Monocular Visual-Inertial SLAM and Self Calibration for Long Term Autonomy

by

Nima Keivan

B.S., University of Queensland, 2009

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Computer Science

2017

This thesis entitled:
Monocular Visual-Inertial SLAM and Self Calibration for Long Term Autonomy
written by Nima Keivan
has been approved for the Department of Computer Science

_____

Professor Gabe Sibley

_____

Professor Christoffer Heckman

Date _____

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Keivan, Nima (Ph. D., Computer Science)

Monocular Visual-Inertial SLAM and Self Calibration for Long Term Autonomy

Thesis directed by Professor Gabe Sibley

This thesis is concerned with real-time monocular visual-inertial simultaneous localization and mapping (VI-SLAM) with application to Long Term Autonomy. Given a sensor rig capable of making visual and inertial measurements, accurate real-time estimation of its position and orientation (pose) as well as the creation of a scale-correct map of the surrounding environment is desired. This estimation task requires accurate calibration of both intrinsic and extrinsic properties of the visual and inertial sensors. As such, the continuous estimation of these calibration parameters is also desired. Three novel methods are presented, covering real-time VI-SLAM, self calibration, and change detection. Together they form a basis for long term localization and mapping robust to changes in calibration.

The VI-slam methodology is motivated by the requirement to produce a scale-correct visual map, in an optimization framework that is able to incorporate relocalization and loop closure constraints. Special attention is paid to achieve robustness to many real world difficulties, including degenerate motions and unobservablity. A variety of helpful techniques are used, including: a relative manifold representation, a minimal-state inverse depth parameterization, and robust non-metric initialization and tracking. Also presented is an extensible framework for real-time self-calibration of cameras in the SLAM setting. The system is demonstrated to calibrate both pinhole and fish-eye camera models from unknown initial parameters while seamlessly solving the maximum likelihood online SLAM problem in real-time. Self-calibration is performed by tracking image features, and requires no predetermined calibration target. By automatically identifying and using only those portions of the sequence that contain useful information for the purpose of calibration the system achieves accurate results incrementally and in constant-time vs. the number of images. Finally, a framework for online SLAM and self-calibration is presented which can detect and

handle significant change in the calibration parameters. A novel technique is presented to detect the probability that a significant change is present in the calibration parameters. The system is then able to re-calibrate. Maximum likelihood trajectory and map estimates are computed using an asynchronous and adaptive optimization. The system requires no prior information and is able to initialize without any special motions or routines, or in the case where observability over calibration parameters is delayed. Both self-calibration frameworks are extensible and able to cover any calibration parameters which can be estimated from the measurements. The contributions are individually evaluated in a number of experiments with real data. Specific focus is placed on accuracy and real-time performance.

Acknowledgements

I would like to express my thanks and gratitude to my advisor Gabe Sibley for his continuous support, guidance, and patience. His drive to pursue rigor and his dedication to implementing and proving ideas in action have been inspirational to me. I would also like to thank Sina Aghli, Juan Falquez, Dorian Galvez-Lopez, Mahsa Ghafarianzadeh, Christoffer Heckman, Steven Lovegrove, Lu Ma, Vincent Mamo, and Alonzo Patron who were my fellow students and researchers at the Autonomous Robotics and Perception Group with whom I had no end of fruitful discussions, collaboration, and exchange of ideas. I was fortunate to have the opportunity to work with and learn from them during my PhD. Finally I would like to thank Erin Tunbridge for her unwavering support during setbacks and successes alike, and my family for their continual encouragement throughout all my endeavors.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

The work presented in this thesis is primarily motivated by Long Term Autonomy: the development of autonomous robotic systems capable of operating for weeks, months, and years without requiring human intervention. The advantage of robots capable of Long Term Autonomy are twofold. First, they are economical. To design systems capable of Long Term Autonomy, robustness must be considered as a primary goal. These systems must deal with all extraneous elements in their operating environments, as well as responding and adapting to any deliberate or accidental changes to their software or hardware components. As such, they are capable of operating without human assistance for long periods which makes them economically feasible for repetitive and simple tasks. Second, they can be sent into environments where human supervision is not readily feasible. Examples of this are long voyages in the ocean or in space as well as hazardous environments such as underground mines and disaster zones. Needless to say, an ambitious goal such as Long Term Autonomy permeates the design of all aspects of an autonomous systems: hardware and software.

Much like for any autonomous robot, Long Term Autonomy demands that the robot be able to both sense its environment and act appropriately to achieve its objectives. We are primarily concerned with the former requirement: the perception of the robot's environment. This requirement is further refined into a perception of one's location within an environment, creating representations of this environment, and finally the semantics of objects and components within this environment which may be useful in fulfilling the objectives of the robot. Our focus is then further narrowed

onto the first two components: knowing one's location within the environment which is referred to as *localization*, and constructing a representation of the environment which is conducive to future attempts at localization, which is referred to as the *map*.

The two previously mentioned components of robotic perception must ideally be performed simultaneously when faced with unknown or changing environments. Simultaneous Localization and Mapping (SLAM) has therefore been a mainstay of robotics research for decades. SLAM is agnostic to the sensing modality used to perceive the robots environment. Early SLAM research used sonar sensors which returned range measurements by bouncing ultrasonic sound waves off the environment. Laser rangefinders (Lidar) were also used to perform SLAM, using their better range and bearing resolution to increase the localization accuracy and create more accurate maps. Finally, visible light and infrared cameras have been used to perform SLAM since the computing power necessary to perform the heavy calculations have become available.

We will focus our attention once again on SLAM performed with a specific sensing modality: cameras. This type of SLAM is referred to as Visual SLAM. Since our intended application is robotics, out focus will also be mainly narrowed to algorithms and approaches which are real-time and therefore can be executed online to aid robotic perception.

1.1 Structure

In the rest of this chapter, an introduction will be given to the concepts utilized in this thesis. This introduction will cover various falvors of visual and visual-inertial SLAM, as well as key concepts such as marginalization and conditioning. The applications of the presented work will also be described. In Chapter 2, visual-inertial SLAM is explored. A novel method is introduced which eschews marginalization in favor of conditioning and enables accuarate localization and mapping. Chapter 3 introduces a framework for self-calibration which allows lifelong calibration of parameters in constant computational complexity. This framework is then applied to the task of calibrating camera intrinsic parameters. In Chapter 4, the previously mentioned self-calibration framework is extended to handle the case where calibration parameters change. The extension is

able to detect the onset of change, and handle the subsequent re-calibration of the parameters. Finally in Chapter 5, the outcomes and contributions of the work as well as potential avenues for future research are discussed.

1.2 Visual SLAM

In Visual SLAM, visible or infrared light cameras are used to take measurements of and perceive the environment around the robot for the purposes of localization and mapping. To accomplish this, a measure of *correspondence* between measurements taken at different locations and different times is needed. Since cameras perceive the environment through captured images, the correspondence required is that between images taken at different times and locations. If correspondence is established between images taken at consecutive times by the camera, information about its continuous motion can be obtained. This is referred to as visual odometry. Similarly, if correspondence is established against images which the camera took at different times and places in the past, we can obtain information about the location of the robot relative to those prior measurements and by association, the map. Furthermore, of is interest is *metric* SLAM, where the location of the robot and also the map which is created for the environment are geometrically and metrical consistent with the real world. For this reason, focus will be placed on correspondence between images which is in some way related to the metric relationship between the cameras at the time of capturing the images, and the 3D scene.

Visual SLAM can be divided into two broad categories based on the type of correspondence that is establish between images: Dense and Sparse.

1.2.1 Dense Visual SLAM

The defining characteristic of dense methods is that correspondence is established at the level of every pixel of an image captured by the camera. Consider the visible or infrared light intensity which reaches the 2D sensing array of the camera as a continuous function $\tilde{\mathcal{I}}(\tilde{u}, \tilde{v})$ where $\tilde{u}$ and $\tilde{v}$ are continuous coordinates which parameterize the surface of the 2D sensing array. The resulting

image returned by the camera can be written as $\mathcal{I}(u, v)$ where u and v are now discrete variables representing the two coordinates of the pixels on the array. Given two such images $\mathcal{I}_1$ and $\mathcal{I}_2$ a relationship between the two can then be established

$$\mathcal{I}_1(\mathbf{p}) = \mathcal{F}(\mathcal{I}_2(\mathcal{W}(\mathbf{q}, \mathbf{x}))) \tag{1.1}$$

where $\mathbf{p}$ is the 2D image coordinates of a pixel, $\mathcal{F}()$ is the correspondence function, $\mathbf{x}$ is a parameter vector describing the metric relationship between the two images and the 3D scene, and $\mathcal{W}()$ is a warping function which given the metric parameters $\mathbf{x}$ will warp the image coordinates of $\mathcal{I}_2$ to their corresponding coordinates of $\mathcal{I}_1$. Note that the vector $\mathbf{x}$ typically consists of the parameters representing the relative translation and rotation between the two cameras which captured $\mathcal{I}_1$ and $\mathcal{I}_2$, and parameters describing the 3D scene which they observed. This vector may also contain other extrinsic and intrinsic parameters of the cameras or their metric relationship. Examples of such parameters include focal length and lens distortion.

Given the relationship defined in Equation 1.1, any of the parameters which define the correspondence can be set as unknowns and solved for in an optimization framework. A typical example is setting the metric parameter vector $\mathbf{x}$ as unknown and solving for it using multiple correspondences between pixels in the two images.

The correspondence function $\mathcal{F}()$ defines how the values of the pixels in the two images are related to one another. The simplest form of correspondence is photometric [8]. In photometric correspondence, the expectation is that two corresponding pixel values would have similar intensities as measured by the camera. This expectation can be written in the form of a residual function:

$$r_{\mathbf{p}} = \mathcal{I}_1(\mathbf{p}) - \mathcal{F}(\mathcal{I}_2(\mathcal{W}(\mathbf{q}, \mathbf{x}))) \tag{1.2}$$

As an example, if the metric parameter vector $\mathbf{x}$ is unknown, an optimization can be formed around minimizing the residual $r_{\mathbf{p}}$:

$$\hat{\mathbf{x}} = \arg\min_{\mathbf{x}} \left(r_{\mathbf{p}} \left(\mathbf{x} \right) \right) \tag{1.3}$$

Depending on the dimensionality of the unknown vector $\mathbf{x}$ and the formulation of the residual, it may not be sufficient to constraint the values of the unknown parameters. In this case multiple residuals are used to constrain and obtain accurate estimate for the parameters. In a typical dense optimization, many hundreds of thousands of pixel residuals in the optimization. The metric parameter vector $\mathbf{x}$ in Equation 1.1 must also contain information about the 3D scene being observed by the camera in order to be able to correctly warp the coordinates of a pixel [30]. This is typically represented as the distance from the scene point that was captured by a pixel of the camera to the camera, referred to as the *depth*. In the case of dense SLAM, since there are hundreds and thousands of pixels involved in the optimization, the depth values are often estimated separately. Common sources of depth values are depth cameras such as the Kinect, as well stereo depth estimation using a pair of synchronized cameras.

Some dense SLAM methods indeed estimate pixel depth values as well [80][87], however due to the complexity of the required optimization, these methods generally split it into two steps: a reconstruction step where depth values for the most recent image are estimated, and a localization step where the camera's position is refined based on current and past images and depth values.

The correspondence function $\mathcal{F}()$ does not necessarily have to be based on photometric correspondence. As an example, Mutual Information based approaches such as [110] establish correspondence by maximizing the mutual information between two images. Correspondence between the two images is established based on a joint intensity histogram which is computed by warping one image into the other. While these images require dense depth information to be able to accomplish the warping, they are exceptionally robust to changes in illumination and lighting, from which the photometric correspondence methods suffer heavily. However, these methods are computationally expensive and are generally difficult to optimize as they reduce the entire image alignment problem to a single residual.

More recently, binary descriptors have been shown to be applicable to dense methods in real-time [4]. Rather than operating on the photometric error between a pixel and its reprojection, a small support region around the pixel is converted to a binary descriptor. This descriptor is then used in the residual in place of the photometric error. Implications of this are that dense methods may become much more resilient to lighting and material based effects, whilst avoiding the high computational cost and poor optimization form of information theoretical alignment methods.

1.2.2 Semi Dense Methods

While the methods described in this section are applied to every pixel of the image, a subset of dense methods instead opt to assign a weight to each pixel residual. The residual described in 1.2 then becomes:

$$r_{\mathbf{p}} = w_{\mathbf{p}} \left[\mathcal{I}_1 \left(\mathbf{p} \right) - \mathcal{F} \left(\mathcal{I}_2 \left(\mathcal{W} \left(\mathbf{q}, \mathbf{x} \right) \right) \right) \right] \tag{1.4}$$

where $w_{\mathbf{p}}$ is the weight assigned to the residual formed by the reprojection of pixel $\mathbf{p}$. The weight is a representation of how important the residual should be to the optimization. The assigned weight can also be binary, depending on some criteria that may or may not be satisfied for the pixel $\mathbf{p}$. In this case, only a subset of pixels in the image will be used to optimize the unknown parameters. A common metric to decide whether or not a pixel is include in the optimization is the magnitude of the image gradient. Intuitively, such a metric will ignore pixels where the magnitude of the image gradient is below a certain value, thereby focusing the optimization on regions with discriminative potential. Due to the reduced size of the optimization, some semi-dense methods [17][16] also estimate the depth of pixels included in the optimization.

1.2.3 Sparse Visual SLAM

Conversely to dense methods, sparse visual SLAM methods create an abstraction over the image by extracting a set of interest points referred to as *features*. The primary characteristic for

these features is that they must be repeatable: in another image of the same scene, given the same 3D location for which the feature was originally extracted, it must be possible to both recognize that 3D location as a feature in the new image, and associate it to the feature in the original image. This is generally accomplished in two steps. First, the image is processed by a keypoint detector which is tasked with finding points of interest. These keypoints are generally located in areas of the image with enough detail to constrain their 2D positions. An example is a sharp corner in the image. There are a multifarious range of keypoint detectors [29][101][96][62], however they share the goal of being invariant to the same 3D point from different viewing locations and angles.

The second step is to obtain a *descriptor* for the keypoint. The descriptor serves as a signature for that particular keypoint. In another image of the same 3D point, once the keypoint detector has marked a location as being a point of interest, a descriptor can be extracted for the keypoint and compared with the original descriptor. This can be used as a means to establish correspondence between two keypoints in different images. Much like keypoint detectors, there is a broad range of feature descriptors, some examples of which are presented in [62][10][11][97]. The goal of the descriptor is to produce the same response when presented with two keypoints which represent the same 3D scene point in two different images. This invariance to the viewpoint from which the image was captured is what allows sparse keypoints and descriptors to form correspondences across multiple images. Descriptors vary significantly among their ranks in terms of invariance and performance. Some descriptors are invariant to size and rotation, meaning that the same response would be generated for keypoints when viewed from different distances and camera orientations. Some descriptors produce descriptors which are in a binary format, vs. real-valued descriptors produced by others.

By virtue of the abstraction generated over the image using sparse features and their descriptors, the correspondence residual which is minimized in sparse slam is simplified compared to the dense correspondence residual in Equation 1.2:

$$\mathbf{r_p} = \mathbf{p} - \mathcal{W}(\mathbf{q}, \mathbf{x}) \tag{1.5}$$

where $\mathbf{r_p} \in \mathbb{R}^2$ is now a 2 dimensional error vector computed from the difference between the 2d location of a feature in image 2, $\mathbf{q}$, and the warped 2d location of the *corresponding* feature in image 1, $\mathbf{p}$. As before, the warping function $\mathcal{W}()$ encodes the metric information of the 3D scene as well as the relative position and rotation between the cameras which captured the two images. As evident in this formulation, image intensities no longer factor in the image residual. This is the advantage of the abstraction which features generate over the image. This absence of image intensities in the residual ostensibly gives sparse visual SLAM methods improved invariance to illumination changes, but this is only true insofar as the feature descriptors themselves are invariant to changes in illumination. Otherwise, correct correspondence cannot be achieved across images. In practice, the use of illumination invariant descriptors does afford an improved invariance to illumination changes to sparse vs. dense methods.

Contrary to dense methods, sparse methods typically need far fewer residuals in order to solve for the unknown metric parameters of the scene and cameras. Features extracted for each image are typically within the range of $100 - 1000$. This affords sparse methods the option of simultaneously estimating the camera pose parameters as well as the depth of the scene, which is now only required at feature locations as opposed to every pixel. To this end, correspondences can be established for features across multiple images. The camera pose estimates for each image as well as the depth estimates for each feature are then solved simultaneously via *bundle adjustment* [119][3]. To solve for these estimates, the correspondence residual outlined in Equation 1.5 is formulated across sparse features observed from multiple images and the entire system of equations solved at once.

1.2.3.1 Direct Feature Tracking

In the previous sections, the sparse visual slam problem has been framed as requiring discrete feature matches across multiple frames using a variant of descriptor. Feature tracking methods

however are not limited to this approach. Rather than extract descriptors for every keypoint extracted at every frame for using in frame-to-frame matching, Direct methods [22] track features by directly matching appearance from frame-to-frame, without extracting new keypoints. The first step is to initialize keypoints in areas of the image where the number of tracked features may be less than what is desired. In the method presented in [22], these new keypoints are then projected into subsequent frames, given the camera motion. An optimization is then performed over the surrounding support region of the keypoint, parameterized by the depth of the feature. In this way, each feature is tracked individually based on the appearance of its projection in subsequent frames.

Alternatively, features can be tracked without any geometric projection and purely in the 2D image plane. The well known KLT tracking method [63][118] is an example of such an approach. A support region around an original keypoint extracted in an image is tracked in subsequent images by directly minimizing its photometric error, parameterized by a 2D homography applied to the original support region. This tracking method obviates the need to extract keypoints in subsequent images. However, due to the use of a photometric error term, this method of tracking will be more susceptible to illumination and material based effects and is most effective when used for short duration frame-to-frame tracking.

Regardless of the method used to track features, sparse methods eventually minimize the 2D reprojection error of tracked features in order to estimate scene and camera parameters.

1.2.4 Filtering

Figure 1.1 introduces the graphical model representation of the SLAM problem which will be used from here on out. The figure shows a series of camera pose parameters, corresponding to the 6DOF pose of the camera where a particular image was taken. Furthermore, the figure shows a number of landmarks, which are points of interest in the scene observed in one or more images. The edges connecting these parameters represent constraints formed using the correspondences established between the images. Note that while this graphical model is commonly used to describe sparse SLAM formulations, it applies equally to the dense SLAM problem. The difference is in

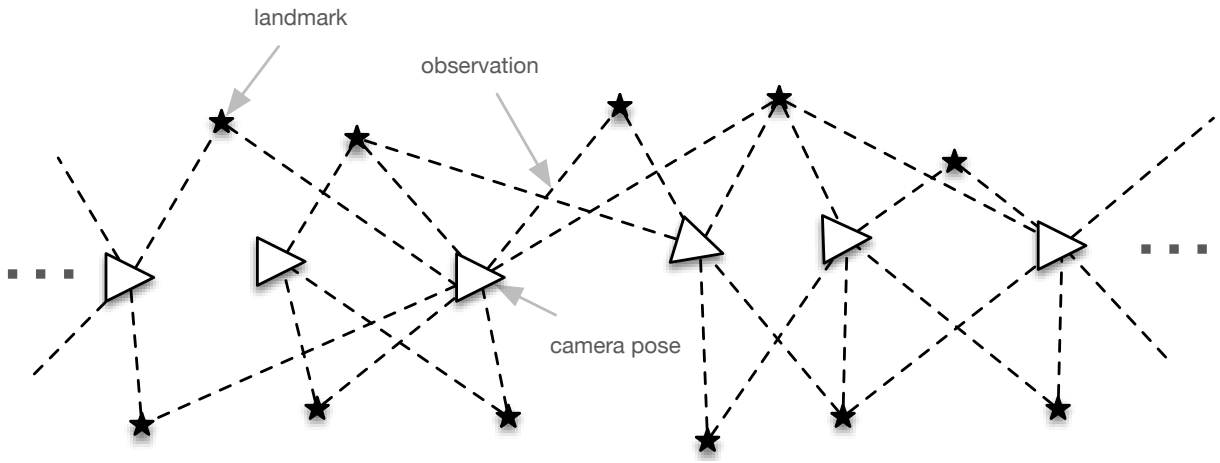


Figure 1.1: Graphical model representing the batch SLAM problem. The camera poses refer to 6DOF parameters added to the optimization at every instance the camera captures an image. The landmarks refer to the parameters which define the position of the landmarks. This can either be a 3DOF vector or an inverse depth represntation [75]. The edges connecting the elements of the graph are constraints. In this case, the constraints are formed by correspondences found in images taken by the camera. The optimal value of the parameters for the landmarks and poses can then be computed by jointly minimizing the error of all constraints. As apparent in the graphical model, the influence of landmarks and poses on other landmarks and poses is bidirectional and extends the length of the graph. This is what renders the batch visual SLAM problem difficult to factorize and solve efficiently.

the dimensionality of the *observation* edge. For sparse SLAM formulations, this edge is a 2D measurement of the image-plan postion of a feature, whereas for dense SLAM formulations, it will be a form of visual similarity metric such as photometric error.

