$$

$$\mathbf{R}_j \equiv \mathbf{R}(\mathbf{q}_j) \tag{3.117}$$

Substitute (3.112) into (3.115)

$$\mathbf{r}_\theta \approx \operatorname{Log}\left\{\left[\mathbf{R}_{\bar{\gamma}_{ij}} \operatorname{Exp}\left[\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i}\right)\right]\right]^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_j\right\} \tag{3.118}$$

Apply an additive perturbation to the left hand side of (3.118), rotational perturbations $\mathbf{R}_i$ and $\mathbf{R}_j$, and an additive perturbation to the gyro bias $\mathbf{b}_{\omega,i}$.

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \approx \operatorname{Log}\left\{\left[\mathbf{R}_{\bar{\gamma}_{ij}} \operatorname{Exp}\left[\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i} + \delta\mathbf{b}_{w,i}\right)\right]\right]^{\mathrm{T}} \left[\mathbf{R}_i \operatorname{Exp}(\delta\boldsymbol{\theta}_i)\right]^{\mathrm{T}} \mathbf{R}_j \operatorname{Exp}(\delta\boldsymbol{\theta}_j)\right\} \tag{3.119}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.13)}{\approx} \operatorname{Log}\left\{\operatorname{Exp}\left[-\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i} + \delta\mathbf{b}_{w,i}\right)\right] \mathbf{R}_{\bar{\gamma}_{ij}}^{\mathrm{T}} \operatorname{Exp}(-\delta\boldsymbol{\theta}_i) \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_j \operatorname{Exp}(\delta\boldsymbol{\theta}_j)\right\} \tag{3.120}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.46)}{\approx} \operatorname{Log}\left\{\operatorname{Exp}\left[-\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i} + \delta\mathbf{b}_{w,i}\right)\right] \mathbf{R}_{\bar{\gamma}_{ij}}^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_j \operatorname{Exp}(-\mathbf{R}_j^{\mathrm{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i) \operatorname{Exp}(\delta\boldsymbol{\theta}_j)\right\} \tag{3.121}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.46)}{\approx} \operatorname{Log}\left\{\mathbf{R}_{\bar{\gamma}_{ij}}^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_j \operatorname{Exp}\left[-\mathbf{R}_j^{\mathrm{T}} \mathbf{R}_i \mathbf{R}_{\bar{\gamma}_{ij}} \mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i} + \delta\mathbf{b}_{w,i}\right)\right] \operatorname{Exp}(-\mathbf{R}_j^{\mathrm{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i) \operatorname{Exp}(\delta\boldsymbol{\theta}_j)\right\} \tag{3.122}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.49)}{\approx} \operatorname{Log}\left\{\mathbf{R}_{\bar{\gamma}_{ij}}^{\mathrm{T}} \mathbf{R}_i^{\mathrm{T}} \mathbf{R}_j \operatorname{Exp}\left[-\mathbf{R}_j^{\mathrm{T}} \mathbf{R}_i \mathbf{R}_{\bar{\gamma}_{ij}} \mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i} + \delta\mathbf{b}_{w,i}\right) - \mathbf{R}_j^{\mathrm{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_j\right]\right\} \tag{3.123}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(3.115)}{\approx} \operatorname{Log}\left\{\operatorname{Exp}(\mathbf{r}_\theta) \operatorname{Exp}\left[-\mathbf{R}_j^{\mathrm{T}} \mathbf{R}_i \mathbf{R}_{\bar{\gamma}_{ij}} \mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i} + \delta\mathbf{b}_{w,i}\right) - \mathbf{R}_j^{\mathrm{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_j\right]\right\} \tag{3.124}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.48)}{\approx} \mathbf{r}_\theta + \mathbf{J}_r^{-1}(\mathbf{r}_\theta)\left[-\mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\mathbf{R}_{\bar{\gamma}_{ij}}\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{w,i} - \bar{\mathbf{b}}_{w,i} + \delta\mathbf{b}_{w,i}\right) - \mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\delta\boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_j\right] \tag{3.125}$$

$$\delta\mathbf{r}_\theta \approx \mathbf{J}_r^{-1}(\mathbf{r}_\theta)\left[-\mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\mathbf{R}_{\bar{\gamma}_{ij}}\mathbf{J}_{\theta b_\omega}\delta\mathbf{b}_{w,i} - \mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\delta\boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_j\right] + const \tag{3.126}$$

Apply the small angle approximation $\mathbf{J}_r^{-1}(\boldsymbol{\phi}) \approx \mathbf{I} + \frac{1}{2}\left[\boldsymbol{\phi}\right]_\times$ for the inverse of the right Jacobian of $SO(3)$.

$$\delta\mathbf{r}_\theta \approx \left(\mathbf{I} + \frac{1}{2}\left[\mathbf{r}_\theta\right]_\times\right)\left[-\mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\mathbf{R}_{\bar{\gamma}_{ij}}\mathbf{J}_{\theta b_\omega}\delta\mathbf{b}_{w,i} - \mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\delta\boldsymbol{\theta}_i + \delta\boldsymbol{\theta}_j\right] + const \tag{3.127}$$

The approximate Jacobians of the orientation residual are

$$\frac{\partial\mathbf{r}_\theta}{\partial\delta\boldsymbol{\theta}_i} = -\left(\mathbf{I} + \frac{1}{2}\left[\mathbf{r}_\theta\right]_\times\right)\mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i \tag{3.128}$$

$$\frac{\partial\mathbf{r}_\theta}{\partial\mathbf{b}_{\omega,i}} = -\left(\mathbf{I} + \frac{1}{2}\left[\mathbf{r}_\theta\right]_\times\right)\mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\mathbf{R}_{\bar{\gamma}_{ij}}\mathbf{J}_{\theta b_\omega} \tag{3.129}$$

$$\frac{\partial\mathbf{r}_\beta}{\partial\delta\boldsymbol{\theta}_j} = \mathbf{I} + \frac{1}{2}\left[\mathbf{r}_\theta\right]_\times \tag{3.130}$$

3.4 Reprojection Factor

A feature observed from a pair of images establishes a geometric constraint between the camera poses from which the images were taken. The reprojection factor encodes this information as a cost term in a visual-inertial optimization problem's factor graph.

3.4.1 Reprojection Geometry

Let the feature with index l be observed in keyframe images indexed i and j. Let $\{b\}$ denote the body frame, $\{c\}$ denote the camera frame, and $\{w\}$ denote the world frame. Let $\mathbf{R}_{bc} \in SO(3)$ be the orientation of the camera frame with respect to the body frame. Let $\mathbf{p}_{bc} \in \mathbb{R}^3$ be the position of the camera's optical center with respect to the body frame[6]. Let $\mathbf{z}(\cdot)$ be the function that projects a camera frame position to the normalized image plane, which is the plane perpendicular to the camera's optical axis at a unit distance in front of the camera frame origin.

$$\mathbf{z}\left(\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}\right) = \begin{bmatrix} \frac{a_x}{a_z} \\ \frac{a_y}{a_z} \end{bmatrix} \tag{3.131}$$

Let $\mathbf{h}_{l,i} = \begin{bmatrix} h_{lix} & h_{liy} & 1 \end{bmatrix}^{\mathrm{T}}$ be the homogeneous point representation of feature l's observation in keyframe i and $\mathbf{h}_{l,j} = \begin{bmatrix} h_{ljx} & h_{ljy} & 1 \end{bmatrix}^{\mathrm{T}}$ be the homogeneous point representation of feature l's observation in keyframe j. Let $\rho_l \in \mathbb{R}^+$ be feature l's inverse depth in keyframe i. Let $\mathbf{p}_i$ and $\mathbf{p}_j$ be the positions of the body frame origin in the world frame at the times corresponding to keyframes i and j.

$$\mathbf{p}_i = \mathbf{p}_{wb_i} \tag{3.132}$$

$$\mathbf{p}_j = \mathbf{p}_{wb_j} \tag{3.133}$$

Let $\mathbf{q}_i$ and $\mathbf{q}_j$ be quaternions representing the orientation of the body frame with respect to the world frame at the times corresponding to keyframes i and j.

$$\mathbf{R}(\mathbf{q}_i) = \mathbf{R}_{wb_i} \tag{3.134}$$

[6] See Appendices D.2 and E.1, respectively, for offline and online methods to estimate $\mathbf{p}_{bc}$ and $\mathbf{R}_{bc}$.

$$\mathbf{R}(\mathbf{q}_j) = \mathbf{R}_{wb_j} \tag{3.135}$$

The position of feature l in the camera frame associated with keyframe i is

$$\mathbf{p}_{l,c_i} = \frac{1}{\rho_l}\mathbf{h}_{l,i} \tag{3.136}$$

The position of feature l in the body frame associated with keyframe i is

$$\mathbf{p}_{l,b_i} = \mathbf{R}_{bc}\mathbf{p}_{l,c_i} + \mathbf{p}_{bc} \tag{3.137}$$

The position of feature l in the world frame is

$$\mathbf{p}_{l,w} = \mathbf{R}(\mathbf{q}_i)\mathbf{p}_{l,b_i} + \mathbf{p}_i \tag{3.138}$$

The position of feature l in the body frame associated with keyframe j is

$$\mathbf{p}_{l,b_j} = \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) \tag{3.139}$$

The position of feature l in the camera frame associated with keyframe j is

$$\mathbf{p}_{l,c_j} = \mathbf{R}_{bc}^{\mathrm{T}}\left(\mathbf{p}_{l,b_j} - \mathbf{p}_{bc}\right) \tag{3.140}$$

Combine (3.136)-(3.140) to obtain an expression for feature l's estimated position in the camera frame associated with keyframe j in terms of feature l's observation in the camera frame associated with keyframe i.

$$\mathbf{p}_{l,c_j} = \mathbf{R}_{bc}^{\mathrm{T}}\left(\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{R}(\mathbf{q}_i)\left(\mathbf{R}_{bc}\frac{1}{\rho_l}\mathbf{h}_{l,i} + \mathbf{p}_{bc}\right) + \mathbf{p}_i - \mathbf{p}_j\right) - \mathbf{p}_{bc}\right) \tag{3.141}$$

3.4.2 Residual

Under idealized conditions, the reprojection of feature l's estimated world frame location onto the camera's image plane at keyframe j exactly matches the image observation of feature l at keyframe j. Because camera pose and feature depth are estimated quantities, we use the reprojection residual to represent the error between feature observations in keyframe j and the reprojections associated with the estimated camera poses and feature depth in keyframe i.

The reprojection residual is a 2D quantity that represents the 2D vector difference on the normalized image plane between feature l's observed and reprojected locations in keyframe j, where the reprojection is computed using the observation of feature l in keyframe i.

$$\mathbf{r}_{\text{proj}}\left(\mathbf{p}_i, \mathbf{q}_i, \mathbf{p}_j, \mathbf{q}_j, \rho_l\right) = \mathbf{z}\left(\mathbf{p}_{l,c_j}\right) - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\mathbf{h}_{l,j} \tag{3.142}$$

In the standard reprojection residual, the extrinsic transform quantities $\mathbf{R}_{bc}$ and $\mathbf{p}_{bc}$ are constant. Note that we have chosen to write the residual as (prediction - observation) rather than the common convention of (observation - prediction) in order to avoid an extra negative sign in front of all the Jacobians.

3.4.3 Covariance

The covariance associated with the reprojection residual error is related to the accuracy of feature tracking between image keyframes. Because features are matched across image frames by minimizing grayscale intensity differences between a small surrounding patch of pixels in each image, features in regions with small intensity variations are more likely to be matched incorrectly and thus have higher uncertainty. Mathematically, feature location uncertainty is described by a covariance matrix computed from the inverse of the grayscale intensity Hessian over a patch of pixels surrounding the feature location [74]. However, in many cases assuming isotropic uniform covariance is sufficient for good optimization performance.

3.4.4 Jacobians

The standard reprojection residual written as a function of any of the variables $\mathbf{x} = \{\mathbf{p}_i, \mathbf{q}_i, \mathbf{p}_j, \mathbf{q}_j, \rho_l\}$ has the form $\mathbf{z}\left(\mathbf{a}(\mathbf{x})\right) + constant$, where $\mathbf{z}(\cdot)$ is defined in (3.131) and $\mathbf{a}(\cdot) \triangleq \mathbf{p}_{l,c_j}(\cdot)$ represents the expression in (3.141). By chain rule, the Jacobian for any parameter block $\mathbf{x}$ is given by

$$\frac{\partial \mathbf{z}}{\partial \mathbf{x}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{x}} \tag{3.143}$$

The first term on the left hand side is the Jacobian of the image plane projection function

$$\frac{\partial \mathbf{z}}{\partial \mathbf{a}} = \begin{bmatrix} \frac{1}{a_z} & 0 & -\frac{a_x}{a_z^2} \\ 0 & \frac{1}{a_z} & -\frac{a_y}{a_z^2} \end{bmatrix} \tag{3.144}$$

and is obtained by straightforward differentiation of (3.131).

Vector Space Jacobians

The Jacobians with respect to vector space parameter blocks $\mathbf{p}_i$, $\mathbf{p}_j$, and ρ_l can be obtained by expanding out (3.141), isolating the term that involves each parameter block, and then taking the derivative. Using this method, the Jacobians of $\mathbf{a}$ with respect to $\mathbf{p}_i$ and $\mathbf{p}_j$ are

$$\frac{\partial \mathbf{a}}{\partial \mathbf{p}_i} = \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \tag{3.145}$$

$$\frac{\partial \mathbf{a}}{\partial \mathbf{p}_j} = -\mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \tag{3.146}$$

The Jacobian of $\mathbf{a}$ with respect to ρ_l is

$$\frac{\partial \mathbf{a}}{\partial \rho_l} = \frac{\partial \mathbf{a}}{\partial (\frac{1}{\rho_l})} \cdot \frac{\partial}{\partial \rho_l} \left(\frac{1}{\rho_l} \right) \tag{3.147}$$

$$= \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) \mathbf{R}_{bc} \mathbf{h}_{l,i} \cdot \frac{-1}{\rho_l^2} \tag{3.148}$$

Manifold Jacobians

Instead of taking Jacobians with respect to the quaternions $\mathbf{q}_i$ and $\mathbf{q}_j$, we take Jacobians with respect to minimal parameterizations of their perturbations, $\delta\boldsymbol{\theta}_i$ and $\delta\boldsymbol{\theta}_j$.

We use the perturbation method to obtain the Jacobian of $\mathbf{a}$ with respect to $\delta\boldsymbol{\theta}_i$. First rewrite (3.140) to separate the term that depends on $\mathbf{q}_i$ from all other terms, denoted as *const*.

$$\mathbf{a} = \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) \mathbf{p}_{l,b_i} + const \tag{3.149}$$

Apply an additive perturbation to the left hand side of (3.149) and a rotational perturbation to the base rotation $\mathbf{R}(\mathbf{q}_i)$ on the right hand side of (3.149)

$$\mathbf{a} + \delta\mathbf{a} = \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) \operatorname{Exp}(\delta\boldsymbol{\theta}_i) \mathbf{p}_{l,b_i} + const \tag{3.150}$$

Assuming the rotational perturbation $\delta\boldsymbol{\theta}_i$ is small, use the approximation (A.14)

$$\mathbf{a} + \delta\mathbf{a} \approx \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) \left(\mathbf{I} + [\delta\boldsymbol{\theta}_i]_\times \right) \mathbf{p}_{l,b_i} + const \tag{3.151}$$

$$\mathbf{a} + \delta\mathbf{a} \approx \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) \mathbf{p}_{l,b_i} + const + \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) [\delta\boldsymbol{\theta}_i]_\times \mathbf{p}_{l,b_i} \tag{3.152}$$

$$\delta\mathbf{a} \overset{(3.149)}{\approx} \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) [\delta\boldsymbol{\theta}_i]_\times \mathbf{p}_{l,b_i} \tag{3.153}$$

$$\delta\mathbf{a} \overset{(A.3)}{\approx} -\mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\mathbf{R}(\mathbf{q}_i)\left[\mathbf{p}_{l,b_i}\right]_{\times}\delta\boldsymbol{\theta}_i \tag{3.154}$$

We obtain the approximate Jacobian of $\mathbf{a}$ with respect to $\delta\boldsymbol{\theta}_i$ by substituting (3.137) into (3.154).

$$\frac{\partial\mathbf{a}}{\partial\delta\boldsymbol{\theta}_i} \approx -\mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\mathbf{R}(\mathbf{q}_i)\left[\mathbf{R}_{bc}\frac{1}{\rho_l}\mathbf{h}_{l,i} + \mathbf{p}_{bc}\right]_{\times} \tag{3.155}$$

We use the perturbation method to obtain the Jacobian of $\mathbf{a}$ with respect to $\delta\boldsymbol{\theta}_j$. First rewrite (3.140) to separate the term that depends on $\mathbf{q}_j$ from all other terms, denoted as *const*.

$$\mathbf{a} = \mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) + const \tag{3.156}$$

Apply an additive perturbation to the left hand side of (3.156) and a rotational perturbation to the base rotation $\mathbf{R}(\mathbf{q}_j)$ on the right hand side of (3.156).

$$\mathbf{a} + \delta\mathbf{a} = \mathbf{R}_{bc}^{\mathrm{T}}\left[\mathbf{R}(\mathbf{q}_j)\,\mathrm{Exp}(\delta\boldsymbol{\theta}_j)\right]^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) + const \tag{3.157}$$

Assuming the rotational perturbation $\delta\boldsymbol{\theta}_j$ is small, use the approximation (A.14)

$$\mathbf{a} + \delta\mathbf{a} \approx \mathbf{R}_{bc}^{\mathrm{T}}\left[\mathbf{R}(\mathbf{q}_j)\left(\mathbf{I} + \left[\delta\boldsymbol{\theta}_j\right]_{\times}\right)\right]^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) + const \tag{3.158}$$

$$\mathbf{a} + \delta\mathbf{a} \approx \mathbf{R}_{bc}^{\mathrm{T}}\left(\mathbf{I} + \left[\delta\boldsymbol{\theta}_j\right]_{\times}\right)^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) + const \tag{3.159}$$

$$\mathbf{a} + \delta\mathbf{a} \approx \mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) + const + \mathbf{R}_{bc}^{\mathrm{T}}\left[\delta\boldsymbol{\theta}_j\right]_{\times}^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) \tag{3.160}$$

$$\delta\mathbf{a} \overset{(3.156)}{\approx} \mathbf{R}_{bc}^{\mathrm{T}}\left[\delta\boldsymbol{\theta}_j\right]_{\times}^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) \tag{3.161}$$

$$\delta\mathbf{a} \overset{(A.4)}{\approx} -\mathbf{R}_{bc}^{\mathrm{T}}\left[\delta\boldsymbol{\theta}_j\right]_{\times}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right) \tag{3.162}$$

$$\delta\mathbf{a} \overset{(A.3)}{\approx} \mathbf{R}_{bc}^{\mathrm{T}}\left[\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{p}_{l,w} - \mathbf{p}_j\right)\right]_{\times}\delta\boldsymbol{\theta}_j \tag{3.163}$$

We obtain the approximate Jacobian of $\mathbf{a}$ with respect to $\delta\boldsymbol{\theta}_j$ by substituting (3.138) into (3.163).

$$\frac{\partial\mathbf{a}}{\partial\delta\boldsymbol{\theta}_j} \approx \mathbf{R}_{bc}^{\mathrm{T}}\left[\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{R}(\mathbf{q}_i)\left(\mathbf{R}_{bc}\frac{1}{\rho_l}\mathbf{h}_{l,i} + \mathbf{p}_{bc}\right) + \mathbf{p}_i - \mathbf{p}_j\right)\right]_{\times} \tag{3.164}$$

3.5 Single Camera Optimization

3.5.1 Problem Definition

The sliding window visual-inertial indirect bundle adjustment problem seeks to estimate IMU motion, biases, and feature locations over a finite sliding window of image keyframes by minimizing a nonlinear objective function that encodes vision-derived and IMU-derived motion constraints.

Consider a sliding window of n image keyframes corresponding to timestamps $t_1 < \ldots < t_n$. Each keyframe is associated with a motion state that consists of world frame position, orientation, and velocity and IMU accelerometer and gyroscope biases.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_{wb,k} \\ \mathbf{q}_{wb,k} \\ \mathbf{v}_{wb,k} \\ \mathbf{b}_{a,k} \\ \mathbf{b}_{\omega,k} \end{bmatrix} \in \mathbb{R}^{16} \tag{3.165}$$

In subsequent equations we drop the frame subscripts wb for conciseness.

We assume that all images are obtained from a single[7] camera and time synchronized with IMU observations after applying time offset compensation. Suppose there are m feature trails that have been observed in at least two consecutive keyframes, starting on or before a threshold number of keyframes before the sliding window's last keyframe. The location of feature i in the world frame is parametrized by its inverse depth (ρ_i) in the camera frame of the first keyframe in which it was observed.

We concatenate position, orientation, velocity, IMU bias, and inverse depths to form the full state vector $\mathbf{x}$.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} & \dots & \mathbf{x}_n^{\mathrm{T}} & \rho_1 & \dots & \rho_m \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{16n+m} \tag{3.166}$$

The unconstrained optimization problem has the form:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[f_{\text{prior}}(\mathbf{x}) + f_{\text{imu}}(\mathbf{x}) + f_{\text{cam}}(\mathbf{x}) \right] \tag{3.167}$$

Because each of the objective function's terms consists of sums of squared residual norms, (3.167) is a nonlinear least squares problem that can be solved via the methods in Appendix C. After associating individual squared residual norm terms with the negative log likelihoods of factors within a factor graph, it becomes clear that the minimization problem performs *maximum a posterior* estimation.

The prior cost encodes constraints on current sliding window states associated with previous observations and states that have been discarded. Section 3.6 describes the prior residual (also known as the marginalization residual) in more detail.

$$f_{\text{prior}}(\mathbf{x}) = \frac{1}{2} \left\| \mathbf{r}_{\text{prior}}(\mathbf{x}) \right\|^2 \tag{3.168}$$

The IMU cost encodes constraints on sliding window motion and IMU bias states using preintegrated IMU motion deltas. Section 3.3.1 describes the IMU preintegration residual in more detail.

$$f_{\text{imu}}(\mathbf{x}) = \frac{1}{2} \sum_{k=1}^{n-1} \left\| \mathbf{r}_{\text{imu}}(\mathbf{x}_k, \mathbf{x}_{k+1}) \right\|^2_{\mathbf{P}_{k,k+1}} \tag{3.169}$$

The camera cost encodes geometric constraints on sliding window poses and inverse depths derived from feature matches across sliding window keyframes.

$$f_{\text{cam}}(\mathbf{x}) = \frac{1}{2} \sum_{l=1}^{m} \sum_{i=2}^{|\mathcal{K}_l|} \lambda \left(\left\| \mathbf{r}_{\text{proj}}\left(\mathbf{p}_{k(l,1)}, \mathbf{q}_{k(l,1)}, \mathbf{p}_{k(l,i)}, \mathbf{q}_{k(l,i)}, \rho_l\right) \right\|^2_{\boldsymbol{\Sigma}} \right) \tag{3.170}$$

$\mathcal{K}_l$ is the set of keyframe indices where feature l is observed, $k(l,i)$ is a function that returns the keyframe index of the i^{th} observation of feature l, $\lambda(\cdot)$ is a loss function, and $\boldsymbol{\Sigma} \in \mathbb{R}^{2\times 2}$ is the feature observation noise covariance matrix. Section 3.4 describes the reprojection residual in more detail.

3.5.2 Incorporation of New Feature Trails into Optimization

Newly detected features must be successfully tracked for a threshold number of keyframes before their inverse depths are included as states in the optimization vector. While this threshold may be set as low as two keyframes, which is the minimum number required to obtain an initial guess for the inverse depth via triangulation, in practice a higher threshold is used to improve triangulation quality. This helps avoid situations where introducing a large number of new inverse depth states with poor initial guesses causes the visual-inertial optimization to diverge.

[7] See Chapter 5 for the multi-camera problem formulation.

When running on a system with limited computation, it is desirable to keep the total optimization vector size constant. This requires the number of keyframes in the sliding window, n, as well as the number of feature trails, m, to remain constant over successive optimizations. A method to maintain a constant sliding window size is described in Sect. 3.6.3, while the number of features may be kept constant by initially adding new features up to a fixed feature budget and subsequently adding new features only when older features are discarded.

3.5.3 Implementation Details

Rotational Perturbation Convention

The quaternion $\boxplus$ operator is defined using right perturbations (C.24).

Loss Function

The Cauchy loss function is used in (3.170).

$$\lambda(s) = \log(s+1) \tag{3.171}$$

Jacobian Scaling

Cost function terms defined using the Mahalanobis norm can be rewritten as L2 norms by replacing the original residual and Jacobian with

$$\mathbf{r}_{\boldsymbol{\Sigma}}(\mathbf{x}) = \mathbf{L}^{\mathrm{T}}\mathbf{r}(\mathbf{x}) \tag{3.172}$$

$$\mathbf{J}_{\boldsymbol{\Sigma}}(\mathbf{x}) = \mathbf{L}^{\mathrm{T}}\mathbf{J}(\mathbf{x}) \tag{3.173}$$

where $\boldsymbol{\Sigma}^{-1} = \mathbf{L}\mathbf{L}^{\mathrm{T}}$.

Ignore IMU Factors Associated with Long Preintegration Time Intervals

The IMU cost function (3.169) is modified to exclude IMU factors associated with preintegration time intervals that are greater than a user-defined threshold in order to prevent their inaccurate motion constraints from degrading the optimization problem. Long preintegration time intervals typically occur during motionless periods when the marginalization strategy (Sect. 3.6.3) repeatedly discards the second newest keyframe and causes the time gap between the latest two sliding window keyframes to grow. As a result, the IMU factor between the latest two sliding window keyframes is based on the preintegration of a large number of IMU observations. Such IMU factors often represent severely inaccurate motion constraints because errors from imperfectly estimated IMU bias estimates are propagated over non-trivial time intervals. In practice, this problem can be avoided by removing long time interval IMU factors from the visual-inertial optimization problem and relying solely on vision-based geometric constraints to relate keyframes with large temporal separation.

IMU-Rate Odometry Upsampling

Sliding window optimization occurs at the keyframe rate, which is typically around 10 Hz. In order to obtain odometry estimates at higher frequencies for closed loop control, the latest available optimization motion state is propagated forward with more recent IMU observations according to the following recursive midpoint integration scheme:

$$\Delta t_k = t_{k+1} - t_k \tag{3.174}$$

$$\mathbf{a}_k = \mathbf{R}(\mathbf{q}_k)\left(\hat{\mathbf{a}}_k - \mathbf{b}_{a,k}\right) + \mathbf{g} \tag{3.175}$$

$$\bar{\boldsymbol{\omega}}_{k,k+1} = \frac{1}{2}\left(\hat{\boldsymbol{\omega}}_k + \hat{\boldsymbol{\omega}}_{k+1}\right) - \mathbf{b}_{\omega,k} \tag{3.176}$$

$$\mathbf{q}_{k+1} = \mathbf{q}_k \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\Delta t_k \bar{\boldsymbol{\omega}}_{k,k+1} \end{bmatrix} \tag{3.177}$$

$$\mathbf{a}_{k+1} = \mathbf{R}(\mathbf{q}_{k+1})\left(\hat{\mathbf{a}}_{k+1} - \mathbf{b}_{a,k}\right) + \mathbf{g} \tag{3.178}$$

$$\bar{\mathbf{a}}_{k,k+1} = \frac{1}{2}\left(\mathbf{a}_k + \mathbf{a}_{k+1}\right) \tag{3.179}$$

$$\mathbf{p}_{k+1} = \mathbf{p}_k + \Delta t_k \mathbf{v}_k + \frac{1}{2}\bar{\mathbf{a}}_{k,k+1}\Delta t_k^2 \tag{3.180}$$

$$\mathbf{v}_{k+1} = \mathbf{v}_k + \Delta t_k \bar{\mathbf{a}}_{k,k+1} \tag{3.181}$$

Whenever a new optimization motion state becomes available, the IMU-rate odometry estimate is updated by repropagating all IMU observations received since the delay-compensated timestamp of the sliding window's latest keyframe. The magnitude of the IMU-rate odometry discontinuities associated with these resets depends on the accuracy of camera extrinsics and time delay parameters as well as IMU bias estimates.

3.6 Marginalization

The difference between sliding window optimization and full batch estimation is that the former optimizes a subset of recent states and observations, while the latter optimizes the entire history of states and observations. Because problem size remains constant in sliding window optimization but grows linearly with time in full batch estimation, the former is more suitable for real-time performance on computationally constrained systems. In order to maintain a constant problem size, sliding window optimization culls less informative states from its factor graph as new states are added.

Marginalization is the process of encoding the information associated with removed states as constraints on remaining states in order to approximate the full batch optimization solution. Over successive sliding window optimizations, these additional constraints increase the sliding window factor graph's density, which in turn increases the Jacobian matrix's fill-in[8] and ultimately solve time. Although it is possible to sparsify the information matrix associated with the optimization to mitigate marginalization-induced fill-in while reducing linearization error [75–77], in many cases the sparsification utilizes an extra optimization whose computational cost outweighs the computational savings associated with a sparser information matrix. Consequently, we do not employ such techniques in this work for the sake of computational efficiency.

3.6.1 Marginalization as an Approximation of Batch Optimization

Consider a batch optimization problem where the full history of states can be partitioned into marginalized states ($\mathbf{x}_m$), remaining states ($\mathbf{x}_r$), and new states ($\mathbf{x}_n$). When the total number of states is sufficiently small (i.e. $\mathbf{x} = \{\mathbf{x}_m, \mathbf{x}_r\}$), the batch and sliding window optimization problem formulations are the same. As more states ($\mathbf{x}_n$) are added, batch estimation keeps all past states in its optimization but sliding window estimation discards $\mathbf{x}_m$ to keep the problem size constant.

The goal of marginalization is to approximate the batch optimization cost function that depends on $\mathbf{x}_m$, $\mathbf{x}_r$, and $\mathbf{x}_n$ with a sliding window cost function that only depends on $\mathbf{x}_r$ and $\mathbf{x}_n$. We first rewrite the batch optimization cost function as

$$c\left(\mathbf{x}_m, \mathbf{x}_r, \mathbf{x}_n\right) = c_m\left(\mathbf{x}_m, \mathbf{x}_r\right) + c_n\left(\mathbf{x}_r, \mathbf{x}_n\right) \tag{3.182}$$

where c_m contains all cost terms involving one or more states in $\mathbf{x}_m$ and c_n contains all other cost terms. In the sliding window framework, there are no cost terms involving states in both $\mathbf{x}_m$ and $\mathbf{x}_n$ because the former are discarded before the latter are incorporated.

[8] Fill-in refers to the number of non-zero entries in a matrix

With the cost factorization (3.182), the batch optimization problem can be rewritten as

$$\min_{\mathbf{x}_m,\mathbf{x}_r,\mathbf{x}_n} c\left(\mathbf{x}_m,\mathbf{x}_r,\mathbf{x}_n\right) = \min_{\mathbf{x}_r,\mathbf{x}_n}\left(\min_{\mathbf{x}_m} c\left(\mathbf{x}_m,\mathbf{x}_r,\mathbf{x}_n\right)\right) = \min_{\mathbf{x}_r,\mathbf{x}_n}\left(c_n\left(\mathbf{x}_r,\mathbf{x}_n\right) + \min_{\mathbf{x}_m} c_m\left(\mathbf{x}_m,\mathbf{x}_r\right)\right) \tag{3.183}$$

The second-order Taylor series approximation of c_m is

$$c_m\left(\mathbf{x}_m,\mathbf{x}_r\right) \approx c_m\left(\hat{\mathbf{x}}_m,\hat{\mathbf{x}}_r\right) + \begin{bmatrix}\mathbf{b}_m\\ \mathbf{b}_r\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\Delta\mathbf{x}_m\\ \Delta\mathbf{x}_r\end{bmatrix} + \frac{1}{2}\begin{bmatrix}\Delta\mathbf{x}_m\\ \Delta\mathbf{x}_r\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{A}_{mm} & \mathbf{A}_{mr}\\ \mathbf{A}_{rm} & \mathbf{A}_{rr}\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}_m\\ \Delta\mathbf{x}_r\end{bmatrix} \tag{3.184}$$

$$\Delta\mathbf{x}_m = \mathbf{x}_m - \hat{\mathbf{x}}_m$$
$$\Delta\mathbf{x}_r = \mathbf{x}_r - \hat{\mathbf{x}}_r$$

where $\mathbf{b}$ and $\mathbf{A}$ are respectively the Jacobian and Hessian of c_m evaluated at the old sliding window estimates $\hat{\mathbf{x}}_m$ and $\hat{\mathbf{x}}_r$.

We find a closed form solution to the approximation of the subproblem $\min_{\mathbf{x}_m} c_m\left(\mathbf{x}_m,\mathbf{x}_r\right)$ from (3.183) by setting the gradient of (3.184)'s right hand side to zero.

$$\begin{bmatrix}\mathbf{A}_{mm} & \mathbf{A}_{mr}\\ \mathbf{A}_{rm} & \mathbf{A}_{rr}\end{bmatrix}\begin{bmatrix}\Delta\mathbf{x}_m\\ \Delta\mathbf{x}_r\end{bmatrix} = -\begin{bmatrix}\mathbf{b}_m\\ \mathbf{b}_r\end{bmatrix} \tag{3.185}$$

Rearranging the upper block row of (3.185) yields the minimizing value of $\Delta\mathbf{x}_m$.

$$\Delta\mathbf{x}_m = -\mathbf{A}_{mm}^{-1}\left(\mathbf{b}_m + \mathbf{A}_{mr}\Delta\mathbf{x}_r\right) \tag{3.186}$$

Substitute (3.186) into (3.184) to obtain the minimum cost.

$$c_{m,\min}\left(\mathbf{x}_m,\mathbf{x}_r\right) \approx C + \tilde{\mathbf{b}}^{\mathrm{T}}\Delta\mathbf{x}_r + \frac{1}{2}\Delta\mathbf{x}_r^{\mathrm{T}}\tilde{\mathbf{A}}\Delta\mathbf{x}_r \tag{3.187}$$

$$\tilde{\mathbf{A}} = \mathbf{A}_{rr} - \mathbf{A}_{rm}\mathbf{A}_{mm}^{-1}\mathbf{A}_{mr} \tag{3.188}$$

$$\tilde{\mathbf{b}} = \mathbf{b}_r - \mathbf{A}_{rm}\mathbf{A}_{mm}^{-1}\mathbf{b}_m \tag{3.189}$$

Substitute (3.187) into (3.183) to eliminate the latter's dependence on $\mathbf{x}_m$.

$$\min_{\mathbf{x}_m,\mathbf{x}_r,\mathbf{x}_n} c\left(\mathbf{x}_m,\mathbf{x}_r,\mathbf{x}_n\right) \approx \min_{\mathbf{x}_r,\mathbf{x}_n}\left[c_n\left(\mathbf{x}_r,\mathbf{x}_n\right) + \tilde{\mathbf{b}}^{\mathrm{T}}\left(\mathbf{x}_r - \hat{\mathbf{x}}_r\right) + \frac{1}{2}\left(\mathbf{x}_r - \hat{\mathbf{x}}_r\right)^{\mathrm{T}}\tilde{\mathbf{A}}\left(\mathbf{x}_r - \hat{\mathbf{x}}_r\right)\right] \tag{3.190}$$

Note that on the right hand side of (3.190), information about $c_{m,\min}$ is encoded in $\tilde{\mathbf{A}}$, $\tilde{\mathbf{b}}$, and $\hat{\mathbf{x}}_r$. In sliding window estimation, these terms are stored after each optimization and used in the cost function of the next optimization to form the prior residual (Sect. 3.6.2).

The only approximation made when deriving the sliding window cost function (3.190) from the original batch cost function (3.183) is permanently replacing $c_m\left(\mathbf{x}_r,\mathbf{x}_r\right)$ with a second-order Taylor series evaluated at the previous optimization's final solution's marginalized states. The effect of this early linearization point fixation is small if the marginalized state estimates are very close to their true values at the time of marginalization, which is usually the case when the marginalized states are the oldest states in the sliding window. Consequently, marginalization enables sliding window estimators to maintain a constant problem size at the cost of a small loss of accuracy.

3.6.2 Prior Residual

The prior residual encodes constraints between current sliding window states that were used in previous sliding window residuals that involve marginalized states. In this section we rewrite the second and third terms of (3.190) as a squared norm (3.168).

We begin by taking the eigendecomposition of (3.188):

$$\tilde{\mathbf{A}} = \tilde{\mathbf{V}}\tilde{\mathbf{D}}\tilde{\mathbf{V}}^{-1} \tag{3.191}$$

$$\tilde{\mathbf{D}} = \text{diag}\left\{\tilde{\lambda}_1, \ldots, \tilde{\lambda}_{\dim(\Delta\mathbf{x}_r)}\right\} \tag{3.192}$$

$$\tilde{\mathbf{V}} = \begin{bmatrix}\tilde{\mathbf{v}}_1 & \ldots & \tilde{\mathbf{v}}_{\dim(\Delta\mathbf{x}_r)}\end{bmatrix} \tag{3.193}$$

$\tilde{\mathbf{V}}$ is an orthogonal matrix because $\tilde{\mathbf{A}}$ is symmetric:

$$\tilde{\mathbf{A}} = \tilde{\mathbf{V}}\tilde{\mathbf{D}}\tilde{\mathbf{V}}^{\text{T}} \tag{3.194}$$

$$\tilde{\mathbf{V}}\tilde{\mathbf{V}}^{\text{T}} = \mathbf{I} \tag{3.195}$$

Define diagonal matrices corresponding to the square root and inverse square root of $\tilde{\mathbf{D}}$:

$$\tilde{\mathbf{D}}^{\frac{1}{2}} = \text{diag}\left\{\sqrt{\tilde{\lambda}_1}, \ldots, \sqrt{\tilde{\lambda}_{\dim(\Delta\mathbf{x}_r)}}\right\} \tag{3.196}$$

$$\tilde{\mathbf{D}}^{-\frac{1}{2}} = \text{diag}\left\{\frac{1}{\sqrt{\tilde{\lambda}_1}}, \ldots, \frac{1}{\sqrt{\tilde{\lambda}_{\dim(\Delta\mathbf{x}_r)}}}\right\} \tag{3.197}$$

Complete the square on the second and third terms of (3.190) by adding a constant term.

$$f_{\text{prior}}(\mathbf{x}) = \frac{1}{2}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right)^{\text{T}} \tilde{\mathbf{A}} \left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right) + \tilde{\mathbf{b}}^{\text{T}}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right) + \frac{1}{2}\tilde{\mathbf{b}}^{\text{T}}\tilde{\mathbf{A}}^{-1}\tilde{\mathbf{b}} \tag{3.198}$$

Substitute (3.194)-(3.197) into (3.198) and factor:

$$f_{\text{prior}}(\mathbf{x}) = \frac{1}{2}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right)^{\text{T}} \tilde{\mathbf{V}}\tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{V}}^{\text{T}}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right) + \tilde{\mathbf{b}}^{\text{T}}\tilde{\mathbf{V}}\tilde{\mathbf{V}}^{\text{T}}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right) + \frac{1}{2}\tilde{\mathbf{b}}^{\text{T}}\tilde{\mathbf{V}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{V}}^{\text{T}}\tilde{\mathbf{b}} \tag{3.199}$$

$$f_{\text{prior}}(\mathbf{x}) = \frac{1}{2}\left\|\tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{V}}^{\text{T}}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right) - \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{V}}^{\text{T}}\tilde{\mathbf{b}}\right\|_2^2 \tag{3.200}$$

$$f_{\text{prior}}(\mathbf{x}) = \frac{1}{2}\left\|\mathbf{A}_{\text{prior}}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right) + \mathbf{b}_{\text{prior}}\right\|_2^2 \tag{3.201}$$

The prior residual is

$$\mathbf{r}_{\text{prior}}(\mathbf{x}) = \mathbf{A}_{\text{prior}}\left(\mathbf{x}_r \boxminus \hat{\mathbf{x}}_r\right) + \mathbf{b}_{\text{prior}} \tag{3.202}$$

The $\boxminus$ operator generalizes Euclidean vector space subtraction to manifolds and is equivalent to vector space subtraction for parameter blocks defined over Euclidean space. The Jacobian of the prior residual is:

$$\frac{\partial \mathbf{r}_{\text{prior}}}{\partial \delta\mathbf{x}_r} = \mathbf{A}_{\text{prior}} \tag{3.203}$$

3.6.3 Selection of States to Marginalize

One of the simplest strategies for maintaining a constant size sliding window is to marginalize the oldest states after each optimization. However, such a strategy renders the scale of features unobservable during hover intervals when the sliding window does not contain sufficient acceleration excitation. To address this issue, we use the two-way marginalization strategy described in [78].

Before a new keyframe is incorporated into the current sliding window, either the oldest keyframe is marginalized or the newest frame is dropped. If the current sliding window's two newest keyframes have sufficient parallax (i.e. their associated poses are separated by a sufficient translation), the oldest keyframe is marginalized. Otherwise, the newest keyframe is dropped without being marginalized[9].

This strategy keeps older states containing nontrivial acceleration excitation in the sliding window when the robot is hovering, rendering the scale observable. Old states are marginalized during dynamic motion because newer states have sufficient acceleration excitation. Note that during constant speed motion, there is no way to prevent scale drift without loop closure.

3.6.4 Implementation Details

This section describes the steps to form the prior residual from the previous optimization's final estimate ($\hat{\mathbf{x}}$), final Jacobian ($\mathbf{J}$), and final residual ($\mathbf{r}$).

The first step is to identify the subset of states to be marginalized ($\hat{\mathbf{x}}_m$). When marginalizing the oldest frame, $\hat{\mathbf{x}}_m$ consists of the first motion state ($\hat{\mathbf{p}}_1$, $\hat{\mathbf{q}}_1$, $\hat{\mathbf{v}}_1$), first IMU bias state ($\hat{\mathbf{b}}_{a,1}$, $\hat{\mathbf{b}}_{\omega,1}$), and inverse depths of feature trails first observed in the first frame $\{\hat{\rho}_i\}$.

Identify the residuals corresponding to c_m $(\hat{\mathbf{x}}_m, \hat{\mathbf{x}}_r)$ by finding all rows of $\mathbf{J}$ that have nonzero entries in any of the columns corresponding to $\hat{\mathbf{x}}_m$. Permute the rows of $\mathbf{J}$ and $\mathbf{r}$ to move all c_m residuals to a contiguous block at the top while preserving relative ordering within the c_m and non-c_m blocks.

Identify the $\hat{\mathbf{x}}_r$ states by finding columns of $\mathbf{J}$ not already corresponding to $\hat{\mathbf{x}}_m$ that have nonzero entries in any of the rows corresponding to c_m residuals. Permute the columns of $\mathbf{J}$ to move all $\hat{\mathbf{x}}_m$ to a contiguous block on the left, $\hat{\mathbf{x}}_r$ to a contiguous block in the center, and all remaining states to a contiguous block the right while preserving relative column ordering within each block. After row and column permutations, $\mathbf{J}$ and $\mathbf{r}$ should have the following block structure:

$$\mathbf{J} = \begin{bmatrix} \mathbf{J}_{mm} & \mathbf{J}_{mr} & * \\ * & * & * \end{bmatrix}, \quad \mathbf{r} = \begin{bmatrix} \mathbf{r}_m \\ * \end{bmatrix} \tag{3.204}$$

Construct the quantities used in (3.185):

$$\begin{bmatrix} \mathbf{A}_{mm} & \mathbf{A}_{mr} \\ \mathbf{A}_{rm} & \mathbf{A}_{rr} \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{mm}^{\mathrm{T}}\mathbf{J}_{mm} & \mathbf{J}_{mm}^{\mathrm{T}}\mathbf{J}_{mr} \\ \mathbf{J}_{mr}^{\mathrm{T}}\mathbf{J}_{mm} & \mathbf{J}_{mr}^{\mathrm{T}}\mathbf{J}_{mr} \end{bmatrix} \tag{3.205}$$

$$\begin{bmatrix} \mathbf{b}_m \\ \mathbf{b}_r \end{bmatrix} = \begin{bmatrix} \mathbf{J}_{mm}^{\mathrm{T}}\mathbf{r}_m \\ \mathbf{J}_{mr}^{\mathrm{T}}\mathbf{r}_m \end{bmatrix} \tag{3.206}$$

Compute the Schur complement quantities (3.188)-(3.189) and compute the eigendecomposition (3.191). When forming the prior residual (3.202), use the following expressions in place of (3.196)-(3.197):

$$\tilde{\mathbf{D}}^{\frac{1}{2}} = \mathrm{diag}\left\{ \tilde{\lambda}_1 > \epsilon \; ? \; \sqrt{\tilde{\lambda}_1} : 0, \ldots, \tilde{\lambda}_{\dim(\Delta\mathbf{x}_r)} > \epsilon \; ? \; \sqrt{\tilde{\lambda}_{\dim(\Delta\mathbf{x}_r)}} : 0 \right\} \tag{3.207}$$

$$\tilde{\mathbf{D}}^{-\frac{1}{2}} = \mathrm{diag}\left\{ \tilde{\lambda}_1 > \epsilon \; ? \; \frac{1}{\sqrt{\tilde{\lambda}_1}} : 0, \ldots, \tilde{\lambda}_{\dim(\Delta\mathbf{x}_r)} > \epsilon \; ? \; \frac{1}{\sqrt{\tilde{\lambda}_{\dim(\Delta\mathbf{x}_r)}}} : 0 \right\} \tag{3.208}$$

[9] Because the estimation error of newer states is typically higher than that of older states within a sliding window, not marginalizing newer states avoids the risk of adding constraints that are based on potentially inaccurate estimates.

$0 < \epsilon \ll 1$ is an eigenvalue threshold for numerical stability, while the ternary operator $a \; ? \; b : c$ returns b if predicate a is true and c otherwise.

When evaluating the prior residual in the next optimization, each parameter block in $\mathbf{x}_r$ must be associated with the matching parameter block in $\hat{\mathbf{x}}_r$. For blocks of $\mathbf{x}_r$ and $\hat{\mathbf{x}}_r$ corresponding to quaternion states, the $\boxminus$ operator in (3.202) is defined as

$$\mathbf{q}_r \boxminus \hat{\mathbf{q}}_r = \text{Log}\left(\hat{\mathbf{q}}_r^{-1} \otimes \mathbf{q}_r\right) \approx 2\text{vec}\left(\hat{\mathbf{q}}_r^{-1} \otimes \mathbf{q}_r\right) \tag{3.209}$$

3.7 Initialization

Visual-inertial bundle adjustment is a highly nonlinear optimization problem that requires accurate initialization in order to achieve acceptable performance. This section describes a procedure to estimate initial motion, gyroscope biases, and feature locations by aligning IMU preintegration with up-to-scale vision-based estimates of scene structure. The initialization procedure is based on [79] with minor modifications.

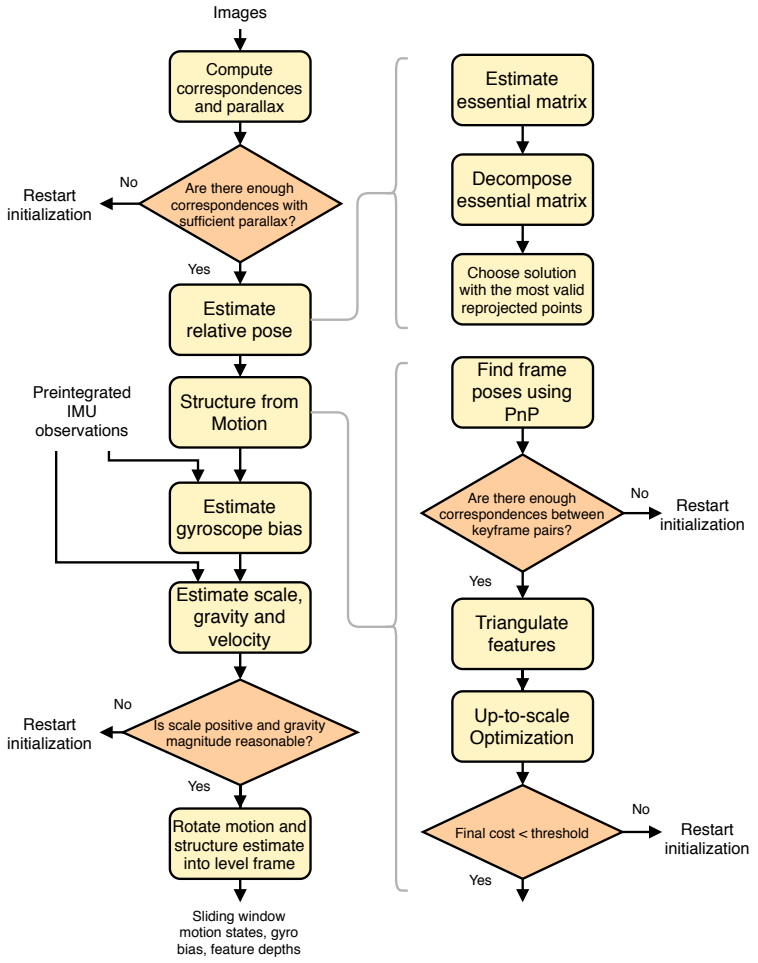


Figure 3.5: Flowchart of initialization process

3.7.1 Structure From Motion

The vision-only component of the initialization procedure estimates scaled camera poses and feature locations given a sliding window of image keyframes. This procedure is also known as Structure from Motion (SFM).

Relative Pose Estimation

The first step is to compute feature correspondences between the last keyframe and all previous frames. Starting from the oldest keyframe and moving forward in time, identify the first keyframe that has a sufficient number of feature correspondences and sufficient parallax with the last keyframe. These conditions ensure that most sliding window keyframes contain stable feature trails and non-trivial camera translation. Finally, find the relative rotation and arbitrarily scaled translation between the chosen pair of keyframes using the 5-point algorithm [80]. The 5-point algorithm's main steps are finding the essential matrix within a RANSAC scheme to reject outliers, decomposing the essential matrix into four rotation-translation pairs, and selecting the rotation-translation pair associated with the greatest number of positive depth points.

Initial Pose and Feature Estimation

Consider a sliding window of n keyframes used in initialization. Let l be the index of the first sliding window keyframe that has sufficient correspondences and parallax with the last keyframe. Set the pose of the last keyframe to the relative rotation and arbitrarily scaled translation found in Sect. 3.7.1. Set the pose of keyframe l to identity and triangulate features observed in keyframes l and n. Estimate the poses of all other sliding window keyframes relative to keyframe l by triangulating remaining features and solving perspective-n-point problems [81]. Note that when relying solely on feature correspondences, keyframe positions and triangulated feature positions can only be estimated up to an arbitrary scale factor.

Bundle Adjustment

The structure from motion bundle adjustment optimization problem is

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \sum_{l=1}^{L} \sum_{j=1}^{M_l} \left\| \mathbf{r}_{\text{sfm}} \left(\mathbf{p}_{k(l,j)}, \mathbf{q}_{k(l,j)}, \mathbf{l}_l, \mathbf{h}_{lj} \right) \right\|^2 \tag{3.210}$$

$$\mathbf{r}_{\text{sfm}} \left(\mathbf{p}, \mathbf{q}, \mathbf{l}, \mathbf{h} \right) = \mathbf{z} \left(\mathbf{p}' \right) - \mathbf{z} \left(\mathbf{h} \right) \tag{3.211}$$

$$\mathbf{p}' = \mathbf{R} \left(\mathbf{q} \right) \mathbf{l} + \mathbf{p} \tag{3.212}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_1^{\mathrm{T}} & \mathbf{q}_1^{\mathrm{T}} & \cdots & \mathbf{p}_n^{\mathrm{T}} & \mathbf{q}_n^{\mathrm{T}} & \mathbf{l}_1^{\mathrm{T}} & \cdots & \mathbf{l}_L^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \tag{3.213}$$

L is the number of successfully triangulated landmarks from Sect. 3.7.1, M_l is the number of sliding window keyframes that the l^{th} landmark was observed in, $\mathbf{z}(\cdot)$ is the image plane projection function defined in (3.131), $k(l, j)$ is a function that returns the keyframe index of the l^{th} landmark's j^{th} observation, $\mathbf{h}_{lj}$ is the homogeneous point associated with the l^{th} landmark's j^{th} observation, and n is the number of keyframes in the sliding window.

The optimization vector consists of the orientations and scaled positions of all sliding window keyframes, $\{\mathbf{p}_i, \mathbf{q}_i\}_{i=1}^{n}$, together with the scaled 3D positions of all successfully triangulated landmarks, $\{\mathbf{l}_l\}_{l=1}^{L}$. Due to the presence of quaternions in the optimization vector, manifold optimization (Appendix C.5) must be used. The SFM residual (3.211) represents the reprojection error of global frame landmarks with respect to their associated normalized image plane homogeneous point observations.

The Jacobians of the SFM residual are

$$\frac{\partial \mathbf{r}_{\text{sfm}}}{\partial \delta \mathbf{p}} = \frac{\partial \mathbf{z}(\mathbf{p}')}{\partial \mathbf{p}'} \tag{3.214}$$

$$\frac{\partial \mathbf{r}_{\text{sfm}}}{\partial \delta \boldsymbol{\theta}} = -\frac{\partial \mathbf{z}(\mathbf{p}')}{\partial \mathbf{p}'} \left[\mathbf{R}(\mathbf{q}) \mathbf{l} \right]_{\times} \tag{3.215}$$

$$\frac{\partial \mathbf{r}_{\text{sfm}}}{\partial \delta \mathbf{l}} = \frac{\partial \mathbf{z}(\mathbf{p}')}{\partial \mathbf{p}'} \mathbf{R}(\mathbf{q}) \tag{3.216}$$

where $\frac{\partial \mathbf{z}(\mathbf{p}')}{\partial \mathbf{p}'}$ is the image plane projection Jacobian defined in (3.144).

To avoid seeding the sliding window estimator with a bad initial guess, initialization is allowed to proceed beyond SFM bundle adjustment only if the final optimization cost is less than a user-defined threshold.

3.7.2 Gyroscope Bias Estimation

After obtaining a successful structure from motion bundle adjustment result, camera extrinsic parameters are used to convert camera frame orientations and scaled positions to their body frame equivalents. The SFM-derived rotation deltas between keyframes can be compared with IMU preintegration rotation deltas to estimate gyroscope bias.

The minimization problem for gyroscope bias estimation is

$$\min_{\mathbf{b}_\omega} \sum_{k=1}^{n-1} \left\| \text{Log}\left(\mathbf{q}_{k+1}^{-1} \otimes \mathbf{q}_k \otimes \boldsymbol{\gamma}_{k,k+1}(\mathbf{b}_\omega) \right) \right\|^2 \tag{3.217}$$

where $\{\mathbf{q}_k\}_{k=1}^{n-1}$ are the SFM-derived body-in-world orientations associated with keyframe images and the first order bias-corrected IMU preintegration rotation delta is

$$\boldsymbol{\gamma}_{k,k+1}(\mathbf{b}_\omega) \approx \hat{\boldsymbol{\gamma}}_{k,k+1} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{\theta,b_\omega}(k,k+1)\mathbf{b}_\omega \end{bmatrix} \tag{3.218}$$

$\hat{\boldsymbol{\gamma}}_{k,k+1}$ is the local frame delta rotation obtained by integrating IMU angular velocity observations without any bias compensation and $\mathbf{J}_{\theta,b_\omega}$ is defined in (3.71).

Achieving perfect alignment between SFM-derived and bias-compensated IMU-derived delta rotations results in the following condition:

$$\boldsymbol{\gamma}_{k,k+1}(\mathbf{b}_\omega) = \mathbf{q}_k^{-1} \otimes \mathbf{q}_{k+1} \tag{3.219}$$

Substitute (3.218) into (3.219):

$$\begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{\theta,b_\omega}(k,k+1)\mathbf{b}_\omega \end{bmatrix} \approx \hat{\boldsymbol{\gamma}}_{k,k+1}^{-1} \otimes \mathbf{q}_k^{-1} \otimes \mathbf{q}_{k+1} \tag{3.220}$$

Because the right hand side is supposed to be a very small rotation, we apply the small angle approximation

$$\begin{bmatrix} 1 \\ \frac{1}{2}\mathbf{J}_{\theta,b_\omega}(k,k+1)\mathbf{b}_\omega \end{bmatrix} \approx \begin{bmatrix} 1 \\ \text{vec}\left(\hat{\boldsymbol{\gamma}}_{k,k+1}^{-1} \otimes \mathbf{q}_k^{-1} \otimes \mathbf{q}_{k+1} \right) \end{bmatrix} \tag{3.221}$$

where $\text{vec}(\cdot)$ is the operator that extracts the vector component of a quaternion (A.25). Construct an overdetermined linear system to capture the constraint that SFM-derived and bias-compensated IMU-derived delta rotations should match between all successive keyframe pairs within the sliding window:

$$\begin{bmatrix} \mathbf{J}_{\theta,b_\omega}(1,2) \\ \vdots \\ \mathbf{J}_{\theta,b_\omega}(k,k+1) \\ \vdots \\ \mathbf{J}_{\theta,b_\omega}(n-1,n) \end{bmatrix} \mathbf{b}_\omega = \begin{bmatrix} 2\text{vec}\left(\hat{\boldsymbol{\gamma}}_{1,2}^{-1} \otimes \mathbf{q}_1^{-1} \otimes \mathbf{q}_2 \right) \\ \vdots \\ 2\text{vec}\left(\hat{\boldsymbol{\gamma}}_{k,k+1}^{-1} \otimes \mathbf{q}_k^{-1} \otimes \mathbf{q}_{k+1} \right) \\ \vdots \\ 2\text{vec}\left(\hat{\boldsymbol{\gamma}}_{n-1,n}^{-1} \otimes \mathbf{q}_{n-1}^{-1} \otimes \mathbf{q}_n \right) \end{bmatrix} \tag{3.222}$$

$$\mathcal{A}\mathbf{b}_\omega = \mathcal{B} \tag{3.223}$$

Find the $\mathbf{b}_\omega$ that minimizes $\|\mathcal{A}\mathbf{b}_\omega - \mathcal{B}\|$ by solving the normal equations

$$\left(\mathcal{A}^{\mathrm{T}}\mathcal{A}\right)\mathbf{b}_\omega = \mathcal{A}^{\mathrm{T}}\mathcal{B} \tag{3.224}$$

which can be rewritten as

$$\mathbf{A}\mathbf{b}_\omega = \mathbf{b} \tag{3.225}$$

$$\mathbf{A} = \sum_{k=1}^{n-1} \mathbf{J}_{\theta,b_w}(k,k+1)^{\mathrm{T}}\mathbf{J}_{\theta,b_\omega}(k,k+1) \tag{3.226}$$

$$\mathbf{b} = \sum_{k=1}^{n-1} \mathbf{J}_{\theta,b_w}(k,k+1)^{\mathrm{T}}\left[2\mathrm{vec}\left(\hat{\boldsymbol{\gamma}}_{k,k+1}^{-1} \otimes \mathbf{q}_k^{-1} \otimes \mathbf{q}_{k+1}\right)\right] \tag{3.227}$$

After solving the linear system (3.225) to obtain a gyroscope bias estimate, repropagate all IMU preintegration motion deltas (3.34)-(3.36) using the updated gyroscope bias estimate.

3.7.3 Velocity, Gravity, and Scale Estimation

This section describes a method for jointly estimating gravity in the first sliding window camera frame ($\mathbf{g}_c$), metric scale (s), and body frame velocities of all keyframes ($\{\mathbf{v}_k\}_{k=1}^n$) by aligning SFM-derived scene structure with IMU-derived preintegration motion deltas. The concatenated vector of estimated quantities is

$$\mathbf{x} = \begin{bmatrix}\mathbf{v}_1^{\mathrm{T}} & \dots & \mathbf{v}_n^{\mathrm{T}} & \mathbf{g}_c^{\mathrm{T}} & s\end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{3n+4} \tag{3.228}$$

Let $\{\mathbf{R}_k\}_{k=1}^n$ be the SFM-estimated rotation matrices that represent sliding window keyframe body orientations with respect to the first keyframe's camera frame. Let $\{\bar{\mathbf{p}}_k\}_{k=1}^n$ be the SFM-estimated scaled translations of sliding window keyframe camera positions with respect to the first keyframe's camera frame. Let $\Delta t_k = t_{k+1} - t_k$. The IMU preintegration position ($\hat{\boldsymbol{\alpha}}_{k,k+1}$) and velocity ($\hat{\boldsymbol{\beta}}_{k,k+1}$) deltas between keyframes k and $k+1$ are

$$\hat{\boldsymbol{\alpha}}_{k,k+1} = \mathbf{R}_k^{\mathrm{T}}\left[s\left(\bar{\mathbf{p}}_{k+1} - \bar{\mathbf{p}}_k\right) + \frac{1}{2}\mathbf{g}_c\Delta t_k^2 - \mathbf{R}_{k+1}\mathbf{p}_{bc}\right] + \mathbf{p}_{bc} - \Delta t_k\mathbf{v}_k \tag{3.229}$$

$$\hat{\boldsymbol{\beta}}_{k,k+1} = \mathbf{R}_k^{\mathrm{T}}\left[\mathbf{R}_{k+1}\mathbf{v}_{k+1} + \mathbf{g}_c\Delta t_k\right] - \mathbf{v}_k \tag{3.230}$$

where $\mathbf{p}_{bc}$ is the camera's position with respect to the body frame. Rewrite (3.229)-(3.230) as the following system of equations:

$$\begin{bmatrix} -\mathbf{I}\Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{R}_k^{\mathrm{T}}\Delta t_k^2 & \mathbf{R}_k^{\mathrm{T}}\left(\bar{\mathbf{p}}_{k+1} - \bar{\mathbf{p}}_k\right) \\ -\mathbf{I} & \mathbf{R}_k^{\mathrm{T}}\mathbf{R}_{k+1} & \mathbf{R}_k^{\mathrm{T}}\Delta t_k & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}_k \\ \mathbf{v}_{k+1} \\ \mathbf{g}_c \\ s \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{k,k+1} - \mathbf{p}_{bc} + \mathbf{R}_k^{\mathrm{T}}\mathbf{R}_{k+1}\mathbf{p}_{bc} \\ \hat{\boldsymbol{\beta}}_{k,k+1} \end{bmatrix} \tag{3.231}$$

Stack equations of the form (3.231) for $k = 1, \dots, n-1$ (i.e. between every pair of successive keyframes) to form a linear least squares problem in $\mathbf{x}$ and solve for velocity, gravity, and metric scale using the normal equations method.

