Towards Robust Multi Camera Visual Inertial Odometry

Joshua Jaekel

CMU-RI-TR-20-24

July 2020

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

Thesis Committee:
Michael Kaess, Chair
Simon Lucey
Alexander Spitzer

Submitted in partial fulfillment of the requirements
for the degree of Master of Science.

Keywords: SLAM, state estimation, visual inertial odometry, multi camera

For my grandparents

Abstract

Visual inertial odometry has become an increasingly popular method of obtaining a state estimate on board smaller robots like micro aerial vehicles (MAVs). While VIO has demonstrated impressive results in certain environments, there is still work to be done in improving the robustness of these algorithms. In this work we present a novel multi-camera VIO framework which aims to improve the robustness of a robot's state estimate during aggressive motion and in visually challenging environments. Our system uses a fixed-lag smoother which jointly optimizes for poses and landmarks across all stereo pairs. We propose a 1-point RANdom SAmple Consensus (RANSAC) algorithm which is able to perform outlier rejection across features from multiple cameras. To handle the problem of noisy extrinsics, we account for uncertainty in the calibration of each stereo pair and model it in both our front-end and back-end. The result is a VIO system which is able to maintain an accurate state estimate under conditions that have typically proven to be challenging for traditional state-of-the-art VIO systems. We demonstrate the benefits of our proposed multi-stereo algorithm by evaluating it with both simulated and real world data. We show that our proposed algorithm is able to maintain a state estimate in scenarios where traditional VIO algorithms fail.

Acknowledgments

I would first like to start by thanking my advisor, Michael Kaess. His guidance and support made this work possible, and his leadership created a lab environment which was consistently positive, supportive and enjoyable to work in. To the members of RPL, past and present, thanks for all the help along the way. Zimo, Jack, Eric W, Ming, Suddhu, Monty, Eric D, Chen, Wei, Allie, Akshay, Paloma, Josh, Allison, Akash, Dominic, you've all helped make 2204 a place I'll feel nostalgic about for years to come. To the many others who I've had the pleasure of working with in the Air Lab and on Team Explorer, I look forward to seeing what you all will accomplish in the months ahead. I want to thank my other committee members, Simon Lucey and Alex Spitzer. Your time and feedback is appreciated and I'm certain this thesis is better because of it.

I'm grateful for all my friends in Pittsburgh who've I've gotten the pleasure to know over the past two years. Anish, Vai, Ryan, Eric, and Will, I've enjoyed the conversations and the laughter, but most of all I've enjoyed the foosball. To my roommates Nick and Jackie, thanks for making my time in Pittsburgh constantly entertaining.

Finally to my family, your support that really did make all this possible. Mom and Dad, you've constantly encouraged me to pursue my interests, and I doubt I would be at this place without your guidance. Alana, you've stuck with me for the past six years and you've helped me manage all the ups and downs along the way. I can't wait to see what else is in store for us.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

State estimation is one of the most fundamental problems in robotics. In many cases, core functionalities of a robot such as motion planning, mapping, and control all depend on a reliable state estimate. Several different types of methods exist for the purpose of obtaining an accurate state estimate. Robots can either rely on information from external sensors to estimate their state, such as GPS or motion capture systems, or they can rely on information from their own sensors and attempt to estimate their own motion. In several applications, like robots operating in subterranean or dense urban environments, as seen in Figure 1.1 relying on GPS is not possible. Given the obvious need for a robot to be able to estimate its own state, the next obvious question is what types of sensors would be most useful for state estimation. This question is complex and depends on a long list of different factors including the payload capacity, the available processing power and memory, the cost of the sensors, the duration of navigation and the type of environment. While lidar based odometry has delivered impressive results, they are heavy sensors and often very expensive. For many robots, like micro aerial vehicles (MAVs), it's not feasible to carry a lidar. This motivates the need for robust state estimation using sensors which are both light weight and low cost.

Cameras and inertial measurement units (IMUs) are two of the most popular sensors used to obtain a state estimate, they are both relatively light weight and inexpensive, and when used together they have a very complementary nature. IMUs provide high frequency data which can give useful information about short-term dynamics, while cameras provide useful exteroceptive information about the structure of the environment over longer periods of time. Visual-inertial odometry (VIO) is a technique which uses visual information from one or more cameras, and inertial information from an IMU to estimate the state of a robot relative to some fixed world frame. Specifically, a VIO system aims to estimate the six degree of freedom rigid body transformation between a starting pose and the current pose of the robot. Although VIO frameworks are

Figure 1.1: One of Team Explorer's drones taking part in preparation for the DARPA Subterranean Challenge. The drone is tasked with exploring large underground environments in order to perform search and rescue. Being able to rely on vision as the primary means of obtaining a state estimate would obviate the need to carry heavy sensors like lidar and buy crucial minutes of exploration.

able to obtain accurate state estimates in many environments, improving the robustness of these algorithms remains a significant challenge. In certain environments, such as those with sparse visual features or inconsistent lighting, current VIO algorithms are prone to failure. Furthermore, certain types of fast or aggressive motions can lead to failures in state estimation. In traditional frameworks, where information from only a single monocular camera or single stereo pair is used, a single point of failure is introduced. If the field of view of the camera were to become suddenly occluded or experience rapid exposure changes, the accuracy of the state estimate could drastically decrease or the VIO algorithm could fail all together.

One of the more intuitive methods of improving the robustness of visual inertial odometry is to use more than one camera. If features from one of the cameras were suddenly lost, the VIO algorithm could continue to maintain a state estimate using only features from the other cameras and IMU. Furthermore, if the cameras are configured to have perpendicular optical axes, then when the robot undergoes fast rotation it is possible that at least one of the cameras' optical axes will be closely aligned with the axis of rotation and will be able to track features during the motion. Given that the use of multiple cameras clearly has the potential to improve robustness, we aim to develop a method of using information from multiple cameras effectively.

1.2 Contributions and Organization

Chapter 2 of this thesis covers relevant preliminaries that aid in understanding the contributions of this work. The main technical contributions of this thesis are contained in Chapters 3 and 4. Chapter 3 describes the design and implementation of a traditional single stereo VIO system. Chapter 4 explains strategies to extend the traditional VIO system to incorporate multiple cameras. Finally Chapter 5 provides a summary of the proposed work and results as well as a discussion on relevant future work. The main contributions of this thesis are:

- The description of a smoothing based Visual Inertial Odometry pipeline designed specifically for accuracy and computational efficiency
- An outlier rejection scheme which operates jointly across features from multiple cameras
- To our best knowledge, the first VIO framework which explicitly accounts for uncertainty in camera extrinsics in both the frontend feature handling and backend optimization
- Evaluation of our system on both real world and simulated data

1.3 Scope and Approach

One of the biggest limiting factors surrounding autonomous exploration using MAVs is the reliability of the state estimate. Being able to maintain an accurate state estimate for long periods of time, and being able to trust the state estimate in challenging environments would facilitate longer and more useful flights. Visual inertial odometry is well suited for MAVs and has shown promising potential to solve the state estimation problem for these smaller robots. Currently, the biggest challenges associated with VIO surround its robustness and reliability.

In this thesis we present the design of a multi-camera visual inertial odometry pipeline. The end goal is to develop a system which is able to efficiently make use of information from multiple cameras to achieve a more robust state estimate than traditional VIO pipelines.

The first part of this work describes the design and implementation of a traditional, single stereo, visual inertial odometry system. It includes an in depth explanation of both the front-end feature tracking and handling as well as the back-end optimization. We explore several key design decision, discuss the benefits and trade-offs of several of these choices, and compare the work to existing state of the art methods in literature. This portion of the thesis also serves as a foundation for the later extension to multi-camera VIO. In that part of the work, we explore how to efficiently select features for optimization from multiple cameras, how to properly account for extrinsic uncertainty, and how to incorporate both monocular and stereo cameras into the pipeline.

Chapter 2

Preliminaries

2.1 Factor Graphs in SLAM

2.1.1 Factor Graph Fundamentals

Most current SLAM methods can be separated into two major categories: filtering and smoothing based techniques. Filtering based methods, such as the Kalman Filter, Extended Kalman Filter, and Particle Filter, iteratively attempt to infer the current state of a robot given a belief about its previous states and the measurements made by the robot. Smoothing based methods differ in that they attempt to jointly infer a set of previous states at each iteration instead of only the current state. Factor graphs have been adopted as the state-of-the-art method of representing smoothing based state estimate problems. Factor graphs, like the one shown in Figure 2.1 are bipartite graphs which encode a factorization of a probability density $\phi(X)$.

$$\phi(X) = \prod_{x_i \in X} \phi(x_i) \tag{2.1}$$

Factor graphs are composed of variables to be optimized, and factors which constrain those variables in the optimization. In applications related to SLAM, the factor graph encodes a factorization of the posterior $P(X|Z)$.

$$P(X|Z) \propto \prod_{z \in Z} p(z|X). \tag{2.2}$$

Each factor in the factor graph represents the likelihood of a measurement given the state variables corresponding to that measurement. By choosing to model each factor in the factor graph as a Gaussian distribution over the measurement function, the problem of maximum a posteriori (MAP) inference over the posterior $P(X|Z)$ is equivalent to minimizing a sum of the squared residual errors over the set of factors in the factor graph. If the measurement model function

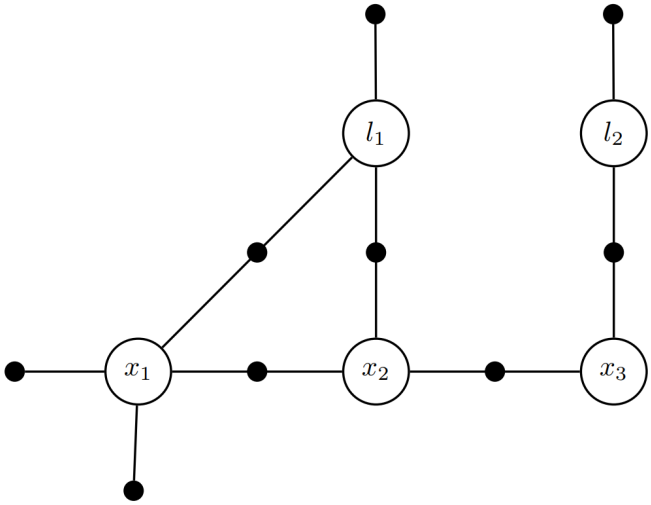

Figure 2.1: A sample factor graph taken from [3], made up of variables to be optimized for, in this case poses x_1, x_2, and x_3, as well as landmarks l_1, and l_2 and factors constraining these variables.

corresponding to factor i is h_i, then we can represent this relationship as,

$$\arg\max_X \prod_{z_i \in Z} p(z_i|X) = \arg\min_X \sum_{z_i \in Z} ||h_i(X) - z_i||^2_{\Sigma_i}. \tag{2.3}$$

Here the term $||h_i(X) - z_i||^2_{\Sigma_i}$ is known as the residual of the factor, and Σ_i is the co-variance associated with the measurement. Under this formulation we can use well known methods of solving non-linear least squares problems, such as Gauss-Newton and Levenberg-Marquardt, to solve the MAP inference problem and achieve a state estimate.

2.1.2 Fixed-Lag Smoothing

As a robot navigates through an environment, the amount of measurements it accumulates will increase. This means that the size of the factor graph will increase over time as well. If we perform batch optimization each time we want to estimate our state then time it takes to perform inference will eventually increase to a point where real time performance is no longer possible. Several techniques have been developed in order to address this problem. Incremental solvers such as iSAM [15] and iSAM2 [16] attempt to make use of previous optimization results in order to compute subsequent state estimates without having to perform a batch optimization at every step. However, even these incremental methods cannot guarantee constant time computation, which is an import requirement for many robot applications which rely on their state estimate for control and obstacle avoidance.

Fixed-lag smoothing attempts to constrain the size of the factor graph by continuously marginalizing out previous states every time a new state enters the optimization window, as shown in

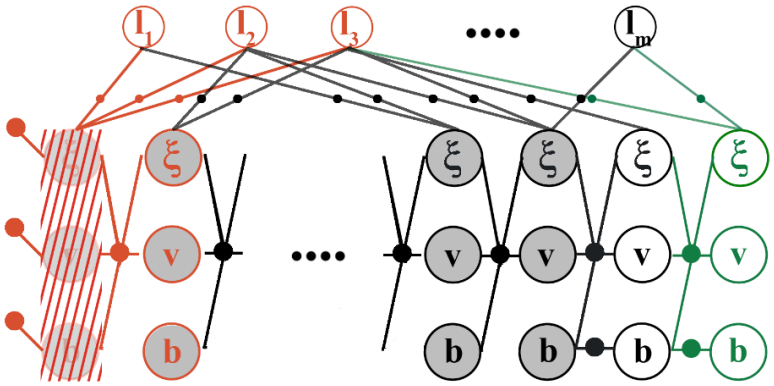

Figure 2.2: In fixed lag smoothing we maintain a window of a fixed number of states by marginalizing out previous states (in red) as new states (in green) are added to the factor graph. Figure is from [14].