It is evident that while the camera pose parameters are each estimated independently, they are linked together via observations of co-visible landmarks. This co-visibility chain extends beyond the portion of the graph shown in the figure all the way to the point where information collection began, and on the other extreme it extends to the current time. In this form, every parameter's estimate needs to be jointly estimated with *all* other parameters, as there a two-way dependency that spans the entire graph.

Solving for all parameters in this way is usually referred to as the *batch* solution. This involves setting an optimization comprising of all observations and solving for the parameters from scratch every time the graph is modified. For real-time and robotic systems where the current pose of the

camera is desired, this is clearly intractable as the complexity of the problem grows as time goes on. These systems require solutions with *constant* $O(1)$ complexity.

Given that the batch solution is intractable, many different solutions have been proposed in order to solve the problem in a more computationally efficient way for real-time applications. In this section some of these methods will be reviewed as they serve as a good introduction to the problems faced specifically for Visual-Inertial SLAM.

1.2.5 Incremental Factorization

The visual SLAM problem is nonlinear. To optimize the pose parameters of the camera, the projection equations which are used to form the residuals are linearized around the current estimates. When there are no estimates, the problem must therefore be initialized to provide an adequate linearization point. This linearization allows the optimization to approximate the nonlinear form of the problem using a Taylor series expansion up to the second order. This series of equations can then be solved using a 2nd order method such as Gauss-Newton. The formation of the linearized forms of the equations, as well as forming and factorizing the resulting hessian matrix in order to solve for the iteration update is computationally expensive and in the worst case is $O(n^3)$ with respect to the number of parameters in the optimization. This process is typically repeated from scratch after every iteration, as the linearization point changes as the parameters are updated.

Incremental methods such as those presented in [40] and [41] reduce the computational overhead associated with each iteration by storing the result of the linearization in a form which allows updates to the graph to be incorporated into the solution efficiently. This is very effective in situations where a SLAM system is operating in *odometry* mode. In this mode, while simultaneous localization and mapping is indeed performed, no attempt is made to detect whether the camera is observing a scene it has previously mapped. This type of event is referred to as a *loop closure*, as it typically occurs after the camera travels in a loop and returns to a previously observed point. Figure 1.2 shows some of the situations in which loop closures can occur. In the case of a loop

closure, the complexity of incremental approaches is no longer constant, or $O(1)$.

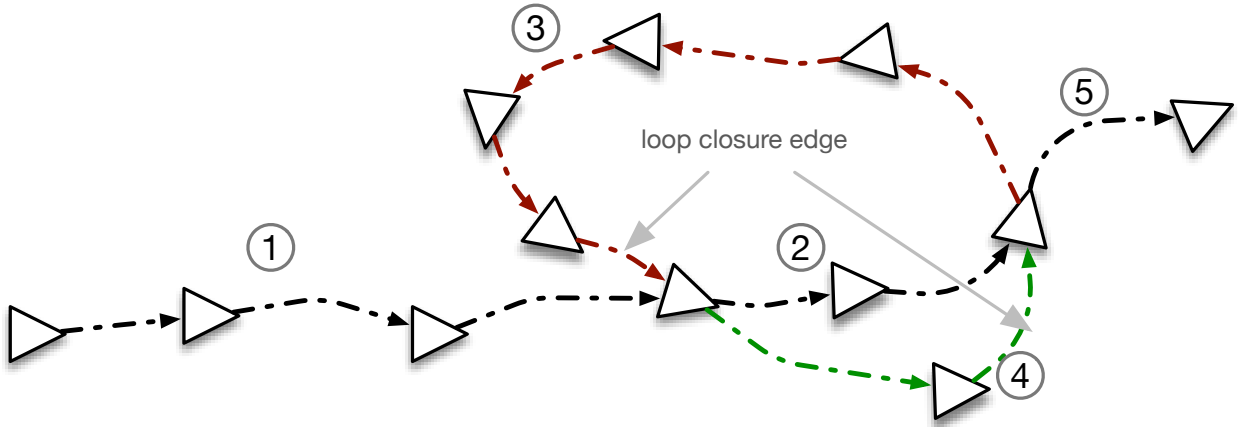


Figure 1.2: Example graphical model when loop closures are detected. Here, the landmarks of Figure 2.3 have been omitted for brevity. The co-visible connections between two edges are shown as curves instaed. In sections 1 and 2, the system is odometry mode and new pose parameters are added to the optimization. During section 3, a loop closure occurs and landmarks previously seen in section 2 in live images are able to be located. This allows us to establish our position relative to the landmarks of section 2 rather than initializing new ones. In section 4 odometry mode is once again active, adding new pose parameters before closing a loop against section 2 once more. Finally in section 5 odometry mode is active. Similar to the example shown in this Figure, loop closures are far from rare when SLAM is deployed in practice and are a critical tool in creating consistent and useful maps for localization.

1.2.6 Marginalization

Figure 1.3: Once a landmark is marginalized, it induces new edges into the graphical model. These edges represent the influence of the landmark parameters on the remaining parameters of the optimization. As evident in the figure, the induced edges connect ever pair of parameters which took part in a constraint with the original landmark parameters.

Another method used to tame the ever-growing size of the batch SLAM problem is marginalization. In this technique, parameters are removed by marginalizing them out of the optimization, turning their influence on other parameters into constraints between them. Figures 1.3 and 1.4

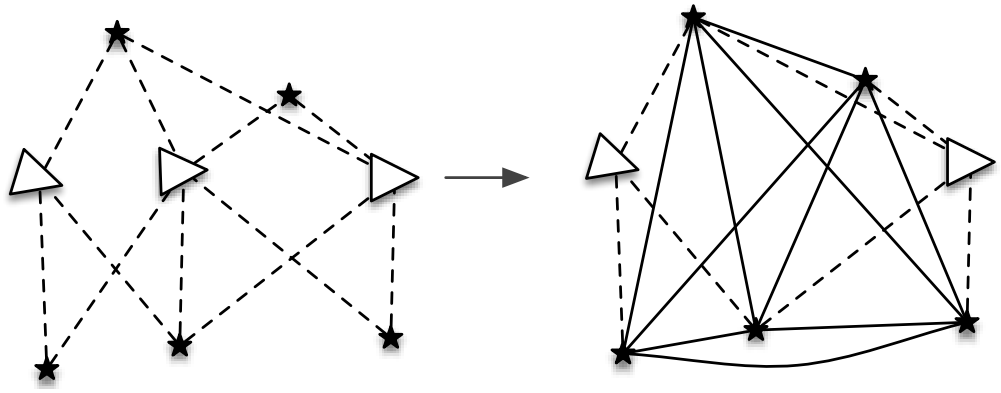


Figure 1.4: Once a pose is marginalized, it induces new edges into the graphical model. These edges represent the influence of the pose parameters on the remaining parameters of the optimization. As evident in the figure, the induced edges connect ever pair of parameters which took part in a constraint with the original pose parameters.

illustrate the effects of marginalizing landmark and pose parameters on the graphical model respectively. When a landmark is marginalized, a new constraint, represented as an edge in the graphical model, is formed between each pair of poses which were originally part of a constraint with the landmark. Similarly, when a pose is marginalized as in Figure 1.4, a new constraint is formed between each par of landmarks which were originally took part in a constraint with the pose.

At first glance, marginalization is a powerful method to reduce the complexity of the problem by removing parameters, without paying the penalty of losing the information that the parameters and the constraints they took part in provided to the optimization. In the case of an optimization consisting of only constraints, this is indeed the case. In this case, marginalization of parameters is performed using the Schur Complement of the matrix of coefficients for the system of equations [24]. However, the constraints which are formed between the parameters of the SLAM problem are nonlinear. Correspondingly, the optimization that must be solved is also nonlinear, and is generally solved by iteratively linearizing the constraints and solving for the next step the optimization must take in order to minimize the residuals. In this case, marginalized parameters will induce the linearized forms of their constraints on the remaining parameters of the problem. There are two issues with this approach. If the parameter being marginalized was not optimally estimate,

the induced constraints on the rest of the problem will be based on this sub-optimal estimate. Second and perhaps more importantly, if the parameters on which the constraints were induced are themselves re-estimated, the induced constraints may introduce *inconsistency* into the optimization . For these reasons, it is preferable to marginalize parameters for which the estimates have largely settled. Estimation of parameters over the long term (such as the case of self-calibration in Chapter 4) is one example where marginalization should be approached with careful consideration.

An example of the application of marginalization to the SLAM problem is the Sliding Window Filter [107]. In this method, a temporal window of camera poses and landmarks are maintained in the optimization. As new measurements are taken, the parameters held in the filter are re-estimated. Parameters at the other end of the filter are marginalized out into a prior which will affect the remaining parameters. In this way, parameters are given many iterations and measurements to settle on estimates before being linearized into a prior.

1.2.7 Conditioning

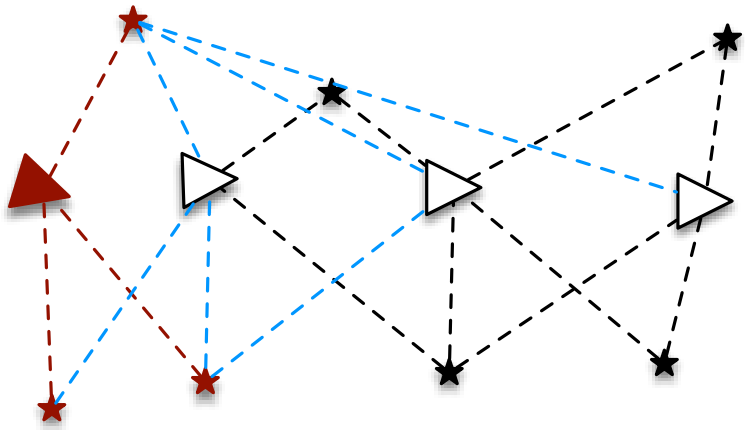


Figure 1.5: Graphical model representing conditioning for the visual slam problem. Poses and landmarks in dark red have been set to constant. The rest of the graphical model has been *conditioned* on these constant parameters through the *conditioning edges* which are in blue. These edges represent constraints between the variable and constant parameters and constrain the graph based on the solutions of the constant parameters.

Conditioning is yet another method to combat the growing size of the full batch SLAM problem. Tt involves fixing certain parameters as constant after their estimates have settled or

after a certain amount of time has passed. The method in [105] is an example of an optimization which uses conditioning to constrain the number of parameters which are solved for. Note that this method is *adaptive*. A parameter is not permanently set constant. The constancy assumption is purely based on the assumption that the parameter's estimate has converged and there is no further need to optimize it. However, this assumption may be challenged in the future. In that case, continuing to use the constant estimate for the parameter will introduce errors into the optimization. At this point, an adaptive optimization can choose to re-enable the parameter and use the new information to obtain a new estimate. Figure 1.5 shows an example of conditioning in the visual SLAM graphical model. The conditioning edges highlighted in blue link the constant parameters to the rest of the graph which is iteratively re-estimated. For constant-time SLAM, parameters are set to constant once their estimates have converged. Compared to marginalization, linearization errors are not a source of concern when marginalizing parameters. However, care must be taken to re-estimate parameters as more information becomes available as to their estimates. Otherwise, errors due to mis-estimation will be introduced into the problem through the conditioning edges.

1.3 Visual-Inertial SLAM

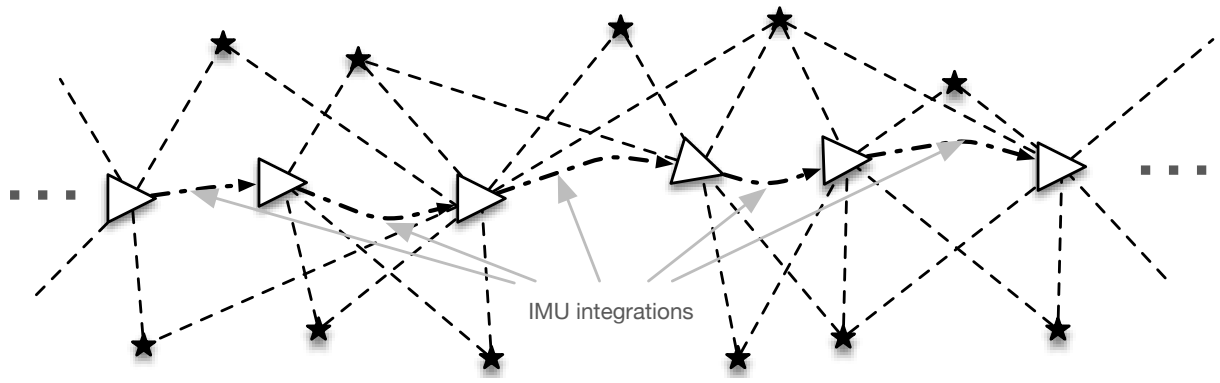


Figure 1.6: Graphical model representing the batch visual-inertial SLAM problem. The visual measurements between landmarks and poses are present as in the visual-only version of the SLAM problem. The inertial measurements introduce a new class of constrain to the optimization: one that connects the poses directly to one another through the inertial measurements. This constraint is formed by integrating the IMU measurements taken between subsequent poses.

In the preceding sections, the SLAM problem has been framed mainly in terms of visual measurements, be it direct or indirect measurements per-pixel, or abstractions extracted in the form of features. However, in the quest for accurate localization and mapping, it would be remiss to ignore any other sources of information. Visual measurements may be corrupted or unavailable due to extraneous factors such as motion blur, insufficient or too might lighting, and blocked cameras. The additional information used to supplement the visual SLAM problem can take the form of measurements taken from a wide range of modalities such as Sonar, Radar, Lidar, encoders, and contact switches. One such sensing modality which is of great interest to visual SLAM is body acceleration and rotation rates. These quantities are typically measured by gyroscopes and accelerometers in an *inertial measurement unit* (IMU). Similar to visual measurements of landmark projections, IMU measurements are taken at discrete points of time, but are measurements of continuous variables. Generally, IMU measurements are taken at might higher frequencies (100-1000Hz) vs. visual measurements (30-60Hz).

Figure 1.6 shows the graphical model representing the batch visual-inertial SLAM problem. A new class of constraint can be seen in the model, linking the poses directly together through the integration of inertial measurements. This graph shows the visual-inertial SLAM problem in its *tightly coupled* form, where visual and inertial measurements are considered simultaneously and all sensor states are optimized. This is chiefly important due to the introduction of new sensors states when considering inertial measurements. Accelerometers measure the second derivative of the quantity of interest to the optimization: the position of the camera. For this reason, velocities must be estimated in order to use inertial measurements. Similarly, IMU measurements are affected by sensor biases which have to be continuously estimated as they change based on extraneous factors such as temperature. Failure to estimate these parameters relegates IMU measurements to their use in *loosely coupled* formulations, where they are used as hints of inclination or rotation, but not tightly integrated with visual measurements. In this thesis the primary focus will be placed on a tightly coupled formulation of visual-inertial SLAM. While inertial measurements can be integrated with dense or sparse visual measurements, in this thesis the primary focus will be on integrating

sparse visual and inertial measurements taken with a monocular camera-IMU setup.

Incorporating IMU measurements in a tightly coupled framework requires their use in residuals involving the same parameters that were used with visual measurements. Sparse visual residuals involve the pose parameters representing the position of the camera when the image was captured, as well as the parameters representing the location of the landmark in 3D space (whether via inverse depth or 3DOF representation). Since landmarks are in no way involved in inertial measurements, the only parameters which are shared between inertial and visual residuals are those representing the camera pose. Referring to these parameters as $\mathbf{x}_p$, we can write the general form of the pose portion of the IMU residual:

$$\mathbf{r}_{i_p} = \mathbf{x}_{p_{k+1}} \boxminus \mathcal{I}\left(\mathbf{x}_{p_k}, \mathbf{x}_i, \mathbf{z}_i\right) \tag{1.6}$$

where $\mathcal{I}\left(\right)$ represents a function which given pose parameters $\mathbf{x}_{p_k}$, and a set of inertial measurements $\mathbf{z}_i$, results in a new set of pose parameters which were *integrated* forward in time from $\mathbf{x}_{p_k}$ using the inertial measurements. The measurements used for the integration must cover the span of time between the kth and the $k+1$th frames. Due to the out-of-sync nature of typical IMU and camera measurements, the inertial measurements are typically interpolated to align precisely to the frame capture times. Once integrated, the new pose can then be compared with the pose parameter for the following frame, $k+1$. The error between the integrated pose and the pose parameter $\mathbf{x}_{p_{k+1}}$ then forms the pose portion of the inertial residual, denoted here as $\mathbf{r}_{i_p}$. The comparison is not a subtraction due to the nonlinearity of the rotation component of the pose. This comparison is denoted here by the $\boxminus$ operator and is explored in detail in Chapter 2. The integration function $\mathcal{I}\left(\right)$ also requires another parameter vector $\mathbf{x}_i$ for the integration. This parameter is required due to the additional state needed to integrate inertial measurements which are of the 1st and 2nd derivatives of rotation and translation respectively. This additional state is generally comprised of a 3DOF body velocity which must be estimated, as well as a representation of the gravity vector. A body velocity estimate is required in order to integrate the 2nd derivative of translation. Similarly, since

the accelerometers sense both body accelerations and those imparted by gravity, and estimate of the gravity vector is required to subtract from the accelerometer measurements. Note that these estimates can also potentially add further residuals to the optimization, for instance to ensure the continuity of the body velocity estimate.

The integration function $\mathcal{I}()$ must be able to accurately model the generation of the inertial measurements in order to precisely integrate the pose forward. Given that body accelerations due to rotation are relative to the privileged frame of reference chosen in the body, the IMU itself is typically set as the privileged frame of reference. This choice decouples the rotation and translation terms in the IMU residual, as body rotations no longer result in measured acceleration. In reality, the gyroscopes and accelerometers if the IMU are not perfectly collocated, voiding this assumption. However this does not generally results in a significant loss of accuracy with MEMS (Microelectromechanical systems) based IMUs, as the distance between the gyroscopes and accelerometers is small.

Moving the privileged frame of reference to the IMU presents a secondary challenge: all visual residuals must now be formed with respect to the IMU frame of reference. This requires an accurate estimate of the transformation between the camera and IMU frames of reference. This extrinsic calibration parameter is critical for accurate visual-inertial SLAM.

1.4 Contributions

The contributions in this thesis cover 3 distinct areas which are covered in the three following chapters. We present a novel framework for visual-inertial SLAM which breaks with the conventional methodology of using marginalization to reduce the computational complexity of the problem and instead uses an adaptive form of conditioning. The motivation for this decision was foremost to alleviate some of the downsides inherent to marginalization such as potential early linearization and inconsistency, as well as presenting a way forward to integrate visual-inertial SLAM into a long-term mapping solution where frequent loop-closure and map localization events are expected. The framework is tested using a number of visual-inertial datasets and is shown to be accurate as

well as running in constant-time.

In order to achieve accurate results, the visual-inertial SLAM framework requires an accurate calibration of the sensor extrinsic and intrinsic parameters. These include parameters such as focal length, image center, and translation/rotation between the IMU and camera. Motivated by long-term autonomy, a novel framework is presented which enables self-calibration of these parameters using nothing but the data which would normally be captured from the sensors. As such, no bespoke calibration targets or procedures are required. Furthermore this framework allows the continued improvement of the calibration estimate in constant-time by detecting and prioritizing the best segments of data for the purposes of calibration. The framework is tested on the problem of camera intrinsic calibration and is shown to closely match the batch optimal estimate in constant time, as well as yielding accurate calibration estimates when compared to offline methods which used bespoke targets. The framework is adaptable to any calibration task as long as constraints can be established between the sensor data and the parameters of interest.

While the self-calibration framework is able to continually improve the estimates of the calibration parameters, it assumes that the underlying parameters themselves are constant. In reality, extraneous influences may change the intrinsic or extrinsic calibration parameters, voiding this assumption. With this in mind, an extension to the framework is presented which can probabilistically determine a significant change in the calibration parameters and trigger a re-calibration process while seamlessly continuing the SLAM estimation task.

1.5 Applications

The contributions in this thesis are applicable to a variety of applications due to the focus on robustness and long-term operation. As previously discussed, long-term autonomy remains a strong focus of the work as well as a viable application. The methods presented here provide both the means to use robust long-term localization and mapping, as well as providing the means to maintain accuracy by continuously monitoring and estimation calibration. Another avenue of interest is hand-held, head-mounted, and other wearable devices which require localization and mapping.