The next step is to solve a modified version of the above problem subject to a gravity norm constraint. This requires the gravity vector to be parametrized by a 2D perturbation on the tangent space of the sphere with gravity norm radius at the point associated with the direction of the current gravity estimate. Let g be the known gravity magnitude and $\mathbf{g}_c$ be the gravity estimate from the previous linear least squares problem.

Let $\mathbf{b}_1$ and $\mathbf{b}_2$ be arbitrary orthogonal unit vectors that span the tangent space associated with the direction of $\mathbf{g}_c$. The magnitude-constrained parameterization of gravity has 2 degrees of freedom:

$$\mathbf{g}_c(w_1, w_2) = g\frac{\mathbf{g}_c}{\|\mathbf{g}_c\|} + w_1\mathbf{b}_1 + w_2\mathbf{b}_2 \tag{3.232}$$

Substituting (3.232) into (3.231) produces

$$\begin{bmatrix} -\mathbf{I}\Delta t_k & \mathbf{0} & \frac{1}{2}\mathbf{R}_k^{\mathrm{T}}\Delta t_k^2\begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix} & \mathbf{R}_k^{\mathrm{T}}\left(\bar{\mathbf{p}}_{k+1} - \bar{\mathbf{p}}_k\right) \\ -\mathbf{I} & \mathbf{R}_k^{\mathrm{T}}\mathbf{R}_{k+1} & \mathbf{R}_k^{\mathrm{T}}\Delta t_k\begin{bmatrix}\mathbf{b}_1 & \mathbf{b}_2\end{bmatrix} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{v}_k \\ \mathbf{v}_{k+1} \\ w_1 \\ w_2 \\ s \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{\alpha}}_{k,k+1} - \mathbf{p}_{bc} + \mathbf{R}_k^{\mathrm{T}}\left(\mathbf{R}_{k+1}\mathbf{p}_{bc} - \frac{\Delta t_k^2}{2}\mathbf{g}_c\right) \\ \hat{\boldsymbol{\beta}}_{k,k+1} - \mathbf{R}_k^{\mathrm{T}}\mathbf{g}_c\Delta t_k \end{bmatrix} \tag{3.233}$$

Stack equations of the form (3.233) for $k = 1, \ldots, n-1$ to form a linear least squares problem and solve for velocity, gravity perturbation, and metric scale using the normal equations method. Given w_1 and w_2 estimated from the above procedure, the magnitude-constrained gravity vector estimate is updated as

$$\mathbf{g}_c \leftarrow \frac{\mathbf{g}_c'(w_1, w_2)}{\|\mathbf{g}_c'(w_1, w_2)\|}g \tag{3.234}$$

The constrained gravity magnitude linear system is solved several times until the gravity estimate converges.

To avoid seeding the optimization with a bad initial guess, initialization is allowed to proceed beyond velocity, gravity, and scale estimation only if the final scale estimate is positive.

3.7.4 Initial Guess for Optimization

After successfully estimating gravity and scale, scale sliding window positions and velocities as well as feature depths to metric units.

The world frame used for sliding window optimization is a level frame[10] with the same origin location and heading direction as the body frame associated with the first keyframe in the sliding window used in a successful initialization. Consequently, the rotation $\mathbf{R}_{wc}$ between the camera world frame used in initialization and the level world frame used for sliding window optimization has zero yaw and can be computed by rotating $\mathbf{g}_c$ to the vector $\begin{bmatrix}0 & 0 & \|\mathbf{g}_c\|\end{bmatrix}^{\mathrm{T}}$. Rotate all sliding window position, velocity, and quaternion states to be in the world frame and populate the initial optimization vector with them. Also populate the initial optimization vector with gyroscope biases and inverse metric feature depths. Note that the initial optimization vector's acceleration biases are set to zero because they are not estimated during initialization.

3.8 Public Dataset Evaluation

We evaluate the proposed single camera sliding window state estimator on the EuRoC dataset [82]. The EuRoC dataset consists of 11 trajectories recorded on an aerial robot with an IMU and a stereo camera pair inside two different rooms (V1, V2) and an industrial environment (MH). Within each environment, the trajectories are classified as easy, medium, or difficult depending on how aggressive the motions are and the presence of unfavorable lighting conditions. Although images from a stereo pair of cameras are available, we only use images from the left camera to evaluate monocular visual-inertial odometry.

We compare the proposed approach against VINS-Mono [5], a recent state-of-the-art monocular visual-inertial state estimator[11]. In order to compare only the fixed lag smoothing functionality of VINS-Mono

[10] A *level frame* is a frame whose z-axis is aligned with the gravity vector

[11] Open source code available at https://github.com/HKUST-Aerial-Robotics/VINS-Mono

with our approach, we disabled VINS-Mono's loop closure module. Additionally, VINS-Mono's online extrinsics and time delay calibration functionalities were disabled. To facilitate a fair comparison, we fixed an error[12] in VINS-Mono's IMU preintegration noise Jacobian. Lastly, we rectified camera images before passing them to VINS-Mono instead of utilizing its non-pinhole distortion models to bring it in line with the visual front-end implemented in the proposed approach.

All results were generated via postprocessing on a Lenovo Thinkpad T470p laptop[13] running Ubuntu 16.04 LTS. The multiple position correspondences trajectory alignment method (Appendix H.3.2) and root mean square absolute trajectory error (H.13) were used to evaluate each estimated trajectory against ground truth. Table 3.1 shows that our proposed approach slightly outperforms VINS-Mono on 10 out of the 11 EuRoC trajectories. These results indicate that the single-camera formulation and implementation of our approach is comparable to a state-of-the-art strategy and thus constitutes a suitable baseline for extensions detailed in subsequent chapters.

Table 3.1: EuRoC dataset root mean square absolute trajectory error in meters

Trajectory	VINS-Mono	Proposed
V1_01_easy	0.116	**0.114**
V1_02_medium	0.120	**0.116**
V1_03_difficult	0.260	**0.217**
V2_01_easy	0.134	**0.132**
V2_02_medium	0.218	**0.200**
V2_03_difficult	0.378	**0.364**
MH_01_easy	0.165	**0.162**
MH_02_easy	0.168	**0.158**
MH_03_medium	0.459	**0.418**
MH_04_difficult	**0.436**	0.437
MH_05_difficult	0.425	**0.389**

3.9 Realtime Evaluation on Flight Platform

This section presents real-time state estimation performance results on flight experiments conducted with a hexrotor aerial robot (Sect. G.1.1) and quadrotor aerial robot (Sect. G.1.2).

3.9.1 Motion Capture Flight

The trajectory in the motion capture (Sect. G.1.3) arena consists of horizontal and vertical line segments at various speeds over two minutes (Fig. 3.6). Although the robot's controller used motion capture odometry for repeatability, the visual-inertial state estimation pipeline was running passively on the onboard computer during flight.

The initial pose trajectory alignment method (Appendix H.3.1) was used to align the state estimate with motion capture ground truth. All estimated quantities depicted in this section's plots are aligned with motion capture ground truth. The notation used in the captions of this section's figures is defined in Appendix H.3.1.

The final position drift (as defined in Appendix H.4.2) is 14.6 cm over 52.9 m, which is equivalent to 0.28% of the entire trajectory. The translational and rotational absolute trajectory errors, given by (H.13)-(H.14), are 11.4 cm and 2.0 deg, respectively.

[12]Missing negative signs in lines 113-115 and 117 of https://github.com/HKUST-Aerial-Robotics/VINS-Mono/blob/master/vins_estimator/src/factor/integration_base.h as of November 1, 2019.

[13]Intel Core i7-7820HQ 2.9 GHz × 8, 32 GB RAM

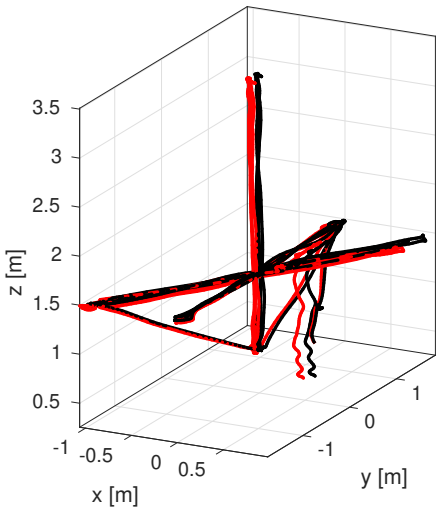
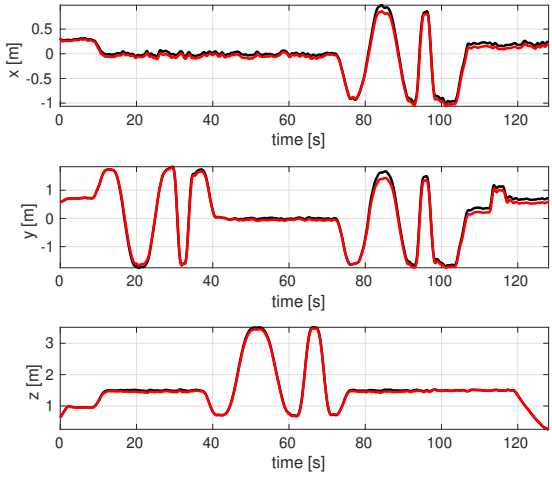
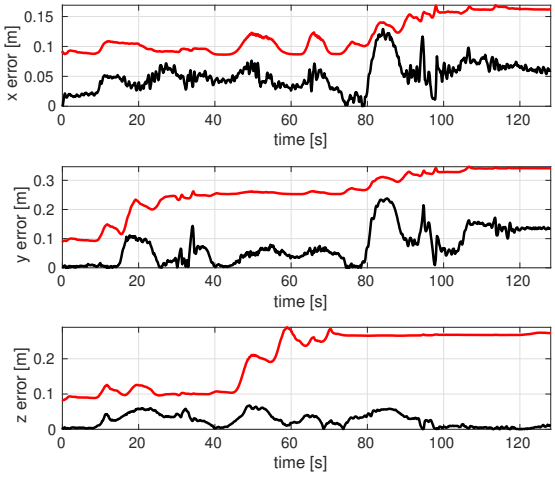


Figure 3.6: A 3D plot of the trajectory used for state estimation performance evaluation. Estimated position is in red, while motion capture capture ground truth is in black.

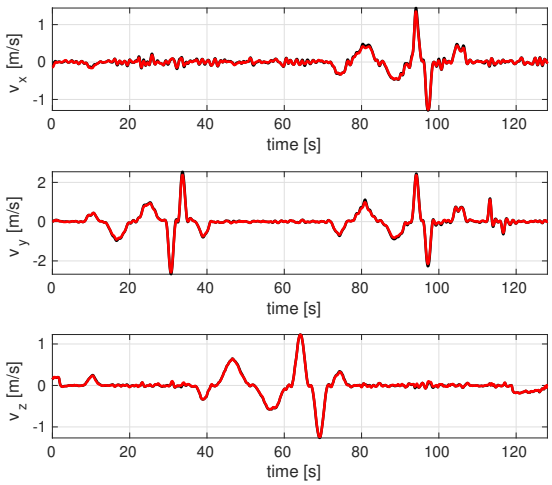


(a) Aligned position estimate $\mathbf{p}_{\text{al}}(t)$ (red) vs. motion capture ground truth position $\mathbf{p}_{\text{gt}}(t)$ (black)

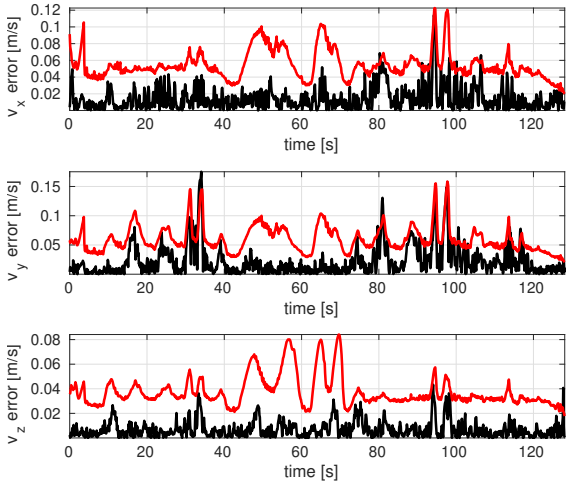


(b) Aligned position estimate covariance 3 σ envelope $3\sqrt{\text{diag}(\mathbf{\Sigma}_{\text{pp,al}}(t))}$ (red) vs. absolute value of position error with respect to ground truth $|\mathbf{p}_{\text{al}}(t) - \mathbf{p}_{\text{gt}}(t)|$ (black)

Figure 3.7: Position estimation performance

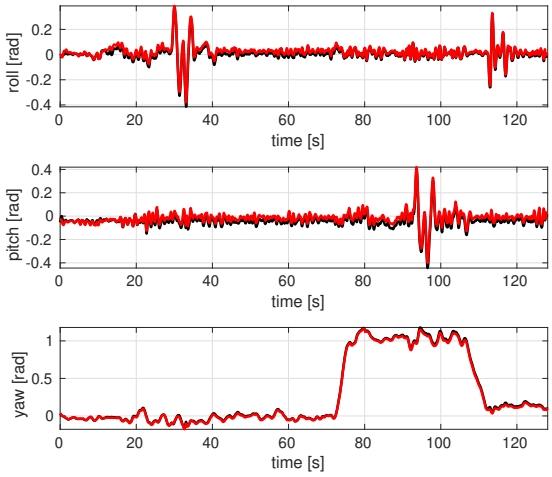


(a) Aligned velocity estimate $\mathbf{v}_{\text{al}}(t)$ (red) vs. motion capture ground truth velocity $\mathbf{v}_{\text{gt}}(t)$ (black)

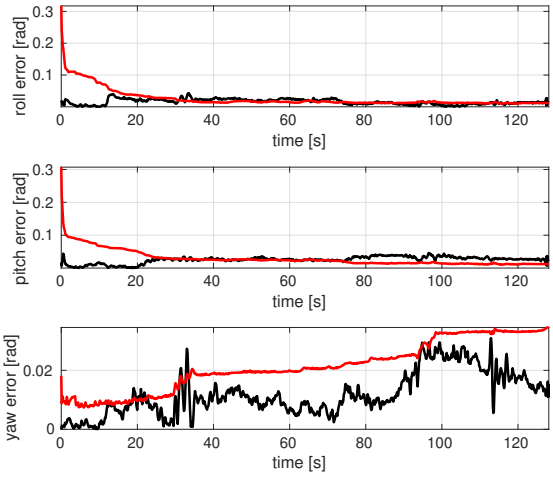


(b) Aligned velocity estimate covariance 3 σ envelope $3\sqrt{\text{diag}(\mathbf{\Sigma}_{\text{vv,al}}(t))}$ (red) vs. absolute value of velocity error with respect to ground truth $|\mathbf{v}_{\text{al}}(t) - \mathbf{v}_{\text{gt}}(t)|$ (black)

Figure 3.8: Velocity estimation performance



(a) Aligned attitude estimate $[\phi_{\text{al}}(t), \theta_{\text{al}}(t), \psi_{\text{al}}(t)]$ (red) vs. motion capture ground truth velocity $[\phi_{\text{gt}}(t), \theta_{\text{gt}}(t), \psi_{\text{gt}}(t)]$ (black)



(b) Attitude estimate covariance 3 σ envelope $3\sqrt{\text{diag}(\mathbf{\Sigma}_{\text{rr}}(t))}$ (red) vs. absolute value of ZYX Euler angle error with respect to ground truth $[|\phi_{\text{al}}(t) - \phi_{\text{gt}}(t)|, |\theta_{\text{al}}(t) - \theta_{\text{gt}}(t)|, |\psi_{\text{al}}(t) - \psi_{\text{gt}}(t)|]$ (black)

Figure 3.9: Attitude estimation performance

The 3 σ envelopes associated with the position, velocity, and yaw covariance estimates (Fig. 3.7b, 3.8b, 3.9b) are greater than their corresponding estimation errors for most of the flight. This indicates that the estimator is consistent in those dimensions of the state space. On the other hand, roll and pitch covariances are sometimes lower than their corresponding estimation errors due to minor motion capture model misalignments. Position and yaw covariances increase over time, which accurately reflects the increase in uncertainty in the unobservable directions of a visual-inertial odometry system. Velocity covariances are strongly correlated with speed because higher speeds at the same scene depth tend to result in shorter feature trails, which reduces the number of reprojection residuals in the optimization problem that can counteract motion drift in the IMU residuals.

3.9.2 Outdoor Flight

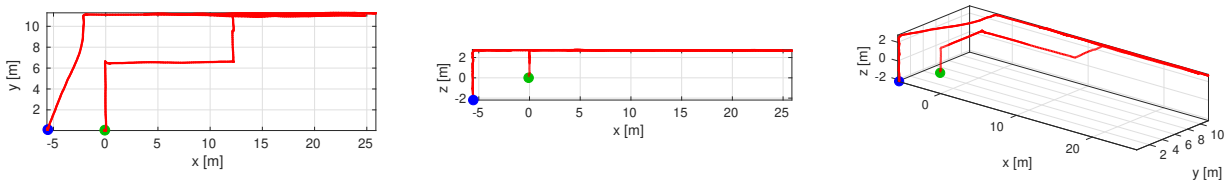
This section presents real-time state estimation performance results on flight experiments conducted with a hexrotor aerial robot in and above a forested outdoor environment. In these flights, the visual-inertial state estimator's output was used by the local planner and controller to enable the robot to track teleoperation motion primitives associated with the operator's joystick inputs. Because no ground truth was available for these outdoor flights, final position drift (Appendix H.4.2) was used to assess state estimation performance.

Treetop Flight

This flight consisted of horizontal line segments at a constant altitude and heading above the treetops of a wooded area (Fig. 3.10-3.12). The total distance traveled is approximately 129 m, while the final position drift is 2.4 m (2% of estimated path length). Given that the takeoff and landing locations have the same elevation, Figure 3.12b indicates that most of the final position drift is in the vertical direction.

Figure 3.10: Motion overlay of robot flight over treetops

Figure 3.11: Estimated flight path over treetops superimposed on satellite imagery



(a) Top-down view (b) Elevation view (c) 3D view

Figure 3.12: Estimated flight path over treetops

Forest and Treetop Flight

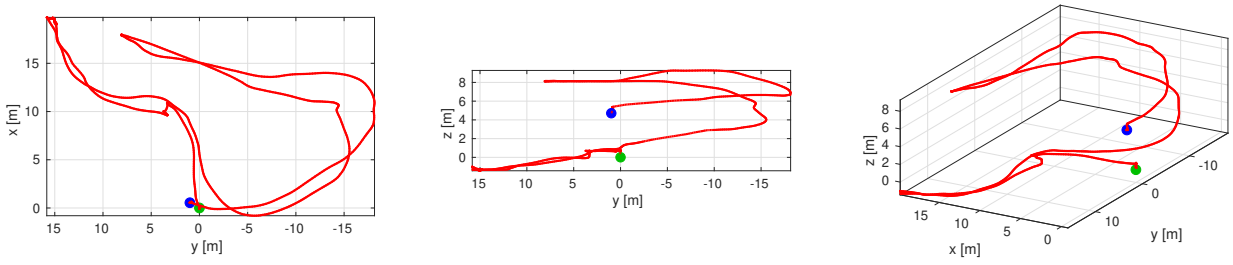
This flight consists of an initial segment along a paved forest path and a final segment over the forest treetops (Fig. 3.13-3.15). The total distance traveled is approximately 167 m, while the final position drift is 5 m (3% of estimated path length). Given that the takeoff and landing locations are at the same elevation, Figure 3.15b indicates that most of the final position drift is in the vertical direction.

Figure 3.13: Motion overlay of robot flight along forest path

3.9.3 Cave Flight

This section presents real-time state estimation performance results on flight experiments conducted on a quadrotor aerial robot inside a natural cave. In this flight, the visual-inertial state estimator's output was used by a planner and controller to enable the robot to autonomously explore an underground chamber that forms part of the Laurel Caverns cave system in Southwestern Pennsylvania. Onboard lights provided sufficient illumination to enable feature tracking on the downward camera over a rocky environment (Fig. 3.16). Because no ground truth was available for this flight, final position drift (Appendix H.4.2) was used to assess state estimation performance. The total distance traveled is approximately 46 m, while the final position drift is 0.47 m, or 1% of estimated path length (Fig. 3.17).

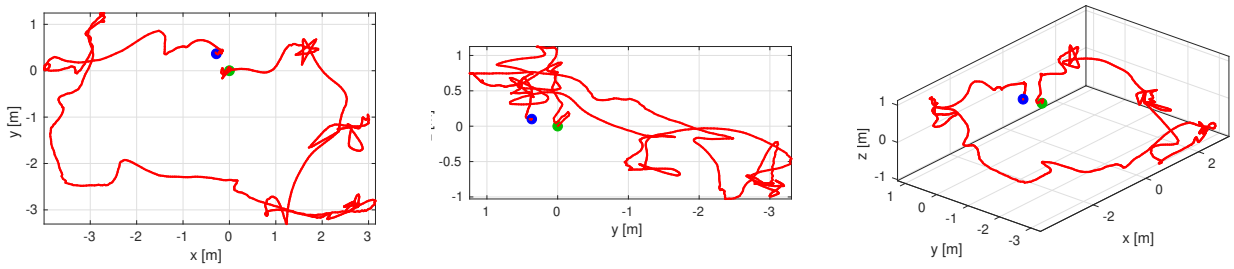
Figure 3.14: Motion overlay of robot flying from inside the forest to above the treetops



(a) Top-down view (b) Elevation view (c) 3D view

Figure 3.15: Estimated flight path inside forest and over treetops

Figure 3.16: Aerial robot flying autonomously in a chamber of the Laurel Caverns cave system in Southwestern Pennsylvania.



(a) Top-down view (b) Elevation view (c) 3D view

Figure 3.17: Estimated flight path inside cave chamber

Chapter 4

Practical Considerations for VIO Deployment in Real World Scenarios

This chapter describes a number of techniques that improve the accuracy and consistency of sliding window visual-inertial odometry as well as enable it to be used for closed loop control of multirotor aerial robots. Section 4.1 describes a computationally efficient covariance computation method, while Section 4.2 describes a way to maintain covariance consistency by enforcing visual-inertial odometry's lack of observability in global position and yaw. Section 4.3 describes an auxiliary estimator used for closed loop control during takeoff and before the primary optimization-based estimator has initialized. Section 4.4 describes a finite state machine for switching between primary and auxiliary estimator odometry sources for closed loop control as well as triggering mid-air reinitialization when the optimization-based state estimate becomes degraded.

4.1 Efficient Covariance Estimation

The covariance associated with the solution of a nonlinear least squares (NLS) optimization problem can be found by taking the inverse of its final iteration's Hessian matrix. However, sliding window bundle adjustment problems typically involve hundreds of variables and thousands of residuals, which yield Hessians that are too large to invert in real time with limited computational resources.

To obtain the covariance matrix more efficiently, we employ [83]'s method of first performing a Cholesky decomposition on the Hessian matrix and using the result to recursively compute entries of the inverse Hessian starting from the bottom right corner. Although this method still takes a long time to compute the entire covariance matrix, it is very quick to compute the covariance matrix's bottom right corner. Given this property, we exploit the fact that we are typically only interested in the covariance of the most recent pose and twist in the sliding window optimization and permute the Hessian's rows and columns so that the block corresponding to the latest motion state is at the bottom right corner prior to performing the Cholesky decomposition. Consequently, the cost of computing motion state covariance is reduced to the cost of a Cholesky decomposition on the NLS problem's Hessian matrix.

4.2 Gauge Ambiguity Handling

Gauge ambiguity refers to the unobservability of global position and yaw in a visual-inertial odometry system [84]. If all poses and feature locations are translated and rotated about the gravity axis by the same amount, the visual-inertial optimization problem's IMU residuals and reprojection residuals remain unchanged. On the other hand, roll and pitch are observable because applying a roll or pitch rotation to all estimates changes their gravity alignment and hence IMU residual values.

4.2.1 Problems Caused By Gauge Ambiguity

Gauge ambiguity causes the visual-inertial optimization problem to have infinitely many minimizers. During the nonlinear least squares regression, the Hessian matrix of a cost function comprised solely of IMU and reprojection residuals has a rank deficiency of 4 corresponding to global position and yaw (see Fig. 4.2). This makes it difficult to solve the linear system for the Gauss-Newton step during individual optimization iterations. Additionally, the rank deficiency caused by gauge ambiguity prevents covariance computation, which requires inverting the final Hessian.

4.2.2 Gauge Ambiguity Mitigation Strategies

There are three main approaches to mitigate the problems caused by gauge ambiguity.

Free Gauge

The free gauge approach used in VINS-Mono [5] relies on Levenberg-Marquardt damping to overcome Hessian rank deficiency when computing Gauss-Newton steps. This approach suffers from reduced step accuracy, which may potentially cause the optimization's iterations to diverge. To prevent optimization solution drift along the four unobservable directions, the free gauge approach applies a 4DOF transform to the final solution to align with the position and yaw of the initial guess.

The free gauge approach[1] computes covariance by taking the pseudo-inverse of the rank deficient Hessian. However, this produces a geometrically meaningless quantity where uncertainty is spread out over all poses (see Fig. 4.3). Although [85] proposed a method to transform free gauge covariance into a geometrically meaningful quantity, it requires extra computationally expensive matrix multiplications

Fixed Gauge

The fixed gauge approach reparameterizes the optimization variables such that the first sliding window frame's attitude is decomposed into a roll/pitch component and a yaw component. This breakdown enables the first frame's position and yaw to be held constant during optimization, which reduces the problem dimension by four and makes the Hessian full rank. Although gauge fixation enables Gauss-Newton step and covariance computation, parameterizing the first sliding window keyframe's orientation differently from all other keyframes increases implementation complexity.

Gauge Prior

The gauge prior approach adds a residual term to the visual-inertial odometry cost function that acts as a soft constraint to keep the sliding window's first position and yaw constant. This extra term, called a gauge prior, eliminates the Hessian's 4D nullspace and enables numerically stable covariance computation[2] (see Fig. 4.1). The advantage of using a gauge prior over gauge fixation is that it does not require any reparameterization of optimization variables. Additionally, it enables the user to specify the confidence of the initial position and yaw estimate via a prior covariance matrix that weights gauge prior relative to other cost function terms. For these reasons, we use the gauge prior approach to handle gauge ambiguity in our visual-inertial odometry system.

4.2.3 Gauge Prior Implementation Details

Let $\mathbf{p}$ and $\mathbf{q}$ be the components of the current optimization vector corresponding respectively to the first position vector and orientation quaternion of the sliding window. Let $\mathbf{p}_0$ and $\mathbf{q}_0$ be their corresponding initial values.

[1] Note that VINS-Mono does not compute covariance.

[2] The gauge prior only needs to be used in the very first sliding window if marginalization is enabled because it gets embedded in the marginalization residuals of subsequent sliding window optimizations.

Cost Function

The gauge prior cost function is given by

$$f_{\text{prior}} = \frac{1}{2} \left\| \mathbf{r}_{\text{prior}} \right\|^2_{\mathbf{\Sigma}_{\text{prior}}} \tag{4.1}$$

$$f_{\text{prior}} = \frac{1}{2} \mathbf{r}^{\text{T}}_{\text{prior}} \mathbf{\Sigma}^{-1}_{\text{prior}} \mathbf{r}_{\text{prior}} \tag{4.2}$$

$$f_{\text{prior}} = \frac{1}{2} \left\| \mathbf{L}^{\text{T}}_{\text{prior}} \mathbf{r}_{\text{prior}} \right\|^2 \tag{4.3}$$

where $\mathbf{\Sigma}^{-1}_{\text{prior}} = \mathbf{L}_{\text{prior}} \ \mathbf{L}^{\text{T}}_{\text{prior}}$ and the prior covariance is diagonal:

$$\mathbf{\Sigma}_{\text{prior}} = \text{diag} \left\{ \sigma^2_{x0}, \sigma^2_{y0}, \sigma^2_{z0}, \sigma^2_{\psi 0} \right\} \tag{4.4}$$

In actual flight experiments, the prior uncertainty take on the following values:

$$\sigma_{x0} = \sigma_{y0} = \sigma_{y0} = 0.01 \text{ m} \tag{4.5}$$

$$\sigma_{\psi 0} = 1 \times 10^{-4} \text{ rad} \tag{4.6}$$

Residual

The position component of the prior residual is simply vector subtraction, while the yaw component is the signed shortest angular distance between the initial yaw and the yaw estimate.

$$\mathbf{r}_{\text{prior}} = \begin{bmatrix} \mathbf{p} - \mathbf{p}_0 \\ \text{shortest\_angular\_distance} \left(\text{QuatToYaw} \left(\mathbf{q}_0 \right), \text{QuatToYaw} \left(\mathbf{q} \right) \right) \end{bmatrix} \tag{4.7}$$

The QuatToYaw function is given by (A.40).

Jacobian

The prior Jacobian consists of the prior residuals' partial derivatives with respect to perturbations $\delta \mathbf{p} \in \mathbb{R}^3$ and $\delta \boldsymbol{\theta} \in \mathbb{R}^3$ of the first position and orientation. The perturbed position and quaternion are expressed as

$$\mathbf{p} = \bar{\mathbf{p}} + \delta \mathbf{p} \tag{4.8}$$

$$\mathbf{q} = \bar{\mathbf{q}} \otimes \text{Exp}(\delta \boldsymbol{\theta}) \approx \bar{\mathbf{q}} \otimes \begin{bmatrix} 1 \\ \frac{1}{2} \delta \boldsymbol{\theta} \end{bmatrix} \tag{4.9}$$

Substitute the above expressions into the prior residual and differentiate with respect to the perturbations to obtain the partial derivatives that make up the prior Jacobian. The partials of the position residual and the partial of the yaw residual with respect to the position perturbation are trivial.

$$\frac{\partial \mathbf{r}_{\text{prior}}(1:3)}{\partial \delta \mathbf{p}} = \mathbf{I}_{3 \times 3} \tag{4.10}$$

$$\frac{\partial \mathbf{r}_{\text{prior}}(1:3)}{\partial \delta \boldsymbol{\theta}} = \mathbf{0}_{3 \times 3} \tag{4.11}$$

$$\frac{\partial \mathbf{r}_{\text{prior}}(4)}{\partial \delta \mathbf{p}} = \mathbf{0}_{1 \times 3} \tag{4.12}$$

The partial of the yaw residual with respect to the attitude perturbation is more involved.

$$\mathbf{r}_{\text{prior}}(4) = \text{shortest\_angular\_distance} \left(\text{QuatToYaw} \left(\mathbf{q}_0 \right), \text{QuatToYaw} \left(\mathbf{q} \right) \right) \tag{4.13}$$

Because additive offsets do not influence the derivative, we take the derivative of the related function

$$f = \text{QuatToYaw}\left(\mathbf{q}\right) - \text{QuatToYaw}\left(\mathbf{q}_0\right) \tag{4.14}$$

The second term is a constant, so we ignore it for the purpose of taking derivatives with respect to $\delta\boldsymbol{\theta}$. Call the remaining term g:

$$g = \text{atan2}\left(b, a\right) \tag{4.15}$$
$$a = 1 - 2(q_y^2 + q_z^2) \tag{4.16}$$
$$b = 2(q_w q_z + q_x q_y) \tag{4.17}$$

where

$$\begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \bar{q}_w \\ \bar{q}_x \\ \bar{q}_y \\ \bar{q}_z \end{bmatrix} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\delta\boldsymbol{\theta} \end{bmatrix} \tag{4.18}$$

We apply the chain rule for total derivatives to

$$\frac{\partial g}{\partial \delta\boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial g}{\partial \delta\theta_x} & \frac{\partial g}{\partial \delta\theta_y} & \frac{\partial g}{\partial \delta\theta_z} \end{bmatrix} \tag{4.19}$$
$$\frac{\partial g}{\partial \delta\boldsymbol{\theta}} = \begin{bmatrix} \frac{\partial g}{\partial a}\frac{\partial a}{\partial \delta\theta_x} + \frac{\partial g}{\partial b}\frac{\partial b}{\partial \delta\theta_x} & \frac{\partial g}{\partial a}\frac{\partial a}{\partial \delta\theta_y} + \frac{\partial g}{\partial b}\frac{\partial b}{\partial \delta\theta_y} & \frac{\partial g}{\partial a}\frac{\partial a}{\partial \delta\theta_z} + \frac{\partial g}{\partial b}\frac{\partial b}{\partial \delta\theta_z} \end{bmatrix} \tag{4.20}$$

where

$$\frac{\partial g}{\partial a} = \frac{-b}{a^2 + b^2} \tag{4.21}$$
$$\frac{\partial g}{\partial b} = \frac{a}{a^2 + b^2} \tag{4.22}$$
$$\frac{\partial a}{\partial \delta\theta_x} = -\left(\bar{q}_y^2 + \bar{q}_z^2\right)\delta\theta_x + \left(\bar{q}_x\bar{q}_y - \bar{q}_w\bar{q}_z\right)\delta\theta_y + \left(\bar{q}_w\bar{q}_y + \bar{q}_x\bar{q}_z\right)\delta\theta_z \tag{4.23}$$
$$\frac{\partial a}{\partial \delta\theta_y} = \left(\bar{q}_x\bar{q}_y - \bar{q}_w\bar{q}_z\right)\delta\theta_x - \left(\bar{q}_w^2 + \bar{q}_x^2\right)\delta\theta_y - 2\left(\bar{q}_w\bar{q}_y + \bar{q}_x\bar{q}_z\right) \tag{4.24}$$
$$\frac{\partial a}{\partial \delta\theta_z} = \left(\bar{q}_w\bar{q}_y + \bar{q}_x\bar{q}_z\right)\delta\theta_x - \left(\bar{q}_w^2 + \bar{q}_x^2\right)\delta\theta_z + 2\left(\bar{q}_x\bar{q}_y - \bar{q}_w\bar{q}_z\right) \tag{4.25}$$
$$\frac{\partial b}{\partial \delta\theta_x} = \left(\bar{q}_w\bar{q}_z + \bar{q}_x\bar{q}_y\right)\delta\theta_x + \frac{1}{2}\left(\bar{q}_w^2 + \bar{q}_y^2 - \bar{q}_x^2 - \bar{q}_z^2\right)\delta\theta_y + \left(\bar{q}_y\bar{q}_z - \bar{q}_w\bar{q}_x\right)\delta\theta_z \tag{4.26}$$
$$\frac{\partial b}{\partial \delta\theta_y} = \frac{1}{2}\left(\bar{q}_w^2 + \bar{q}_y^2 - \bar{q}_x^2 - \bar{q}_z^2\right)\delta\theta_x - \left(\bar{q}_w\bar{q}_z + \bar{q}_x\bar{q}_y\right)\delta\theta_y + 2\left(\bar{q}_w\bar{q}_x - \bar{q}_y\bar{q}_z\right) \tag{4.27}$$
$$\frac{\partial b}{\partial \delta\theta_z} = \left(\bar{q}_y\bar{q}_z - \bar{q}_w\bar{q}_x\right)\delta\theta_x - \left(\bar{q}_w\bar{q}_z + \bar{q}_x\bar{q}_y\right)\delta\theta_z + \left(\bar{q}_w^2 + \bar{q}_y^2 - \bar{q}_x^2 - \bar{q}_z^2\right) \tag{4.28}$$

4.2.4 Numerical Example

The effect of using a gauge prior term on the visual-inertial bundle adjustment optimization problem's Hessian matrix is shown in Figure 4.1, which compares singular values. Singular values and singular vectors obtained by performing singular value decomposition on the Hessian provide information about its effective rank and nullspace. The bottom plot of Figure 4.1 shows that the four smallest singular values of the no-gauge-prior Hessian are three orders of magnitude smaller than the next smallest singular value. Because

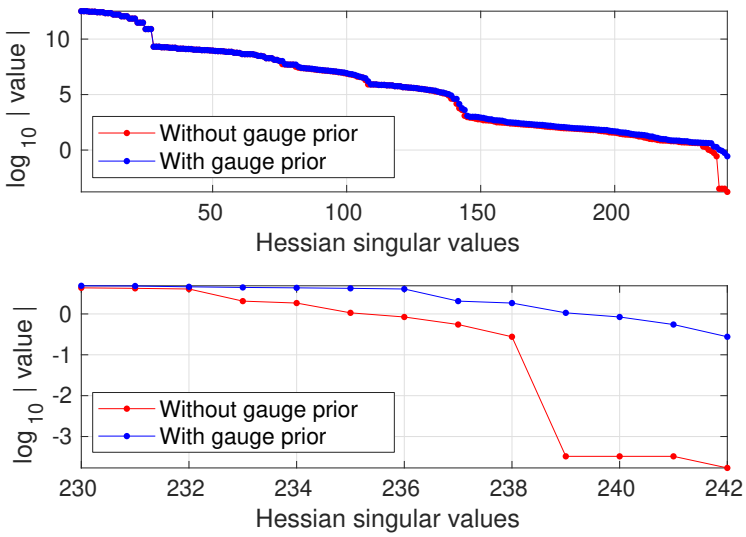


Figure 4.1: Hessian singular values with and without a gauge prior. The lower plot shows the smallest singular values at the extreme right of the upper plot.

these singular values are very close to zero, the effective rank of the no-gauge-prior Hessian is four less than its dimension. On the other hand, using a gauge prior causes the Hessian to be full rank, as shown by the lack of near-zero singular values for the with-gauge-prior Hessian in Figure 4.1.

The effective nullspace spanned by the singular vectors corresponding to the no-gauge-prior Hessian's four near-zero singular values is depicted in Figure 4.2. In this numerical example, the visual-inertial bundle adjustment problem involves 10 sliding window keyframes, each of which is parameterized by 15 local states. Within each keyframe's 15×1 parameter block, the first three components represent position while the next three components represent attitude. Figure 4.2 shows that the effective nullspace basis vectors have nonzero components corresponding to the position and yaw of all keyframes, which implies that the no-gauge-prior Hessian's effective nullspace corresponds to global position and heading.

Figure 4.3 depicts the 1-σ uncertainty envelopes associated with the position and yaw states of sliding window keyframes when covariance is computed by taking the pseudo-inverse of the no-gauge-prior Hessian and when it is computed by inverting the with-gauge-prior Hessian. The former approach distributes uncertainty across keyframes, while the latter approach assigns $\boldsymbol{\Sigma}_{\text{prior}}$ to the first keyframe and yields mostly increasing uncertainties for subsequent keyframes. Only the latter approach reflects the growth of uncertainty due to lack of global position and yaw observations or constraints.

4.3 Auxiliary Estimator

The auxiliary estimator provides odometry for closed-loop control during time intervals when the primary optimization-based estimator's odometry is unavailable. The requirement of non-trivial translational motion in order to obtain a set of image frames with sufficient parallax and spatial displacement (Sect. 3.7) prevents aerial robots from successfully initializing monocular visual-inertial estimators before takeoff. To enable fully autonomous flight without relying on feedforward-only or RC-assisted takeoff, we develop an auxiliary state estimator that provides odometry during times such as takeoff or mid-air failure recovery when the primary estimator is still initializing.

The auxiliary state estimator uses an Unscented Kalman filter (UKF) that fuses IMU, downward camera horizontal velocity, and downward rangefinder observations to estimate velocity and height. Attitude and angular velocity are taken directly from the IMU, while horizontal position is integrated without correction from the UKF's estimated horizontal velocity. Note that horizontal translation and heading drift are not

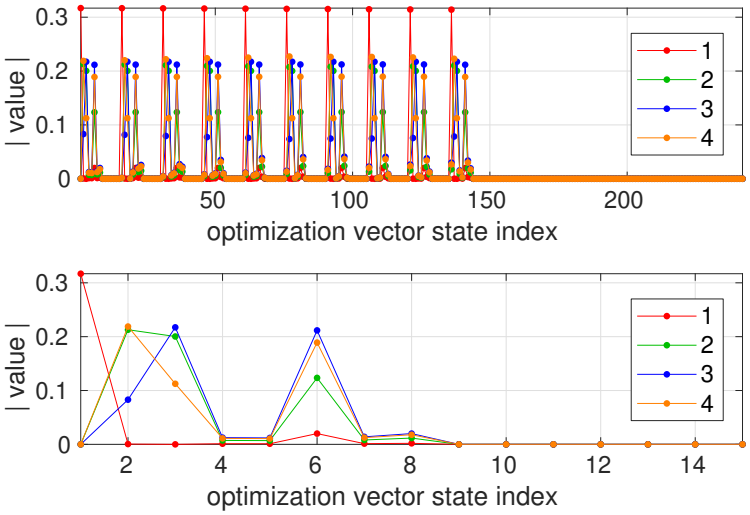


Figure 4.2: The null space basis vectors of the no-gauge-prior Hessian are each shown in a different color. The lower plot shows the first 15 components of the singular vectors corresponding to the four smallest singular values of the no-gauge-prior Hessian. The four basis vectors span position (components 1-3) and yaw (component 6) of all sliding window keyframes.

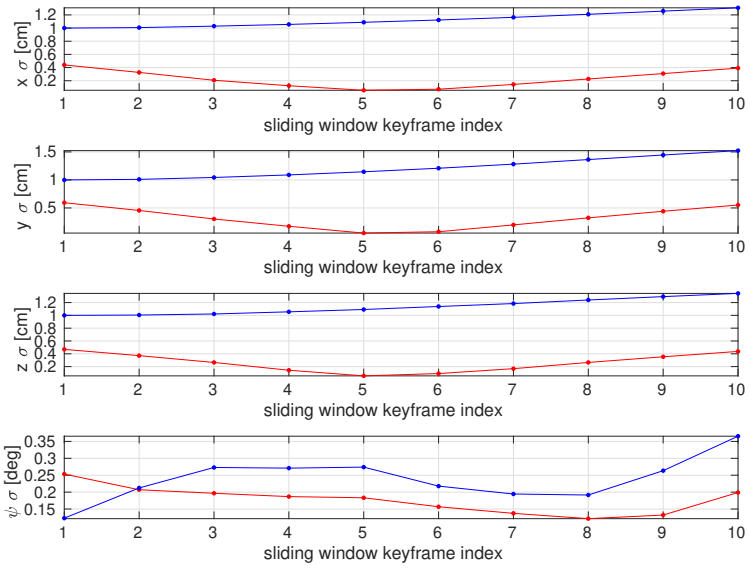


Figure 4.3: 1-σ uncertainty envelopes associated with the position and yaw states of sliding window keyframes. Red denotes covariance computed from taking the pseudo-inverse of the no-gauge-prior Hessian, while blue denotes covariance computed by inverting the with-gauge prior Hessian. The rank deficient Hessian spreads out uncertainty over the entire sliding window's keyframes, while the full rank Hessian correctly models the increase in uncertainty over the sliding window.

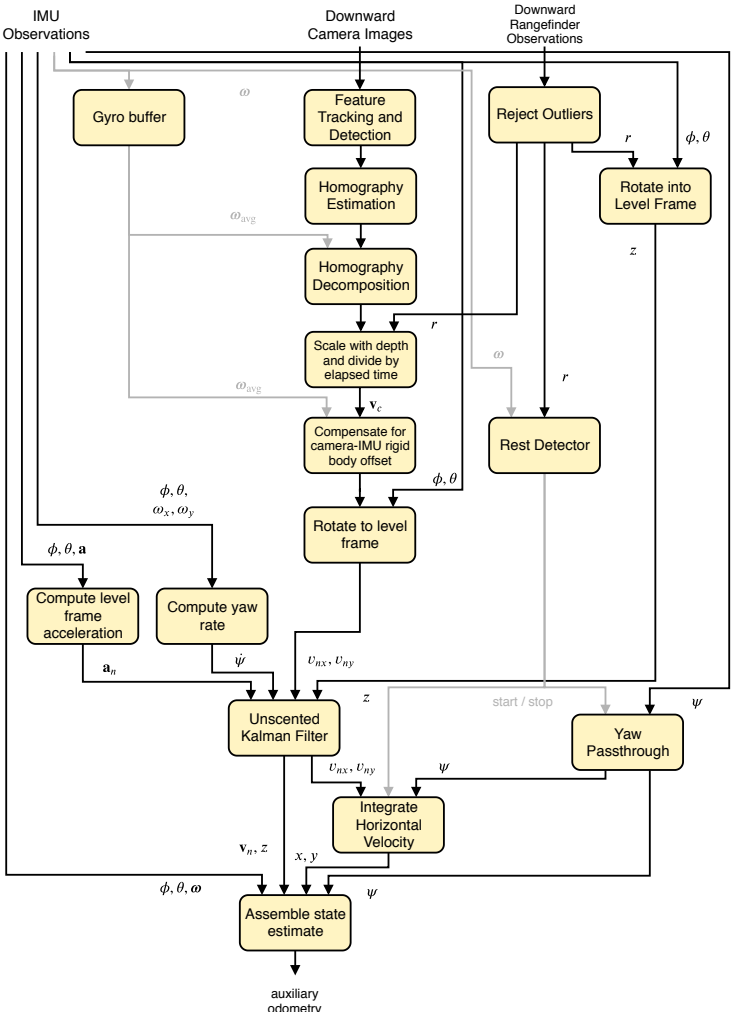


Figure 4.4: Auxiliary state estimator data flow

corrected because in typical flight scenarios the robot only uses auxiliary odometry for a short time before primary odometry is available.

4.3.1 Horizontal Velocity Estimation

In near-hover flight regimes above an approximately level surface, the robot's horizontal velocity can be estimated using images from a downward-facing camera, angular velocity from an IMU gyroscope, and range from a downward rangefinder. This section describes a horizontal velocity estimation method based on the correction model of [86] that utilizes the planar ground assumption, which is valid in most takeoff scenarios.

Given a set of feature correspondences between downward camera image frames k and $k+1$ that lie on a common plane, a point observation $\mathbf{p}_{k+1}$ in the later image frame is related to its corresponding point $\mathbf{p}_k$ in the earlier image frame via a homography $\mathbf{H}$.

$$\mathbf{p}_{k+1} = \mathbf{H}\mathbf{p}_k \tag{4.29}$$

$$\mathbf{H} = \mathbf{K}\left(\mathbf{R} + \mathbf{t}\cdot\mathbf{n}^{\mathrm{T}}\right)\mathbf{K}^{-1} \tag{4.30}$$

$\mathbf{K} \in \mathbb{R}^{3\times 3}$ is the pinhole camera intrinsic matrix, $\mathbf{n}$ is the unit normal of the plane, and $(\mathbf{R}, \mathbf{t})$ are the rotation and up-to-scale translation that take vectors from frame k to $k+1$. $\mathbf{H}$ is found in a robust manner via a RANSAC scheme that uses reprojection error as a measure of model quality.

The homography matrix is decomposed into four $(\mathbf{R}, \mathbf{t}, \mathbf{n})$ triplets of potential solutions. Two of the four potential solutions whose planes are behind the cameras (i.e. their plane normals satisfy $\mathbf{e}_3^{\mathrm{T}}\mathbf{n} < 0$) are rejected due to their physical impossibility. Of the remaining two solutions, the one whose relative rotation $\mathbf{R}$ is closest to the IMU gyro-derived relative rotation (3.2) is selected.

We assume that the downward rangefinder's measurement axis is parallel to the downward camera's z-axis and that the origins of the two frames are coincident. The first assumption is easy to ensure by mounting both sensors pointing downward underneath the body of an aerial robot. Although the second assumption is physically impossible to meet, in typical takeoff scenarios with a level ground plane and low roll and pitch angles, the downward rangefinder's observations will be quite close to the downward camera's true scene depth regardless of how far apart the two sensors are mounted.

Let r be the range observation from the downward rangefinder. The intersection point of the rangefinder's measurement axis with the ground plane is $\mathbf{r} = [0, 0, r]^{\mathrm{T}}$ in the camera frame. The dot product of $\mathbf{r}$ with the level ground plane's unit normal $\mathbf{n}$ yields rangefinder's altitude d, which is also the downward camera's scene depth.

$$d = \mathbf{r}^{\mathrm{T}}\mathbf{n} \tag{4.31}$$

Camera velocity is obtained by scaling $\mathbf{t}/\left\|\mathbf{t}\right\|$ (unit direction vector associated with the homography translation) by d and dividing by elapsed time.

$$\mathbf{v}_c = \frac{\mathbf{t}}{\left\|\mathbf{t}\right\|}\frac{d}{t_{k+1} - t_k} \tag{4.32}$$

The body frame velocity associated with $\mathbf{v}_c$ is

$$\mathbf{v}_b = \mathbf{R}_{bc}\left(\mathbf{v}_c + \mathbf{R}_{bc}^{\mathrm{T}}\boldsymbol{\omega}_b \times \mathbf{p}_{cb}\right) \tag{4.33}$$

where $\mathbf{p}_{cb}$ is the body frame's location in the camera frame, $\mathbf{R}_{bc}$ is the rotation that takes camera frame vectors to the body frame, and $\boldsymbol{\omega}_b$ is the body frame angular velocity.

Finally, the level frame horizontal velocity $\mathbf{v}_l$ is obtained by compensating for the roll and pitch of the current body frame:

$$\mathbf{v}_l = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\mathbf{v}_b \tag{4.34}$$

$$\mathbf{R}_{wb}\left(\phi,\theta,\psi\right)=\mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{4.35}$$

4.3.2 Altitude Estimation

The downward rangefinder provides time-of-flight range observations to the first surface in the direction it is pointing. Assume that the downward rangefinder is mounted at the body frame z-axis ($\mathbf{p}_{br}=\begin{bmatrix}0 & 0 & z_{\text{offset}}\end{bmatrix}^{\text{T}}$) and points along the body frame $-z$ axis. The body frame range observation is $\mathbf{r}_b=\begin{bmatrix}0 & 0 & -r\end{bmatrix}^{\text{T}}$, which can be transformed into the world frame using an attitude estimate derived solely from IMU measurements.

$$\mathbf{r}_w=\mathbf{R}_{\text{imu}}\mathbf{r}_b \tag{4.36}$$

The z-component of $\mathbf{r}_w$ is unaffected by yaw drift in the IMU attitude estimate because it only depends on the drift-free roll (ϕ) and pitch (θ) components of the IMU attitude estimate. The altitude observation used to update the UKF is

$$z_{\text{alt}}=r\cos\phi\cos\theta+z_{\text{offset}} \tag{4.37}$$

To suppress noise in raw range observations, apply a low pass filter them before using them in (4.37).

4.3.3 Unscented Kalman Filter

Theory

The Kalman filter is a recursive, online algorithm that estimates the values of unknown variables from sequences of noisy measurements. Let $\mathbf{x}$ be the concatenation of the unknown variables of interest. The Kalman filter recursively updates an estimate, $\hat{\mathbf{x}}$, of the true state vector together with the estimated covariance of the estimation error, $\mathbf{P}=\mathbb{E}[(\hat{\mathbf{x}}-\mathbf{x})(\hat{\mathbf{x}}-\mathbf{x})^{\text{T}}]$. In the Kalman filter's prediction step, a state transition model (usually derived from physical laws) is used to propagate the state estimate forward in time (with optional input from external observations). In the correction step, the state estimate is updated with external observations using an observation model. The influence of the prediction step observations and correction step observations on the state estimate is determined by their relative uncertainties. While the standard Kalman filter only supports linear prediction and correction models, the unscented Kalman filter [36] supports nonlinear models by using the unscented transform to approximate the propagation of Gaussian probability distributions through nonlinear functions with a fixed set of sample points.

Implementation

The auxiliary state estimator uses a discrete time UKF that fuses IMU, altimeter, and horizontal velocity observations to produce smooth altitude and velocity estimates (see Fig. 4.5). The state vector consists of the world frame altitude (z), navigation frame[3] velocity ($\mathbf{v}_n$), and IMU acceleration bias ($\mathbf{b}_a$):

$$\mathbf{x}=\begin{bmatrix}z\\ \mathbf{v}_n\\ \mathbf{b}_a\end{bmatrix}\in\mathbb{R}^7 \tag{4.38}$$

The initial states are all set to zero with the exception of altitude, which is set from the first downward altimeter observation. The initial covariance is set to a diagonal matrix comprised of user-defined sigma values:

$$\mathbf{P}_0=\text{diag}\left\{\sigma_{z0}^2,\sigma_{vx0}^2,\sigma_{vy0}^2,\sigma_{vz0}^2,\sigma_{bax0}^2,\sigma_{bay0}^2,\sigma_{baz0}^2\right\} \tag{4.39}$$

[3]The navigation frame is obtained by rotating the body frame's z-axis to align with the gravity vector while keeping heading unchanged.

The discrete time process update model is

$$\begin{bmatrix} z_{k+1} \\ \mathbf{v}_{n,k+1} \\ \mathbf{b}_{a,k+1} \end{bmatrix} = \begin{bmatrix} z_k \\ \mathbf{v}_{n,k} \\ \mathbf{b}_{a,k} \end{bmatrix} + \Delta t_k \begin{bmatrix} \mathbf{e}_3^{\mathrm{T}} \mathbf{v}_{n,k} \\ \mathbf{R}_y(\theta_m)\mathbf{R}_x(\phi_m)\left(\mathbf{a}_m - \mathbf{b}_{a,k} - \mathbf{n}_{a,k}\right) - g\mathbf{e}_3 - \dot{\psi}_m \mathbf{e}_3 \times \mathbf{v}_{n,k} \\ \mathbf{n}_{b,k} \end{bmatrix} \tag{4.40}$$

$$\dot{\psi}_m = \frac{1}{\cos\theta_m}\left(\omega_{m,y}\sin\phi_m + \omega_{m,z}\cos\phi_m\right) \tag{4.41}$$

where $\mathbf{e}_3 = \begin{bmatrix} 0 & 0 & 1 \end{bmatrix}^{\mathrm{T}}$. The process input observation vector consists of IMU roll, pitch, linear acceleration, and y and z angular velocities:

$$\mathbf{u} = \begin{bmatrix} \phi_m & \theta_m & a_{m,x} & a_{m,y} & a_{m,z} & \omega_{m,y} & \omega_{m,z} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^7 \tag{4.42}$$

The process noise vector consists of additive acceleration noise and acceleration bias derivative noise.

$$\mathbf{n} = \begin{bmatrix} \mathbf{n}_a \\ \mathbf{n}_b \end{bmatrix} \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{4.43}$$

The process noise covariance is a diagonal matrix that can used to adjust the state estimate's smoothness.

$$\mathbf{Q} = \mathrm{diag}\left\{\sigma_{ax}^2, \sigma_{ay}^2, \sigma_{az}^2, \sigma_{bx}^2, \sigma_{by}^2, \sigma_{bz}^2\right\} \tag{4.44}$$

The altitude correction update has the form

$$z_{\text{alt}} = z + n_{\text{alt}} \tag{4.45}$$

$$n_{\text{alt}} \sim \mathcal{N}(0, \sigma_{\text{alt}}^2) \tag{4.46}$$

where z_{alt} is defined in (4.37) and σ_{alt} is the empirical standard deviation of altitude observations.

The horizontal velocity correction update has the form

$$\mathbf{z}_{\text{vxy}} = \begin{bmatrix} v_{nx} \\ v_{ny} \end{bmatrix} + \mathbf{n}_{\text{vxy}} \tag{4.47}$$

$$\mathbf{n}_{\text{vxy}} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} s_{vx}^2 & 0 \\ 0 & s_{vy}^2 \end{bmatrix}\right) \tag{4.48}$$

where $\mathbf{z}_{\text{vxy}}$ is the final output of Sect. 4.3.1. The horizontal velocity observation's covariance at timestep k is assumed to be diagonal and computed empirically from running standard deviations of $\mathbf{z}_{\text{vxy}}$ observations over a window of size N.

$$\bar{v}_{nj,k} = \frac{1}{N}\sum_{i=1}^{N} v_{nj,k-N+i} \quad j \in \{x, y\} \tag{4.49}$$

$$s_{vj,k} = s_{vj,\text{base}} + s_{vj,\text{scale}}\sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(v_{nj,k-N+i} - \bar{v}_{nj,k}\right)} \quad j \in \{x, y\} \tag{4.50}$$

The parameters $s_{vx,\text{base}}$, $s_{vy,\text{base}}$, $s_{vx,\text{scale}}$, and $s_{vy,\text{scale}}$ are used to adjust the magnitude of the horizontal velocity observation's covariance.

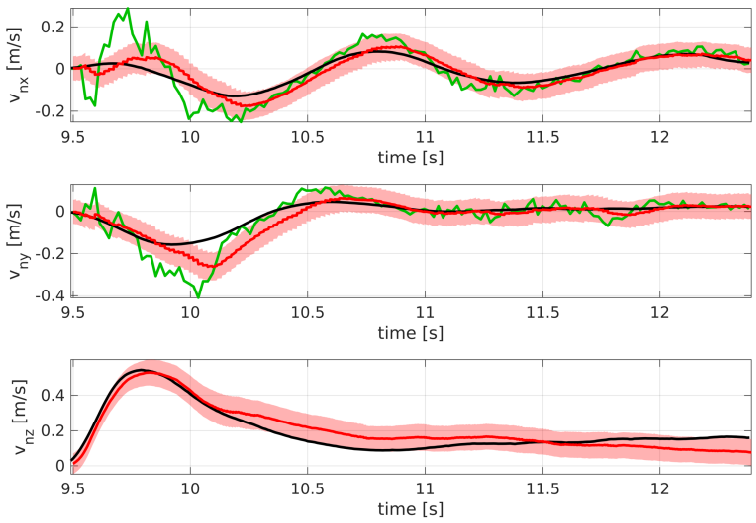


Figure 4.5: Auxiliary UKF velocity estimation. The UKF navigation frame velocity $\mathbf{v}_n$ is depicted as a red line, its associated 3 σ uncertainty envelope is in pink, the motion capture ground truth is in black, and the homography-derived horizontal velocity is in green.

4.3.4 Odometry Assembly

The auxiliary estimator odometry's vertical position and linear velocity components are set directly from the states of the UKF described in Sect. 4.3.3.

The auxiliary estimator's orientation is obtained directly from an IMU complementary attitude filter that fuses gyroscope angular velocities and accelerometer linear accelerations. Although it is possible to use magnetometer heading corrections in the IMU's complementary attitude filter, we avoid using the magnetometer because we need to operate in indoor environments with significant magnetic anomalies. Due to the lack of a heading correction, only the roll and pitch components of the complementary filter attitude estimate are drift-free. We also use the yaw component of the estimate because it does not incur significant drift over the typical lengths of time the auxiliary estimator needs to run for (i.e. when waiting for primary estimator initialization to finish) when the filter parameters are properly tuned. We set the auxiliary estimator's angular velocity to the IMU complementary attitude filter's bias-corrected gyroscope observations.

The horizontal position component of the auxiliary estimator odometry are obtained by rotating the horizontal navigation frame velocities from the UKF into the world frame and integrating the result with respect to time. By default, the world frame is set to the navigation frame at the time the UKF initializes. Although forward integration of velocities without correction leads to significant drift over time, we mitigate the total amount of horizontal position drift by only initializing the UKF upon takeoff see Sect. 4.4.1). This ensures that the auxiliary estimator runs for the shortest possible amount of time before the primary estimator finishes initialization.

4.4 Odometry Management

The odometry management system acts as a switch on primary and auxiliary odometry signals, outputting a single odometry signal that can be used for closed loop control. To reduce unnecessary computation, a finite state machine is used to start or stop the primary and auxiliary state estimators depending on flight conditions. 4DOF level transforms are applied to the current odometry source to ensure a smooth transition during odometry source switches. Figure 4.6 depicts the information flow between the primary estimator, auxiliary estimator, and odometry switcher.

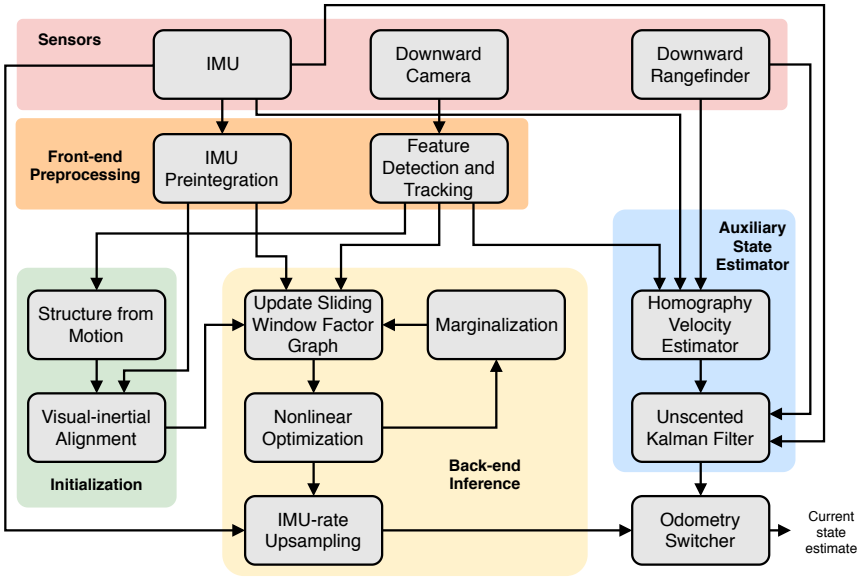


Figure 4.6: Information flow between primary estimator, auxiliary estimator, and odometry switcher

4.4.1 Finite State Machine

The finite state machine that manages starting, stopping, and switching between the primary and auxiliary estimators is depicted in Figure 4.7, while Table 4.1 describes the states in more detail.