Figure 2.2. By doing this the VIO algorithm can maintain a constant number of states in the optimization window, and only perform inference on this smaller factor graph, which is quick to do. This technique works especially well for VIO since most systems do not perform loop closing, and are therefore unlikely to obtain measurements that would effect states in the past, specially if there are few common landmark observations.

While there are a number of practical reasons why fixed-lag smoothing is well suited for visual inertial odometry, there are still drawbacks from an information-theoretic stand point. When we marginalize out a state, it's no longer able to be optimized for and essentially takes on a fixed value corresponding to the current linearization point at the time of marginalization. A draw back of fix-lagged smoothing and the frequent marginalization specifically, is the fill-in that occurs which can slow the optimization over long periods of time. This problem is discussed in greater detail in Section 3.5.

2.2 Paramaterizing 3D Poses

Visual inertial odometry systems typically perform optimization over 6-DOF poses. 6-DOF poses live on a non-linear manifold known as the Special Euclidean Group $SE(3)$. Each member of $SE(3)$ consists of a point in $\mathbb{R}^3$ and a member of the Special Orthogonal Group $SO(3)$, which represents an orientation. Solvers like Gauss-Newton and Levenberg-Marquardt optimize by making incremental adjustment from a fixed linearization point until a solution satisfies some convergence criteria. However, this presents a problem for poses since it's unclear how to make additive incremental changes to a rotation matrix in $SO(3)$. If we naively represent the pose by vectorizing the rotation matrix and concatenating the point vector to create an element in $\mathbb{R}^{12}$ then any additive increment to this vector will almost certainly correspond to a matrix that is not a member of $SE(3)$. This makes intuitive sense, since the the rotation matrix only has 3 degrees

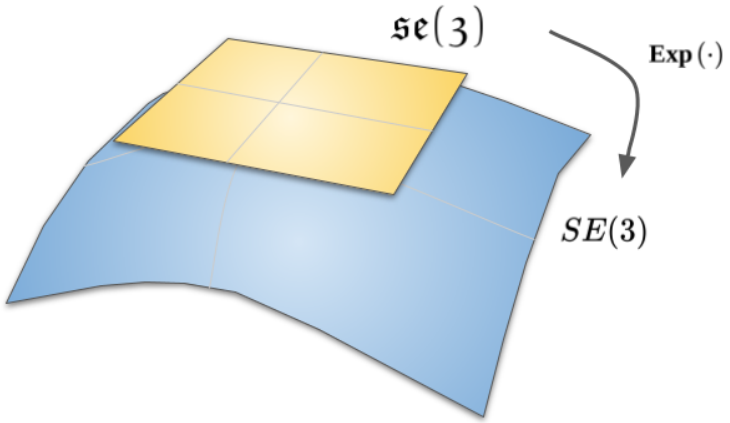

Figure 2.3: Poses live on the non-linear manifold $SE(3)$ (shown in blue). We perform optimization in the Lie algebra $\mathfrak{se}(3)$ (shown in yellow), which is a linear space. We then project the solution back onto $SE(3)$ using the exponential map.

of freedom instead of 9, and the pose only has 6 degrees of freedom instead of 12.

In order to properly perform optimization on 6-DOF poses, we take advantage of the fact that $SE(3)$ forms a Lie group. This means for each pose increment $\xi \in \mathbb{R}^6$, we can associate a member of the Lie algebra using the hat operator. In context of Special Euclidean Group the hat operator is a function which performs the following mapping:

$$\mathbb{R}^6 \rightarrow \mathfrak{g}, \tag{2.4}$$

where $\mathfrak{g}$ is a member of the Lie algebra. We can use the exponential map, which performs the following mapping:

$$\mathfrak{g} \rightarrow G \tag{2.5}$$

where once again $\mathfrak{g}$ is a member of the Lie algebra and G is a member of the associated Lie group. We can use this to map a member of the Lie algebra corresponding to an incremental pose change, $\xi^\wedge$, to a neighbourhood around some initial pose estimate $a \in SE(3)$.

$$a \oplus \xi = a \cdot \exp(\xi^\wedge) \tag{2.6}$$

This process allows us to optimize on poses directly, as a would represent a given linearization point and ξ would represent the incremental update as part of Gauss-Newton or Levenberg-Marquardt step. As we will see in Section 4.5, this parameterization is also extremely useful for associating uncertainty with a 6-DOF transformation.

2.3 RANSAC and Outlier Rejection

We've seen how it's possible to formulate the inference problem on a factor graph as a least squares minimization problem over the residuals associated with each factor. This process works well assuming all of the factors in the factor graph are valid and consistent. However, least squares problems are known to be extremely sensitive to outliers, and as such incorrect measurements in the factor graph can drastically degrade the accuracy of the resulting state estimate. To address this potential problem, many systems attempt to perform robust estimation. While the theory of robust estimation and robust statistics has applications well beyond the domain of robotics, a useful concept is that instead of using all the available measurements to form a model, we attempt to form models based only on the subset of measurements which we deem as "inliers".

For applications related to geometric computer vision specifically, Random Sample Consensus (RANSAC) have proven to be a useful technique for robust estimation. Given a set of discrete points RANSAC works by iteratively repeating the following steps:

1. Randomly select a subset of points, where the size of the subset is the minimum numbers of points required to uniquely generate a model.
2. Generate a model given the selected points.
3. Apply the model to the entire set of points, and identify inliers by determining if a point is consistent with that model.

After repeating this process for a fixed number of iterations, we consider the largest group of inliers to be valid measurements. If we want to obtain a robust estimate for the model then we can take the additional step of generating a final model using all the inliers. Unless we exhaustively iterate through all possible models, RANSAC can only provide probabilistic guarantees on the validity of the inlier set and the corresponding final model. If we start with an assumed probability of a point being an outlier of ϵ, then we can say that the chance of a randomly sampled model being correct is:

$$m_{valid} = (1 - \epsilon)^s, \tag{2.7}$$

where s is the minimum number of points that define a model. A model is defined as being valid if all the points used to generate the model are inliers. This means that the probability the model is invalid is:

$$m_{invalid} = 1 - (1 - \epsilon)^s, \tag{2.8}$$

If we iteratively repeat this process N times, then we can assume that RANSAC will have failed to generate a valid solution if none of the sampled models were valid. If we assume each model was sampled independently then the probability of this occurring, p', is:

$$p' = (1 - (1 - \epsilon)^s)^N \tag{2.9}$$

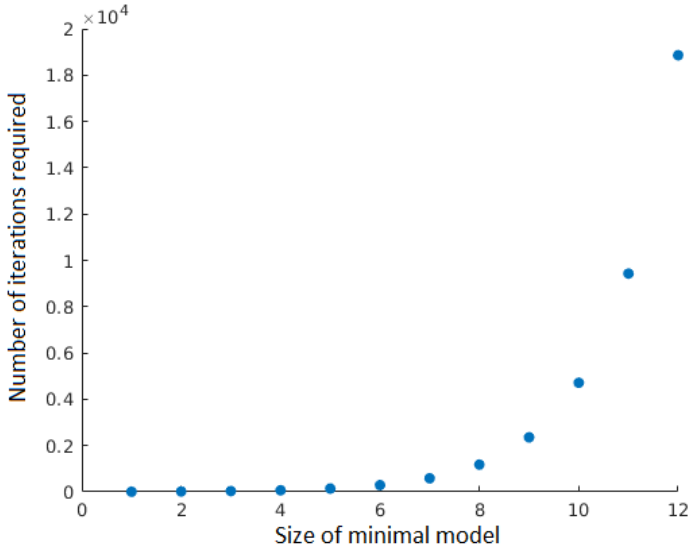

Figure 2.4: The number of iterations needed to reach a confidence of $p = 0.99$, with an assumed outlier percentage of $\epsilon = 0.50$ for different model sizes s computed according Equation 2.10

This means that if we want a certain level of confidence, p, that RANSAC found a valid solution, we need to perform N iterations where

$$N = \frac{\log(1-p)}{\log(1-(1-\epsilon)^s)}, \tag{2.10}$$

where $p' = 1 - p$. This is the standard way of determining the number of iterations necessary for a given confidence level. An important point is that for a fixed confidence level, the number of iterations necessary grows exponentially with the minimal number of points required to generate a model. This can be seen in Figure 2.4. Because of this, reducing the number of points in the minimal model is a very effective way of reducing the amount of computation needed to perform outlier rejection.

RANSAC is especially well suited for problems related to geometric computer vision for a number of reason. First, RANSAC works best when we can rely on the assumption that a majority of the observed points are inliers, which is often the case with measurements from cameras. Furthermore, in these types of problems, minimal parameterizations are often very well defined and because these geometric principles provide straight forward ways of determining inliers given a model.

2.4 Multi View Geometry

2.4.1 Epipolar Geometry

Epipolar geometry is a well studied field that describes the geometric constraints provided by two projective cameras observing the same 3D scene. We refer readers to [10] for a derivation of the underlying principles and a broad discussion about the applications of epipolar geometry. For the purposes of this thesis, it is enough to get an intuition about how the epipolar constraint is derived in order to understand how it's used in triangulation and stereo feature matching.

Given an observation of a single feature point in an image plane, the 3D position of that feature is constrained to lie on a ray defined by connecting the optical center of camera and the observation on the image plane. That means if we know the relative position of another camera in the scene we can project the entire ray onto the image plane of that second camera. The projection forms a line in the image plane, known as the epipolar line. This is illustrated in Figure 2.5.

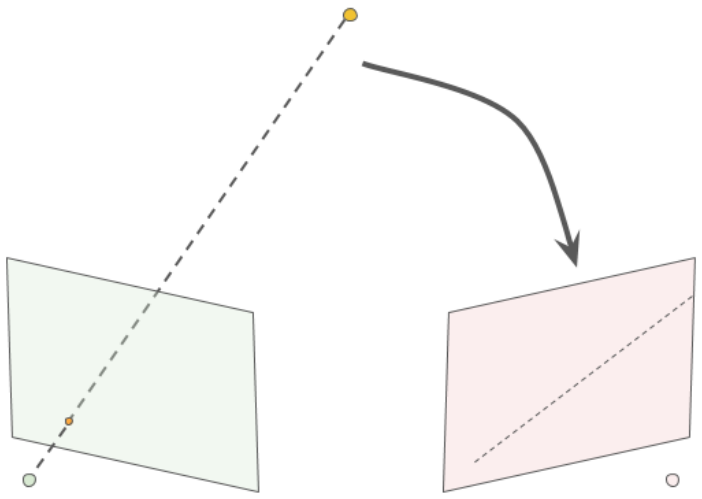

Figure 2.5: A visualization of how the epipolar line in the red image plane is derived by projecting the ray defined by the observation in the green image plane.

If we assume that the relative transformation between the cameras is perfectly known, and that there is no noise in the measurements then we would expect that the observation of the feature in the second image plane would lie on the epipolar line. We can leverage this relationship in two main ways. Given a known point correspondence we can estimate the transformation between camera in order to minimize the distance between observations and their corresponding epipolar lines. Assuming we have an estimate for the transformation we can also use the epipolar constraint to estimate whether two candidate observations of the same feature actually correspond to the same 3D point. We can use the Fundamental matrix, $\mathbf{F}$, in order to evaluate the epipolar constraint. We can construct the Fundamental matrix given only information about

the camera instrinsics and the transformation between optical centers according to the following formula

$$\mathbf{F} = \mathbf{K}_2^{-\top}[\mathbf{t}]_\times \mathbf{R}\mathbf{K}_1^{-1}. \tag{2.11}$$

Once we calculate $\mathbf{F}$, we can determine the distance from the epipolar line as

$$d \propto \hat{\mathbf{x}}_2^\top \mathbf{F} \hat{\mathbf{x}}_1. \tag{2.12}$$

Under perfect conditions, where the observed point lies directly on the epipolar line, we expect to have $d = 0$.