These devices often undergo sharp motions or periods where visual information may be denied. As such, the visual-inertial methods presented in this thesis could be used to maintain accurate localization during these motions. Furthermore, the self-calibration frameworks presented would facilitate continued calibration for accurate localization and mapping. Finally, an increasingly relevant topic where the methods presented here could be of use is self-driving vehicles. The localization, mapping, and self-calibration frameworks presented could all be used in conjunction with other sensors such as wheel odometry and GPS to provide accurate localization to vehicles, as well as continuously maintain calibration between sensors.

1.6 Publications

The work presented in this thesis is based on the following peer-reviewed publications: [47]

- Nima Keivan and Gabe Sibley, "**Asynchronous Adaptive Conditioning for Visual-Inertial SLAM**", In Proceedings of the International Symposium on Experimental Robotics (ISER) 2014
- Nima Keivan and Gabe Sibley, "**Constant-Time Monocular Self-Calibration**", In Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO) 2014
- Nima Keivan and Gabe Sibley, "**Online SLAM with Any-time Self-calibration and Automatic Change Detection**", In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2015
- Nima Keivan and Gabe Sibley, "**Asynchronous Adaptive Conditioning for Visual-Inertial SLAM**", The International Journal of Robotics Research (IJRR), Volume 34, Issue 13, 2015.

Chapter 2

Visual Inertial SLAM

2.1 Introduction

In this chapter, a novel method is introduced to combine visual and inertial measurements for SLAM which avoids the pitfalls of current methods for visual-inertial SLAM which utilize marginalization in order to roll up past parameters into a prior to acheive constant-time computation of the SLAM solution. As discussed in Chapter 1, if compute resources were not a concern, the batch solution would always be desired. It is well known that the batch bundle-adjustment solution to monocular SLAM is the gold standard, in that its form defines the Cramer-Rao lower bound and that it takes advantage of all measurements over all time to compute the maximum likelihood parameter estimate (MLE) ([18, 119]). Due to the addition of a new modality of measurement, the resulting additional states which must be estimated, synchronization of measurements, and two separate frames of references for the sensors, visual-inertial bundle adjustment is significantly more challenging than vision-only BA ([56]). Apart from robustness against the unavailability of high quality visual measurements (such as in the case of fast motions resulting in motion blur or too much or too little lighting), the addition of inertial measurements is particularly important in the case of visual SLAM with a singlae camera. Vision-only monocular systems suffer from a well-studied scale ambiguity, in that the metric scale of the position and scene estimates are ambiguous. Considering inertial measurements can make scale observable, however inertial measurements complicate matters when it comes to computing the global MLE solution incrementally in real-time.

For bundle adjustment to be real-time for use on robots, a **local** approach is typically employed ([77]). With an IMU this is difficult since the local adjustment region may need to be very large in order to ensure observability of certain parameters. This is particularly noticeable regarding parameters which are purely introduced in order to integrate the inertial measurements, such as the body velocity parameters or explicitly or implicitly parameterized direction of the gravity vector. Indeed, under certain degenerate motions such as constant velocity forward motion, some parameters may never be observable (though this rarely if ever happens in practice) ([50, 39]).

An alternative to local-bundle adjustment is to only keep a sliding window of the most recent poses and landmarks active, and marginalize the rest into a prior distribution ([104, 78, 103]). This is equivalent to a fixed-lag Kalman smoother ([67, 26]) and recently such systems have shown remarkable results ([31, 59, 57]). As discussed in chapter 1, marginalization can be an effective way to approximate the batch solution in constant-time, thereby enabling visual-inertial SLAM systems to operate in real-time applications.

Marginalization into a prior distribution as such is predominantly employed for computational efficiency – if it were possible to compute the batch MLE solution in real-time it would be preferable. Marginalization is also costly because it introduces conditional dependencies between the remaining parameters causing "fill-in". An example of fill-in can be seen in Figures 1.3 and 1.4. Once a pose or landmark parameter is marginalized, its effects on the parameters with which it shared constraints are linearized into measurements between the remaining parameters. Fill-in can be addressed by cutting feature tracks and carefully marginalizing poses and landmarks simultaneously ([79]). Marginalization is also potentially dangerous because it bakes in linearization errors which can lead to over-confident estimates or divergence unless one is careful to maintain consistency ([31], [57]). Carrying prior distributions induced from marginalization also necessitates an expensive global optimization at loop-closure to obtain the correct marginal. This is particularly problematic in the case of long term autonomoy, where many loop closures are to be expected as shown in Figure 1.2, due to the creation and maintenance of a consistent map against which continuous localization is desired. The methods presented in this chapter attempt to remedy these downsides

by avoiding marginalization altogether, while still striving to approximate the batch visual-inertial SLAM solution.

2.2 Adaptive Asynchronous Conditioning

Instead of relying on marginalization, conditioning is used in order to reduce the computational complexity of visual-inertial SLAM. Conditioning has has shown surprisingly robust and accurate results when applied to visual SLAM problems ([53, 18]). It also avoids locking in incorrect parameter estimates when used adaptively ([105]). Using a relative manifold to represent the map is also helpful because optimal relative transformation estimates in $\mathbf{SE}(3)$ are by definition near zero. This fact allows multiple threads to asynchronously optimize and update different overlapping subsets of the full problem without detriment.

Adaptive asynchronous conditioning has other benefits: it can **a)** perform robust initialization even under degenerate motions, **b)** allow constant-time loop closure without expensive loop-long re-linearization, **c)** operate even during poor observability conditions, **e)** avoid inconsistency associated with early marginalization and re-linearization, **f)** track the relative-space maximum likelihood solution in constant time. It is shown through experiments that adaptive asynchronous conditioning is accurate, and closely tracks the global batch optimal solution at a fraction of the computational cost, which enables real-time operation.

With sparse visual SLAM, the objective is generally the estimation of keyframe ([54]) and landmark poses, based on the image measurements of tracked 3D features ([119]). The addition of Gyroscope and Accelerometer measurements however, necessitates the estimation of the body velocity and the sensor biases. With these added parameters, the state vector for the batch visual-inertial SLAM problem is defined as

$$\mathbf{X} = \begin{bmatrix} \begin{Bmatrix} \mathbf{x}_{wp_n}^T & \mathbf{v}_{w_n}^T & \mathbf{b_{g_n}}^T & \mathbf{b_{a_n}}^T \end{Bmatrix} & \{\rho_k\} \end{bmatrix}^T, \tag{2.1}$$

where $\begin{Bmatrix} \mathbf{x}_{wp_n}^T & \mathbf{v}_{w_n}^T & \mathbf{b_{g_n}}^T & \mathbf{b_{a_n}}^T \end{Bmatrix}$ is the set of keyframe parameters, defined as follows: $\mathbf{x}_{wp_n} \in$

$\mathbf{SE}(3)$ is the transformation from the coordinates of the nth keyframe to world coordinates, $\mathbf{v}_{w_n} \in \mathbb{R}^3$ is the velocity vector of the nth keyframe in world coordinates, and $\mathbf{b}_{\mathbf{g_n}} \in \mathbb{R}^3$ and $\mathbf{b}_{\mathbf{a_n}} \in \mathbb{R}^3$ are the gyroscope and accelerometer bias parameters for the nth keyframe respectively. Similarly $\{\rho_k\}$ is the 1-d inverse-depth ([85]) parameter for the kth landmark.

Note that in this case, the world frame denotes a "lifted" local frame, where a breadth-first search is used to obtain a local coordinate system from the relative map representation ([68], [105]). This local coordinate frame simplifies the visual and inertial constraints which are minimized in the optimization, in comparison to their relative formulations. Once the optimization is complete, the relative representation of the optimized parameters is written back into the map. Note that the relative map representation is not a prerequisite of the proposed method, and is used as it facilitates updates to a single map structure from multiple asynchronous optimizations. Section 2.2.5 further expands on the specifics of using a relative map representation with visual-inertial SLAM.

The parameterization of $\mathbf{x}_{wp_n}$ deserves special notice. The transformation has 6 degrees of freedom: 3 for translation, and 3 for rotation. However, 6DOF representations of transformations suffer from singularities, due to what is known as Gimbal Lock in the rotation degrees of freedom. To avoid this problem, the rotation component of the transformation is represented as a quaternion

$$\mathbf{x}_{wp} = \begin{bmatrix} \mathbf{p}_{wp}^T & \mathbf{q}_{wp}^T \end{bmatrix}^T, \tag{2.2}$$

where $\mathbf{q}_{wp} \in \mathbb{R}^4$ is a quaternion representing the rotation from keyframe to world coordinates, and $\mathbf{p}_{wp} \in \mathbb{R}^3$ is the translation vector from keyframe to world coordinates. This representation has 7 degrees of freedom and as such, is over-parameterized for the underlying 6 degrees of freedom. To avoid the null spaces caused by over-parameterizing the space in an optimization, a *local parameterization* of the space is utilized as follows

$$\mathbf{x}_{wp'} = \mathbf{x}_{wp} \oplus \mathbf{x}_{pp'}\left(\Delta\mathbf{x}_q, \Delta\mathbf{x}_p\right) = \begin{bmatrix} \left(\mathbf{q}_{wp} \otimes \exp_{\mathbf{q}}\left(\Delta\mathbf{x}_q\right)\right)^T & \left(\mathbf{p}_{wp} + \Delta\mathbf{x}_p\right)^T \end{bmatrix}^T, \tag{2.3}$$

where $\mathbf{x}_{pp'}(\Delta\mathbf{x}_q, \Delta\mathbf{x}_p)$ represents an *update* applied to the transformation $\mathbf{x}_{wp}$, composed of a 3DOF translation delta $\Delta\mathbf{x}_p$ and a 3DOF rotation delta $\Delta\mathbf{x}_q$, and $\otimes$ is the quaternion multiplication operator. The translation delta is additive and is simply added to the translation vector of $\mathbf{x}_{wp}$. The rotation delta is in the $\mathfrak{so}(3)$ *tangent space* ([111]) and is transformed into the $\mathbf{SO}(3)$ manifold using the exponential operator. The resulting quaternion can then be multiplied by $\mathbf{q}_{wp}$ to apply the update to the rotation. Here $\oplus$ is the update operator, equivalent to matrix multiplication if $\mathbf{x}_{wp}$ and $\mathbf{x}_{pp'}$ were represented as 4×4 homogenous transformation matrices. $\Delta\mathbf{x}_q$ and $\Delta\mathbf{x}_p$ now represent a 6dof manifold which can be used to update the keyframe pose without over-parameterization.

2.2.1 Probabilistic Derivation

Given the state vector in Equation 2.1, a probabilistic method will be derived to obtain the optimal estimates for the parameters given visual and inertial measurements. The joint probability distribution of the state vector $\mathbf{X}$ and the visual-inertial measurement set $\mathbf{Z}$ can be factored using Baye's rule as follows:

$$P(\mathbf{X}, \mathbf{Z}) = P(\mathbf{Z}|\mathbf{X})\, P(\mathbf{X}), \tag{2.4}$$

where $P(\mathbf{Z}|\mathbf{X})$ is the measurement *likelihood* and $P(\mathbf{X})$ is the *prior* term. Assuming conditional independence between measurements, the likelihood term can be factored:

$$P(\mathbf{X}, \mathbf{Z}) = \prod_{i=1}^{n} P(\mathbf{z}_i|\mathbf{X})\, P(\mathbf{X}), \tag{2.5}$$

where $P(\mathbf{z}_i|\mathbf{X})$ is the likelihood of the ith measurement given the state vector $\mathbf{X}$. The optimal estimate for the state vector parameters is then one that maximizes the joint probability of the state and measurements in Equation 2.4, which is also achieved by maximizing the measurement likelihood and prior probabilities:

$$\widehat{\mathbf{X}} = \arg\max_{\mathbf{X}} P\left(\mathbf{X}, \mathbf{Z}\right) = \arg\max_{\mathbf{X}} \left(\prod_{i=1}^{n} P\left(\mathbf{z}_i | \mathbf{X}\right) P(\mathbf{X}) \right). \tag{2.6}$$

With the assumption that the measurement terms are normally distributed, the likelihood term for the ith measurement term (Equation 2.5) can be written as a multivariate normal distribution with mean $h(\mathbf{X})$ and covariance $\Sigma_{\mathbf{z}}$ as follows

$$P\left(\mathbf{z}_i | \mathbf{X}\right) = \frac{1}{\sqrt{\left(2\pi\right)^k \left|\Sigma_{\mathbf{z}}\right|}} \exp\left(-\frac{1}{2} \left(\mathbf{z}_i - h(\mathbf{X})\right)^T \Sigma_{\mathbf{z}}^{-1} \left(\mathbf{z}_i - h(\mathbf{X})\right) \right) \propto \exp\left(-\frac{1}{2} \|\mathbf{z}_i - h(\mathbf{X})\|_{\Sigma_{\mathbf{z}}}^2 \right), \tag{2.7}$$

where the proportional relation ($\propto$) is used to omit the normalization term, and $\|\mathbf{z} - h(\mathbf{X})\|_{\Sigma_{\mathbf{z}}}^2 = \left(\mathbf{z} - h(\mathbf{X})\right)^T \Sigma_{\mathbf{z}}^{-1} \left(\mathbf{z} - h(\mathbf{X})\right)$ denotes the squared Mahalanobis distance. $h(\mathbf{X})$ is the measurement function, which predicts the measurement $\mathbf{z}$ given the state vector.

Likewise, with the assumption that the prior term is normally distributed with mean Π and covariance Σ_{Π}, it can be written as

$$P(\mathbf{X}) \propto \exp\left(-\frac{1}{2} \|\mathbf{X} - \Pi\|_{\Sigma_{\Pi}}^2 \right). \tag{2.8}$$

Given Equations 2.7 and 2.8, Equation 2.6 can be written as

$$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} P\left(\mathbf{X}, \mathbf{Z}\right) = \arg\max_{\mathbf{X}} \left[\prod_{i=1}^{n} \exp\left(-\frac{1}{2} \|\mathbf{z}_i - h(\mathbf{X})\|_{\Sigma_{\mathbf{z}}}^2 \right) \exp\left(-\frac{1}{2} \|\mathbf{X} - \Pi\|_{\Sigma_{\Pi}}^2 \right) \right]. \tag{2.9}$$

Taking the negative log of Equation 2.9, a cost function can be obtained, in order to obtain the maximum a posteriori estimate for the state vector $\mathbf{X}$

$$\hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \left(\sum_{i=1}^{n} \|\mathbf{z}_i - h(\mathbf{X})\|_{\Sigma_{\mathbf{z}}}^2 + \|\mathbf{X} - \Pi\|_{\Sigma_{\Pi}}^2 \right), \tag{2.10}$$

where the first term represents a sum over all measurements residuals, and the second term represents the prior residual. Note that since both the state vector $\mathbf{X}$ and prior mean Π contain rotations, the subtraction operator is not sufficient to obtain a residual between the two. Instead, due to the existence of discontinuities in the space of rotations, the difference between two quaternions is used as a measure of distance:

$$\Delta\mathbf{q} = \log_{\mathbf{q}}\left(\mathbf{q}_{\mathbf{X}} \otimes \mathbf{q}_{\Pi}^{-1}\right), \tag{2.11}$$

where the $\log_{\mathbf{q}}$ operator transforms the 4DOF difference quaternion from the $\mathbf{SO}(3)$ manifold to the 3DOF $\mathfrak{so}(3)$ tangent space. In Section 2.2.3, it will be shown that in the case of inertial measurements, a similar approach is required in order to compute the residual $\mathbf{z}_i - h(\mathbf{X})$ as rotation terms are involved. This detail does not however, invalidate the aforementioned derivation.

Note that although the aforementioned maximum a posteriori formulation includes a prior distribution on the state parameters, the proposed method does not make use of a prior, and as such uses maximum likelihood estimation. This is further explained in Section 2.2.4.

2.2.2 Visual Measurements

Visual measurements are formed by tracking the 2D image projection location of 3D landmarks in the scene. A residual is then computed from the difference in the predicted 2D image position of the landmark and the actual measured 2D position. Figure 2.1 shows the parameters involved in a single visual residual. The measurement function $h(\mathbf{X})$ for visual residuals is defined as follows (Note that the transformation $\mathbf{T}_{wp}$ is the equivalent 4×4 matrix representation of $\mathbf{x}_{wp}$, which is used here for brevity. The underlying implementation uses the quaternion and translation components of $\mathbf{x}_{wp}$):

$$h(\mathbf{X}) = \mathcal{P}\left(\mathbf{p}_r, \mathbf{X}\right) = \mathcal{P}\left(\mathbf{T}_{pc}^{-1}\mathbf{T}_{wp_m}^{-1}\mathbf{T}_{wp_r}\mathbf{T}_{pc}\mathcal{P}^{-1}\left(\mathbf{p}_r, \rho\right)\right), \tag{2.12}$$

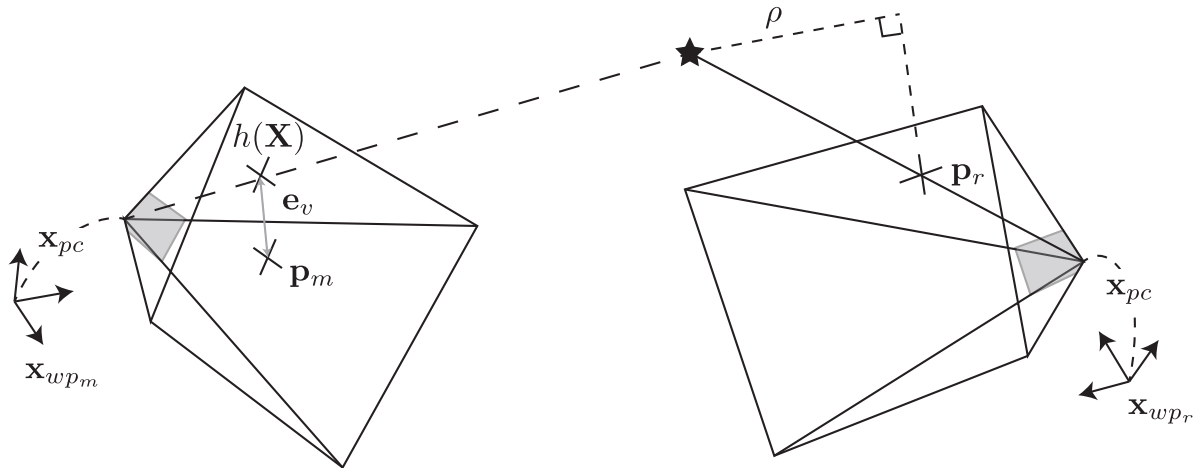


Figure 2.1: The formation of a visual residual given a landmark and the reference and measurement keyframes. The landmark is first formed by corner detection, where $\mathbf{p}_r$ denotes the image location of the detected corner in the reference frame $\mathbf{x}_{wp_r}$. The landmark inverse depth parameter ρ specifies the perpendicular distance between the landmark and the image plane. The measurement keyframe reference frame $\mathbf{x}_{wp_m}$ specifies the transformation from the measurement keyframe to world coordinates, and $\mathbf{p}_m$ is the detected image location of the landmark in the measurement frame. As the keyframe pose corresponds to the IMU frame (Section 2.2.3), $\mathbf{x}_{pc}$ denotes the transformation from the camera to the IMU frame. Given the aforementioned values, the landmark can be projected into the measurement frame, where an error vector ($\mathbf{e}_\mathcal{V}$) is formed with the difference between the predicted and measured landmark location in the image.

where ρ is the inverse depth of the landmark, $\mathbf{T}_{wp_r}$ is the transformation from the coordinates of the reference keyframe (in which the landmark was first seen and initialized) to world coordinates, $\mathbf{T}_{wp_m}$ is the transformation from the measurement keyframe to world coordinates, $\mathbf{p}_r$ is the 2D image location where the original feature was initialized in the reference keyframe, $\mathbf{p}_m$ is the measured 2D image location in the measurement keyframe, $\mathbf{T}_{pc}$ is the transformation from the camera to the keyframe coordinates, $\mathcal{P}^{-1}$ is a 2D to 3D back-projection function which returns the homogenous landmark position given the reference 2D image location $\mathbf{p}_r$ and the inverse depth ρ, and $\mathcal{P}$ is a 3D to 2D camera projection function which returns the predicted 2D image coordinates. This operation forms a transform function from the image plane of the reference camera to that of the measurement camera, given their respective poses and the inverse depth of the landmark. The camera to keyframe transformation $\mathbf{T}_{pc}$ is non-zero as the keyframe is collocated on the inertial frame (the frame in which inertial measurements are made), to simplify the inertial integration

equations. As shown in Figure 2.1, the visual measurement residual of the kth landmark in the mth frame is calculated from the error vector between the predicted and measured 2D location, $\mathbf{e}_v$ as follows

$$r_{\mathcal{V}_{m,k}} = \|\mathbf{e}_{\mathcal{V}_{m,k}}\|^2_{\Sigma_{\mathbf{p}_{m,k}}} = \|\mathbf{p}_{m,k} - \mathcal{P}\left(\mathbf{T}_{pc}^{-1}\mathbf{T}_{wp_m}^{-1}\mathbf{T}_{wp_{r,k}}\mathbf{T}_{pc}\mathcal{P}^{-1}\left(\mathbf{p}_r, \rho_k\right)\right)\|^2_{\Sigma_{\mathbf{p}_{m,k}}}, \tag{2.13}$$

where $\mathbf{p}_{m,k}$ is the measured 2D image location of the kth landmark in the mth keyframe with covariance $\Sigma_{\mathbf{p}_{m,k}}$, and $\mathbf{T}_{wp_{r,k}}$ is the transform from coordinates of the reference keyframe of the kth landmark to world coordinates.