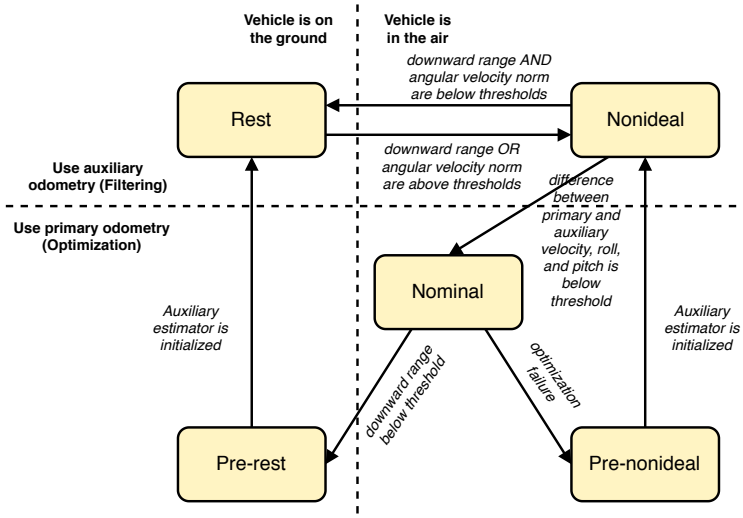


Figure 4.7: Finite state machine for switching between primary and auxiliary estimator

In a typical operating scenario, the aerial robot starts on the ground in the *rest* state. Although the auxiliary estimator's UKF initializes and runs, the auxiliary odometry's horizontal position and yaw states are clamped to avoid drift during the arbitrarily long time interval that the robot can remain on the ground. The primary estimator is disabled in *rest* state to avoid unnecessary computation.

Takeoff is detected when either the downward rangefinder range or the IMU's angular velocity norm exceeds predefined thresholds. After takeoff, the FSM enters the *non-ideal* state and enables the primary estimator. The primary estimator begins receiving keyframes and preintegrated IMU deltas only after takeoff

to improve initialization quality by maximizing the translational motion contained within its sliding window. Once the vehicle is in the air, the auxiliary estimator begins using IMU complementary filter yaw and integrating horizontal velocities to obtain horizontal position. Because both primary and auxiliary estimators are active, *non-ideal* state is the most computationally expensive FSM state. If the robot comes back to rest on the ground before the primary odometry is safe to use, the FSM re-enters the *rest* state. This transition only occurs when both the downward rangefinder range and the IMU angular velocity norm are below predefined thresholds.

The FSM transitions from *non-ideal* to *nominal* only when the primary odometry is safe to use, which is true when the following conditions are met:

- norm of the velocity difference between primary and auxiliary odometries falls belows a predefined threshold
- absolute difference of the primary and auxiliary roll estimates falls below a predefined threshold
- absolute difference of the primary and auxiliary pitch estimates falls below a predefined threshold

These conditions enable the user to control the magnitude of the velocity, roll, and pitch discontinuities in the final odometry sent to closed loop control that arise due to instantaneously switching from auxiliary to primary odometry. If the FSM remains in the *non-ideal* state for more than a threshold amount of time after a successful initialization, it is likely that the mismatch between primary and auxiliary odometries is caused by an inaccurate initialization. In such situations, initialization is restarted while remaining in the *non-ideal* state (Figure 4.8).

Table 4.1: Finite state machine state descriptions

FSM State	Description	Actions upon entry	Persistent Actions	Active estimator
Rest	Initial state, on the ground	Restart primary estimator	Clamp auxiliary odometry yaw and horizontal position	auxiliary
Non-ideal	Primary estimator is uninitialized or primary odometry is unsafe to use		Unclamp auxiliary odometry yaw and horizontal position	primary and auxiliary
Nominal	Primary estimator is initialized and primary odometry is ready to use			primary
Pre-rest	Using primary odometry after landing while waiting for auxiliary estimator to initialize	Trigger auxiliary estimator initialization		primary
Pre-nonideal	Using primary odometry after in-flight optimization failure while waiting for auxiliary estimator to initialize	Trigger auxiliary estimator initialization		primary

Upon entering the *nominal* state, the auxiliary estimator shuts down and closed loop control uses primary odometry. The FSM remains in the *nominal* state until the downward rangefinder range falls below a predefined threshold during landing, after which it transitions to *pre-rest*. *Pre-rest* is an intermediate state between *nominal* and *rest* whose only function is to wait for the auxiliary estimator's UKF to initialize. Upon entering *rest*, the primary estimator is stopped to prevent its sliding window from accumulating IMU observations into the second-latest preintegration motion delta as dictated by the two-way marginalization

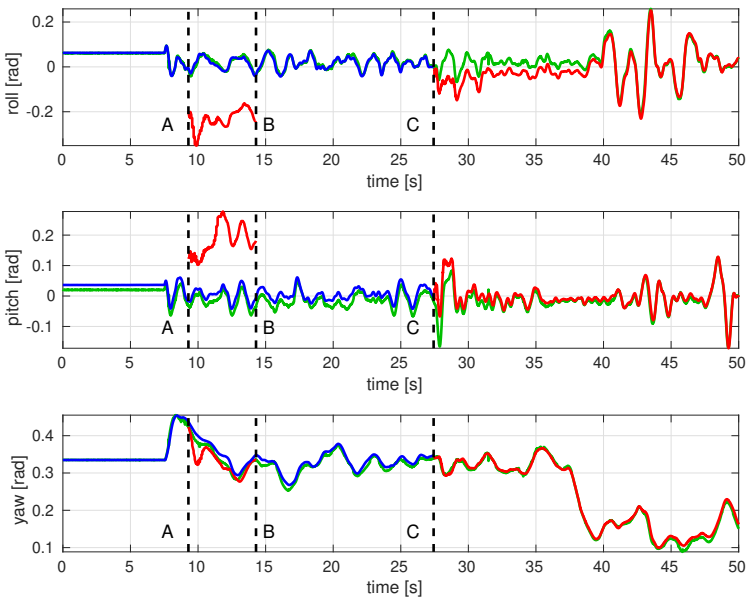


Figure 4.8: This plot compares the ZYX Euler angle estimates from the primary estimator (red), auxiliary estimator (blue), and mocap (green) during and after takeoff. Although the primary estimator initializes at point A, the odometry switcher continues using auxiliary estimator odometry because the roll and pitch differences between the two estimators is too large. After this condition persists for more than a threshold amount of time, the odometry switcher retriggers initialization at point B. At point C the primary estimator initializes for the second time with roll and pitch estimates that are sufficiently close to the auxiliary estimator. The odometry switcher stops the auxiliary estimator and begins using the primary estimator's odometry from this point forward.

strategy (Sect. 3.6.3). Because primary odometry obtained from such temporally imbalanced sliding windows tends to diverge (see Figure 4.9), auxiliary odometry is passed to the control system.

If any of the below optimization failure conditions are met while the FSM is in *nominal*, it transitions to *pre-nonideal*.

- Norm of difference between current optimization's last sliding window frame position and previous optimization's last sliding window frame position exceeds a threshold $d_{\max}$
- Rotation angle between current optimization's last sliding window orientation and previous optimization's last sliding window frame orientation exceeds a threshold $\varphi_{\max}$
- Norm of current optimization's acceleration bias estimate exceeds threshold $b_{a,\max}$
- Norm of current optimization's gyroscope bias estimate exceeds threshold $b_{\omega,\max}$
- Average value of reprojection residuals after applying loss function exceeds threshold $\lambda_{\text{proj},\max}$ on more than a threshold number of consecutive optimizations $n_{\text{proj},\max}$

Table 4.2 lists the threshold parameter values used in actual flight experiments. *pre-nonideal* is an intermediate state between *nominal* and *non-ideal* whose only function is to wait for the auxiliary estimator's UKF to initialize. After transitioning to *non-ideal*, the primary estimator begins reinitialization while the controller switches to using auxiliary odometry.

4.4.2 Smooth Odometry Source Switching

In order to avoid significant discontinuities in the final odometry, switching between primary and auxiliary odometry sources must ensure continuity in at least position and yaw.

Let $\{b\}$ be the body frame, $\{n\}$ be the navigation frame (the same as $\{b\}$ but with zero roll and pitch), $\{p_i\}$ be the primary odometry's frame on its i^{th} reinitialization, and $\{a_j\}$ be the auxiliary odometry's frame

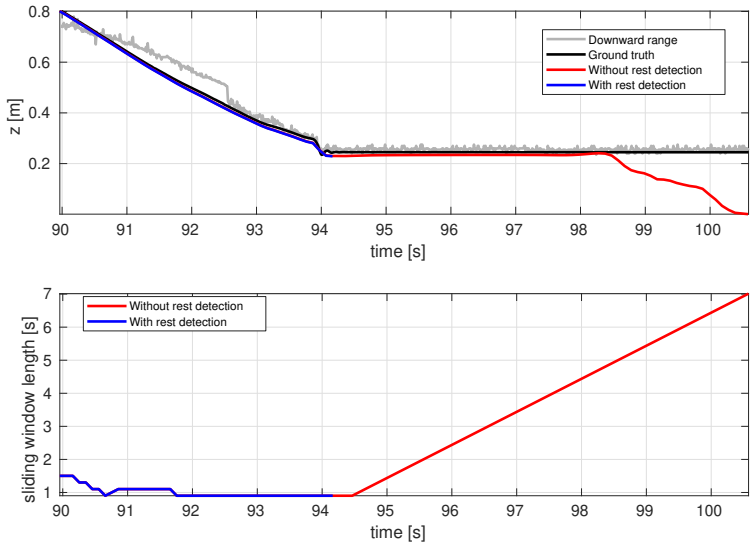


Figure 4.9: When the vehicle lands and comes to rest, the two-way marginalization strategy (without rest detection) repeatedly discards the second-newest sliding window keyframe. This causes the total time interval covered by sliding window keyframes to rise, resulting in temporally imbalanced sliding windows and estimator divergence (red line after t = 98 s in top plot). The FSM's rest detector stops the primary estimator after the vehicle comes to rest on the ground (blue line at t = 94 s) and avoids estimator divergence.

Table 4.2: Optimization failure threshold parameters

Threshold	**Value**
$d_{\max}$	5.0 m
$\varphi_{\max}$	0.873 rad
$b_{a,\max}$	2.5 m/s^2
$b_{\omega,\max}$	1.0 rad/s
$\lambda_{\text{proj},\max}$	2
$n_{\text{proj},\max}$	5

on its j^{th} reinitialization. We assume that both primary and auxiliary odometry are defined in level reference frames corresponding to the true position and yaw at the time of their most recent initializations.

The auxiliary estimator's odometry after its first initialization is $\mathbf{T}_{a_0 b}$, while the primary estimator's odometry after its first initialization is $\mathbf{T}_{p_0 b}$. At the moment of switchover, create level transforms $\mathbf{T}_{a_0 n}$ and $\mathbf{T}_{p_0 n}$ by zeroing out the auxiliary and primary odometry's roll and pitch. To ensure a smooth switchover, pre-multiply the primary odometry pose with the level transform $\mathbf{T}_{a_0 p_0} = \mathbf{T}_{a_0 n}\mathbf{T}_{p_0 n}^{-1}$ to align its position and yaw with the auxiliary odometry pose. Additionally, rotate the primary odometry's linear velocity by the yaw associated with $\mathbf{T}_{a_0 p_0}$. This process is performed with the roles of the auxiliary and primary odometries reversed when switching from primary to auxiliary odometry.

The proposed odometry alignment method does not ensure that velocity, roll, pitch, or angular velocity states are aligned when switching between primary and auxiliary odometry. However, angular velocity usually does not differ significantly between the different odometry sources because it is a body frame quantity that closely follows gyroscope observations. Discontinuities in roll, pitch, and velocity are mitigated by only switching between odometry sources when roll, pitch, and velocity estimates are sufficiently close.

Chapter 5

VIO for Multiple Cameras with Disjoint Fields of View

Although a single camera and IMU form a minimal sensor suite for visual-inertial odometry, achieving good performance depends on good template matching or feature tracking, which require the viewed scene to have adequate lighting, no moving objects, and no motion blur. A common method to obtain more visual information is to use a stereo configuration where two hardware-synchronized cameras observe the same spatial volume. Assuming the baseline between the two cameras is known, stereo configurations render scene depth observable in the absence of sensor motion. However, stereo visual-inertial odometry methods still suffer if the visible scene has poor lighting, moving objects, or non-trivial motion blur.

We propose to use multiple cameras with disjoint fields of view to improve the accuracy and robustness of visual-inertial state estimation by increasing the probability that image observations from at least one of the cameras contain sufficient visual information for motion estimation. In many flight scenarios, the benefit of additional visual information is greater than the benefit of vision-only scale recovery.

5.1 Multi-Camera Optimization Problem Formulation

We first extend the single camera optimization problem formulation from Sect. 3.5.1 to multiple synchronized cameras (Sect. 5.1.1) and then present modifications that are needed to support asynchronous cameras (Sect. 5.1.2).

5.1.1 Synchronized Cameras

This section adapts the single camera optimization problem formulation in Sect. 3.5.1 to the case where there are multiple synchronized cameras. Because corresponding image frames from multiple cameras are taken at the same instance in time, corresponding keyframes from multiple cameras are exactly matched in time. In this setup, a single set of IMU motion states corresponding to all sliding window keyframes, augmented with camera-IMU extrinsic transforms to each camera, is sufficient to fully characterize reprojection geometry relationships for all cameras.

Let there be p synchronized cameras with feature observations in the sliding window. Suppose there are m_c feature trails from the c^{th} camera that have been observed in at least two consecutive keyframes, starting on or before a threshold number of keyframes before the sliding window's last keyframe. The location of feature l of camera c in the world frame is parametrized by its inverse depth (ρ_{cl}) in camera c's frame at the pose corresponding to the first sliding window keyframe in which it was observed.

We concatenate motion, IMU bias, and inverse depths of each camera's features to form the full state vector $\mathbf{x}$.

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1^{\mathrm{T}} & \dots & \mathbf{x}_n^{\mathrm{T}} & \rho_{11} & \dots & \rho_{1m_1} & \dots & \rho_{p1} & \dots & \rho_{pm_p} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{16n+\sum_{c=1}^{p} m_c} \tag{5.1}$$

This is the multi-camera equivalent of (3.166). The multi-camera unconstrained optimization problem has the form:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \left[f_{\text{prior}}\left(\mathbf{x}\right) + f_{\text{imu}}\left(\mathbf{x}\right) + \sum_{c=1}^{p} f_{\text{cam},c}\left(\mathbf{x}\right) \right] \tag{5.2}$$

This is the multi-camera equivalent of (3.167).

The camera cost encodes feature-derived geometric constraints on sliding window poses and inverse depths.

$$f_{\text{cam},c}\left(\mathbf{x}\right) = \frac{1}{2} \sum_{l=1}^{m_c} \sum_{i=2}^{|\mathcal{K}_{cl}|} \lambda \left(\left\| \mathbf{r}_{\text{proj}} \left(\mathbf{p}_{k(c,l,1)}, \mathbf{q}_{k(c,l,1)}, \mathbf{p}_{k(c,l,i)}, \mathbf{q}_{k(c,l,i)}, \rho_{cl} \right) \right\|_{\mathbf{\Sigma}}^2 \right) \tag{5.3}$$

$\mathcal{K}_{cl}$ is the set of keyframe indices where feature l from camera c is observed, $k(c,l,i)$ is a function that returns the keyframe index of the i^{th} observation of feature l from camera c, $\lambda(\cdot)$ is a loss function, and $\mathbf{\Sigma} \in \mathbb{R}^{2\times 2}$ is the feature observation noise covariance matrix. This is the multi-camera equivalent of (3.170).

5.1.2 Asynchronous Cameras

Although a synchronized camera rig enables a straightforward extension of single camera optimization to the multi-camera case, asynchronous camera rigs have two advantages:

- Hardware synchronization of multi-camera rigs incurs additional engineering effort and complexity.
- Asynchronous triggering is often required to enable different auto-exposure settings (i.e. different image frequencies) across multiple cameras that view scenes with different lighting conditions.

Problem

Without specialized hardware to synchronize the triggering of multiple cameras, image observations from each camera are taken at different times. Consequently, image frames and keyframes from different cameras occur at different points in time. A straightforward way to incorporate observations from multiple cameras is to include pose states for each camera in the optimization vector. However, this leads to a number of consequences:

1. The optimization vector grows by *number of keyframes in sliding window* $\times$ *number of motion states per keyframe* for each additional camera whose observations are incorporated.
2. The number of IMU factors between keyframe motion states increases by approximately *number of keyframes in sliding window*, since the action of inserting a keyframe into the sliding window splits a single IMU factor into two IMU factors unless inserting at the beginning or end of the sliding window.
3. Although keyframes from different cameras are not synchronized, it is possible for some keyframes from different cameras to be very close together in time by chance. The IMU factor between such a pair of keyframes is associated with a preintegration motion delta formed from very few IMU observations, which is undesirable because the effect of IMU noise will be passed through to the preintegration motion delta without the averaging effect of a large number of observations.

In the remainder of this section we describe three methods for formulating a multi-camera optimization problem that avoids these consequences.

Pose Interpolation

The pose interpolation strategy only includes motion states associated with a single camera's keyframes in the optimization vector and linearly interpolates pose at the times corresponding to all other camera keyframes [87]. The camera whose keyframes are associated with IMU motion states in the optimization vector is designated as the *master* camera, while all other cameras are designated as *secondary* cameras.

Ideally, all secondary camera timestamps fall within or not too far outside of the time interval formed by the first and last master camera keyframe observations. The key assumption of the pose interpolation method is that the rigid body on which all cameras are mounted moves with constant linear and angular velocity between any two successive master camera keyframes.

The optimization problem formulation for the pose interpolation strategy is largely the same as synchronized camera formulation in Sect. 5.1.1 with the exception that reprojection factors for secondary camera features are replaced with interpolated pose reprojection factors. Appendix F formulates the interpolated pose reprojection factor and provides detailed derivations of its residual and Jacobians. The interpolated reprojection factor is more expensive to evaluate than the standard reprojection factor because it involves up to two extra pose states. This increases the connectedness of the sliding window bundle adjustment factor graph, which increases the fill-in of the Jacobian matrix and therefore increases the cost of iterative optimization. Another disadvantage of the pose interpolation strategy is that the assumption of constant linear and angular velocity over typical inter-keyframe time intervals is often violated. Errors in reprojection geometry caused by the incorrect constant velocity assumption have a significant impact on depth estimation and can quickly cause iterative nonlinear solvers to converge to incorrect solutions.

Image Plane Feature Trail Interpolation

See Section 5.2.

Continuous Trajectory Estimation

The continuous trajectory estimation approach reformulates the sliding window bundle adjustment optimization problem by parameterizing motion as linear combinations of splines instead of discrete sets of poses and twists. In [88], the optimization vector's position, velocity, orientation, and IMU bias states at a fixed set of keyframe times are replaced by a set of B-spline coefficients. The main benefit of a continuous trajectory representation is that poses can be evaluated at arbitrary points in time, which enables modeling secondary camera reprojection residuals in an asynchronous multi-camera rig.

5.2 Image Plane Feature Trail Interpolation

In a similar manner as the pose interpolation strategy, the image plane feature trail interpolation strategy only includes motion states associated with a single camera's keyframes in the optimization vector. Instead of interpolating IMU pose at secondary camera keyframe times with respect to IMU pose at temporally proximal master camera keyframes, the image plane feature trail interpolation strategy interpolates secondary camera feature locations at master camera keyframe times with respect to their locations in temporally proximal secondary camera frames. A minimum of two frames are required for linear interpolation, while three frames are required for quadratic interpolation. Each secondary camera feature observation's interpolant is evaluated at the time of the closest master camera keyframe to obtain an estimate of its location as captured by a hypothetical secondary camera that is time synchronized with the master camera. This pseudo-observation enables the synchronized multi-camera optimization formulation (Sect. 5.1.1) to be used without modification.

Unlike in the pose interpolation method, secondary camera reprojection geometry can be represented by standard reprojection factors (Sect. 3.4), avoiding the need to use the more costly interpolated reprojection factors (Appendix F) that not only involve two extra pose states but also require evaluation of the capitalized exponential map and capitalized logarithmic map. When using standard reprojection factors for secondary cameras, feature observations $\mathbf{h}_{c,l,i}$ and $\mathbf{h}_{c,l,j}$ in (3.141)-(3.142) are approximated by either linear (5.4) or quadratic (5.5) interpolation. Table 5.1 compares the differences of the pose interpolation and image plane feature trail interpolation methods.

Table 5.1: Comparison of pose interpolation and image plane feature trail interpolation

	Pose Interpolation	**Image Plane Feature Trail Interpolation**
Interpolation space	$SE(3)$ or $SO(3) \times \mathbb{R}^3$	$\mathbb{R}^2$
Time grid points used for interpolation correspond to	closest *master camera keyframes* before and after the considered secondary camera keyframe	closest *secondary camera frames* before and after the considered secondary camera keyframe
Typical separation of neighboring points on time grid used for interpolation	~ 0.1 s	~ 0.016 s
Interpolant is evaluated at	secondary camera keyframe time	master camera keyframe time
Assumption	Camera moves with constant linear and angular velocity over interpolation time interval	Feature location moves with constant velocity (or acceleration) on the image plane over interpolation time interval

5.2.1 Lagrange Polynomial Interpolation of Feature Observations

Let τ_{j-1}, τ_j, and τ_{j+1} be three successive secondary camera image frame timestamps. Let t_i be a master camera keyframe that satisfies $\tau_j < t_i < \tau_{j+1}$. Let $\mathbf{h}_{c,l}(t)$ be the homogeneous coordinate representation of feature l in camera c at time t. Feature tracking yields observations $\mathbf{h}_{c,l}(\tau_{j-1})$, $\mathbf{h}_{c,l}(\tau_j)$, and $\mathbf{h}_{c,l}(\tau_{j+1})$. The image plane feature trail interpolation strategy estimates $\mathbf{h}_{c,l}(t_i)$ from these observations and their corresponding timestamps via componentwise Lagrange polynomial interpolation.

Linear interpolation only requires observations at τ_j and τ_{j+1}.

$$\mathbf{h}_{c,l,\text{lin}}(t_i) = \frac{t_i - \tau_{j+1}}{\tau_j - \tau_{j+1}} \mathbf{h}_{c,l}(\tau_j) + \frac{t_i - \tau_j}{\tau_{j+1} - \tau_j} \mathbf{h}_{c,l}(\tau_{j+1}) \tag{5.4}$$

Quadratic interpolation requires observations at τ_{j-1}, τ_j, and τ_{j+1}.

$$\begin{aligned} \mathbf{h}_{c,l,\text{quad}}(t_i) = &\frac{t_i - \tau_j}{\tau_{j-1} - \tau_j} \frac{t_i - \tau_{j+1}}{\tau_{j-1} - \tau_{j+1}} \mathbf{h}_{c,l}(\tau_{j-1}) + \frac{t_i - \tau_{j-1}}{\tau_j - \tau_{j-1}} \frac{t_i - \tau_{j+1}}{\tau_j - \tau_{j+1}} \mathbf{h}_{c,l}(\tau_j) \\ &+ \frac{t_i - \tau_{j-1}}{\tau_{j+1} - \tau_{j-1}} \frac{t_i - \tau_j}{\tau_{j+1} - \tau_j} \mathbf{h}_{c,l}(\tau_{j+1}) \end{aligned} \tag{5.5}$$

Linear interpolation models the observation of feature l as moving with constant velocity on the image plane, while quadratic interpolation models it as moving with constant acceleration on the image plane. These assumptions are valid over typical interpolation time intervals ($\tau_{j+1} - \tau_j$ or $\tau_{j+1} - \tau_{j-1}$) on the order of tens of milliseconds unless the multi-camera rig undergoes highly dynamic motions.

5.2.2 Simulation Comparison of Pose Interpolation and Image Plane Feature Trail Interpolation

In this section we present a small toy example to illustrate the differences between the pose interpolation and image plane feature trail interpolation strategies. A simulated camera moves over and observes a set of known features in 3D space. The camera-body extrinsic transform is

$$\mathbf{p}_{bc} = \begin{bmatrix} 0.0615 \\ 0.015 \\ -0.093 \end{bmatrix} \text{ m} \tag{5.6}$$

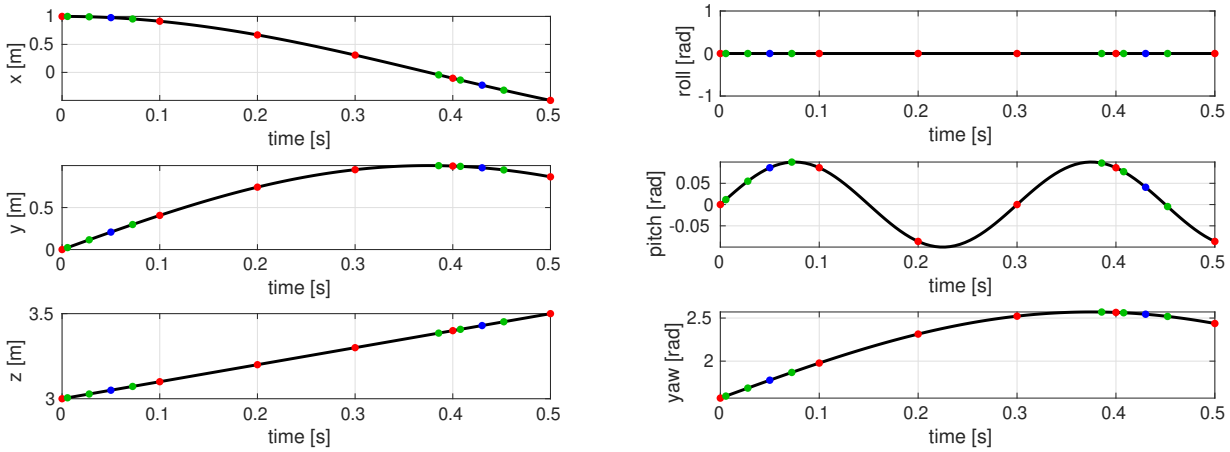
$$\phi_{bc} = 3.064 \text{ rad} \tag{5.7}$$
$$\theta_{bc} = 0.016 \text{ rad} \tag{5.8}$$
$$\psi_{bc} = -1.611 \text{ rad} \tag{5.9}$$

The body frame's position and attitude with respect to the world frame are given by

$$\mathbf{p}_{wb}(t) = \begin{bmatrix} \cos\left(\frac{2\pi t}{1.5}\right) \\ \sin\left(\frac{2\pi t}{1.5}\right) \\ t+3 \end{bmatrix} \text{ m} \tag{5.10}$$
$$\phi_{wb}(t) = 0 \text{ rad} \tag{5.11}$$
$$\theta_{wb}(t) = 0.1 \sin\left(\frac{2\pi t}{0.3}\right) \text{ rad} \tag{5.12}$$
$$\psi_{wb}(t) = \frac{\pi}{2} + \sin\left(\frac{2\pi t}{1.5}\right) \text{ rad} \tag{5.13}$$

over the trajectory time interval of $0 \leq t \leq 5$ s (Fig. 5.2). The camera's frame rate is 45 Hz and the set of camera keyframe times is $\{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$ s. We simulate 49 features in a regular horizontal grid below the trajectory (Fig. 5.2a). Feature heights are normally distributed around $z = 0.5$ m with a standard deviation of 0.3 m.

The objective of this toy example is to compare the pose interpolation, linear image plane feature trail interpolation, and quadratic image plane feature trail interpolation strategies for predicting feature locations at non-keyframe evaluation times $t = 0.05$ s and $t = 0.43$ s. While the pose interpolation strategy is restricted to using information at *keyframe* times, the image plane feature trail interpolation strategies have access to information from grids of two or three *frames* around the evaluation times. Linear interpolation only uses the two frames closest to each evaluation time, while quadratic interpolation uses all three frames corresponding to each evaluation time.

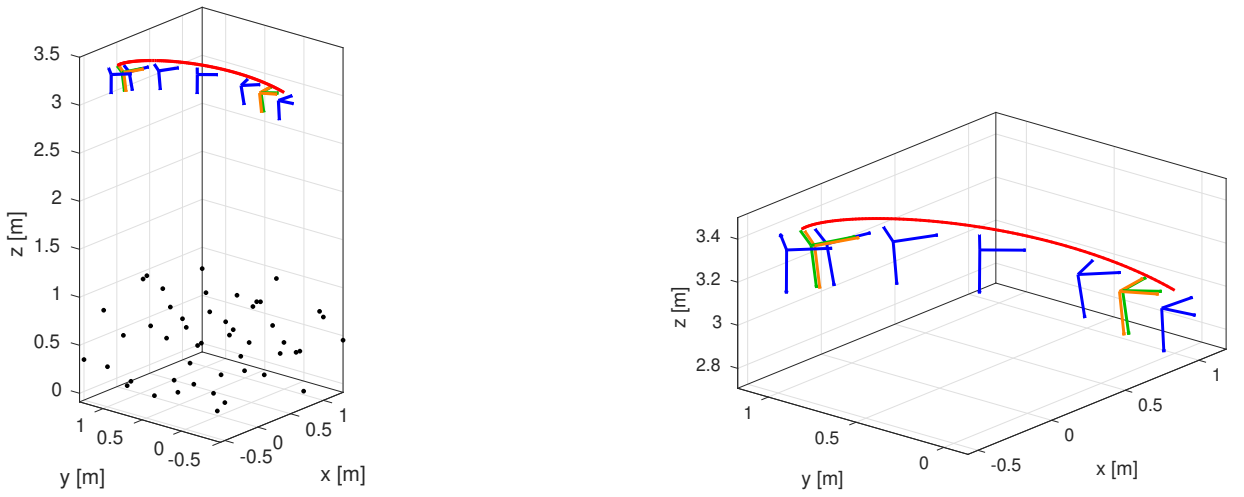


(a) Position vs. time, (5.10)

(b) Attitude vs. time, (5.11)-(5.13)

Figure 5.1: Simulated trajectory: ground truth (black), keyframe times (red), image plane feature trail interpolation grid times (green), evaluation times (blue)

The pose interpolation strategy uses the poses of the two keyframes closest to each evaluation time to interpolate pose at each evaluation time via (F.5), (F.7)-(F.8). The non-trivial difference between the ground truth and interpolated poses in Figure 5.2b indicates that the assumption of constant linear and angular velocity between successive keyframes is violated for the relatively simple trajectory in this toy example.



(a) 3D trajectory and feature locations

(b) Close-up of 3D trajectory

Figure 5.2: Simulated trajectory in 3D: ground truth path (red), features (black), ground truth poses at keyframe times (blue), ground truth poses at evaluation times (green), interpolated poses at evaluation times (orange)

Figure 5.3 compares reprojected features from each interpolation method with the ground truth feature observations at the evaluation times. The pose interpolation method predicts feature observations by triangulating 3D feature locations from the two surrounding keyframes and projecting the result into the camera frame associated with the interpolated pose at the evaluation time. The image plane feature trail interpolation methods predict feature observations by linear or quadratic interpolation of corresponding feature observations in a grid of frames clustered around the evaluation time. The inset figure shows that the reprojection error with respect to ground truth is the greatest for the pose interpolation method and the lowest for the quadratic image plane feature trail interpolation method. This trend is more clearly illustrated by the inverse cumulative distribution plots of Figure 5.4. Both figures additionally suggest that the benefit of using quadratic over linear image plane feature trail interpolation is smaller than the benefit of using linear image plane feature trail interpolation over pose interpolation.

Figure 5.5 depicts the feature trails predicted by each method over time intervals around each evaluation time. The time interval around $t = 0.05$ s corresponds to a large rotation about the camera axis that is associated with high optical flow. Consequently, the contrast between the feature trails predicted by different interpolation strategies is significant. On the other hand, the time interval around $t = 0.43$ s corresponds to lower optical flow and yields a lesser difference between the feature trails predicted by different interpolation strategies.

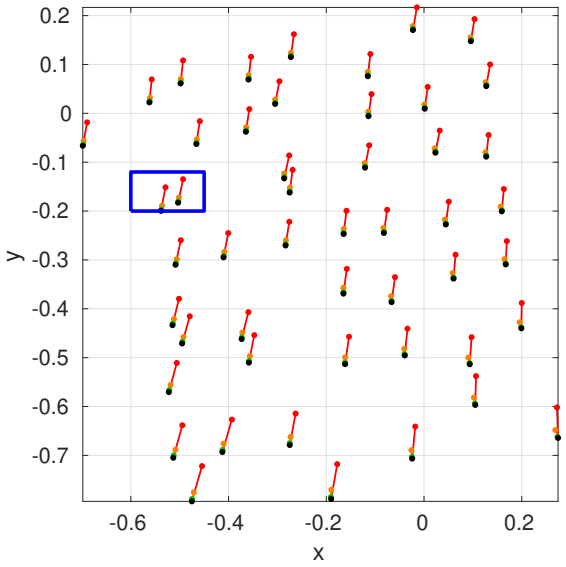
This toy example demonstrated that the image plane interpolation method yields more accurate feature predictions than the pose interpolation method under a typical operating scenario. The two main factors that account for the former method's superior performance are greater validity of its assumptions and the greater temporal proximity of its interpolation grids to their corresponding evaluation points.

5.3 Modifications to Single Camera VIO

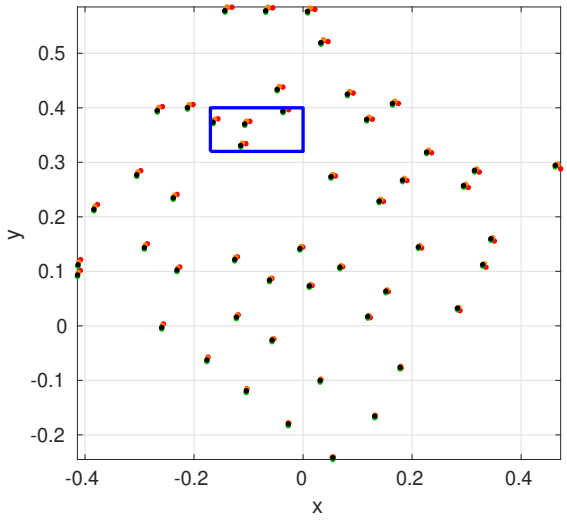
This section describes aspects of the single camera visual-inertial odometry algorithm that need to be modified for compatibility with multiple cameras.

5.3.1 Selection of Keyframes from Streaming Image Frames

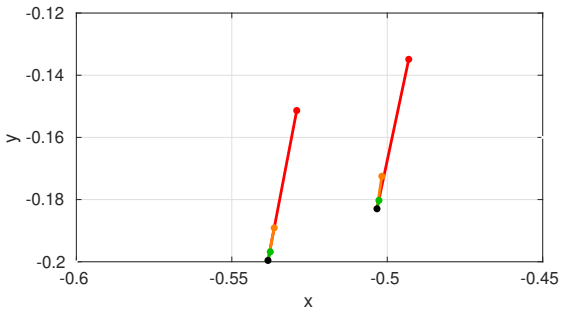
Figure 5.6 depicts the relationship of secondary camera image frames and interpolated pseudo-keyframes with master camera image frames and keyframes. The secondary camera pseudo-keyframe is interpolated



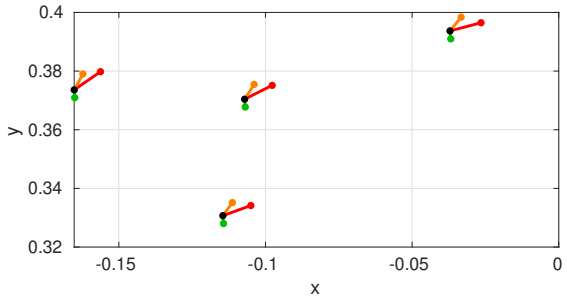
(a) Reprojected features at $t = 0.05$ s



(b) Reprojected features at $t = 0.43$ s

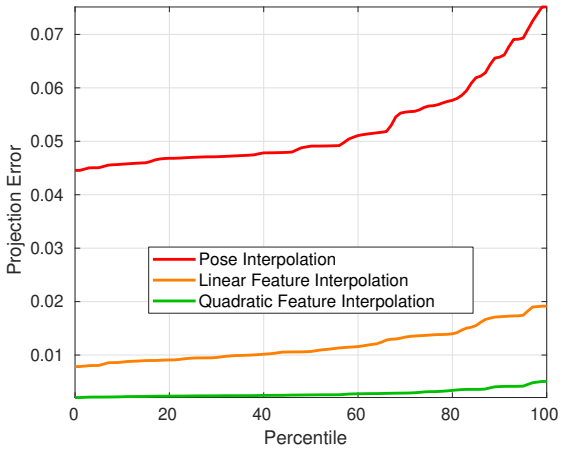


(c) Close-up of blue inset from (a)

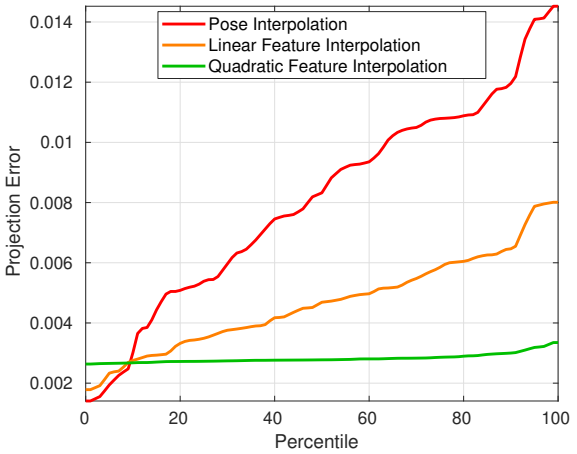


(d) Close-up of blue inset from (b)

Figure 5.3: Comparison of feature reprojections computed via the pose interpolation (red), linear image plane feature trail interpolation (orange), and quadratic image plane feature trail interpolation (green) methods with the ground truth feature observation (black) at the evaluation times.

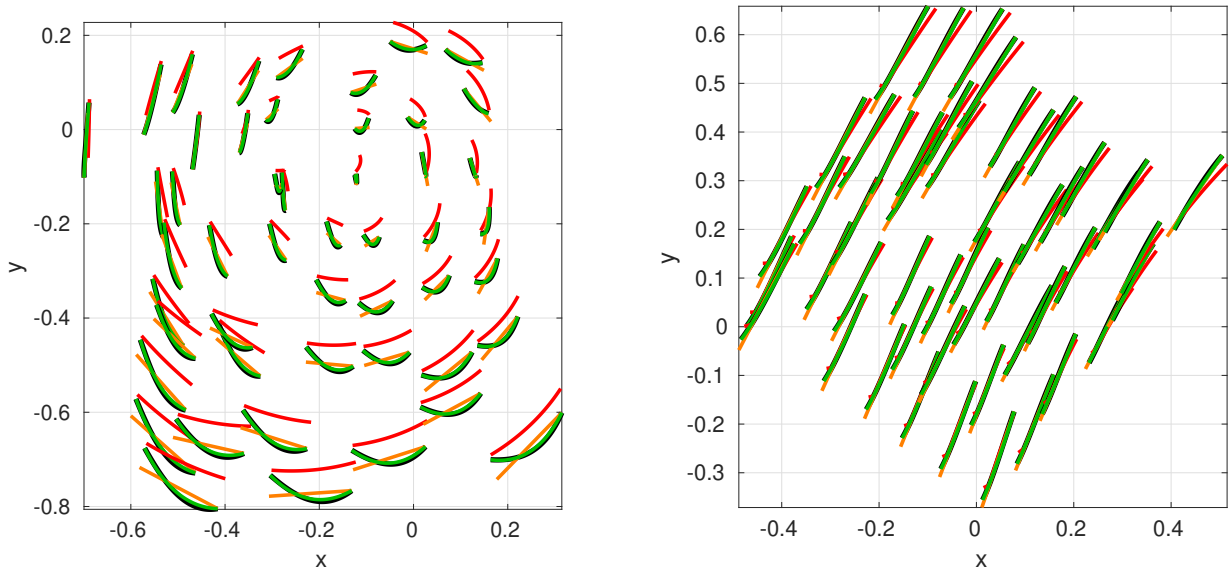


(a) $t = 0.05$ s



(b) $t = 0.43$ s

Figure 5.4: Inverse cumulative distribution plots for projection error at each evaluation time



(a) 0.02 s $< t <$ 0.07 s (b) 0.39 s $< t <$ 0.45 s

Figure 5.5: Comparison of ground truth feature trails (black) with corresponding feature trails predicted by pose interpolation (red), linear image plane feature trail interpolation (orange), and quadratic image plane feature trail interpolation (green) over time intervals around each evaluation time.

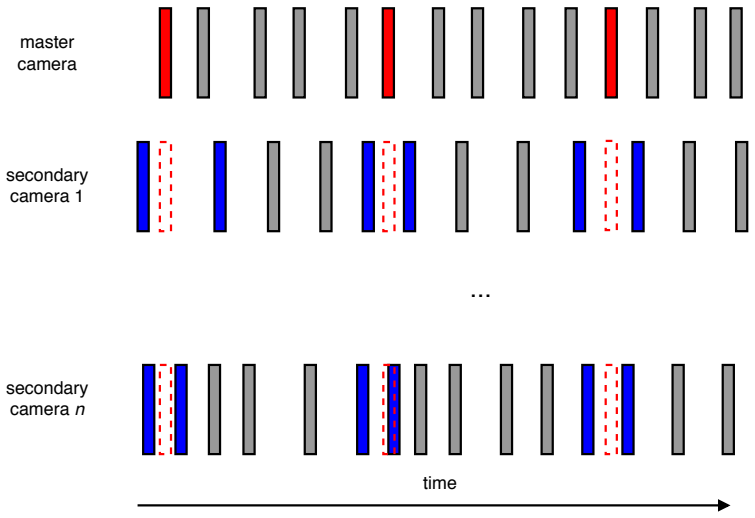


Figure 5.6: Each master camera keyframe (solid red rectangle) is associated with the two secondary camera frames that bracket it (blue rectangles). A secondary camera pseudo-keyframe observation (red dotted border rectangle) at the same time as the corresponding master camera keyframe is interpolated from neighboring secondary camera frames (blue rectangles). Note that three neighboring secondary camera frames, rather than the two depicted in this figure, are required for quadratic interpolation.

from the set of two or three secondary camera frames whose compensated timestamps are closest to the master camera keyframe's compensated timestamp. This set is chosen such that the second last and last frames' compensated times bracket the latest master camera keyframe's compensated time. Each secondary camera's feature tracking front-end sends a batch of frames associated with the master camera keyframe's compensated timestamp to the optimization back-end in order to provide the latter with raw observations for feature trail interpolation.

5.3.2 Optimization

The sliding window optimization back-end only utilizes secondary camera observations that have arrived before the latest master camera keyframe, which enables optimization frequency to solely depend on master camera keyframe rate (as opposed to having to wait for secondary camera observations to arrive). This arrangement enables the sliding window optimization to naturally handle the case where no secondary camera observations are available.

Interpolated feature observations must be used when triangulating feature depths using sliding window poses that correspond to the times of master camera keyframes. These triangulated depths are used as initial guesses when incorporating new feature trails into the optimization problem.

5.3.3 Incorporation of New Feature Trails into Optimization

The criteria for adding new feature trails to the optimization problem described in Sect. 3.5.2 applies to secondary cameras as well as the master camera. When computation is limited and the optimization size must be kept constant, a single feature budget must be enforced across all cameras when incorporating new feature trails into the optimization problem. A default strategy is to evenly split the feature budget across all cameras and enforce the per-camera feature budget when selecting new feature trails from an individual camera to add to the optimization problem. We will call this the *even split* strategy to distinguish it from a more sophisticated information gain-based approach for selecting new features to include in optimization that will be described in Sect. 6.

5.3.4 Initialization

The full initialization procedure described in Sect. 3.7 is only performed for the master camera. Assuming that each master camera keyframe is associated with a batch of frames from each secondary camera, after a successful master camera initialization the poses at master camera keyframe compensated timestamps are used to triangulate depths of interpolated secondary camera features.

5.3.5 Selection of Keyframes for Marginalization

The marginalization strategy described in Sect. 3.6.3 is applied to the master camera's keyframes in order to prevent secondary camera visual information from influencing keyframe selection. This design decision was made to simplify implementation changes with respect to the single camera case. Adapting the marginalization strategy to consider parallax in multiple cameras is an avenue for potential future work.

5.4 Experimental Results

This section presents postprocessed results that demonstrate the superiority of multiple camera sliding window visual-inertial odometry over single camera sliding window visual-inertial odometry on in-flight datasets taken in environments with different visual textures and lighting conditions. To achieve this, images and IMU observations collected during each trajectory were used to run sliding window visual-inertial odometry in three settings - with the downward camera as the only source of visual observations, with the forward camera as the only source of visual observations, and with both cameras supplying visual observations.

5.4.1 Motion Capture Arena

In this dataset the aerial robot flies along a horizontal square trajectory (Fig. 5.7) consisting of four short segments separated by aggressive stopping maneuvers. Because this flight occurs inside a motion capture arena (Fig. G.4), full ground truth is available.

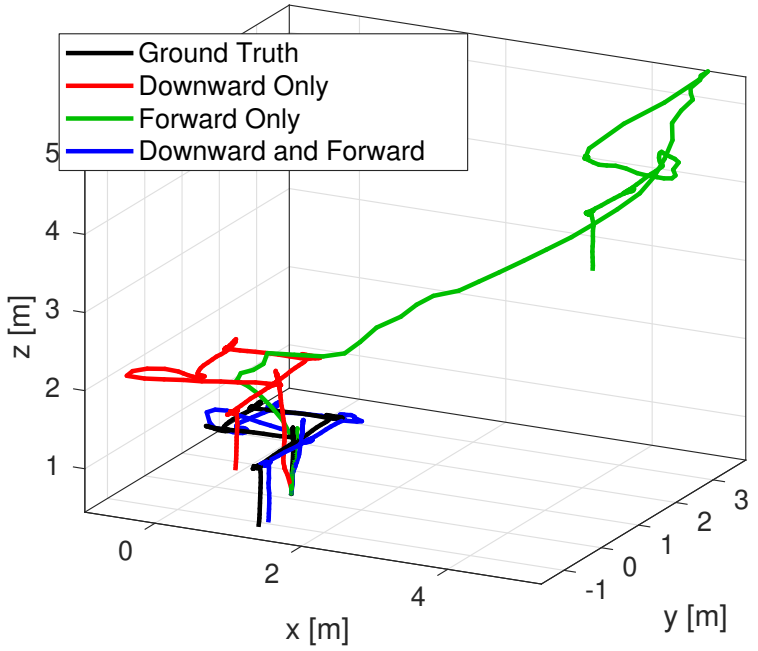


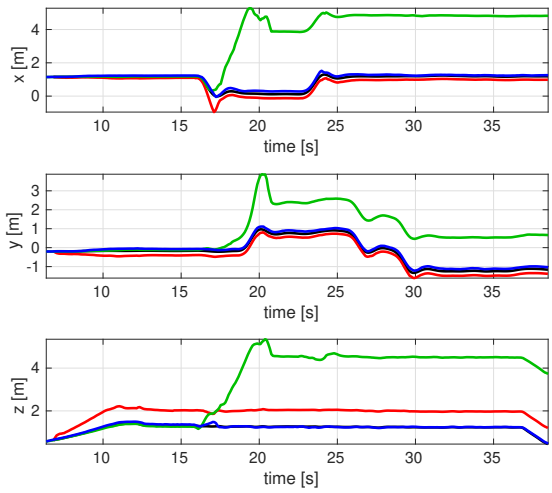
Figure 5.7: 3D plots of the trajectories estimated by VIO with different camera combinations on the motion capture dataset

Figures 5.8-5.9 show that forward camera only estimation experiences a sharp increase in position and velocity error after $t = 15$ s. Around this time downward camera only estimation experiences a smaller increase in translational error, while two camera estimation experiences the smallest increase in translational error. However, two camera estimation incurs the greatest heading error after $t = 15$ s (Fig. 5.10b).

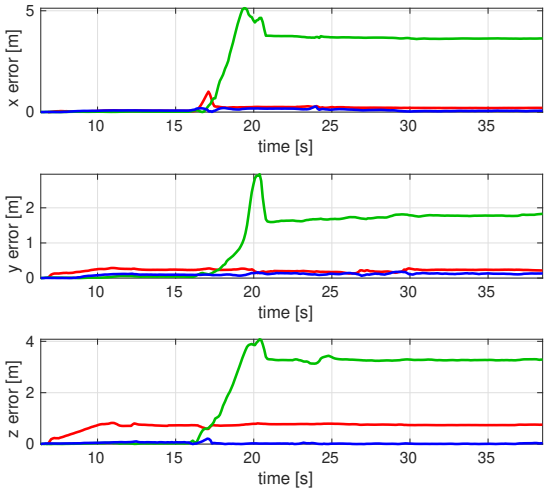
Estimation performance degradation at $t = 15$ s is caused by a high forward acceleration, which results in a rapid change in pitch angle (Fig. 5.10a). This rotation caused a rapid field of view change in each camera, resulting in most existing feature trails disappearing and requiring the initialization of many new feature trails. Triangulation is required to initialize the value of each new feature trail added to the optimization problem. Because the forward camera's average scene depth at this point in the trajectory was greater than the downward camera's, triangulation was much less accurate for the forward camera. This is reflected by the fact that after $t = 15$ s a high proportion of downward camera reprojection residuals have low values after optimization (Fig. 5.11a, Fig. 5.11c bottom) while a low proportion of forward camera reprojection residuals have low values after optimization (Fig. 5.11b, Fig. 5.11c top).

Poor feature triangulation is directly responsible for the greater translational error in forward camera only estimation. Although downward only estimation avoids a rapid increase in translational error between $t = 16$ s and $t = 21$ s, it has a non-trivial elevation error due to poor scale initialization during takeoff. By using the forward camera to initialize during takeoff and relying more heavily on downward camera observations immediately after $t = 15$ s, two camera estimation is able to achieve lower translational error than with either camera alone.

Despite having better translational accuracy, two camera estimation has worse heading accuracy than when using either camera alone (Table 5.2). This is mainly due to the optimizations in the dynamic motion

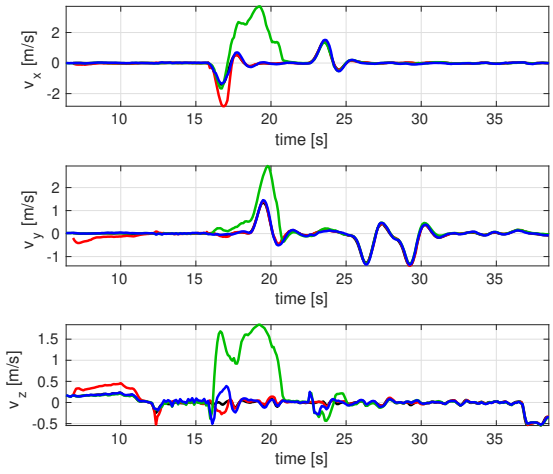


(a) Position estimates and motion capture ground truth (black)

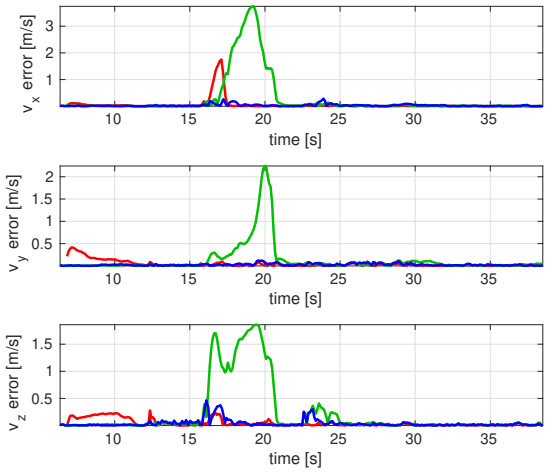


(b) Absolute value of position error

Figure 5.8: Position estimates and errors of visual-inertial odometry using only the downward camera (red), only the forward camera (green), and both cameras (blue) on the motion capture dataset.

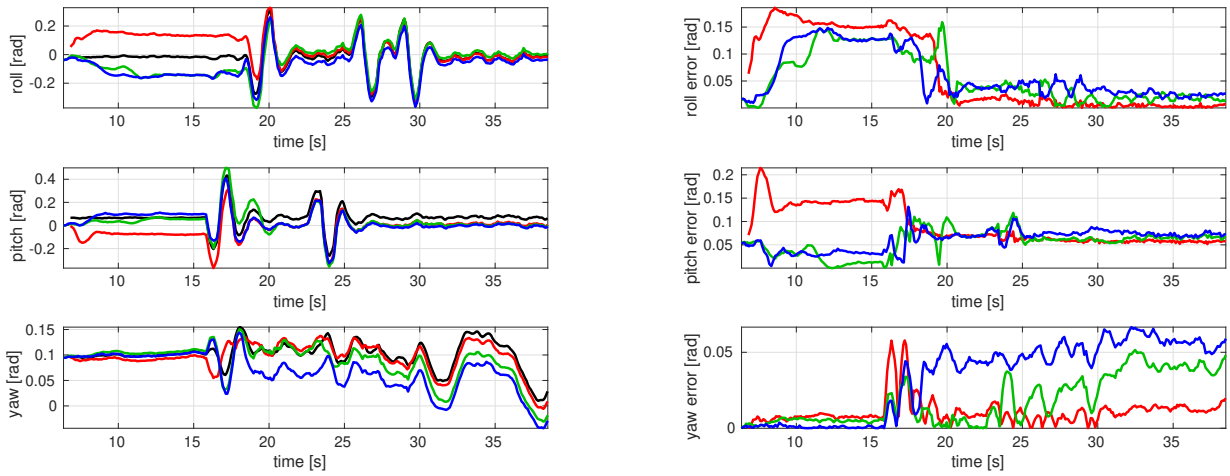


(a) Velocity estimates and motion capture ground truth (black)



(b) Absolute value of velocity error

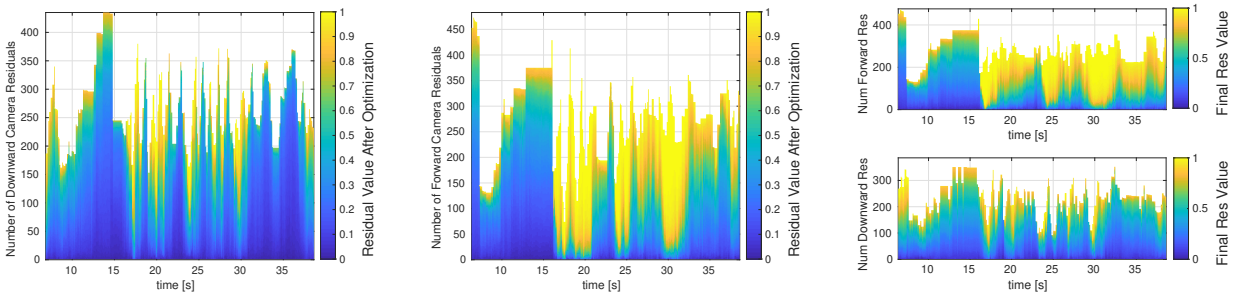
Figure 5.9: Velocity estimates and errors of visual-inertial odometry using only the downward camera (red), only the forward camera (green), and both cameras (blue) on the motion capture dataset.



(a) Attitude estimates and motion capture ground truth (black)

(b) Absolute value of attitude error

Figure 5.10: Attitude estimates and errors of visual-inertial odometry using only the downward camera (red), only the forward camera (green), and both cameras (blue) on the motion capture dataset.



(a) Downward camera only

(b) Forward camera only

(c) Both cameras

Figure 5.11: Distribution of post-optimization reprojection residual values for VIO with different camera configurations on the motion capture dataset. The upper boundary of the colored region represents the total number of reprojection residuals from a given camera as a function of time. The color at location (x, y) denotes the after-optimization value of the y^{th} smallest value reprojection residual at time x, where blue denotes smaller values and yellow denotes larger values.

interval between $t = 16$ s and $t = 21$ s that achieved greater horizontal translational accuracy at the expense of elevation and heading accuracy. The results of postprocessing on the motion capture arena flight dataset demonstrate that although using more cameras does not always lead to better estimation accuracy in all degrees of freedom, it is a reliable way to avoid failures in scenarios that are unfavorable for single camera estimation.

Table 5.2: Drift in position and heading estimates at the end of the motion capture arena flight. The total trajectory length is 9.67 *m.*

Camera Configuration	Position Error [m]	Heading Error [deg]
Downward only	0.81	1.1
Forward only	5.23	2.8
Downward and forward	0.15	3.4

5.4.2 Parking Garage

In this dataset, the aerial robot (Sect. G.1.1) started at the bottom level of a multi-storey parking garage (Fig. 5.12) and flew along the center of the single driving lane until it landed two floors above its takeoff location. This approximately 250 m long trajectory consists of two clockwise rectangular segments with continuously increasing elevation that follow the driving lane as it spirals up two floors (see Fig. 5.13).

Figure 5.12: The parking garage environment contains variable lighting conditions.

Figure 5.13 shows that using both downward and forward cameras for state estimation yields a more accurate horizontal position estimate than either downward or forward cameras alone. Although horizontal ground truth was not available for this dataset, we can qualitatively determine that the two-camera estimate is closest to being rectangular, which is the approximate shape of the actual trajectory. Using the downward camera alone results in high drift because ground features move out of the field of view too quickly to impose constraints that bound motion estimate drift. When using the forward camera alone, initialization does not succeed until almost a minute into the four-minute trajectory (Fig. 5.14). This may be caused by poor feature tracking due to high contrast induced by variable lighting conditions that affect the forward camera image more than the downward camera image. Despite this, forward-camera only estimation yields a qualitatively better horizontal trajectory estimate (Fig. 5.13) and a quantitatively better heading estimate (Table 5.3) than downward-camera only estimation. Both sets of single-camera estimation results have high elevation estimate drift in different directions. By fusing both cameras together, two-camera estimation achieves lower elevation error (Fig. 5.14, Table 5.3).

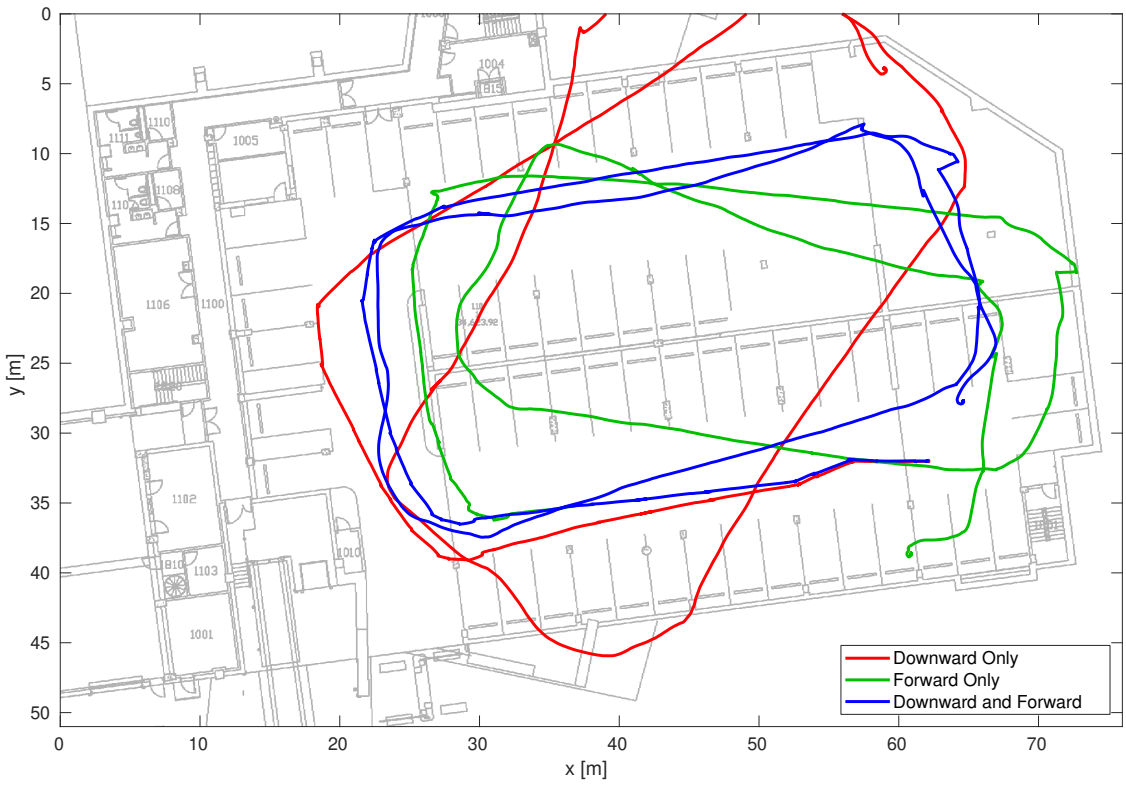


Figure 5.13: Trajectory estimates using downward camera, forward camera, and both cameras are overlaid on a floor plan of the parking garage's lower level. Although the second loop of the trajectory occurs in the parking garage's upper level, we do not show the upper level floor plan to avoid clutter.

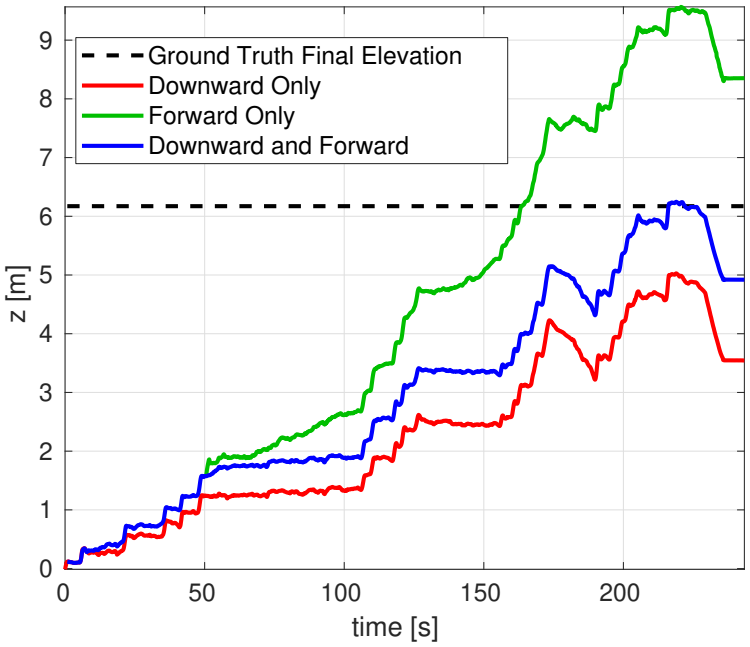


Figure 5.14: Elevation estimates over the length of the trajectory for each camera configuration compared with the ground truth final elevation.