2.4.2 Triangulation

Triangulation is the problem of trying to estimate the 3D position of a point given its position in two or more images. Stereo triangulation specifically deals with an observation from two cameras. In this section we assume that the stereo correspondence has already been made using the methods described in Section 3.4. Geometrically the problem of triangulation can be thought of as the problem of finding the intersection of two rays, each defined by a camera center and the observation of the 3D point on the image plane. In situations where the rays intersect perfectly, we can solve the triangulation easily and uniquely. However, in most situations, any noise in the observations or camera positions means that there won't be an exact solution, and as a result an approximation needs to be made. There are two main strategies of estimating the 3D position of the observed points:

1. Minimize the geometric error
2. Minimize the algebraic error

Assume we are given two observations $\mathbf{x}_1$ and $\mathbf{x}_2$, as shown in Figure 2.6, and corresponding camera projection matrices $\mathbf{P}_1$ and $\mathbf{P}_2$. For a given 3D landmark position $\tilde{X}$, we can define the following simple relationships:

$$\begin{aligned} \hat{\mathbf{x}}_1 &= \mathbf{P}_1 \tilde{X} \\ \hat{\mathbf{x}}_2 &= \mathbf{P}_2 \tilde{X} \end{aligned} \tag{2.13}$$

Using the method described in [10] we can define a cost function corresponding to geometric error as:

$$C(\tilde{X}) = d(\mathbf{x}_1, \hat{\mathbf{x}}_1)^2 + d(\mathbf{x}_2, \hat{\mathbf{x}}_2)^2, \tag{2.14}$$

In order to solve for the triangulated point, this cost function is minimized subject to the epipolar constraint $\hat{\mathbf{x}}_2^\top \mathbf{F} \hat{\mathbf{x}}_1 = 0$. This problem is typically solved using an iterative solver, and as a result is quite expensive to compute.

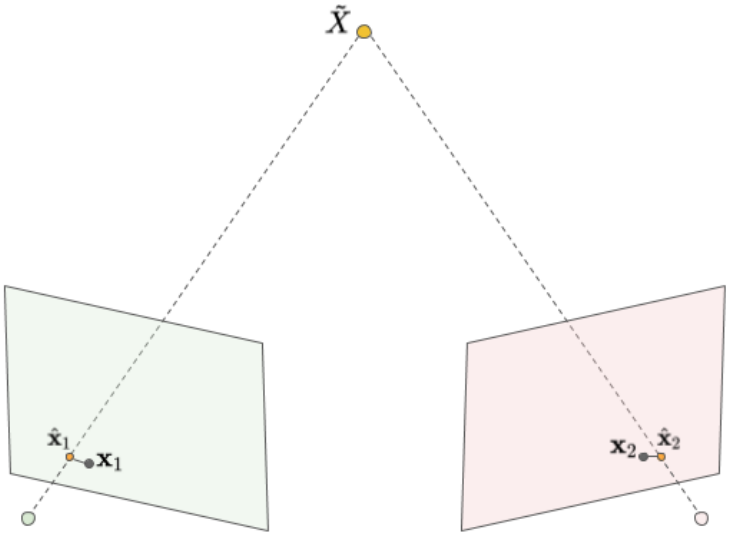

Figure 2.6: An illustration of an estimated 3D point $\tilde{X}$ projected into two camera images. The re-projection error iterative approaches aim to minimize is distance of the straight line between each projected point $\hat{\mathbf{x}}_i$ and the observed point $\mathbf{x}_i$.

An alternative method is to minimize the algebraic error. We take advantage of the fact that the cross product of parallel vectors is zero which means that if Eq 2.13 holds then we can say that:

$$\begin{aligned} \mathbf{x}_1 \times \hat{\mathbf{x}}_1 &= 0 \\ \mathbf{x}_2 \times \hat{\mathbf{x}}_2 &= 0 \end{aligned} \tag{2.15}$$

If we represent row i of $\mathbf{P}_1$ as ${\mathbf{p}^i}_1^\top$ then we can create the following linear system of equations:

$$\begin{aligned} \mathbf{A}\tilde{X} &= 0 \\ \mathbf{A} &= \begin{bmatrix} x_1 {\mathbf{p}^3}_1^\top - {\mathbf{p}^1}_1^\top \\ y_1 {\mathbf{p}^3}_1^\top - {\mathbf{p}^2}_1^\top \\ x_2 {\mathbf{p}^3}_2^\top - {\mathbf{p}^1}_2^\top \\ y_2 {\mathbf{p}^3}_2^\top - {\mathbf{p}^2}_2^\top \end{bmatrix}, \end{aligned} \tag{2.16}$$

where x_i and y_i are the components of $\mathbf{x}_i$. This method is known as DLT. We can solve for $\tilde{X}$ by taking an SVD decomposition of $\mathbf{A}$, and taking the singular vector corresponding to the smallest singular value of $\mathbf{A}$.

DLT obviously has its drawbacks, since there is no guarantee that the error being minimized corresponds to a geometric minimization. The main benefit of DLT is that since it is a linear system, it can be solved extremely quickly. In Section 4.3, we will see how we use DLT as part of the multi-stereo outlier rejection scheme.

Chapter 3

Visual Inertial Odomtery

3.1 Introduction

VIO is a useful tool for estimating the state of a robot given information from a camera and inertial measurement unit. Since VIO algorithms are often used on smaller robots with limited payload capacities and limited computation resources, it's important that the VIO algorithms which are developed are done with the computational constraints in mind. This section provides a detailed explanation of VIO algorithm we've implemented. It includes a discussion about:

- Other state-of-the-art methods, and how our method compares
- The different measurement models used, and how they fit into the factor graph
- The types of features that are tracked
- How we perform temporal feature tracking and stereo tracking
- Strategies to deal with the fill-in created from marginalization

While the scope of this chapter is limited to a traditional VIO system (one with a single camera or single stereo pair), several of the design decisions carry over to the multi-camera case.

3.2 Background and Related Work

VIO and simultaneous localization and mapping (SLAM) algorithms can be roughly categorized into two main groups, *direct* and *indirect* methods. Direct methods [4, 5, 6, 34] estimate temporal motion by continuously aligning consecutive camera frames as to minimize the photometric error between them. On the other hand, indirect methods [17, 24, 27, 31] track landmarks in the

scene and estimate motion by attempting to minimize the reprojection error between the observed location of features in an image and the projection of their 3D estimated locations.

Qin et at.'s VINS-Mono, and its stereo extension VINS-Fusion, [27] is one of the most popular VIO systems used today. It shares several similarities to our method in both the front-end and back-end. It performs KLT tracking on Shi-Tomasi features between consecutive images. It also uses a fixed-lag smoother structure for its backend optimizer. Like many popular methods, the factor graph consists of IMU pre-integration factors and projection factors, which is described in greater detail in Section 3.3.2. For outlier rejection, this system uses a fundamental matrix RANSAC, as described in [10].

Another popular system for visual-inertial odometry is Sun et al.'s [31] implementation of stereo MSCKF [23]. The most fundamental difference between this method and most other state of the art methods is that the state estimate is computed via filtering, not smoothing based backend. The main benefit of using a filtering method is the substantially reduced computational burden. In this implementation, FAST features [29] are used since they are extremely inexpensive to compute. They are tracked from frame to frame and between stereo images using KLT. Despite the fact that [31] has shown that descriptor based methods provide better accuracy overall, Sun et al., argue that the marginal increase in performance is not worth the added computation of descriptor based methods. Because of this we also use KLT methods in place of descriptor matching in our system. A 2-point RANSAC approach described in [33] is used. They first compensate for temporal rotation by integrating the IMU. Instead of performing outlier detection on a triangulated 3D point, they apply an independent RANSAC to both the left and right image points and only accept the feature if it is an inlier in both images.

3.3 Relevant Measurement Models

3.3.1 Camera Measurement Model

For indirect methods, camera measurements usually consist of the observation of multiple point features in the scene. There are several methods for keeping track of these features and handling data association. The method we use is described in greater detail in Section 3.4. The residual corresponding to these measurements is the reprojection error between the observed location of the feature in the image and the projection of the estimated location of the feature in the 3D world into the estimated position of the camera. Since the robot pose, which is a variable in the factor graph, represents the pose of the body frame, the camera extrinsics need to be used to determine the pose of the camera at each measurement. In Section 4.5, we discuss in greater detail how the noise model corresponding to the camera measurements accounts for uncertainty in the extrinsic calibration of the camera. We explore two different landmark parameterizations which each have their own benefits and drawbacks.

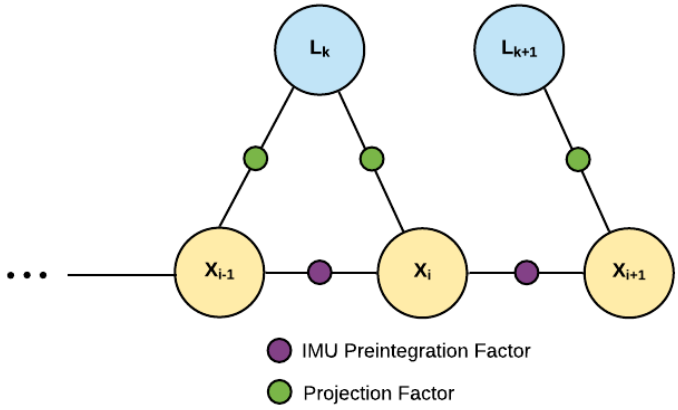

Figure 3.1: A factor graph demonstrating both the full 3D and inverse depth parameterizations of camera projection factors. These factors connect a single landmark and a single pose.

Full 3D Parameterization

The first parameterization is perhaps the most intuitive. Each landmark is represent as a 3D point in world coordinates. We can see in Figure 3.1, that in this parameterization, the projection factor simply connects a landmark with a pose. Given the 3D coordinates of the landmark, we can project the feature into the image given a known body pose and camera extrinsic. The main drawback with this parameterization compared to others is that optimizing for multiple landmarks can be slow. As a rule of thumb, the full 3D parameterization of each landmark is better suited for applications with less strict computational constraints.

Inverse Depth Parameterization

The inverse depth parameterization as described in [2], assumes that the landmark lies on a ray defined by the first observation at the first pose it was observed at. The 3D position of the landmark is completely defined by the inverse depth along that ray. The result of the optimization corresponding to a landmark is a single number, the inverse depth, which described the position of the landmark on the ray. The obvious drawback is that if the estimated pose at the first observation is incorrect, or if the observation itself is wrong then the true position of the landmark will likely not fall on the ray. If enough landmarks are initialized incorrectly, it has the potential to significantly degrade the accuracy of the state estimate. This further motivates the need for a strict outlier rejection step and a conservative approach to selecting features to use in an optimization.

Another relevant parameterization, shown in Figure 3.2 is the one used in VINS-Mono and VINS-Fusion [27]. It follows an inverse depth parameterization, but also includes the pose of the first observation as part of the factor. It has the added benefit of being an inverse depth

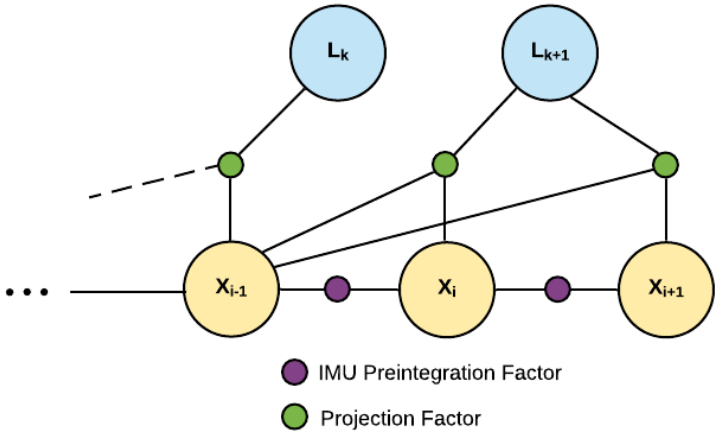

Figure 3.2: An alternative parameterization of the inverse depth factor. The main difference is that the pose of the first observation is included as part of the optimization of each factor. While this may provide improved accuracy, it creates a more densely connected structure which may slow down optimization.

parameterization which is efficient for optimization, but also allows for the initial pose to be refined during optimization which in turn refines the ray on which the landmark is constrained to. While this parameterization could be a useful middle ground the two approaches we explored, we found that in most applications either the full 3D or inverse depth parameterization suffices.