Since the camera intrinsics are assumed constant, the back-projection of the reference feature location onto the $z = 1$ plane can be computed once and then stored for all reprojections involving that landmark. This pre-computation step provides significant benefits where non-invertible camera models are used, as the inverse projection function for these camera models can be computationally expensive. The back-projection is undertaken as follows:

$$[\mathbf{x}_k\,;\,1.0] = \mathcal{P}^{-1}\left(\mathbf{p}_r, 1.0\right), \tag{2.14}$$

where $\mathbf{x}_k \in \mathbb{R}^3$ is the 3D vector representing the feature on the $z = 1$ plane of the reference camera. Given this precalculated vector, the residual can be re-written by simply replacing the $z = 1$ inverse depth value with the parameter ρ, obviating the inverse projection function in Equation 2.13:

$$r_{\mathcal{V}_{m,k}} = \|\mathbf{e}_{\mathcal{V}_{m,k}}\|^2_{\Sigma_{\mathbf{e}_{m,k}}} = \|\mathbf{p}_{m,k} - \mathcal{P}\left(\mathbf{T}_{pc}^{-1}\mathbf{T}_{wp_m}^{-1}\mathbf{T}_{wp_{r,k}}\mathbf{T}_{pc}\left[\mathbf{x}_k\,;\,\rho\right]\right)\|^2_{\Sigma_{\mathbf{p}_{m,k}}}. \tag{2.15}$$

Here the covariance of the error vector $\mathbf{e}_{\mathcal{V}_{m,k}}$ is equal to the covariance of the 2D pixel location detected in the image $\Sigma_{\mathbf{p}_{m,k}}$, due to the fact that the measurement and error vector are in the same space. More formally, if the image measurement is defined as

$$\mathbf{p}_{m,k} \sim \mathcal{N}\left(\bar{\mathbf{p}}_{m,k}, \Sigma_{\mathbf{p}_{m,k}}\right), \tag{2.16}$$

where $\bar{\mathbf{p}}_{m,k}$ is the true measurement, then the distribution over the error vector can be written as

$$P\left(\mathbf{e}_{\mathcal{V}_{m,k}}|\mathbf{X}\right) = P\left(\mathbf{p}_{m,k} - h\left(\mathbf{X}\right)\right) \sim \mathcal{N}\left(h\left(\mathbf{X}\right), \Sigma_{\mathbf{p}_{m,k}}\right). \tag{2.17}$$

Therefore the measurement uncertainty is used directly in the mahalanobis distance used for the residual. This distinction is made to motivate the covariance derivation in Section 2.2.3, as the inertial measurement and error vector are not in the same space. Linear error propagation must then be used to obtain the covariance of the error vector. The covariance $\Sigma_{\mathbf{p}_{m,k}}$ can be obtained from the autocorrelation matrix used in extracting harris corners ([29]). However, due to the fact that the salient corners used for tracking are generally thresholded to have high autocorrelation, a standard value of 1 pixel is used for the x and y dimension covariance. In other words, $\Sigma_{\mathbf{p}_{m,k}} = \mathbf{I}_{2\times 2}$.

2.2.3 Inertial Measurements

Similar to Section 2.2.2, inertial measurements obtained from the inertial measurement unit (IMU) are used in order to form constraints over state variables. The values obtained from the IMU are 3 degrees of freedom acceleration and angular rate measurements. IMU measurements are taken in the IMU coordinate frame. This fact motivates the use of the IMU coordinate frame as the privileged frame, as the equations governing the rigid body dynamics are greatly simplified. Both accelerometer and gyroscope measurements are assumed to have additive Gaussian noise, and are therefore defined as follows

$$\omega_m \sim \mathcal{N}\left(\mathbf{q}_{wp}^{-1} \otimes \left(\omega_w\right) + \mathbf{b}_g, \Sigma_\omega\right) \tag{2.18}$$

$$\mathbf{a}_m \sim \mathcal{N}\left(\mathbf{q}_{wp}^{-1} \otimes \left(\mathbf{a}_w - \mathbf{g}_w\right) + \mathbf{b}_a, \Sigma_{\mathbf{a}}\right), \tag{2.19}$$

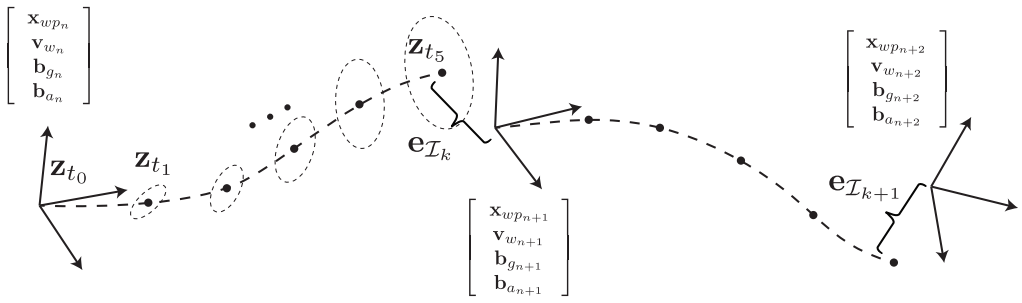


Figure 2.2: Inertial residuals formed by integrating inertial measurements between subsequent keyframes. The residual vector $\mathbf{e}_{\mathcal{I}_{k+1}}$ is then formed using the error between the integrated inertial measurements from keyframe n and the state parameters of keyframe $n+1$. The ellipsoids represent the (exaggerated) uncertainty in the integrated state, which grows as more noisy measurements are used in the integration. The final state uncertainty is then used in computing the weight of the residual.

where ω_m refers to the angular rate measurements in the IMU frame obtained from the sensor, $\mathbf{q}_{wp}$ is the quaternion defining the rotation from IMU to world coordinates, ω_w is the true angular rate of the IMU in the world frame, $\mathbf{b}_g$ is the gyroscope bias vector, and $\Sigma_\omega \in \mathbb{R}^{3\times3}$ is the covariance matrix specifying the uncertainty in gyroscope measurements. Similarly for acceleration measurements, ω_m is the acceleration measurements in the IMU frame obtained from the sensor, $\mathbf{a}_w$ refers to the true acceleration of the IMU in the world frame, $\mathbf{g}_w$ is the gravity vector in the world frame, $\mathbf{b}_a$ is the accelerometer bias vector, and $\Sigma_\mathbf{a} \in \mathbb{R}^{3\times3}$ is the covariance matrix specifying the uncertainty in the accelerometer measurements. The effects of the Earth's rotation have been ignored in these equations.

Note that the angular velocity vector $\omega_w \in \mathbb{R}^{3\times1}$ is the minimal representation of the $\mathfrak{so}(3)$ tangent space. As the measurement is in the IMU frame, the tangent space needs to be transformed between two different $\mathbf{SO}(3)$ reference frames. This operation is the Adjoint Map ([111]), which for $\mathbf{SO}(3)$ is simply multiplication by the rotation matrix transforming from the start to end reference frame. Therefore multiplication by the equivalent quaternion $\mathbf{q}_{wp}^{-1}$ can be used to transform the angular velocities in the world tangent space to the IMU frame tangent space.

In order to constrain the state variables, the equations of rigid body motion are used to obtain

analytical relationships between the IMU measurements and the state parameters. The integration state for the kth step ($\mathbf{x}_k \in \mathbb{R}^{16}$) is arranged as follows:

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_{wp_k}^T & \mathbf{q}_{wp_k}^T & \mathbf{v}_{w_k}^T & \mathbf{b}_{g_k}^T & \mathbf{b}_{a_k}^T \end{bmatrix}^T. \tag{2.20}$$

The relationship between the state parameters and IMU measurements is derived as follows:

$$\begin{aligned} \dot{\mathbf{p}}_{wp} &= \mathbf{v}_w \\ \dot{\mathbf{v}}_w &= \mathbf{a}_w = \mathbf{q}_{wp} \otimes (\mathbf{a}_m - \mathbf{b}_a) + \mathbf{g}_w \\ \dot{\mathbf{b}}_g &= \mathbf{w}_g \\ \dot{\mathbf{b}}_a &= \mathbf{w}_a. \end{aligned} \tag{2.21}$$

The time-derivatives of translation ($\dot{p}_{wp}$) and velocity ($\dot{\mathbf{v}}_w$) are straightforward. The accelerometer and gyroscope biases are modeled as Gaussian random walk processes. Consequently, their discrete time derivatives are given by the white Gaussian noise vectors $\mathbf{w}_g \in \mathbb{R}^3$, and $\mathbf{w}_a \in \mathbb{R}^3$, where elements of each vector are independently drawn from a zero-mean Gaussian distribution with variance of σ_{b_g}, and σ_{a_g} respectively. The case of rotation derivatives deserves special attention. Given that the skew-symmetric angular velocity tensor $\widehat{\omega}$, denoted by the hat operator ([111]) is in the $\mathfrak{so}(3)$ tangent space, Euler integration of the angular velocities to form an incremental rotation in $\mathbf{SO}(3)$ can be performed using the exponential operator: $\Delta\mathbf{q}_{wp} = \exp_{\mathbf{q}}(\widehat{\omega}_w dt)$, where $\exp_{\mathbf{q}}$ returns the $\mathbf{SO}(3)$ rotation in the form of a quaternion.

This incremental rotation can then be compounded with the original rotation matrix. Using a quaternion to represents elements of $\mathbf{SO}(3)$, the discrete integration of of the angular velocities for a single time step can then be written as

$$\mathbf{q}_{wp_{k+1}} = \exp(\hat{\omega}_{w_k} dt) \otimes \mathbf{q}_{wp_k} = \exp_{\mathbf{q}}\left(\widehat{\mathbf{q}_{wp} \otimes (\omega_{m_k} - \mathbf{b}_{g_k})} dt\right) \otimes \mathbf{q}_{wp_k}, \tag{2.22}$$

where the IMU-frame angular rate measurement ω_{w_k} is transformed into the world frame by the $\mathbf{SO}(3)$ Adjoint. The other parameters are similarly discretely integrated for a single time step:

$$\begin{aligned}
\mathbf{p}_{wp_{k+1}} &= \mathbf{p}_{wp_k} + \mathbf{v}_{w_k} dt \\
\mathbf{v}_{w_{k+1}} &= \mathbf{v}_{w_k} + \mathbf{a}_w dt = \mathbf{v}_{w_k} + \left(\mathbf{q}_{wp} \otimes \left(\mathbf{a}_{m_k} - \mathbf{b}_{a_k}\right) + \mathbf{g}_w\right) dt \\
\mathbf{b}_{g_{k+1}} &= \mathbf{b}_{g_k} \\
\mathbf{b}_{a_{k+1}} &= \mathbf{b}_{a_k}.
\end{aligned} \tag{2.23}$$

The gyroscope and accelerometer biases are modeled as random walk processes, and as such are not modified in the integration. However, the uncertainty over the bias parameters grows with the integration. Note that although Euler integration is used here for brevity, in the actual implementation, Runge-Kutta 45 integration is used for higher accuracy.

Given the aforementioned integration approach, the inertial measurement set $\mathbf{Z} = \left\{\mathbf{z}_{t_0}, \ldots, \mathbf{z}_{t_n}\right\}$ where $\mathbf{z} = \begin{bmatrix} \omega & \mathbf{a} \end{bmatrix}$ can be used to integrate the $\mathbf{SE}(3)$ pose, velocity and biases from time t_0 to t_n. A residual can then be formed between the integrated state ($\mathbf{x}'$) and the parameters in the next keyframe as shown in Figure 2.2:

$$r_{\mathcal{I}_j} = \|\mathbf{e}_{\mathcal{I}_j}\|^2_{\Sigma_{\mathbf{e}_{ij}}} = \left\| \begin{bmatrix} \mathbf{p}_{wp_{j+1}} - \mathbf{p}' \\ \log_{\mathbf{q}}\left(\mathbf{q}^{-1}_{wp_{j+1}} \otimes \mathbf{q}'\right) \\ \mathbf{v}_{w_{j+1}} - \mathbf{v}' \\ \mathbf{b}_{g_{j+1}} - \mathbf{b}_{g_j} \\ \mathbf{b}_{a_{j+1}} - \mathbf{b}_{a_j} \end{bmatrix} \right\|^2_{\Sigma_{\mathbf{e}_{\mathcal{I}_j}}}, \tag{2.24}$$

where $\log_{\mathbf{q}}\left(\mathbf{q}^{-1}_{wp_{j+1}} \otimes \mathbf{q}'\right) \in \mathbb{R}^3$ is used to calculate the $\mathfrak{so}(3)$ difference between the integrated orientation $\mathbf{q}'$ and the orientation of the $(j+1)$th keyframe, $\mathbf{q}_{wp_{j+1}}$. This is required, as the quaternion parameterization is redundant, and the underlying space has only 3 degrees of freedom.

2.2.3.1 Residual Covariance

Unlike the visual residual covariance in Equation 2.17, the covariance of the inertial residual $\Sigma_{\mathbf{e}_{i_j}}$ is not due to the uncertainty of a single measurement. Rather, the uncertainty of multiple measurements inflate the covariance of the integrated state as shown in Figure 2.2. In order to obtain this covariance, the uncertainty of the integrated state must be calculated at each stage of the integration. This involves the propagation of the current uncertainty for the time-step k to time-step $k+1$, as well as adding the uncertainty introduced from the measurements integrated at time k as follows:

$$\Sigma_{\mathbf{x}_{k+1}}\left(\mathbf{x}_k, \Sigma_{\mathbf{x}_k}, \mathbf{z}_k, \Sigma_{\mathbf{z}_k}\right) = \left(\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k}\right) \Sigma_{\mathbf{x}_k} \left(\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{x}_k}\right)^T + \left(\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{z}_k}\right) \Sigma_{\mathbf{z}_k} \left(\frac{\partial \mathbf{x}_{k+1}}{\partial \mathbf{z}_k}\right)^T, \tag{2.25}$$

where $\Sigma_{\mathbf{x}_k} \in \mathbb{R}^{16\times16}$ is the singular covariance matrix of the most recent integration state, and $\Sigma_{\mathbf{z}_k} \in \mathbb{R}^{12\times12}$ is the covariance matrix of the IMU measurements and the bias drift model. It should be noted that $\Sigma_{\mathbf{x}_k}$ is singular due to the redundant 4D representation of rotation, during the integration on the $\mathbf{SE}(3)$ manifold. However, this covariance is propagated into the minimal $\mathfrak{se}(3)$ tangent space before being used in the optimization, and will no longer be singular. (See Equation 2.28). $\mathbf{x}_k \in \mathbb{R}^{16}$ is the integration state matrix arranged as

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{p}_{wp_k}^T & \mathbf{q}_{wp_k}^T & \mathbf{v}_{w_k}^T & \mathbf{b}_{g_k}^T & \mathbf{b}_{a_k}^T \end{bmatrix}^T, \tag{2.26}$$

and $\mathbf{z} = \begin{bmatrix} \omega & \mathbf{a} \end{bmatrix}$ is the measurement vector at time-step k. The IMU measurement and bias covariance matrix is usually diagonal, as the measurement uncertainties of the separate axes of the gyroscope and accelerometer and their biases are not correlated. It is formulated as

$$\Sigma_{\mathbf{z}_k} = \text{diag}\left(\sigma_g^2 \cdot \mathbf{I}_{3\times3}, \sigma_a^2 \cdot \mathbf{I}_{3\times3}, \sigma_{b_g}^2 \cdot \mathbf{I}_{3\times3}, \sigma_{b_a}^2 \cdot \mathbf{I}_{3\times3}\right), \tag{2.27}$$

where σ_g, σ_a, σ_{b_g}, and σ_{b_a} are the uncertainties for the gyroscope, accelerometer, gyroscope bias

and accelerometer bias respectively. The derivatives ${}^{\partial \mathbf{x}_{k+1}}/_{\partial \mathbf{x}_k}$ and ${}^{\partial \mathbf{x}_{k+1}}/_{\partial \mathbf{z}_k}$ are straightforward and are obtained by differentiating the single-step integrations in Equation 2.23.

Using the uncertainty propagation in Equation 2.25, the uncertainty of the final state $\Sigma_{\mathbf{x}'}$ can be obtained. Note that at the first step of the integration, the uncertainty of the state is set to 0. More formally: $\Sigma_{\mathbf{x}_0} = 0_{16\times16}$. The uncertainty of the error vector $\mathbf{e}_{\mathcal{I}}$ can then be obtained from $\Sigma_{\mathbf{x}'}$:

$$\Sigma_{\mathbf{e}_{\mathcal{I}}} \in \mathbb{R}^{15\times15} = \left(\frac{\partial \mathbf{e}_i}{\partial \mathbf{x}'}\right) \Sigma_{\mathbf{x}'} \left(\frac{\partial \mathbf{e}_i}{\partial \mathbf{x}'}\right)^T, \tag{2.28}$$

where ${}^{\partial \mathbf{e}_{\mathcal{I}}}/_{\partial \mathbf{x}'} \in \mathbb{R}^{15\times16}$ is the Jacobian of the function which forms the error residual from the integrated state $\mathbf{x}'$, and the subsequent keyframe:

$$\frac{\partial \mathbf{e}_{\mathcal{I}}}{\partial \mathbf{x}'} = \begin{bmatrix} -\mathbf{I}_{3\times3} & 0_{3\times4} & 0_{3\times9} \\ 0_{3\times3} & {}^{\partial \log_{\mathbf{q}}\left(\mathbf{q}_{wp_{j+1}}^{-1} \otimes \mathbf{q}'\right)}/_{\partial \mathbf{q}'} & 0_{3\times9} \\ 0_{9\times3} & 0_{9\times4} & -\mathbf{I}_{9\times9} \end{bmatrix}. \tag{2.29}$$

Due to the simple subtractions used to form the errors, this Jacobian is mostly comprised of the negative identity matrix $-\mathbf{I}$. The only exception is the orientation term. As a redundant 4D quaternion parameterization is used during the integration, the covariance term for it will be singular. The orientation block in ${}^{\partial \mathbf{e}_{\mathcal{I}}}/_{\partial \mathbf{x}'}$ is then used to propagate this uncertainty into the $\mathfrak{se}(3)$ tangent-space where the 3D error will be formed.

Note that depending on the uncertainty and number of IMU measurements used in the integration, $\Sigma_{\mathbf{e}_{\mathcal{I}}}$ could become singular or badly conditioned, in which case a pseudo-inverse should be used to compute the mahalanobis distance in Equation 2.7. Unlike the visual residual covariance, the inertial residual covariance depends on the state, via propagation through the state integration. However, this dependence is assumed to be negligible, as the residual covariances are recomputed before every optimization iteration.