Table 5.3: Drift in elevation and heading estimates at the end of the parking garage flight. The total elevation change and heading change over the trajectory are $+6.17$ *m and* -11.78 *rad, respectively.*

Camera Configuration	**Elevation Error [m]**	**Heading Error [deg]**
Downward only	2.63	72
Forward only	2.18	31
Downward and forward	1.25	4

Chapter 6

Sensor Resource Allocation

State estimation with limited computational resources is an important capability for robots with non-trivial size, weight, and power constraints. For visual-inertial odometry, the main computational cost comes from solving optimization problems to perform probabilistic inference over sliding window motion states and features. The time required to perform an iteration of nonlinear least squares optimization scales with the size of the optimization vector, which consists of motion states and feature trails. For a sliding window with a fixed number of motion states, achieving computationally efficient state estimation requires balancing the competing objectives of maximizing localization accuracy while minimizing the number of feature trails in the optimization vector.

In standard sparse visual-inertial odometry algorithms, the front-end typically selects new features based only on their distinctiveness while enforcing heuristics such as uniform distribution across the camera's field of view (Sect. 3.1.1). However, the utility of each individual feature for state estimation varies. Features that remain in the field of view for long periods of time and can be reliably tracked provide more useful information than features that lack these properties. An algorithm that selects the most informative features requires a way to simulate them over the near-term time horizon, which in turn requires knowledge about future platform motion and the surrounding environment. In real-world scenarios, simplifying assumptions and approximations must be made to facilitate forward simulations of robot dynamics and feature observations.

6.1 Problem Formulation

The proposed problem formulation is inspired by [48] with a few modifications. We divide the set of all feature trails of sufficient length that are visible in the current keyframe into $\mathcal{H}$, the features that are already being used in optimization, and $\mathcal{F}$, the features that have not yet been used in optimization. The sensor resource allocation problem selects a subset $\mathcal{G}$ from $\mathcal{F}$ for inclusion in visual-inertial optimization to maximize predicted motion estimation accuracy while respecting a limit on optimization problem size imposed by the amount of available computation.

$$\begin{aligned} &\max_{\mathcal{G} \subset \mathcal{F}} \log\det \mathbf{M}(\mathcal{G} \cup \mathcal{H}) \\ &|\mathcal{G} \cup \mathcal{H}| \leq L \end{aligned} \tag{6.1}$$

Predicted motion estimation accuracy is represented by the log determinant of the joint information matrix, $\mathbf{M}$, associated with n position estimates over the upcoming sliding window, $\{\mathbf{p}_i\}_{i=1}^{n}$. The determinant of the joint covariance matrix $\mathbf{M}^{-1}$ represents the volume of the confidence ellipsoid, which is a probabilistic bound on estimation error. Increasing estimation accuracy is equivalent to decreasing the volume of the covariance ellipsoid or increasing the volume of the information ellipsoid, which yields the objective function of (6.1).

For a fixed sliding window size, the number of motion states in the visual-inertial optimization problem is constant. Therefore, keeping a constant optimization problem size requires limiting the number of features used in optimization to a fixed feature budget, L. Assuming currently visible features that are already used in optimization will remain in the next optimization, a maximum of $L - |H|$ new feature trails may be selected from $\mathcal{F}$.

Although the upcoming sliding window information gain matrix is the sum of contributions from inertial as well as visual observations, the former does not depend on the subset of features that is chosen.

$$\mathbf{M}(\mathcal{G} \cup \mathcal{H}) = \mathbf{M}_{\text{imu}} + \mathbf{M}_{\text{visual}}(\mathcal{G} \cup \mathcal{H}) \tag{6.2}$$

Therefore, unlike [48] we do not compute the information gain associated with IMU observations via an explicit IMU dynamics model to avoid unnecessary computational cost. Instead, we approximate it with a diagonal matrix whose single parameter is a user-defined constant.

$$\mathbf{M}_{\text{imu}} = \frac{1}{\sigma_{\text{imu}}^2}\mathbf{I}_{3n\times 3n} \tag{6.3}$$

The information gain due to visual observations is the sum of information gains associated with using a set of feature trails in estimation, $\{\Lambda_l\}_{l\in\mathcal{G}\cup\mathcal{H}}$. We model the probability that the l^{th} feature trail is successfully tracked in all keyframes it is visible in with the Bernoulli random variable b_l, where $b_l = 1$ denotes success and $b_l = 0$ denotes failure (i.e. the feature is not tracked in any keyframe).

$$\mathbf{M}_{\text{visual}}(\mathcal{G} \cup \mathcal{H}) = \sum_{l\in\mathcal{G}\cup\mathcal{H}} b_l\Lambda_l \tag{6.4}$$

Because of the introduction of stochasticity, we modify the objective in (6.1) to use the expectation over $\{b_l\}_{l\in\mathcal{G}\cup\mathcal{H}}$.

$$g(\mathcal{G}) = \mathbb{E}_{\{b_l\}_{l\in\mathcal{G}\cup\mathcal{H}}}\left[\log\det\left(\mathbf{M}_{\text{imu}} + \sum_{l\in\mathcal{G}\cup\mathcal{H}} b_l\Lambda_l\right)\right] \tag{6.5}$$

Next, apply Jensen's inequality to obtain a lower bound on the expectation:

$$g(\mathcal{G}) \geq \log\det\left(\mathbb{E}_{\{b_l\}_{l\in\mathcal{G}\cup\mathcal{H}}}\left[\mathbf{M}_{\text{imu}} + \sum_{l\in\mathcal{G}\cup\mathcal{H}} b_l\Lambda_l\right]\right) \tag{6.6}$$

We assume that the probability p_l of the outcome $b_l = 1$ to be correlated with Shi-Tomasi corner quality score in order to capture the intuition that more distinctive feature trails are more likely to be successfully tracked. This implies $\mathbb{E}[b_l] = p_l$, which can be substituted into (6.6):

$$g(\mathcal{G}) \geq \log\det\left(\mathbf{M}_{\text{imu}} + \sum_{l\in\mathcal{G}\cup\mathcal{H}} p_l\Lambda_l\right) = f(\mathcal{G}) \tag{6.7}$$

The right hand side of (6.7) is used as the objective function in (6.1).

6.1.1 Log Determinant Evaluation

To ensure numerical stability when evaluating the objective function of (6.1), the log determinant of a $m \times m$ positive definite symmetric matrix $\mathbf{M}$ is computed via Cholesky decomposition.

$$\mathbf{M} = \mathbf{L}\mathbf{L}^{\text{T}} \tag{6.8}$$

$$\log\det\mathbf{M} = \log\left[\det\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)\right] \tag{6.9}$$
$$\log\det\mathbf{M} = \log\left[\det\mathbf{L}\det\mathbf{L}^{\mathrm{T}}\right] \tag{6.10}$$
$$\log\det\mathbf{M} = \log\left[\det\mathbf{L}\det\mathbf{L}\right] \tag{6.11}$$
$$\log\det\mathbf{M} = 2\log\left[\det\mathbf{L}\right] \tag{6.12}$$

Because the Cholesky factor $\mathbf{L}$ is triangular, its determinant is the product of its diagonal elements $\{l_i i\}_{i=1}^{m}$.

$$\log\det\mathbf{M} = 2\log\left[\prod_{i=1}^{m} l_{ii}\right] \tag{6.13}$$
$$\log\det\mathbf{M} = 2\sum_{i=1}^{m}\log l_{ii} \tag{6.14}$$

6.1.2 Feature Observation Prediction Model Assumptions

- There are no occlusions. Features are either tracked in all keyframes in which they are in the field of view or in no keyframes at all.
- The set of currently visible features represents the set of all features in the environment that will be visible in the upcoming sliding window's keyframes.
- The set of currently visible features is accurately triangulated.
- The sensor platform perfectly tracks the trajectory reference used to simulate upcoming sliding window observations.

6.1.3 Feature Information Gain Evaluation

A simplified linear measurement model is used to evaluate feature trail l's information gain. Let $\mathbf{u}_{ckl}$ be the unit vector corresponding to the observation of feature l in keyframe k of camera c. A geometric relationship between the unit vector observation and the estimated variables comes from the fact that that the unit direction vector should be parallel to the feature's 3D position in the camera frame (Fig. 6.1).

$$\mathbf{u}_{ckl} \times \left[\left(\mathbf{R}_k\mathbf{R}_{bc}\right)^{\mathrm{T}}\left(\mathbf{l} - \left(\mathbf{p}_k + \mathbf{R}_k\mathbf{p}_{bc}\right)\right)\right] = \mathbf{0}_{3\times 1} \tag{6.15}$$

In the above expression, $(\mathbf{R}_k, \mathbf{p}_k)$ is the rigid body transform that takes vectors from the k^{th} keyframe's body frame to the world frame, $(\mathbf{R}_{bc}, \mathbf{p}_{bc})$ is the rigid body transform that takes vectors from the camera c frame to the body frame, and $\mathbf{l}$ is feature l's location in the world frame.

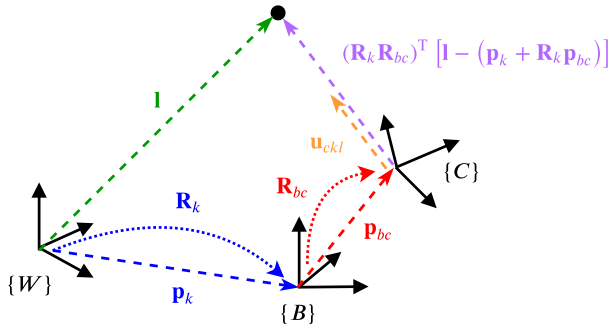


Figure 6.1: Geometry associated with the simplified linear measurement model (6.15)

Expand and rearrange terms in (6.15) to isolate the observation on the left hand side.

$$-\left[\mathbf{u}_{ckl}\right]_{\times}\left(\mathbf{R}_k\mathbf{R}_{bc}\right)^{\mathrm{T}}\mathbf{R}_k\mathbf{p}_{bc} = \left[\mathbf{u}_{ckl}\right]_{\times}\left(\mathbf{R}_k\mathbf{R}_{bc}\right)^{\mathrm{T}}\mathbf{p}_k - \left[\mathbf{u}_{ckl}\right]_{\times}\left(\mathbf{R}_k\mathbf{R}_{bc}\right)^{\mathrm{T}}\mathbf{l} \tag{6.16}$$

$$\mathbf{z}_{ckl} = \mathbf{F}_{ckl}\mathbf{p}_k + \mathbf{E}_{ckl}\mathbf{l} \tag{6.17}$$

Note that the observation model (6.17) is linear because it only depends on keyframe and feature positions by assuming keyframe orientations to be known. This assumption is reasonable because roll and pitch are directly observable in a visual-inertial odometry problem, while typical sliding window horizons are short enough that yaw does not drift significantly.

For each feature l, we stack all keyframe observations (6.17) into a single matrix equation:

$$\mathbf{z}_{cl} = \mathbf{F}_{cl}\mathbf{p}_{1:n} + \mathbf{E}_{cl}\mathbf{l} \tag{6.18}$$

Rewrite (6.18) in terms of the joint state that consists of positions and feature locations.

$$\mathbf{z}_{cl} = \begin{bmatrix}\mathbf{F}_{cl} & \mathbf{E}_{cl}\end{bmatrix}\begin{bmatrix}\mathbf{p}_{1:n} \\ \mathbf{l}\end{bmatrix} \tag{6.19}$$

The information matrix associated with the position and feature location states is given by

$$\begin{bmatrix}\mathbf{F}_{cl} & \mathbf{E}_{cl}\end{bmatrix}^{\mathrm{T}}\begin{bmatrix}\mathbf{F}_{cl} & \mathbf{E}_{cl}\end{bmatrix} = \begin{bmatrix}\mathbf{F}_{cl}^{\mathrm{T}}\mathbf{F}_{cl} & \mathbf{F}_{cl}^{\mathrm{T}}\mathbf{E}_{cl} \\ \mathbf{E}_{cl}^{\mathrm{T}}\mathbf{F}_{cl} & \mathbf{E}_{cl}^{\mathrm{T}}\mathbf{E}_{cl}\end{bmatrix} \tag{6.20}$$

Marginalize out the feature location using the Schur complement to obtain the information gain for the position states of the upcoming sliding window.

$$\Lambda_l = \mathbf{F}_{cl}^{\mathrm{T}}\mathbf{F}_{cl} - \mathbf{F}_{cl}^{\mathrm{T}}\mathbf{E}_{cl}\left(\mathbf{E}_{cl}^{\mathrm{T}}\mathbf{E}_{cl}\right)^{-1}\mathbf{E}_{cl}^{\mathrm{T}}\mathbf{F}_{cl} \tag{6.21}$$

Note that we ignore the information gain of keyframe orientations as well as feature locations in (6.21) and ultimately in the predicted estimation accuracy objective function (6.7). Ignoring the former is justified because orientation estimation performance is typically quite good over short time horizons, while ignoring the latter is justified because the goal is ego-motion estimation rather than mapping.

Evaluating (6.21) for all features in $\mathcal{G} \cup \mathcal{H}$ requires near-term pose predictions for all keyframes in the upcoming sliding window, $\{\mathbf{R}_k, \mathbf{p}_k\}_{k=1}^{n}$. These values can be obtained by interpolating the desired position and orientation references output by the planner under the assumption that state estimation and control are capable of accurately tracking the reference signal. Unit direction vectors $\mathbf{u}_{ckl}$ are computed by projecting features into the appropriate keyframe's camera frame, assuming that their 3D world frame locations have been accurately triangulated and that there are no occlusions.

6.2 Greedy Feature Selection

Algorithm 1 describes a greedy algorithm for selecting a subset of features to maximize estimation accuracy[1]. Starting with an empty set, it adds the most informative feature on each iteration until the feature budget is reached.

The main computational cost associated with Algorithm 1 is the repeated evaluation of (6.7) on line 8. A much smaller computational cost is associated with evaluating the predicted future information gain of all currently visible features, $\{\Lambda_l\}_{l\in\mathcal{F}\cup\mathcal{H}}$. The additional computational cost incurred by running the greedy informative feature selection algorithm must be lower than the computational savings resulting from a reduced optimization problem size in order for the sensor allocation strategy to be viable[2].

[1] This is a simplified version of Algorithm 1 in [48]

[2] Figure 6.9c in Sect. 6.4 shows that this condition is satisfied when running on real sensor data

Algorithm 1 Greedy selection of informative features

Input: σ_{imu}, $\{\Lambda_l, p_l\}_{l \in \mathcal{F} \cup \mathcal{H}}$, $N = \min(|\mathcal{F}|, L - |H|)$
Output: $\mathcal{G}$
$\mathcal{G} = \emptyset$
for $i = 1, \ldots, N$ **do**
    $f_{\max} = -1$
    $l_{\max} = -1$
    for $l \in \mathcal{F} \setminus \mathcal{G}$ **do**
        if $f(\mathcal{G} \cup l) > f_{\max}$ **then**
            $f_{\max} = f(\mathcal{G} \cup l)$
            $l_{\max} = l$
        end if
    end for
    $\mathcal{G} \leftarrow \mathcal{G} \cup l_{\max}$
end for

6.3 Simulation Results

We ran three simulation experiments to compare the performance of the proposed greedy feature selection strategy with a baseline feature selection strategy. The simulation scenarios involve an aerial robot equipped with a forward-facing and downward-facing camera in an environment with known feature locations along planar surfaces. The 4 m $\times$ 2 m $\times$ 2 m hallway environment has features along the floor, ceiling, and two lateral walls. The 4 m $\times$ 4 m $\times$ 2 m room environment has features along the floor, ceiling, and four lateral walls. In each scenario, the robot has knowledge of the trajectory it is tracking over the duration of the next sliding window time interval. The trajectory is evaluated at 5 keyframe times to obtain the position states of the sliding window. Known camera intrinsic and extrinsic parameters enable the computation of future feature observations along the sliding window.

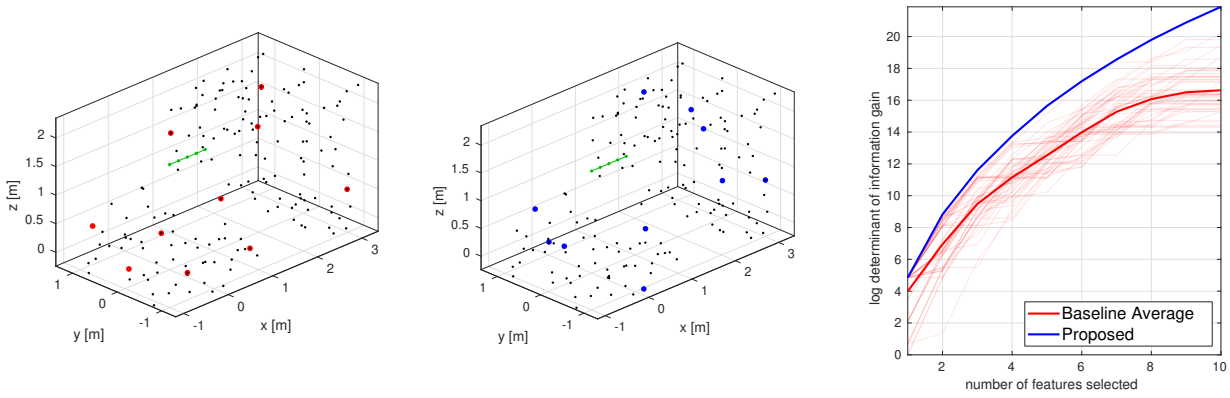
The hall-ceiling scenario consists of robot motion down the hallway environment's longitudinal axis while maintaining velocity-aligned heading and cross-sectional position on the hallway centerline 40 cm below the ceiling. The hall-floor scenario is the same as hall-ceiling, but with the robot maintaining a height of 50 cm above the floor. The room-circle scenario consists of a counterclockwise motion with velocity-aligned heading along the circumference of a 0.5 m radius circle at a height of 1 m in the center of the room environment.

The baseline feature selection strategy splits up the feature budget evenly between the cameras. Within each camera, features are chosen randomly in a way that maximizes the minimum angular distance between neighboring feature bearing vectors to ensure uniform coverage of the field of view. If any camera does not have enough features in its field of view that satisfy the above condition, its excess feature budget is distributed amongst the other cameras. The proposed method is implemented as described in Sections 6.1-6.2. In order to highlight the impact of newly added features on the optimization, both the baseline and proposed methods assume that the optimization at the time of feature selection does not include any previously used features (i.e. $\mathcal{H} = \emptyset$).

We run both the baseline and proposed feature selection strategies on each scenario with a feature budget of 10. Because of the randomness in the baseline method, we run it 50 times and show average results. For the sake of simplicity, feature quality is assumed to be constant for all features.

In each of the scenarios, the log determinant of the predicted information gain is higher for the proposed strategy than for the baseline strategy. This is not surprising, as the greedy feature selection strategy specifically seeks to maximize information gain. Figures 6.2c, 6.3c, and 6.4c show that the proposed method yields

higher information gain and hence lower uncertainty for estimated position states over the upcoming sliding window keyframes.



(a) Baseline feature selection (b) Proposed feature selection (c) Predicted information gain

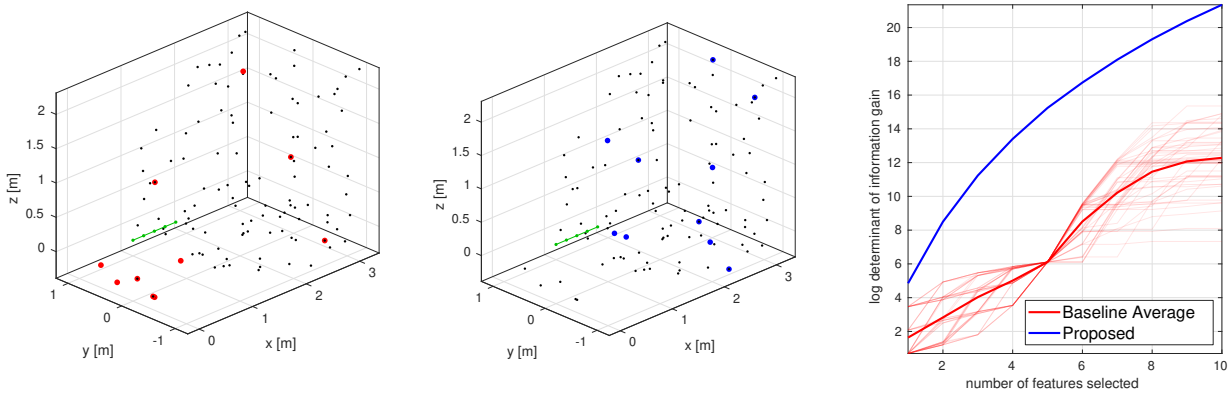
Figure 6.2: Comparison of baseline and proposed feature selection strategies for a robot with forward and downward-facing cameras moving down the centerline of a hallway while staying close to the ceiling. The robot both faces and moves along the $+x$-axis. In (a)-(b) the trajectory connecting sliding window positions is depicted as a green line, features visible in the first sliding window keyframe are denoted by black dots, and large colored dots denote selected features. Plot (a) shows the selected features for one of the many randomized baseline method trials. The plot in (c) compares $f(\mathcal{G})$ from (6.7) for the proposed and baseline feature selection strategies. Light red lines denote individual Monte Carlo runs for the baseline method, while the dark red line denotes their average.

The features selected by the proposed approach in the hall-ceiling scenario are evenly split between forward and downward cameras (Fig. 6.2b), while those in the hall-floor scenario are all from the forward camera (Fig. 6.3b). When the robot is close to the floor, floor features in the downward camera's field of view go outside of it very quickly and are therefore less useful for position estimation. On the other hand, wall features in the forward camera's field of view remain in it for the entire sliding window and provide more utility for position estimation. When the robot is close to the ceiling, floor features in the downward camera's field of view remain in it for longer and are more useful for estimation. Although both the baseline and proposed approaches evenly split the feature budget between the two cameras (Fig. 6.2a-6.2b), the latter still has a higher utility because it considers anticipated motion over the sliding window.

The features selected by the proposed approach in the room-circle scenario are all from the downward camera (Fig. 6.4b). When executing a velocity-aligned circular trajectory, wall features visible to the forward camera go out of view very quickly, while floor features visible to the downward camera remain in the field of view for the entire sliding window.

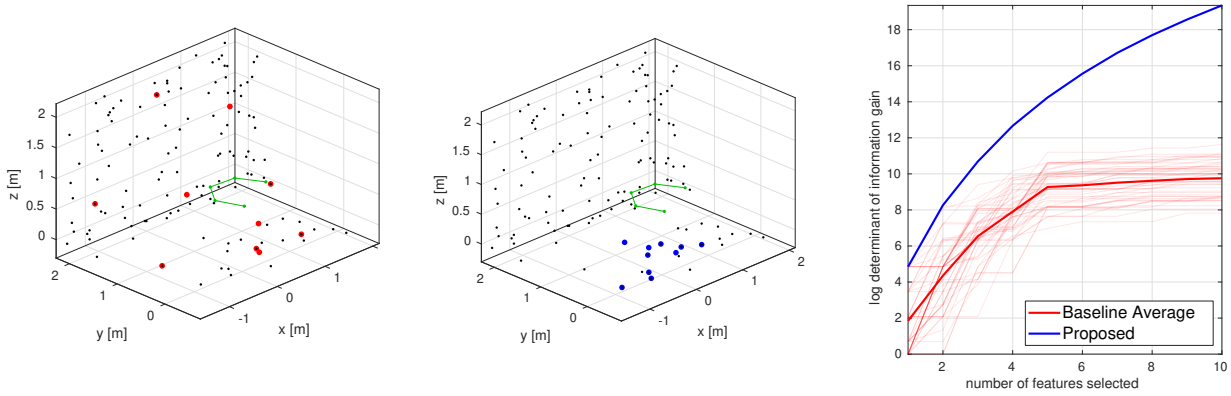
The horizontal distance between the red and blue lines in the information gain plots of Figures 6.2-6.4 depicts the reduction in the number of features required to achieve a given level of estimation accuracy. For example, the proposed method in the hall-floor scenario can achieve the same level of estimation strategy using 3 features that the baseline strategy achieves with 8 features.

One note of caution is that the simulation results presented in this section do not represent the position accuracy obtained from running the proposed 2-camera visual inertial state estimator over the upcoming sliding window, but rather the position accuracy of a batch triangulation of the upcoming sliding window with currently visible features. Achieving the former results requires implementing an interface between the toy scenario's oracle front-end and the proposed state estimator's back-end, which is an item of proposed work. The similarity between actual position uncertainty reduction and the simulation model will depend on the validity of the simulation's feature observation prediction model assumptions for the visual environment under consideration.



(a) Baseline feature selection (b) Proposed feature selection (c) Predicted information gain

Figure 6.3: Comparison of baseline and proposed feature selection strategies for a robot with forward and downward-facing cameras moving down the centerline of a hallway while staying close to the floor. The robot both faces and moves along the $+x$-axis. In (a)-(b) the trajectory connecting sliding window positions is depicted as a green line, features visible in the first sliding window keyframe are denoted by black dots, and large colored dots denote selected features. Plot (a) shows the selected features for one of the many randomized baseline method trials. The plot in (c) compares $f(\mathcal{G})$ from (6.7) for the proposed and baseline feature selection strategies. Light red lines denote individual Monte Carlo runs for the baseline method, while the dark red line denotes their average.



(a) Baseline feature selection (b) Proposed feature selection (c) Predicted information gain

Figure 6.4: Comparison of baseline and proposed feature selection strategies for a robot with forward and downward-facing cameras moving in a tight counterclockwise circle in a fully enclosed rectangular room with constant height and velocity-aligned heading. In (a)-(b) the trajectory connecting sliding window positions is depicted as a green line, features visible in the first sliding window keyframe are denoted by black dots, and large colored dots denote selected features. Plot (a) shows the selected features for one of the many randomized baseline method trials. The plot in (c) compares $f(\mathcal{G})$ from (6.7) for the proposed and baseline feature selection strategies. Light red lines denote individual Monte Carlo runs for the baseline method, while the dark red line denotes their average.

6.4 Experimental Results

This section presents postprocessed results that demonstrate the superiority of the proposed information gain feature selection strategy for two cameras over the even split strategy for two cameras (described in Sect. 5.3.3) and for one camera (described in Sect. 3.5.2) with a fixed total feature budget. Images and IMU observations collected on a multirotor aerial robot were used to run sliding window visual-inertial odometry with different feature selection strategies and compare their performance. The experimental results show that the proposed information gain feature selection strategy achieves greater estimation accuracy than the alternative methods given the same computational cost footprint.

All results were generated via postprocessing on a Lenovo Thinkpad T430s laptop[3] running Ubuntu 16.04 LTS.

Information Gain Feature Selection Implementation Details

We use a prediction horizon of 3 s for all information gain feature selection postprocessing results. This means that the future poses used to evaluate information gain are associated with n time instances spread evenly over the 3 s after the latest sliding window keyframe, where n is the sliding window size. Note that the prediction horizon is not need to be equal to the total time covered by the sliding window used in optimization. Using longer prediction horizons encourages the selection of features that will be visible for longer periods of time, while using shorter prediction horizons conforms more closely to feature observation prediction model's assumptions (Sect. 6.1.2).

Future trajectory predictions used by the information gain feature selection strategy were obtained by querying ground truth or a previously computed single camera visual-inertial trajectory estimate (in datasets without ground truth) at future timestamps. The motion capture results (Sect. 6.4.1) used ground truth, while the outdoor results (Sect. 6.4.2-6.4.3) used previously computed single camera trajectory estimates because total station ground truth measurements were incomplete and lack attitude. 4DOF alignment was used to achieve position and yaw continuity between the estimate at the latest sliding window keyframe and the queried future poses. Although ground truth and previously computed trajectory estimates are a non-causal source of future pose predictions and therefore unsuitable for real-time operation, we use them in postprocessing because our datasets lack planner reference signals from which future poses may be queried during real-time operation. Future ground truth observations are queried in [48] to evaluate feature selection in postprocessing on datasets that lack control or planner reference signals. Overall, the impact of using ground truth instead of planner references is insignificant for short prediction horizons and when the controller is capable of reliably following motion reference signals.

6.4.1 Motion Capture Arena

In this dataset the aerial robot flies along a 3D trajectory inside a motion capture arena (Fig. G.4). Although the entire flight duration is over three minutes and postprocessing uses observations from the entire time interval, the results in this section only pertain to the time interval between $t = 45$ s and $t = 80$ s. This 35 second time interval was chosen as a representative example of a dynamic motion where the choice of feature selection strategy has a non-trivial impact on estimation accuracy. $t = 45$ s is much later than the time of initialization to ensure that the sliding window optimization has marginalized out initial transients prior to the interval of interest. Trajectory estimation accuracy values reported in this section are translational absolute trajectory errors (H.13) over the time interval of interest ($45 \leq t \leq 80$ s) obtained using the multiple position correspondences method (Appendix H.3.2).

Fixed Feature Budget Performance Comparison

In this set of experiments, each of the three feature selection strategies was run with a total feature budget of 30. The 1 camera approach allocated the entire feature budget to the forward camera, the 2 camera even

[3] Intel Core i7-3520M at 2.9 GHz $\times$ 4, 8 GB RAM

split approach allocated a feature budget of 15 to each of the forward and downward cameras, and the 2 camera information gain approach allocated a feature budget of 30 for both cameras. Figure 6.5 depicts the aligned position estimates and ground truth over the 35 second time interval of interest.

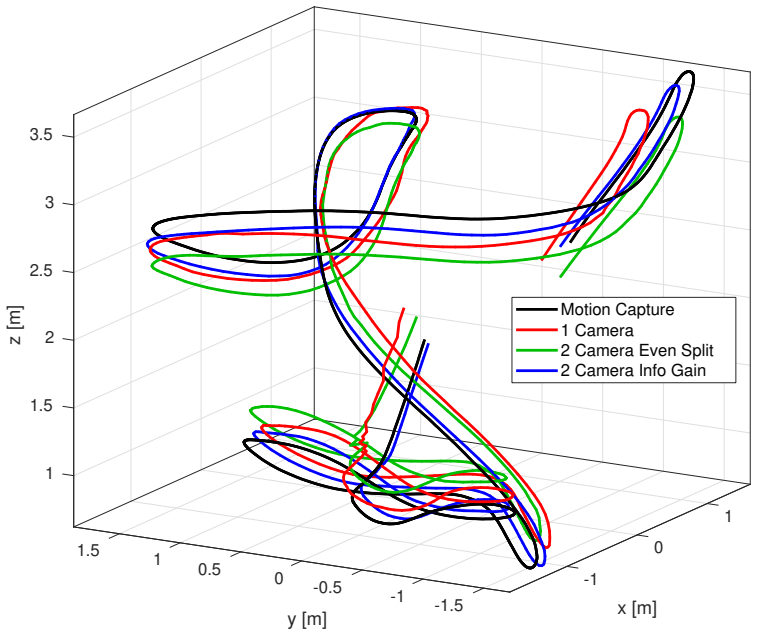


Figure 6.5: 3D plots of the trajectories estimated by VIO with different feature selection strategies for the motion capture flight. The ground plane is located at $z = 0$ m.

The top plot of Figure 6.6 shows that the trajectory segment of interest begins at a low altitude of around 1 m and transitions to a high altitude of around 3 m at $t = 62$ s. Accordingly, the downward camera is closer to the ground and has higher optical flow during the first half the time interval than the last (compare Fig. 6.7b with Fig. 6.7d). Higher optical flow causes features to exit the downward camera's field of view faster and yield feature reprojection factors that are less effective at mitigating motion drift in the optimization problem. The 2 camera even split strategy allocates half of its feature budget to the downward camera and incurs the highest z estimation error (middle plot in Fig. 6.6). The 2 camera information gain strategy starts reducing the downward camera feature budget and increasing the forward camera feature budget at around $t = 50$ s (bottom plot in Fig. 6.6). By increasing the number of forward camera features that remain in the field of view for longer periods of time, the information gain strategy is able to achieve the lowest error. Although this reasoning may also be applied to the 1 camera case where the entire feature budget is allocated to the forward camera, it has lower accuracy than the 2 camera information gain strategy due to its poorer scale estimate. The 2 camera information gain strategy outperforms the 1 camera case by virtue of choosing a few longer-lived downward camera features that improve scale estimation and outperforms the 2 camera even split strategy by virtue of avoiding short-lived downward camera features that yield poor motion constraints.

In the second half of the trajectory segment, the vehicle flies higher but also varies its heading significantly. The downward camera's optical flow is reduced, but the forward camera's optical flow is increased. Anticipating this by querying future motion predictions, the 2 camera information gain strategy increases the downward camera feature budget and decreases the forward camera feature budget to maximize the number of long-lived feature trails that yield good motion constraints for optimization. On the other hand, the 2 camera even split and 1 camera methods both use a greater number of forward camera features in optimization, many of which are short-lived and yield poorer motion constraints. Consequently, the 2 camera information gain strategy achieves the lowest z estimation error (middle plot in Fig. 6.6) as well as the

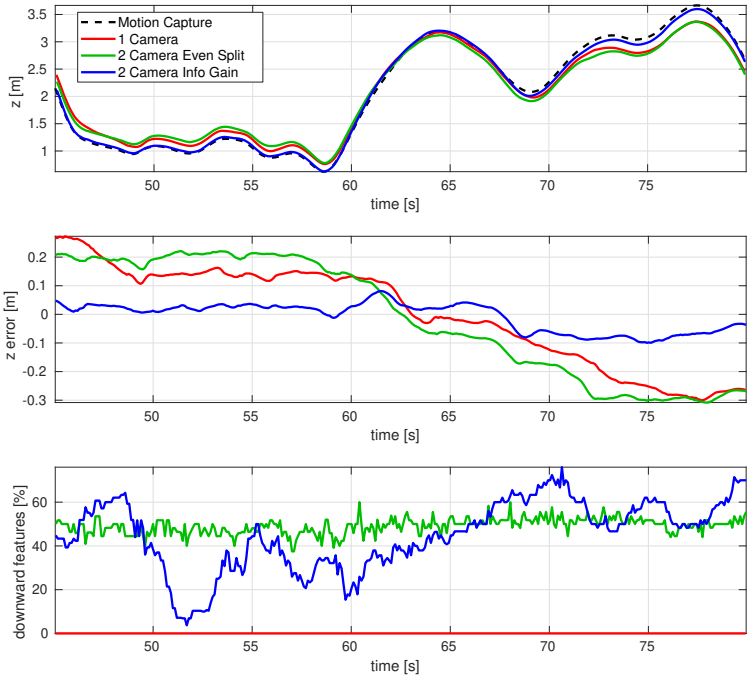


Figure 6.6: Top: Altitude estimates compared with motion capture ground truth. Middle: Altitude estimation error with respect to motion capture. Bottom: Percentage of the total feature budget allocated to the downward camera. The 2 camera information gain feature selection strategy achieves the lowest altitude estimation error because it changes its feature budget allocation between forward and downward cameras based on the current scene structure and predicted future motion.

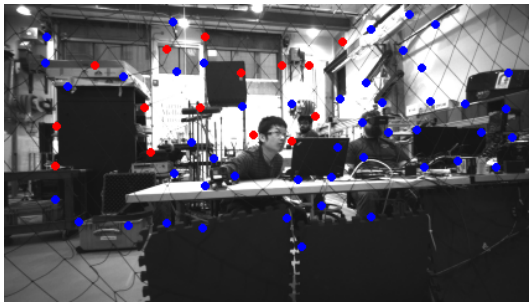
lowest translational absolute trajectory error (Tab. 6.1) over the time interval of interest.

Table 6.1: Translational absolute trajectory errors of the 3 feature selection strategies between $t = 45$ s and $t = 80$ s on the motion capture dataset

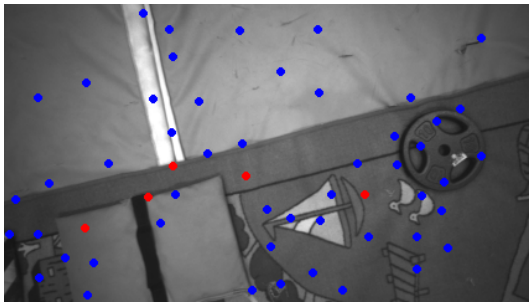
	1 Camera	**2 Camera Even Split**	**2 Camera Info Gain**
Absolute Trajectory Error (m)	0.12	0.16	0.07

Figures 6.7 and 6.8 show the features selected for optimization by the 2 camera even split and information gain strategies at $t = 51.6$ s and $t = 70.6$ s. Note that the number of selected features across the forward and downward cameras does not sum to the total feature budget because the figures only depict features that are visible in the latest sliding window keyframe. This discrepancy is explained by the fact that some of the feature trails used in optimization have exited the camera field of view before the latest sliding window keyframe.

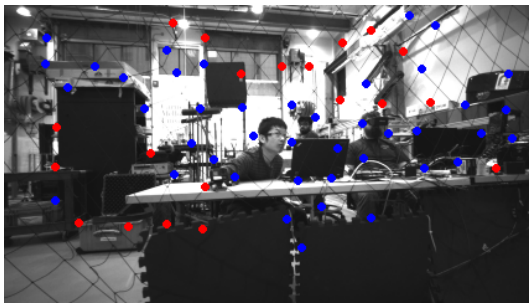
At $t = 51.6$ s when the robot is closer to the ground, the 2 camera information gain approach allocates a smaller number of downward camera features than the 2 camera even split approach (compare Fig. 6.7d and Fig. 6.7b). At $t = 70.6$ s when the robot is at a higher altitude, the 2 camera information gain approach allocates a greater number of forward camera features than than the 2 camera even split approach (compare Fig. 6.8c and Fig. 6.8a). Furthermore, features selected by the 2 camera information gain approach are spread out over a larger portion of the field of view than the 2 camera even split approach due to awareness of the vehicle's near-term yawing motion.



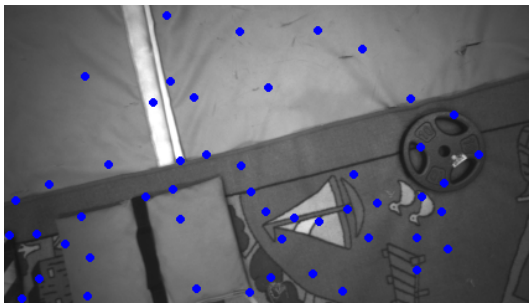
(a) Even split - forward camera features



(b) Even split - downward camera features

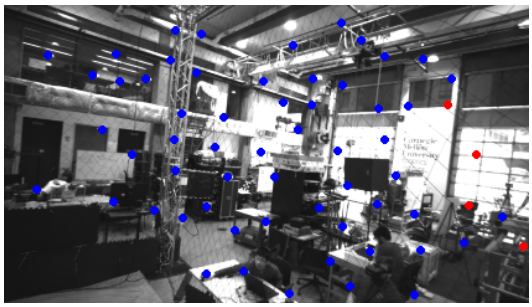


(c) Info gain - forward camera features

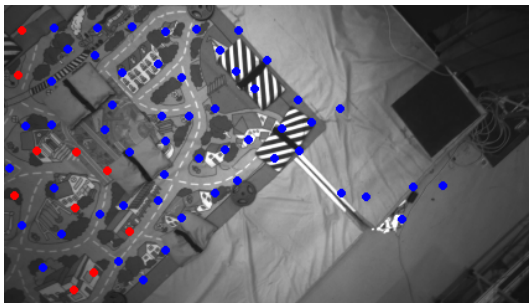


(d) Info gain - downward camera features

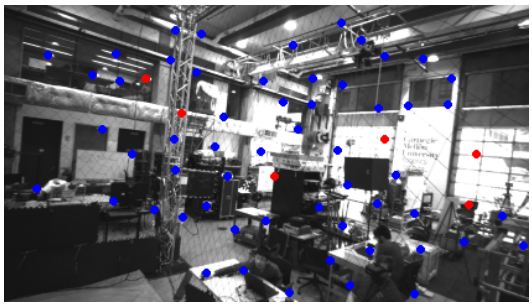
Figure 6.7: Distribution of features used in optimization between forward and downward camera during the motion capture flight at $t = 51.6$ s. Red dots denote features used in optimization, while blue dots denote tracked features ignored by optimization.



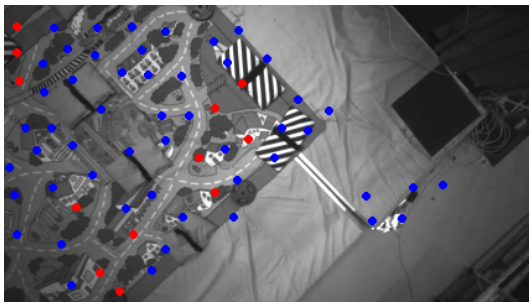
(a) Even split - forward camera features



(b) Even split - downward camera features



(c) Info gain - forward camera features

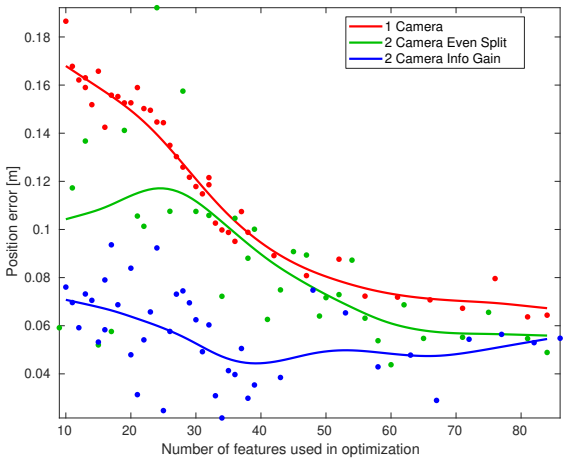


(d) Info gain - downward camera features

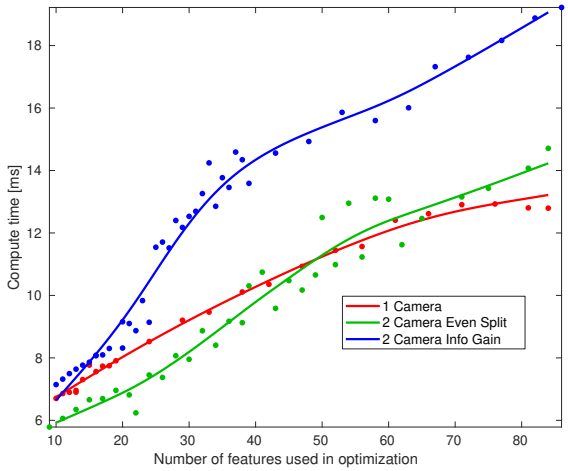
Figure 6.8: Distribution of features used in optimization between forward and downward camera during the motion capture flight at $t = 70.6$ s. Red dots denote features used in optimization, while blue dots denote tracked features ignored by optimization.

Compute Time and Estimation Accuracy

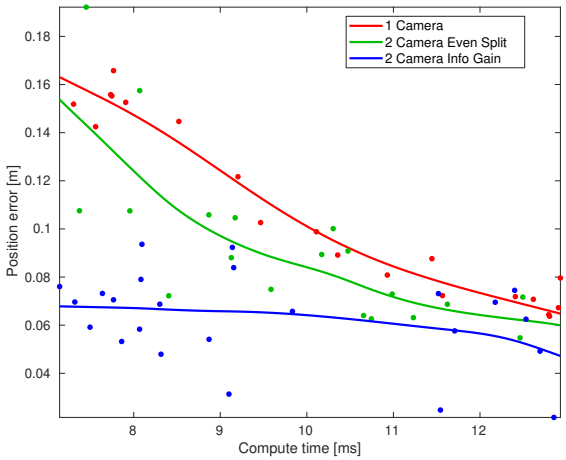
In this set of experiments, the total feature budget was varied from 10 to 90 and the motion capture dataset was postprocessed using each of the three feature selection strategies at each total feature budget value. Estimation accuracy is characterized using translational absolute trajectory error over the interval $45 \leq t \leq 80$ s, while compute time is the sum of optimization solve time and feature selection time. Feature selection time is negligible for the 1 camera and 2 camera even split strategies, but it is non-negligible for the 2 camera information gain strategy due to greedy feature selection (Sect. 6.2).



(a) Translational absolute trajectory error vs. number of features used in optimization



(b) Compute time vs. number of features used in optimization



(c) Translational absolute trajectory error vs. compute time

Figure 6.9: Relationships between estimation accuracy, number of features used in optimization, and compute time for the three feature selection methods over the time interval $45 \leq t \leq 80$ s on the motion capture dataset

Figure 6.9 summarizes the relationships between estimation accuracy, compute time, and the number of features used in optimization[4]. Each scatter plot data point represents a triplet of position error, number of features used in optimization, and compute time extracted from the $45 \leq t \leq 80$ s time interval of a postprocessing run performed with one of the three feature selection strategies. The number of features

[4]The number of features actually used in optimization is sometimes less than the total feature budget when not enough suitable features are tracked.

actually used in optimization and the compute time are both median values taken over all optimizations that occurred in the interval $45 \leq t \leq 80$ s.

Trendlines in Figure 6.9a show that over the entire range of total feature budgets, the 2 camera information gain strategy achieves lower position error than the 2 camera even split strategy, which in turn achieves lower position error than the single camera approach. Figure 6.9b shows the intuitive trend that compute time increases as the number of features used in optimization increases for all feature selection approaches. However, the greedy feature selection algorithm causes the 2 camera feature selection strategy to incur a higher time cost for any given number of features used in optimization. Despite this, the 2 camera feature selection strategy still achieves lower position error than the other strategies for any given amount of compute time (Fig. 6.9c). This advantage is greater for lower compute time limits for two reasons:

1. Greedy feature selection runs much faster on lower feature budgets.
2. There is less overlap between the set of features chosen by greedy feature selection and the set of features chosen by the heuristics used in the other approaches when the total feature budget is low.

The motion capture dataset postprocess results indicate that the proposed 2 camera information gain feature selection strategy achieves greater accuracy than standard feature selection strategies for a given compute budget. This section's results suggest that multi-camera information gain feature selection is an effective technique for resource-constrained visual-inertial state estimation.

6.4.2 Pavement

In this dataset the aerial robot flies at a low altitude above a sloped asphalt surface in an industrial loading dock and driveway area (Fig. 6.10). Position ground truth was obtained using a Leica TCRP 1205 total station to track a crystal mounted on top of the aerial robot. Trajectory estimation accuracy values reported in this section are translational absolute trajectory errors (H.13) obtained using the multiple position correspondences method (Appendix H.3.2). Each of the three feature selection strategies was run with a total feature budget of 30. Figure 6.11 depicts the aligned position estimates and ground truth over the entire trajectory.

Figure 6.10: Paved asphalt driveway and loading bay environment used for a multirotor data collection flight test

Figure 6.11 and the top plot of Figure 6.12 show that the trajectory begins with the vehicle far away from the origin. The vehicle quickly flies the approximately 40 m distance from its starting location back to the origin while losing altitude between $t = 5$ s and $t = 15$ s. Finally, the vehicle turns around and regains altitude between $t = 15$ s and $t = 25$ s.

Table 6.2 shows that the 2 camera information gain feature selection strategy achieves lower position

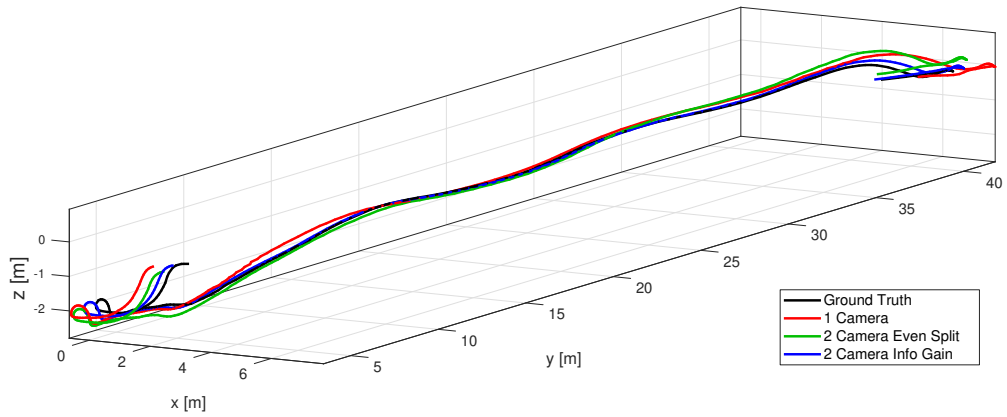


Figure 6.11: 3D plots of the trajectories estimated by VIO with different feature selection strategies for the pavement flight.

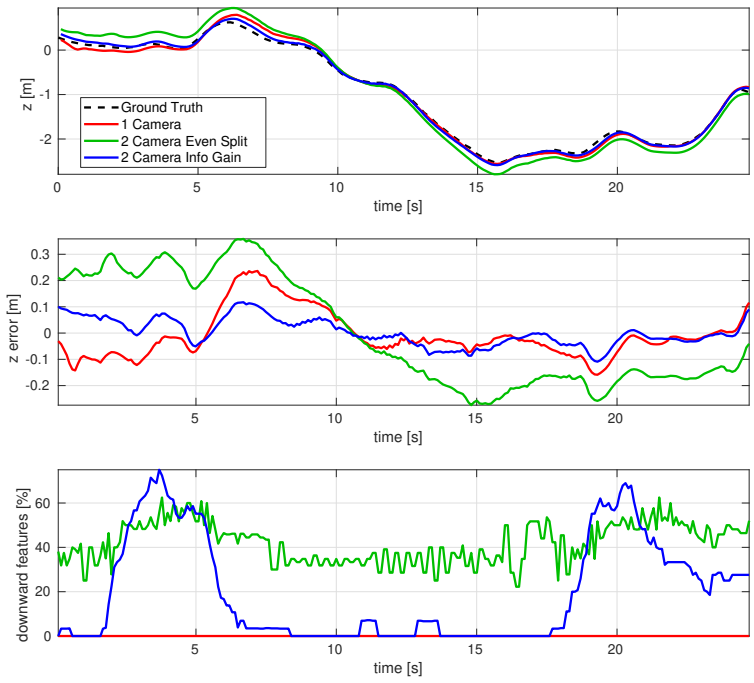


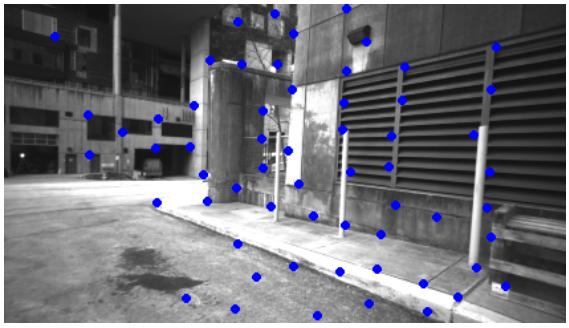
Figure 6.12: Top: Altitude estimates compared with ground truth. Middle: Altitude estimation error with respect to total station ground truth. Bottom: Percentage of the total feature budget allocated to the downward camera. The 2 camera information gain feature selection strategy achieves the lowest altitude estimation error because it changes its feature budget allocation between forward and downward cameras based on the current scene structure and predicted future motion.

error than the 2 camera even split strategy, which in turn achieves lower position error than the 1 camera approach. When the vehicle yaws 180 degrees at $t = 4$ s and $t = 20$ s, the 2 camera information gain strategy allocates a majority of its feature budget to the downward camera (bottom plot of Fig. 6.12) because those downward camera features remain in the field of view longer than forward camera features during in-place yaw maneuvers. At $t = 5$ s, the feature budget begins to heavily favor the forward camera as the vehicle simultaneously increases its forward translational velocity and descends. By $t = 10$ s, the entire feature budget has been allocated to the forward camera to avoid using downward camera frames that quickly exit the field of view when flying fast and close the ground. However, as soon as the vehicle stops moving forward ($t = 18$ s) and begins to rise while yawing, the feature budget swings back to favor the downward camera. Intelligently changing feature allocation between forward and downward cameras because on near-future motion predictions enables the information gain feature selection strategy to achieve lower position estimation error than the alternative methods.

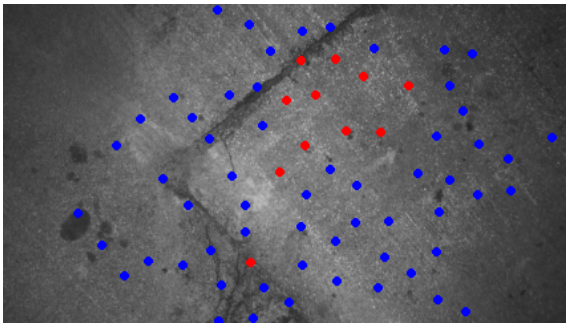
Table 6.2: Translational absolute trajectory errors of the 3 feature selection strategies on the pavement dataset

	1 Camera	**2 Camera Even Split**	**2 Camera Info Gain**
Absolute Trajectory Error (m)	0.93	0.46	0.34

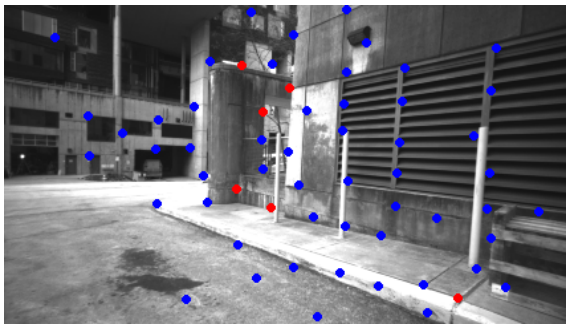
Figures 6.13-6.14 show the features selected for optimization by the 2 camera even spplit and information gain strategies at $t = 4.1$ s and $t = 7.6$ s.



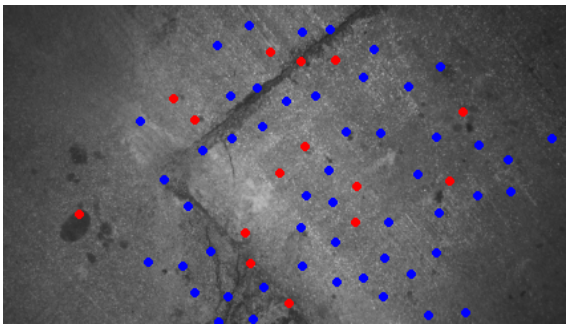
(a) Even split - forward camera features



(b) Even split - downward camera features



(c) Info gain - forward camera features

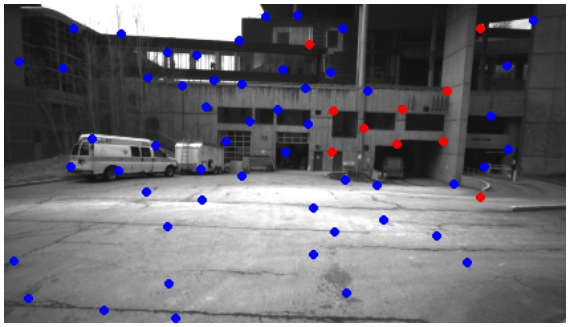


(d) Info gain - downward camera features

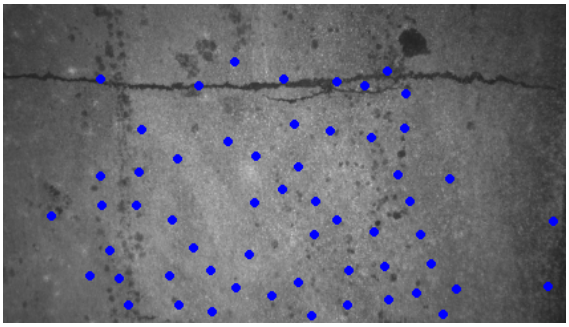
Figure 6.13: Distribution of features used in optimization between forward and downward camera on the over-pavement flight at $t = 4.1$ s. Red dots denote features used in optimization, while blue dots denote tracked features ignored by optimization.

When the vehicle is performing its initial in-place yaw at $t = 4.1$ s, the 2 camera information gain approach allocates more features to the downward camera than the 2 camera even split approach (compare

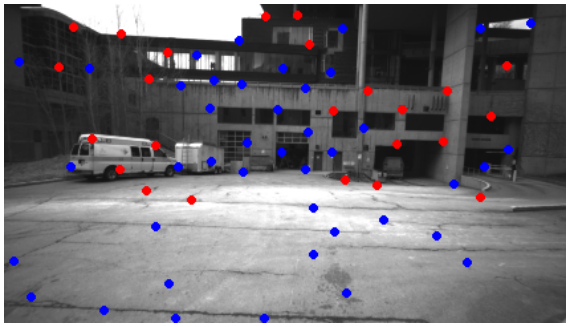
Fig. 6.13d with Fig. 6.13b). Even though the 2 camera even split approach allocates more features to the forward camera, its lack of consideration of future motion causes its forward camera features to not be visible in the current keyframes (Fig. 6.13a). In contrast, the 2 camera information gain approach has a greater number of forward camera features that are visible in the current keyframe even though it has allocated a lower feature budget for the forward camera (Fig. 6.13c).



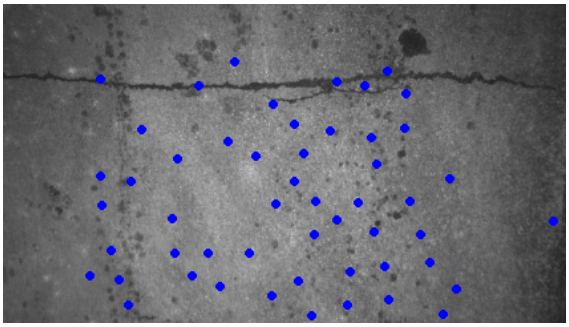
(a) Even split - forward camera features



(b) Even split - downward camera features



(c) Info gain - forward camera features



(d) Info gain - downward camera features

Figure 6.14: Distribution of features used in optimization between forward and downward camera on the over-pavement flight at t = 7.6 s. Red dots denote features used in optimization, while blue dots denote tracked features ignored by optimization.

When the vehicle begins to fly fast and lose altitude at $t = 7.6$ s, downward camera features used in optimization are not visible in the latest keyframes of any of the feature selection strategies due to high optical flow (compare Fig. 6.14b and Fig. 6.14d). However, the 2 camera information gain strategy achieves a better distribution of forward camera features with respect to field of view coverage (compare Fig. 6.14c with Fig. 6.14a) because it selects longer-lived features. Note that in this case, the 2 camera even split strategy has selected more features that have already exited the field of view due to heading change.

6.4.3 Volleyball Court

This dataset consists of six different flight trajectories around a bowl-shaped hillside surrounding a volleyball court (Fig. 6.15). Trajectory estimation accuracy was assessed using final position drift (Appendix H.4.2) because tracking with the total station was unreliable. Each of the three feature selection strategies was run with a total feature budget of 30 on each trajectory. Figure 6.16 depicts the unaligned position estimates over each trajectory. Figure 6.17 shows that the two camera information gain strategy achieves lower final position drift than both the two camera even split and single camera approaches. The superior performance of the proposed 2 camera information gain strategy on the varied trajectories in the volleyball court dataset indicate that it is a robust approach that improves estimation accuracy in multiple different flight conditions.

Figure 6.15: Volleyball court used for outdoor flight test data collection on a multirotor aerial robot

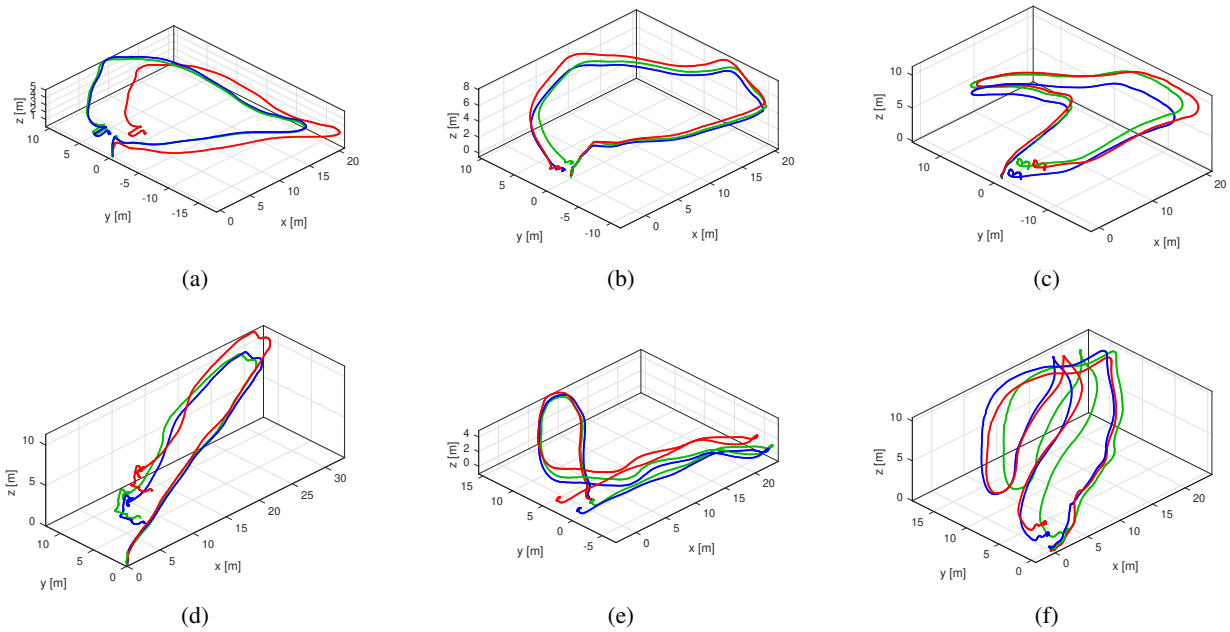


Figure 6.16: 3D plots of the trajectories estimated by VIO with different feature selection strategies for each of the 6 flight datasets taken around the volleyball court.