3.3.2 IMU Measurement Model

Data from inertial measurement units are extremely useful in VIO frameworks. Since IMUs are typically sampled at a higher rate than cameras, they provide information about the short term dynamics that can often be missed by visual measurements alone. IMU measurements between consecutive poses in the factor graph are collected and combined into a single factor. A naive measurement model corresponding to the inertial measurements would be to start at the first pose, integrate the IMU measurements to arrive at an estimated pose and then compute a residual between that estimate and the actual value of the pose at a given particular linearization point. However, this method requires reintegration of the IMU measurements every time the residual corresponding to that IMU factor needs to be evaluated. Since the residual is re-computed for different linearization points several times during a single optimization, this method becomes extremely inefficient and in practice makes this type of factor unusable for real time applications. In 2012 Lupton et al. [21] introduced the theory of IMU preintegration, and in 2016 Forster et al. [7], extended the theory to cover optimization on the SE(3) manifold. We refer to [7] for a derivation of this preintegrated IMU factor. The main benefit of the preintegrated factor is that under that particular formulation IMU measurements only need to be integrated once, in the body frame of the robot, rather than each time the residual is evaluated. This significantly reduces the

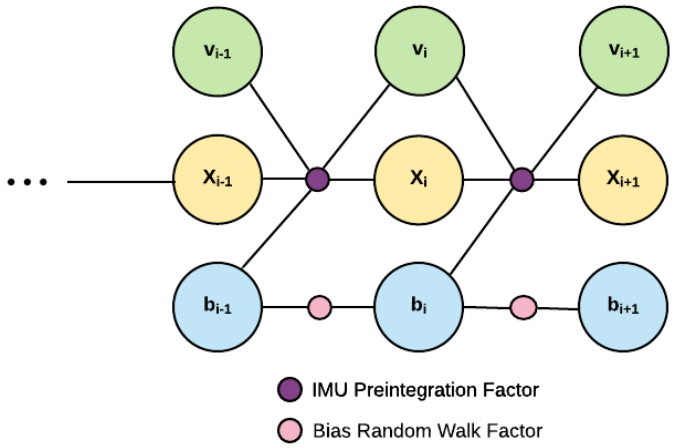

Figure 3.3: A factor graph depicting a detailed structure of the IMU preintegration factor used for this work. Often this structure is simplified by combining the preintegation factor and bias random walk factor into a single constraint between consecutive states.

computational burden required to use IMU measurements in a factor graph and have therefore become the standard method of using inertial data for localization.

The IMU preintegration factor connects consecutive states in the factor graph and computes a residual based on the previous pose, velocity, IMU bias, and the next pose and velocity. Some implementations of the IMU preintegration factor will also include a "random walk" factor between consecutive bias values, while other require that factor to be declared separately. Figure 3.3 is an illustration of how the preintegration factor connects two consecutive states. For clarity we explicitly represent the pose, velocity, and bias terms as separate nodes.

We choose to model the noise associated with the IMU factor with 4 parameters. The parameters are the white noise and bias random walk parameters for both the accelerometer and gyroscope. Noise from IMU data is modelled by a time varying bias and an additive zero-mean Gaussian white noise. The white noise parameter is the standard deviation of the additive white noise, while the random walk parameter describes how quickly the bias term changes. It's the random walk parameter which constrains two consecutive bias terms in the factor graph. All of these parameters can either be obtained from a manufacturer data sheet or estimated through an IMU calibration procedure such as [28].

3.4 Feature Tracking

A important aspect of the frontend of the VIO system is the feature tracking pipeline. There are several key design choices in the feature tracker that impact the overall performance of the state

estimate. These decision range from the types of features that are tracked, methods of tracking features temporally, and methods of determining stereo correspondences.

3.4.1 Types of features

Several computer vision applications require the detection of important pixels in the image. One of the most important properties of these image points is that they must be easy to identify, and locally unique so that they aren't confused with nearby image points during tracking. Visually these points usually correspond to corners in the image. Corners are points that have strong spatial gradients in more than one direction. There are several existing types of corners which vary in how they are extracted from the image. Popular features include Harris corners, Fast features, SURF features, SIFT features, and Shi-Tomasi corners. We will focus on Shi-Tomasi corners since they were used in this work.

In [30] Shi and Tomasi propose the notion that the features which should be used in structure from motion problems are more than those points which exhibit a high corner response, but specifically those points which are easy to track. This may seem like a minor distinction, since often points with a high corner response are also the easiest to track, but in their work they demonstrate that the assumption is not always true. Shi-Tomasi corners introduce a quality factor associated with each feature related to how well it can be tracked by accounting for the texture of the image patch surrounding the feature. We refer to [30] for a full derivation.

3.4.2 KLT Tracking

The Kanade-Lucas-Tomasi (KLT) feature tracker is combination of the work in [30] as well as the work in [20]. Features are tracked by performing Lucas-Kanade registration on local patches surrounding each corner. In our application we use KLT tracking to get both temporal correspondences and stereo matches.

In order to reduce the number of features which need to be tracked, we perform feature bucketing. We separate each image into a fixed number of rectangular bins, and set a minimum and maximum limit on the number of features allowed in each rectangle. If the number of features in a bucket falls below the minimum threshold, we initialize new features in that bucket. If the number of features in a rectangle exceeds the maximum threshold, we discard features starting with the most recently initialized features. This is done on the assumption that older features which were observed by the most number of previous poses provide the strongest constraints on the optimization. In general feature bucketing provides a straight forward way of limiting the number of features in the image while still ensuring that features are evenly distributed so that the measurements don't provide redundant constraints. Temporal matching is performed using the following procedure:

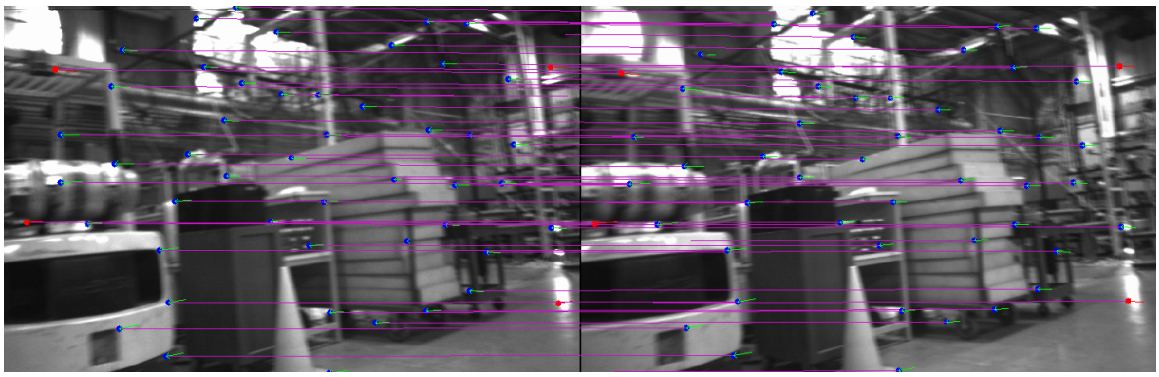

Figure 3.4: A example of the stereo matching between the left and right images of the forward facing camera on the experimental camera rig shown in Figure 4.5. The arrows show the temporal motion of the features, and red features denote outliers from the joint RANSAC.

1. Perform KLT tracking between features in the previous left image and features in the current left image. We initialize KLT according the the temporal rotation of the robot as measured by the IMU.
2. Perform stereo matching between the features which were successfully tracked temporally.
3. Replenish buckets which lost features with new Shi-Tomasi corners.

The main benefit of using KLT for stereo matching as opposed to descriptor based methods is the saved computation, a trade-off which is also made by [31]. However, this method relies on the assumption that the disparity of the observed features is relatively small. If our stereo baseline is too large, or if the robot gets too close to objects in the scene then KLT may fail to find valid stereo correspondences. Although features are tracked using KLT, we still verify the distance between each tracked feature and the epipolar line before accepting the match.

3.5 Marginalization and Sparsification

As we discussed in Section 2.1.2, the process of fixed-lag smoothing requires us to marginalize out a previous state every time a new state enters the optimization window in order to maintain a window of states of constant size. For a general joint probability distribution $p(x, y)$, the process of marginalizing out variable x can be achieved simply by integrating the entire probability density function over all values of x to obtain a new probability density function which only depends on y.

$$p(y) = \int_x p(x, y) \tag{3.1}$$

Since a factor graph represents a factorization of joint probability density function as discussed in Section 2.1.1, marginalizing out variables in the factor graph is analogous to marginalization for general probability distributions. If we represent our probability function in *covariance form*,

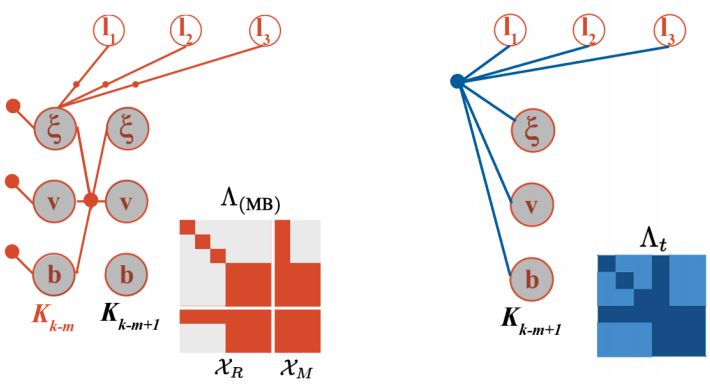

Figure 3.5: The variables to be marginalized are shown on the left. On the right we see the dense prior which is created after marginalization. Also shown is the fill-in created in the information matrix. The image is taken from [14].

then we can obtain the new covariance matrix after marginalization trivially since it exists as a sub-block of the original covariance matrix. However, if we choose to represent our probability distribution in *information form* then we need to take a Schur complement in order to obtain the new information matrix after marginalization. A more detailed explanation of this process can be found in [3].

A major draw-back of constantly marginalizing out variables is that over time the resulting information matrix becomes less sparse. This is because marginalization creates densely connected priors as is illustrated in Figure 3.5. Since the efficiency of back-end solvers relies heavily on the sparsity of the information matrix, over time it will take longer to perform inference on the factor graph. This can pose a problem for applications where a robot needs to navigate for longer periods of time. To address this issues, several existing methods, such as VINS [27] and OKVIS [19] selectively discard measurements. While this approach succeeds in maintaining sparsity, it throws away what could be valuable information that can't be recovered.

Instead of selectively discarding measurements, we adopt the marginalization strategy proposed in [14]. This method maintains sparsity by enforcing a sparse topological factor graph structure around the marginalized variables and then minimizing the Kullback-Leibler divergence between the probability density function induced by the original densely connected prior and the new sparse factor graph structure. This maintains sparsity while also maintaining more of the information contained in the original dense prior.

Chapter 4

Incorporating Multiple Cameras

4.1 Introduction

Visual inertial odometry often suffers from a lack of robustness, and is prone to failure in certain types of environments such as those with sparse visual features or inconsistent lighting. Furthermore, certain types of fast or aggressive motions can lead to failures in state estimation. In indirect systems which track features in the scene, these failures can often be attributed to poor feature tracking which results in incorrect camera measurements being used in back-end optimization. In traditional frameworks, where information from only a single monocular camera or single stereo pair is used, a single point of failure is introduced. If the field of view of the camera were to become suddenly occluded or experience rapid exposure changes, the accuracy of the state estimate could drastically decrease or the VIO algorithm could fail all together.

To address these problems we explore the possibility of using information from multiple cameras in order to improve the robustness of a robot's state estimate. If features from one of the cameras were suddenly lost, the VIO algorithm could continue to maintain a state estimate using only features from the other cameras and IMU. Furthermore, if the cameras are configured to have perpendicular optical axes, then when the robot undergoes fast rotation it is possible that at least one of the cameras' optical axes will be closely aligned with the axis of rotation and will be able to track features during the motion. Since a major focus of this work is to improve the overall robustness, we decided to develop a method which uses information from multiple cameras in a single optimization. This is in contrast to other methods, which are expanded on in greater detail in Section 4.2, which obtain independent state estimates from different cameras and fuse them together. While these methods may achieve a relatively accurate state estimate, each estimation running on individual cameras suffers from potential issues with robustness. In fact, these methods are often prone to failure if even one of the estimates is significantly inaccurate.

Having established that using multiple cameras can be useful for visual inertial odometry pipelines, there are still questions about the best ways of using this information. This chapter

describes a methods of incorporating measurements from multiple cameras and an inertial measurement unit into a single joint optimization. While there are several aspects of a traditional VIO framework that are transferable to the mutli-camera case, there are still several components of the pipeline which need to be specifically addressed. The main contributions of this chapter are as follows:

1. A description of the necessary modifications to a traditional VIO framework in order to accommodate multiple cameras
2. The design of an outlier rejection scheme which is able to jointly consider features from all cameras
3. A demonstration of the problems which arise from uncertain extrinsic calibrations and a proposed strategy to handle them

4.2 Background and Related Work

While the idea of improving the robustness of localization using multiple cameras isn't new, the methods in which the information is fused varies. Oskiper et al. [25] proposed a multi-stereo VIO which extracts frame-to-frame motion constraints through a 3-point RANSAC and used an extended Kalman filter (EKF) to fuse those constraints with data from an IMU. Houben et al. [13] explored using a multi-camera system in a graph based SLAM framework with their proposed extension of ORB-SLAM [24]. Their system added a factor in the pose graph between key-frames observed from different cameras at the same time step based on the known extrinsic calibration of the multi-camera system. Tribou et al. [32] proposed a multi-camera extension of Parallel Tracking and Mapping [17] (PTAM) using a spherical camera model.

In [9], Müller et al. propse a multi-camera VIO system which calculates independent odometry estimates for each camera and fuses them with data from an inertial measurement unit using a Kalman filter. They perform outlier rejection on each odometery measurement by thresholding the Mahalanobis distance between the actual odometry measurement and the predicted measurement from the filter. They claim that a main advantage of fusing independent odometries is they are able to select key-frames for each camera independently.