2.2.3.2 Jacobian Calculation

In order to optimize the residual in Equation 2.24 in a maximum-likelihood estimator, partial derivatives of the error vector $\mathbf{e}_{\mathcal{I}_j}$ with respect to the involved state parameters are required. The error vector $\mathbf{e}_{\mathcal{I}_j}$ is formed by integrating IMU measurements starting at the jth keyframe, and comparing the integrated state with that of the $(j+1)$th keyframe. Therefore the involved state parameters are the position, orientation, velocity and biases of these two keyframes. Given the inertial residual in Equation 2.24, it can be observed that the derivatives with respect to the $(j+1)$th keyframe are easily obtained, as they do not involve the integration of the inertial measurements. However, the same cannot be said for the jth keyframe, as it is the starting state for the integration. The partial derivatives would then need to be computed step-by-step through the integration using the chain rule. In order to simplify the required partial derivatives, some of the starting keyframe state parameters can be factored in the integration, as shown in section . The error vector and residual can then be re-written given this factorization, by replacing the components of the final integration state $\mathbf{x}'$ with their factorized forms:

$$r_{\mathcal{I}_j} = \|\mathbf{e}_{\mathcal{I}_j}\|^2_{\Sigma_{\mathbf{e}_{i_j}}} = \left\| \begin{bmatrix} \mathbf{p}_{wp_{j+1}} - \left(\mathbf{p}_{wp_j} + \mathbf{v}_{w_j}\Delta t + \frac{1}{2}\mathbf{g}_w\Delta t^2 + \mathbf{q}_{wp_j} \otimes \Delta\mathbf{p}\right) \\ \log_{\mathbf{q}}\left(\mathbf{q}^{-1}_{wp_{j+1}} \otimes \left(\mathbf{q}_{wp_j} \otimes \Delta\mathbf{q}\right)\right) \\ \mathbf{v}_{w_{j+1}} - \left(\mathbf{v}_{w_j} + \mathbf{g}_w\Delta t + \mathbf{q}_{wp_j} \otimes \Delta\mathbf{v}\right) \\ \mathbf{b}_{g_{j+1}} - \mathbf{b}_{g_j} \\ \mathbf{b}_{a_{j+1}} - \mathbf{b}_{a_j} \end{bmatrix} \right\|^2_{\Sigma_{\mathbf{e}_{\mathcal{I}_j}}}, \tag{2.30}$$

where Δt is the total time of the integration, and $\Delta\mathbf{p}$, $\Delta\mathbf{q}$, and $\Delta\mathbf{v}$ are specially integrated deltas of position, orientation and velocity which do not depend on any state parameters, as described in section . Given this factorization, partial derivatives with respect to velocity, position and orientation of the jth keyframe need not be computed through the integration, and can be directly evaluated. Unfortunately the same cannot be said about the partial derivatives with respect to the bias: $\partial\mathbf{e}_{\mathcal{I}_j}/\partial\mathbf{b}_{g_j}$ and $\partial\mathbf{e}_{\mathcal{I}_j}/\partial\mathbf{b}_{a_j}$, which have to be computed using the chain rule as follows:

$$\frac{\partial \mathbf{e}_{\mathcal{I}_j}}{\partial \mathbf{b}_j} = \frac{\partial \mathbf{e}_{\mathcal{I}_j}}{\partial \mathbf{x}'} \cdot \frac{\partial \mathbf{x}_n}{\partial \mathbf{b}_j} = \frac{\partial \mathbf{e}_{\mathcal{I}_j}}{\partial \mathbf{x}'} \cdot \left(\frac{\partial \mathbf{x}_n}{\partial \mathbf{x}_{n-1}} \cdot \frac{\partial \mathbf{x}_{n-1}}{\partial \mathbf{x}_{n-2}} \cdot \ldots \cdot \frac{\partial \mathbf{x}_1}{\partial \mathbf{b}_j} \right). \tag{2.31}$$

It is worth noting that due to the additive noise model used for IMU measurements, (Equations 2.18 and 2.19), the partial derivative of the integration step with respect to the biases ($\partial \mathbf{x}_n / \partial \mathbf{b}_j$) is in fact equal to the partial derivative with respect to the measurement noise ($\partial \mathbf{x}_{k+1} / \partial \mathbf{z}_k$), which is used in the propagation of the measurement uncertainty in Equation 2.25. This partial derivative can be easily computed from the single-step integration in Equations 2.22 and 2.23 and used for both the partial derivatives of the error vector, and to propagate the measurement uncertainties.

2.2.4 Optimization Formulation

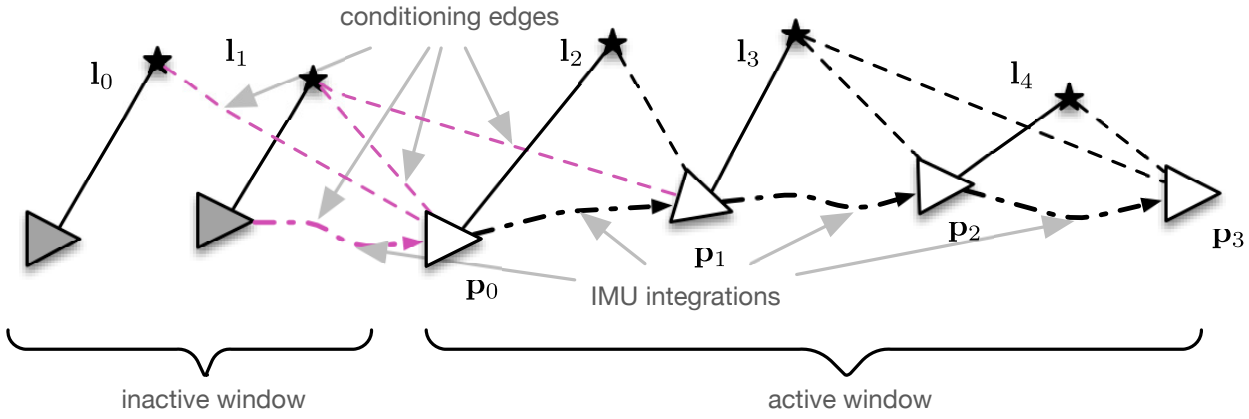


Figure 2.3: The graphical model representing the conditioning sliding window optimization over a number of active keyframes. Edges in the graph are formed by either visual or inertial residuals and are shown as dashed lines. The solid lines represent implicit edges modeled by the inverse depth parameterization between landmarks and their reference keyframes. The conditioning edges are formed by residuals which involve both active and inactive parameters.

The previous sections outline the formulation of inertial and visual residuals. In a batch setting, all measurements collected would be used in order to optimize the state parameters. However, this is clearly not an acceptable solution for online systems, as the computational complexity of the problem is unbounded as more measurements are collected. One way to deal with this problem is to recursively marginalize landmarks and keyframes that fall outside a fixed-size window of active

parameters. These older keyframes then form a prior distribution which is used alongside the visual and inertial residuals ([78],[58], [56], [57]). This approach however generally results in inconsistencies in the estimator, due to the multiple linearization points using in the marginalization process. One approach to fixing the inconsistencies is to use *first estimate* Jacobians ([57]), however this issue will greately reduce the tolerance of the estimator towards nonlinearities. Another issue with carrying a marginal distribution is difficulties when closing loops, as the marginal distribution will no longer be valid after the loop closure.

An alternate approach is to condition on parameters that fall outside the fixed-size window instead. By conditioning, an assumption is made that the estimates for these past parameters have converged and are correct. Figure 2.3 shows the graphical model representing the conditioning approach, where the conditioning edges are composed of measurements that involve both the active and inactive parameters. Conversely, active edges are composed of measurements that involve only active parameters. Equation 2.6 is then modified to remove the prior and to split the measurement likelihood term in two: active and conditioning measurements:

$$\widehat{\mathbf{X}}_a = \arg\max_{\mathbf{X}_a} P\left(\mathbf{X}_a, \mathbf{Z}\right) = \arg\max_{\mathbf{X}_a} \left(\prod_{i=1}^{n_a} P\left(\mathbf{z}_i | \mathbf{X}_a\right) \prod_{j=1}^{n_c} P\left(\mathbf{z}_j | \mathbf{X}_a, \mathbf{X}_i\right) \right), \tag{2.32}$$

where n_a and n_c are the number of active and conditioning residuals respectively. Note that the likelihood term for the conditioning residuals is conditioned on both the active ($\mathbf{X}_a$) and inactive ($\mathbf{X}_i$) state parameters, but only the active parameters are optimized. Given the inertial and visual residuals discussed in the previous section, and the aforementioned conditioning approach the total cost optimized is formulated as :

$$e = \sum_{m=1}^{n_k} \sum_{k=1}^{n_l} \| \mathbf{e}_{\mathcal{V}_{m,k}} \|^2_{\Sigma_{\mathbf{e}_{\mathcal{V}_{m,k}}}} + \sum_{j=1}^{n_k} \| \mathbf{e}_{\mathcal{I}_j} \|^2_{\Sigma_{\mathbf{e}_{\mathcal{I}_j}}}, \tag{2.33}$$

where n_k and n_l are the total number of keyframes and landmarks respectively. Residuals which involve inactive parameters in $\mathbf{X}_i$ are evaluated normally. However, the partial derivatives of these

residuals with respect to the inactive parameters are omitted. This method is used extensively with a fixed-window optimization in visual SLAM ([68], [54]).

In order to minimize the given cost and obtain the maximum likelihood state estimates, the normal equations are utilized to iteratively update the active parameters $\mathbf{X}_a$:

$$\mathbf{J}^T\mathbf{W}\mathbf{J}\Delta\mathbf{X}_a = \mathbf{J}^T\mathbf{W}\mathbf{e}, \tag{2.34}$$

where $\mathbf{e} = \begin{bmatrix} \mathbf{e}_{\mathcal{V}_0} & \dots & \mathbf{e}_{\mathcal{V}_n}, & \mathbf{e}_{\mathcal{I}_0} & \dots & \mathbf{e}_{\mathcal{I}_m} \end{bmatrix}^T$ is the residual vector, $\mathbf{J} = {}^{\partial \mathbf{e}}/_{\partial \mathbf{X}_a}$ is the Jacobian of the residual vector with respect to the active state parameters, and the block diagonal weight matrix is composed of the inverse covariance matrix for every residual: $\mathbf{W} = \text{diag}\left(\Sigma^{-1}_{\mathbf{e}_{\mathcal{V}_0}}, \dots, \Sigma^{-1}_{\mathbf{e}_{\mathcal{V}_n}}, \Sigma^{-1}_{\mathbf{e}_{\mathcal{I}_0}}, \dots, \Sigma^{-1}_{\mathbf{e}_{\mathcal{I}_m}}\right)$. The dog-leg trust region method ([92]) is used to solve Equation 2.34. Note that the update to the state vector $\Delta\mathbf{X}_a$ is additive for all parameters except the orientation parameter $\mathbf{q}_{wp}$, where the update is applied through the exponential operator as per Equation 2.3.

Due to the connectivity of the visual-inertial SLAM graph, new residuals may significantly alter the estimates for parameters which are no longer in the active window. Due to this same connectivity, active parameters which are conditioned on badly estimated past parameters are themselves poorly estimated, and can result in total divergence of the solution. In the case of visual-inertial slam, parameters such as velocity, and implicitly estimated parameters such as the direction of gravity are particularly sensitive to mis-estimation, and can derail the solution if not updated in the presence of new measurements. As such, a fixed-window optimization will not approximate the batch maximum likelihood solution. This motivates the introduction of the adaptive window optimization.

2.2.5 Adaptive window implementation

The adaptive local bundle adjustment dynamically sets the window size, in order to appropriately fold in parameters as needed. As is especially prominent when using an IMU, future measurements can significantly affect estimates of past and inactive parameters for a fixed window

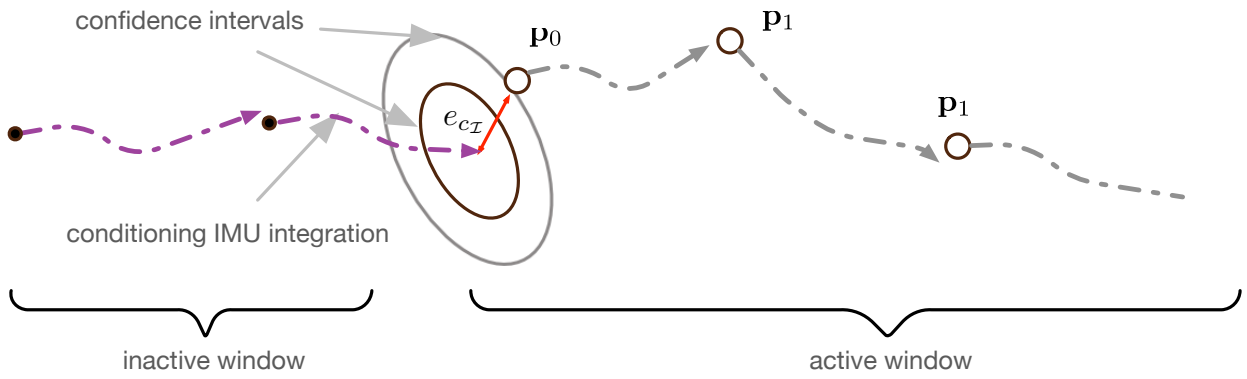


Figure 2.4: Graphical model of the adaptive condition shown for inertial residuals. The ellipsoids show the confidence intervals of the Mahalanobis distance between the final integration pose and the first active keyframe as obtained from a χ^2 test. The size of the adaptive window is then expanded if the Mahalanobis distance is larger than a threshold confidence interval. Once expanded, the optimization is run over the larger window and the process is repeated by checking the same edge (which will no longer be a conditioning edge, as it will involve only active parameters).

size. Dynamically adjusting the window serves to allow the optimization to include parameters when new measurements are available which change their estimates, and when these new estimates affect the current active set of parameters Examples of these parameters are velocities, accelerometer and gyroscope biases, and the direction of gravity, which is implicitly parameterized.

Similar to the adaptive method previously presented in [105], the condition used to determine the size of the optimization window is based on the residuals. In [105], parameters were included in the optimization if changes in the visual residuals were larger than a specific threshold. For the case of visual-inertial SLAM, since multiple sensor modalities are used, the Mahalanobis distance for the residuals is computed instead, and thresholded in a χ^2 test, in order to probabilistically determine when residuals are outside their expected intervals. In particular, the condition used to assess whether the size of the window needs to be increased between two optimization iterations is based on the residuals observed in the conditioning edges shown in Fig. 2.3, after the kth optimization has converged. The measurement covariances can then be used to assess whether the conditioning residuals are within expected bounds using a χ^2 test and to adjust the size for the $(k+1)$th optimization. The conditioning Mahalanobis distance for projection residuals is

$$e_{c\mathcal{V}} = \sum_{i \in C} \|\mathbf{r}_{\mathcal{V}_i}\|^2_{\Sigma_{V_i}}, \tag{2.35}$$

where the summation is over the set C comprising of all conditioning visual residuals as shown in Figure 2.3. Likewise, the conditioning Mahalanobis distance for the single inertial residual connecting the active and inactive sets is

$$e_{c\mathcal{I}} = \|\mathbf{r}_{\mathcal{I}_c}\|^2_{\Sigma_{\mathcal{I}_c}}. \tag{2.36}$$

Given either $e_{c\mathcal{I}}$ or $e_{c\mathcal{V}}$, their corresponding adaptive condition variables $\alpha_{k_\mathcal{V}}$ and $\alpha_{k_\mathcal{I}}$ can be defined as

$$\begin{aligned} \alpha_{k_\mathcal{V}} &= \frac{e_{c\mathcal{V}}}{\text{Inv}\,\chi^2\,(\beta, d)}, \\ \alpha_{k_\mathcal{I}} &= \frac{e_{c\mathcal{I}}}{\text{Inv}\,\chi^2\,(\beta, d)}, \end{aligned} \tag{2.37}$$

where $d = 2|C| + 15$, and $\alpha_{k_\mathcal{V}}$ and $\alpha_{k_\mathcal{I}}$ represent the adaptive condition variables for visual and inertial residuals respectively, $\text{Inv}\,\chi^2\,(\beta, d)$ is the inverse cumulative χ^2 distribution for d dimensions evaluated for the confidence interval β, where d is the total dimension of the conditioning edge vector (15 for the conditioning inertial residual, and $2 \times |C|$ for the conditioning visual residuals). Since the multivariate Mahalanobis distance has a d-dimensional χ^2 distribution, the cumulative form of the χ^2 can be used to estimated if the given Mahalanobis distance lies in a certain confidence interval. Initially if $\alpha_{k_\mathcal{V}} > 1$ or $\alpha_{k_\mathcal{I}} > 1$, the conditioning residuals for either visual or inertial measurements lie outside the βth percentile probability as expected from the residual covariance (as shown in Figure 2.4), so the window size is increased, and the optimization is run to convergence with the now larger window. This increase in the window size is continued while the following conditions hold:

$$(\alpha_{k+1_\mathcal{V}} > 1 \vee \alpha_{k+1_\mathcal{I}} > 1) \wedge ((\alpha_{k+1_\mathcal{V}} + \alpha_{k+1_\mathcal{I}}) \leq \gamma \cdot (\alpha_{k_\mathcal{V}} + \alpha_{k_\mathcal{I}})), \tag{2.38}$$

where the $\wedge$ and $\vee$ signify the `AND` and `OR` boolean algebra operators, and $\gamma = (1 - 1e^{-5})$ is a tuning parameter to ensure that the condition variables decrease more than a specific threshold with regards to the previous optimization. If the condition in Equation 2.38 is not met, the window is resized to its default minimum length and new frames are added to the window. Note that during the expansion, even though the conditioning edges of iteration k are replaced with earlier edges in the graph, the condition variables $\alpha_{k_\mathcal{V}}$ and $\alpha_{k_\mathcal{I}}$ are still computed at the same edge as the first optimization instance where the condition in Equation 2.38 was met.

In the case of the inertial residual Mahalanobis distance (Equation 2.36), the residual covariance actually depends on the state, as it is propagated through the inertial integration as per Equation 2.25. Consequently, the true Mahalanobis distance after the optimization must be computed by propagating the IMU measurement uncertainties given the latest estimate of the states. However, since the residual covariances are recomputed before each iteration of the optimization, and changes to the parameters are expected to be small at convergence, the change in the covariance of the inertial residual is assumed to be negligible in the final optimization iteration. Note that in the case of visual residuals, the residual covariance does not depend on the state.

The intuition behind the adaptive criterion on Equation 2.38 is that if new residuals would affect the estimates of past parameters, and those parameters are not part of the active window, tension will be introduced in the conditioning edges which connect the active to the inactive parameters in the form of errors which are not explained by the measurement uncertainty. If the errors in the conditioning edges are indeed caused by mis-estimation of inactive parameters outside the window, including these parameters in the window should reduce these errors until they are within expected intervals based on measurement uncertainties. In the case that the conditioning error is not decreasing but is still outside expected bounds, the window size is returned to its default minimal value and the expansion is stopped, as the error is more likely explained by outlier measurements.

In order to use a relative map representation with visual-inertial measurements, the architecture presented in [105] has to be extended to accomodate the new parameters which are estimated:

keyframe velocities, gravity direction and IMU biases. As the biases are estimated in the IMU frame, they are already relative and so need no further attention. The velocities are estimated in the "lifted" local frame, and are therefore relative to the orientation of the first lifted pose. The same can be said about the direction of gravity in the lifted frame, $\mathbf{g}_w$. In order to store the velocity estimates and gravity direction in the map, they are converted to a relative parameterization, which for the kth keyframe is as follows:

$$\begin{aligned} \mathbf{v}_{p_k} &= \mathbf{q}_{wp_k}^{-1} \otimes \mathbf{v}_{w_k} \\ \mathbf{g}_{p_k} &= \mathbf{q}_{wp_k}^{-1} \otimes \mathbf{g}_w, \end{aligned} \tag{2.39}$$

where $\mathbf{v}_{p_k}$ and $\mathbf{g}_{p_k}$ are the velocity and gravity direction vectors relative to the keyframe orientation. These values are stored in the map along with the relative transform between keyframes. Given this map representation, a local window can be "lifted" by performing a breadth first search starting from a given root node. The local gravity vector $\mathbf{g}_w$ can then be computed by inverting Equation 2.39. The local velocity for each keyframe $\mathbf{v}_{w_k}$ is computed similarly. These values, alongside the local position and orientation of the keyframe obtained as per [105], are then used as the optimization state parameters.

The use of a relative map representation ensures that updates to the map remain small. While the optimization is performed in a "lifted" local frame (referred to as the world frame), the resulting parameters are pushed back into the map in their relative form, allowing multiple BAs to update it asynchronously without clashing.

Note that if the window size expands to the point where a batch optimization is performed, the relevant null-spaces are regularized to prevent the hessian becoming singular. For the visual-inertial case, the null-spaces for the batch maximum likelihood estimation are the 3 unobservable translations of the root pose, and the unobservable yaw degree of freedom around gravity. In order to remove the yaw null-space, the rotation axis most parallel to that of the gravity vector is regularized in the first pose.

Rig	IMU	Camera
A	Project Tango Peanut IMU @ 120Hz	Project Tango Peanut Wide-Angle Camera @ 30fps, 640 × 480
B	Microstrain 3DM-GX3-25 @ 200Hz	Point Grey FL3-U3-13Y3M-C @ 30fps, 640 × 480

Table 2.1: Specifications for the sensors used in each rig.