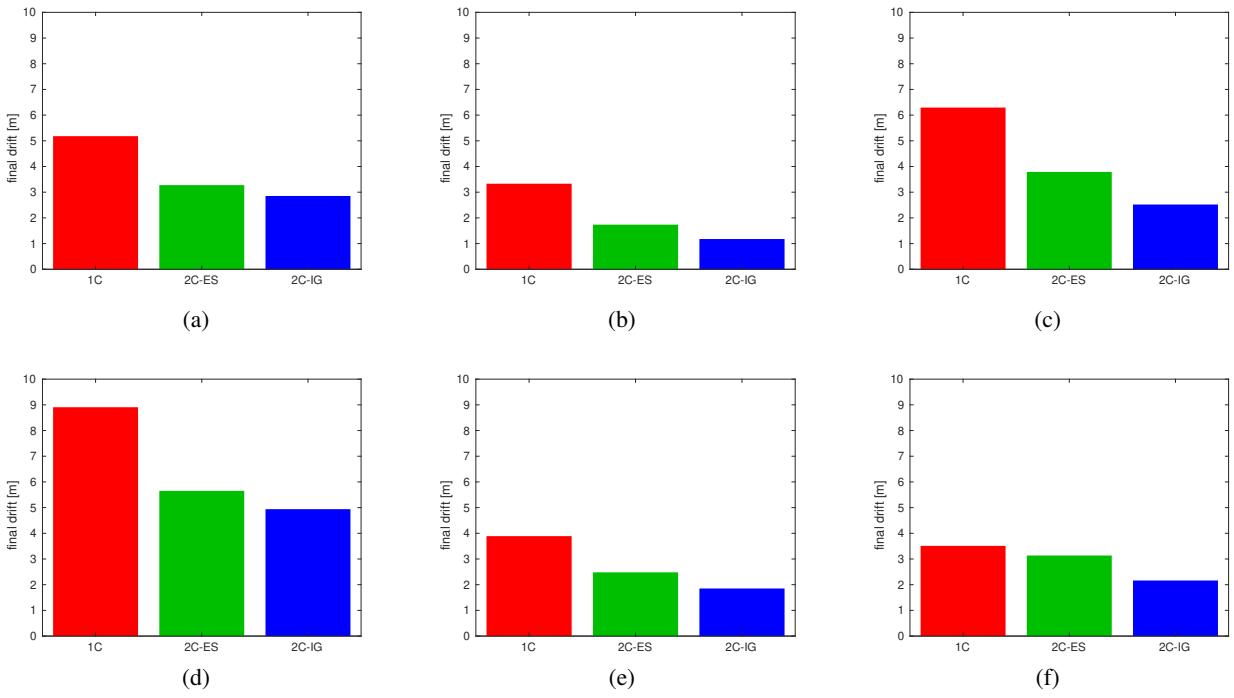


Figure 6.17: Final position drift of the 3 feature selection strategies for each of the 6 volleyball court trajectories

Chapter 7

Visual-Inertial-Depth Odometry

A major limitation of visual-inertial odometry systems is that they perform poorly in low light and repetitive visual texture. We propose to address this shortcoming by integrating depth observations into the visual-inertial odometry system described in Chapters 3-4. Depth sensors provide 3D metric information about the non-occluded regions of the environment within their fields of view in the form of range images or point clouds. Given a static scene, successive depth observations can be used to infer the sensor platform's motion. In order to exploit correspondences between depth and image data, we plan to use a depth sensor whose field of view exhibits a high degree of overlap with one of the regular cameras. Many consumer grade RGB-D sensors with hardware-level RGB-to-depth registration satisfy this requirement.

We use different strategies to incorporate depth observations into the primary and auxiliary estimators. The optimization-based primary estimator utilizes depth in a loosely-coupled manner by computing 6DOF transforms between keyframe depth images and including the resultant pseudo-observations as factors in a sliding window factor graph. The filter-based auxiliary estimator computes frame-to-frame transformations via 3D-to-2D feature correspondences between depth-augmented image features in the previous frame and regular image features in the current frame. Relative pose pseudo-observations are loosely combined with IMU observations via an Unscented Kalman filter.

7.1 Depth Sensor

We use a forward-facing Intel Realsense D435 RGB-D camera (henceforth referred to as Realsense) to provide depth observations to the state estimation framework. The Realsense contains a rolling shutter RGB camera and a global shutter depth sensor. The depth sensor has a 87 $\times$ 58 degree field of view and an advertised range of 0.1 to 10 m. It consists of a stereo pair of infrared cameras that use triangulation to compute depth images[1]. Passive sensing enables the Realsense to work in outdoor environments where ambient light can easily overwhelm projections of structured light patterns. If ambient illumination is insufficient, an onboard projector projects an infrared pattern in front of the stereo pair. The ability to use both active and passive sensing enables the Realsense to work in a variety of environments.

7.2 Relative Pose from Dense Depth Observations

We use Dhawale et al. [89]'s SLAM method for obtaining relative pose from dense depth images. Incoming depth images are simultaneously used to update a global map of the environment and localize the depth sensor with respect to the global map. In order to facilitate both efficient registration and high fidelity representation of the environment, the global map is a hierarchical structure where each level consists of multiple Gaussian distributions that represent raw depth observations.

[1] Stereo matching and triangulation are performed onboard the Realsense.

To compute the depth sensor's global 6DOF pose associated with an input depth image, an ICP-like strategy is used to align raw depth observations with the Gaussians in the hierarchical map representation. Unlike standard ICP, the cost function for rigid body alignment is a point to distribution distance rather than point to point distance. Global pose is computed via a nonlinear optimization problem whose objective function represents the log likelihood that raw observations are from the global map's Gaussian distributions. The covariance associated with the pose estimate is approximated as $\mathbf{\Sigma} \approx \mathbf{J}^{\mathrm{T}}\mathbf{J}$, where $\mathbf{J}$ is the Jacobian at the final iteration of the optimization. If a history of pose estimates corresponding to all received depth observations is stored as $\{\mathbf{T}_k\}_{k=1}^{n}$, the relative pose observation corresponding to depth sensor motion between times t_i and t_j is given by $\mathbf{T}_i^{-1}\mathbf{T}_j$.

7.3 Relative Pose from Depth-Aided 3D-to-2D Feature Correspondences

We use Zhang et al. [54]'s method for obtaining relative pose from 3D-to-2D feature correspondences over successive image frames. Consider grayscale image observations at times t_{k-1} and t_k, together with a set of tracked features contained within them. Let $\mathbf{h}_{i,k-1}$ be the normalized image coordinates of the i^{th} feature detected in image frame $k-1$ and $\mathbf{h}_{i,k}$ be its normalized image coordinates in image frame k. Let $d_{i,k-1}$ be the feature's depth with respect to the camera frame at the time t_{k-1}, and $d_{i,k}$ be its depth in the camera frame at time t_k. On a sensor with aligned depth and grayscale images, the value of d may be obtained by querying the depth image at the feature's pixel location within the grayscale image. Let $\mathbf{R} \equiv \mathbf{R}_{c_k,c_{k-1}}$ and $\mathbf{p} \equiv \mathbf{p}_{c_k,c_{k-1}}$ be the rotational and translational components of the rigid body transform that takes vectors in the previous image's camera frame $\{C_{k-1}\}$ to the current image's camera frame $\{C_k\}$. The relationship between $\mathbf{h}_{i,k-1}$ and $\mathbf{h}_{i,k}$ is

$$d_{i,k}\mathbf{h}_{i,k} = \mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1} + \mathbf{p} \tag{7.1}$$

Expand this equation into its three component rows, where $\mathbf{h}_{i,k} = [x_{i,k}\ y_{i,k}\ 1]^{\mathrm{T}}$:

$$d_{i,k}x_{i,k} = \mathbf{e}_1^{\mathrm{T}}\mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1} + \mathbf{e}_1^{\mathrm{T}}\mathbf{p} \tag{7.2}$$

$$d_{i,k}y_{i,k} = \mathbf{e}_2^{\mathrm{T}}\mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1} + \mathbf{e}_2^{\mathrm{T}}\mathbf{p} \tag{7.3}$$

$$d_{i,k} = \mathbf{e}_3^{\mathrm{T}}\mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1} + \mathbf{e}_3^{\mathrm{T}}\mathbf{p} \tag{7.4}$$

We define $\{\mathbf{e}_i\}_{i=1}^{3}$ as the columns of the 3×3 identity matrix

$$\mathbf{I} = \begin{bmatrix}\mathbf{e}_1 & \mathbf{e}_2 & \mathbf{e}_3\end{bmatrix} = \begin{bmatrix}1 & 0 & 0\\ 0 & 1 & 0\\ 0 & 0 & 1\end{bmatrix} \tag{7.5}$$

Multiply (7.4) by $x_{i,k}$, substitute it into the left hand side of (7.2), and rearrange terms

$$\left(\mathbf{e}_1^{\mathrm{T}} - x_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1} + \left(\mathbf{e}_1^{\mathrm{T}} - x_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{p} = 0 \tag{7.6}$$

Multiply (7.4) by $y_{i,k}$, substitute it into the left hand side of (7.3), and rearrange terms

$$\left(\mathbf{e}_2^{\mathrm{T}} - y_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1} + \left(\mathbf{e}_2^{\mathrm{T}} - y_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{p} = 0 \tag{7.7}$$

Note that (7.6)-(7.7) do not require the feature's depth in the current image frame, $d_{i,k}$.

The frame-to-frame 6DOF delta transform $\mathbf{p}, \mathbf{R}$ may be solved via nonlinear least squares optimization (Appendix C) of the objective function

$$L\left(\mathbf{p}, \mathbf{q}\right) = \frac{1}{2}\sum_i \left\|\mathbf{r}_i\left(\mathbf{p}, \mathbf{q}\right)\right\|^2 \tag{7.8}$$

where $\mathbf{r}_i$ consists of the stacked left hand sides of (7.6)-(7.7).

$$\mathbf{r}_i\left(\mathbf{p},\mathbf{q}\right)=\begin{bmatrix}\left(\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1}+\left(\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{p}\\ \left(\mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{R}d_{i,k-1}\mathbf{h}_{i,k-1}+\left(\mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{p}\end{bmatrix} \tag{7.9}$$

We derive the Jacobian for position by inspection

$$\frac{\partial\mathbf{r}_i}{\partial\mathbf{p}}=\begin{bmatrix}\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\\ \mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\end{bmatrix} \tag{7.10}$$

The Jacobian for the minimal coordinate rotational perturbation to $\mathbf{q}$ is derived by the perturbation method. Apply an additive perturbation $\delta\mathbf{r}_i$ to the left side of (7.9) and a rotational perturbation $\mathrm{Exp}\left(\delta\boldsymbol{\theta}\right)$ to $\mathbf{R}$ on the right side of (7.9).

$$\mathbf{r}_i+\delta\mathbf{r}_i=\begin{bmatrix}\left(\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{R}\left(\mathbf{I}+\left[\delta\boldsymbol{\theta}\right]_\times\right)d_{i,k-1}\mathbf{h}_{i,k-1}+\left(\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{p}\\ \left(\mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{R}\left(\mathbf{I}+\left[\delta\boldsymbol{\theta}\right]_\times\right)d_{i,k-1}\mathbf{h}_{i,k-1}+\left(\mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\right)\mathbf{p}\end{bmatrix} \tag{7.11}$$

We assume that the rotational perturbation is small, so we make the small angle approximation $\mathrm{Exp}\left(\delta\boldsymbol{\theta}\right)\approx\mathbf{I}+\left[\delta\boldsymbol{\theta}\right]_\times$. Subtract (7.9) from (7.11).

$$\delta\mathbf{r}_i=\begin{bmatrix}\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\\ \mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\end{bmatrix}\left[\delta\boldsymbol{\theta}\right]_\times d_{i,k-1}\mathbf{h}_{i,k-1} \tag{7.12}$$

Apply the cross product anti-symmetric property

$$\delta\mathbf{r}_i=-d_{i,k-1}\begin{bmatrix}\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\\ \mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\end{bmatrix}\left[\mathbf{h}_{i,k-1}\right]_\times\delta\boldsymbol{\theta} \tag{7.13}$$

The Jacobian with respect to the rotational perturbation is

$$\frac{\partial\mathbf{r}_i}{\partial\boldsymbol{\theta}}=-d_{i,k-1}\begin{bmatrix}\mathbf{e}_1^{\mathrm{T}}-x_{i,k}\mathbf{e}_3^{\mathrm{T}}\\ \mathbf{e}_2^{\mathrm{T}}-y_{i,k}\mathbf{e}_3^{\mathrm{T}}\end{bmatrix}\left[\mathbf{h}_{i,k-1}\right]_\times \tag{7.14}$$

7.4 Primary Estimator Modifications

Relative pose pseudo-observations from Sect. 7.2 are incorporated into the factor graph using a relative pose factor. The cost of the relative pose factor is added to the visual-inertial optimization objective function as the term

$$f_{\text{relpose}}(\mathbf{x})=\frac{1}{2}\left\|\mathbf{r}_{\text{relpose}}\left(\mathbf{x}_i,\mathbf{x}_j\right)\right\|_{\boldsymbol{\Sigma}}^2, \tag{7.15}$$

where $\mathbf{x}$ is the optimization vector, $\mathbf{x}_i$ is the motion state at keyframe i, $\mathbf{x}_j$ is the motion state at keyframe j, and $\boldsymbol{\Sigma}\in\mathbb{R}^{6\times6}$ is the relative pose covariance matrix. We select i to be the oldest sliding window keyframe and j to be the newest to enable the relative pose factor constrain drift over the longest possible time interval that can be optimized in the sliding window framework.

7.4.1 Relative Pose Residual

Under perfect sensing and estimation, the relationship between state variables and a relative pose observation, $\mathbf{T}_{\text{obs}}\in SE(3)$ is

$$\mathbf{T}_{\text{obs}}=\mathbf{T}_{bc}^{-1}\mathbf{T}_{wb_i}^{-1}\mathbf{T}_{wb_j}\mathbf{T}_{bc} \tag{7.16}$$

where $\mathbf{T}_{bc} \in SE(3)$ is the camera-to-body extrinsic transform and $\mathbf{T}_{wb_k} \in SE(3)$ is the estimated pose of keyframe k with respect to a world frame. Due to sensing and estimation errors, there is always a pose offset $\mathbf{T}_{\text{res}}$ between the observed and expected relative pose transform from keyframe i to keyframe j.

$$\mathbf{T}_{\text{res}} = \mathbf{T}_{\text{obs}}^{-1}\mathbf{T}_{bc}^{-1}\mathbf{T}_{wb_i}^{-1}\mathbf{T}_{wb_j}\mathbf{T}_{bc} \tag{7.17}$$

Expand (7.17)

$$\begin{bmatrix}\mathbf{R}_{\text{res}} & \mathbf{p}_{\text{res}} \\ \mathbf{0} & 1\end{bmatrix} = \begin{bmatrix}\mathbf{R}_{\text{obs}}^{\text{T}} & -\mathbf{R}_{\text{obs}}^{\text{T}}\mathbf{p}_{\text{obs}} \\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\mathbf{R}_{bc}^{\text{T}} & -\mathbf{R}_{bc}^{\text{T}}\mathbf{p}_{bc} \\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\mathbf{R}_{i}^{\text{T}} & -\mathbf{R}_{i}^{\text{T}}\mathbf{p}_{i} \\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\mathbf{R}_{j} & \mathbf{p}_{j} \\ \mathbf{0} & 1\end{bmatrix}\begin{bmatrix}\mathbf{R}_{bc} & \mathbf{p}_{bc} \\ \mathbf{0} & 1\end{bmatrix} \tag{7.18}$$

and write translational and rotational components separately:

$$\mathbf{r}_p \equiv \mathbf{p}_{\text{res}} = \mathbf{R}_{\text{obs}}^{\text{T}}\left\{\mathbf{R}_{bc}^{\text{T}}\left[\mathbf{R}_{i}^{\text{T}}\left(\mathbf{R}_j\mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i\right) - \mathbf{p}_{bc}\right] - \mathbf{p}_{\text{obs}}\right\} \tag{7.19}$$

$$\text{Exp}(\mathbf{r}_\theta) \equiv \mathbf{R}_{\text{res}} = \mathbf{R}_{\text{obs}}^{\text{T}}\mathbf{R}_{bc}^{\text{T}}\mathbf{R}_{i}^{\text{T}}\mathbf{R}_j\mathbf{R}_{bc} \tag{7.20}$$

The translational component of the relative pose residual, $\mathbf{r}_p \in \mathbb{R}^3$, is given by (7.19). The rotational component of the relative pose residual is given by the capitalized log map of (7.20).

$$\mathbf{r}_\theta = \text{Log}\left(\mathbf{R}_{\text{obs}}^{\text{T}}\mathbf{R}_{bc}^{\text{T}}\mathbf{R}_{i}^{\text{T}}\mathbf{R}_j\mathbf{R}_{bc}\right) \in \mathbb{R}^3 \tag{7.21}$$

7.4.2 Relative Pose Jacobians

This section derives the nonzero Jacobians of the relative pose factor's residuals.

Vector Space Jacobians

The Jacobians of the translational residual with respect to vector space parameter blocks $\mathbf{p}_i$ and $\mathbf{p}_j$ are

$$\frac{\partial \mathbf{r}_p}{\partial \mathbf{p}_i} = -\mathbf{R}_{\text{obs}}^{\text{T}}\mathbf{R}_{bc}^{\text{T}}\mathbf{R}_{i}^{\text{T}} \tag{7.22}$$

$$\frac{\partial \mathbf{r}_p}{\partial \mathbf{p}_j} = \mathbf{R}_{\text{obs}}^{\text{T}}\mathbf{R}_{bc}^{\text{T}}\mathbf{R}_{i}^{\text{T}} \tag{7.23}$$

Manifold Jacobians

Instead of taking Jacobians with respect to the quaternions $\mathbf{q}_i$ and $\mathbf{q}_j$, we take Jacobians with respect to the minimal parameterizations of their perturbations, $\delta\boldsymbol{\theta}_i$ and $\delta\boldsymbol{\theta}_j$.

We use the perturbation method to determine the Jacobian of $\mathbf{r}_p$ with respect to $\delta\boldsymbol{\theta}_i$ and $\delta\boldsymbol{\theta}_j$. Apply an additive perturbation to the left hand side of (7.19) and rotational perturbations to the right hand side.

$$\mathbf{r}_p + \delta\mathbf{r}_p = \mathbf{R}_{\text{obs}}^{\text{T}}\left\{\mathbf{R}_{bc}^{\text{T}}\left[\left(\mathbf{R}_i\,\text{Exp}(\delta\boldsymbol{\theta}_i)\right)^{\text{T}}\left(\mathbf{R}_j\,\text{Exp}(\delta\boldsymbol{\theta}_j)\mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i\right) - \mathbf{p}_{bc}\right] - \mathbf{p}_{\text{obs}}\right\} \tag{7.24}$$

Assuming the rotational perturbations $\delta\boldsymbol{\theta}_i$ and $\delta\boldsymbol{\theta}_j$ are small, use the approximation (A.14).

$$\mathbf{r}_p + \delta\mathbf{r}_p \approx \mathbf{R}_{\text{obs}}^{\text{T}}\left\{\mathbf{R}_{bc}^{\text{T}}\left[\left(\mathbf{R}_i\left(\mathbf{I} + [\delta\boldsymbol{\theta}_i]_\times\right)\right)^{\text{T}}\left(\mathbf{R}_j\left(\mathbf{I} + [\delta\boldsymbol{\theta}_j]_\times\right)\mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i\right) - \mathbf{p}_{bc}\right] - \mathbf{p}_{\text{obs}}\right\} \tag{7.25}$$

$$\mathbf{r}_p + \delta\mathbf{r}_p \overset{(A.4)}{\approx} \mathbf{R}_{\text{obs}}^{\text{T}}\left\{\mathbf{R}_{bc}^{\text{T}}\left[\left(\mathbf{I} - [\delta\boldsymbol{\theta}_i]_\times\right)\mathbf{R}_i^{\text{T}}\left(\mathbf{R}_j\mathbf{p}_{bc} + [\delta\boldsymbol{\theta}_j]_\times\mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i\right) - \mathbf{p}_{bc}\right] - \mathbf{p}_{\text{obs}}\right\} \tag{7.26}$$

$$\mathbf{r}_p + \delta\mathbf{r}_p \approx \mathbf{R}_{\text{obs}}^{\text{T}}\left\{\mathbf{R}_{bc}^{\text{T}}\left[\mathbf{R}_i^{\text{T}}\left(\mathbf{R}_j\mathbf{p}_{bc} + [\delta\boldsymbol{\theta}_j]_\times\mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i\right) - [\delta\boldsymbol{\theta}_i]_\times\mathbf{R}_i^{\text{T}}\left(\mathbf{R}_j\mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i\right) - \mathbf{p}_{bc}\right] - \mathbf{p}_{\text{obs}}\right\} \tag{7.27}$$

$$\delta\mathbf{r}_p \overset{(7.19)}{\approx} \mathbf{R}_{\text{obs}}^{\text{T}}\left\{\mathbf{R}_{bc}^{\text{T}}\left[\mathbf{R}_i^{\text{T}}\left[\delta\boldsymbol{\theta}_j\right]_\times\mathbf{p}_{bc} - [\delta\boldsymbol{\theta}_i]_\times\mathbf{R}_i^{\text{T}}\left(\mathbf{R}_j\mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i\right) - \mathbf{p}_{bc}\right] - \mathbf{p}_{\text{obs}}\right\} \tag{7.28}$$

$$\delta\mathbf{r}_p \overset{(A.3)}{\approx} \mathbf{R}_{\text{obs}}^{\text{T}} \left\{ \mathbf{R}_{bc}^{\text{T}} \left[-\mathbf{R}_i^{\text{T}} \left[\mathbf{p}_{bc} \right]_\times \delta\boldsymbol{\theta}_j + \left[\mathbf{R}_i^{\text{T}} \left(\mathbf{R}_j \mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i \right) \right]_\times \delta\boldsymbol{\theta}_i - \mathbf{p}_{bc} \right] - \mathbf{p}_{\text{obs}} \right\} \tag{7.29}$$

$$\delta\mathbf{r}_p \approx -\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_i^{\text{T}} \left[\mathbf{p}_{bc} \right]_\times \delta\boldsymbol{\theta}_j + \mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \left[\mathbf{R}_i^{\text{T}} \left(\mathbf{R}_j \mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i \right) \right]_\times \delta\boldsymbol{\theta}_i + const \tag{7.30}$$

The approximate Jacobians of the translational residual with respect to rotational parameter blocks are

$$\frac{\partial \mathbf{r}_p}{\partial \delta\boldsymbol{\theta}_i} \approx \mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \left[\mathbf{R}_i^{\text{T}} \left(\mathbf{R}_j \mathbf{p}_{bc} + \mathbf{p}_j - \mathbf{p}_i \right) \right]_\times \tag{7.31}$$

$$\frac{\partial \mathbf{r}_p}{\partial \delta\boldsymbol{\theta}_j} \approx -\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_i^{\text{T}} \left[\mathbf{p}_{bc} \right]_\times \tag{7.32}$$

We use the perturbation method to derive the Jacobians of $\mathbf{r}_\theta$ with respect to $\delta\boldsymbol{\theta}_i$ and $\delta\boldsymbol{\theta}_j$. Apply an additive perturbation to the left hand side of (7.21) and rotational perturbations to the right hand side.

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta = \text{Log} \left[\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \left(\mathbf{R}_i \, \text{Exp}(\delta\boldsymbol{\theta}_i) \right)^{\text{T}} \mathbf{R}_j \, \text{Exp}(\delta\boldsymbol{\theta}_j) \mathbf{R}_{bc} \right] \tag{7.33}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.13)}{\approx} \text{Log} \left[\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \, \text{Exp}(-\delta\boldsymbol{\theta}_i) \mathbf{R}_i^{\text{T}} \mathbf{R}_j \, \text{Exp}(\delta\boldsymbol{\theta}_j) \mathbf{R}_{bc} \right] \tag{7.34}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.46)}{\approx} \text{Log} \left[\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_i^{\text{T}} \mathbf{R}_j \, \text{Exp}(-\mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i) \, \text{Exp}(\delta\boldsymbol{\theta}_j) \mathbf{R}_{bc} \right] \tag{7.35}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.46)}{\approx} \text{Log} \left[\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_i^{\text{T}} \mathbf{R}_j \, \text{Exp}(-\mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i) \mathbf{R}_{bc} \, \text{Exp}(\mathbf{R}_{bc}^{\text{T}} \delta\boldsymbol{\theta}_j) \right] \tag{7.36}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.46)}{\approx} \text{Log} \left[\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_i^{\text{T}} \mathbf{R}_j \mathbf{R}_{bc} \, \text{Exp}(-\mathbf{R}_{bc}^{\text{T}} \mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i) \, \text{Exp}(\mathbf{R}_{bc}^{\text{T}} \delta\boldsymbol{\theta}_j) \right] \tag{7.37}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.49)}{\approx} \text{Log} \left[\mathbf{R}_{\text{obs}}^{\text{T}} \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_i^{\text{T}} \mathbf{R}_j \mathbf{R}_{bc} \, \text{Exp}(-\mathbf{R}_{bc}^{\text{T}} \mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i + \mathbf{R}_{bc}^{\text{T}} \delta\boldsymbol{\theta}_j) \right] \tag{7.38}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(7.21)}{\approx} \text{Log} \left[\text{Exp}(\mathbf{r}_\theta) \, \text{Exp}(-\mathbf{R}_{bc}^{\text{T}} \mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i + \mathbf{R}_{bc}^{\text{T}} \delta\boldsymbol{\theta}_j) \right] \tag{7.39}$$

$$\mathbf{r}_\theta + \delta\mathbf{r}_\theta \overset{(A.48)}{\approx} \mathbf{r}_\theta + \mathbf{J}_r^{-1}(\mathbf{r}_\theta) \left[-\mathbf{R}_{bc}^{\text{T}} \mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i + \mathbf{R}_{bc}^{\text{T}} \delta\boldsymbol{\theta}_j \right] \tag{7.40}$$

$$\delta\mathbf{r}_\theta \approx \mathbf{J}_r^{-1}(\mathbf{r}_\theta) \left[-\mathbf{R}_{bc}^{\text{T}} \mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i + \mathbf{R}_{bc}^{\text{T}} \delta\boldsymbol{\theta}_j \right] \tag{7.41}$$

Apply the small angle approximation $\mathbf{J}_r^{-1}(\boldsymbol{\phi}) \approx \mathbf{I} + \frac{1}{2} \left[\boldsymbol{\phi} \right]_\times$ for the inverse of the right Jacobian of $SO(3)$.

$$\delta\mathbf{r}_\theta \approx -\left(\mathbf{I} + \frac{1}{2} \left[\mathbf{r}_\theta \right]_\times \right) \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_j^{\text{T}} \mathbf{R}_i \delta\boldsymbol{\theta}_i + \left(\mathbf{I} + \frac{1}{2} \left[\mathbf{r}_\theta \right]_\times \right) \mathbf{R}_{bc}^{\text{T}} \delta\boldsymbol{\theta}_j \tag{7.42}$$

The approximate Jacobians of the rotational residual with respect to rotational parameter blocks are

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta\boldsymbol{\theta}_i} \approx -\left(\mathbf{I} + \frac{1}{2} \left[\mathbf{r}_\theta \right]_\times \right) \mathbf{R}_{bc}^{\text{T}} \mathbf{R}_j^{\text{T}} \mathbf{R}_i \tag{7.43}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta\boldsymbol{\theta}_j} \approx \left(\mathbf{I} + \frac{1}{2} \left[\mathbf{r}_\theta \right]_\times \right) \mathbf{R}_{bc}^{\text{T}} \tag{7.44}$$

7.5 Auxiliary Estimator Modifications

In contrast with the original auxiliary estimator (Sect. 4.3, Fig. 4.4), the depth-aided auxiliary estimator does not require a rangefinder and instead uses a depth sensor with aligned grayscale and depth image streams. The horizontal velocity pseudo-observation computed via frame-to-frame planar homography (Sect. 4.3.1) is replaced with a 3D frame-to-frame velocity using depth-aided point correspondences (Sect. 7.3).

To obtain planar frame velocity from camera frame relative pose, it is first necessary to compute camera frame velocity by dividing the relative pose's translational component ($\mathbf{p}$) by the elapsed time between frames.

$$\mathbf{v}_c = \frac{\mathbf{p}}{t_k - t_{k-1}} \tag{7.45}$$

Next, compute the body frame velocity, $\mathbf{v}_b$, associated with $\mathbf{v}_c$ using (4.33). Finally, the level frame velocity, $\mathbf{v}_l$, is obtained by compensating for the roll and pitch of the current body frame.

$$\mathbf{v}_l = \mathbf{R}_y(\theta)\mathbf{R}_x(\phi)\mathbf{v}_b \tag{7.46}$$

Note that unlike in (4.34), the z-component of the level velocity is kept.

The UKF logic is the same as in the original auxiliary estimator formulation, except that the velocity correction model (4.47)-(4.48) is extended to the third dimension.

$$\mathbf{z}_v = \mathbf{v}_l + \mathbf{n}_v \tag{7.47}$$

$$\mathbf{n}_v \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} s_{vx}^2 & 0 & 0 \\ 0 & s_{vy}^2 & 0 \\ 0 & 0 & s_{vz}^2 \end{bmatrix}\right) \tag{7.48}$$

Note that the altitude observation model (Sect. 4.3.2) may still be used alongside the depth-aided frame-to-frame velocity observation model.

7.6 Experimental Results

This section presents results that demonstrate the benefit of the depth-related modifications described in Sections 7.4 and 7.5. Visual, inertial, and depth observations collected on a multirotor aerial robot were used to run sliding window visual-inertial odometry and visual-inertial-depth odometry. Utilizing depth observations enables the primary estimator to provide estimates in environments with poor feature tracking and increases the auxiliary estimator's accuracy. All results were generated via postprocessing on a Lenovo Thinkpad T470p laptop[2] running Ubuntu 16.04 LTS.

7.6.1 Primary Estimator

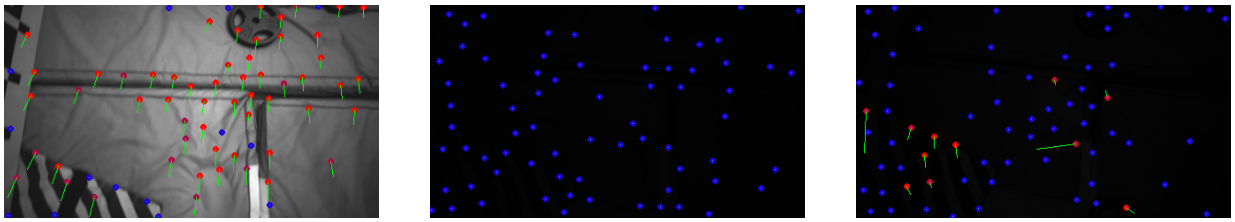
In this subsection, the nominal primary estimator is denoted as *visual-inertial* and the modified primary estimator is denoted as *visual-inertial-depth*.

Motion Capture

We evaluated the modified primary estimator (Sect. 7.4) against ground truth and the original primary estimator (Chapter 3) on a dataset obtained inside a motion capture arena. The dataset is comprised of forward depth images, downward grayscale images, and IMU observations. Ambient illumination is drastically reduced roughly in the middle of the dataset via turning off indoor lights to simulate a transition between light and dark environments.

Figure 7.1 visualizes feature tracking in the downward camera shortly before and after the lights turn off. The sudden change in image brightness causes optical flow-based tracking methods to fail and feature trails to be lost, which significantly reduces the number of feature trails used in the optimization problem (Fig. 7.2 top). With almost no reprojection factors in the sliding window optimization after the lights turn off, the nominal primary estimator's position estimate diverges because it no longer has any motion constraints to counteract IMU preintegration factors (Fig. 7.2 bottom). On the other hand, the modified primary estimator is able to mitigate position drift by relying on depth-derived 6DOF relative pose motion constraints when vision-only reprojection factors become unavailable. Figure 7.3 shows that both the nominal and modified primary estimators have comparable performance until the lights turn off, after which the former diverges much more than the latter. Despite its superior performance, the modified primary estimator still has non-trivial position drift (especially in the vertical direction) due to degeneracies in scene geometry.

[2]Intel Core i7-7820HQ at 2.9 GHz × 8, 32 GB RAM



(a) 150 ms before lights off (b) 50 ms after lights off (c) 250 ms after lights off

Figure 7.1: Downward camera feature tracking images before and after the motion capture arena's ambient lights are turned off. Red denotes denote long-lived feature trails while blue dots denote newly detected features. A sudden decrease in illumination causes all feature trails to be lost (Fig. 7.1b). Afterwards, features are much fewer in number and more likely to be incorrectly tracked (Fig. 7.1c).

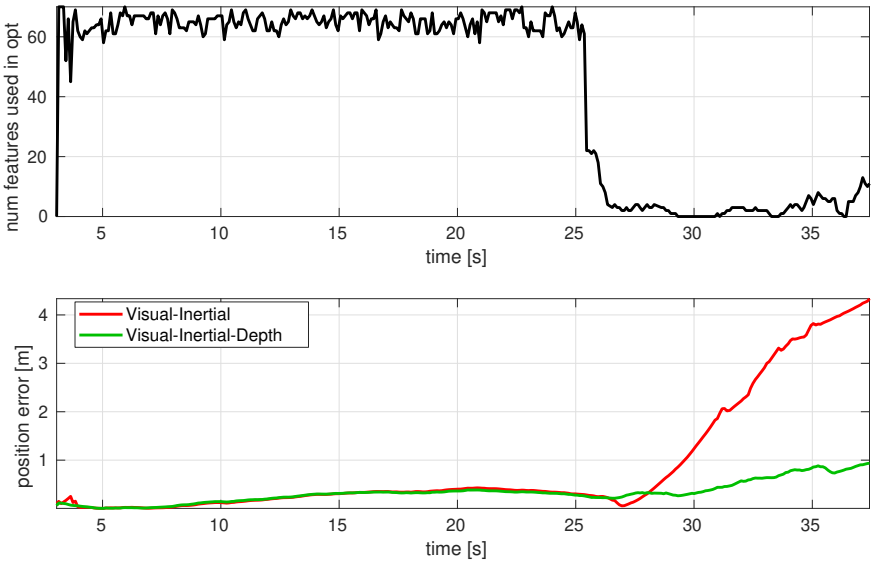


Figure 7.2: Top: Number of features used in optimization vs. time. The lights turn off at approximately $t = 25$ s, after which the number of features drops sharply. Bottom: The position error of visual-inertial odometry increases significantly after the lights turn off due to a lack of features. However, the position error of visual-inertial-depth odometry does not increase because depth observations still provide motion constraints in the dark.

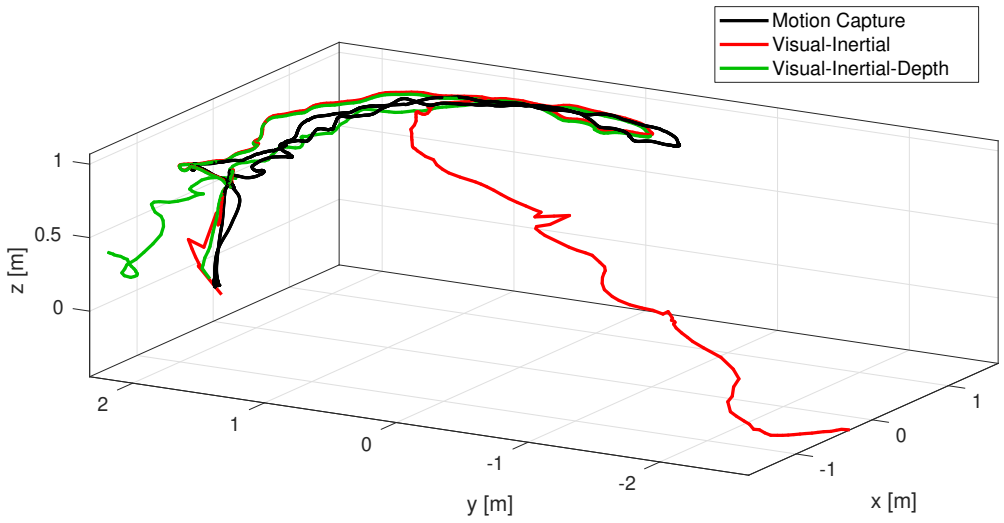


Figure 7.3: A comparison of visual-inertial odometry and visual-inertial-depth odometry on a dataset where the illumination decreases significantly.

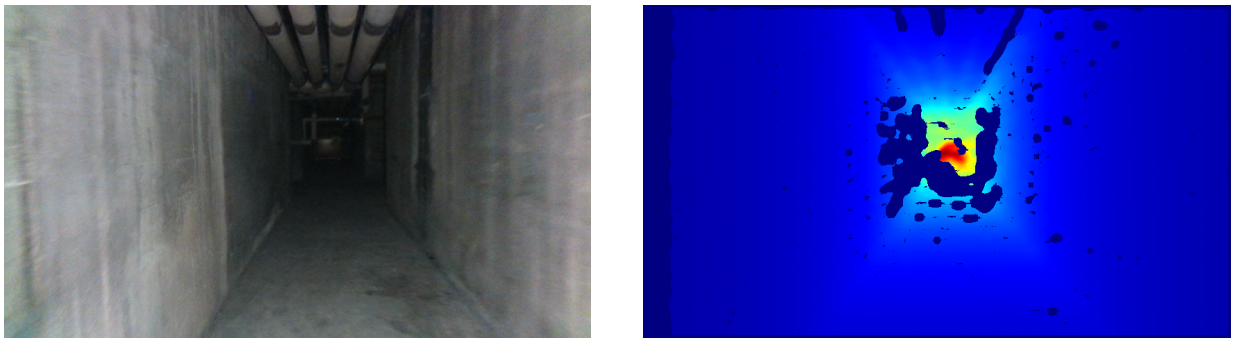
Industrial Tunnels

We evaluated the modified primary estimator on a dataset where an aerial robot flies in a loop through four connected industrial tunnels (Fig. 7.4) that together form a square. Although the ambient lighting in the environment is low, the vehicle's onboard light provides sufficient illumination for successful downward camera feature tracking.

Figure 7.4: An aerial robot equipped with an onboard light flies around a corner in an industrial tunnel environment.

Figure 7.5 depicts the forward-facing realsense's color and depth images when the vehicle travels along the last of the four tunnels it encounters. The color image shows that the side walls and floor are smooth concrete, while four pipes run in an axial direction along the ceiling. This effectively provides the final tunnel with a constant cross section, which results in a geometric degeneracy along the corridor's longitudinal axis. The geometric degeneracy arises because the depth image (Fig. 7.5b) remains the same when the camera

moves forward or backward along the $x = 0$ m corridor in Figure 7.6.



(a) Color image (b) Depth image

Figure 7.5: The majority of the forward-facing depth image remains constant as the vehicle flies along the last corridor because it has a mostly constant cross-section. Consequently, depth-based relative pose observations yield motion constraints that underestimate the vehicle's translation along the corridor axis.

As a consequence of the geometric degeneracy, relative pose observations computed by alignment of depth images against a local map may differ significantly from the sensor's true motion. Incorporating such incorrect relative pose observations in the sliding window optimization leads to high position drift, as illustrated by the L-shaped segment at the end of the visual-inertial-depth trajectory estimate from (-1,3) m to (2,8) m in Figure 7.6. In this example, incorporating depth-based relative pose observations in a geometrically degenerate environment leads to underestimation of translation in the direction of the degeneracy. On the other hand, avoiding depth observations entirely enables the final segment of the trajectory to align with the initial segment along the $y = 12$ m corridor in Figure 7.6.

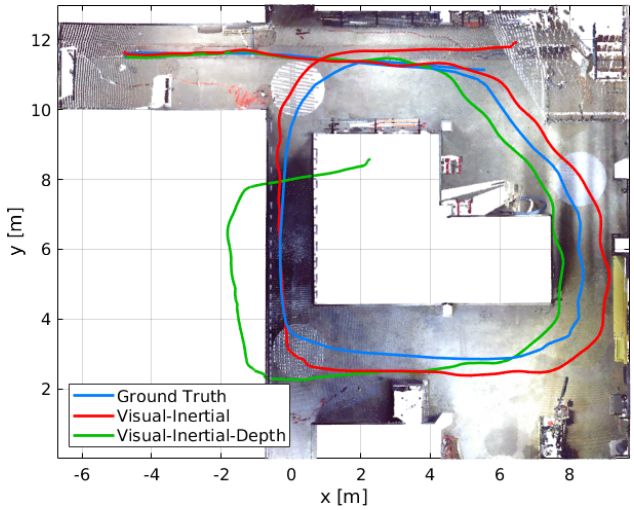


Figure 7.6: A comparison of visual-inertial odometry and visual-inertial-depth odometry on a flight through a square-shaped collection of industrial tunnels. The starting point is $(-4.7, 11.6)$ m. The trajectory estimates are aligned against a colorized point cloud of the industrial tunnel environment formed by stitching multiple FARO scans together. Ground truth is obtained by manually matching raw depth sensor scans against the composite FARO point cloud.

The motion capture and industrial tunnels results indicate that while depth-based relative pose observations are critical for avoiding motion estimate divergence in poor illumination conditions, they also degrade the motion estimate when the surrounding environment is geometrically degenerate. Therefore, techniques

that incorporate depth observations in sliding window optimization should weight them with covariances that represent the level of geometric degeneracy in the environment.

7.6.2 Auxiliary Estimator

We evaluated the modified auxiliary estimator (Sect. 7.5) against ground truth and the original auxiliary estimator (Sect. 4.3) on a dataset obtained inside a motion capture arena. The dataset was collected on a platform with aligned forward realsense grayscale and depth images, downward grayscale images, IMU, and downward rangefinder observations. Both cameras' fields of view contain rich visual texture, while the forward camera's field of view also contains non-degenerate geometry (Fig. 7.7).

(a) Forward camera image (b) Forward depth image (c) Downward camera image

Figure 7.7: Example images from the realsense forward camera and depth sensors (left and middle) and the mvBluefox downward camera. The realsense's depth sensor is aligned with the forward-facing left infrared sensor, which is used as the forward camera. The aligned depth image serves as a lookup table for depth values of features at arbitrary pixel locations in the forward camera image.

Figure 7.8 depicts the navigation frame[3] horizontal velocities of the original and modified auxiliary estimator. Vertical velocity is not compared because it is not directly observed in the original auxiliary estimator formulation. Using 3D-to-2D feature correspondences (Sect. 7.5) between successive frames produces a velocity estimate with much less noise than using frame-to-frame planar homographies scaled by a downward rangefinder (Sect. 4.3). While the former method incorporates depth information for every tracked feature, the latter method uses a single depth value from the rangefinder and assumes that all features have that depth. Because downward camera features only satisfy this assumption if the downward camera's optical axis is parallel to the normal of the planar scene it is viewing, the homography-based method yields noisier velocity estimates. The consequence of this noise difference is that depth-aided navigation frame pseudo-observations may be used in the UKF with lower correction noise than their homography-based counterparts, leading to more accurate translation estimation during takeoff and emergency takeover scenarios.

[3]The navigation frame is a level reference frame whose x-axis is aligned with the instantaneous heading and z-axis is antiparallel to the global gravity vector.

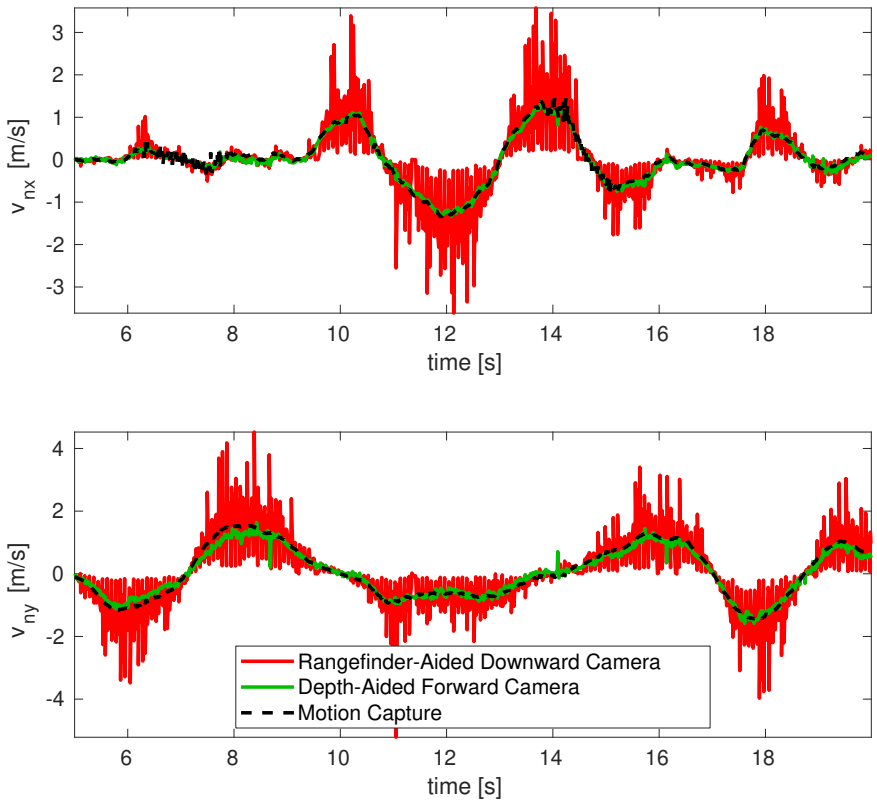


Figure 7.8: Navigation frame horizontal velocity estimates using the rangefinder-aided downward camera frame-to-frame planar homography (red) vs. depth-aided forward camera 3D-to-2D frame-to-frame feature correspondences (green).

Chapter 8

Conclusion

This thesis addresses the problem of achieving accurate, reliable state estimation with multiple onboard sensors on computationally constrained robotic platforms. Visual-inertial odometry achieves efficient motion estimation in many common flight scenarios, but struggles under visually degraded conditions such as non-constant lighting, low brightness, and repetitive texture. These shortcomings may be mitigated with the introduction of observations from additional cameras and depth sensors at the cost of increased computation that may not be available on size, weight, and power-constrained platforms. This thesis proposes a set of state estimation strategies to

1. Efficiently incorporate observations from multiple asynchronous cameras into an optimization-based sliding window visual-inertial odometry algorithm
2. Improve estimation accuracy given a fixed computational budget by selecting features that are predicted to be highly informative over a near term time horizon
3. Leverage depth observations to enable operation in visually degraded environments with sufficient geometric saliency

8.1 Summary of Contributions

Chapters 3-7 describe methodologies that achieve these objectives. Thus, the contributions of this thesis are:

- **Computationally efficient sliding window visual-inertial odometry with multiple asynchronous cameras:** A tightly-coupled optimization-based state estimator fuses IMU and sparse feature observations from multiple asynchronous cameras. Image plane feature trail interpolation enables features from secondary camera keyframes that are not synchronized with master camera keyframes to be incorporated into the optimization problem without increasing the number of keyframes in the sliding window. This lowers the computational cost associated with incorporating more diverse visual observations that increase estimation accuracy. (Chapters 3-5)
- **Sensor resource allocation for computationally constrained visual-inertial odometry:** The information value of visible features is predicted using a short-term motion model, which enables the highest value features to be selected greedily. Knowledge of a platform's future motion drives the selection of longer-lived feature trails that improve estimation accuracy for a given feature limit or computational footprint. (Chapter 6)
- **Depth-aided visual-inertial odometry:** Depth observations are incorporated into a visual-inertial odometry framework as relative pose constraints between keyframes that are efficiently computed via an efficient local occupancy map representation. The use of depth information facilitates smooth state estimation in visually degraded but geometrically salient environments. (Chapter 7)

Figure 8.1 shows the overall state estimation system together with the information flow between the

modules described in this thesis.

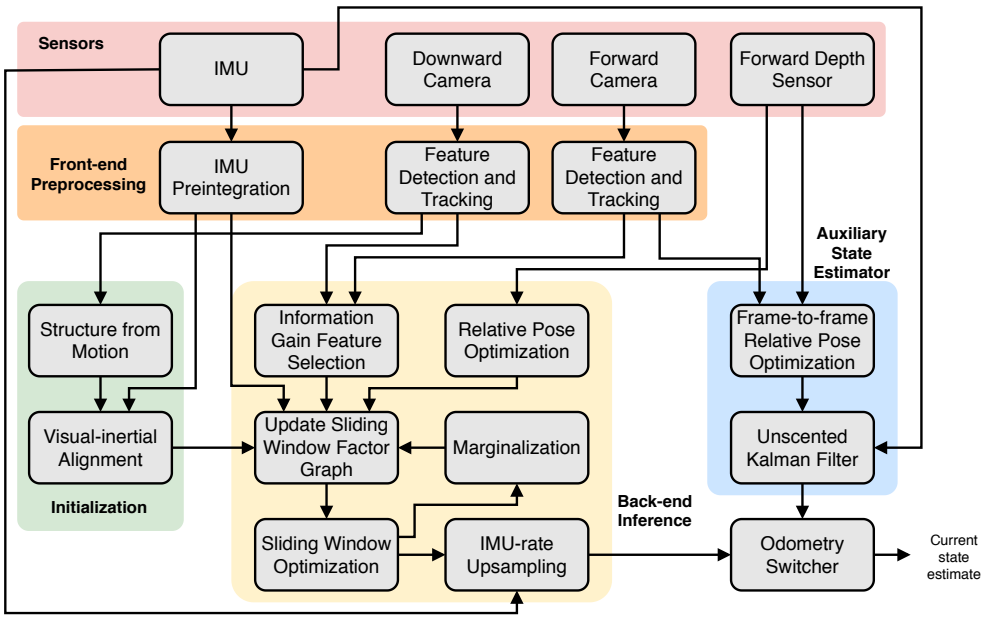


Figure 8.1: Information flow diagram of proposed state estimation system

Table 8.1 summarizes the results achieved with the techniques described in this thesis. Taken as a whole, the results indicate that the proposed methodologies are able to achieve accurate and computationally efficient state estimation in varying environmental conditions.

Table 8.1: Summary of thesis results

Result	Source
Baseline VIO has equal or better performance than VINS-Mono on all EuRoC datasets	Table 3.1
Baseline VIO achieves 3% position drift on 167 m forest and treetop flight	Section 3.9.2
Baseline VIO achieves 1% position drift on 46 m cave flight	Section 3.9.3
Multi-camera VIO reduces position and yaw drift over single camera VIO in parking garage dataset	Section 5.4.2
Information gain feature selection halves position estimation error compared to even split feature selection for a compute time of 7 ms per optimization on a motion capture dataset	Figure 6.9c
Information gain feature selection reduces position estimation error by 25% compared with even split feature selection on pavement dataset	Table 6.2
Information gain feature selection reduces final position drift over even split and single camera approaches on 6 volleyball court flight datasets	Figure 6.17
Dense depth-based relative pose factors prevent estimator divergence in dark conditions	Table 7.3

8.2 Future Work

Several avenues for future research build upon the approaches presented in this thesis. While this work describes separate strategies to efficiently select visual feature observations and incorporate depth observations, further performance benefits may be achieved by more tightly integrating depth observations with

the proposed visual-inertial odometry framework. For example, sparse depth observations can improve the primary estimator's initialization routine by enabling metric scale estimation during structure from motion (Sect. 3.7.1). By using depth observations instead of IMU accelerometer observations to estimate scale, it becomes possible to estimate acceleration bias independently of scale in the initialization routine. Another direction of research would be to extend the sensor resource allocation method described in Chapter 6 to depth sensors, which would allow for a more principled approach to allocate computational resources between depth and purely visual observations.

Appendix A

Lie Theory

This appendix provides an overview of select concepts from Lie group theory that are used in this thesis document.

A.1 Cross and Vee Operators

The cross operator $[\cdot]_\times$ turns a 3×1 vector into a 3×3 skew-symmetric matrix.

$$[\mathbf{a}]_\times = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_\times = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \tag{A.1}$$

The resultant matrix is also known as the cross product matrix because it represents a cross product as a matrix multiplication.

$$[\mathbf{a}]_\times \mathbf{b} = \mathbf{a} \times \mathbf{b} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \times \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = \begin{bmatrix} a_y b_z - a_z b_y \\ a_z b_x - a_x b_z \\ a_x b_y - a_y b_x \end{bmatrix} \tag{A.2}$$

The cross product has the anti-commutative property, which is expressed as

$$[\mathbf{b}]_\times \mathbf{a} = -[\mathbf{a}]_\times \mathbf{b} \tag{A.3}$$

The skew-symmetric property of the cross operator is expressed as

$$[\mathbf{a}]_\times^{\mathrm{T}} = -[\mathbf{a}]_\times \tag{A.4}$$

Matrix multiplication of two cross product matrices can be expressed in terms of the inner and outer products of their corresponding vectors.

$$[\mathbf{a}]_\times [\mathbf{b}]_\times = \mathbf{b}\mathbf{a}^{\mathrm{T}} - \left(\mathbf{a}^{\mathrm{T}}\mathbf{b}\right)\mathbf{I} \tag{A.5}$$

The cross operator applied to a vector that itself is a cross product can be expressed as the difference of outer products of the vectors that form the cross product.

$$\left[[\mathbf{a}]_\times \mathbf{b}\right]_\times = \mathbf{b}\mathbf{a}^{\mathrm{T}} - \mathbf{a}\mathbf{b}^{\mathrm{T}} \tag{A.6}$$

The cross operator is distributive across addition.

$$[\mathbf{a} + \mathbf{b}]_\times = [\mathbf{a}]_\times + [\mathbf{b}]_\times \tag{A.7}$$

The vee operator $[\cdot]^\vee$ is the inverse of the cross operator. It turns a 3×3 skew-symmetric matrix into a 3×1 vector.

$$\begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}^\vee = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix} \tag{A.8}$$

A.2 Rotation Matrices

The Special Orthogonal Group $SO(3)$ is the group of matrices that represent rotations in 3D space.

$$SO(3) = \left\{ \mathbf{R} \in \mathbb{R}^{3\times 3} \mid \mathbf{R}^{\mathrm{T}}\mathbf{R} = \mathbf{I}, \det \mathbf{R} = 1 \right\} \tag{A.9}$$

The product of two rotation matrices is a rotation matrix, and the inverse of a rotation matrix is simply its transpose. $SO(3)$ is a smooth manifold whose tangent space at $\mathbf{R} = \mathbf{I}$ is the set of all 3×3 skew-symmetric matrices

$$\mathfrak{so}(3) = \left\{ [\boldsymbol{\phi}]_\times \mid \boldsymbol{\phi} \in \mathbb{R}^3 \right\} \tag{A.10}$$

A.2.1 Exponential Map

The exponential map from $\mathfrak{so}(3)$ to $SO(3)$ is defined as

$$\exp\left([\boldsymbol{\phi}]_\times\right) = \mathbf{I} + \frac{\sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|}[\boldsymbol{\phi}]_\times + \frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2}[\boldsymbol{\phi}]_\times^2 \tag{A.11}$$

For convenience, the capitalized exponential map from $\mathbb{R}^3$ to $SO(3)$ is defined as

$$\mathrm{Exp}(\boldsymbol{\phi}) = \exp\left([\boldsymbol{\phi}]_\times\right) = \mathbf{I} + \frac{\sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|}[\boldsymbol{\phi}]_\times + \frac{1-\cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2}[\boldsymbol{\phi}]_\times^2 \tag{A.12}$$

Evaluate (A.12) with $-\boldsymbol{\phi}$ and apply property (A.4) to obtain the following identity:

$$\mathrm{Exp}(-\boldsymbol{\phi}) = \mathrm{Exp}(\boldsymbol{\phi})^{\mathrm{T}} \tag{A.13}$$

For small rotations, the capitalized exponential map can be approximated by

$$\mathrm{Exp}(\boldsymbol{\phi}) \approx \mathbf{I} + [\boldsymbol{\phi}]_\times \tag{A.14}$$

A.2.2 Logarithmic Map

The logarithmic map from $SO(3)$ to $\mathfrak{so}(3)$ is

$$\log(\mathbf{R}) = [\boldsymbol{\phi}]_\times = \begin{cases} \mathbf{0}_{3\times 3} & \text{if } \mathbf{R} = \mathbf{I} \\ \frac{\|\boldsymbol{\phi}\|}{2\sin\|\boldsymbol{\phi}\|}\left(\mathbf{R} - \mathbf{R}^{\mathrm{T}}\right) & \text{otherwise} \end{cases} \tag{A.15}$$

$$\|\boldsymbol{\phi}\| = \cos^{-1}\left(\frac{\mathrm{trace}(\mathbf{R}) - 1}{2}\right) \tag{A.16}$$

For convenience, the capitalized logarithmic map from $SO(3)$ to $\mathbb{R}^3$ is

$$\mathrm{Log}(\mathbf{R}) = [\log(\mathbf{R})]^\vee = \boldsymbol{\phi} \tag{A.17}$$

A.3 Quaternions

We use the Hamilton convention (scalar part first) for representing quaternions as 4D vectors.

$$\mathbf{q} = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} \in \mathbb{H} \tag{A.18}$$

The set of unit quaternions is a smooth manifold whose tangent space at the identity element is the set of all quaternions with zero-valued scalar components, or pure quaternions. The inverse of a unit quaternion is

$$\mathbf{q}^{-1} = \begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix}^{-1} = \begin{bmatrix} q_w \\ -\mathbf{q}_v \end{bmatrix} \tag{A.19}$$

The rotation matrix corresponding to a unit quaternion is

$$\mathbf{R}(\mathbf{q}) = \left(q_w^2 - \mathbf{q}_v^{\mathrm{T}}\mathbf{q}_v\right)\mathbf{I} + 2\mathbf{q}_v\mathbf{q}_v^{\mathrm{T}} + 2q_w\left[\mathbf{q}_v\right]_\times \in SO(3) \tag{A.20}$$

The product of two quaternions is defined as

$$\mathbf{p} \otimes \mathbf{q} = \begin{bmatrix} p_w q_w - \mathbf{p}_v^{\mathrm{T}}\mathbf{q}_v \\ p_w\mathbf{q}_v + q_w\mathbf{p}_v + \mathbf{p}_v \times \mathbf{q}_v \end{bmatrix} \tag{A.21}$$

and can be expressed as matrix multiplication

$$\mathbf{p} \otimes \mathbf{q} = Q_L(\mathbf{p})\mathbf{q} = Q_R(\mathbf{q})\mathbf{p} \tag{A.22}$$

$$Q_L(\mathbf{q}) = \begin{bmatrix} q_w & -\mathbf{q}_v^{\mathrm{T}} \\ \mathbf{q}_v & q_w\mathbf{I} + \left[\mathbf{q}_v\right]_\times \end{bmatrix} \tag{A.23}$$

$$Q_R(\mathbf{q}) = \begin{bmatrix} q_w & -\mathbf{q}_v^{\mathrm{T}} \\ \mathbf{q}_v & q_w\mathbf{I} - \left[\mathbf{q}_v\right]_\times \end{bmatrix} \tag{A.24}$$

$Q_L(\cdot)$ and $Q_R(\cdot)$ are functions that return 4×4 matrices representing left quaternion multiplication and right quaternion multiplication of the input quaternion.

We define the $\mathrm{vec}(\cdot)$ function to extract the vector component of a quaternion

$$\mathrm{vec}\left(\mathbf{q}\right) = \mathrm{vec}\left(\begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix}\right) = \mathbf{q}_v \in \mathbb{R}^3 \tag{A.25}$$

and the $\mathrm{sca}(\cdot)$ function to extract the scalar component of a quaternion

$$\mathrm{sca}\left(\mathbf{q}\right) = \mathrm{sca}\left(\begin{bmatrix} q_w \\ \mathbf{q}_v \end{bmatrix}\right) = q_w \in \mathbb{R} \tag{A.26}$$

A.3.1 Exponential Map

The capitalized exponential map for quaternions, $\mathrm{Exp} : \mathbb{R}^3 \to \mathbb{H}$, is

$$\mathrm{Exp}(\boldsymbol{\phi}) \triangleq \begin{bmatrix} \cos\left(\frac{\|\boldsymbol{\phi}\|}{2}\right) \\ \frac{\boldsymbol{\phi}}{\|\boldsymbol{\phi}\|}\sin\left(\frac{\|\boldsymbol{\phi}\|}{2}\right) \end{bmatrix} \tag{A.27}$$

and can be approximated for small $\boldsymbol{\phi}$ as

$$\mathrm{Exp}(\boldsymbol{\phi}) \approx \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{\phi} \end{bmatrix} \tag{A.28}$$

A.3.2 Logarithmic Map

The capitalized logarithmic map for quaternions, $\text{Log} : \mathbb{H} \to \mathbb{R}^3$, is the inverse of the capitalized exponential map.

$$\text{Log}(\mathbf{q}) \triangleq \begin{cases} 2\frac{\text{vec}(\mathbf{q})}{\|\text{vec}(\mathbf{q})\|}\tan^{-1}\left(\frac{\|\text{vec}(\mathbf{q})\|}{\text{sca}(\mathbf{q})}\right) & \text{if sca}(\mathbf{q}) \geq 0 \\ -2\frac{\text{vec}(\mathbf{q})}{\|\text{vec}(\mathbf{q})\|}\tan^{-1}\left(\frac{\|\text{vec}(\mathbf{q})\|}{-\text{sca}(\mathbf{q})}\right) & \text{if sca}(\mathbf{q}) < 0 \end{cases} \tag{A.29}$$

For quaternions that represent small rotations, the capitalized logarithmic map can be approximated as

$$\text{Log}(\mathbf{q}) \approx 2\text{vec}(\mathbf{q}) \tag{A.30}$$

This approximation is often used when evaluating residuals and Jacobians during optimization because it does not require expensive calls to trigonometric functions.

Although the exponential and logarithmic maps for both rotation matrices and quaternions are denoted by the same symbols, it should be clear which one is being used from context by looking at inputs and outputs.