For joint multi-camera outlier rejection, most existing methods use the generalized camera model (GCM) and generalized epipolar constraint (GEC) introduced by Pless in [26]. In this framework, feature points are parameterized by Plücker vectors which pass through the optical center of the camera in which the feature was observed and the normalized image point. Lee et al. [18] propose a 4-point solution based on the GEC for a multi-camera setup on board an autonomous vehicle. This system assumes the roll and pitch can be directly measured from the IMU but estimates the temporal yaw as part of the RANSAC formulation. In [12] Heng et al. propose a similar 3-point algorithm for a multi-stereo system on board a MAV. Like our proposed method, they also use an estimated rotation from IMU integration, but their algorithm

is degenerate in the case of no temporal rotation and no inter-camera correspondences. Although their platform contains stereo cameras, they do not triangulate feature points and instead must treat each camera in the stereo pair independently to ensure there will always be inter-camera correspondences.

In a separate work [11], Heng et al. describe a 1-point RANSAC scheme similar to ours in that it uses rotation measured from the IMU and estimates the relative translation between 3D features observed from a RGB-D camera as the RANSAC model. Our work extends theirs by formulating how this 1-point RANSAC scheme can be used for joint multi-camera outlier rejection. We also characterize uncertainty in both stereo triangulation and camera extrinsics as part of our RANSAC.

4.3 Multi-Stereo RANSAC

4.3.1 Motivation

Determining the correct set of features to use in a back-end optimization is a non-trivial task. Although several outlier rejection algorithms exist in traditional VIO pipelines, most of these methods cannot take advantage of the strong constraints provided by a calibrated multi-stereo system, and as a result perform outlier rejection independently for each camera. Outlier rejection methods which rely on RANSAC attempt to find the largest subset of consistent features, and assume that these features are consistent with the global motion of the robot. However, there are several scenarios where the largest subset of features which are locally consistent for a particular camera may not be globally consistent with the entire set of features across multiple cameras. We can imagine a situation where a camera's field of view largely captures a moving object, and a majority of the features tracked in that camera lie on the moving object. In this scenario, which is plausible in certain dynamic environments, an outlier rejection scheme which is unable to identify these inconsistent features as outliers, would add them as constraints in the back-end optimization which would likely result in a unreliable state estimate. However, if other cameras on the robot were able to obtain accurate feature tracks of static features in the environment, then by performing a joint RANSAC on all of the features would make it much more likely to identify outliers.

As with most aspects of a VIO pipeline, the computational resources demanded by the outlier rejection scheme needs to be considered. The number of iterations necessary for a given confidence level scales exponentially with the numbers of correspondences required in the minimal model according to Equation 2.10. A main advantage of our proposed outlier rejection scheme is that the RANSAC algorithm only requires a single correspondence for the minimal model. This is done by taking advantage of the stereo observation of each feature. This means that we can satisfy a given confidence level of RANSAC with the smallest amount of iterations possible.

4.3.2 Assumptions and Limitations

Figure 4.1 provides an outline of the structure of the multi-stereo VIO pipeline.

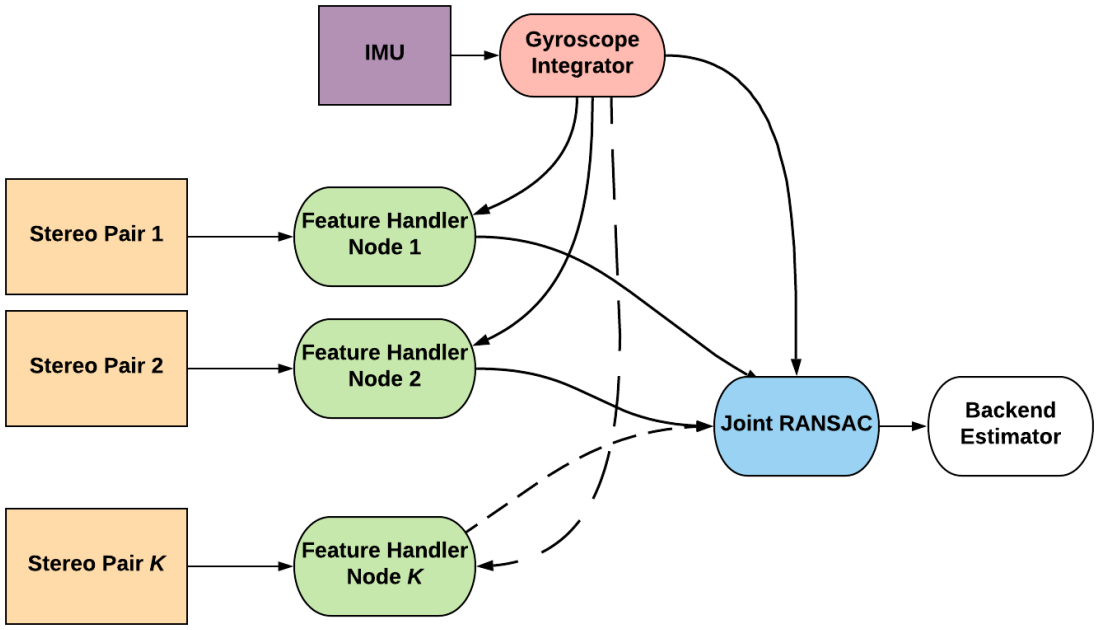

Figure 4.1: A flowchart describing the multi-stereo VIO pipeline. Features are tracked in each stereo pair individually using the method described in Section 3.4. We then perform joint outlier detection and pass inliers to a backend for optimization. An extension of this pipeline to the integration of monocular cameras is described in Section 4.4.

4.3.3 Notation and Conventions

Table 4.1 is a summary of the notation used in this section. We start by defining a set of camera measurements $\mathcal{O}_t$ that contains all the measurements across the K stereo cameras:

$$\mathcal{O}_t = \bigcup_{j=1}^{K} \mathcal{O}_t^j. \tag{4.1}$$

$\mathcal{O}_t^j$ is the subset of $\mathcal{O}_t$ containing the measurements observed in stereo pair j. The goal of the proposed outlier rejection algorithm is to filter the set of candidate landmarks, $\mathcal{O}_t$, and extract a smaller subset of features $\mathcal{C}_t$, to be added as stereo projection factors in the optimization such that each feature in $\mathcal{C}_t$ is consistent with the motion of the robot. With this formulation we can define $\mathcal{O}_t^j$ as follows,

$$\mathcal{O}_t^j = \mathcal{C}_{t-1}^j \cup \mathcal{N}_{t-1}^j, \tag{4.2}$$

Table 4.1: Summary of the notation used to describe the multi-stereo RANSAC algorithm in Section 4.3

Problem Formulation	
$\mathbf{p}_L^t, \mathbf{p}_R^t \in \mathbb{R}^2$	Left and right candidate feature image coordinates for stereo pair at time step t
$\mathbf{p}_L^{t-1}, \mathbf{p}_R^{t-1} \in \mathbb{R}^2$	The time step $t-1$ image coordinates corresponding to $\mathbf{p}_L^t, \mathbf{p}_R^t$
S_i	The i-th stereo pair
$\mathbf{T}_{S_i}$	Extrinsics of the left camera of S_i with respect to the body frame
$\mathcal{O}_t^j$	Set of potential features to track at time step t in stereo pair j
$\mathcal{N}_t^j$	Set of new feature points added at time step t in stereo pair j
$\mathcal{C}_t$	Final set of image points to be used for VIO at time step t
Multi-Camera RANSAC	
$\mathcal{F}_t$	Set of successfully temporally tracked image point pairs
$\mathbf{P}_B^t \in \mathbb{R}^3$	Triangulated 3D coordinate of a feature in the body frame at time step t
$\mathbf{P}_B^{t-1} \in \mathbb{R}^3$	The time step $t-1$ triangulated 3D feature coordinate corresponding with $\mathbf{P}_B^t$ represented in the body frame
$\mathbf{P}_B^{(t-1)'} \in \mathbb{R}^3$	The 3D feature coordinate $\mathbf{P}_B^{t-1}$ after being rotated into the current (time t) frame via $\hat{\mathbf{R}}_t$
$\mathcal{I}$	Set of candidate inliers for a given iteration of RANSAC
$\mathcal{X}$	Set of triangulated feature points
$\hat{\mathbf{R}}_t \in \mathrm{SO}(3)$	Estimated temporal rotation matrix produced via IMU integration
$\hat{\mathbf{\Upsilon}}_{\tilde{\mathbf{p}}_L}$	Covariance matrix in image pixel space
$\hat{\mathbf{t}} \in \mathbb{R}^3$	Candidate temporal translation from RANSAC
δ	RANSAC threshold
π_j	Projection function into stereo pair j

where $\mathcal{N}_{t-1}^j$ represents the new features that were added at the previous iteration. This is to say that at any time step t, O_t^j will contain the features that were successfully tracked from the previous image, and new features that were added. At each new image we:

(i) Perform KLT tracking from features in previous left image ($\mathcal{O}_t^j$) to the current left image.

(ii) Perform KLT tracking from the successfully tracked features in the current left image to the current right image. The result is $\mathcal{F}_t^j$.

(iii) Replenish the buckets which lost features during Steps i and ii by adding new Shi-Tomasi features ($\mathcal{N}_t^j$).

The set $\mathcal{F}_t^j$ contains the features that were successfully tracked, while $\mathcal{C}_t^j$ is the subset of $\mathcal{F}_t^j$ that were marked as inliers during RANSAC.

4.3.4 Algorithm

The multi-stereo RANSAC process is described with reference to Algorithm 1. We triangulate each candidate feature point in $\mathcal{F}_t$ in its respective stereo frame. This is done for the features in the current image (line 7) as well as the corresponding features from the previous image (line 6). We estimate the temporal rotation, $\hat{\mathbf{R}}_t$ between consecutive camera frames by integrating measurements from the on board gyroscope (line 3). Using this estimate for temporal rotation we rotate the triangulated points from the previous time step into the current time frame (line 10). At this point we expect that the landmarks in $\mathbf{P}_B^{(t-1)'}$ and $\mathbf{P}_B^t$ only differ by the temporal translation of the robot. We obtain an estimate for the temporal translation by randomly selecting a single feature correspondence and subtracting their 3D positions (line 16 and 17). Using this estimate for translation, we then project all the triangulated feature points in the previous temporal frame into the current image frame (line 20). In this step we also calculate a covariance in the pixel space of the image based on the uncertainty of the extrinsic parameters. This is described in more depth in Section 4.5. We perform outlier rejection by thresholding the Mahalanobis distance between the projected points in the left camera frame and the tracked points (line 21). Our RANSAC based outlier rejection scheme iteratively repeats this process and selects the largest set of inliers to insert as measurements in the factor graph.

4.4 Incorporating Monocular Cameras

4.4.1 Assumptions and Limitations

In order to generalize our system to be a true multi-camera VIO algorithm it's necessary to incorporate monocular cameras. While stereo cameras provide extremely valuable information about observed features that monocular cameras cannot, in real world applications there are

Algorithm 1: Multi-Stereo RANSAC

$\mathcal{X} \leftarrow \emptyset$
$\mathcal{C}_t \leftarrow \emptyset$
$\hat{\mathbf{R}}_t \leftarrow$ IMUIntegration()
for $j := 1$ *to* K **do**
  for $(\mathbf{p}_L^{t-1}, \mathbf{p}_R^{t-1}, \mathbf{p}_L^t, \mathbf{p}_R^t) \in \mathcal{F}_t^j$ **do**
    $\mathbf{P}_{S_j}^{t-1} \leftarrow$ Triangulate$(\mathbf{p}_L^{t-1}, \mathbf{p}_R^{t-1})$
    $\mathbf{P}_{S_j}^{t} \leftarrow$ Triangulate$(\mathbf{p}_L^{t}, \mathbf{p}_R^{t})$
    $\mathbf{P}_B^t \leftarrow \mathbf{T}_{S_j}^B \mathbf{P}_{S_j}^t$
    $\mathbf{P}_B^{t-1} \leftarrow \mathbf{T}_{S_j}^B \mathbf{P}_{S_j}^{t-1}$
    $\mathbf{P}_B^{(t-1)'} \leftarrow \begin{bmatrix} \hat{\mathbf{R}}_t & \mathbf{0} \\ \mathbf{0} & 1 \end{bmatrix} \mathbf{P}_B^{t-1}$
    $\mathcal{X} \leftarrow \mathcal{X} \cup \{(\mathbf{P}_B^{(t-1)'}, \mathbf{P}_B^t, \mathbf{p}_L^t, \mathbf{p}_R^t, j)\}$
  end
end
for $i := 1$ *to* N **do**
  $\mathcal{I} \leftarrow \emptyset$
  $(\hat{\mathbf{P}}_B^{(t-1)'}, \hat{\mathbf{P}}_B^t, \dots) \xleftarrow{Rand} \mathcal{X}$
  $\hat{\mathbf{t}} \leftarrow \hat{\mathbf{P}}_B^t - \hat{\mathbf{P}}_B^{(t-1)'}$
  for $(\mathbf{P}_B^{(t-1)'}, \mathbf{P}_B^t, \mathbf{p}_L^t, \mathbf{p}_R^t, j) \in \mathcal{X}$ **do**
    $\tilde{\mathbf{P}}_B \leftarrow \mathbf{T}_{\hat{\mathbf{t}}} \mathbf{P}_B^{(t-1)'}$
    $(\tilde{\mathbf{p}}_L^t, \mathbf{\Upsilon}_{\tilde{\mathbf{p}}_L}) \leftarrow \pi_j(\mathbf{T}_B^{S_j} \tilde{\mathbf{P}}_B)$
    if $(||\tilde{\mathbf{p}}_L^t - \mathbf{p}_L^t||^2_{\mathbf{\Upsilon}_{\tilde{\mathbf{p}}_L}} < \delta)$ **then**
      $\mathcal{I} \leftarrow \mathcal{I} \cup \{(\mathbf{p}_L^t, \mathbf{p}_R^t)\}$
    end
  end
  if $(|\mathcal{I}| > |\mathcal{C}_t|)$ **then**
    $\mathcal{C}_t \leftarrow \mathcal{I}$
  end
end
return $\mathcal{C}_t$

several practical reasons why monocular cameras may be used instead of stereo cameras. On small mobile robots with both limited compute and limited payload capacity, it's often difficult to include the extra hardware required for stereo cameras and allocate the necessary computational resources to process images from the extra cameras. In order to integrate monocular cameras into the system there are 3 key considerations that need to be made.