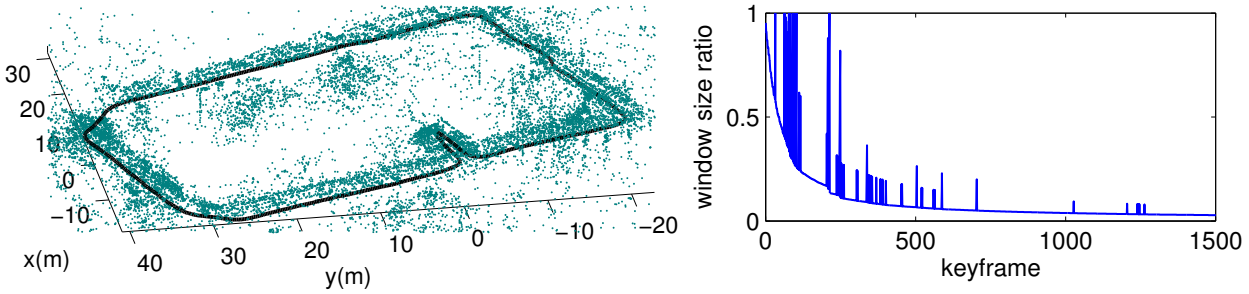
Since the time needed to run the optimization to convergence scales with the size of the window, the aforementioned solution could potentially be too slow for real-time operation, if the window size expands too much. To obtain a real-time solution, two optimization windows are run simultaneously in a multi-threaded environment. One thread runs a constant-width window optimization which is guaranteed to run at framerate, while the other runs the adaptive window optimization, which could potentially run slower than framerate. Special care is taken to ensure that these two optimizations which run on different subsets of parameters do not destructively interfere with one another. Fortunately, the relative map representation is quite conducive to this approach. Since only relative values are stored in the map, updates to the parameters are small. This would not be the case for a global map representation, where changes to past keyframes could potentially have significant downstream effects on more recent keyframes. Given the relative map representation, a greedy update approach is chosen, where synchronization between the threads is only performed during the **lift** and **write** operations, but otherwise updates are written to the map as soon as either optimization thread has run to convergence.

2.3 Experiments

In order to evaluate the proposed method, experiments were run on two sensor platforms: A hand-held mobile device (hereby referred to as rig A) with a global shutter wide field-of-view camera with 640×480 resolution and a commercial MEMS IMU sampled at 120Hz, and a custom hand-held rig (referred to as rig B) with a wide field-of-view camera at 1280×960 resolution downsampled to 640×480 with a commercial grade IMU sampled at 200Hz. The sensor specifications are listed in Table 2.1. Both cameras capture images at 30 frames per second. A comparison between the images captured from the rigs is shown in Figure 2.5. In all experiments, the AAC system

(a) (b)

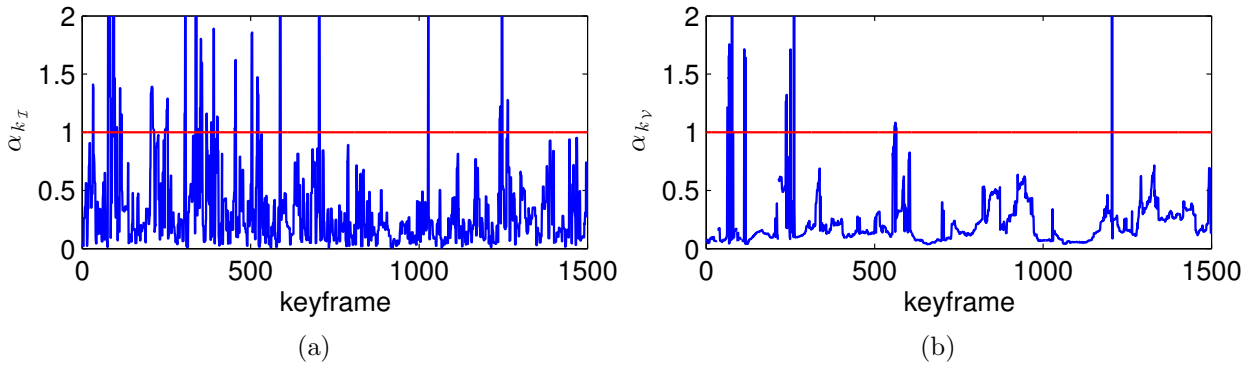
Figure 2.5: Comparison of images obtained from rig A (a) and rig B (b) as described in Section 4.3. Rig A is a mobile hand-held device with a wide-angle lens and a global shutter camera capturing at 30 fps and with 640×480 resolution. Rig B is a custom hand-held device with a global shutter camera capturing at 30fps and with 1280×960 resolution downsampled to 640×480. The picture quality and contrast of rig B is generally much better. Both rigs use a commercial grade MEMS IMUs: rig A sampled at 120Hz and rig B at 200Hz. Rig B features a much larger lens and higher quality camera.



(a) (b)

Figure 2.6: a) Trajectory and structure estimate resulting from running the poposed solution on a ∼200m outdoor dataset captured using rig B. The error between the start and end keyframe poses is 0.33% of the distance travelled. b) The window size ratio (computed by dividing the adaptive window size by the total number of keyframes) for the trajectory, the initialization region is at the beginning of the trajectory with the ratio reaching 1, signifying a batch solution.

comprises of a fixed-window estimator (as per Section 2.2.4) with a 10 keyframe window width and an asynchronous adaptive estimator (as per Section 2.2.5) with a minimum window size of 20 keyframes.



(a) (b)

Figure 2.7: Inertial (a) and visual (b) adaptive condition variable plots (see Section 2.2.5) for the trajectory in Figure 2.6a. The red line indicates the threshold over which the condition in Equation 2.38 will be true, and therefore the window size will be expanded. The expansion spikes in Figure 2.6b are correlated to the keyframes at which either the visual or inertial condition variable is > 1 in (a) or (b)

.

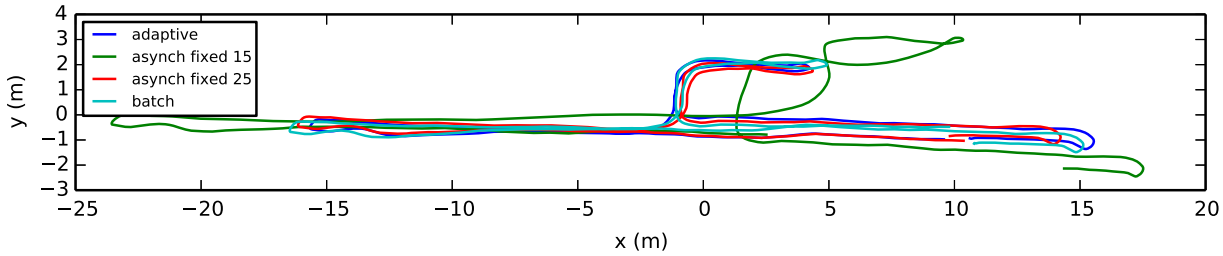


Figure 2.8: Comparison of trajectories estimated by different bundle adjustment configurations. It can be seen that the 25 keyframe asynchronous fixed-size window BA and the adaptive window BA both produce trajectories close to the batch solution, however the 15 keyframe fixed-size asynchronous fixed window BA diverges substantially from the batch solution.

2.3.1 Tracking and Keyframing

Visual residuals are established as per Section 2.2.2 by tracking salient image points between frames, to form a *feature track*. This is accomplished via a method inspired by the tracking component of ([22]) , where the photometric error of a re-projected feature patch is directly minimized to obtain the new location of the feature. In order to initialize new feature tracks, Harris corners ([29]) are used in the areas of the image where not enough active tracks are present. In the direct approach employed, epipolar geometry is respected when minimizing the photometric error to lo-

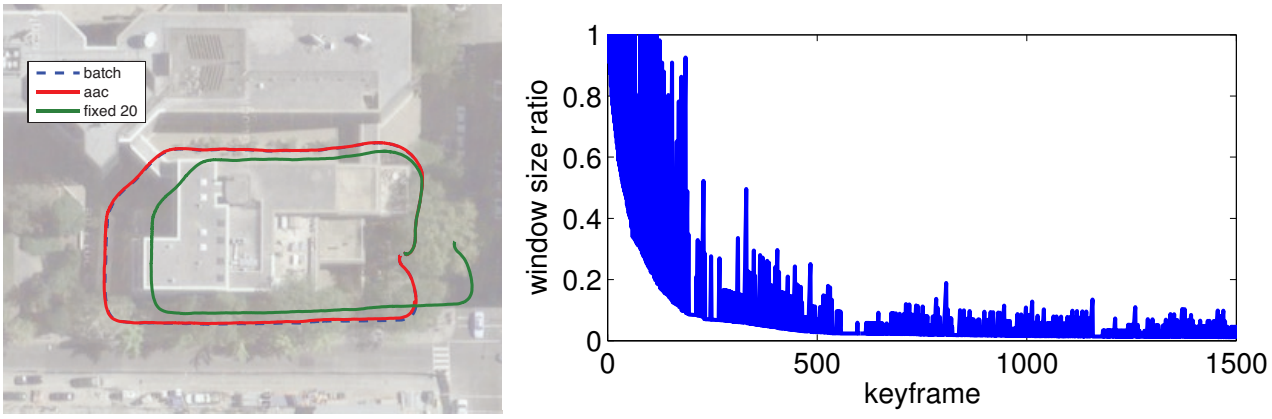
calize the tracked features in new images. As such, the RANSAC ([21]) step generally employed when descriptor-based matching is used is no longer required. There is slight tolerance to violations of epipolar geometry to enable tracking if the landmarks or keyframe poses are not well estimated, and as such non-static objects in the scene may still violate epipolar geometry over the length of a feature track. These outliers are rejected in the fixed and adaptive window optimizations based on the ratio of outlier visual residuals to the total number of residuals in the feature track. A normalized cross-correlation (NCC) score is computed between the current and original feature patches and is thresholded (at 0.875) to reject feature patches which have changed too much in appearance. In all examples, the tracker was configured to attempt to track 128 landmarks across the image. The value of the tuning parameter β from Equation 2.37 is set to 0.1, meaning that $\alpha_k > 1$ if the conditioning Mahalanobis distance for either visual or inertial residuals is larger than the 10% χ^2 confidence interval for the given dimension d. The value of the tuning parameter β was experimentally chosen to ensure that the window size was expanded when required estimate past and present parameters on all datasets. Statistically, a threshold on the 10th percentile is rather strict, as only a small percentage of errors should reside in this range, and most errors should be higher. The experimentally obtained value of 10% could potentially be due to measurement covariances which are set too low, resulting in under-estimated Mahalanobis distances.

Keyframing is used as a means to both increase performance and to alleviate problems arising from a stationary camera. The keyframing approach is similar to that of [54]. Heuristics are used over the total distance and rotation since the last keyframe, as well as the percentage of feature tracks still active. If any of the heuristics are met, a new keyframe is placed and the feature track measurements are inserted into the map. Images which are not keyframes are simply used for localization against the map. Note that IMU measurements are recorded irrespective of keyframing. The system runs at 30 frames per second on a 2.5Ghz Core i7 laptop with two threads. One thread running the synchronous estimator as described in Section 2.2.4 and the other running the AAC (Section 2.2.5).

2.3.2 Results

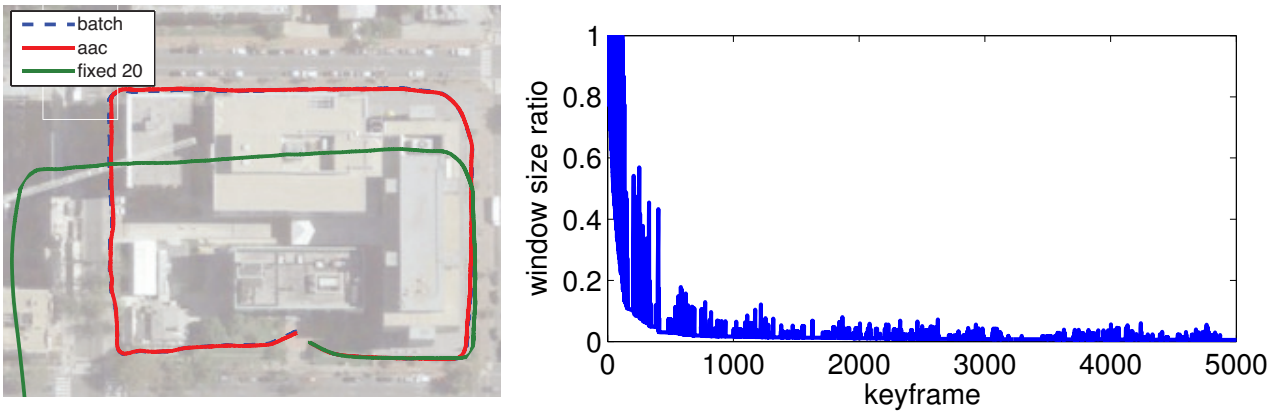
The first experiment consists of a ∼200m outdoor loop captured with rig B with identical start and end positions. The trajectory and reconstruction are shown in Figure 2.6a, where the final error between the start and end keyframe poses as a percentage of traveled distance is 0.33%. Extension 1 shows the captured images and estimated trajectory for this experiment. Figure 2.6b shows the asynchronous adaptive window size ratio, calculated by dividing the adaptive window size by the total number of keyframes. It can be observed that near the beginning of the trajectory, the asynchronous window ratio does reach one, which signifies that the asynchronous estimator is solving the batch solution with all keyframes and measurements. However, this period is short lived, after which the size of the adaptive window settles to an approximately constant number of keyframes. The spikes in 2.6b correspond to increases in the adaptive window size due to the condition in Equation 2.38 being met. These spikes can be matched against Figures 2.7a and 2.7b which show the plots for $\alpha_{k_\mathcal{V}}$, and $\alpha_{k_\mathcal{I}}$ (as explained in Section 2.2.5), where the red line indicates the threshold after which the condition is met and the adaptive window size is expanded.

Figure 2.9 shows the trajectory reconstruction of a short indoor sequence obtained using rig A showing the comparison between the batch solution, which uses all keyframes and measurements, and several options for the asynchronous estimator. It is observed that the adaptive window size is the one that most closely matches the batch. While the fixed-size asynchronous window with 25 keyframes performs fairly well compared to the batch solution, the 15 keyframe fixed-size window clearly diverges. Rig A was also used to capture two outdoor datasets shown in Figures 2.9 and 2.10. Figure 2.9a shows the trajectory reconstruction superimposed over an aerial photo for a ∼200m outdoor loop. It can be seen that the adaptive result closely matches the batch result, while a fixed-window optimization has significantly higher error. The batch translation error between the start and end keyframes for this trajectory is 0.71% of the traveled distance and 0.72% when using the AAC estimator. The window size ratio (similar to Figure Fig 2.6b) for this dataset is shown in Figure 2.9b, where it is observed that after an initialization period where the size ratio is frequently



(a) (b)

Figure 2.9: A loop consisting of an outdoor ∼200m dataset taken on foot with rig A, superimposed over satellite imagery. The batch translation error between the start and end keyframes as a percentage of traveled distance is 0.71% while the AAC error is 0.72%. The resulting poses obtained by running AAC, batch and a fixed-window optimization over the data are shown in (a), and the ratio of the AAC window to the total number of keyframes is shown in (b). A ratio of 1.0 indicates a batch solve. It can be seen that a fixed window optimization is unable to accurately match the batch solution.



(a) (b)

Figure 2.10: A loop consisting of an outdoor ∼400m dataset taken on foot with rig A, superimposed over satellite imagery. The batch translation error between the start and end keyframes as a percentage of traveled distance is 1.33% while the AAC error is 1.42. The resulting poses obtained by running AAC, batch and a fixed-window optimization over the data are shown in (a), and the ratio of the AAC window to the total number of keyframes is shown in (b). A ratio of 1.0 indicates a batch solve. It can be seen that the solution obtained using a fixed-window batch solver diverges completely.

1 (indicating a batch solution), the window size decreases to an approximately constant number.

Figure 2.10a shows the reconstructed trajectory for a ∼400m outdoor trajectory obtained using rig A. The 20 keyframe fixed-window optimization clearly diverges and is unable to estimate the trajectory, while the batch and AAC results match closely, with error of 1.33% and 1.42% respectively. Figure 2.10b shows the window size ratio for this trajectory, where once again after a short initialization period, the AAC window size remains approximately constant.

While the approach is observed to be accurate, a large discrepancy in accuracy is observed between datasets captured with the two different rigs in Section 4.3. Given that both rigs were calibrated with the same offline calibration tool, the discrepancy can be attributed to both the higher image quality of rig B due to the higher quality camera and much larger lens, and to the higher sampling rate for the IMU of rig B. These effects are also visible in the window size ratio plots between rigs A and B, (Figure 2.6b vs. Figure 2.9b or Figure 2.10b), where it can be seen that adaptive window expansion is much more prevalent in Figure 2.9b and Figure 2.10b which were captured with rig A. This signifies that good parameter estimates were less likely to be computed given the minimum asynchronous window size, and frequent expansions were necessary. This is in contrast to the higher quality data obtained with rig B, and the resulting window size ratio shown in Figure 2.6b. Another factor which has a large impact on accuracy is the value of the IMU measurement covariance matrix, $\Sigma_{\mathbf{z}_k}$. The weight placed on the IMU residual is derived from the propagation of this covariance matrix, and an underestimation of the uncertainty can lead to significant reduction of the system accuracy due to over-reliance on noisy IMU data.

2.3.3 Initialization

In Figures 2.6b, 2.9b and 2.10b it can be observed that for a short time after the start of the sequence, the window size ratio tends to be equal to 1, indicating a batch solution. This behavior arises naturally due to the adaptive window formulation (Section 2.2.5) and serves to initialize values which may not be immediately observed at the beginning of the trajectory. Among the values which tend to fluctuate the most in this initialization phase are the accelerometer and

gyroscope biases, the two degrees of freedom of orientation with respect to gravity, and the velocity vectors for the keyframes. As these quantities are not immediately available, new information about them will *stress* the conditioning edges and force the AAC estimator to expand the window in order to update all affected poses. However once the initial estimates for these parameters have converged, the AAC window seldom expands to the batch solution.

2.4 Discussion

It was observed that in real-life situations, parameters such as velocity, gravity and bias are observable with adaptive conditioning. This is of course contingent upon sufficient excitation of the sensors. In the hand-held datasets there is an ever present oscillatory acceleration due to the walking motion, which quickly renders the unknown parameters observable. Given this, a short required window size is observed in order to closely estimate the batch MLE solution. As expected, window growth is also seen in situations where scale and consequently velocity are ambiguous. An example of this is at sharp turns which introduce a slew of uninitialized new landmarks while simultaneously cutting tracks from established landmarks. The net result is a scale ambiguity that requires a larger window size to resolve, which is automatically discovered.

It must be noted that in the case of prolonged degenerate motion (such as constant velocity), the size of the window required to properly estimate parameters can grow without bound to the batch solution. However, in the real datasets that were used to evaluate the system, such a scenario was never encountered, as enough excitation of the sensors was generally observed, and the window size was bounded.

The system is observed to be accurate, however consistent discrepancies in accuracy are observed between reconstructions captured with different visual-inertial rigs. Given equal offline calibration of the two rigs, the effect of sensor quality and IMU sampling frequency on the accuracy of the reconstruction is observed to be significant, and is a candidate for further study.

When using asynchronous BA, care must be taken so as to ensure sufficient update frequency of the asynchronous solution in order to ensure overlap with the synchronous BA. This is required

to keep the synchronous BA in the overall solution basin as solved by the asynchronous BA. As expected from the relative framework, the updates to the edges and inverse depth parameters for landmarks are small and no interference was observed between the two threads, even when the asynchronous BA is solving a batch solution during initialization (4.2.2), in which case its updates will be markedly slower than the synchronous BA.

The use of conditioning in lieu of marginalization presents desirable properties. However, it must be noted that marginal covariances are no longer computed for state estimates, and parameters which are conditioned upon are assumed to be correctly estimated. Since these marginal covariances are ignored, a loss of accuracy can generally be expected. In this case of adaptive conditioning, a concerted effort is to re-estimate past parameters in the presence of new measurements, constantly reducing their error where possible. As a result, it is shown that the adaptive conditioning method is still able to obtain very high accuracy, especially when using quality sensors.

2.5 IMU measurement integration

In this section, the IMU integration equations are expanded to factorize the starting parameter values where possible, similar to the factorizations presented in [59] and [64]. The motivation behind this is to obviate the need to compute the derivatives with respect to the starting parameters by traversing the integration via the chain rule. A successful factorization allows the derivatives to be calculated after the integration is performed, both increasing the accuracy of the derivatives as well as increasing performance. It is shown that this factorization is possible for all starting parameters except for the IMU biases.