A.4 Euler Angles

Euler angles are a 3DOF minimum parameterization of a rotation. We use the ZYX Euler angle convention, which represents a roll (ϕ) rotation about the initial body x-axis, a pitch (θ) rotation about the intermediate y-axis, and finally a yaw (ψ) rotation about the resultant z-axis.

$$\mathbf{R}(\phi,\theta,\psi) = \mathbf{R}_z(\psi)\mathbf{R}_y(\theta)\mathbf{R}_x(\phi) \tag{A.31}$$

$$\mathbf{R}(\phi,\theta,\psi) = \begin{bmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{bmatrix} \tag{A.32}$$

$$\mathbf{R}(\phi,\theta,\psi) = \begin{bmatrix} \cos\theta\cos\psi & \sin\phi\sin\theta\cos\psi - \cos\phi\sin\psi & \cos\phi\sin\theta\cos\psi + \sin\phi\sin\psi \\ \cos\theta\sin\psi & \cos\phi\cos\psi + \sin\phi\sin\theta\sin\psi & -\sin\phi\cos\psi + \cos\phi\sin\theta\sin\psi \\ -\sin\theta & \sin\phi\cos\theta & \cos\phi\cos\theta \end{bmatrix} \tag{A.33}$$

The ZYX Euler angle convention has a singularities at $\theta = \frac{\pi}{2} + n\pi$, for $n \in \mathbb{Z}$. Physically, this means that roll and yaw are indistinguishable when the vehicle is pitched up or down 90 degrees. Consequently, the ZYX Euler angle convention is unsuitable for describing vertical or inverted maneuvers.

As a minimum parameterization, Euler angles are useful for plotting the time history of a rigid body's orientation in a format that can be easily understood by humans. The ZYX Euler angle convention (or any convention that has a separate yaw component) enables separate handling of the gravity-dependent (roll, pitch) and gravity-independent (yaw) components of rotation, which is useful for defining level reference frames and enforcing observability constraints.

A rotation matrix can be converted into ZYX Euler angles using (A.34)-(A.36).

$$\phi = \begin{cases} \text{atan2}(R_{32}, R_{33}) & \text{if } 1 - R_{31}^2 > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{A.34}$$

$$\theta = -\text{asin}(R_{31}) \tag{A.35}$$

$$\psi = \begin{cases} \text{atan2}(R_{21}, R_{11}) & \text{if } 1 - R_{31}^2 > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{A.36}$$

ZYX Euler angles can be converted to a quaternion using (A.37).

$$\mathbf{q} = \begin{bmatrix} q_w \\ q_x \\ q_y \\ q_z \end{bmatrix} = \begin{bmatrix} \cos\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \sin\frac{\phi}{2}\cos\frac{\theta}{2}\cos\frac{\psi}{2} - \cos\frac{\phi}{2}\sin\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} + \sin\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} \\ \cos\frac{\phi}{2}\cos\frac{\theta}{2}\sin\frac{\psi}{2} - \sin\frac{\phi}{2}\sin\frac{\theta}{2}\cos\frac{\psi}{2} \end{bmatrix} \tag{A.37}$$

A quaternion can be converted into ZYX Euler angles using (A.38)-(A.40).

$$\phi = \text{atan2}\left(2(q_w q_x + q_y q_z), 1 - 2(q_x^2 + q_y^2)\right) \tag{A.38}$$

$$\theta = \text{asin}\left(2(q_w q_y - q_x q_z)\right) \tag{A.39}$$

$$\psi = \text{atan2}\left(2(q_w q_z + q_x q_y), 1 - 2(q_y^2 + q_z^2)\right) \tag{A.40}$$

A.5 Rigid Body Transforms

A 4×4 matrix can be used to represent a 6DOF rigid body transform that rotates and translates a point in 3D space

$$\mathbf{T}(\mathbf{p}, \mathbf{R}) = \begin{bmatrix} \mathbf{R} & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \tag{A.41}$$

where $\mathbf{R} \in SO(3)$ and $\mathbf{p} \in \mathbb{R}^3$. The inverse transform is

$$\mathbf{T}(\mathbf{p}, \mathbf{R})^{-1} = \begin{bmatrix} \mathbf{R}^{\mathrm{T}} & -\mathbf{R}^{\mathrm{T}}\mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \tag{A.42}$$

A 4DOF rigid body transform is defined by a heading and a position and is equivalent to a 6DOF rigid body transform whose roll and pitch components are set to zero.

$$\mathbf{T}(\mathbf{p}, \psi) = \begin{bmatrix} \mathbf{R}_z(\psi) & \mathbf{p} \\ \mathbf{0} & 1 \end{bmatrix} \tag{A.43}$$

4DOF rigid body transforms are also known as *level* transforms because they describe poses whose z-axes are aligned with the gravity vector.

A.6 Useful SO(3) Identities and Approximations

The right Jacobian of $SO(3)$ and its inverse are given by:

$$\mathbf{J}_r(\boldsymbol{\phi}) = \mathbf{I} - \frac{1 - \cos\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^2}[\boldsymbol{\phi}]_\times + \frac{\|\boldsymbol{\phi}\| - \sin\|\boldsymbol{\phi}\|}{\|\boldsymbol{\phi}\|^3}[\boldsymbol{\phi}]_\times^2 \tag{A.44}$$

$$\mathbf{J}_r^{-1}(\boldsymbol{\phi}) = \mathbf{I} + \frac{1}{2}[\boldsymbol{\phi}]_\times + \left(\frac{1}{\|\boldsymbol{\phi}\|^2} - \frac{1 + \cos\|\boldsymbol{\phi}\|}{2\|\boldsymbol{\phi}\|\sin\|\boldsymbol{\phi}\|}\right)[\boldsymbol{\phi}]_\times^2 \tag{A.45}$$

The following identity is taken from equation 11 of [70]:

$$\text{Exp}(\boldsymbol{\phi})\mathbf{R} = \mathbf{R}\,\text{Exp}(\mathbf{R}^{\mathrm{T}}\boldsymbol{\phi}) \tag{A.46}$$

The following approximation for a 3×1 rotation vector $\boldsymbol{\phi}$ and a small 3×1 rotation vector $\delta\boldsymbol{\phi}$ is taken from equation 68 of [90]:

$$\text{Exp}\left(\boldsymbol{\phi} + \delta\boldsymbol{\phi}\right) \approx \text{Exp}(\boldsymbol{\phi})\,\text{Exp}\left(\mathbf{J}_r(\phi)\delta\boldsymbol{\phi}\right) \tag{A.47}$$

The following approximation for a 3×1 rotation vector $\boldsymbol{\phi}$ and a small 3×1 rotation vector $\delta\boldsymbol{\phi}$ is taken from equation 70 of [90]:

$$\mathrm{Log}\left(\mathrm{Exp}(\boldsymbol{\phi})\,\mathrm{Exp}(\delta\boldsymbol{\phi})\right) \approx \boldsymbol{\phi} + \mathbf{J}_r^{-1}(\boldsymbol{\phi})\delta\boldsymbol{\phi} \tag{A.48}$$

Let $\delta\boldsymbol{\phi}$ and $\delta\boldsymbol{\theta}$ be two small rotation vectors. Small rotations can be approximated as being commutative:

$$\mathrm{Exp}\left(\delta\boldsymbol{\phi} + \delta\boldsymbol{\theta}\right) \approx \mathrm{Exp}(\delta\boldsymbol{\phi})\,\mathrm{Exp}(\delta\boldsymbol{\theta}) \approx \mathrm{Exp}(\delta\boldsymbol{\theta})\,\mathrm{Exp}(\delta\boldsymbol{\phi}) \tag{A.49}$$

Appendix B

IMU Preintegration Derivations

The derivations in this appendix use terms defined in Section 3.2.

B.1 Continuous Time Error State Dynamics

Expand out the true state dynamics by substituting (3.37)-(3.42) into (3.43)-(3.47).

$$\dot{\bar{\boldsymbol{\alpha}}} + \delta\dot{\boldsymbol{\alpha}} = \bar{\boldsymbol{\beta}} + \delta\boldsymbol{\beta} \tag{B.1}$$

$$\dot{\bar{\boldsymbol{\beta}}} + \delta\dot{\boldsymbol{\beta}} = \mathbf{R}\left(\bar{\boldsymbol{\gamma}} \otimes \operatorname{Exp}\delta\boldsymbol{\theta}\right)\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a - \delta\mathbf{b}_a - \mathbf{n}_a\right) \tag{B.2}$$

$$\frac{d}{dt}\left(\bar{\boldsymbol{\gamma}} \otimes \operatorname{Exp}\delta\boldsymbol{\theta}\right) = \frac{1}{2}\left(\bar{\boldsymbol{\gamma}} \otimes \operatorname{Exp}\delta\boldsymbol{\theta}\right) \otimes \begin{bmatrix} 0 \\ \mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega - \delta\mathbf{b}_\omega - \mathbf{n}_\omega\right) \end{bmatrix} \tag{B.3}$$

$$\dot{\bar{\mathbf{b}}}_a + \delta\dot{\mathbf{b}}_a = \mathbf{n}_{ba} \tag{B.4}$$

$$\dot{\bar{\mathbf{b}}}_\omega + \delta\dot{\mathbf{b}}_\omega = \mathbf{n}_{b\omega} \tag{B.5}$$

B.1.1 Position Error State Dynamics

Subtract (3.48) from (B.1) to obtain

$$\delta\dot{\boldsymbol{\alpha}} = \delta\boldsymbol{\beta} \tag{B.6}$$

B.1.2 Linear Velocity Error State Dynamics

Rewrite (B.2) using rotation matrices instead of quaternions.

$$\dot{\bar{\boldsymbol{\beta}}} + \delta\dot{\boldsymbol{\beta}} = \mathbf{R}(\bar{\boldsymbol{\gamma}})\operatorname{Exp}(\delta\boldsymbol{\theta})\,\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a - \delta\mathbf{b}_a - \mathbf{n}_a\right) \tag{B.7}$$

Apply the small angle approximation (A.14) for the capitalized exponential map.

$$\dot{\bar{\boldsymbol{\beta}}} + \delta\dot{\boldsymbol{\beta}} \approx \mathbf{R}(\bar{\boldsymbol{\gamma}})\left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right)\,\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a - \delta\mathbf{b}_a - \mathbf{n}_a\right) \tag{B.8}$$

$$\dot{\bar{\boldsymbol{\beta}}} + \delta\dot{\boldsymbol{\beta}} \approx \mathbf{R}(\bar{\boldsymbol{\gamma}})\,\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a - \delta\mathbf{b}_a - \mathbf{n}_a\right) + \mathbf{R}(\bar{\boldsymbol{\gamma}})\,[\delta\boldsymbol{\theta}]_\times\,\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a - \delta\mathbf{b}_a - \mathbf{n}_a\right) \tag{B.9}$$

Subtract (3.49) from (B.9).

$$\delta\dot{\boldsymbol{\beta}} \approx -\mathbf{R}(\bar{\boldsymbol{\gamma}})\,\mathbf{L}_a\left(\delta\mathbf{b}_a + \mathbf{n}_a\right) + \mathbf{R}(\bar{\boldsymbol{\gamma}})\,[\delta\boldsymbol{\theta}]_\times\,\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a\right) - \mathbf{R}(\bar{\boldsymbol{\gamma}})\,[\delta\boldsymbol{\theta}]_\times\left(\delta\mathbf{b}_a + \mathbf{n}_a\right) \tag{B.10}$$

Drop the last term of (B.10) because it is second order in perturbations.

$$\delta\dot{\boldsymbol{\beta}} \approx -\mathbf{R}(\bar{\boldsymbol{\gamma}})\,\mathbf{L}_a\left(\delta\mathbf{b}_a + \mathbf{n}_a\right) + \mathbf{R}(\bar{\boldsymbol{\gamma}})\,[\delta\boldsymbol{\theta}]_\times\,\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a\right) \tag{B.11}$$

Apply (A.3) to the last term of (B.11) and expand.

$$\delta\dot{\boldsymbol{\beta}} \approx -\mathbf{R}(\bar{\boldsymbol{\gamma}})\,\mathbf{L}_a\delta\mathbf{b}_a - \mathbf{R}(\bar{\boldsymbol{\gamma}})\,\mathbf{L}_a\mathbf{n}_a - \mathbf{R}(\bar{\boldsymbol{\gamma}})\left[\mathbf{L}_a\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a\right)\right]_\times\delta\boldsymbol{\theta} \tag{B.12}$$

B.1.3 Attitude Error State Dynamics

Rewrite (B.3) in terms of rotation matrices instead of quaternions.

$$\frac{d}{dt}\left(\bar{\mathbf{R}}\,\mathrm{Exp}(\delta\boldsymbol{\theta})\right) = \bar{\mathbf{R}}\,\mathrm{Exp}(\delta\boldsymbol{\theta})\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega - \delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.13}$$

Rewrite (3.50) in terms of rotation matrices instead of quaternions.

$$\dot{\bar{\mathbf{R}}} = \bar{\mathbf{R}}\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times \tag{B.14}$$

Apply product rule to the left hand side of (B.13)

$$\dot{\bar{\mathbf{R}}}\,\mathrm{Exp}(\delta\boldsymbol{\theta}) + \bar{\mathbf{R}}\frac{d}{dt}\mathrm{Exp}(\delta\boldsymbol{\theta}) = \bar{\mathbf{R}}\,\mathrm{Exp}(\delta\boldsymbol{\theta})\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega - \delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.15}$$

Substitute (B.14) into (B.15) and then pre-multiply both sides by $\bar{\mathbf{R}}^{\mathrm{T}}$ to get rid of the leading $\bar{\mathbf{R}}$ terms.

$$\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times \mathrm{Exp}(\delta\boldsymbol{\theta}) + \frac{d}{dt}\mathrm{Exp}(\delta\boldsymbol{\theta}) = \mathrm{Exp}(\delta\boldsymbol{\theta})\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega - \delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.16}$$

Apply the small angle approximation (A.14) for the capitalized exponential map.

$$\frac{d}{dt}\left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right) \approx -\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times\left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right) + \left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right)\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega - \delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.17}$$

Apply the cross operator's distributive property (A.7).

$$\begin{aligned}\frac{d}{dt}\left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right) \approx &-\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times\left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right) + \left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right)\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times \\ &+ \left(\mathbf{I} + [\delta\boldsymbol{\theta}]_\times\right)\left[\mathbf{L}_\omega\left(-\delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times\end{aligned} \tag{B.18}$$

Expand both sides, cancel out a $\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times$ term, and ignore second order terms.

$$\frac{d}{dt}[\delta\boldsymbol{\theta}]_\times \approx -\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times[\delta\boldsymbol{\theta}]_\times + [\delta\boldsymbol{\theta}]_\times\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times + \left[\mathbf{L}_\omega\left(-\delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.19}$$

Apply the identity (A.5) to each of the first two right hand side terms.

$$\begin{aligned}\frac{d}{dt}[\delta\boldsymbol{\theta}]_\times \approx &-\left\{\delta\boldsymbol{\theta}\left(\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right)^{\mathrm{T}} - \left(\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right)^{\mathrm{T}}\delta\boldsymbol{\theta}\,\mathbf{I}\right\} \\ &+ \mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\delta\boldsymbol{\theta}^{\mathrm{T}} - \delta\boldsymbol{\theta}^{\mathrm{T}}\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\mathbf{I} + \left[\mathbf{L}_\omega\left(-\delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times\end{aligned} \tag{B.20}$$

Cancel out the two terms with the identity matrices.

$$\frac{d}{dt}[\delta\boldsymbol{\theta}]_\times = -\left\{\delta\boldsymbol{\theta}\left(\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right)^{\mathrm{T}} - \mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\delta\boldsymbol{\theta}^{\mathrm{T}}\right\} + \left[\mathbf{L}_\omega\left(-\delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.21}$$

Apply the identity (A.2) to the term in the curly brackets.

$$\frac{d}{dt}[\delta\boldsymbol{\theta}]_\times \approx -\left[\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times\delta\boldsymbol{\theta}\right]_\times + \left[\mathbf{L}_\omega\left(-\delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.22}$$

Apply the cross operator's distributive property (A.7).

$$\frac{d}{dt}[\delta\boldsymbol{\theta}]_\times \approx \left[-\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times\delta\boldsymbol{\theta} + \mathbf{L}_\omega\left(-\delta\mathbf{b}_\omega - \mathbf{n}_\omega\right)\right]_\times \tag{B.23}$$

Apply the vee operator to both sides.

$$\delta\dot{\boldsymbol{\theta}} \approx -\left[\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega\right)\right]_\times\delta\boldsymbol{\theta} - \mathbf{L}_\omega\delta\mathbf{b}_\omega - \mathbf{L}_\omega\mathbf{n}_\omega \tag{B.24}$$

B.1.4 IMU Bias Error State Dynamics

Subtract (3.51) from (B.4) to obtain

$$\delta\dot{\mathbf{b}}_a = \mathbf{n}_{ba} \tag{B.25}$$

Subtract (3.52) from (B.5) to obtain

$$\delta\dot{\mathbf{b}}_\omega = \mathbf{n}_{b\omega} \tag{B.26}$$

B.1.5 Assembly into Linear Time Varying System

Combine (B.6), (B.12), (B.24), (B.25), and (B.26) and write the result in matrix form to obtain (3.56)-(3.58).

B.2 Midpoint Rule Preintegration

Let $k-1$ and k be the time indices of two successive IMU observations. Define the time interval between two successive IMU observations as $\Delta t_k \equiv t_k - t_{k-1}$.

B.2.1 Nominal State

Compute the average angular velocity with respect to the body frame at the beginning of the preintegration time interval ($k = 0$) using bias-compensated gyroscope observations at time indices $k-1$ and k.

$$\bar{\boldsymbol{\omega}}_{\text{avg}} = \mathbf{L}_\omega \left(\frac{1}{2} \left(\hat{\boldsymbol{\omega}}_{k-1} + \hat{\boldsymbol{\omega}}_k \right) - \bar{\mathbf{b}}_{\omega,k-1} \right) \tag{B.27}$$

The attitude nominal state gets updated with the average of the bias-compensated angular velocities at time indices $k-1$ and k.

$$\bar{\boldsymbol{\gamma}}_k = \bar{\boldsymbol{\gamma}}_{k-1} \otimes \text{Exp}\left[\bar{\boldsymbol{\omega}}_{\text{avg}} \Delta t_k \right] \tag{B.28}$$

Compute the average acceleration with respect to the body frame at the beginning of the preintegration time interval ($k = 0$) using bias-compensated accelerometer observations and attitude estimates at time indices $k-1$ and k.

$$\bar{\mathbf{a}}_{\text{avg}} = \frac{1}{2} \left[\mathbf{R}(\bar{\boldsymbol{\gamma}}_{k-1}) \mathbf{L}_a \left(\hat{\mathbf{a}}_{k-1} - \bar{\mathbf{b}}_{a,k-1} \right) + \mathbf{R}(\bar{\boldsymbol{\gamma}}_k) \mathbf{L}_a \left(\hat{\mathbf{a}}_k - \bar{\mathbf{b}}_{a,k-1} \right) \right] \tag{B.29}$$

Propagate the position and velocity nominal states forward using (B.29).

$$\bar{\boldsymbol{\alpha}}_k = \bar{\boldsymbol{\alpha}}_{k-1} + \bar{\boldsymbol{\beta}}_{k-1} \Delta t_k + \frac{1}{2} \bar{\mathbf{a}}_{\text{avg}} \Delta t_k^2 \tag{B.30}$$

$$\bar{\boldsymbol{\beta}}_k = \bar{\boldsymbol{\beta}}_{k-1} + \bar{\mathbf{a}}_{\text{avg}} \Delta t_k \tag{B.31}$$

B.2.2 Error State

The midpoint rule Jacobian that updates the error state from time index $k-1$ to time index k is

$$\mathbf{F}_k = \begin{bmatrix} \mathbf{I} & \mathbf{F}_{\alpha\beta} & \mathbf{F}_{\alpha\theta} & \mathbf{F}_{\alpha b_a} & \mathbf{F}_{\alpha b_\omega} \\ \mathbf{0} & \mathbf{I} & \mathbf{F}_{\beta\theta} & \mathbf{F}_{\beta b_a} & \mathbf{F}_{\beta b_\omega} \\ \mathbf{0} & \mathbf{0} & \mathbf{F}_{\theta\theta} & \mathbf{0} & \mathbf{F}_{\theta b_\omega} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} \end{bmatrix} \in \mathbb{R}^{15 \times 15} \tag{B.32}$$

with the following 3×3 sub-blocks:

$$\mathbf{F}_{\alpha\beta} = \Delta t_k \mathbf{I} \tag{B.33}$$

$$\mathbf{F}_{\alpha\theta} = \frac{1}{2}\Delta t_k \mathbf{F}_{\beta\theta} \tag{B.34}$$

$$\mathbf{F}_{\alpha b_a} = \frac{1}{2}\Delta t_k \mathbf{F}_{\beta b_a} \tag{B.35}$$

$$\mathbf{F}_{\alpha b_\omega} = \frac{1}{2}\Delta t_k \mathbf{F}_{\beta b_\omega} \tag{B.36}$$

$$\mathbf{F}_{\beta\theta} = -\frac{1}{2}\Delta t_k \left\{ \mathbf{R}(\bar{\boldsymbol{\gamma}}_{k-1}) \left[\mathbf{L}_a \left(\hat{\mathbf{a}}_{k-1} - \bar{\mathbf{b}}_{a,k-1} \right) \right]_\times + \mathbf{R}(\bar{\boldsymbol{\gamma}}_k) \left[\mathbf{L}_a \left(\hat{\mathbf{a}}_k - \bar{\mathbf{b}}_{a,k-1} \right) \right]_\times \mathbf{F}_{\theta\theta} \right\} \tag{B.37}$$

$$\mathbf{F}_{\beta b_a} = -\frac{1}{2}\Delta t_k \left[\mathbf{R}(\bar{\boldsymbol{\gamma}}_{k-1}) + \mathbf{R}(\bar{\boldsymbol{\gamma}}_k) \right] \mathbf{L}_a \tag{B.38}$$

$$\mathbf{F}_{\beta b_\omega} = -\frac{1}{2}\Delta t_k \mathbf{R}(\bar{\boldsymbol{\gamma}}_k) \left[\mathbf{L}_a \left(\hat{\mathbf{a}}_k - \bar{\mathbf{b}}_{a,k-1} \right) \right]_\times \mathbf{F}_{\theta b_\omega} \tag{B.39}$$

$$\mathbf{F}_{\theta\theta} = \mathbf{I} - \Delta t_k \left[\bar{\boldsymbol{\omega}}_{\text{avg}} \right]_\times \tag{B.40}$$

$$\mathbf{F}_{\theta b_\omega} = -\Delta t_k \mathbf{L}_\omega \tag{B.41}$$

The midpoint rule Jacobian that captures the noise state at time index $k-1$'s contribution to the error state at time index k is

$$\mathbf{G}_k = \begin{bmatrix} \mathbf{G}_{\alpha n_a} & \mathbf{G}_{\alpha n_\omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{G}_{\beta n_a} & \mathbf{G}_{\beta n_\omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{G}_{\theta n_\omega} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{G}_{b_a n_{ba}} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{G}_{b_\omega n_{b\omega}} \end{bmatrix} \in \mathbb{R}^{15\times 12} \tag{B.42}$$

with the following 3×3 sub-blocks:

$$\mathbf{G}_{\alpha n_a} = \frac{1}{2}\Delta t_k \mathbf{G}_{\beta n_a} \tag{B.43}$$

$$\mathbf{G}_{\alpha n_\omega} = \frac{1}{2}\Delta t_k \mathbf{G}_{\beta n_\omega} \tag{B.44}$$

$$\mathbf{G}_{\beta n_a} = -\frac{1}{2}\Delta t_k \left[\mathbf{R}(\bar{\boldsymbol{\gamma}}_{k-1}) + \mathbf{R}(\bar{\boldsymbol{\gamma}}_k) \right] \mathbf{L}_a \tag{B.45}$$

$$\mathbf{G}_{\beta n_\omega} = -\frac{1}{2}\Delta t_k \mathbf{R}(\bar{\boldsymbol{\gamma}}_k) \left[\mathbf{L}_a \left(\hat{\mathbf{a}}_k - \bar{\mathbf{b}}_{a,k-1} \right) \right]_\times \mathbf{G}_{b_\omega n_{b\omega}} \tag{B.46}$$

$$\mathbf{G}_{\theta n_\omega} = -\Delta t_k \mathbf{L}_\omega \tag{B.47}$$

$$\mathbf{G}_{b_a n_{ba}} = \Delta t_k \mathbf{I} \tag{B.48}$$

$$\mathbf{G}_{b_\omega n_{b\omega}} = \Delta t_k \mathbf{I} \tag{B.49}$$

Substitute (B.32) and (B.42) into (3.59) to perform midpoint rule integration of the error state dynamics.

B.3 Extension to Support Online IMU Intrinsic Calibration

This section repeats the developments of Sections 3.2.3 and 3.3 for the case where $\mathbf{L}_a$ and $\mathbf{L}_\omega$ are being jointly estimated in optimization. For convenience, we define

$$\mathbf{l}_a = \begin{bmatrix} L_{a,xx} & L_{a,yx} & L_{a,yy} & L_{a,zx} & L_{a,zy} & L_{a,zz} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^6 \tag{B.50}$$

$$\mathbf{l}_\omega = \begin{bmatrix} L_{\omega,xx} & L_{\omega,xy} & L_{\omega,xz} & L_{\omega,yx} & L_{\omega,yy} & L_{\omega,yz} & L_{\omega,zx} & L_{\omega,yz} & L_{\omega,zz} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^9 \tag{B.51}$$

as the vectorized forms of $\mathbf{L}_a$ and $\mathbf{L}_\omega$ in row-major order. We extend the true, nominal, and error states with the following IMU intrinsic parameters

$$\mathbf{l}_a = \bar{\mathbf{l}}_a + \delta\mathbf{l}_a \tag{B.52}$$

$$\mathbf{l}_\omega = \bar{\mathbf{l}}_\omega + \delta\mathbf{l}_\omega \tag{B.53}$$

such that

$$\mathbf{x} = \begin{bmatrix} \boldsymbol{\alpha} \\ \boldsymbol{\beta} \\ \boldsymbol{\gamma} \\ \mathbf{b}_a \\ \mathbf{b}_\omega \\ \mathbf{l}_a \\ \mathbf{l}_\omega \end{bmatrix}, \quad \bar{\mathbf{x}} = \begin{bmatrix} \bar{\boldsymbol{\alpha}} \\ \bar{\boldsymbol{\beta}} \\ \bar{\boldsymbol{\gamma}} \\ \bar{\mathbf{b}}_a \\ \bar{\mathbf{b}}_\omega \\ \bar{\mathbf{l}}_a \\ \bar{\mathbf{l}}_\omega \end{bmatrix}, \quad \delta\mathbf{x} = \begin{bmatrix} \delta\boldsymbol{\alpha} \\ \delta\boldsymbol{\beta} \\ \delta\boldsymbol{\gamma} \\ \delta\mathbf{b}_a \\ \delta\mathbf{b}_\omega \\ \delta\mathbf{l}_a \\ \delta\mathbf{l}_\omega \end{bmatrix} \tag{B.54}$$

For convenience we define the operators $[\cdot]_\triangle$ and $[\cdot]_\square$ for the 3×1 vector $\mathbf{a} = [a_x \; a_y \; a_z]^{\mathrm{T}}$ as

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_\triangle = \begin{bmatrix} a_x & 0 & 0 & 0 & 0 & 0 \\ 0 & a_x & a_y & 0 & 0 & 0 \\ 0 & 0 & 0 & a_x & a_y & a_z \end{bmatrix} \in \mathbb{R}^{3\times 6} \tag{B.55}$$

$$\begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_\square = \begin{bmatrix} a_x & a_y & a_z & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & a_x & a_y & a_z & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & a_x & a_y & a_z \end{bmatrix} \in \mathbb{R}^{3\times 9} \tag{B.56}$$

These operators allow us to write

$$\delta\mathbf{L}_a \, \mathbf{a} = [\mathbf{a}]_\triangle \, \delta\mathbf{l}_a \tag{B.57}$$

$$\delta\mathbf{L}_\omega \, \mathbf{a} = [\mathbf{a}]_\square \, \delta\mathbf{l}_\omega \tag{B.58}$$

B.3.1 True and Nominal State Continuous Time Dyanmics

The true state continuous time dynamics are given by (3.43)-(3.47) together with the following two differential equations:

$$\dot{\mathbf{l}}_a = \mathbf{0} \tag{B.59}$$

$$\dot{\mathbf{l}}_\omega = \mathbf{0} \tag{B.60}$$

The nominal state continuous time dynamics are given by (3.48)-(3.52) together with the following two differential equations:

$$\dot{\bar{\mathbf{l}}}_a = \mathbf{0} \tag{B.61}$$

$$\dot{\bar{\mathbf{l}}}_\omega = \mathbf{0} \tag{B.62}$$

The expanded true state continuous time dynamics are given by (B.1)-(B.5) with (B.2) replaced by

$$\dot{\bar{\boldsymbol{\beta}}} + \delta\dot{\boldsymbol{\beta}} = \mathbf{R}\left(\bar{\boldsymbol{\gamma}} \otimes \operatorname{Exp}\delta\boldsymbol{\theta}\right)\left(\bar{\mathbf{L}}_a + \delta\bar{\mathbf{L}}_a\right)\left(\hat{\mathbf{a}} - \bar{\mathbf{b}}_a - \delta\mathbf{b}_a - \mathbf{n}_a\right) \tag{B.63}$$

and (B.3) replaced by

$$\frac{d}{dt}\left(\bar{\boldsymbol{\gamma}} \otimes \operatorname{Exp}\delta\boldsymbol{\theta}\right) = \frac{1}{2}\left(\bar{\boldsymbol{\gamma}} \otimes \operatorname{Exp}\delta\boldsymbol{\theta}\right) \otimes \begin{bmatrix} 0 \\ \left(\mathbf{L}_\omega + \delta\mathbf{L}_\omega\right)\left(\hat{\boldsymbol{\omega}} - \bar{\mathbf{b}}_\omega - \delta\mathbf{b}_\omega - \mathbf{n}_\omega\right) \end{bmatrix} \tag{B.64}$$

and the addition of

$$\dot{\bar{\mathbf{l}}}_a + \delta\dot{\mathbf{l}}_a = \mathbf{0} \tag{B.65}$$

$$\dot{\bar{\mathbf{l}}}_\omega + \delta\dot{\mathbf{l}}_\omega = \mathbf{0} \tag{B.66}$$

B.3.2 Error State Continuous Time Dynamics

Applying the techniques of Appendix B.1.2 to (B.63), we obtain

$$\delta\dot{\boldsymbol{\beta}} \approx -\mathbf{R}(\bar{\boldsymbol{\gamma}})\,\bar{\mathbf{L}}_a\delta\mathbf{b}_a - \mathbf{R}(\bar{\boldsymbol{\gamma}})\,\bar{\mathbf{L}}_a\mathbf{n}_a - \mathbf{R}(\bar{\boldsymbol{\gamma}})\left[\bar{\mathbf{L}}_a\left(\hat{\mathbf{a}}-\bar{\mathbf{b}}_a\right)\right]_\times\delta\boldsymbol{\theta} + \mathbf{R}(\bar{\boldsymbol{\gamma}})\,\delta\mathbf{L}_a\left(\hat{\mathbf{a}}-\bar{\mathbf{b}}_a\right) \tag{B.67}$$

and rewrite the last term using (B.57)

$$\delta\dot{\boldsymbol{\beta}} \approx -\mathbf{R}(\bar{\boldsymbol{\gamma}})\,\bar{\mathbf{L}}_a\delta\mathbf{b}_a - \mathbf{R}(\bar{\boldsymbol{\gamma}})\,\bar{\mathbf{L}}_a\mathbf{n}_a - \mathbf{R}(\bar{\boldsymbol{\gamma}})\left[\bar{\mathbf{L}}_a\left(\hat{\mathbf{a}}-\bar{\mathbf{b}}_a\right)\right]_\times\delta\boldsymbol{\theta} + \mathbf{R}(\bar{\boldsymbol{\gamma}})\left[\hat{\mathbf{a}}-\bar{\mathbf{b}}_a\right]_\triangle\delta\mathbf{l}_a \tag{B.68}$$

Applying the techniques of Appendix (B.1.3) to (B.64), we obtain

$$\delta\dot{\boldsymbol{\theta}} \approx -\left[\bar{\mathbf{L}}_\omega\left(\hat{\boldsymbol{\omega}}-\bar{\mathbf{b}}_\omega\right)\right]_\times\delta\boldsymbol{\theta} - \bar{\mathbf{L}}_\omega\delta\mathbf{b}_\omega - \bar{\mathbf{L}}_\omega\mathbf{n}_\omega + \delta\mathbf{L}_\omega\left(\hat{\boldsymbol{\omega}}-\bar{\mathbf{b}}_\omega\right) \tag{B.69}$$

and rewrite the last term using (B.58)

$$\delta\dot{\boldsymbol{\theta}} \approx -\left[\bar{\mathbf{L}}_\omega\left(\hat{\boldsymbol{\omega}}-\bar{\mathbf{b}}_\omega\right)\right]_\times\delta\boldsymbol{\theta} - \bar{\mathbf{L}}_\omega\delta\mathbf{b}_\omega - \bar{\mathbf{L}}_\omega\mathbf{n}_\omega + \left[\hat{\boldsymbol{\omega}}-\bar{\mathbf{b}}_\omega\right]_\square\delta\mathbf{l}_\omega \tag{B.70}$$

The continuous time error state dynamics for the accelerometer intrinsic parameters are obtained by subtracting (B.61) from (B.65).

$$\delta\dot{\mathbf{l}}_a = \mathbf{0} \tag{B.71}$$

The continuous time error state dynamics for the gyroscope intrinsic parameters are obtained by subtracting (B.62) from (B.66)

$$\delta\dot{\mathbf{l}}_\omega = \mathbf{0} \tag{B.72}$$

Combine (B.6), (B.68), (B.70), (B.25), (B.26), (B.71), and (B.72) and write the result in matrix form to obtain

$$\delta\dot{\mathbf{x}} = \mathbf{F}_c\delta\mathbf{x} + \mathbf{G}_c\mathbf{n} \tag{B.73}$$

$$\mathbf{F}_c = \left[\begin{array}{ccccccc}
\mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times9} \\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\mathbf{R}(\bar{\boldsymbol{\gamma}})\left[\bar{\mathbf{L}}_a\left(\hat{\mathbf{a}}-\bar{\mathbf{b}}_a\right)\right]_\times & -\mathbf{R}(\bar{\boldsymbol{\gamma}})\bar{\mathbf{L}}_a & \mathbf{0}_{3\times3} & \mathbf{R}(\bar{\boldsymbol{\gamma}})\left[\hat{\mathbf{a}}-\bar{\mathbf{b}}_a\right]_\triangle & \mathbf{0}_{3\times9} \\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & -\left[\bar{\mathbf{L}}_\omega\left(\hat{\boldsymbol{\omega}}-\bar{\mathbf{b}}_\omega\right)\right]_\times & \mathbf{0}_{3\times3} & -\bar{\mathbf{L}}_\omega & \mathbf{0}_{3\times6} & \left[\hat{\boldsymbol{\omega}}-\bar{\mathbf{b}}_\omega\right]_\square \\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times9} \\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times9} \\
\hline
\multicolumn{7}{c}{\mathbf{0}_{15\times30}}
\end{array}\right] \tag{B.74}$$

$$\mathbf{G}_c = \left[\begin{array}{cccc}
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\
-\mathbf{R}(\bar{\boldsymbol{\gamma}})\bar{\mathbf{L}}_a & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\
\mathbf{0}_{3\times3} & -\bar{\mathbf{L}}_\omega & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} \\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} \\
\mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} \\
\hline
\multicolumn{4}{c}{\mathbf{0}_{15\times12}}
\end{array}\right] \tag{B.75}$$

to replace (3.56)-(3.58).

B.3.3 Midpoint Rule for Error State Dynamics

We assume that $\mathbf{L}_a$ and $\mathbf{L}_\omega$ are constant over any preintegration time interval. The midpoint rule Jacobian that updates the error state from time index $k-1$ to time index k is

$$\mathbf{F}_k = \left[\begin{array}{ccccccc} \mathbf{I}_{3\times3} & \mathbf{F}_{\alpha\beta} & \mathbf{F}_{\alpha\theta} & \mathbf{F}_{\alpha b_a} & \mathbf{F}_{\alpha b_\omega} & \mathbf{F}_{\alpha l_a} & \mathbf{F}_{\alpha l_\omega} \\ \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{F}_{\beta\theta} & \mathbf{F}_{\beta b_a} & \mathbf{F}_{\beta b_\omega} & \mathbf{F}_{\beta l_a} & \mathbf{F}_{\beta l_\omega} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{F}_{\theta\theta} & \mathbf{0}_{3\times3} & \mathbf{F}_{\theta b_\omega} & \mathbf{0}_{3\times6} & \mathbf{F}_{\theta l_\omega} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times9} \\ \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{0}_{3\times3} & \mathbf{I}_{3\times3} & \mathbf{0}_{3\times6} & \mathbf{0}_{3\times9} \\ \hline \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{0}_{6\times3} & \mathbf{I}_{6\times6} & \mathbf{0}_{6\times9} \\ \mathbf{0}_{9\times3} & \mathbf{0}_{9\times3} & \mathbf{0}_{9\times3} & \mathbf{0}_{9\times3} & \mathbf{0}_{9\times3} & \mathbf{0}_{9\times6} & \mathbf{I}_{9\times9} \end{array}\right] \in \mathbb{R}^{30\times30} \tag{B.76}$$

with 3×3 sub-blocks defined in (B.33)-(B.41) (substituting $\mathbf{L}_a \rightarrow \bar{\mathbf{L}}_a$ and $\mathbf{L}_\omega \rightarrow \bar{\mathbf{L}}_\omega$) as well as the following new sub-blocks:

$$\mathbf{F}_{\alpha l_a} = \frac{1}{2}\Delta t_k \mathbf{F}_{\beta l_a} \tag{B.77}$$

$$\mathbf{F}_{\alpha l_\omega} = \frac{1}{2}\Delta t_k \mathbf{F}_{\beta l_\omega} \tag{B.78}$$

$$\mathbf{F}_{\beta l_a} = \frac{1}{2}\Delta t_k \left\{ \mathbf{R}(\bar{\gamma}_{k-1}) \left[\hat{\mathbf{a}}_{k-1} - \bar{\mathbf{b}}_{a,k-1}\right]_\triangle + \mathbf{R}(\bar{\gamma}_k) \left[\hat{\mathbf{a}}_k - \bar{\mathbf{b}}_{a,k-1}\right]_\triangle \right\} \tag{B.79}$$

$$\mathbf{F}_{\beta l_\omega} = -\frac{1}{2}\Delta t_k \mathbf{R}(\bar{\gamma}_k) \left[\bar{\mathbf{L}}_a \left(\hat{\mathbf{a}}_k - \bar{\mathbf{b}}_{a,k-1}\right)\right]_\times \mathbf{F}_{\theta l_w} \tag{B.80}$$

$$\mathbf{F}_{\theta l_w} = \Delta t_k \left[\frac{1}{2}\left(\hat{\boldsymbol{\omega}}_{k-1} + \hat{\boldsymbol{\omega}}_k\right) - \bar{\mathbf{b}}_{\omega,k-1}\right]_\square \tag{B.81}$$

The midpoint rule Jacobian that captures the noise state at time index $k-1$'s contribution to the error state at time index k is obtained by padding the bottom of (B.42) with 15 rows of zeros.

B.3.4 IMU Bias and Intrinsics Perturbation Model

Replace (3.68)-(3.71) with the following:

$$\left[\begin{array}{c} \delta\boldsymbol{\alpha}_n \\ \delta\boldsymbol{\beta}_n \\ \delta\boldsymbol{\theta}_n \\ \delta\mathbf{b}_{a,n} \\ \delta\mathbf{b}_{\omega,n} \\ \delta\mathbf{l}_{a,n} \\ \delta\mathbf{l}_{\omega,n} \end{array}\right] = \left[\begin{array}{ccccccc} \mathbf{J}_{\alpha\alpha} & \mathbf{J}_{\alpha\beta} & \mathbf{J}_{\alpha\theta} & \mathbf{J}_{\alpha b_a} & \mathbf{J}_{\alpha b_\omega} & \mathbf{J}_{\alpha l_a} & \mathbf{J}_{\alpha l_\omega} \\ \mathbf{J}_{\beta\alpha} & \mathbf{J}_{\beta\beta} & \mathbf{J}_{\beta\theta} & \mathbf{J}_{\beta b_a} & \mathbf{J}_{\beta b_\omega} & \mathbf{J}_{\beta l_a} & \mathbf{J}_{\beta l_\omega} \\ \mathbf{J}_{\theta\alpha} & \mathbf{J}_{\theta\beta} & \mathbf{J}_{\theta\theta} & \mathbf{J}_{\theta b_a} & \mathbf{J}_{\theta b_\omega} & \mathbf{J}_{\theta l_a} & \mathbf{J}_{\theta l_\omega} \\ \mathbf{J}_{b_a\alpha} & \mathbf{J}_{b_a\beta} & \mathbf{J}_{b_a\theta} & \mathbf{J}_{b_a b_a} & \mathbf{J}_{b_a b_\omega} & \mathbf{J}_{b_a l_a} & \mathbf{J}_{b_a l_\omega} \\ \mathbf{J}_{b_\omega\alpha} & \mathbf{J}_{b_\omega\beta} & \mathbf{J}_{b_\omega\theta} & \mathbf{J}_{b_\omega b_a} & \mathbf{J}_{b_\omega b_\omega} & \mathbf{J}_{b_\omega l_a} & \mathbf{J}_{b_\omega l_\omega} \\ \mathbf{J}_{l_a\alpha} & \mathbf{J}_{l_a\beta} & \mathbf{J}_{l_a\theta} & \mathbf{J}_{l_a b_a} & \mathbf{J}_{l_a b_\omega} & \mathbf{J}_{l_a l_a} & \mathbf{J}_{l_a l_\omega} \\ \mathbf{J}_{l_\omega\alpha} & \mathbf{J}_{l_\omega\beta} & \mathbf{J}_{l_\omega\theta} & \mathbf{J}_{l_\omega b_a} & \mathbf{J}_{l_\omega b_\omega} & \mathbf{J}_{l_\omega l_a} & \mathbf{J}_{l_\omega l_\omega} \end{array}\right] \left[\begin{array}{c} \delta\boldsymbol{\alpha}_0 \\ \delta\boldsymbol{\beta}_0 \\ \delta\boldsymbol{\theta}_0 \\ \delta\mathbf{b}_{a,0} \\ \delta\mathbf{b}_{\omega,0} \\ \delta\mathbf{l}_{a,0} \\ \delta\mathbf{l}_{\omega,0} \end{array}\right] + \ldots \tag{B.82}$$

$$\delta\boldsymbol{\alpha}_n \approx \mathbf{J}_{\alpha b_a}\delta\mathbf{b}_{a,0} + \mathbf{J}_{\alpha b_\omega}\delta\mathbf{b}_{w,0} + \mathbf{J}_{\alpha l_a}\delta\mathbf{l}_{a,0} + \mathbf{J}_{\alpha l_\omega}\delta\mathbf{l}_{\omega,0} + \ldots \tag{B.83}$$

$$\delta\boldsymbol{\beta}_n \approx \mathbf{J}_{\beta b_a}\delta\mathbf{b}_{a,0} + \mathbf{J}_{\beta b_\omega}\delta\mathbf{b}_{w,0} + \mathbf{J}_{\beta l_a}\delta\mathbf{l}_{a,0} + \mathbf{J}_{\beta l_\omega}\delta\mathbf{l}_{\omega,0} + \ldots \tag{B.84}$$

$$\delta\boldsymbol{\theta}_n \approx \mathbf{J}_{\theta b_\omega}\delta\mathbf{b}_{\omega,0} + \mathbf{J}_{\theta l_\omega}\delta\mathbf{l}_{\omega,0} + \ldots \tag{B.85}$$

Note that $\mathbf{J}_{\theta l_a} = \mathbf{0}_{3\times6}$ because the accelerometer intrinsic parameters do not affect rotation. In the same manner as (3.72)-(3.73), we obtain

$$\delta\mathbf{l}_{a,0} = \mathbf{l}_{a,0} - \bar{\mathbf{l}}_{a,0} \tag{B.86}$$

$$\delta\mathbf{l}_{\omega,0} = \mathbf{l}_{\omega,0} - \bar{\mathbf{l}}_{\omega,0} \tag{B.87}$$

The right hand side terms of (3.74)-(3.76) now additionally depend on $\bar{\mathbf{l}}_{a,0}$ and $\bar{\mathbf{l}}_{\omega,0}$.

First order approximations of changes in motion deltas now depend on changes in IMU intrinsic parameter estimates as well as changes in initial IMU bias estimates. Replace (3.77)-(3.79) with the following:

$$\boldsymbol{\alpha}'_n\left(\mathbf{b}_{a,0},\mathbf{b}_{\omega,0},\mathbf{l}_{a,0},\mathbf{l}_{\omega,0}\right) \approx \bar{\boldsymbol{\alpha}}_n\left(\bar{\mathbf{b}}_{a,0},\bar{\mathbf{b}}_{\omega,0},\bar{\mathbf{l}}_{a,0},\bar{\mathbf{l}}_{\omega,0}\right) + \mathbf{J}_{\alpha b_a}\left(\mathbf{b}_{a,0}-\bar{\mathbf{b}}_{a,0}\right) + \mathbf{J}_{\alpha b_\omega}\left(\mathbf{b}_{\omega,0}-\bar{\mathbf{b}}_{\omega,0}\right) + \mathbf{J}_{\alpha l_a}\left(\mathbf{l}_{a,0}-\bar{\mathbf{l}}_{a,0}\right) + \mathbf{J}_{\alpha l_\omega}\left(\mathbf{l}_{\omega,0}-\bar{\mathbf{l}}_{\omega,0}\right) \tag{B.88}$$

$$\boldsymbol{\beta}'_n\left(\mathbf{b}_{a,0},\mathbf{b}_{\omega,0},\mathbf{l}_{a,0},\mathbf{l}_{\omega,0}\right) \approx \bar{\boldsymbol{\beta}}_n\left(\bar{\mathbf{b}}_{a,0},\bar{\mathbf{b}}_{\omega,0},\bar{\mathbf{l}}_{a,0},\bar{\mathbf{l}}_{\omega,0}\right) + \mathbf{J}_{\beta b_a}\left(\mathbf{b}_{a,0}-\bar{\mathbf{b}}_{a,0}\right) + \mathbf{J}_{\beta b_\omega}\left(\mathbf{b}_{\omega,0}-\bar{\mathbf{b}}_{\omega,0}\right) + \mathbf{J}_{\beta l_a}\left(\mathbf{l}_{a,0}-\bar{\mathbf{l}}_{a,0}\right) + \mathbf{J}_{\beta l_\omega}\left(\mathbf{l}_{\omega,0}-\bar{\mathbf{l}}_{\omega,0}\right) \tag{B.89}$$

$$\boldsymbol{\gamma}'_n\left(\mathbf{b}_{\omega,0},\mathbf{l}_{\omega,0}\right) \approx \bar{\boldsymbol{\gamma}}_n\left(\bar{\mathbf{b}}_{\omega,0},\bar{\mathbf{l}}_{\omega,0}\right) \otimes \mathrm{Exp}\left[\mathbf{J}_{\theta b_\omega}\left(\mathbf{b}_{\omega,0}-\bar{\mathbf{b}}_{\omega,0}\right) + \mathbf{J}_{\theta l_\omega}\left(\mathbf{l}_{\omega,0}-\bar{\mathbf{l}}_{\omega,0}\right)\right] \tag{B.90}$$

B.3.5 IMU Preintegration Factor

Residual

The IMU preintegration factor residual vector is obtained by replacing (3.77)-(3.79) with (B.88)-(B.90) in (3.82)-(3.86).

$$\mathbf{r}_{\text{imu}}(\mathbf{x}) = \begin{bmatrix} \mathbf{r}_\alpha\left(\mathbf{p}_i,\mathbf{v}_i,\mathbf{q}_i,\mathbf{b}_{a,i},\mathbf{b}_{\omega,i},\mathbf{p}_j,\mathbf{l}_a,\mathbf{l}_\omega\right) \\ \mathbf{r}_\beta\left(\mathbf{v}_i,\mathbf{q}_i,\mathbf{b}_{a,i},\mathbf{b}_{\omega,i},\mathbf{v}_j,\mathbf{l}_a,\mathbf{l}_\omega\right) \\ \mathbf{r}_\theta\left(\mathbf{q}_i,\mathbf{b}_{\omega,j},\mathbf{q}_j,\mathbf{l}_\omega\right) \\ \mathbf{r}_{ba}\left(\mathbf{b}_{a,i},\mathbf{b}_{a,j}\right) \\ \mathbf{r}_{b\omega}\left(\mathbf{b}_{\omega,i},\mathbf{b}_{\omega,j}\right) \end{bmatrix} \in \mathbb{R}^{15} \tag{B.91}$$

$$\mathbf{x} = \begin{bmatrix} \mathbf{p}_i^{\mathrm{T}} & \mathbf{v}_i^{\mathrm{T}} & \mathbf{q}_i^{\mathrm{T}} & \mathbf{b}_{a,i}^{\mathrm{T}} & \mathbf{b}_{\omega,i}^{\mathrm{T}} & \mathbf{p}_j^{\mathrm{T}} & \mathbf{v}_i^{\mathrm{T}} & \mathbf{q}_j^{\mathrm{T}} & \mathbf{b}_{a,j}^{\mathrm{T}} & \mathbf{b}_{\omega,j}^{\mathrm{T}} & \mathbf{l}_a^{\mathrm{T}} & \mathbf{l}_\omega^{\mathrm{T}} \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^{47} \tag{B.92}$$

The IMU preintegration factor residual's covariance is the top-left 15×15 block of the covariance of the motion delta error dynamics[1] at the end of the preintegration time interval.

Jacobians

The Jacobians defined in (3.89)-(3.101), (3.110), (3.111), and (3.128)-(3.130) are still valid. The following additional non-zero Jacobians with respect to the IMU intrinsics parameters are required.

$$\frac{\partial \mathbf{r}_\alpha}{\partial \delta \mathbf{l}_a} = -\mathbf{J}_{\alpha l_a} \tag{B.93}$$

$$\frac{\partial \mathbf{r}_\alpha}{\partial \delta \mathbf{l}_\omega} = -\mathbf{J}_{\alpha l_\omega} \tag{B.94}$$

$$\frac{\partial \mathbf{r}_\beta}{\partial \delta \mathbf{l}_a} = -\mathbf{J}_{\beta l_a} \tag{B.95}$$

$$\frac{\partial \mathbf{r}_\beta}{\partial \delta \mathbf{l}_\omega} = -\mathbf{J}_{\beta l_\omega} \tag{B.96}$$

$$\frac{\partial \mathbf{r}_\theta}{\partial \delta \mathbf{l}_\omega} = -\left(\mathbf{I} + \frac{1}{2}\left[\mathbf{r}_\theta\right]_\times\right)\mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\mathbf{R}_{\bar{\gamma}_{ij}}\mathbf{J}_{\theta l_\omega} \tag{B.97}$$

B.3.6 IMU Intrinsics Factor

A single IMU intrinsics factor is added to the sliding window optimization problem if we wish to estimate IMU intrinsic parameters jointly with the other optimization variables. The residual is given by

$$\mathbf{r}_{\text{imucalib}}(\mathbf{l}_a,\mathbf{l}_\omega) = \begin{bmatrix} \mathbf{l}_a - \bar{\mathbf{l}}_a \\ \mathbf{l}_\omega - \bar{\mathbf{l}}_\omega \end{bmatrix} \in \mathbb{R}^{15} \tag{B.98}$$

where $\bar{\mathbf{l}}_a$ and $\bar{\mathbf{l}}_\omega$ are the current estimates of the IMU intrinsic parameters at the beginning of the optimization. The corresponding Jacobian is $\mathbf{I}_{15\times 15}$. The covariance for the IMU intrinsics residual is a 15×15 diagonal matrix with constant user-defined values.

[1] The motion delta error dynamics are given by (3.64) evaluated with $\mathbf{F}_k$ and $\mathbf{G}_k$ defined in Appendix B.3.3.

Appendix C

Nonlinear Least Squares Regression

A nonlinear least squares problem over a Euclidean parameter space $\mathbf{x} \in \mathbb{R}^n$ can be expressed in the following form:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{r}(\mathbf{x})\|^2 \tag{C.1}$$

$$\mathbf{r}(\mathbf{x}) = \begin{bmatrix} r_1(\mathbf{x}) & \dots & r_m(\mathbf{x}) \end{bmatrix}^{\mathrm{T}} \in \mathbb{R}^m \tag{C.2}$$

Define the Jacobian of $\mathbf{r}$ with respect to $\mathbf{x}$ as

$$\mathbf{J}(\mathbf{x}) = \frac{\partial \mathbf{r}}{\partial \mathbf{x}} = \begin{bmatrix} \frac{\partial r_1}{\partial x_1}\Big|_{\mathbf{x}} & \cdots & \frac{\partial r_1}{\partial x_n}\Big|_{\mathbf{x}} \\ \vdots & \ddots & \vdots \\ \frac{\partial r_m}{\partial x_1}\Big|_{\mathbf{x}} & \cdots & \frac{\partial r_m}{\partial x_n}\Big|_{\mathbf{x}} \end{bmatrix} \in \mathbb{R}^{m \times n} \tag{C.3}$$

and the gradient as

$$\mathbf{g}(\mathbf{x}) = \mathbf{J}(\mathbf{x})^{\mathrm{T}} \mathbf{r}(\mathbf{x}) \in \mathbb{R}^n \tag{C.4}$$

Nonlinear least squares optimizers seek to find a local minimum $\mathbf{x}^*$ by minimizing a sequence of quadratic functions that locally approximate the original nonlinear function. Given an initial guess $\mathbf{x}_0$, the first-order approximation of $\mathbf{r}(\mathbf{x})$ in the vicinity is $\mathbf{r}(\mathbf{x}_0) + \mathbf{J}(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0)$. Solve the linear least squares problem

$$\Delta\mathbf{x} = \arg\min_{\Delta\mathbf{x}'} \frac{1}{2} \left\|\mathbf{J}(\mathbf{x}_0)\Delta\mathbf{x}' + \mathbf{r}(\mathbf{x}_0)\right\|^2 \tag{C.5}$$

to compute a step $\Delta\mathbf{x}$ and use it to update the initial guess as $\mathbf{x}_1 = \mathbf{x}_0 + \Delta\mathbf{x}$. This process repeats until $\Delta\mathbf{x}$ is very small (i.e. convergence) or the iteration limit is reached.

C.1 Trust Region Minimization

Trust region methods control the size of $\Delta\mathbf{x}^*$ to improve convergence. They solve a modified version of (C.5) subject to the constraint that restricts the step to lie within a neighborhood of the existing estimate called the *trust region*.

$$\begin{aligned} \Delta\mathbf{x} = \arg\min_{\Delta\mathbf{x}'} & \frac{1}{2} \left\|\mathbf{J}(\mathbf{x})\Delta\mathbf{x}' + \mathbf{r}(\mathbf{x})\right\|^2 \\ & \|\mathbf{D}(\mathbf{x})\Delta\mathbf{x}\|^2 \leq \mu \end{aligned} \tag{C.6}$$

μ is the trust region radius, while $\mathbf{D}(\mathbf{x})$ is a typically diagonal matrix that rescales the dimensions of the input space. After solving the constrained optimization, a step quality factor is computed based on how closely the quadratic approximation predicted the actual objective function change in the trust region. μ is increased to enable larger step sizes if step quality factor is high and decreased to restrict step sizes if step quality factor is low. The trust region inequality constraint can be incorporated into the linear least squares cost function (C.6) using μ^{-1} as a Lagrange multiplier. The extra term corresponding to the soft constraint can be eliminated by concatenating $\frac{1}{\sqrt{\mu}}\mathbf{D}(\mathbf{x})$ below $\mathbf{J}$ and concatenating $\mathbf{0} \in \mathbb{R}^n$ below $\mathbf{r}$.

C.2 Dogleg Strategy

The Dogleg strategy computes two step candidates for (C.6):

$$\Delta\mathbf{x}_{\text{GN}} = \arg\min_{\Delta\mathbf{x}'} \frac{1}{2} \left\|\mathbf{J}(\mathbf{x})\Delta\mathbf{x}' + \mathbf{r}(\mathbf{x})\right\|^2 \tag{C.7}$$

$$\Delta\mathbf{x}_{\text{Cauchy}} = -\frac{\|\mathbf{g}(\mathbf{x})\|^2}{\|\mathbf{J}(\mathbf{x})\mathbf{g}(\mathbf{x})\|^2}\mathbf{g}(\mathbf{x}) \tag{C.8}$$

The Gauss-Newton step (C.7) is the solution to the unconstrained linear least squares problem (C.5), while the Cauchy step (C.8) is the minimum of the approximated objective function in the direction of steepest descent (i.e. the negative gradient direction).

If $\Delta\mathbf{x}_{\text{GN}}$ is inside the trust region, the Dogleg strategy uses it as the trust region step. If both $\Delta\mathbf{x}_{\text{GN}}$ and $\Delta\mathbf{x}_{\text{Cauchy}}$ are outside the trust region, the Dogleg strategy chooses a step parallel to $\Delta\mathbf{x}_{\text{Cauchy}}$ as large as the trust region radius. If $\Delta\mathbf{x}_{\text{GN}}$ is outside the trust region but $\Delta\mathbf{x}_{\text{Cauchy}}$ is inside the trust region, the Dogleg strategy constructs the trust region step by finding a linear combination of $\Delta\mathbf{x}_{\text{Cauchy}}$ and $\Delta\mathbf{x}_{\text{GN}}$ that intersects the trust region boundary.

C.3 Schur Complement Linear Solver

The linear least squares problem (C.7) for finding the Gauss-Newton step in the Dogleg strategy is equivalent to a linear system called the *normal equations*:

$$\mathbf{J}(\mathbf{x})^{\mathrm{T}}\mathbf{J}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{J}(\mathbf{x})^{\mathrm{T}}\mathbf{r}(\mathbf{x}) \tag{C.9}$$

$$\mathbf{H}(\mathbf{x})\Delta\mathbf{x} = -\mathbf{g}(\mathbf{x}) \tag{C.10}$$

where $\mathbf{H}(\mathbf{x})$ is the Hessian and $\mathbf{g}(\mathbf{x})$ is the gradient. We rewrite (C.10) as

$$\mathbf{A}\Delta\mathbf{x} = \mathbf{b} \tag{C.11}$$

to reduce clutter. The Schur complement method takes advantage of the structure of bundle adjustment problems, where the parameter vector can be partitioned into camera poses $\Delta\mathbf{x}_c$ and feature locations $\Delta\mathbf{x}_f$. Substituting these expressions into (C.11) and decomposing $\mathbf{A}$ and $\mathbf{b}$ into sub-blocks yields:

$$\begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{C}^{\mathrm{T}} & \mathbf{D} \end{bmatrix} \begin{bmatrix} \Delta\mathbf{x}_c \\ \Delta\mathbf{x}_f \end{bmatrix} = \begin{bmatrix} \mathbf{c} \\ \mathbf{d} \end{bmatrix} \tag{C.12}$$

Using block elimination, we have

$$\left(\mathbf{B} - \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{\mathrm{T}}\right)\Delta\mathbf{x}_c = \mathbf{c} - \mathbf{C}\mathbf{D}^{-1}\mathbf{d} \tag{C.13}$$

$$\mathbf{D}\Delta\mathbf{x}_f = \left(\mathbf{d} - \mathbf{C}^{\mathrm{T}}\Delta\mathbf{x}_c\right) \tag{C.14}$$

Because bundle adjustment cost function terms never involve more than one feature, $\mathbf{D}$ is block diagonal and $\mathbf{D}^{-1}$ is relatively easy to invert. The Schur complement method first solves (C.13) to get $\Delta\mathbf{x}_c$ and then

back-substitutes $\Delta\mathbf{x}_c$ into (C.14) to solve for $\Delta\mathbf{x}_f$. In time-critical applications, both (C.13) and (C.14) are usually solved via Cholesky factorization followed by back-substitution.

Since the Schur complement $\mathbf{B} - \mathbf{C}\mathbf{D}^{-1}\mathbf{C}^{\mathrm{T}}$ does not have any special sparsity pattern, solving (C.13) ($\mathcal{O}(\dim(\Delta\mathbf{x}_c)^3)$ time complexity) is the dominant cost. This compares favorably with the naive method of directly solving (C.11), which has $\mathcal{O}(\dim(\Delta\mathbf{x})^3)$ time complexity, because typical bundle adjustment problems involve much fewer cameras than features (i.e. $\dim(\Delta\mathbf{x}_c) \ll \dim(\Delta\mathbf{x})$).

C.4 Loss Functions

Loss functions are used to reduce the influence of outlier observations on the optimization solution by down-weighting their large residuals. A loss function $\lambda(s) : \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ satisfies the following conditions

$$\begin{aligned} \lambda(0) &= 0 \\ \lambda'(0) &= 1 \\ \lambda'(s) &\geq 0 \end{aligned} \tag{C.15}$$

Additionally, loss functions are typically sublinear for values of s large enough to be considered outliers:

$$\lambda'(s) < 1 \tag{C.16}$$

$$\lambda''(s) < 0 \tag{C.17}$$

Loss functions correspond to residual error probability distributions with longer tails than the Gaussian distribution and therefore assign greater probability to outliers.

Applying a loss function to a cost term $0.5\left\|\mathbf{r}(\mathbf{x})\right\|^2$ turns it into $0.5\lambda(\left\|\mathbf{r}(\mathbf{x})\right\|^2) \equiv 0.5\lambda$. With this change, the robustified gradient and Hessian from (C.10) become [91]

$$\mathbf{g}_\lambda(\mathbf{x}) = \lambda'\mathbf{J}^{\mathrm{T}}(\mathbf{x})\mathbf{r}(\mathbf{x}) \tag{C.18}$$

$$\mathbf{H}_\lambda(\mathbf{x}) \approx \mathbf{J}^{\mathrm{T}}(\mathbf{x})\left(\lambda' + 2\lambda''\mathbf{r}(\mathbf{x})\mathbf{r}(\mathbf{x})^{\mathrm{T}}\right)\mathbf{J}(\mathbf{x}) \tag{C.19}$$

The condition (C.16) down-weights the gradient and Hessian of an outlier residual, while (C.17) reduces the Hessian's curvature.