1. Measurement model and integration into the factor graph
2. Landmark initialization
3. Integration into outlier rejection scheme

As will be discussed greater detail in subsequent sections, we notice that monocular features can be easily integrated into the factor graph, but it is less obvious how they fit into the multi-

stereo RANSAC algorithm. This is because a key aspect of the multi-stereo RANSAC is the ability to triangulate features and retrieve the scale of translation. Because of this our system accepts monocular cameras with the assumption that the robot has at least one stereo camera pair.

4.4.2 Landmark Initialization

Given an observation of a 3D landmark from a stereo camera, it's possible to triangulate that landmark and use the extrinsic calibration of the camera to obtain an estimate of either the 3D position of the landmark or the associated inverse depth, depending on the parameterization being used. For monocular cameras this isn't possible. Given only a single observation of a landmark from a monocular camera, the position of that features is only constrained to lie on a ray but it's impossible to determine where on that ray the landmark is located.

Since a single observation of from a monocular camera doesn't provide enough information to constrain either the 3D position or inverse depth of the landmark, it's not possible to initialize the landmark in the factor graph on the first observation. Because of this, when dealing with monocular cameras there is additional book-keeping step that needs to happen in the back-end.

Once we have an initial observation of a monocular feature, we keep track of the measurement and the associated pose of the robot at the time of the observation. The front-end feature tracker will continue to track the feature over incoming frames and provide measurements from subsequent poses. We store each incoming measurement and associated pose. At each new measurement we check to see if there is enough parallax between the current pose and the pose of the robot during the original observation such that triangulation will be well constrained. If we are able to triangulate we do so and obtain an estimate of the location of the landmark. At this point we can initialize the landmark variable in the factor graph. When this happens we iterate through the previously stored measurements and add factors if the associated poses are still in the optimization window. Once this initialization step is complete additional measurements can be added as they come in and monocular observations are treated the same way as stereo observations.

4.4.3 Outlier Rejection

The multi-stereo outlier rejection scheme has the benefit of being able to jointly select inliers from multiple stereo cameras. However, as part of the multi-stereo RANSAC formulation, it's necessary to triangulate features using DLT. Since this step is obviously not possible given only monocular observations of feature points, we must determine how to incorporate monocular features into the joint RANSAC.

As explained previously, the RANSAC model used to determine inliers is the temporal trans-

lation of the robot between consecutive frames. Given feature correspondences from a single monocular camera, it is only possible to solve for this translation up to scale. If we once again assume the rotation of the robot is known from short term integration of the IMU measurements, then the translation up to scale can be solved with a 2-point algorithm as described in [33]. While we can use an epipolar constraint to determine inliers among features in that camera, it's not possible to use the model to jointly determine inliers from other cameras. However it is possible to use the RANSAC model from a single stereo correspondence (which gives exact translation) to evaluate inliers in a monocular camera. Therefore, in order to perform joint RANSAC, we can only sample models from the stereo cameras. In the scenario where there are few or no valid correspondences observed in a stereo camera then a 2-point RANSAC is performed independently on the features from each monocular camera. We identify this scenario when the total number of inliers generated from the best stereo model is less than the total number of features observed from a monocular camera.

4.5 Accounting for Extrinsic Uncertainty

4.5.1 Motivation

Obtaining an accurate extrinsic calibration between multiple sensors is a significant challenge in robotics in general, and is especially difficult for systems with multiple cameras and inertial sensors. In the context of our RANSAC based outlier rejection scheme, an uncertain extrinsic calibration can lead to a higher re-projection error. In the traditional single camera case, extrinsic uncertainty can usually be dealt with by adjusting the inlier selection threshold in RANSAC. However, in the multi-camera case, we have the added problem that some cameras may have a more accurate calibration than others. In these scenarios simply adjusting the inlier selection threshold does not resolve the problem. Without addressing this issue directly, our RANSAC scheme will have an inherent bias towards cameras with more accurate extrinsics.

We demonstrate the problem using a simulated MAV in an outdoor environment using the RotorS simulator. In the simulated environment we have direct control over the camera extrinsics, and can easily perturb them to see the effects. The results are shown in Figure 4.2. In this demonstration 3 different experiments were run:

1. Both cameras with perfect extrinsics
2. One stereo pair with noise artificially added to extrinsics running without uncertainty compensation
3. One stereo pair with noise artificially added to extrinsics running with uncertainty compensation

In this two stereo configuration, we will analyze the percentage of inliers identified in each stereo pair. We notice that when we add noise to the extrinsics of one of the cameras, the amount of

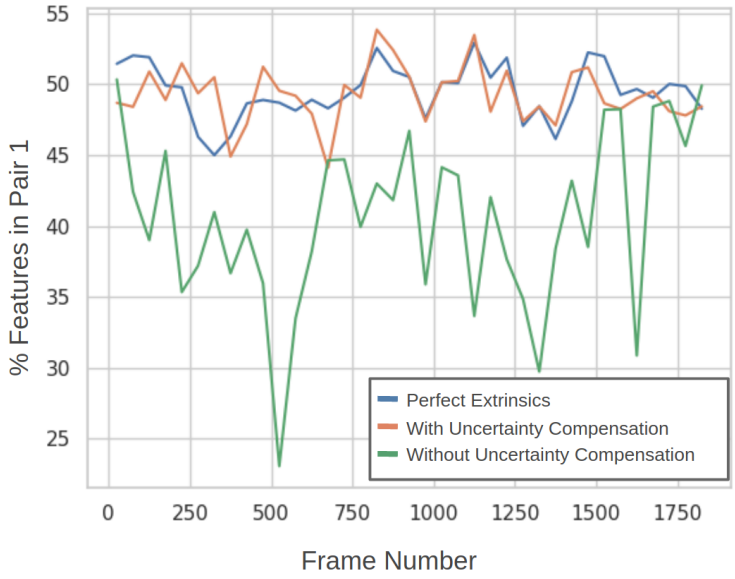

Figure 4.2: This graph displays the percentage of identified inliers in single stereo pair which has a noisy extrinsic calibration. Without compensating for uncertainty, our outlier rejection scheme has a bias towards features observed in cameras with stronger calibration. After compensating for uncertainty we see that the feature distribution more closely matches the original distribution, which are the results with perfect extrinsics.

inliers in the stereo pair decrease significantly. If we use the uncertainty compensation method described in this Section, we can see that the feature distribution returns to a similar level as the original even with the noise added.

4.5.2 Derivation

We choose to represent the uncertainty in the extrinsics by modeling uncertainty in the transformation between the left camera of each stereo pair and the IMU. Using the same convention as [1], we represent each of these estimated transformations $\mathbf{T}_{S_i}^B$ as a member of the special Euclidean group $SE(3)$. We can model each transformation as some "true" transformation $\bar{\mathbf{T}}_{S_i}^B$ perturbed by some noise:

$$\begin{aligned} &\xi_i \in \mathbb{R}^6 \\ &\xi_i \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{S_i}^B). \end{aligned} \tag{4.3}$$

Our proposed method propagates the uncertainty in the transformation through both the camera-to-IMU transformation as well as the reprojection to obtain an uncertainty in the pixel space of the image. With an uncertainty in the pixel space, we can determine inliers by setting a threshold on the Mahalanobis distance between the projected features and their actual observed locations.

We refer readers to [1] for a detailed derivation of uncertainty propagation used in this section. We start with a triangulated 3D point in one of the camera frames of the robot. It is well known that the error in triangulation is quadratic with respect to the depth of the point. We model an initial uncertainty on the triangulated 3D point by propagating the pixel noise in the image using the method described in [22]. We propagate the uncertain point through the uncertain camera-to-IMU transformation:

$$\bar{\mathbf{P}}_B = \bar{\mathbf{T}}_{S_i}^B \bar{\mathbf{P}}_{S_i} \tag{4.4}$$

If $\bar{\mathbf{P}}_B = [\mathbf{h}^\top, \lambda]^\top$ then the 4×9 Jacobian of the homogenous transformed point with respect to the parameters of both the transformation and the original point is:

$$\mathbf{J} = \begin{bmatrix} \lambda \mathbf{I}_3 & -\mathbf{h}^\wedge & \mathbf{R}_{S_i}^B \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \end{bmatrix} \tag{4.5}$$

We obtain a covariance matrix estimating the uncertainty of the point in the body frame using a first order approximation.

$$\mathbf{\Sigma}_{\mathbf{P}_B} = \mathbf{J} \begin{bmatrix} \mathbf{\Sigma}_{S_i}^B & \mathbf{0}_{6\times 3} \\ \mathbf{0}_{3\times 6} & \mathbf{\Sigma}_{\mathbf{P}_{S_i}} \end{bmatrix} \mathbf{J}^\top \tag{4.6}$$

We use the method described in [1] to propagate the uncertainty of the extrinsics and the uncertainty of the point in the body frame through projection into a nonlinear camera model. For

each candidate motion model in RANSAC, we obtain a 3D point in the body frame, $\tilde{\mathbf{P}}_B$, which needs to be projected back into the original image to determine inliers. We model an uncertain 3D point as:

$$\tilde{\mathbf{P}}_B = \bar{\tilde{\mathbf{P}}}_B + \mathbf{D}\zeta \tag{4.7}$$

where $\zeta \in \mathbb{R}^3$ and $\zeta \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\tilde{\mathbf{P}}_B})$ and $\mathbf{D}$ is the 4×3 matrix defined by:

$$\mathbf{D} = \begin{bmatrix} \mathbf{I}_{3\times 3} \\ \mathbf{0}^\top \end{bmatrix}. \tag{4.8}$$

The projection function $\pi : \mathbb{R}^4 \to \mathbb{R}^2$ is a nonlinear function which takes a homogeneous 3D point in the left camera frame and projects it to an image pixel. We define the Jacobian of the projection function with respect to the homogenous points in the camera frame as

$$\mathbf{\Pi} = \frac{\partial \pi}{\partial \mathbf{w}} w = \begin{bmatrix} \frac{f_x}{w_3} & 0 & -\frac{f_x w_1}{{w_3}^2} & 0 \\ 0 & \frac{f_y}{w_3} & -\frac{f_y w_2}{{w_3}^2} & 0 \end{bmatrix} \tag{4.9}$$

Similarly to Eq.4.5 we define $\mathbf{G}$ as the Jacobian of the transformed homogenous point with respect to the parameters of the transformation and original point. In this case the transformation we are considering is from the body frame back to the camera frame.

$$\mathbf{G} = \begin{bmatrix} \lambda \mathbf{I}_3 & -\mathbf{d}^\wedge & \mathbf{R}_B^{S_i} \\ \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} & \mathbf{0}_{1\times 3} \end{bmatrix} \tag{4.10}$$

where $\tilde{\mathbf{P}}'_{S_i} = \mathbf{T}_B^{S_i} \tilde{\mathbf{P}}_B = [\mathbf{d}^\top, \lambda]^\top$. We make the important distinction that $\mathbf{P}_{S_i}$ is the original triangulated point in the camera frame, while $\mathbf{P}'_{S_i}$ is the point after it has been transformed back into the camera frame as part of the RANSAC scheme. From here we define the Jacobian of the entire reprojection function, from the point in the body frame to a pixel in the image as:

$$\mathbf{H} = \mathbf{\Pi}\mathbf{G} \tag{4.11}$$

and the covariance in the image space as:

$$\mathbf{\Upsilon}_{\tilde{\mathbf{p}}_L} = \mathbf{H}\mathbf{\Xi}\mathbf{H}^\top \tag{4.12}$$

$$\mathbf{\Xi} = \begin{bmatrix} \mathbf{\Sigma}_B^{S_i} & \mathbf{0}_{6\times 3} \\ \mathbf{0}_{3\times 6} & \mathbf{\Sigma}_{\tilde{\mathbf{P}}_B} \end{bmatrix} \tag{4.13}$$

It's worth exploring why a bias towards cameras with a more accurate calibration is a bad thing, as it is not immediately obvious. Intuitively features from cameras with more accurate calibrations would provide better constraints on the optimization than features observed from cameras with noisy calibrations. It's tempting to think that the bias might be a benefit to the overall performance of the system. While this intuition may seem correct, it's important to remember that the only goal of outlier rejection is to address the issue of incorrect data association.