2.5.1 Rotation Integration Factorization

In the interest of brevity, the derivation of the factorized rotation equations is first performed using rotation matrices (denoted as $\mathbf{R}$). The factorized form can then easily be written using quaternions (Equation 2.45), which is the final form used in the implementation. Consequently, the exponential and adjunct operators performed on rotation matrices are denoted as $\exp_{\mathbf{R}}$ and

$\text{Ad}_{\mathbf{R}}$ to disambiguate them from their quaternion counterparts. The integration of angular velocities from time 0 to n can be written as

$$\mathbf{R}_{wp_n} = \exp_{\mathbf{R}}\left(\widehat{\omega}_{w_{n-1}} dt_{n-1}\right) \times \ldots \times \exp_{\mathbf{R}}\left(\widehat{\omega}_{w_0} dt_0\right) \mathbf{R}_{wp_0}, \tag{2.40}$$

where $\mathbf{R}_{wp_0}$ is the rotation matrix describing the starting rotation. Since this rotation is in fact a parameter in the optimization, derivatives with respect to it will be required in order to construct the problem Jacobian. Since the measurement in world coordinates depends on the orientation of the IMU frame, its formulation must be taken into account to obtain the derivative. Fortunately, the starting orientation can be factored from the integration. To do this, a slightly different form of the adjunct is used where: $\widehat{\text{Ad}_{\mathbf{R}} \cdot \omega} = \text{Ad}_{\mathbf{R}} \cdot \widehat{\omega} \cdot \text{Ad}_{\mathbf{R}}^{-1}$ ([111]). Since for SO(3), $\text{Ad}_{\mathbf{R}} = \mathbf{R}$, Equation 2.22 can be rewritten as

$$\mathbf{R}_{wp_{k+1}} = \exp_{\mathbf{R}}(\hat{\omega}_{w_k} dt)\mathbf{R}_{wp_k} = \exp_{\mathbf{R}}\left(\mathbf{R}_{wp}\widehat{(\omega_{m_k} - \mathbf{b}_{g_k})} dt \mathbf{R}_{wp}^{-1}\right) \mathbf{R}_{wp_k}, \tag{2.41}$$

which gives the slightly modified form of Equation 2.40:

$$\mathbf{R}_{wp_n} = \exp_{\mathbf{R}}\left(\mathbf{R}_{wp_{n-1}}\widehat{(\omega_{m_{n-1}} - \mathbf{b}_{g_{n-1}})} dt_{n-1} \mathbf{R}_{wp_{n-1}}^T\right) \times \ldots \times \exp_{\mathbf{R}}\left(\mathbf{R}_{wp_0}\widehat{(\omega_{m_0} - \mathbf{b}_{g_0})} dt_0 \mathbf{R}_{wp_0}^{-1}\right) \mathbf{R}_{wp_0}. \tag{2.42}$$

Since $\mathbf{R}_{wp_{n-1}}$ is the orientation as integrated up to step $n-1$, it can be factored into the starting orientation $\mathbf{R}_{wp_0}$, and an integrated orientation delta up to time-step $n-1$, $\mathbf{R}_{p_0 p_{n-1}}$. More formally: $\mathbf{R}_{wp_n} = \mathbf{R}_{wp_0}\mathbf{R}_{p_0 p_n}$. Using this factorization, Equation 2.42 becomes:

$$\begin{aligned}\mathbf{R}_{wp_n} = &\exp_{\mathbf{R}}\left(\mathbf{R}_{wp_0}\mathbf{R}_{p_0 p_{n-1}}\widehat{(\omega_{m_{n-1}} - \mathbf{b}_{g_{n-1}})} dt_{n-1} \mathbf{R}_{p_0 p_{n-1}}^{-1}\mathbf{R}_{wp_0}^{-1}\right) \times \ldots \times \\ &\exp_{\mathbf{R}}\left(\mathbf{R}_{wp_0}\widehat{(\omega_{m_k} - \mathbf{b}_{g_k})} dt_0 \mathbf{R}_{wp_0}^{-1}\right) \mathbf{R}_{wp_0}.\end{aligned} \tag{2.43}$$

Using the Lie matrix exponential identity $\exp_{\mathbf{R}}(\mathbf{A}\mathbf{X}\mathbf{A}^{-1}) = \mathbf{A}\exp_{\mathbf{R}}(\mathbf{X})\mathbf{A}^{-1}$, Equation 2.42 becomes:

$$\begin{aligned}\mathbf{R}_{wp_n} =& \mathbf{R}_{wp_0}\mathbf{R}_{p_0p_{n-1}}\exp_{\mathbf{R}}\left(\widehat{(\omega_{m_{n-1}} - \mathbf{b}_{g_{n-1}})}dt_{n-1}\right)\mathbf{R}^{-1}_{p_0p_{n-1}}\mathbf{R}^{-1}_{wp_0} \times \ldots \times \\ & \mathbf{R}_{wp_0}\exp_{\mathbf{R}}\left(\widehat{(\omega_{m_k} - \mathbf{b}_{g_k})}dt_0\right)\mathbf{R}^{-1}_{wp_0}\mathbf{R}_{wp_0}. \end{aligned} \tag{2.44}$$

Canceling out terms, and using quaternions to represent the rotation matrix $\mathbf{R}$, the final form of the integration is obtained:

$$\begin{aligned}\mathbf{q}_{wp_n} &= \mathbf{q}_{wp_0} \otimes \left[\mathbf{q}_{p_0p_{n-1}} \otimes \exp_{\mathbf{q}}\left(\widehat{(\omega_{m_{n-1}} - \mathbf{b}_{g_{n-1}})}dt_{n-1}\right) \otimes \mathbf{q}^{-1}{}_{p_0p_{n-1}} \otimes \ldots \otimes \exp\left(\widehat{(\omega_{m_k} - \mathbf{b}_{g_k})}dt_0\right)\right] \\ &= \mathbf{q}_{wp_0} \otimes \Delta\mathbf{q}, \end{aligned} \tag{2.45}$$

where $\Delta\mathbf{q}$ represents the integration of bias-corrected angular velocities starting from an identity orientation, and does not depend on $\mathbf{q}_{wp_0}$. The derivative of $\partial\mathbf{q}_{wpn}/\partial\mathbf{q}_{wp_0}$ can therefore be computed without the need to use the chain rule through the integration.

2.5.2 Velocity Integration Factorization

Similar to Section 2.5.1, the final integrated velocity is obtained by integrating the accelerometer measurements in the world frame from time 0 to n:

$$\begin{aligned}\mathbf{v}_{w_n} &= \mathbf{v}_{w_0} + \mathbf{a}_{w_0}dt_0 + \ldots + \mathbf{a}_{w_{n-1}}dt_{n-1} \\ &= \mathbf{v}_{w_0} + \left(\mathbf{q}_{wp_0} \otimes (\mathbf{a}_{m_0} - \mathbf{b}_{a_0}) + \mathbf{g}_w\right)dt_0 + \ldots + \left(\mathbf{q}_{wp_{n-1}} \otimes \left(\mathbf{a}_{m_{n-1}} - \mathbf{b}_{a_{n-1}}\right) + \mathbf{g}_w\right)dt_{n-1}, \end{aligned} \tag{2.46}$$

where accelerometer measurements in the IMU frame are rotated into the world frame as per Equation 2.21. It is immediately obvious that the effect of the initial velocity $\mathbf{v}_{w_0}$ is already factorized. Further inspection reveals that the gravity term is always added in the world frame, and so it can be factorized out as well:

$$\mathbf{v}_{w_n} = \mathbf{v}_{w_0} + \mathbf{g}_w \Delta t + \mathbf{q}_{wp_0} \otimes (\mathbf{a}_{m_0} - \mathbf{b}_{a_0})\, dt_0 + \ldots + \mathbf{q}_{wp_{n-1}} \otimes \left(\mathbf{a}_{m_{n-1}} - \mathbf{b}_{a_{n-1}}\right) dt_{n-1}. \tag{2.47}$$

Similar to the operation applied to Equation 2.42, the rotation matrix taking the accelerometer measurements from the IMU to world coordinates can be decomposed as $\mathbf{R}_{wp_n} = \mathbf{R}_{wp_0}\mathbf{R}_{p_0p_n}$, allowing the factorization of the initial rotation, $\mathbf{R}_{wp_0}$.

$$\begin{aligned}\mathbf{v}_{w_n} &= \mathbf{v}_{w_0} + \mathbf{g}_w \Delta t + \mathbf{q}_{wp_0} \otimes (\mathbf{a}_{m_0} - \mathbf{b}_{a_0})\, dt_0 + \ldots + \mathbf{q}_{wp_0} \otimes \mathbf{q}_{p_0p_{n-1}} \otimes \left(\mathbf{a}_{m_{n-1}} - \mathbf{b}_{a_{n-1}}\right) dt_{n-1} \\ &= \mathbf{v}_{w_0} + \mathbf{g}_w \Delta t + \mathbf{q}_{wp_0} \otimes \left[(\mathbf{a}_{m_0} - \mathbf{b}_{a_0})\, dt_0 + \ldots + \mathbf{q}_{p_0p_{n-1}} \otimes \left(\mathbf{a}_{m_{n-1}} - \mathbf{b}_{a_{n-1}}\right) dt_{n-1}\right], \end{aligned} \tag{2.48}$$

where the term in square brackets represents the integration of the bias-corrected accelerometer measurements rotated into the identity reference frame and not corrected for gravity. Referring to this term as $\Delta\mathbf{v}$, the final integrated velocity is formulated as follows

$$\mathbf{v}_{w_n} = \mathbf{v}_{w_0} + \mathbf{g}_w \Delta t + \mathbf{q}_{wp_0} \otimes \Delta\mathbf{v}.$$

2.5.3 Translation Integration Factorization

As with Sections 2.5.1 and 2.5.2, the final position is obtained by integrating velocities obtained from the integration in Section 2.5.2:

$$\mathbf{p}_{wp_n} = \mathbf{p}_{wp_0} + \mathbf{v}_{w_0} dt_0 + \ldots + \mathbf{v}_{w_{n-1}} dt_{n-1}. \tag{2.49}$$

The velocity terms can be substituted with the integrated accelerometer measurements as per Section 2.5.2:

$$\begin{aligned}\mathbf{p}_{wp_n} &= \mathbf{p}_{wp_0} + \mathbf{v}_{w_0} dt_0 + (\mathbf{v}_{w_0} + \mathbf{a}_{w_0} dt_0)\, dt_1 + \ldots + \left(\mathbf{v}_{w_0} + \mathbf{a}_{w_0} dt_0 + \ldots + \mathbf{a}_{w_{n-2}} dt_{n-2}\right) dt_{n-1} \\ &= \mathbf{p}_{wp_0} + \mathbf{v}_{w_0} \Delta t + (\mathbf{a}_{w_0} dt_0)\, dt_1 + \ldots + \left(\mathbf{a}_{w_0} dt_0 + \ldots + \mathbf{a}_{w_{n-2}} dt_{n-2}\right) dt_{n-1}, \end{aligned} \tag{2.50}$$

where it is immediately observed that the initial velocity ($\mathbf{v}_{w_0}$) can be factored out since $\mathbf{v}_{w_0}dt_0 + \mathbf{v}_{w_0}dt_1 + \ldots + \mathbf{v}_{w_0}dt_n = \mathbf{v}_{w_0}\Delta t$. The accelerometer measurements in the world frame ($\mathbf{a}_w$) can be replaced by the IMU frame measurements as per Section 2.5.2:

$$\begin{aligned}\mathbf{p}_{wp_n} = \mathbf{p}_{wp_0} + \mathbf{v}_{w_0}\Delta t + \left(\mathbf{q}_{wp_0} \otimes (\mathbf{a}_{m_0} - \mathbf{b}_{a_0}) + \mathbf{g}_w\right) dt_0 dt_1 + \ldots \\ + \left(\left(\mathbf{q}_{wp_0} \otimes (\mathbf{a}_{m_0} - \mathbf{b}_{a_0}) + \mathbf{g}_w\right) dt_0 + \ldots + \left(\mathbf{q}_{wp_{n-1}} \otimes \left(\mathbf{a}_{m_{n-2}} - \mathbf{b}_{a_{n-2}}\right) + \mathbf{g}_w\right) dt_{n-2}\right) dt_{n-1}.\end{aligned} \tag{2.51}$$

Since the gravity term is constant, and is double integrated in the global coordinate system, it can be replaced by its closed form:

$$\begin{aligned}\mathbf{p}_{wp_n} = \mathbf{p}_{wp_0} + \mathbf{v}_{w_0}\Delta t + \frac{1}{2}\mathbf{g}_w\Delta t^2 + \left(\mathbf{q}_{wp_0} \otimes (\mathbf{a}_{m_0} - \mathbf{b}_{a_0})\right) dt_0 dt_1 + \ldots \\ + \left(\left(\mathbf{q}_{wp_0} \otimes (\mathbf{a}_{m_0} - \mathbf{b}_{a_0})\right) dt_0 + \ldots + \left(\mathbf{q}_{wp_{n-1}} \otimes \left(\mathbf{a}_{m_{n-2}} - \mathbf{b}_{a_{n-2}}\right)\right) dt_{n-2}\right) dt_{n-1}.\end{aligned} \tag{2.52}$$

As per section 2.5.1, the rotation term $\mathbf{R}_{wp_n}$ which is used to rotate the accelerometer measurements from the IMU to the world frame can be decomposed as follows: $\mathbf{R}_{wp_n} = \mathbf{R}_{wp_0}\mathbf{R}_{p_0p_n}$. Using this Equation 2.52 can be re-written as follows:

$$\begin{aligned}\mathbf{p}_{wp_n} = \mathbf{p}_{wp_0} + \mathbf{v}_{w_0}\Delta t + \frac{1}{2}\mathbf{g}_w\Delta t^2 \\ + \mathbf{q}_{wp_0} \otimes \left[(\mathbf{a}_{m_0} - \mathbf{b}_{a_0}) dt_0 dt_1 + \ldots + \left((\mathbf{a}_{m_0} - \mathbf{b}_{a_0}) dt_0 + \ldots + \left(\mathbf{q}_{p_0p_{n-1}} \otimes \left(\mathbf{a}_{m_{n-2}} - \mathbf{b}_{a_{n-2}}\right)\right) dt_{n-2}\right) dt_{n-1}\right],\end{aligned} \tag{2.53}$$

where the term in the square brackets represents the double integration of the accelerometer measurements rotated into the identity reference frame and without gravity compensation. Referring to this term as $\Delta\mathbf{p}$, the final integrated velocity is formulated as follows:

$$\mathbf{p}_{wp_n} = \mathbf{p}_{wp_0} + \mathbf{v}_{w_0}\Delta t + \frac{1}{2}\mathbf{g}_w\Delta t^2 + \mathbf{q}_{wp_0} \otimes \Delta\mathbf{p}. \tag{2.54}$$

2.6 Summary

Adaptive asynchronous conditioning (AAC) is a novel solution to real-time visual-inertial SLAM. The approach automatically scales and focuses computation to capture an accurate approximation to the batch MLE solution in both the keyframe poses and the map structure, and avoids the downsides associated with marginalization, such as incorrect linearization and inconsistency. Further, AAC avoids the computational difficulties associated with carrying prior distributions, such as the need to compute global optimizations at loop closure.

The proposed method offers a natural 'front-end' while simultaneously allowing larger portions of the problem to influence the solution. It is thus able to produce estimates in real-time, and also re-estimate past parameters in the presence of new information – an ability which is useful for self calibration, during degenerate motions, or when bias and gravity are poorly observed.

Chapter 3

Constant-Time Self Calibration

3.1 Introduction

In this chapter, a novel extensible probabilistic framework is introduced allowing self-calibration of sensor and robot parameters. Self-calibration enables initial and continuous estimation of calibration parameters without using specialized input data such as calibration targets or specific motions. As discussed in Chapter 1, accurate calibration of intrinsic (e.g. focal length) and extrinsic (e.g. translation between IMU and camera reference frames) is crucial to acheive accurate visual or visual-inertial SLAM, since all residuals that form the constraints in the optimization utilize these calibration parameters. Similar to visual and visual-inertial SLAM problem, the gold-standard for the calibration problem is the batch solution: using all measurements which are available to the system. This presents an ever-growing computational requirement. Since constant-time solutions are required for real-time operation, methods which approximate the batch solution in constant-time are desired. The framework presented in this chapter are related to those in Chapter 2 in that marginalization is eschewed due to the same downsides previously discussed. In its stead, the framework attempts to collect a continuously improving collection of data supporting the calibration parameters of interest, while maintaining a constant computational complexity. In this way, the framework is able to continuously search for and incorporate useful data for the purpose of calibration. The framework has been shown to perform well for the purpose of intrinsic camera calibration, but it is agnostic to the particular calibration parameters desired, as long as constraints can be made derived input data in an optimization setting to estimate the parameters. As such,

it can also be used to calibrate parameters useful for visual-inertial SLAM, such as the extrinsic transformation between the IMU and camera reference frames.

For both mobile and service robots, images obtained from cameras are one of the primary sensor modalities used in localization, mapping, object detection and planning. However an accurate calibration of the camera is required in order to utilize the images for metric estimation. The experiments in Chapter 2 were all performed with offline calibration, where bespoke calibration targets and custom routines were used to calibrate the intrinsic and extrinsic parameters before commencing the experiments. While offline calibration provides an initial estimate to the calibration parameters, it is not always feasible and convenient to calibrate cameras using pre-fabricated calibration targets. Calibration parameters could also change during storage or between uses, requiring recalibration. A true "power on and go" solution would both enable a simple camera calibration solution for mobile robots, as well as removing a major barrier of entry for students and hobbyists wishing to incorporate metric vision in their robotic projects. Camera self-calibration in the SLAM framework is the process of estimating the intrinsic parameters of a camera considering solely a sequence of images, without any assumptions as to the content of the scene, whilst simultaneously estimating the map and camera location. The calibration parameters typically consist of the focal length, principal point and several distortion parameters. An online implementation of self-calibration would recursively estimate these parameters as new images are obtained, attempting to provide the best estimates considering all available information. Traditionally, calibration is performed offline using images of a known calibration target, or as part of a large offline batch optimization. Once calibrated, the parameters are fixed for the lifetime of the robot. This approach presents many drawbacks, such as the vulnerability to any changes in the camera parameters, the inconvenience of producing and imaging a known calibration target, along with the bespoke software written explicitly to detect it. Apart from continually improving the calibration parameter estimates as new images become available, online self-calibration also paves the way for robust change detection and estimation, dealing with situations where the physical calibration parameters intentionally or unintentionally change. For mobile robotics, easy self-calibration and hence "power

on and go" functionality could for instance allow naive users, such as young students, to experiment with visual simultaneous localization and mapping algorithms without first becoming experts in camera calibration.

Due to these advantages, online self-calibration for visual and visual-inertial systems has been of interest to research recently [73, 13, 51]. However, the estimation of calibration parameters in constant-time considering all present and previous data presents challenges which have so far prevented online self-calibration from becoming commonplace. Filtering methods have been the main approach in cases where the estimation of parameters requires processing all present and past information. The estimation is made constant-time by rolling information from past image measurements into a parametric distribution, or prior. Ideally, the prior would incorporate the influence of the now-absent parameters and measurements on the smaller, active set of parameters, enabling a constant-time estimation of these active parameters that considers the entire trajectory up to the present. Unfortunately, filtering methods present some drawbacks that are especially critical for the case of self-calibration. Due to the nonlinearity of camera models, linearized parametric distributions often cannot represent past measurements without introducing inconsistencies in the estimation. In the case of SLAM, it is well known that filtering methods can be inconsistent. Recent work has focused on how to tackle these inconsistencies [59, 32], as their presence can cause overconfidence in the estimated parameters, an especially undesirable situation in the case of self-calibration. Even when complex parametric models are used, early linearization used to obtain the parametric distributions can lead to inaccuracies in the prior which lead to inconsistency. Furthermore, updates to the calibration parameters induce large and often highly non-linear changes to the parameter estimates, further exasperating the effects of early linearization.

These issues motivate the work presented in this paper, which involves estimating the calibration parameters online, by considering only the best segments of the trajectory, without a prior distribution. The motivation behind the work is the realization that not all segments in a SLAM sequence are equal as far as their information content regarding the calibration parameters. This can be easily conceptualized due to the existence of degenerate configurations which provide no in-

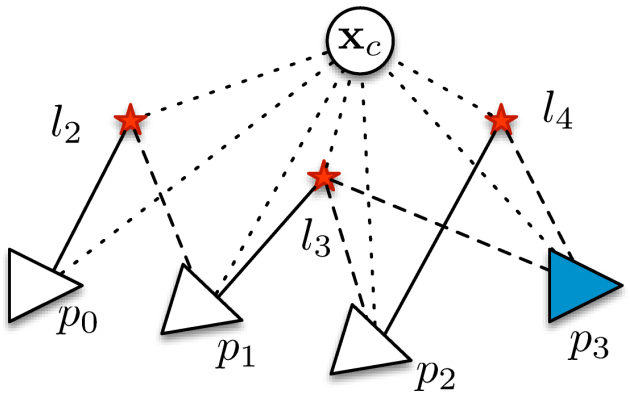


Figure 3.1: Graphical model representing the self-calibrating SLAM problem. Edges between the calibration parameters $\mathbf{x}_c$ and all other parameters make the problem difficult to factor.

formation [112]. If the best segments in the trajectory are considered, a constant time estimate for the calibration parameters can still be obtained. Moreover, due to the lack of a prior, inconsistencies are caused only by modeling and measurement uncertainty errors, rather than a product of the framework, easing their mitigation. Due to the self-selection of these segments, special attention is paid to ensuring unbiased estimates, as not all measurements are considered. While the proposed system is only concerned with the calibration of camera intrinsics, it can easily be extended to camera extrinsics and multi-sensor self-calibration.