In order to use loss functions with the optimization techniques described in previous sections, replace the non-robust residual ($\mathbf{r}(\mathbf{x})$) and Jacobian ($\mathbf{J}(\mathbf{x})$) with their robust counterparts [91]:

$$\mathbf{r}_\lambda(\mathbf{x}) = \frac{\sqrt{\lambda'}}{1-\alpha}\mathbf{r}(\mathbf{x}) \tag{C.20}$$

$$\mathbf{J}_\lambda(\mathbf{x}) = \sqrt{\lambda'}\left(1 - \alpha\frac{\mathbf{r}(\mathbf{x})\mathbf{r}(\mathbf{x})^{\mathrm{T}}}{\left\|\mathbf{r}(\mathbf{x})\right\|^2}\right)\mathbf{J}(\mathbf{x}) \tag{C.21}$$

$$\alpha = \begin{cases} 0 & \text{if } \lambda'' < 0 \\ 1 - \sqrt{1 + 2\frac{\lambda''}{\lambda'}\left\|\mathbf{r}(\mathbf{x})\right\|^2} & \text{otherwise} \end{cases} \tag{C.22}$$

C.5 Manifold Optimization

The previous sections described nonlinear least squares optimization over the Euclidean parameter space $\mathbb{R}^n$. Suppose the optimization variable lies on a manifold $\mathcal{M}$ with fewer degrees of freedom ($p < n$) than the ambient vector space ($\mathbb{R}^n$) it is embedded in. In such cases it is necessary to perform optimization on the manifold's tangent space, $T(\mathcal{M}) = \mathbb{R}^p$, in order to prevent the normal equations (C.9) from becoming underdetermined.

Table C.1: Comparison of optimization in Euclidean space and manifold

Optimization	**Euclidean Space**	**Manifold**
Problem	$\mathbf{x}^* = \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x})$	$\delta\mathbf{x}^* = \arg\min_{\delta\mathbf{x}\in\mathbb{R}^p} f(\mathbf{x} \boxplus \delta\mathbf{x})$
Update	$\mathbf{x} \leftarrow \mathbf{x} + \mathbf{x}^*$	$\mathbf{x} \leftarrow \mathbf{x} \boxplus \delta\mathbf{x}^*$

Given the current estimate $\mathbf{x} \in \mathcal{M}$, the $\boxplus$ operator is a generalization of Euclidean space vector addition that maps elements in the tangent space of $\mathcal{M}$ at $\mathbf{x}$ to the neighborhood of $\mathbf{x}$. Because tangent spaces are Euclidean, we can apply the standard trust region step (Sect. C.1) to find the tangent space vector $\delta\mathbf{x} \in \mathbb{R}^p$ that minimizes the locally approximated objective function. The current estimate is updated using the $\boxplus$ with the optimal tangent space vector to ensure that the result remains in $\mathcal{M}$.

A common application of manifold optimization involves quaternions in bundle adjustment. Quaternions are embedded in a 4-dimensional ambient vector space but represent $SO(3)$ rotations that only have 3 degrees of freedom. The $\boxplus$ operator for quaternions can be defined either as a left-perturbation

$$\mathbf{q} \boxplus \delta\boldsymbol{\theta} = \mathrm{Exp}(\boldsymbol{\theta}) \otimes \mathbf{q} \approx \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{\theta} \end{bmatrix} \otimes \mathbf{q} \tag{C.23}$$

or a right-perturbation

$$\mathbf{q} \boxplus \delta\boldsymbol{\theta} = \mathbf{q} \otimes \mathrm{Exp}(\boldsymbol{\theta}) \approx \mathbf{q} \otimes \begin{bmatrix} 1 \\ \frac{1}{2}\boldsymbol{\theta} \end{bmatrix} \tag{C.24}$$

using the capitalized quaternion exponential map (A.27).

Appendix D

Offline Calibration

D.1 IMU to Motion Capture Model Extrinsics

Let frame $\{w\}$ be the fixed reference frame in which a motion capture system provides its observations. The motion capture system provides position ($\mathbf{p}_{wb}$) and rotation ($\mathbf{R}_{wb}$) observations of a robot's marker centroid frame $\{b\}$ (henceforth called the mocap body frame) relative to $\{w\}$. The robot receives IMU linear acceleration ($\mathbf{a}_i$) and angular velocity ($\boldsymbol{\omega}_i$) observations in the IMU frame $\{i\}$, which in general is not the same as $\{b\}$. This section describes a method to compute the rigid body transform between $\{i\}$ and $\{b\}$ using IMU and mocap observations collected during a period of dynamic motion. We formulate a nonlinear least squares problem to find $\mathbf{R}_{ib}$ and $\mathbf{p}_{ib}$ by minimizing the difference between IMU acceleration expressed in frame $\{b\}$ and mocap-derived acceleration expressed in frame $\{b\}$.

Mocap-Derived Body Acceleration

Double differentiate mocap position observations $\mathbf{p}_{wb}$ to obtain the world frame coordinate acceleration of $\{b\}$.

$$\mathbf{a}_w = \ddot{\mathbf{p}}_{wb} \tag{D.1}$$

We use a Savitzky-Golay filter [92] to smooth the data before taking derivatives in order to suppress noise. The Savitzky-Golay filter window size is left as a tuning parameter to control the level of smoothing and should be set according to the amount of noise in the raw data. To obtain proper acceleration in the body frame, we need to add gravitational acceleration and rotate the sum from the world frame to the body frame.

$$\mathbf{a}_{b,\text{mocap}} = \mathbf{R}_{wb}^{\mathrm{T}} \left(\mathbf{a}_w + g\mathbf{e}_3 \right) \tag{D.2}$$

IMU Acceleration in Mocap-Derived Body Frame

We compute the measured IMU acceleration as seen in the mocap-derived body frame according to equation 2 in [93].

$$\mathbf{a}_{b,\text{imu}} = \mathbf{R}_{ib}^{\mathrm{T}} \left[\mathbf{a}_i + \dot{\boldsymbol{\omega}}_i \times \mathbf{p}_{ib} + \boldsymbol{\omega}_i \times (\boldsymbol{\omega}_i \times \mathbf{p}_{ib}) \right] \tag{D.3}$$

IMU angular acceleration $\dot{\boldsymbol{\omega}}_i$ is computed from angular velocity via Savitzky-Golay differentiation, with window size being a user-tuned parameter chosen for noise suppression.

Nonlinear Least Squares Problem

We form a cost function can by summing the squared norms of the differences between $(\mathbf{a}_{b,\text{mocap}}, \mathbf{a}_{b,\text{imu}})$ pairs,

$$f_0 \left(\mathbf{p}_{ib}, \mathbf{q}_{ib} \right) = \frac{1}{2} \sum_{k=1}^{n} \left\| \mathbf{a}_{b,\text{imu}}(\mathbf{p}_{ib}, \mathbf{q}_{ib}, k) - \mathbf{a}_{b,\text{mocap}}(k) \right\|^2, \tag{D.4}$$

where $\mathbf{q}_{ib}$ is the Hamilton quaternion representation of $\mathbf{R}_{ib}$.

The jacobian of the residual in (D.4) is nontrivial because both $\mathbf{q}_{ib}$ and $\mathbf{p}_{ib}$ appear in the first term (D.3). To simplify the jacobian, multiply each residual by $\mathbf{R}_{ib}$ so that each term in the residual involves only one optimization variable. To save space we omit the index k from the expanded residual expression.

$$f_1\left(\mathbf{p}_{ib}, \mathbf{q}_{ib}\right) = \frac{1}{2}\sum_{k=1}^{n}\left\|\mathbf{a}_i + \dot{\boldsymbol{\omega}}_i \times \mathbf{p}_{ib} + \boldsymbol{\omega}_i \times \left(\boldsymbol{\omega}_i \times \mathbf{p}_{ib}\right) - \mathbf{R}(\mathbf{q}_{ib})\, \mathbf{a}_{b,\text{mocap}}\right\|^2 \tag{D.5}$$

$$f_1\left(\mathbf{p}_{ib}, \mathbf{q}_{ib}\right) = \frac{1}{2}\sum_{k=1}^{n}\left\|\mathbf{a}_i + \left(\dot{\boldsymbol{\omega}}_{i,\times} + \boldsymbol{\omega}_{i,\times}\boldsymbol{\omega}_{i,\times}\right)\mathbf{p}_{ib} - \mathbf{R}(\mathbf{q}_{ib})\, \mathbf{a}_{b,\text{mocap}}\right\|^2 \tag{D.6}$$

$$f_1\left(\mathbf{p}_{ib}, \mathbf{q}_{ib}\right) = \frac{1}{2}\sum_{k=1}^{n}\left\|\mathbf{r}(\mathbf{p}_{ib}, \mathbf{q}_{ib}, k)\right\|^2 \tag{D.7}$$

The jacobians of the residual $\mathbf{r}(\mathbf{p}_{ib}, \mathbf{q}_{ib}, k)$ with respect to $\mathbf{p}_{ib}$ and the Lie algebra of $\mathbf{q}_{ib}$ are

$$\frac{\partial \mathbf{r}}{\partial \mathbf{p}_{ib}} = \dot{\boldsymbol{\omega}}_{i,\times} + \boldsymbol{\omega}_{i,\times}\boldsymbol{\omega}_{i,\times} \tag{D.8}$$

$$\frac{\partial \mathbf{r}}{\partial \boldsymbol{\theta}_{ib}} = \mathbf{R}(\mathbf{q}_{ib})\, \mathbf{a}_{b,\text{mocap},\times} \tag{D.9}$$

Use the above residual and jacobian expressions in a nonlinear least squares optimizer to compute the $\mathbf{p}_{ib}$ and $\mathbf{q}_{ib}$ values that minimize f_1.

Implementation Tip

A naive strategy to select n $(\mathbf{a}_{b,\text{mocap}}, \mathbf{a}_{b,\text{imu}})$ pairs from flight data is to choose them to be equally spaced in time. A more efficient strategy is to compute the IMU acceleration norm over the entire flight duration and choose n pairs from the time intervals when the acceleration norm exceeds a threshold. This strategy biases the subsampling of raw data to favor time intervals of dynamic motion, which provide greater observability of the IMU to mocap body frame transform.

D.2 Camera to IMU Extrinsics

Unlike extrinsic rotation and time offset, estimating extrinsic translation requires visual scale to either be known beforehand or jointly estimated. The former can be achieved by using a known calibration target, while the latter is used in target-free calibration. In both cases, IMU biases and noise must be accurately modeled to obtain physically meaningful estimates. Since collecting enough data to model long-term noise characteristics of an IMU subject to flight-induced vibrations is difficult, we were unable to achieve reasonable calibrations with tools such as Kalibr [94] that jointly estimate spatial and temporal calibration parameters. Consequently, we designed procedures to calibrate extrinsic rotation and time offset independently of extrinsic translation and rely on CAD model measurements for extrinsic translation.

Let $\{i\}$ be the IMU reference frame and $\{c\}$ be the camera reference frame. We compute the extrinsic rotation $\mathbf{R}_{ic} \in SO(3)$ by aligning camera and gyro delta rotations in corresponding time intervals. We compute the time offset between the IMU and camera by wrapping the extrinsic rotation calibration procedure inside a line search to find the time offset value for which alignment error is minimized.

Delta Rotation From Feature Correspondences

Let t_k and t_{k+1} be two successive times when feature observations are published. The k^{th} feature observation set consists of n_k pixel coordinates $\{(\text{ID}_i, (u_{ki}, v_{ki}))\}_{i=1}^{n_k}$ associated with unique feature IDs. The

homogeneous point representation of (u_{ki}, v_{ki}) is given by

$$\mathbf{p}_{ki} = \mathbf{K}^{-1} \begin{bmatrix} u_{ki} \\ v_{ki} \\ 1 \end{bmatrix}, \tag{D.10}$$

where

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{D.11}$$

is the camera's intrinsic matrix. We normalize each homogeneous point to turn it into a bearing observation

$$\bar{\mathbf{p}}_{ki} = \frac{\mathbf{p}_{ki}}{\|\mathbf{p}_{ki}\|}. \tag{D.12}$$

We construct a set of $m \leq \min(n_k, n_{k+1})$ bearing observation pairs from times t_k and t_{k+1} with matching feature IDs,

$$\{\bar{\mathbf{p}}_{k,j}\}_{j=1}^{m}, \quad \{\bar{\mathbf{p}}_{k+1,j}\}_{j=1}^{m}, \tag{D.13}$$

and use the Kabsch algorithm [53] to find the delta rotation $\mathbf{R}_{\text{cam},k,k+1}$ between them.

Delta Rotation From Integrated Angular Velocity

Let $\{\boldsymbol{\omega}_j\}_{j=0}^{n}$ be the set of all IMU gyro observations whose timestamps are within the interval $[t_k, t_{k+1}]$ between two successive feature observation sets. The timestamps corresponding to these gyro observations are $\{t_j\}_{j=0}^{n}$, where $t_0 = t_k$ and $t_n = t_{k+1}$ [1]. We use midpoint integration to compute the delta rotation arising from the set of angular velocities,

$$\mathbf{q}_{k,k+1} = \prod_{j=1}^{n} \begin{bmatrix} 1 \\ \frac{1}{2}(t_j - t_{j-1}) \frac{\boldsymbol{\omega}_{j-1} + \boldsymbol{\omega}_j}{2} \end{bmatrix}, \tag{D.14}$$

where $\prod_{i=1}^{n} \mathbf{q}_i = \mathbf{q}_1 \otimes \ldots \otimes \mathbf{q}_n$. The rotation matrix form of the quaternion $\mathbf{q}_{k,k+1}$ is

$$\mathbf{R}_{\text{gyro},k,k+1} = \text{QuatToR}(\mathbf{q}_{k,k+1}) \tag{D.15}$$

where QuatToR denotes the function defined in (A.20).

Rigid Body Alignment of Vector Sets

The Kabsch algorithm [53] finds the rigid body transformation that minimizes the root-mean-square (RMS) deviation between two sets of D-dimensional vectors. In 3D, the Kabsch algorithm finds the rotation matrix and translation vector that maps each vector in the set $\{\mathbf{a}_i\}_{i=1}^{n}$ as closely as possible to its counterpart in $\{\mathbf{b}_i\}_{i=1}^{n}$.

First, compute the centroid of each set.

$$\bar{\mathbf{a}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{a}_i \tag{D.16}$$

$$\bar{\mathbf{b}} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{b}_i \tag{D.17}$$

[1] Interpolated observations should be used for t_0 and t_n if no IMU observations align perfectly with t_k and t_{k+1}

Define matrices of stacked deviation row vectors.

$$\mathbf{A} = \begin{bmatrix} (\mathbf{a}_1 - \bar{\mathbf{a}})^{\mathrm{T}} \\ \vdots \\ (\mathbf{a}_i - \bar{\mathbf{a}})^{\mathrm{T}} \\ \vdots \\ (\mathbf{a}_n - \bar{\mathbf{a}})^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{n \times 3}, \qquad \mathbf{B} = \begin{bmatrix} \left(\mathbf{b}_1 - \bar{\mathbf{b}}\right)^{\mathrm{T}} \\ \vdots \\ \left(\mathbf{b}_i - \bar{\mathbf{b}}\right)^{\mathrm{T}} \\ \vdots \\ \left(\mathbf{b}_n - \bar{\mathbf{b}}\right)^{\mathrm{T}} \end{bmatrix} \in \mathbb{R}^{n \times 3} \tag{D.18}$$

Construct the cross covariance matrix.

$$\mathbf{C} = \mathbf{A}^{\mathrm{T}}\mathbf{B} \tag{D.19}$$

Perform a singular value decomposition of $\mathbf{C}$.

$$\mathbf{C} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}} \tag{D.20}$$

Apply a correction factor if $\mathbf{V}\mathbf{U}^{\mathrm{T}}$ has a determinant of -1 to ensure that the optimal rotation $\mathbf{R}$ has a determinant of +1 (i.e. it is a rotation between right-handed coordinate systems).

$$d = \det\left(\mathbf{V}\mathbf{U}^{\mathrm{T}}\right) \tag{D.21}$$

$$\mathbf{R} = \mathbf{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & d \end{bmatrix} \mathbf{U}^{\mathrm{T}} \tag{D.22}$$

Obtain the translation vector

$$\mathbf{t} = \bar{\mathbf{b}} - \mathbf{R}\bar{\mathbf{a}} \tag{D.23}$$

The RMS deviation between $\{\mathbf{b}_i\}_{i=1}^n$ and the transformed $\{\mathbf{a}_i\}_{i=1}^n$ is

$$RMSD = \sum_{i=1}^{n} \|\mathbf{R}\mathbf{a}_i + \mathbf{t} - \mathbf{b}_i\|^2 \tag{D.24}$$

Axis Angle Delta Rotation Alignment

First, express corresponding pairs of delta rotations $\left\{\left(\mathbf{R}_{\text{cam},k,k+1}, \mathbf{R}_{\text{gyro},k,k+1}\right)\right\}_{k=1}^N$, in the angle-axis parametrization as $\left\{\left(\mathbf{a}_{\text{cam},k,k+1}, \mathbf{a}_{\text{gyro},k,k+1}\right)\right\}_{k=1}^N$. The relationship between the angle-axis parameterization and the rotation matrix is given by

$$\theta = \arccos\left[\frac{1}{2}\left(\text{trace}\mathbf{R} - 1\right)\right] \tag{D.25}$$

$$\mathbf{a}(\mathbf{R}) = \begin{cases} \mathbf{0}_{3\times 1} & \text{if } |\theta| \ll 1 \\ \frac{\theta}{2\sin\theta} \begin{bmatrix} R_{32} - R_{23} \\ R_{13} - R_{31} \\ R_{21} - R_{12} \end{bmatrix} & \text{otherwise} \end{cases} \tag{D.26}$$

Although angle-axis parameters are not vectors because their addition is non-commutative, they can be treated as vectors when the rotation angle θ is small. With this assumption, we apply the Kabsch algorithm to $\{\mathbf{a}_{\text{cam},k,k+1}\}_{k=1}^N$ and $\{\mathbf{a}_{\text{gyro},k,k+1}\}_{k=1}^N$ to obtain $\mathbf{R}_{ic}$.

Time Offset

The above described procedure can be run repeatedly with different time offset values to find the time offset with the lowest alignment cost. Because the range of possible time offset values is not too large, a simple 1D optimization method such as bisection or golden section search suffices to find the best time offset value.

Appendix E

Online Calibration

E.1 Camera IMU Extrinsic Transform

This section uses the notation introduced in Section 3.4.

If we are interested in jointly optimizing the extrinsic transform between the camera and body alongside the body frame motion states and features, we require Jacobians with respect to the extrinsic translation, $\mathbf{p}_{bc}$, and rotation, $\mathbf{q}_{bc}$. Since $\mathbf{p}_{bc}$ is a vector space parameter block, we obtain the Jacobian of $\mathbf{a}$ with respect to it by expanding out (3.141), isolating the terms that use it, and then taking the derivative.

$$\frac{\partial \mathbf{a}}{\partial \mathbf{p}_{bc}} = \mathbf{R}(\mathbf{q}_{bc})^{\mathrm{T}} \left[\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\mathbf{R}(\mathbf{q}_i) - \mathbf{I}\right] \tag{E.1}$$

Instead of taking a Jacobian with respect to the quaternion $\mathbf{q}_{bc}$, we take a Jacobian with respect to the minimal parameterization of its perturbation $\delta\boldsymbol{\theta}_{bc}$. We use the perturbation method to obtain the Jacobian of $\mathbf{a}$ with respect to $\delta\boldsymbol{\theta}_{bc}$. First rewrite (3.141) and consolidate terms that are not $\mathbf{R}_{bc}$.

$$\mathbf{a} = \mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_j^{\mathrm{T}}\mathbf{R}_i\mathbf{R}_{bc}\frac{1}{\rho_l}\mathbf{h}_{l,i} + \mathbf{R}_{bc}^{\mathrm{T}}\left(\mathbf{R}_j^{\mathrm{T}}\left(\mathbf{R}_i\mathbf{p}_{bc} + \mathbf{p}_i - \mathbf{p}_j\right) - \mathbf{p}_{bc}\right) \tag{E.2}$$

$$\mathbf{a} = \mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}\mathbf{p}_{l,c_i} + \mathbf{R}_{bc}^{\mathrm{T}}\Delta\mathbf{p} \tag{E.3}$$

Apply an additive perturbation to the left hand side of (E.3) and a rotational perturbation to the base rotation $\mathbf{R}_{bc} = \mathbf{R}(\mathbf{q}_{bc})$ on the right hand side of (E.3).

$$\mathbf{a} + \delta\mathbf{a} = \left[\mathbf{R}_{bc}\operatorname{Exp}(\delta\boldsymbol{\theta}_{bc})\right]^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}\operatorname{Exp}(\delta\boldsymbol{\theta}_{bc})\mathbf{p}_{l,c_i} + \left[\mathbf{R}_{bc}\operatorname{Exp}(\delta\boldsymbol{\theta}_{bc})\right]^{\mathrm{T}}\Delta\mathbf{p} \tag{E.4}$$

Assuming the rotational perturbation $\delta\boldsymbol{\theta}_{bc}$ is small, use the approximation (A.14)

$$\mathbf{a} + \delta\mathbf{a} \approx \left[\mathbf{R}_{bc}\left(\mathbf{I} + [\delta\boldsymbol{\theta}_{bc})]_\times\right)\right]^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}\left(\mathbf{I} + [\delta\boldsymbol{\theta}_{bc})]_\times\right)\mathbf{p}_{l,c_i} + \left[\mathbf{R}_{bc}\left(\mathbf{I} + [\delta\boldsymbol{\theta}_{bc})]_\times\right)\right]^{\mathrm{T}}\Delta\mathbf{p} \tag{E.5}$$

$$\mathbf{a} + \delta\mathbf{a} \approx \left(\mathbf{I} + [\delta\boldsymbol{\theta}_{bc}]_\times^{\mathrm{T}}\right)\mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}\left(\mathbf{I} + [\delta\boldsymbol{\theta}_{bc})]_\times\right)\mathbf{p}_{l,c_i} + \left(\mathbf{I} + [\delta\boldsymbol{\theta}_{bc}]_\times^{\mathrm{T}}\right)\mathbf{R}_{bc}^{\mathrm{T}}\Delta\mathbf{p} \tag{E.6}$$

$$\mathbf{a} + \delta\mathbf{a} \overset{(A.4)}{\approx} \left(\mathbf{I} - [\delta\boldsymbol{\theta}_{bc}]_\times\right)\mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}\left(\mathbf{I} + [\delta\boldsymbol{\theta}_{bc})]_\times\right)\mathbf{p}_{l,c_i} + \left(\mathbf{I} - [\delta\boldsymbol{\theta}_{bc}]_\times\right)\mathbf{R}_{bc}^{\mathrm{T}}\Delta\mathbf{p} \tag{E.7}$$

$$\delta\mathbf{a} \overset{(E.3)}{\approx} -[\delta\boldsymbol{\theta}_{bc}]_\times\mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}\mathbf{p}_{l,c_i} + \mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}[\delta\boldsymbol{\theta}_{bc}]_\times\mathbf{p}_{l,c_i} - [\delta\boldsymbol{\theta}_{bc}]_\times\mathbf{R}_{bc}^{\mathrm{T}}\Delta\mathbf{p} \tag{E.8}$$

Note that in (E.8) we ignore terms that are second order in $\delta\boldsymbol{\theta}_{bc}$.

$$\delta\mathbf{a} \approx -[\delta\boldsymbol{\theta}_{bc}]_\times\left[\mathbf{R}_{bc}^{\mathrm{T}}\left(\mathbf{R}_{ji}\mathbf{R}_{bc}\mathbf{p}_{l,c_i} + \Delta\mathbf{p}\right)\right] + \mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}[\delta\boldsymbol{\theta}_{bc}]_\times\mathbf{p}_{l,c_i} \tag{E.9}$$

$$\delta\mathbf{a} \overset{(A.3)}{\approx} \left[\mathbf{R}_{bc}^{\mathrm{T}}\left(\mathbf{R}_{ji}\mathbf{R}_{bc}\mathbf{p}_{l,c_i} + \Delta\mathbf{p}\right)\right]_\times\delta\boldsymbol{\theta}_{bc} - \mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}_{ji}\mathbf{R}_{bc}\left[\mathbf{p}_{l,c_i}\right]_\times\delta\boldsymbol{\theta}_{bc} \tag{E.10}$$

$$\delta\mathbf{a} \approx \left(\left[\mathbf{R}_{bc}^{\mathrm{T}} \left(\mathbf{R}_{ji} \mathbf{R}_{bc} \mathbf{p}_{l,c_i} + \Delta\mathbf{p} \right) \right]_{\times} - \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}_{ji} \mathbf{R}_{bc} \left[\mathbf{p}_{l,c_i} \right]_{\times} \right) \delta\boldsymbol{\theta}_{bc} \tag{E.11}$$

The approximate Jacobian of $\mathbf{a}$ with respect to $\delta\boldsymbol{\theta}_{bc}$ is

$$\frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_{bc}} \approx \left[\mathbf{R}_{bc}^{\mathrm{T}} \left(\mathbf{R}_{ji} \mathbf{R}_{bc} \mathbf{p}_{l,c_i} + \Delta\mathbf{p} \right) \right]_{\times} - \mathbf{R}_{bc}^{\mathrm{T}} \mathbf{R}_{ji} \mathbf{R}_{bc} \left[\mathbf{p}_{l,c_i} \right]_{\times} \tag{E.12}$$

$$\mathbf{R}_{ji} = \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \mathbf{R}(\mathbf{q}_i) \tag{E.13}$$

$$\mathbf{p}_{l,c_i} = \frac{1}{\rho_l} \mathbf{h}_{l,i} \tag{E.14}$$

$$\Delta\mathbf{p} = \mathbf{R}(\mathbf{q}_j)^{\mathrm{T}} \left[\mathbf{R}(\mathbf{q}_i) \mathbf{p}_{bc} + \mathbf{p}_i - \mathbf{p}_j \right] - \mathbf{p}_{bc} \tag{E.15}$$

E.2 Camera IMU Time Offset

We account for time offsets between the IMU and camera timestamps in the reprojection factor using the method proposed in Qin et al. [95].

E.2.1 Feature Velocity

A key assumption is that feature observations can be approximated as moving with constant velocity on the normalized image plane over small time intervals. The feature's velocity can be empirically computed as a finite difference of the feature's observed locations in the two most recent frames[1]. Let $\mathbf{h}_l(t)$ be the homogeneous coordinate representation of the observation of feature l at time t. Let t_{m-1} and t_m be the timestamps of two successive camera frames. The finite difference feature velocity of feature l at time t is

$$\mathbf{v}_l(t_m) = \frac{\mathbf{h}_l(t_m) - \mathbf{h}_l(t_{m-1})}{t_m - t_{m-1}} \tag{E.16}$$

Note that $\begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \mathbf{v}_l = 0$ for all feature velocities because the z-component of a homogeneous point is always 1.

E.2.2 Time Offset Definition

We consider the IMU clock to be true time. Let d be the time offset[2] that must be added to camera frame timestamps in order to make them temporally consistent with IMU timestamps.

$$t_{\text{true}} = t_{\text{raw}} + d \tag{E.17}$$

In (E.17), t_{raw} is the timestamp output by the camera driver for an image observation, while t_{true} is the IMU clock time corresponding to that time instant.

E.2.3 Time Shifted Feature Observations

Consider a feature observation $\mathbf{h}_l(t)$ taken at time t whose empirical instantaneous normalized image plane velocity is $\mathbf{v}_l(t)$. Assuming that the feature moves at a constant velocity of $\mathbf{v}_l(t)$ over the time interval $[t, t + \delta t]$, at time $t + \delta t$ the predicted feature observation l is

$$\mathbf{h}_l(t + \delta t) = \mathbf{h}_l(t) + \mathbf{v}_l(t) \delta t \tag{E.18}$$

[1] Frames are much closer to together in time than keyframes and therefore provide a better approximation of instantaneous feature velocity

[2] See Appendix E.2 for an online method to estimate d.

E.2.4 Incorporating Time Offsets into the Standard Reprojection Residual

We seek to modify the standard reprojection residual (3.142) to incorporate the effect of a changing time offset estimate. Let d_i be the time offset guess immediately before keyframe i was added to the sliding window and d_j be the time offset guess immediately before keyframe j was added to the sliding window. Let $\mathbf{h}_{l,i}$ be a homogeneous point representation of observation l in keyframe i and $\mathbf{v}_{l,i}$ be its empirical instantaneous normalized image plane velocity. Let $\mathbf{h}_{l,j}$ be a homogeneous point representation of observation l in keyframe j and $\mathbf{v}_{l,j}$ be its empirical instantaneous normalized image plane velocity. Let d represent the time offset that is part of the optimization vector.

First, we note that the motion states at keyframe indices i and j correspond to IMU times $t_{\text{raw},i} + d_i$ and $t_{\text{raw},j} + d_j$, respectively, due to step 5 of Sect. E.2.6. When the time offset estimate d changes during optimization, we need to ensure that the shifted feature observations correspond to their original times $t_{\text{raw},i} + d_i$ and $t_{\text{raw},j} + d_j$ in order to maintain consistency with the IMU preintegration motion constraints. This requires feature observations to be shifted by the opposite of the change in the time offset estimate.

$$\mathbf{h}_{l,i}(t_{\text{raw},i} + d_i) = \mathbf{h}_{l,i}(t_{\text{raw},i} + d) - (d - d_i)\,\mathbf{v}_{l,i}(t_{\text{raw},i} + d) \tag{E.19}$$

$\mathbf{h}_{l,i}(t_{\text{raw},i} + d)$ is the raw, unshifted feature observation, while $\mathbf{v}_{l,i}(t_{\text{raw},i} + d)$ is its corresponding feature velocity. We associate these quantities with the compensated timestamp $t_{\text{raw},i} + d$ based on the time offset that is being optimized. The change in the time offset, $d - d_i$, is required to predict the feature observation at the time $t_{\text{raw},i} + d_i$ that is consistent with the IMU preintegration motion constraints that involve keyframe i.

Similarly to (E.19), we have

$$\mathbf{h}_{l,j}(t_{\text{raw},j} + d_j) = \mathbf{h}_{l,j}(t_{\text{raw},j} + d) - (d - d_j)\,\mathbf{v}_{l,j}(t_{\text{raw},j} + d) \tag{E.20}$$

for keyframe j. Substitute (E.19) into (3.141) to obtain

$$\mathbf{p}_{l,c_j} = \mathbf{R}_{bc}^{\mathrm{T}}\left(\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\left(\mathbf{R}(\mathbf{q}_i)\left(\mathbf{R}_{bc}\frac{1}{\rho_l}\left(\mathbf{h}_{l,i} - (d - d_i)\mathbf{v}_{l,i}\right) + \mathbf{p}_{bc}\right) + \mathbf{p}_i - \mathbf{p}_j\right) - \mathbf{p}_{bc}\right) \tag{E.21}$$

Substitute (E.20) and (E.21) into (3.142) to obtain the standard reprojection residual with time offsets,

$$\mathbf{r}_{\text{proj}}\left(\mathbf{p}_i, \mathbf{q}_i, \mathbf{p}_j, \mathbf{q}_j, \rho_l, d\right) = \mathbf{z}\left(\mathbf{p}_{l,c_j}\right) - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\left(\mathbf{h}_{l,j} - (d - d_j)\mathbf{v}_{l,j}\right) \tag{E.22}$$

E.2.5 Time Offset Jacobian

Because the time offset is a vector space quantity, we can take its Jacobian by applying chain rule and product rule on (E.21)-(E.22).

$$\frac{\partial \mathbf{r}}{\partial d} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}}\frac{\partial \mathbf{a}}{\partial d} - \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\frac{\partial}{\partial d}\left(\mathbf{h}_{l,j} - (d - d_j)\mathbf{v}_{l,j}\right) \tag{E.23}$$

$$\frac{\partial \mathbf{r}}{\partial d} = -\frac{\partial \mathbf{z}}{\partial \mathbf{a}}\mathbf{R}_{bc}^{\mathrm{T}}\mathbf{R}(\mathbf{q}_j)^{\mathrm{T}}\mathbf{R}(\mathbf{q}_i)\mathbf{R}_{bc}\frac{1}{\rho_l}\mathbf{v}_{l,i} + \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}\mathbf{v}_{l,j} \tag{E.24}$$

The image projection Jacobian $\frac{\partial \mathbf{z}}{\partial \mathbf{a}}$ is evaluated by substituting (E.21) into (3.144).

E.2.6 Estimation Procedure

When performing sliding window visual-inertial bundle adjustment for a single camera with time offset compensation, the IMU motion states for each keyframe correspond to the estimated IMU-consistent timestamps of each keyframe rather than the raw camera timestamps. The process for estimating time offsets over successive sliding optimizations is as follows:

1. Initialize the time offset to a prior guess if available or 0 if a prior guess is unavailable.
2. On the first optimization, choose keyframe motion states to correspond with the current best estimate of the compensated keyframe timestamps. For each keyframe, store the current time offset estimate when it was added to the sliding window.
3. Form IMU preintegration motion constraints between successive motion states by summarizing IMU observations that fall within pairs of successive compensated keyframe timestamps.
4. Solve an optimization problem to obtain an updated time offset estimate (along with motion states, extrinsics, and feature inverse depths).
5. Add a new keyframe to the sliding window and associate it with the updated time offset.
6. Form an IMU preintegration motion constraint between the latest original keyframe and the incoming keyframe's compensated timestamps. Note that the latest original keyframe's compensated timestamp is based on the previous time offset estimate, while the incoming keyframe's compensated timestamp is based on the updated time offset estimate.
7. Repeat steps 4-6 for as long as the state estimator is running. After each optimization the time offset should approach closer to its true value and more of the sliding window's keyframe motion states will be closer to their true timestamps. Ideally, the time offset should converge to a constant value after a few rounds of optimization as long as there is sufficient scene texture and motion excitation.

Appendix F

Interpolated Reprojection Factor

The interpolated reprojection factor derived in this appendix is related to the formulation described in Eckenhoff et. al [96] for a MSCKF. However, their work represents feature locations as 3D vectors that are marginalized out before updating motion states via a structureless smart factor. In this appendix, we derive the reprojection residual and Jacobians for a bundle adjustment problem where features are parametrized by inverse depth and explicitly included in the optimization vector. We consider the case where there is a single master camera and secondary camera in the optimization problem.

F.1 Master and Secondary Camera Keyframe Timestamps

Let $\{t_{m,1}, \ldots, t_{m,N}\}$ be the set of N monotonically increasing offset-compensated master camera keyframe timestamps. Let $t_{s,i,\text{raw}}$ and $t_{s,j,\text{raw}}$ be the camera driver timestamps associated with secondary camera keyframes with indices i and j. Let d_m denote the master camera's time offset with respect to the IMU clock and d_s denote the secondary camera's time offset with respect to the IMU clock. The compensated secondary camera keyframe timestamps for keyframes i and j are

$$t_{s,i} = t_{s,i,\text{raw}} + d_s \tag{F.1}$$

$$t_{s,j} = t_{s,j,\text{raw}} + d_s \tag{F.2}$$

Define the master camera keyframe indices that bound the secondary camera keyframe i:

$$i_a = \begin{cases} 1 & \text{if } t_{s,i} \leq t_{m,1} \\ \arg\min_k \left(t_{s,i} - t_{m,k}\right) \text{ s. t. } t_{s,i} > t_{m,k} & \text{if } t_{m,1} < t_{s,i} < t_{m,N} \\ N-1 & \text{if } t_{s,i} \geq t_{m,N} \end{cases} \tag{F.3}$$

$$i_b = \begin{cases} 2 & \text{if } t_{s,i} \leq t_{m,1} \\ \arg\min_k \left(t_{m,k} - t_{s,i}\right) \text{ s. t. } t_{m,k} > t_{s,i} & \text{if } t_{m,1} < t_{s,i} < t_{m,N} \\ N & \text{if } t_{s,i} \geq t_{m,N} \end{cases} \tag{F.4}$$

We also define j_a and j_b in an analogous manner as the master camera keyframe indices that bound the secondary camera keyframe j.

F.2 Pose Interpolation

The pose at the offset-compensated time corresponding to secondary keyframe i, $t_{s,i}$, is obtained by interpolating between or extrapolating from the poses at offset-compensated master keyframe times t_{m,i_a} and t_{m,i_b}. Note that when the compensated secondary keyframe timestamp $t_{s,i}$ falls outside of the compensated time range of the master camera keyframes, $[t_{m,1}, t_{m,N}]$, extrapolation is performed with the two closest

master keyframes instead of the two master keyframes that temporally bound $t_{s,i}$. However, mathematically the procedure for extrapolation is the same as that for interpolation, so from this point onward we use the term interpolation to refer to both interpolation and extrapolation.

We define λ_i as

$$\lambda_i \triangleq \frac{t_{s,i} - t_{m,i_a}}{t_{m,i_b} - t_{m,i_a}} \tag{F.5}$$

Expanding out (F.5) with raw timestamps and time offset estimates, we obtain

$$\lambda_i \triangleq \frac{t_{s,i,\text{raw}} + d_s - t_{m,i_a,\text{raw}} - d_m}{t_{m,i_b,\text{raw}} - t_{m,i_a,\text{raw}}} \tag{F.6}$$

We obtain the position at time $t_{s,i}$ via linear interpolation:

$$\mathbf{p}(t_{s,i}) = (1 - \lambda_i)\, \mathbf{p}(t_{m,i_a}) + \lambda_i \mathbf{p}(t_{m,i_b}) \tag{F.7}$$

We obtain the orientation at time $t_{s,i}$ via $SO(3)$ interpolation:

$$\mathbf{R}(t_{s,i}) = \text{Exp}\left[\lambda_i \,\text{Log}\left(\mathbf{R}(t_{m,i_b})\mathbf{R}(t_{m,i_a})^{\text{T}}\right)\right] \mathbf{R}(t_{m,i_a}) \tag{F.8}$$

This can be represented in quaternion form as

$$\mathbf{q}(t_{s,i}) = \text{Exp}\left[\lambda_i \,\text{Log}\left(\mathbf{q}(t_{m,i_b}) \otimes \mathbf{q}(t_{m,i_a})^{-1}\right)\right] \otimes \mathbf{q}(t_{m,i_a}) \tag{F.9}$$

To avoid clutter when deriving Jacobians, we simplify notation in (F.7)-(F.9) by replacing time-dependencies with subscripts.

$$\mathbf{p}_i = (1 - \lambda_i)\, \mathbf{p}_{i_a} + \lambda_i \mathbf{p}_{i_b} \tag{F.10}$$

$$\mathbf{R}_i = \text{Exp}\left[\lambda_i \,\text{Log}\left(\mathbf{R}_{i_b}\mathbf{R}_{i_a}^{\text{T}}\right)\right] \mathbf{R}_{i_a} \tag{F.11}$$

$$\mathbf{q}_i = \text{Exp}\left[\lambda_i \,\text{Log}\left(\mathbf{q}_{i_b} \otimes \mathbf{q}_{i_a}^{-1}\right)\right] \otimes \mathbf{q}_{i_a} \tag{F.12}$$

Using a similar development to (F.5)-(F.9), we define interpolated position and orientation at the offset-compensated time of secondary camera keyframe j.

$$\mathbf{p}_j = (1 - \lambda_i)\, \mathbf{p}_{j_a} + \lambda_j \mathbf{p}_{j_b} \tag{F.13}$$

$$\mathbf{R}_j = \text{Exp}\left[\lambda_j \,\text{Log}\left(\mathbf{R}_{j_b}\mathbf{R}_{j_a}^{\text{T}}\right)\right] \mathbf{R}_{j_a} \tag{F.14}$$

$$\mathbf{q}_j = \text{Exp}\left[\lambda_j \,\text{Log}\left(\mathbf{q}_{j_b} \otimes \mathbf{q}_{j_a}^{-1}\right)\right] \otimes \mathbf{q}_{j_a} \tag{F.15}$$

F.3 Reprojection Residual with Interpolated Poses

We obtain the reprojection residual with interpolated poses by substituting (F.10), (F.12), (F.13), (F.15) into (3.141)-(3.142). However, the extrinsic translation $\mathbf{p}_{bc}$ and rotation $\mathbf{R}_{bc}$ in the resultant expression must be evaluated for the secondary camera instead of the master camera. Also, in this case $\mathbf{h}_l$ and ρ_l pertain to a feature observation in the secondary camera.

Compared with the expression in (3.142), the reprojection residual with interpolated poses depends on two extra position parameter blocks, two extra quaternion parameter blocks, and two extra time offset parameters.

$$\mathbf{r}\left(\mathbf{p}_{i_a}, \mathbf{p}_{i_b}, \mathbf{q}_{i_a}, \mathbf{q}_{i_b}, \mathbf{p}_{j_a}, \mathbf{p}_{j_b}, \mathbf{q}_{j_a}, \mathbf{q}_{j_b}, \rho_l, d_m, d_s\right) \tag{F.16}$$

In the case when $i_b = j_a$, only one extra position parameter block and one extra quaternion parameter block are needed. In the case when $i_a = j_a$ and $i_b = j_b$, no extra position or quaternion parameter blocks are needed.

F.4 Jacobians of the Reprojection Residual with Interpolated Poses

We require Jacobians with respect to each of the parameter blocks in (F.16). The quaternion parameter blocks $\mathbf{q}_{i_a}, \mathbf{q}_{i_b}$, and $\mathbf{q}_{j_a}, \mathbf{q}_{j_b}$ require Jacobians with respect to minimal parameterizations of their perturbations, which are $\delta\boldsymbol{\theta}_{i_a}$, $\delta\boldsymbol{\theta}_{i_b}$, $\delta\boldsymbol{\theta}_{j_a}$, and $\delta\boldsymbol{\theta}_{j_b}$, respectively.

Because the reprojection residual with interpolated poses is a composition of functions (F.10), (F.12), (F.13), and (F.15) into (3.141)-(3.142), we apply chain rule to compute Jacobians.

$$\frac{\partial \mathbf{r}}{\partial \mathbf{p}_{i_a}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial \mathbf{p}_{i_a}} \tag{F.17}$$

$$\frac{\partial \mathbf{r}}{\partial \mathbf{p}_{i_b}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial \mathbf{p}_{i_b}} \tag{F.18}$$

$$\frac{\partial \mathbf{r}}{\partial \delta\boldsymbol{\theta}_{i_a}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_i} \frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}} \tag{F.19}$$

$$\frac{\partial \mathbf{r}}{\partial \delta\boldsymbol{\theta}_{i_b}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_i} \frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}} \tag{F.20}$$

$$\frac{\partial \mathbf{r}}{\partial \mathbf{p}_{j_a}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{p}_j} \frac{\partial \mathbf{p}_j}{\partial \mathbf{p}_{j_a}} \tag{F.21}$$

$$\frac{\partial \mathbf{r}}{\partial \mathbf{p}_{j_b}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \mathbf{p}_j} \frac{\partial \mathbf{p}_j}{\partial \mathbf{p}_{j_b}} \tag{F.22}$$

$$\frac{\partial \mathbf{r}}{\partial \delta\boldsymbol{\theta}_{j_a}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_j} \frac{\partial \delta\boldsymbol{\theta}_j}{\partial \delta\boldsymbol{\theta}_{j_a}} \tag{F.23}$$

$$\frac{\partial \mathbf{r}}{\partial \delta\boldsymbol{\theta}_{j_b}} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_j} \frac{\partial \delta\boldsymbol{\theta}_j}{\partial \delta\boldsymbol{\theta}_{j_b}} \tag{F.24}$$

$$\frac{\partial \mathbf{r}}{\partial \rho_l} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \frac{\partial \mathbf{a}}{\partial \rho_l} \tag{F.25}$$

$$\frac{\partial \mathbf{r}}{\partial d_m} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \left(\frac{\partial \mathbf{a}}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial d_m} + \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_i} \frac{\partial \delta\boldsymbol{\theta}_i}{\partial d_m} + \frac{\partial \mathbf{a}}{\partial \mathbf{p}_j} \frac{\partial \mathbf{p}_j}{\partial d_m} + \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_j} \frac{\partial \delta\boldsymbol{\theta}_j}{\partial d_m} \right) \tag{F.26}$$

$$\frac{\partial \mathbf{r}}{\partial d_s} = \frac{\partial \mathbf{z}}{\partial \mathbf{a}} \left(\frac{\partial \mathbf{a}}{\partial \mathbf{p}_i} \frac{\partial \mathbf{p}_i}{\partial d_s} + \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_i} \frac{\partial \delta\boldsymbol{\theta}_i}{\partial d_s} + \frac{\partial \mathbf{a}}{\partial \mathbf{p}_j} \frac{\partial \mathbf{p}_j}{\partial d_s} + \frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_j} \frac{\partial \delta\boldsymbol{\theta}_j}{\partial d_s} \right) \tag{F.27}$$

In the above expressions, $\delta\boldsymbol{\theta}_i$ and $\delta\boldsymbol{\theta}_j$ are minimal parameterizations of perturbations to the interpolated poses $\mathbf{q}_i$ and $\mathbf{q}_j$.

F.4.1 Position Jacobians

The Jacobians of the interpolated positions with respect to the master camera positions are obtained by differentiating (F.12) and (F.15).

$$\frac{\partial \mathbf{p}_i}{\partial \mathbf{p}_{i_a}} = (1 - \lambda_i)\, \mathbf{I} \tag{F.28}$$

$$\frac{\partial \mathbf{p}_i}{\partial \mathbf{p}_{i_b}} = \lambda_i \mathbf{I} \tag{F.29}$$

$$\frac{\partial \mathbf{p}_j}{\partial \mathbf{p}_{j_a}} = (1 - \lambda_j)\, \mathbf{I} \tag{F.30}$$

$$\frac{\partial \mathbf{p}_j}{\partial \mathbf{p}_{j_b}} = \lambda_j \mathbf{I} \tag{F.31}$$

Table F.1: Evaluation of partial derivatives involving $\mathbf{a}$ *in* (F.17)-(F.27)

Partial Derivative	Evaluation
$\frac{\partial \mathbf{z}}{\partial \mathbf{a}}$	Substitute (F.10), (F.12), (F.13), and (F.15) into (3.141) and then substitute the result into (3.144).
$\frac{\partial \mathbf{a}}{\partial \mathbf{p}_i}$	Substitute (F.15) into (3.145) and use the secondary camera extrinsic rotation for $\mathbf{R}_{bc}$.
$\frac{\partial \mathbf{a}}{\partial \mathbf{p}_j}$	Substitute (F.15) into (3.146) and use the secondary camera extrinsic rotation for $\mathbf{R}_{bc}$.
$\frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_i}$	Substitute (F.12) and (F.15) into (3.155) and use the secondary camera extrinsic transform parameters for $\mathbf{R}_{bc}$ and $\mathbf{p}_{bc}$.
$\frac{\partial \mathbf{a}}{\partial \delta\boldsymbol{\theta}_j}$	Substitute (F.10), (F.12), (F.13), and (F.15) into (3.164) and use the secondary camera extrinsic transform parameters for $\mathbf{R}_{bc}$ and $\mathbf{p}_{bc}$.
$\frac{\partial \mathbf{a}}{\partial \rho_l}$	Substitute (F.12) and (F.15) into (3.148) and use the secondary camera extrinsic rotation for $\mathbf{R}_{bc}$.

F.4.2 Jacobian of Interpolated Attitude Perturbation with respect to Earlier Master Camera Attitude Perturbation

We derive the Jacobian $\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}}$ using the perturbation method. For conciseness, we introduce the variable

$$\boldsymbol{\phi} = \operatorname{Log}\left(\mathbf{R}_{i_b}\mathbf{R}_{i_a}^{\mathrm{T}}\right) \tag{F.32}$$

First, apply a rotational perturbation $\delta\boldsymbol{\theta}_i$ to the left hand side of (F.11) and a rotational perturbation $\delta\boldsymbol{\theta}_{i_a}$ to the base rotation $\mathbf{R}_{i_a}$ on the right hand side of (F.11).

$$\begin{aligned}
\mathbf{R}_i \operatorname{Exp}(\delta\boldsymbol{\theta}_i) &= \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b}\left(\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a})\right)^{\mathrm{T}}\right)\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.33)}\\
&= \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a})^{\mathrm{T}}\mathbf{R}_{i_a}^{\mathrm{T}}\right)\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.34)}\\
&\overset{\text{(A.13)}}{=} \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b}\operatorname{Exp}(-\delta\boldsymbol{\theta}_{i_a})\mathbf{R}_{i_a}^{\mathrm{T}}\right)\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.35)}\\
&\overset{\text{(A.46)}}{=} \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b}\mathbf{R}_{i_a}^{\mathrm{T}}\operatorname{Exp}(-\mathbf{R}_{i_a}\delta\boldsymbol{\theta}_{i_a})\right)\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.36)}\\
&\overset{\text{(F.32)}}{=} \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\operatorname{Exp}(\boldsymbol{\phi})\operatorname{Exp}(-\mathbf{R}_{i_a}\delta\boldsymbol{\theta}_{i_a})\right)\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.37)}\\
&\overset{\text{(A.48)}}{\approx} \operatorname{Exp}\left[\lambda_i\left(\boldsymbol{\phi} - \mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a}\delta\boldsymbol{\theta}_{i_a}\right)\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.38)}\\
&= \operatorname{Exp}\left[\lambda_i\boldsymbol{\phi} - \lambda_i\mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a}\delta\boldsymbol{\theta}_{i_a}\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.39)}\\
&\overset{\text{(A.47)}}{\approx} \operatorname{Exp}\left(\lambda_i\boldsymbol{\phi}\right)\operatorname{Exp}\left[-\mathbf{J}_r\left(\lambda_i\boldsymbol{\phi}\right)\lambda_i\mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a}\delta\boldsymbol{\theta}_{i_a}\right]\mathbf{R}_{i_a}\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.40)}\\
&\overset{\text{(A.46)}}{=} \operatorname{Exp}\left(\lambda_i\boldsymbol{\phi}\right)\mathbf{R}_{i_a}\operatorname{Exp}\left[-\mathbf{R}_{i_a}^{\mathrm{T}}\mathbf{J}_r\left(\lambda_i\boldsymbol{\phi}\right)\lambda_i\mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a}\delta\boldsymbol{\theta}_{i_a}\right]\operatorname{Exp}(\delta\boldsymbol{\theta}_{i_a}) && \text{(F.41)}\\
&\overset{\text{(A.49)}}{\approx} \operatorname{Exp}\left(\lambda_i\boldsymbol{\phi}\right)\mathbf{R}_{i_a}\operatorname{Exp}\left[\delta\boldsymbol{\theta}_{i_a} - \mathbf{R}_{i_a}^{\mathrm{T}}\mathbf{J}_r\left(\lambda_i\boldsymbol{\phi}\right)\lambda_i\mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a}\delta\boldsymbol{\theta}_{i_a}\right] && \text{(F.42)}\\
&\overset{\text{(F.32)}}{=} \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b}\mathbf{R}_{i_a}^{\mathrm{T}}\right)\right]\mathbf{R}_{i_a}\operatorname{Exp}\left[\left(\mathbf{I} - \lambda_i\mathbf{R}_{i_a}^{\mathrm{T}}\mathbf{J}_r\left(\lambda_i\boldsymbol{\phi}\right)\mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a}\right)\delta\boldsymbol{\theta}_{i_a}\right] && \text{(F.43)}
\end{aligned}$$

Use (F.11) to cancel out $\mathbf{R}_i$ from both sides of (F.43) and then apply the capitalized logarithmic map to both sides.

$$\begin{aligned}
\delta\boldsymbol{\theta}_i &\approx \left(\mathbf{I} - \lambda_i\mathbf{R}_{i_a}^{\mathrm{T}}\mathbf{J}_r\left(\lambda_i\boldsymbol{\phi}\right)\mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a}\right)\delta\boldsymbol{\theta}_{i_a} && \text{(F.44)}\\
\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}} &\approx \mathbf{I} - \lambda_i\mathbf{R}_{i_a}^{\mathrm{T}}\mathbf{J}_r\left(\lambda_i\boldsymbol{\phi}\right)\mathbf{J}_r^{-1}(\boldsymbol{\phi})\mathbf{R}_{i_a} && \text{(F.45)}
\end{aligned}$$

Substitute (A.44) and (A.45) into (F.45).

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}} \approx \mathbf{I} - \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} - \frac{1-\cos(\lambda_i \|\boldsymbol{\phi}\|)}{\lambda_i \|\boldsymbol{\phi}\|^2} [\boldsymbol{\phi}]_\times + \ldots \right) \left(\mathbf{I} + \frac{1}{2} [\boldsymbol{\phi}]_\times + \ldots \right) \mathbf{R}_{i_a} \tag{F.46}$$

Expand and drop terms that are second order in $[\boldsymbol{\phi}]_\times$.

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}} \approx \mathbf{I} - \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} + \left(\frac{1}{2} - \frac{1-\cos(\lambda_i \|\boldsymbol{\phi}\|)}{\lambda_i \|\boldsymbol{\phi}\|^2} \right) [\boldsymbol{\phi}]_\times \right) \mathbf{R}_{i_a} \tag{F.47}$$

When $\|\boldsymbol{\phi}\| \ll 1$, $\cos(\lambda_i \|\boldsymbol{\phi}\|) \approx 1$ and $\lambda_i^{-1} \|\boldsymbol{\phi}\|^{-2} (1 - \cos(\lambda_i \|\boldsymbol{\phi}\|)) \approx 0$. We use this approximation to simplify (F.47)

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}} \approx \mathbf{I} - \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} + \frac{1}{2} [\boldsymbol{\phi}]_\times \right) \mathbf{R}_{i_a} \tag{F.48}$$

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}} \overset{\text{(F.32)}}{\approx} \mathbf{I} - \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} + \frac{1}{2} \left[\operatorname{Log} \left(\mathbf{R}_{i_b} \mathbf{R}_{i_a}^{\mathrm{T}} \right) \right]_\times \right) \mathbf{R}_{i_a} \tag{F.49}$$

Note that the expression for $\frac{\partial \delta\boldsymbol{\theta}_j}{\partial \delta\boldsymbol{\theta}_{j_a}}$ is the same as the expression for $\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_a}}$ except for the index (i vs. j).

F.4.3 Jacobian of Interpolated Attitude Perturbation with respect to Later Master Camera Attitude Perturbation

We derive the Jacobian $\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}}$ using the perturbation method. First, apply a rotational perturbation $\delta\boldsymbol{\theta}_i$ to the left hand side of (F.11) and a rotational perturbation $\delta\boldsymbol{\theta}_{i_b}$ to the base rotation $\mathbf{R}_{i_b}$ on the right hand side of (F.11).

$$\mathbf{R}_i \operatorname{Exp}(\delta\boldsymbol{\theta}_i) = \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b} \operatorname{Exp}(\delta\boldsymbol{\theta}_{i_b}) \mathbf{R}_{i_a}^{\mathrm{T}} \right) \right] \mathbf{R}_{i_a} \tag{F.50}$$

$$\overset{\text{(A.46)}}{=} \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b} \mathbf{R}_{i_a}^{\mathrm{T}} \operatorname{Exp}\left(\mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \right) \right) \right] \mathbf{R}_{i_a} \tag{F.51}$$

$$\overset{\text{(F.32)}}{=} \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\operatorname{Exp}(\boldsymbol{\phi}) \operatorname{Exp}\left(\mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \right) \right) \right] \mathbf{R}_{i_a} \tag{F.52}$$

$$\overset{\text{(A.48)}}{\approx} \operatorname{Exp}\left[\lambda_i \left(\boldsymbol{\phi} + \mathbf{J}_r^{-1}(\boldsymbol{\phi}) \mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \right) \right] \mathbf{R}_{i_a} \tag{F.53}$$

$$= \operatorname{Exp}\left[\lambda_i \boldsymbol{\phi} + \lambda_i \mathbf{J}_r^{-1}(\boldsymbol{\phi}) \mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \right] \mathbf{R}_{i_a} \tag{F.54}$$

$$\overset{\text{(A.47)}}{\approx} \operatorname{Exp}\left(\lambda_i \boldsymbol{\phi} \right) \operatorname{Exp}\left[\mathbf{J}_r\left(\lambda_i \boldsymbol{\phi} \right) \lambda_i \mathbf{J}_r^{-1}(\boldsymbol{\phi}) \mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \right] \mathbf{R}_{i_a} \tag{F.55}$$

$$\overset{\text{(A.46)}}{\approx} \operatorname{Exp}\left(\lambda_i \boldsymbol{\phi} \right) \mathbf{R}_{i_a} \operatorname{Exp}\left[\mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r\left(\lambda_i \boldsymbol{\phi} \right) \lambda_i \mathbf{J}_r^{-1}(\boldsymbol{\phi}) \mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \right] \tag{F.56}$$

$$\overset{\text{(F.32)}}{\approx} \operatorname{Exp}\left[\lambda_i \operatorname{Log}\left(\mathbf{R}_{i_b} \mathbf{R}_{i_a}^{\mathrm{T}} \right) \right] \mathbf{R}_{i_a} \operatorname{Exp}\left[\mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r\left(\lambda_i \boldsymbol{\phi} \right) \lambda_i \mathbf{J}_r^{-1}(\boldsymbol{\phi}) \mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \right] \tag{F.57}$$

Use (F.11) to cancel out $\mathbf{R}_i$ from both sides of (F.57) and then apply the capitalized logarithmic map to both sides.

$$\delta\boldsymbol{\theta}_i \approx \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r\left(\lambda_i \boldsymbol{\phi} \right) \mathbf{J}_r^{-1}(\boldsymbol{\phi}) \mathbf{R}_{i_a} \delta\boldsymbol{\theta}_{i_b} \tag{F.58}$$

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}} = \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r\left(\lambda_i \boldsymbol{\phi} \right) \mathbf{J}_r^{-1}(\boldsymbol{\phi}) \mathbf{R}_{i_a} \tag{F.59}$$

Substitute (A.44) and (A.45) into (F.59).

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}} \approx \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} - \frac{1-\cos(\lambda_i \|\boldsymbol{\phi}\|)}{\lambda_i \|\boldsymbol{\phi}\|^2} [\boldsymbol{\phi}]_\times + \ldots \right) \left(\mathbf{I} + \frac{1}{2} [\boldsymbol{\phi}]_\times + \ldots \right) \mathbf{R}_{i_a} \tag{F.60}$$

Expand and drop terms that are second order in $[\boldsymbol{\phi}]_\times$.

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}} \approx \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} + \left(\frac{1}{2} - \frac{1 - \cos(\lambda_i \left\| \boldsymbol{\phi} \right\|)}{\lambda_i \left\| \boldsymbol{\phi} \right\|^2} \right) [\boldsymbol{\phi}]_\times \right) \mathbf{R}_{i_a} \tag{F.61}$$

When $\left\| \boldsymbol{\phi} \right\| \ll 1$, $\cos(\lambda_i \left\| \boldsymbol{\phi} \right\|) \approx 1$ and $\lambda_i^{-1} \left\| \boldsymbol{\phi} \right\|^{-2} (1 - \cos(\lambda_i \left\| \boldsymbol{\phi} \right\|)) \approx 0$.

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}} \approx \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} + \frac{1}{2} [\boldsymbol{\phi}]_\times \right) \mathbf{R}_{i_a} \tag{F.62}$$

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}} \overset{(F.32)}{\approx} \lambda_i \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} + \frac{1}{2} \left[\mathrm{Log} \left(\mathbf{R}_{i_b} \mathbf{R}_{i_a}^{\mathrm{T}} \right) \right]_\times \right) \mathbf{R}_{i_a} \tag{F.63}$$

Note that the expression for $\frac{\partial \delta\boldsymbol{\theta}_j}{\partial \delta\boldsymbol{\theta}_{j_b}}$ is the same as the expression for $\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \delta\boldsymbol{\theta}_{i_b}}$ except for the index (i vs. j).