So long as tracked features still correspond to the same 3D point, then they should be considered inliers.

In order to ensure that features observed in cameras with noisy calibrations don't degrade the quality of the state estimate, we also account for the extrinsics uncertainty in the back-end optimization. This is done by altering the noise model of the projection factors associated with each feature. Most of the popular projection factors used for indirect visual SLAM systems use a measurement noise model associated with some small pixel error in the image. We build on this by adding an uncertainty propagated from the noisy extrinsics. Specifically, we modify Eq. 4 in [14] such that the residual associated with each projection factor is weighted by $\mathbf{\Sigma}_m$ rather than $\mathbf{\Sigma}_{c_{ij}}$:

$$\mathbf{\Sigma}_m = \mathbf{\Sigma}_{c_{ij}} + \mathbf{\Upsilon}_{\mathbf{p}_L^*} \tag{4.14}$$

where $\mathbf{\Sigma}_{c_{ij}}$ is the noise model of the camera and $\mathbf{\Upsilon}_{\mathbf{p}_L^*}$ is the covariance of the given landmark projected from the current linearization point of the optimization into the left camera. Since we are projecting an estimate of the 3D landmark position we have $\mathbf{\Sigma}_{\mathbf{P}_B^*} = \mathbf{0}$. To our best knowledge, this is the first work which explicitly models extrinsic uncertainty as part of a VIO pipeline.

4.5.3 Quantifying Uncertainty

While there are several works which specifically aim to improve the quality of the calibration [8, 12, 28] some uncertainty will always remain. This remaining uncertainty can be attributed to the noisy sensor data used to perform calibration and the physical deformation of the camera rig that can vary with time and temperature. Knowing that it is impossible to obtain a perfect calibration, we decide to model and account for the uncertainty. Although this paper does not specifically focus on strategies to obtain a measurement of extrinsic uncertainty, it can generally be done by either extracting it from a tool which formulates calibration as the optimization of a nonlinear least squares problem or by estimating it based on the physical parameters of the camera rig.

4.6 Results

The proposed multi-camera VIO pipeline was evaluated using a MAV in a simulated Gazebo environment as well as on real world data collected in the Robotics Institute's Highbay. The simulated environment allowed us to obtain ground truth extrinsics and manually add noise to test our system, while the Highbay data allowed us to verify the robustness of our algorithm on real data. For both types of experiments we evaluate the accuracy of the trajectory against VINS-Fusion running on both stereo pairs individually. VINS-Fusion was run with default parameters and the option to optimize for extrinsics enabled. As a comparison we also compare against a version of our full VIO pipeline which uses a fundamental matrix RANSAC for outlier rejection

Table 4.2: Average Computational Time for Outlier Rejection

	Proposed	Proposed with Fundamental Matrix RANSAC
Sim Easy	0.0138 s	0.0435 s
Sim Difficult	0.0118 s	0.0418 s
Highbay	0.00774 s	0.0418 s

Table 4.3: Average Trajectory Error in Simulated Environments

	Proposed	Proposed without Uncertainty Compensation	Proposed with Fundamental Matrix RANSAC	VINS-Fusion Primary	VINS-Fusion Secondary
Easy	0.179 m	0.144 m	0.167 m	0.131 m	**0.128 m**
Difficult	**0.791 m**	0.954 m	0.810 m	Failed	7.510 m

on each stereo pair individually. For the simulated data the primary stereo pair was facing forwards and the secondary pair was facing backwards. For the Highbay data, the primary stereo pair was facing forwards the secondary pair was facing downwards. We also compare against our proposed algorithm without uncertainty compensation in order to precisely observe the effects of uncertainty compensation. Each reported result is the median over 5 trials. Errors are reported in Tables 4.3 and 4.6. Failure is defined as an error over 10% of the length of the trajectory. To demonstrate the computational benefit of our 1-point RANSAC formulation, a comparison of computational time required to run the proposed outlier rejection scheme and the fundamental matrix RANSAC is shown in Table 4.2.

4.6.1 Simulated Results

The simulated data was recorded on a MAV in an outdoor *Gazebo* environment. Noise was randomly added to the extrinsics of each stereo pair. The results of two simulated flights were recorded. One flight was a relatively easy trajectory with no aggressive motion or sudden scene occlusions. Another simulated flight included aggressive turns and flight very close to obstacles which could partially occlude the fields of view of the cameras on board. Images and IMU measurements from the simulator were taken as inputs to the VIO pipeline. All methods are able to achieve a similarly low ATE on the easy flight. The main benefit of our proposed system is apparent in the difficult dataset. A plot of the trajectories on the easy dataset is shown in Figure 4.3 and of the hard dataset in Figure 4.4.

Table 4.4: Final Trajectory Error in NSH Highbay

	Proposed	Proposed without Uncertainty Compensation	Proposed with Fundamental Matrix RANSAC	VINS-Fusion Primary	VINS-Fusion Secondary
Highbay	**1.694 m**	6.980 m	5.217 m	Failed	Failed

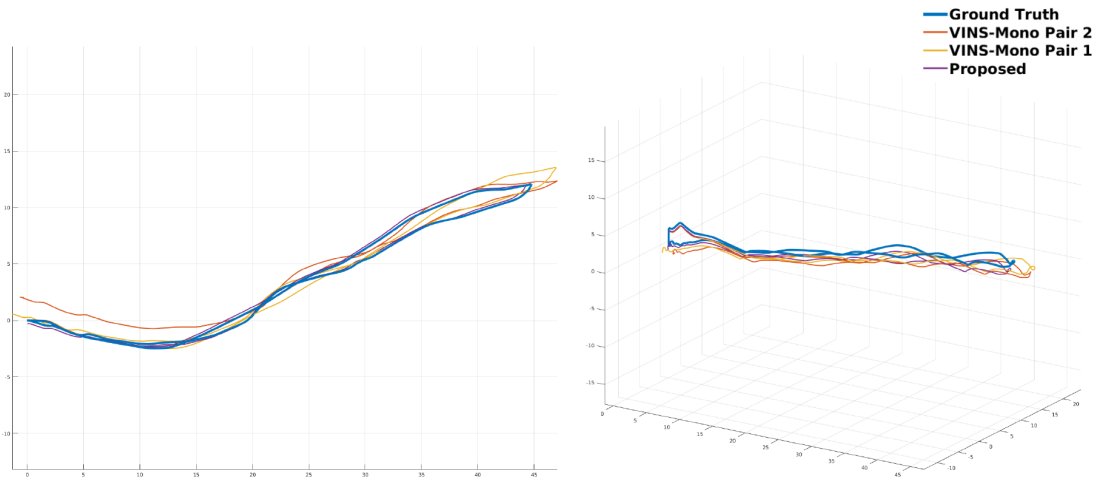

Figure 4.3: The trajectory results on the easy simulated environment. The left image shows an overhead view of the trajectory while the right image shows a side profile. This trajectory was generally slow moving, with no aggressive motion and no sudden occlusions from the camera. Plotted and the results from the proposed method, VINS-Mono running on both pairs individually, and ground truth.

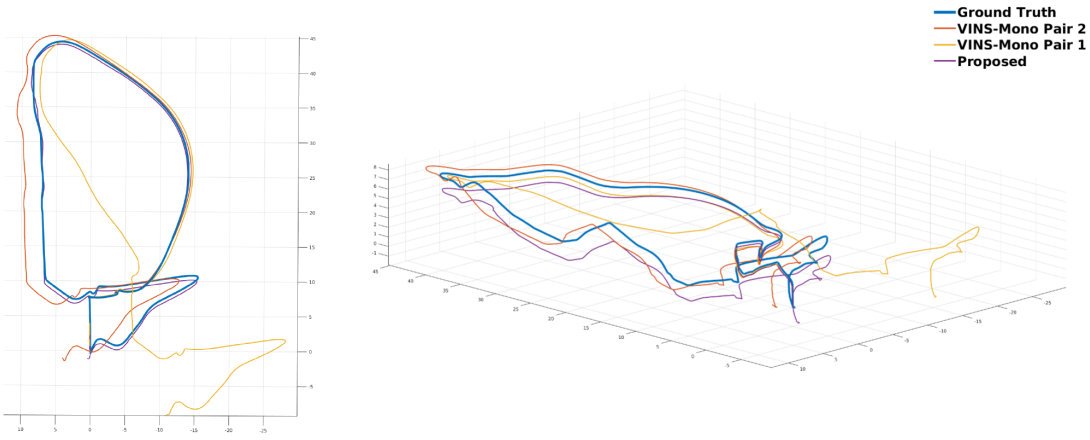

Figure 4.4: The trajectory results on the hard simulated environment. The left image shows an overhead view of the trajectory while the right image shows a side profile. This trajectory was generally fast moving, with aggressive motion and several instances of sudden occlusions of the camera field of view from objects in the scene. Plotted and the results from the proposed method, VINS-Mono running on both pairs individually, and ground truth.

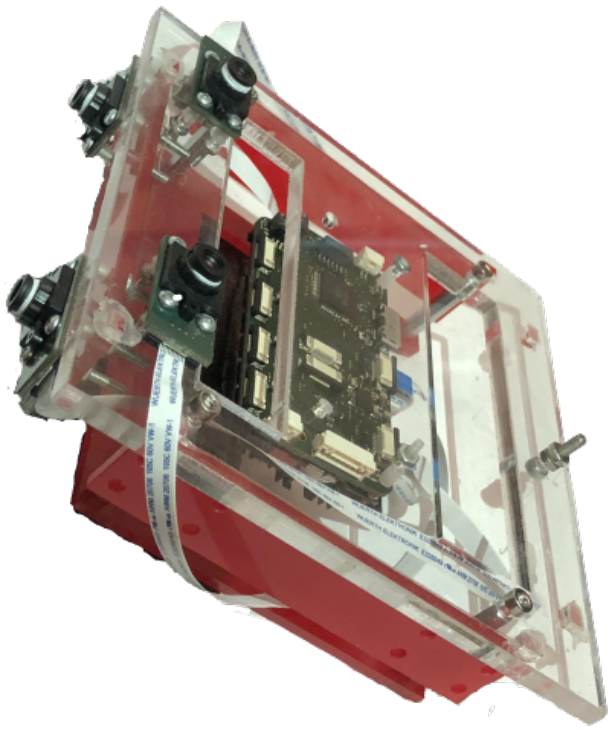

Figure 4.5: The multi-stereo camera rig used to collect experimental results. Images were captured at 25 Hz and inertial data was collected at 200 Hz. All cameras were synchronized using the on-board FPGA.

4.6.2 Highbay Data

Our real world data was collected using a two stereo camera rig with time synchronized images and IMU data, seen in Figure 4.5. Time synchronization was performed using an FPGA, and the camera operated at approximately 25 Hz. Each individual camera had an on-board IMU which operated at 200 Hz. For our experiments we only used the data from a single IMU and discarded information from the other sensors. The multi-camera rig was moved around the Highbay and returned precisely to its original starting position. To test the robustness of our VIO the data was intentionally made to be extremely challenging, with several points of sudden occlusion occurring during the run. We evaluated each algorithm by returning to the same starting location and measuring Final Trajectory Error (FTE), which is the absolute drift of the final position. A visualization of two trajectories is shown in Figure 4.6 for a reference of scale.