F.4.4 Time Offset Jacobians

The Jacobians of interpolated pose states with respect to the master camera to IMU time offset in (F.26) can be further broken down using chain rule.

$$\frac{\partial \mathbf{p}_i}{\partial d_m} = \frac{\partial \mathbf{p}_i}{\partial \lambda_i} \frac{\partial \lambda_i}{\partial d_m} \tag{F.64}$$

$$\frac{\partial \delta\boldsymbol{\theta}_i}{\partial d_m} = \frac{\partial \delta\boldsymbol{\theta}_i}{\partial \lambda_i} \frac{\partial \lambda_i}{\partial d_m} \tag{F.65}$$

$$\frac{\partial \mathbf{p}_j}{\partial d_m} = \frac{\partial \mathbf{p}_j}{\partial \lambda_j} \frac{\partial \lambda_j}{\partial d_m} \tag{F.66}$$

$$\frac{\partial \delta\boldsymbol{\theta}_j}{\partial d_m} = \frac{\partial \delta\boldsymbol{\theta}_j}{\partial \lambda_j} \frac{\partial \lambda_j}{\partial d_m} \tag{F.67}$$

Jacobians of interpolated pose states with respect to the secondary camera to IMU time offset in (F.27) can be further broken down in a similar manner as (F.64)-(F.67), with s replacing m.

$$\frac{\partial \mathbf{p}_i}{\partial \lambda_i} = \mathbf{p}_{i_b} - \mathbf{p}_{i_a} \tag{F.68}$$

$$\frac{\partial \mathbf{p}_j}{\partial \lambda_j} = \mathbf{p}_{j_b} - \mathbf{p}_{j_a} \tag{F.69}$$

We derive the Jacobian $\frac{\partial \delta\boldsymbol{\theta}_i}{\partial \lambda_i}$ using the perturbation method. First, apply a rotational perturbation $\delta\boldsymbol{\theta}_i$ to the left hand side of (F.11) and an additive scalar perturbation $\delta\lambda_i$ to λ_i on the right hand side of (F.11).

$$\mathbf{R}_i \, \mathrm{Exp}(\delta\boldsymbol{\theta}_i) = \mathrm{Exp} \left[(\lambda_i + \delta\lambda_i) \, \mathrm{Log} \left(\mathbf{R}_{i_b} \mathbf{R}_{i_a}^{\mathrm{T}} \right) \right] \mathbf{R}_{i_a} \tag{F.70}$$

$$\overset{(F.32)}{=} \mathrm{Exp} \left[\lambda_i \boldsymbol{\phi} + \delta\lambda_i \boldsymbol{\phi} \right] \mathbf{R}_{i_a} \tag{F.71}$$

$$\overset{(A.47)}{\approx} \mathrm{Exp} \left(\lambda_i \boldsymbol{\phi} \right) \mathrm{Exp} \left[\mathbf{J}_r \left(\lambda_i \boldsymbol{\phi} \right) \delta\lambda_i \boldsymbol{\phi} \right] \mathbf{R}_{i_a} \tag{F.72}$$

$$\overset{(A.46)}{=} \mathrm{Exp} \left(\lambda_i \boldsymbol{\phi} \right) \mathbf{R}_{i_a} \, \mathrm{Exp} \left[\mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r \left(\lambda_i \boldsymbol{\phi} \right) \boldsymbol{\phi} \delta\lambda_i \right] \tag{F.73}$$

$$\overset{(F.32)}{=} \mathrm{Exp} \left(\lambda_i \, \mathrm{Log} \left(\mathbf{R}_{i_b} \mathbf{R}_{i_a}^{\mathrm{T}} \right) \right) \mathbf{R}_{i_a} \, \mathrm{Exp} \left[\mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r \left(\lambda_i \boldsymbol{\phi} \right) \boldsymbol{\phi} \delta\lambda_i \right] \tag{F.74}$$

Use (F.11) to cancel out $\mathbf{R}_i$ from both sides of (F.74) and then apply the capitalized logarithmic map to both sides.

$$\delta\boldsymbol{\theta}_i \approx \mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r \left(\lambda_i \boldsymbol{\phi} \right) \boldsymbol{\phi} \delta\lambda_i \tag{F.75}$$

$$\frac{\partial \delta \boldsymbol{\theta}_i}{\partial \lambda_i} \approx \mathbf{R}_{i_a}^{\mathrm{T}} \mathbf{J}_r \left(\lambda_i \boldsymbol{\phi}\right) \boldsymbol{\phi} \tag{F.76}$$

Substitute (A.44) into (F.76).

$$\frac{\partial \delta \boldsymbol{\theta}_i}{\partial \lambda_i} \approx \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} - \frac{1 - \cos(\lambda_i \left\|\boldsymbol{\phi}\right\|)}{\lambda_i \left\|\boldsymbol{\phi}\right\|^2} \left[\boldsymbol{\phi}\right]_\times + \ldots\right) \boldsymbol{\phi} \tag{F.77}$$

Drop terms that are second order in $\left[\boldsymbol{\phi}\right]_\times$.

$$\frac{\partial \delta \boldsymbol{\theta}_i}{\partial \lambda_i} \approx \mathbf{R}_{i_a}^{\mathrm{T}} \left(\mathbf{I} - \frac{1 - \cos(\lambda_i \left\|\boldsymbol{\phi}\right\|)}{\lambda_i \left\|\boldsymbol{\phi}\right\|^2} \left[\boldsymbol{\phi}\right]_\times\right) \boldsymbol{\phi} \tag{F.78}$$

When $\left\|\boldsymbol{\phi}\right\| \ll 1$, $\cos(\lambda_i \left\|\boldsymbol{\phi}\right\|) \approx 1$ and $\lambda_i^{-1} \left\|\boldsymbol{\phi}\right\|^{-2} (1 - \cos(\lambda_i \left\|\boldsymbol{\phi}\right\|)) \approx 0$.

$$\frac{\partial \delta \boldsymbol{\theta}_i}{\partial \lambda_i} \approx \mathbf{R}_{i_a}^{\mathrm{T}} \boldsymbol{\phi} \tag{F.79}$$

$$\frac{\partial \delta \boldsymbol{\theta}_i}{\partial \lambda_i} \overset{\text{(F.32)}}{\approx} \mathbf{R}_{i_a}^{\mathrm{T}} \operatorname{Log}\left(\mathbf{R}_{i_b} \mathbf{R}_{i_a}^{\mathrm{T}}\right) \tag{F.80}$$

Note that the expression for $\frac{\partial \delta \boldsymbol{\theta}_j}{\partial \lambda_j}$ is the same as the expression for $\frac{\partial \delta \boldsymbol{\theta}_i}{\partial \lambda_i}$ except for the index (i vs. j).

We derive the partials of λ_i and λ_j with respect to d_m and d_s by inspecting (F.6).

$$\frac{\partial \lambda_i}{\partial d_m} = \frac{-1}{t_{m,i_b,\text{raw}} - t_{m,i_a,\text{raw}}} \tag{F.81}$$

$$\frac{\partial \lambda_j}{\partial d_m} = \frac{-1}{t_{m,j_b,\text{raw}} - t_{m,j_a,\text{raw}}} \tag{F.82}$$

$$\frac{\partial \lambda_i}{\partial d_s} = \frac{1}{t_{m,i_b,\text{raw}} - t_{m,i_a,\text{raw}}} \tag{F.83}$$

$$\frac{\partial \lambda_j}{\partial d_s} = \frac{1}{t_{m,j_b,\text{raw}} - t_{m,j_a,\text{raw}}} \tag{F.84}$$

Appendix G

Experiment Infrastructure

An important goal of this thesis is to implement sliding window visual-inertial odometry algorithms on physical systems and accurately estimate motion in real-world environments. To realize this objective, we use multirotor aerial robots equipped with a variety of onboard sensors to assess the performance of proposed state estimation techniques. This appendix describes the experimental infrastructure we have developed to enable multirotor flight.

G.1 Hardware

G.1.1 First Generation Flight Platform

The first generation platform is a 3.5 kg hexrotor micro aerial vehicle that fits inside a box with dimensions 74 cm (L) $\times$ 81 cm (W) $\times$ 29 cm (H). It has a power to weight ratio of around 3 and a flight endurance of 7 minutes. Thrust is provided by six 10$\times$4.5 plastic APC propellers, each mounted on a SunnySky X2212-10 KV1250 motor and controlled via ESC32v3 electronic speed controllers. The motors are powered from a Turnigy 6000 mAh 4 cell lithium polymer battery, while the avionics (sensors and compute) are powered from a Turnigy 3000 mAh 3 cell lithium polymer battery.

The platform consists of a DJI F550 hexrotor frame that sandwiches a custom fabricated printed circuit board assembly that contains most of the avionics (this is called the avionics package). Structural support is provided by carbon fiber rods that form a roll cage. The roll cage, arm tips, and carbon fiber landing gear form a convex hull surrounding the avionics board, protecting sensors and onboard compute in the event of a crash. The most common type of serious physical damage is broken arms, as they are made from plastic rather than carbon fiber.

Computation is performed by an Nvidia Jetson TX2 computer with 6 cores running at 2 GHz and 8 GB RAM. It sits on a CTI Elroy carrier board within the avionics package. The Pixracer autopilot is located within the avionics package at the hexrotor frame's center of symmetry. It contains an Invensense ICM-20608-G IMU and communicates with the TX2 via a serial-to-USB connection.

Vision sensing is performed by two MatrixVision mvBlueFOX MLC-200w grayscale global shutter USB2.0 cameras[1]. One bluefox camera is mounted at the rear end of the avionics package facing downward, while another bluefox is mounted at the vehicle's front end facing roughly 45 degrees above the horizontal. A TeraBee TeraRanger One time-of-flight single-beam range sensor is mounted at the front end of the avionics package facing downward. An Intel Realsense D435 depth camera is mounted at the front end of the avionics package facing forward.

[1]When tracking simple features such as Shi-Tomasi corners, the best results are obtained when the scene changes as little as possible between image frames. For this reason, we use a high rate low resolution operating mode (376$\times$240 at 60 Hz) instead of the default lower rate higher resolution operating mode (752$\times$480 at 30 Hz).

Figures G.1 and G.2 show planform and frontal views of the flight platform and identify key components. Wired and wireless connections between components are depicted in Figure G.7.

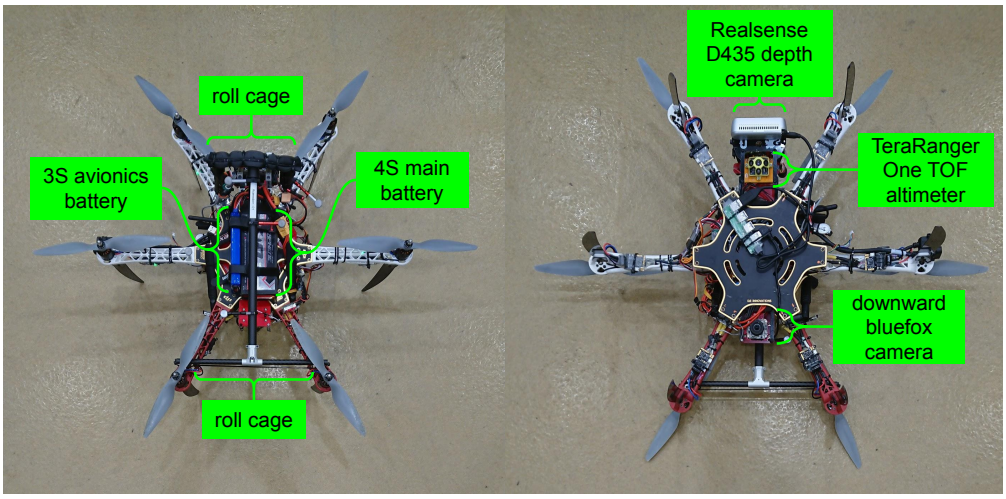


Figure G.1: The left hand side shows the top-down view of the hexrotor flight platform, while the right hand side shows the bottom-up view of the hexrotor flight platform.

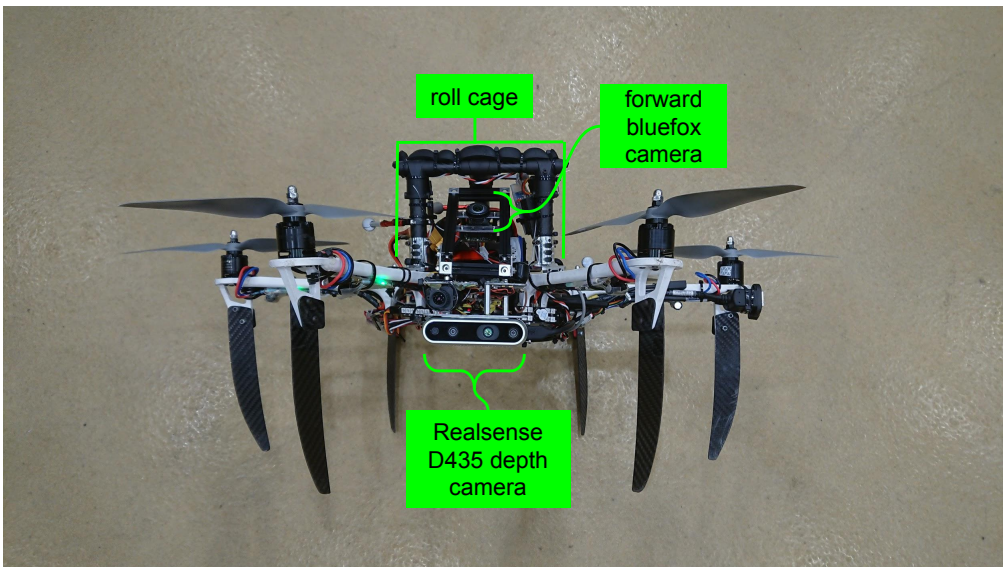


Figure G.2: A frontal view of the hexrotor flight platform.

G.1.2 Second Generation Flight Platform

The second generation platform is a 2.5 kg quadrotor aerial vehicle that fits inside a box with dimensions 36 cm (L) $\times$ 42 cm (W) $\times$ 25 cm (H). It has a thrust to weight ratio of around 3 and a flight endurance of 5 minutes. Thrust is provided by four DAL Cyclone 7056C 7$\times$5.6 plastic propellers, each mounted on a T-Motor F80 Pro 1900KV motor and controlled via a Lumenier BLHeli_32 32bit 50A 4-in-1 electronic speed controller. The motors and avionics (sensors and compute) are powered from a Lumenier 5200 mAh 4 cell lithium polymer battery.

The platform consists of an Armattan Chameleon 7 inch LR Ti quadrotor frame mounted above a sensing and compute payload encased within a rectangular truss comprised of MakerBeam elements. The payload's

structural elements form a convex hull around the onboard computers and sensors for protection in the event of a crash. The payload has four landing legs formed by MakerBeam elements protruding down from the main payload frame.

Onboard compute consists of two computers that communicate via an ethernet connection. The first computer is an Nvidia Jetson TX2 with 6 cores running at 2 GHz and 8 GB RAM. The second computer is a Gigabyte Brix 8550 computer with 8 cores running at 1.8 GHz and 32 GB RAM. The TX2 is used for control and running the IMU, downward mvBlueFOX camera, and downward rangefinder. The more powerful Brix is used for running realsense sensors and performing computationally intensive planning and mapping tasks. A Pixracer autopilot is located at the center of the quadrotor frame and communicates with the TX2 via a serial-to-USB connection.

Vision sensing is performed by two Intel Realsense D435 depth sensors mounted in forward-facing and downward-facing configurations at the front and bottom of the payload, respectively. Although the payload also has a downward-facing MatrixVision mvBlueFOX MLC-200w grayscale global shutter USB2.0 camera, we avoid using it in favor of the downward realsense for the sensor resource allocation (Sect. 6.4) and depth experiments (Sect. 7.6) to ensure that the image rectification applied to the forward and downward cameras is as similar as possible.

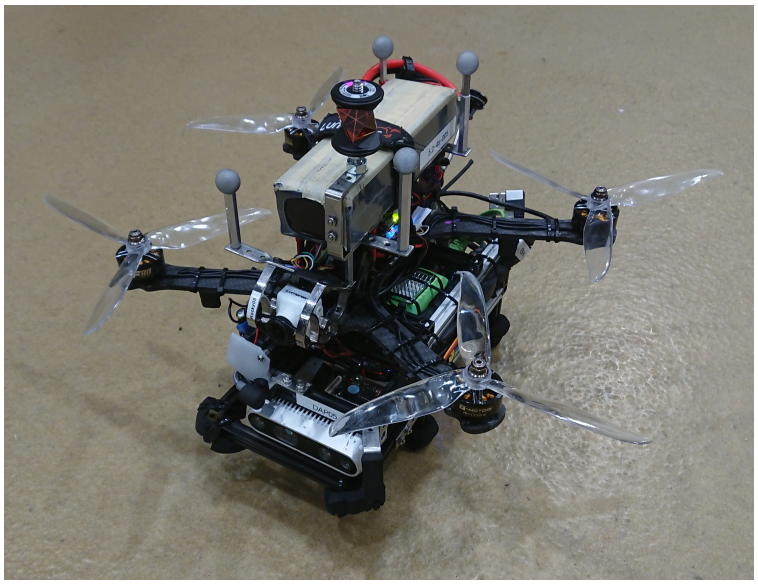


Figure G.3: The second generation quadrotor flight platform. The forward-facing realsense is visible, but the downward-facing realsense is occluded from view (it on the payload's underside).

G.1.3 Motion Capture Arena

We use a motion capture arena (Fig. G.4) for collecting ground truth to evaluate performance of state estimation and control algorithms. Most of the motion capture arena's 5 m $\times$ 5 m $\times$ 4 m volume is contained within the fields of view of at least two out of the thirteen Vicon infrared cameras mounted along the upper edges, enabling sub-centimeter precision tracking of aerial robots equipped with reflective infrared markers. A truss structure provides mounting locations for individual Vicon cameras and carries wires from each Vicon camera to the computer running the motion tracking software. Nets cover the top and four sides of the cubical motion capture arena to prevent aerial robots from escaping in the event of a malfunction, while foam mats cushion the impact of hard landings and crashes. When testing state estimation algorithms, colorful carpets with high visual texture are laid over the largely textureless gray mats to ensure that the aerial robot's downward camera can detect a sufficient number of features.

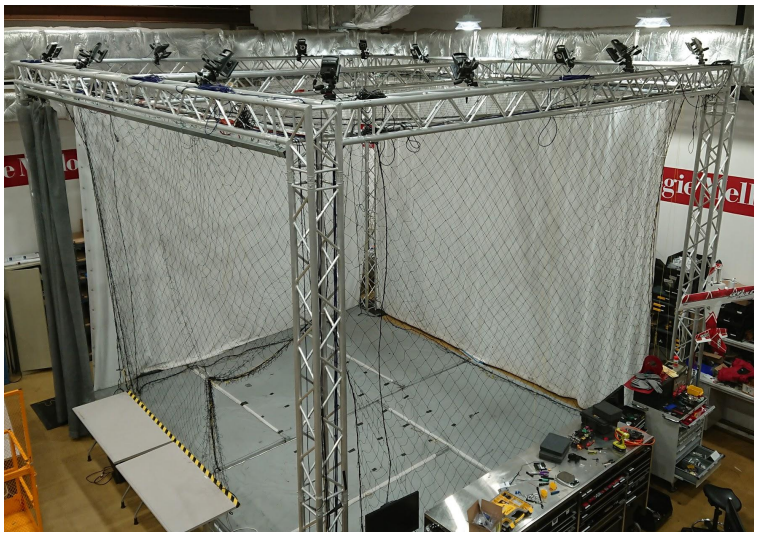


Figure G.4: The Vicon motion capture arena.

G.1.4 Ground Control Station

The ground control station contains the equipment needed to communicate with and send commands to the robot. A Netgear R6400 AC1750 router provides a 2.4 GHz wifi network that enables communication between the robot and the base station laptop. In situations where wifi range is insufficient due to clutter or distance, it is augmented with TP-Link RE210 AC750 Wifi Range Extenders.

A Futaba T8J 2.4 GHz RC transmitter is used for manual RC flight, bypassing the entire autonomy pipeline. This capability is useful for quick flight tests after physical repairs to assess flightworthiness and for emergency manual takeovers if mishaps occur in the autonomy pipeline. During nominal operations, the RC transmitter is only used for arming and disarming the motors.

A Xbox One controller connected to the base station laptop via USB provides the teleoperation interface for autonomous flight. Joystick inputs are converted to forward arc motion primitive trajectory references that are followed by the robot's onboard controller. When running flight experiments outdoors during daylight hours, the laptop screen is often hard to see due to ambient sunlight. We use an IKEA DRONA Storage box as a laptop hood to provide a dark environment to facilitate easy screen viewing. The top of the box also serves as a convenient location to place the Xbox One controller.

Depending on the type of flight being conducted, the flight control station can be configured in two different ways. When flying in a third-person view such that the operator walks behind the aerial robot at all times, the ground control station is carried in front of the operator with a backpack harness and foam-padded hip support bar. This configuration is called the *walking ground control station* (Fig. G.5). When flying in first-person view (using an analog FPV camera mounted at the front of the robot), the ground control station rests on a cart (Fig. G.6). In this configuration, the router is mounted on a vertical extension to improve wifi range.

G.2 Software

G.2.1 Onboard Computer

High-level autonomy algorithms are executed by the robot's onboard Nvidia Jetson TX2 computer using Ubuntu 16.04 LTS. Autonomy modules are implemented in C++ within a ROS-based [97] software architecture. Visual-inertial state estimation algorithms are implemented with the Eigen [98] and OpenCV [99] software libraries.

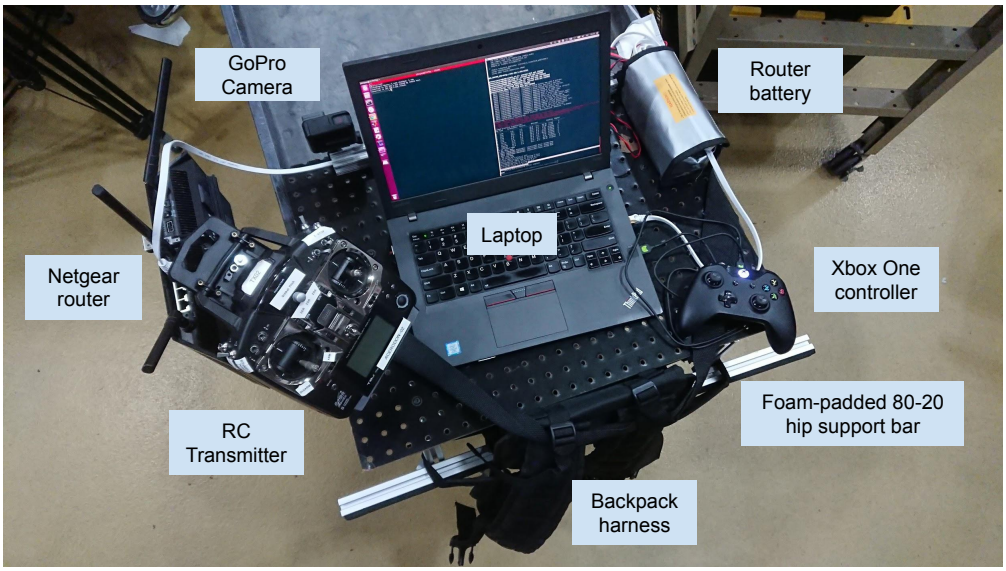


Figure G.5: The walking ground control station is designed to be carried in front of the robot operator.

Figure G.6: Cart-mounted ground control station. The router is on a vertical boom to improve wifi range. A black IKEA DRONA storage box shields the base station laptop from direct sunlight.

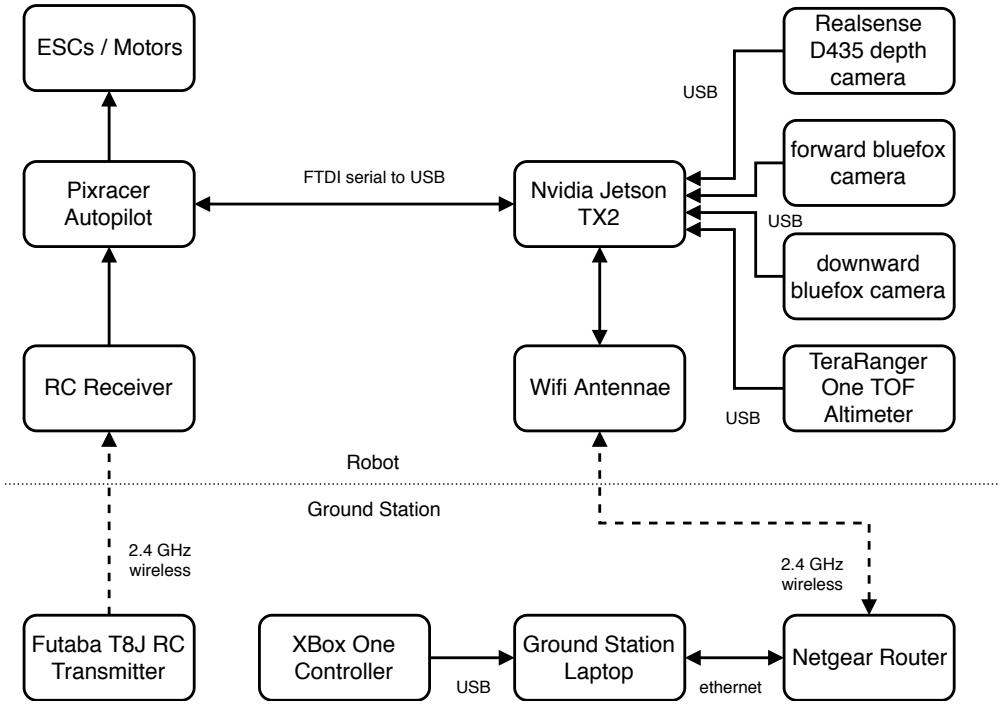


Figure G.7: Diagram of wired and wireless connections between sensors, actuators, compute, and ground station components.

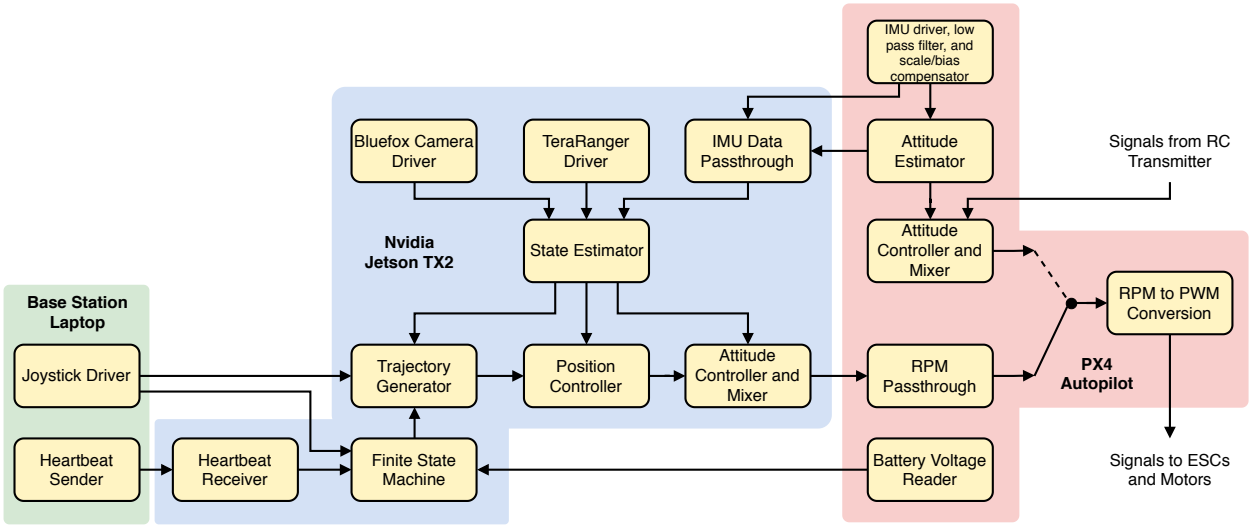


Figure G.8: Diagram of software components running on the base station laptop, onboard computer, and autopilot computer. Information flows between components are indicated with arrows.

The blue panel in Figure G.8 contains the software components that run on the onboard computer. The bluefox camera driver produces raw images from the downward camera that are rectified and fed into the state estimator. The TeraRanger driver provides the state estimator with TOF altimeter range observations. IMU observations and complementary filter attitude estimates from the PX4 are passed to the state estimator via the MAVLink protocol.

The state estimator consumes sensor observations and outputs odometry to the planning and control modules. IMU-rate upsampling and feature tracking occur in the main thread, while computationally intensive tasks such as initialization and optimization are performed in a secondary thread to avoid interrupting the stream of high-rate odometry estimates required by other autonomy modules.

Position and yaw estimates are fed into the trajectory generator for use in planning. Two types of trajectory generators are used for flight experiments. Motion capture flight experiments use a trajectory generator that fits minimum snap polynomial trajectories [100] to pre-loaded waypoints. Outdoor flight experiments use a teleoperation trajectory generator [101] that turns joystick inputs to forward arc motion primitive trajectory references. A finite state machine (different from the one described in Sect. 4.4.1) governs the types of trajectories sent to the position controller for takeoff, hover, teleoperation, and landing. During outdoor experiments, a heartbeat signal is sent from the base station to the onboard computer to verify that the two are still connected via wifi. If the connection is lost for more than a threshold amount of time, the finite state machine sends a hover command to prevent the vehicle from drifting further out of wifi range. The finite state machine also receives battery voltage measurements from the autopilot and lands the vehicle if the battery level is too low.

Position references from the trajectory generator are fed into a backstepping controller that consists of an outer loop position controller using EPC [102] and an inner loop PD attitude controller [103]. The position controller computes desired thrust, roll, and pitch signals at 100 Hz while the attitude controller computes desired torque at 250 Hz. Force disturbances are handled by EPC, while torque disturbances are compensated with the low pass filtered output of a nonlinear Luenberger observer. The desired thrust for each rotor is computed using the Moore-Penrose pseudoinverse of the hexrotor mixing matrix (derived from rotor geometry and moment scale). Finally, desired RPM commands are computed from desired thrust values using quadratic RPM-to-thrust models empirically determined from static thrust tests and sent to the autopilot computer via the MAVLink protocol.

Most of the data transfer between software components running on the base station laptop and onboard computer occurs via publishing and subscribing to ROS messages. The reference signals in the planning and control systems constitute an exception to this pattern, as they are accessed by pointer within the same class object.

G.2.2 Autopilot Computer

A custom modified version of PX4 firmware [104] runs onboard the pixracer autopilot. After being read, raw IMU observations are low pass filtered to eliminate the effects of chassis vibrations and then corrected with calibration parameters obtained from offline calbiration routines. Corrected IMU observations are fed into a complementary filter [105] to estimate attitude and gyro biases. We disable magnetometer correction terms in the complementary filter to avoid large errors in roll and pitch estimates when operating indoors near metallic objects[2]. Bias compensated angular velocity observations are sent to the onboard computer for use in visual-inertial state estimation algorithms, while the attitude estimate is only used for RC control and for the auxiliary estimator (Sect. 4.3.4).

Another important function of the autopilot computer is to send PWM signals to the ESCs to spin the aerial robot's motors at desired speeds. During nominal flight, RPM commands are sent from the onboard computer via the MAVLink protocol and converted to PWM values. During RC flight, RPM commands

[2]The extra yaw estimate drift incurred by disabling the magnetometer correction is inconsequential for manual RC control and for the auxiliary estimator

are computed by an attitude controller that uses RC throttle values as desired attitude and thrust references and the complementary filter's output for the state estimate. The autopilot's attitude controller runs even during nominal flight to enable the operator to instantaneously switch to manual RC flight in the event of an emergency.

Appendix H

Motion Estimate Accuracy Evaluation

Evaluating the difference between an estimated trajectory and its corresponding ground truth is necessary step in assessing a visual-inertial odometry algorithm's performance. Both the estimated trajectory and ground truth consist of a time history of poses (and optionally velocities) corresponding to a rigid body's motion. An estimated trajectory's accuracy is characterized by a scalar value that represents its similarity with the ground truth. Computing this number typically involves the following steps:

1. Shift the estimated trajectory's timestamps to be temporally consistent with ground truth data timestamps[1].
2. Interpolate the ground truth at the estimated trajectory's corrected timestamps.
3. Transform the estimated trajectory into the same reference frame as the ground truth.
4. Summarize the error between all pairs of transformed estimated trajectory poses and interpolated ground truth poses over the time interval of interest.

H.1 Time Offset Compensation

If the clocks used to timestamp trajectory estimates and ground truth measurements are not synchronized before data collection, the time offset between them must be determined prior to trajectory alignment. For visual-inertial trajectory estimates that use a motion capture system for ground truth, the time offset is usually between the IMU clock and the motion capture system's clock. The time offset between two timestamped sequences of observations is determined by identifying an observation from each sequence that corresponds to the same physical event and then taking the difference of timestamps in the corresponding pair.

The angular velocity norm is a convenient quantity to use for identifying time offset between an IMU and a motion capture system because it is observable by both, frame-independent, and captures easily distinguishable physical events in the form of local maxima (peaks). We exploit these properties to compute time offset by identifying a peak in the IMU angular velocity norm plot, finding the corresponding peak in the motion capture angular velocity norm plot, and subtracting their timestamps. The process of finding peak correspondences may either be performed manually by visually inspecting a plot or automatically by first smoothing[2] the IMU and motion capture signals and minimizing their difference as a function of time offset.

[1] This assumes that the clocks used to timestamp the trajectory estimates and the ground truth measurements are drift-free.

[2] IMU gyros are inherently noisy. Motion capture angular velocity observations are noisy because they are numerically differentiated from high rate direct attitude observations. Consequently, automatic alignment algorithms require smoothing to mitigate the impact of noisy observations.

H.2 Trajectory Interpolation

After visual-inertial trajectory estimates are temporally aligned with ground truth, the next step is to interpolate ground truth poses at the estimated trajectory timestamps. In order to avoid extrapolation, the interpolation should only occur over a time interval where the trajectory estimate and ground truth are both available. If the corrected estimate timestamps span $[t_{\text{est,start}}, t_{\text{est,end}}]$ and the ground truth measurement timestamps span $\left[t_{\text{gt,start}}, t_{\text{gt,end}}\right]$, then the overlap time interval is $\left[\max\left(t_{\text{est,start}}, t_{\text{gt,start}}\right), \min\left(t_{\text{est,end}}, t_{\text{gt,end}}\right)\right]$.

Positions may be interpolated with local linear or cubic spline interpolation depending on the rate and smoothness of the ground truth measurements. Rotations may be interpolated linearly using SLERP, or using linear or cubic spline interpolation on individual Euler angles as long as singularities are accounted for.

H.3 4DOF Trajectory Alignment

Interpolation produces a set of corresponding estimate-ground truth pose pairs. The goal of trajectory alignment is to find a transform that will express the estimated trajectory poses in the ground truth reference frame. Visual-inertial odometry has four unobservable degrees of freedom consisting of a 3D translation and a yaw rotation about the gravity vector. Because roll and pitch rotations are directly observable from IMU observations during non-aggressive motions, visual-inertial trajectory estimates are typically aligned with ground truth using a 4DOF rigid body transform (position and yaw only) instead of a full 6DOF rigid body transform.

Let $\{t_i\}_{i=0}^{n}$ be a set of timestamps associated with the pose pair correspondences. Let the subscript est denote the original trajectory estimate, gt denote ground truth, and al denote the aligned trajectory estimate. Let $\mathbf{p} \in \mathbb{R}^3$ be a position, $\mathbf{v} \in \mathbb{R}^3$ be a velocity, $\mathbf{R} \in SO(3)$ be a rotation matrix, and $\psi \in \mathbb{R}$ be a yaw. The 4DOF transform parameters $\Delta\mathbf{p}, \Delta\psi$ are used to compute the aligned estimated trajectory[3].

$$\mathbf{p}_{\text{al}} = \Delta\mathbf{R}\,\mathbf{p}_{\text{est}} + \Delta\mathbf{p} \tag{H.1}$$

$$\mathbf{v}_{\text{al}} = \Delta\mathbf{R}\,\mathbf{v}_{\text{est}} \tag{H.2}$$

$$\psi_{\text{al}} = \psi_{\text{est}} + \Delta\psi \tag{H.3}$$

This notation will be used to describe two methods of performing trajectory alignment.

H.3.1 Using Initial Pose Correspondence

The simplest method of trajectory alignment utilizes the 4DOF transform between the first estimated and ground truth poses at the time t_0. Typically, t_0 is chosen as the time of the visual-inertial optimization's initialization.

$$\Delta\psi = \psi_{\text{gt}}(t_0) - \psi_{\text{est}}(t_0) \tag{H.4}$$

$$\Delta\mathbf{R} = \begin{bmatrix} \cos\Delta\psi & -\sin\Delta\psi & 0 \\ \sin\Delta\psi & \cos\Delta\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{H.5}$$

$$\Delta\mathbf{p} = \mathbf{p}_{\text{gt}}(t_0) - \Delta\mathbf{R}\,\mathbf{p}_{\text{est}}(t_0) \tag{H.6}$$

This method requires ground truth heading, which is provided by motion capture systems but not by GPS or total stations. An advantage of the initial pose correspondence method is it agrees with the intuition that trajectory estimation error increases with time and therefore is appropriate for assessing the consistency of covariances estimates.

[3]Note that roll and pitch are unaffected by the 4DOF transform.

H.3.2 Using Multiple Position Correspondences

If ground truth yaw measurements are unavailable, trajectory alignment is still possible with only position observations. Choose a set of times $\mathcal{T} \subseteq \{t_i\}_{i=1}^n$. The 4DOF transform is obtained by solving the following optimization:

$$\arg \min_{\Delta\mathbf{p}, \Delta\psi} \sum_{t \in \mathcal{T}} \|\mathbf{R}_z(\Delta\psi)\mathbf{p}_{\text{est}}(t) + \Delta\mathbf{p} - \mathbf{p}_{\text{gt}}(t)\|^2 \tag{H.7}$$

The yaw component of the closed form solution of (H.7) is given by

$$\Delta\psi = \text{atan2}\left(p_{12} - p_{21}, p_{11} - p_{22}\right) \tag{H.8}$$

$$\begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{22} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \left(\mathbf{p}_{\text{est}}(t) - \bar{\mathbf{p}}_{\text{est}}\right)\left(\mathbf{p}_{\text{gt}}(t) - \bar{\mathbf{p}}_{\text{gt}}\right)^{\text{T}} \tag{H.9}$$

$$\bar{\mathbf{p}}_{\text{est}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathbf{p}_{\text{est}}(t) \tag{H.10}$$

$$\bar{\mathbf{p}}_{\text{gt}} = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \mathbf{p}_{\text{gt}}(t) \tag{H.11}$$

The translational component of the closed form solution of (H.7) is given by

$$\Delta\mathbf{p} = \bar{\mathbf{p}}_{\text{gt}} - \mathbf{R}_z(\Delta\psi)\bar{\mathbf{p}}_{\text{est}} \tag{H.12}$$

See Section III-C of [106] for a derivation of this result.

Note that the least squares formulation used in the multiple position correspondences alignment procedure assumes that all positions have the same uncertainty. Also, the numerical values of alignment errors (Sect. H.4) computed using this method are lower than those arising from the initial pose correspondence method because it spreads out the error over the entire trajectory.

The size of the set of times chosen for alignment, $\mathcal{T}$, must be at least two in order for (H.7) to be well-posed. However, choosing only two points for $\mathcal{T}$ is not advisable because it is highly suspectible to outlier observations. In practice, $\mathcal{T}$ is chosen to consist of evenly spaced time points in the set $\{t_i\}_{i=1}^n$. The other extreme of choosing $\mathcal{T} = \{t_i\}_{i=1}^n$ may result in a poorer alignment if there are outliers in one or both of the estimated and ground truth position signals.

H.4 Error Metrics

Given pairs of frame-aligned estimate and ground truth poses, the final step is to summarize the high-dimensional data into a single scalar value. In this section we describe two methods for quantifying trajectory estimation error. Note that either error measure may be used with either of the trajectory alignment methods described in Appendix H.3.

H.4.1 Absolute Trajectory Error

Absolute trajectory error is defined as the root mean square of the errors between estimated and ground truth poses for all corresponding pairs.

$$\text{ATE}_{\text{pos}} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left\|\mathbf{p}_{\text{al}}(t_i) - \mathbf{p}_{\text{gt}}(t_i)\right\|^2}, \tag{H.13}$$

$$\text{ATE}_{\text{rot}} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \left\{\cos^{-1}\left[\frac{1}{2}\left(\text{trace}(\mathbf{R}_{\text{gt}}(t_i)^{\text{T}}\mathbf{R}_{\text{al}}(t_i)) - 1\right)\right]\right\}^2} \tag{H.14}$$

Translational absolute trajectory error (H.13) is preferred over its rotational counterpart (H.14) when only a single scalar value is required for trajectory evaluation because rotational error affects translational error but not vice versa. Although absolute trajectory error is easy to compare because it is a single scalar value, it is very sensitive to rotational errors near the beginning of the estimated trajectory.

H.4.2 Final Position Drift

The final position drift is the norm of the difference between the latest pair of estimate and ground truth positions $\left\|\mathbf{p}_{\text{al}}(t_n) - \mathbf{p}_{\text{gt}}(t_n)\right\|$.

Final position drift is typically used when ground truth from motion capture systems or total stations is unavailable. In these situations, the final position drift method can be applied assuming $\mathbf{p}_{\text{gt}}(t_n) = \mathbf{0}$ by ensuring that the robot returns to its starting location at time t_n. More precisely, at time t_n the robot must return to its true physical location at time t_0. For multirotor aerial robots, this condition is satisfied by taking off and landing on a fixed landing pad and ensuring that initialization happens immediately upon takeoff.

H.5 Covariance Alignment

Position ($\mathbf{\Sigma}_{\text{pp}}$) and velocity ($\mathbf{\Sigma}_{\text{vv}}$) covariances must be aligned with ground truth before their uncertainty levels are compared against the error between aligned estimate and ground truth. In the case where a 4DOF transform was used to align the estimated trajectory with ground truth, translational covariance alignment depends only on the yaw component of the translation.

$$\mathbf{\Sigma}_{\text{pp,al}} = \mathbf{R}_z(\Delta\psi)\ \mathbf{\Sigma}_{\text{pp}}\ \mathbf{R}_z(\Delta\psi)^{\text{T}} \tag{H.15}$$

$$\mathbf{\Sigma}_{\text{vv,al}} = \mathbf{R}_z(\Delta\psi)\ \mathbf{\Sigma}_{\text{vv}}\ \mathbf{R}_z(\Delta\psi)^{\text{T}} \tag{H.16}$$

Note that the attitude covariance ($\mathbf{\Sigma}_{\text{rr}}$) does not require alignment.

Bibliography

[1] A. Martinelli, "Visual-inertial structure from motion: Observability and resolvability," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Tokyo, Japan, November 2013, pp. 4235–4242. [Page 3]

[2] G. Klein and D. Murray, "Parallel tracking and mapping for small ar workspaces," in *IEEE and ACM Int. Symp on Mixed and Augmented Reality*, Nara, Japan, November 2007, pp. 225–234. [Page 4]

[3] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardós, "Orb-slam: A versatile and accurate monocular slam system," *IEEE Trans. Robotics*, vol. 31, no. 5, pp. 1147–1163, October 2015. [Pages 4 and 7]

[4] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *Intl. Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015. [Pages 4 and 5]

[5] T. Qin, P. Li, and S. Shen, "Vins-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Trans. Robotics*, vol. 34, no. 4, pp. 1004–1020, August 2018. [Pages 4, 5, 12, 36, and 45]

[6] R. Ranftl, V. Vineet, Q. Chen, and V. Koltun, "Dense monocular depth estimation in complex dynamic scenes," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Las Vegas, USA, June 2016, pp. 4058–4066. [Page 4]

[7] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Trans. on Pattern Anal. and Mach. Intell.*, vol. 40, no. 3, pp. 611–625, March 2018. [Pages 4 and 5]

[8] L. V. Stumberg, V. Usenko, and D. Cremers, "Direct sparse visual-inertial odometry using dynamic marginalization," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Brisbane, Australia, May 2018, pp. 2510–2517. [Page 4]

[9] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *Intl. Conf. on Computer Vision*, Barcelona, Spain, November 2011, pp. 2320–2327. [Page 4]

[10] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *Proc. of the Euro. Conf. on Comp. Vis.*, Zurich, Switzerland, 2014, pp. 834–849. [Page 4]

[11] J. Shi and C. Tomasi, "Good features to track," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Seattle, USA, June 1994, pp. 593–600. [Pages 4 and 10]

[12] C. Tomasi and T. Kanade, "Shape and motion from image streams under orthography: a factorization method," *Intl. Journal of Computer Vision*, vol. 9, no. 2, pp. 137–154, 1992. [Page 4]

[13] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Intl. Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004. [Page 4]

[14] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proc. of the Euro. Conf. on Comp. Vis.* Graz, Austria: Springer, 2006, pp. 404–417. [Page 4]

[15] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *Proc. of the Euro. Conf. on Comp. Vis.* Graz, Austria: Springer, 2006, pp. 430–443. [Page 4]

[16] S. Leutenegger, M. Chli, and R. Y. Siegwart, "Brisk: Binary robust invariant scalable keypoints," in *Intl. Conf. on Computer Vision*, Barcelona, Spain, November 2011, pp. 2548–2555. [Page 4]

[17] S. Yang and S. Scherer, "Direct monocular odometry using points and lines," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Singapore, May 2017, pp. 3871–3877. [Page 4]

[18] Y. Ling, M. Kuse, and S. Shen, "Edge alignment-based visual–inertial fusion for tracking of aggressive motions," *Auton. Robots*, vol. 42, no. 3, pp. 513–528, March 2018. [Page 4]

[19] A. Pumarola, A. Vakhitov, A. Agudo, A. Sanfeliu, and F. Moreno-Noguer, "Pl-slam: Real-time monocular visual slam with points and lines," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Singapore, May 2017, pp. 4503–4508. [Page 4]

[20] X. Zuo, X. Xie, Y. Liu, and G. Huang, "Robust visual slam with point and line features," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, September 2017, pp. 1775–1782. [Page 4]

[21] S. Li, B. Ren, Y. Liu, M. Cheng, D. Frost, and V. A. Prisacariu, "Direct line guidance odometry," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Brisbane, Australia, May 2018, pp. 1–7. [Page 4]

[22] D. B. Goldman, "Vignette and exposure calibration and compensation," *IEEE Trans. on Pattern Anal. and Mach. Intell.*, vol. 32, no. 12, pp. 2276–2288, Dec 2010. [Page 5]

[23] X. Zheng, Z. Moratto, M. Li, and A. I. Mourikis, "Photometric patch-based visual-inertial odometry," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Singapore, May 2017, pp. 3264–3271. [Page 5]

[24] C. Kerl, J. Sturm, and D. Cremers, "Robust odometry estimation for rgb-d cameras," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany, May 2013, pp. 3748–3754. [Pages 5 and 8]

[25] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *Intl. Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013. [Page 5]

[26] S. Lynen, M. W. Achtelik, S. Weiss, M. Chli, and R. Siegwart, "A robust and modular multi-sensor fusion approach applied to mav navigation," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Tokyo, Japan, November 2013, pp. 3923–3929. [Page 5]

[27] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, "Robust visual inertial odometry using a direct ekf-based approach," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Hamburg, Germany, September 2015, pp. 298–304. [Page 5]

[28] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "A quadratic-complexity observability-constrained unscented kalman filter for slam," *IEEE Trans. Robotics*, vol. 29, no. 5, pp. 1226–1243, October 2013. [Page 5]

[29] W. Liu, G. Loianno, K. Mohta, K. Daniilidis, and V. Kumar, "Semi-dense visual-inertial odometry and mapping for quadrotors with swap constraints," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Brisbane, Australia, May 2018, pp. 1–6. [Page 5]

[30] A. I. Mourikis and S. I. Roumeliotis, "A multi-state constraint kalman filter for vision-aided inertial navigation," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Rome, Italy, April 2007, pp. 3565–3572. [Pages 5 and 6]

[31] Z. Huai and G. Huang, "Robocentric visual-inertial odometry," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Madrid, Spain, October 2018, pp. 6319–6326. [Page 5]

[32] G. P. Huang, A. I. Mourikis, and S. I. Roumeliotis, "An observability-constrained sliding window filter for slam," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, San Francisco, USA, September 2011, pp. 65–72. [Page 5]

[33] M. Kaess, A. Ranganathan, and F. Dellaert, "isam: Incremental smoothing and mapping," *IEEE Trans. Robotics*, vol. 24, no. 6, pp. 1365–1378, Dec 2008. [Page 5]

[34] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. J. Leonard, and F. Dellaert, "isam2: Incremental smoothing and mapping using the bayes tree," *Intl. Journal of Robotics Research*, vol. 31, no. 2, pp. 216–235, 2012. [Pages 5 and 7]

[35] T. Yap, M. Li, A. I. Mourikis, and C. R. Shelton, "A particle filter for monocular vision-aided odometry," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011, pp. 5663–5669. [Page 6]

[36] S. J. Julier and J. K. Uhlmann, "A new extension of the kalman filter to nonlinear systems," in *Proc. SPIE*, vol. 3068, Orlando, USA, July 1997, pp. 182–193. [Pages 6 and 52]

[37] R. van der Merwe, E. Wan, and S. Julier, "Sigma-point kalman filters for nonlinear estimation and sensor-fusion: Applications to integrated navigation," in *AIAA Guidance, Navigation, and Control Conference and Exhibit*, 2004. [Page 6]

[38] M. Li and A. I. Mourikis, "High-precision, consistent ekf-based visual-inertial odometry," *Intl. Journal of Robotics Research*, vol. 32, no. 6, pp. 690–711, 2013. [Page 6]

[39] L. E. Clement, V. Peretroukhin, J. Lambert, and J. Kelly, "The battle for filter supremacy: A comparative study of the multi-state constraint kalman filter and the sliding window filter," in *Conf. on Computer and Robot Vision*, Halifax, Canada, June 2015, pp. 23–30. [Page 6]

[40] H. Strasdat, J. Montiel, and A. J. Davison, "Visual slam: Why filter?" *Image and Vision Computing*, vol. 30, no. 2, pp. 65 – 77, 2012. [Page 7]

[41] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, "G2o: A general framework for graph optimization," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Shanghai, China, May 2011, pp. 3607–3613. [Page 7]

[42] S. Agarwal, K. Mierle, and Others, "Ceres solver," http://ceres-solver.org. [Page 7]

[43] V. Ila, L. Polok, M. Solony, and P. Svoboda, "Slam++-a highly efficient and temporally scalable incremental slam framework," *Intl. Journal of Robotics Research*, vol. 36, no. 2, pp. 210–230, 2017. [Page 7]

[44] J. Graeter, A. Wilczynski, and M. Lauer, "Limo: Lidar-monocular visual odometry," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Madrid, Spain, Oct 2018, pp. 7872–7879. [Page 7]

[45] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *Intl. Conf. on Computer Vision*, Nice, France, October 2003, p. 1403. [Page 7]

[46] G. Zhang and P. A. Vela, "Good features to track for visual slam," in *IEEE Conf. on Computer Vision and Pattern Recognition*, Boston, USA, June 2015, pp. 1373–1382. [Page 7]

[47] K. J. Wu, T. Do, L. C. Carrillo-Arce, and S. I. Roumeliotis, "On the vins resource-allocation problem for a dual-camera, small-size quadrotor," in *Proc. of the Intl. Sym. on Exp. Robot.*, 2016, pp. 538–549. [Page 7]

[48] L. Carlone and S. Karaman, "Attention and anticipation in fast visual-inertial navigation," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Singapore, May 2017, pp. 3886–3893. [Pages 7, 75, 76, 78, and 82]

[49] I. Dryanovski, R. G. Valenti, and Jizhong Xiao, "Fast visual odometry and mapping from rgb-d data," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany, May 2013, pp. 2305–2310. [Page 8]

[50] S. M. Prakhya, L. Bingbing, L. Weisi, and U. Qayyum, "Sparse depth odometry: 3d keypoint based pose estimation from dense depth data," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, USA, May 2015, pp. 4216–4223. [Page 8]

[51] A. S. Huang, A. Bachrach, P. Henry, M. Krainin, D. Maturana, D. Fox, and N. Roy, "Visual odometry and mapping for autonomous flight using an rgb-d camera," in *Proc. of the Intl. Sym. of Robot. Research*, Flagstaff, USA, August 2017, pp. 235–252. [Page 8]

[52] M. Nowicki and P. Skrzypezyński, "Combining photometric and depth data for lightweight and robust visual odometry," in *Euro. Conf. on Mobile Robots*, Barcelona, Spain, Sep. 2013, pp. 125–130. [Page 8]

[53] W. Kabsch, "A solution for the best rotation to relate two sets of vectors," *Acta Crystallographica Section A*, vol. 32, no. 5, pp. 922–923, 1976. [Pages 8 and 127]

[54] J. Zhang, M. Kaess, and S. Singh, "Real-time depth enhanced monocular odometry," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Chicago, USA, Sep. 2014, pp. 4973–4980. [Pages 8 and 94]

[55] Y. Lu and D. Song, "Robust rgb-d odometry using point and line features," in *Intl. Conf. on Computer Vision*, Santiago, Chile, December 2015. [Page 8]

[56] P. J. Besl and N. D. McKay, "Method for registration of 3-d shapes," in *Sensor fusion IV: control paradigms and data structures*, vol. 1611. International Society for Optics and Photonics, 1992, pp. 586–606. [Page 8]

[57] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Sacramento, USA, April 1991, pp. 2724–2729 vol.3. [Page 8]

[58] A. Segal, D. Haehnel, and S. Thrun, "Generalized-icp." in *Proc. of Robot.: Sci. and Syst.*, Seattle, USA, June 2009. [Page 8]

[59] J. Shi, B. He, L. Zhang, and J. Zhang, "Vision-based real-time 3d mapping for uav with laser sensor," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Daejeon, South Korea, Oct 2016, pp. 4524–4529. [Page 8]

[60] R. A. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. J. Davison, P. Kohli, J. Shotton, S. Hodges, and A. W. Fitzgibbon, "Kinectfusion: Real-time dense surface mapping and tracking." in *IEEE and ACM Int. Symp on Mixed and Augmented Reality*, vol. 11, no. 2011, 2011, pp. 127–136. [Page 8]

[61] F. Steinbrücker, J. Sturm, and D. Cremers, "Real-time visual odometry from dense rgb-d images," in *Intl. Conf. on Computer Vision Workshops*, Barcelona, Spain, Nov 2011, pp. 719–722. [Page 8]

[62] T. Whelan, H. Johannsson, M. Kaess, J. J. Leonard, and J. McDonald, "Robust real-time visual odometry for dense rgb-d mapping," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Karlsruhe, Germany, May 2013, pp. 5724–5731. [Page 8]

[63] D. Gutiérrez-Gómez, W. Mayol-Cuevas, and J. J. Guerrero, "Inverse depth for accurate photometric and geometric error minimisation in rgb-d dense visual odometry," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, USA, May 2015, pp. 83–89. [Page 8]

[64] Pyojin Kim, Hyon Lim, and H. J. Kim, "Robust visual odometry to irregular illumination changes with rgb-d camera," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Hamburg, Germany, Sep. 2015, pp. 3688–3694. [Page 8]

[65] Z. Fang and S. Scherer, "Real-time onboard 6dof localization of an indoor mav in degraded visual environments using a rgb-d camera," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, USA, May 2015, pp. 5253–5259. [Page 8]

[66] B. D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," in *Intl. Joint Conf. on Artificial Intell.*, Vancouver, Canada, August 1981, pp. 24–28. [Page 10]

[67] L. Kneip, M. Chli, and R. Y. Siegwart, "Robust real-time visual odometry with a single camera and an imu," in *Brit. Mach. Vision Conf.*, 2011. [Page 10]

[68] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. ter Haar Romeny, J. B. Zimmerman, and K. Zuiderveld, "Adaptive histogram equalization and its variations," *Comp. Vis., Graphics, and Im. Proc.*, vol. 39, no. 3, pp. 355 – 368, 1987. [Page 12]

[69] T. Lupton and S. Sukkarieh, "Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions," *IEEE Trans. Robotics*, vol. 28, no. 1, pp. 61–76, February 2012. [Page 12]

[70] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza, "On-manifold preintegration for real-time visual–inertial odometry," *IEEE Trans. Robotics*, vol. 33, no. 1, pp. 1–21, Feb 2017. [Pages 12 and 111]

[71] K. Eckenhoff, P. Geneva, and G. Huang, "Closed-form preintegration methods for graph-based visual–inertial navigation," *Intl. Journal of Robotics Research*, pp. 1–24, 2019. [Page 12]

[72] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart, "Extending kalibr: Calibrating the extrinsics of multiple imus and of individual axes," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Stockholm, Sweden, May 2016, pp. 4304–4311. [Pages 12 and 13]

[73] J. Sola, "Quaternion kinematics for the error-state kalman filter," *arXiv preprint arXiv:1711.02508*, 2017. [Page 14]

[74] Y. Kanazawa and K. Kanatani, "Do we really have to consider covariance matrices for image features?" in *Intl. Conf. on Computer Vision*, vol. 2, Vancouver, Canada, July 2001, pp. 301–306. [Page 23]

[75] Y. Wang, R. Xiong, Q. Li, and S. Huang, "Kullback-leibler divergence based graph pruning in robotic feature mapping," in *Euro. Conf. on Mobile Robots*, Barcelona, Spain, September 2013, pp. 32–37. [Page 28]

[76] M. Mazuran, W. Burgard, and G. D. Tipaldi, "Nonlinear factor recovery for long-term slam," *Intl. Journal of Robotics Research*, vol. 35, no. 1-3, pp. 50–72, 2016.

[77] J. Hsiung, M. Hsiao, E. Westman, R. Valencia, and M. Kaess, "Information sparsification in visual-inertial odometry," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Madrid, Spain, October 2018, pp. 1146–1153. [Page 28]

[78] S. Shen, Y. Mulgaonkar, N. Michael, and V. Kumar, "Initialization-free monocular visual-inertial state estimation with application to autonomous mavs," in *Proc. of the Intl. Sym. on Exp. Robot.*, Marrakech, Morocco, 2014. [Page 31]

[79] T. Qin and S. Shen, "Robust initialization of monocular visual-inertial estimation on aerial robots," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Vancouver, Canada, September 2017, pp. 4225–4232. [Page 32]

[80] D. Nistér, "An efficient solution to the five-point relative pose problem," *IEEE Trans. on Pattern Anal. and Mach. Intell.*, vol. 26, no. 06, pp. 756–777, June 2004. [Page 33]

[81] V. Lepetit, F. Moreno-Noguer, and P. Fua, "Epnp: An accurate o(n) solution to the pnp problem," *Intl. Journal of Computer Vision*, vol. 81, no. 2, July 2008. [Page 33]

[82] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *Intl. Journal of Robotics Research*, vol. 35, no. 10, pp. 1157–1163, 2016. [Page 36]

[83] M. Kaess and F. Dellaert, "Covariance recovery from a square root information matrix for data association," *Robot. Auton. Syst.*, vol. 57, no. 12, pp. 1198–1210, 2009. [Page 44]

[84] E. S. Jones and S. Soatto, "Visual-inertial navigation, mapping and localization: A scalable real-time causal approach," *Intl. Journal of Robotics Research*, vol. 30, no. 4, pp. 407–430, 2011. [Page 44]

[85] Z. Zhang, G. Gallego, and D. Scaramuzza, "On the comparison of gauge freedom handling in optimization-based visual-inertial state estimation," *IEEE Robot. Autom. Letters*, vol. 3, no. 3, pp. 2710–2717, July 2018. [Page 45]

[86] B. Fu, K. S. Shankar, and N. Michael, "Rad-vio: Rangefinder-aided downward visual-inertial odometry," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Montreal, Canada, May 2018. [Page 51]

[87] P. Geneva, K. Eckenhoff, and G. Huang, "Asynchronous multi-sensor fusion for 3d mapping and localization," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Brisbane, Australia, May 2018, pp. 1–6. [Page 61]

[88] S. Lovegrove, A. Patron-Perez, and G. Sibley, "Spline fusion: A continuous-time representation for visual-inertial fusion with application to rolling shutter cameras." in *Brit. Mach. Vision Conf.*, vol. 2, no. 5, 2013, p. 8. [Page 62]

[89] A. Dhawale, K. S. Shankar, and N. Michael, "Hierarchical gaussian distributions for real-time slam," in *Proc. of the Euro. Conf. on Comp. Vis.*, Glasgow, United Kingdom, August 2020. [Page 93]

[90] J. Solà, J. Deray, and D. Atchuthan, "A micro lie theory for state estimation in robotics," *arXiv preprint arXiv:1812.01537*, 2018. [Pages 111 and 112]

[91] B. Triggs, P. F. McLauchlan, R. I. Hartley, and A. W. Fitzgibbon, "Bundle adjustment—a modern synthesis," in *Intl. Workshop on Vision Algorithms*, 1999, pp. 298–372. [Page 123]

[92] A. Savitzky and M. J. E. Golay, "Smoothing and differentiation of data by simplified least squares procedures." *Analytical Chemistry*, vol. 36, no. 8, pp. 1627–1639, 1964. [Page 125]

[93] J. B. Bancroft and G. Lachapelle, "Data fusion algorithms for multiple inertial measurement units," *Sensors*, vol. 11, no. 7, pp. 6771–6798, 2011. [Page 125]

[94] P. Furgale, J. Rehder, and R. Siegwart, "Unified temporal and spatial calibration for multi-sensor systems," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Tokyo, Japan, Nov 2013, pp. 1280–1286. [Page 126]

[95] T. Qin and S. Shen, "Online temporal calibration for monocular visual-inertial systems," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Madrid, Spain, October 2018, pp. 3662–3669. [Page 130]

[96] K. Eckenhoff, P. Geneva, J. Bloecker, and G. Huang, "Multi-camera visual-inertial navigation with online intrinsic and extrinsic calibration," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Montreal, Canada, May 2019. [Page 133]

[97] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, Kobe, Japan, 2009. [Page 143]

[98] G. Guennebaud, B. Jacob, *et al.*, "Eigen v3," http://eigen.tuxfamily.org, 2010. [Page 143]

[99] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000. [Page 143]

[100] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Proc. of the Intl. Sym. of Robot. Research*, Singapore, December 2013. [Page 146]

[101] X. Yang, A. Agrawal, K. Sreenath, and N. Michael, "Online adaptive teleoperation via motion primitives for mobile robots," *Auton. Robots*, vol. 43, no. 6, pp. 1357–1373, 2019. [Page 146]

[102] V. R. Desaraju and N. Michael, "Experience-driven Predictive Control," in *Robot Learn. and Plan. Workshop at RSS*, Ann Arbor, MI, June 2016. [Page 146]

[103] T. Lee, M. Leok, and N. H. McClamroch, "Geometric tracking control of a quadrotor uav on se(3)," in *Proc. of the IEEE Conf. on Decision and Control*, Atlanta, USA, December 2010, pp. 5420–5425. [Page 146]

[104] L. Meier, D. Honegger, and M. Pollefeys, "Px4: A node-based multithreaded open source robotics framework for deeply embedded platforms," in *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Seattle, United States, May 2015, pp. 6235–6240. [Page 146]

[105] R. Mahony, T. Hamel, and J. Pflimlin, "Nonlinear complementary filters on the special orthogonal group," *IEEE Trans. Autom. Control*, vol. 53, no. 5, pp. 1203–1218, June 2008. [Page 146]

[106] Z. Zhang and D. Scaramuzza, "A tutorial on quantitative trajectory evaluation for visual(-inertial) odometry," in *Proc. of the IEEE/RSJ Intl. Conf. on Intell. Robots and Syst.*, Madrid, Spain, October 2018, pp. 7244–7251. [Page 150]