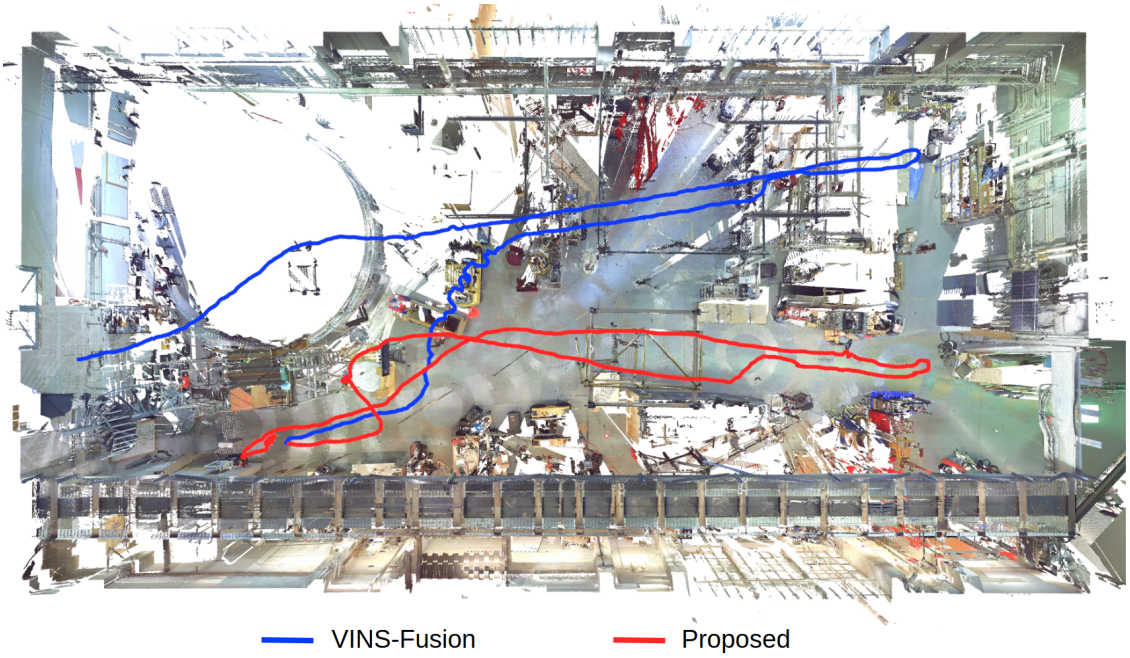

Figure 4.6: Experimental results in the NSH highbay. The trajectory generated from our proposed system is shown in red and the result from VINS-Fusion running on a single stereo camera is shown in blue.

Chapter 5

Conclusion

Contributions In this thesis we presented a method of incorporating information from multiple cameras into a visual inertial odometry framework. The main goal of using multiple cameras is to improve the robustness of VIO and by doing so, expand the number of real world uses cases. We described a framework for using features from multiple cameras which is a simple extension of traditional smoothing based indirect VIO methods. We show that using multiple cameras does improve the robustness of the state estimating by evaluating on both real world and simulated datasets which present difficult to traditional VIO pipelines. Specifically we show how performing efficient joint outlier rejection is an important consideration in multi-camera VIO systems, and propose a novel 1-point RANSAC algorithm specific to multi-stereo configurations. We also explain why the problem of uncertain extrinsic calibrations is not only a problem in VIO in general but especially problematic in multi-camera configurations. We deal with the issue of uncertain extrinsics by proposing, what is to our best knowledge, the first VIO pipeline which explicitly models extrinsic uncertainty in both the outlier rejection and back-end optimization step.

Chapter 3 outlines the basic design of a VIO system. This includes details about both the front-end and back-end. For the front-end we described how features were tracked, the types of features that were tracked, and how we used feature bucketing to reduce computation. Related to the back-end we discussed the measurement model for both IMU data and camera data. We described the preintegrated IMU factor, and why it offers such a significant computational advantage over alternative methods of incorporating inertial information. We discussed two different parameterizations for visual measurements, full 3D and inverse depth, and discussed the benefits and drawbacks of each method. We also described the marginalization strategy we used which is based on information sparsification to maintain a fast real-time system without having to discard useful information.

Chapter 4 outlined how we could extend the traditional VIO system described in Chapter 3 to the mutli-camera case. We first describe a joint RANSAC model which is applicable to multi-stereo configurations. We describe the benefits of performing joint outlier detection rather

than independent outlier rejection, and derive the 1-point RANSAC model. This outlier rejection scheme exploits the constraints provided by a calibrated multi-camera rig, the benefits of having stereo observations of the scene, and the measured rotation from the IMU to create an efficient and effective outlier rejection system. We then outline the need to model uncertainty in the extrinsic calibration of each camera and account for uncertain extrinsics by propagating the uncertainty through each step of the RANSAC process to obtain an uncertainty in the pixel space of the image. We do the same in the back-end and modify the noise model associated with each projection factor to include this newly obtained pixel space uncertainty. The result is a RANSAC system which is more lenient when selecting features from cameras with more uncertain extrinsics but a back-end system which compensates for this by trusting those measurements less. We argue that this is the correct way of dealing with uncertain calibrations, since we will ideally not reject otherwise valid features simply because of a poor calibration, but we will also no let the noise in the measurement deteriorate the result of the optimization.

5.1 Futue Work

There are several interesting areas of potential future work. The first, and most obvious, is a more extensive evaluation of this system on real-world data, and specifically datasets with corresponding ground truth trajectories. This has proven to be a significant challenge since multi-camera and specifically multi-stereo datasets are not widely available to evaluate against.

Another interesting direction would be to explore strategies to limit the number of features to be used by the back-end beyond just outlier rejection. As the number of cameras in a multi-camera system increases so does the number of tracked features. There will inevitably come a point where even with feature bucketing and strict outlier rejection, it won't be computationally possible to use all the features for optimization and still maintain real-time performance. Ideally a strategy could be developed which would only optimize the features which provide strong constraints of the current state, and the number of features used could dynamically scale depending on the amount of remaining compute.

While we've explored the use of multiple cameras in VIO, another emerging area of research is the use of multiple IMUs. Since IMU data typically comes in at a very high frequency (usually in the range of hundreds of Hz), processing data from multiple IMUs is a significant challenge that would need to be addressed before multi-IMU systems can be used in real-world applications. Despite these limitations, there is still interesting work going on in the field of multi-IMU navigation, as each different IMU helps provides a constraint that could help improve the state estimate accuracy.

Finally, an important next direction of work is to create workarounds to address some of the assumptions made in the proposed pipeline. For example, we assume we have time synchronized cameras which simplifies the structure of our factor graph. Implementing a system to estimate a time offset between images and to compensate for this offset would allow the system to be used

on more real world systems which often don't have hardware synchronized cameras.

Bibliography

[1] T. G Barfoot and P. T Furgale. Associating uncertainty with three-dimensional poses for use in estimation problems. *IEEE Trans. on Robotics (TRO)*, 30(3):679–693, 2014. 4.5.2, 4.5.2, 4.5.2

[2] J. Civera, A. J Davison, and M. J. M Montiel. Inverse depth parametrization for monocular slam. *IEEE Trans. on Robotics (TRO)*, 24(5):932–945, 2008. 3.3.1

[3] F. Dellaert and M. Kaess. Factor graphs for robot perception. *Foundations and Trends in Robotics*, 6(1-2):1–139, August 2017. 3.5

[4] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In *Eur. Conf. on Computer Vision (ECCV)*, September 2014. 3.2

[5] J. Engel, J. Stueckler, and D. Cremers. Large-scale direct SLAM with stereo cameras. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, September 2015. 3.2

[6] J. Engel, V. Koltun, and D. Cremers. Direct sparse odometry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(3):611–625, March 2018. 3.2

[7] C. Forster, L. Carlone, F. Dellaert, and D. Scaramuzza. On-manifold preintegration for real-time visual-inertial odometry. *IEEE Trans. on Robotics (TRO)*, 33(1):1–21, 2017. 3.3.2

[8] P. Furgale, T. D. Barfoot, and G. Sibley. Unified temporal and spatial calibration for multi-sensor systems. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1280–1286, Tokyo, Japan, 2013. 4.5.3

[9] M. Muller G, M. J Schuster, P. Lutz, M. Maier, S. Stoneman, T. Tomic, and W. Struzl. Robust visual-inertial state estimation with multiple odometries and efficient mapping on an MAV with ultra-wide FOV stereo vision. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018. 4.2

[10] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, New York, NY, USA, 2 edition, 2003. ISBN 0521540518. 2.4.1, 2.4.2, 3.2

[11] L. Heng, G. H Lee, F. Fraundorfer, and M. Pollefeys. Real-time photo-realistic 3D mapping for micro aerial vehicles. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4012–4019, San Francisco, CA, USA, September 2011. 4.2

[12] L. Heng, G. H Lee, and M. Pollefeys. Self-calibration and visual SLAM with a multi-camera system on a micro aerial vehicle. *Autonomous Robots (AURO)*, 39:259–277, 2015.

4.2, 4.5.3

[13] S. Houben, J. Quenzel, N. Krombach, and S. Behnke. Efficient multi-camera visual-interial SLAM for micro aerial vehicles. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1616–1622, Daejeon, Korea, October 2016. 4.2

[14] J. Hsiung, M. Hsiao, E. Westman, R. Valencia, and M. Kaess. Information sparsification in visual-inertial odometry. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1146–1153, Madrid, Spain, 2018. 3.5, 4.5.2

[15] M. Kaess, A. Ranganathan, and F. Dellaert. iSAM: Fast incremental smoothing and mapping with efficient data association. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1670–1677, Rome, Italy, April 2007. 2.1.2

[16] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 3281–3288, Shanghai, China, May 2011. 2.1.2

[17] G. Klein and D. Murray. Parallel tracking and mapping on a camera phone. *International Symposium on Mixed and Augmented Reality (ISMAR)*, pages 83–86, 2009. 3.2, 4.2

[18] G. H. Lee, M. Pollefeyes, and F. Fraundorfer. Relative pose estimation for a multi-camera system with known vertical direction. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2014. 4.2

[19] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale. Keyframe-based visual–inertial odometry using nonlinear optimization. *Intl. J. of Robotics Research (IJRR)*, 34(3):314–334, 2015. 3.5

[20] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, pages 674–679, Vancouver, Canada, 1981. 3.4.2

[21] T. Lupton and S. Sukkarieh. Efficient integration of inertial observations into visual SLAM without initialization. In *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 1547–1552, St. Louis, USA, 2009. 3.3.2

[22] L. Matthies and S. A. Shafer. Error modeling in stereo navigation. *IEEE Journal of Robotics and Automation*, 3:239 – 248, June 1987. 4.5.2

[23] A. I Mourikis and S. I Roumeliotis. A multi-state Kalman filter for vision-aided inertial navigation. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 10–14, April 2007. ISBN 1424406021. 3.2

[24] R. Mur-Artal, J.M.M. Montiel, and J.D. Tardos. ORB-SLAM: a versatile and accurate monocular SLAM system. *IEEE Trans. on Robotics (TRO)*, 31(5):1147–1163, 2015. 3.2, 4.2

[25] T. Oskiper, Z. Zhu, S. Samarasekera, and R. Kumar. Visual odometry system using multiple stereo cameras and inertial measurement unit. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2007. 4.2

[26] R. Pless. Using many cameras as one. In *IEEE Computing Society Conf. on Computer*

Vision and Pattern Recognition (CVPR), June 2003. 4.2

[27] T. Qin, P. Li, and S. Shen. VINS-Mono: A robust and versatile monocular visual-inertial state estimator. *IEEE Trans. on Robotics (TRO)*, 34(4):1004–1020, 2018. 3.2, 3.3.1, 3.5

[28] J. Rehder, J. Nikolic, T. Schneider, T. Hinzmann, and R. Siegwart. Extending kalibr: Calibrating the extrinsics of multiple IMUs and of individual axes. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 4304–4311, Stockholm, Sweden, 2013. 3.3.2, 4.5.3

[29] E. Rosten and T. Drummond. Machine learning for high-speed corner detection. In *Eur. Conf. on Computer Vision (ECCV)*, pages 430–443, May 2006. 3.2

[30] J. Shi and C. Tomasi. Good features to track. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994. 3.4.1, 3.4.2

[31] K. Sun, K. Mohta, B. Pfrommer, M. Watterson, S. Liu, Y. Mulgaonkar, C. J. Taylor, and V. Kumar. Robust stereo visual inertial odometry for fast autonomous flight. *IEEE Robotics and Automation Letters (RA-L)*, 3(2):965–972, 2018. 3.2, 3.4.2

[32] M. J. Tribou, A. Harmat, D. W. L. Wang, I. Sharf, and S. L Waslander. Multi-camera parallel tracking and mapping with non-overlapping fields of view. *Intl. J. of Robotics Research (IJRR)*, 34(12):1480–1500, 2015. 4.2

[33] C. Troiani, A. Martinelli, C. Laugier, and D. Scaramuzza. 2-point-based outlier rejection for camera-IMU systems with applications to micro aerial vehicles. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Hong Kong, China, June 2014. 3.2, 4.4.3

[34] V. Usenko, J. Engel, J. Stückler, and D. Cremers. Direct visual-inertial odometry with stereo cameras. In *IEEE Intl. Conf. on Robotics and Automation (ICRA)*, pages 1885–1892, May 2016. 3.2