3.2 Priority Queue Based Self Calibration

The self-calibrating SLAM problem can be represented as a graphical model as per Fig. 3.1. It is observed that the calibration parameters $\mathbf{x}_c$ are linked to all pose and landmark parameters through measurements, making a direct solution infeasible in an online setting. However, the amount of information obtained about the calibration parameters varies from each measurement. Depending on the motion of the camera, and the structure of the scene, measurements may add little to the observability of the calibration parameters. In certain degenerate cases of camera motion, measurements may indeed provide no information.

To obtain a constant time estimate for the calibration parameters, the best segments of the trajectory are stored in a priority queue as follows: When a new image is received, the set of

measurements obtained from it and the additional measurements obtained from a fixed-length of previous images are used to estimate the camera pose and landmark parameters in the window, as well as the camera calibration parameters. This window is then scored based on the uncertainty of the calibration parameters, with a low uncertainty signifying a high score. This score is then compared against each window already in the priority queue. If the score is better than the worst score in the priority queue, the worst window in the priority queue is replaced with the new candidate window, as shown in Fig. 3.2. Once such an update takes place, all the windows in the priority queue are used to jointly re-estimate the camera calibration parameters. Since the maximum number of windows in the priority queue is fixed and pre-determined, the calibration parameters are estimated in constant-time in the event of a priority update.

3.2.1 Optimization Formulation

The calibration parameters are estimated alongside the camera pose and landmark parameters in a non-linear maximum likelihood estimation framework. This framework is also used to extract the uncertainty of the calibration parameters. Measurements are formed by tracking salient points of the image across multiple frames. Each measurement is modeled as a projection of a landmark parameterized in inverse depth [75] into an image, to form a minimal landmark representation. Note: parameterizing the landmark in this way is consistent with the maximum likelihood estimate, as measurements are formed by tracking an initial reference patch, which remains along the initial reference ray. The projected pixel coordinate $\mathbf{p}_2$ of a landmark in the current frame is formulated via the transfer function $\mathcal{W}$ as

$$\begin{aligned}\mathbf{p}_2 &= \mathcal{W}\left(\mathbf{p}_1, \mathbf{T}_{21}, \rho\right) \\ &= \mathcal{P}\left(\pi_{3d}\left(\mathbf{T}_{wc_2}^{-1}\mathbf{T}_{wc_1}\left[\mathcal{P}^{-1}\left(\mathbf{p}_1, \mathbf{x}_c\right); \rho\right]\right), \mathbf{x}_c\right)\end{aligned} \tag{3.1}$$

The transferred pixel coordinate is obtained by first back-projecting a reference pixel coordinate obtained via harris corner detection, $\mathbf{p}_1$, given the back-projection function $\mathcal{P}^{-1}$, to obtain the

reference ray. This ray is then homogenized given the inverse depth parameter ρ. The resulting 4d homogeneous vector is transformed into the current camera reference frame by the homogeneous transformation $\mathbf{T}_{wc_2}^{-1}\mathbf{T}_{wc_1}$ where $\mathbf{T}_{wc_n} \in \text{SE3}$ is the 4×4 transformation matrix from coordinates of frame n to world coordinates. The result is de-homogenized by π_{3d} and projected into the current image by the projection function $\mathcal{P}$. More specifically, $\mathcal{P}^{-1}(\mathbf{p}_1, \mathbf{x}_c) \in \mathbb{R}^3$ is the 2d to 3d camera back-projection function which outputs a unit ray given the reference 2d pixel coordinates $\mathbf{p}_1$ and the camera calibration parameters $\mathbf{x}_c$, $\mathcal{P}(\mathbf{u}, \mathbf{x}_c) \in \mathbb{R}^2$ is the 3d to 2d projection function which outputs a 2d pixel coordinate given a de-homogenized 3d point and $\pi_{3d}()$ is the 3d de-homogenization function defined as:

$$\pi_{3d}\left(\begin{bmatrix} x & y & z & w \end{bmatrix}^T\right) = \begin{bmatrix} x/w & y/w & z/w \end{bmatrix}^T$$

As an example, in the case of the pinhole camera model, the calibration parameters $\mathbf{x}_c = \begin{bmatrix} f_x & f_y & c_x & c_y \end{bmatrix}$ are comprised of the focal length and principal point parameters in x and y, and the projection function $\mathcal{P}$ is defined as $\mathcal{P}(\mathbf{u}, \mathbf{x}_c) = \pi_{2d}(\mathbf{K}(\mathbf{x}_c)\,\mathbf{u})$ where $\mathbf{K}(\mathbf{x}_c)$ forms the camera projection matrix given the intrinsic parameters, $\mathbf{u} \in \mathbb{R}^3$ is a 3d point location, and π_{2d} is the 2d de-homogenization function defined as $\pi_{2d}\left(\begin{bmatrix} x & y & z \end{bmatrix}^T\right) = \begin{bmatrix} x/z & y/z \end{bmatrix}^T$. Similarly, the back projection function $\mathcal{P}^{-1}$ for a pinhole camera is defined as $\mathbf{u}_1 = \mathcal{P}^{-1}(\mathbf{p}_1, \mathbf{x}_c) = \mathbf{K}^{-1}(\mathbf{x}_c)\begin{bmatrix} \mathbf{p}_1 \\ 1 \end{bmatrix}$.

Given the aforementioned residual model, the state space vector of the optimization is defined as

$$\mathbf{x} = \begin{bmatrix} \{\mathbf{x}_{wi} : i = 1, \ldots, n\} & \{\rho_j : j = 1, \ldots, m\} & \mathbf{x}_c \end{bmatrix}^T$$

where $\mathbf{x}_{wi} \in \text{se3}$ is the 6d tangent space representation of the update to transformation $\mathbf{T}_{wi}$, ρ_j is the inverse depth parameter for landmark j, and $\mathbf{x}_c$ is the vector of calibration parameters. Given the projection in (3.1), the residual for the measurement of the jth landmark, with reference (first measurement) frame k, in the ith frame is formulated as

$$\mathbf{r}_{ij} = \mathbf{z}_{ij} - \mathcal{W}\left(\mathbf{p}_j, \mathbf{T}_{wc_i}^{-1}\mathbf{T}_{wc_k}, \rho_j\right) \tag{3.2}$$

where $z_{ij} \in \mathbb{R}^2$ is the pixel coordinate measurement of the jth landmark in the ith frame, $\mathbf{T}_{wc_k}$ is the transformation from the reference frame coordinates for landmark j to world coordinates, $\mathbf{T}_{wc_i}$ is the transformation from the coordinates of the measurement frame i to world coordinates, $\mathbf{p}_j$ is the reference pixel coordinate for landmark j, and ρ_j is the inverse depth parameter for landmark j. The total error minimized in the optimization is formulated as

$$e = \sum_{i=1}^{n} \sum_{j=1}^{m} \|\mathbf{r}_{ij}\|^2_{\Sigma_{ij}} \tag{3.3}$$

where the notation $\|\mathrm{x}\|^2_{\Sigma_{ij}}$ signifies the mahalanobis distance given the measurement uncertainty $\Sigma_{ij} \in \mathbb{R}^{2\times 2}$.

The problem is then solved by iteratively updating the state vector $\mathbf{x}$ in a dog-leg trust region minimization framework[93]. The dog-leg update consists of a mixture of the Gauss-Newton and steepest descent solutions denoted as δ_{GN} and δ_{SD} respectively which are formulated as

$$\delta_{GN} = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\mathbf{g} \quad \delta_{SD} = \frac{\mathbf{g^T g}}{\mathbf{g}^T\mathbf{J}^T\mathbf{J}\mathbf{g}}\mathbf{g} = \frac{\|\mathbf{g}\|}{\|\mathbf{J}\mathbf{g}\|}\mathbf{g}$$

where $\mathbf{g} = \mathbf{J}^T\mathbf{r}$, $\mathbf{J} = \mathbf{W}^{\frac{1}{2}}\frac{\partial \mathbf{r}}{\partial \mathbf{x}}$ is the Jacobian matrix of the residual vector with respect to the state vector, $\mathbf{r} = [r_1 \dots r_n]^T$ is the vector of residuals formed as per (3.2), and $\mathbf{W}$ is the residual weight matrix. The weight matrix $\mathbf{W}$ is a combination of residual weights given by the inverse of the measurement covariance Σ_{ij} from (3.3), as well as a re-weighted Huber norm for outlier rejection. Note that the Jacobian has been represented in standard form, which includes the square root of the weight matrix $\mathbf{W}$. This is necessary for the correct calculation of the steepest descent update δ_{SD}. The Gauss-Newton delta is obtained by solving for δ_{SD} using the Cholesky factorization of the reduced-camera matrix, obtained by the Schur complement trick[107][3]. The optimization is iterated until either the error or parameter change is smaller than a specific threshold.

Since the MLE framework presented above does not make use of a prior, the system will exhibit a number of unconstrained degrees of freedom, manifesting as null spaces in the system Hessian. The monocular SLAM problem in particular exhibits 7 null spaces. The first 6 null spaces are due to the unobservability of the global translation and rotation, and are handled by removing

the parameters for the first pose ($\mathbf{x}_{w0}$) from the optimization. The seventh nullspace is due to scale unobservability and is handled by removing a single inverse depth parameter ρ_j from the optimization. The landmark j for which the inverse depth parameter is removed is chosen to have the largest number of measurements, and therefore be well estimated, to ensure minimal impact on the optimization.

3.2.2 Priority Queue

The proposed method requires each candidate segment of the trajectory to be scored depending on the observability of the camera parameters during the segment. This enables a comparison between it and the segments already in the priority queue. The proposed score is based on the entropy of the distribution of the calibration parameters for a given segment, with the ultimate goal of obtaining the segments which have the least uncertainty about their individual estimates for the calibration parameters. Given the assumption of Gaussian distributions, a posterior can be computed over the calibration parameters by inverting the Fisher information matrix $\mathcal{I}$ and extracting the submatrix associated with the parameters $\mathbf{x}_c$:

$$\Sigma_{\mathbf{x}_c} = \mathcal{I}\left[\mathbf{i}_{\mathbf{x}_c}, \mathbf{i}_{\mathbf{x}_c}\right] = \left(\mathbf{J}^T\mathbf{J}\right)^{-1}\left[\mathbf{i}_{\mathbf{x}_c}, \mathbf{i}_{\mathbf{x}_c}\right] \tag{3.4}$$

where the notation $[\mathbf{x}, \mathbf{y}]$ denotes a submatrix extracted from rows specified in the vector $\mathbf{x}$ and columns specified in the vector $\mathbf{y}$, $\mathbf{J}$ is the Jacobian matrix as per section 3.2.1, and $\mathbf{i}_{\mathbf{x}_c} \in \mathbb{R}^m$ is a vector containing the indices of the m camera calibration parameters in the state vector $\mathbf{x}$. Since the calibration parameters are not of the same scale, an entropy calculated based on $\Sigma_{\mathbf{X_c}}$ will be skewed towards variables with larger variances. In order to correct for this skew, the posterior covariance matrix is normalized as follows:

$$\Sigma'_{ij} = \frac{1}{\sqrt{\sigma_i}\sqrt{\sigma_j}}\Sigma_{ij}$$

where σ_i and σ_j are the expected true variances of the ith and jth camera calibration parameters respectively. These quantities should encode the underlying differences between the variances of

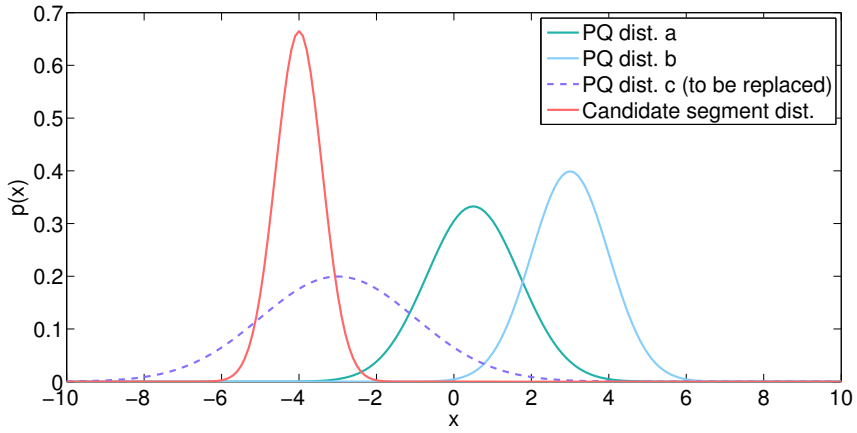


Figure 3.2: One dimensional example of the priority queue update operation. Segment c in the priority queue (denoted by the dashed line) has the highest entropy and will be replaced with the candidate segment which has a lower entropy.

the parameters given their units and range, and are obtained from the posterior $\Sigma_{\mathbf{x}_c}$ obtained in a batch solution over a large sample trajectory. For the experiments discussed in this chapter, a sample trajectory of 2000 frames was used. The quantity $\Sigma'_{\mathbf{X_c}}$ is similar to the posterior correlation matrix, but with variances from a different, sample trajectory used as the normalizing factors. Since the Jacobian matrix $\mathbf{J}$ in (3.4) contains terms for all poses, landmarks, and the calibration parameters, the inverse can be very costly. Therefore, in order to obtain the marginal distribution, the method outlined in [19] is used:

$$\mathcal{I}\Sigma = \mathbf{I} \rightarrow \mathcal{I}\Sigma_i = \mathbf{e}_i \tag{3.5}$$

where the equality in (3.5) is due to the duality between the information matrix $\mathcal{I}$ and the covariance Σ, and consequently the ith column of the covariance Σ_i can form an equality with the ith unit vector. Equation (3.5) can be solved using the Cholesky factorization of $\mathcal{I}$, requiring a forward/backward substitution per column of the covariance. As the calibration parameters $\mathbf{x}_c$ are at the end of the state vector, the number of substitutions per covariance column can be limited to the dimensionality of $\mathbf{x}_c$[19]. Given the normalized covariance of the calibration parameters, the entropy of the distribution is then given by $h = \frac{1}{2} ln \left|2\pi e \Sigma'_{\mathbf{x}_c}\right|$ where the bars denote the matrix determinant. The priority queue update condition is therefore:

$$h_c < \alpha \max \{h_i : i = 1, \ldots, n\}$$

where h_c is the entropy of the candidate segment, $\{h_i : i = 1, \ldots, n\}$ is the set of the entropies of the segments already in the priority queue, and $\alpha = 0.95$ is a heuristic to ensure at least a 5% reduction of entropy for an update operation. If this condition is met, the candidate segment is replaced with the segment with the largest entropy. If a candidate segment overlaps with a segment already in the priority queue, the update condition is only checked against the overlapping segment. If the condition is met, the overlapping segments are swapped. This is to ensure that no overlapping segments are present in the priority queue, as the optimization would then double-count the overlapping measurements. Once a swap takes place, the optimization is run again to minimize the error in (3.3) jointly over all segments of the priority queue. The resulting calibration parameters are then used to continue the SLAM estimation. The distribution of the calibration parameters of the priority queue can then be extracted from the converged Hessian of the problem. It must be noted that the nullspace regularization discussed in section 3.2.1 must be applied to each segment in the joint priority queue optimization, as the segments do not overlap and therefore each exhibit the aforementioned 7 nullspaces.

It is important that the priority queue update condition not be biased in any way, as rather than solely optimizing over the calibration parameters, the measurements over which the optimization takes place are actively being selected. Any selection criteria which attempts to reduce the entropy of the priority queue distribution, or increase the information gained as a result of the update, risks biasing the solution towards the current estimates for $\mathbf{x}_c$, given the assumptions of Gaussian distribution. As an example, the true distribution for a particular parameter may be multi-modal, with measurements first observing one mode, and then the other. In a batch solution with a Gaussian distribution assumption, the observation of the second mode would in fact increase the entropy of the distribution over the parameter, as the Gaussian assumption is incorrect, and cannot explain the underlying distribution. It is important that this behavior also be replicated in the priority queue solution. The proposed scoring solution is therefore solely based on the entropy

of the posterior distribution over the segment itself, not how it relates to the posterior distribution of the priority queue.

3.2.3 Online Self-Calibration

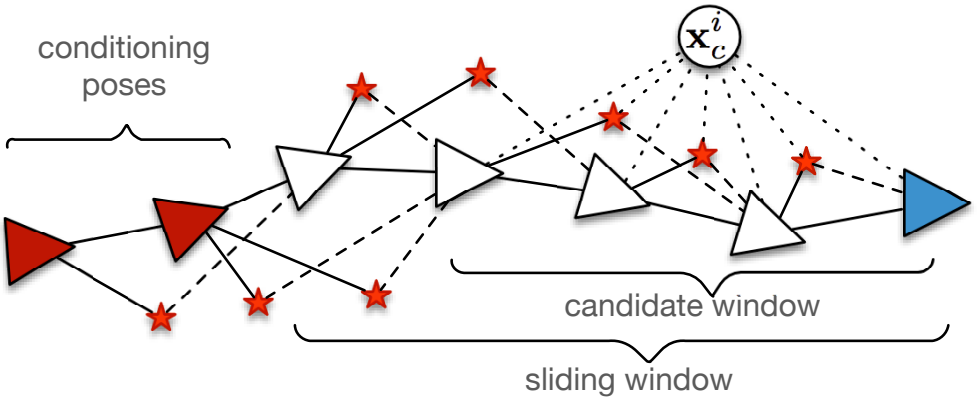


Figure 3.3: The sliding and candidate windows, on a sample segment of the trajectory. The sliding window is conditioned on poses outside the window based on co-visible landmarks, whereas the candidate window is considered separately. $\mathbf{x}_c^i$ is the calibration parameter vector estimated by considering only the measurements in the ith candidate window. The sliding, and candidate windows can be processed in parallel.

The online implementation of the proposed method consists of two windowed optimizations. The first is a conditioning sliding window[68] estimator, which does not optimize the calibration parameters. It is tasked with estimating poses and landmarks within an active window, while being conditioned on past poses. The secondary window estimates poses, landmarks as well as the calibration parameters, and is used to compute the marginals $\Sigma_{\mathbf{x}_c}$. This optimization does not condition on information outside the window. The marginals are then used in the update condition of the priority queue. These two estimators are separated since the observability of the calibration parameters is not guaranteed over the candidate window. Poorly observed calibration parameters could then affect the quality of the pose and landmark estimates, affecting the reliability of navigation and localization.

If an update to the priority queue is carried out, an optimization over the entire queue is performed resulting in new estimates for the calibration parameters. These new estimates are then

fed back to the sliding window estimator. Special attention has been paid to the startup sequence where no prior information is assumed over the calibration parameters, which are initialized to common values as per section 3.3. To handle this case, a batch optimization is run which includes the calibration parameters, until the entropy of the posterior over the calibration parameters falls below a preset heuristic. This batch optimization is run in conjunction with the priority queue update procedure. Once the batch entropy is lower than a specific heuristic, further estimation is handled by the windowed optimization shown in Fig. 3.3, and the priority queue is used to further update the estimates of the calibration parameters.

A keyframing system [68] is implemented in order to increase performance, and the information content of each pose regarding the calibration parameters. As each new image is received, a number of heuristics are used to decide whether or not a keyframe should be created. These heuristics are formed on the distance and angle between the current frame and the previous keyframe, as well as on the percentage of landmarks successfully tracked. The result is that no new keyframes are placed if images from a semi-stationary camera are received. A secondary result is that keyframes are only placed when there is sufficient excitation of the camera to trigger one of the heuristics. Keyframing does not completely guarantee observability of the camera parameters due to the existence of degenerate motions which still trigger new keyframes. However, a number of cases of degenerate motion will be successfully avoided, such as the stationary camera case.

3.3 Experiments

The proposed system was validated with real data to evaluate its performance and convergence characteristics. In all cases, 2d feature tracks were obtained from the images and used as measurements in the aforementioned optimizations. The experimental datasets were captured with a wide-angle lens and calibrated using the FOV model [14]. The calibration parameters were initialized to common values as follows: the x and y focal lengths were set equally to $90°$, the x and y principal point parameters were set to half the image width and height respectively, and the distortion parameter w of the FOV model was set to represent an ideal fisheye lens ($w = 1.0$).

As discussed in section 3.2.3, an initial batch optimization comprising all poses, landmarks and calibration parameters is run until its entropy h is below a certain threshold, at which point the calibration parameter estimation is handed over to the priority queue. This batch estimation stage lasts for approximately 10 keyframes. A large convergence basin was observed for both the focal length and the distortion parameter w. Focal lengths corresponding to FOVs between 30° and 110° combined with values of w between 0.5 and 1.0 were observed to converge, while outside values generally diverged due to a combination of tracking and batch estimation failure.

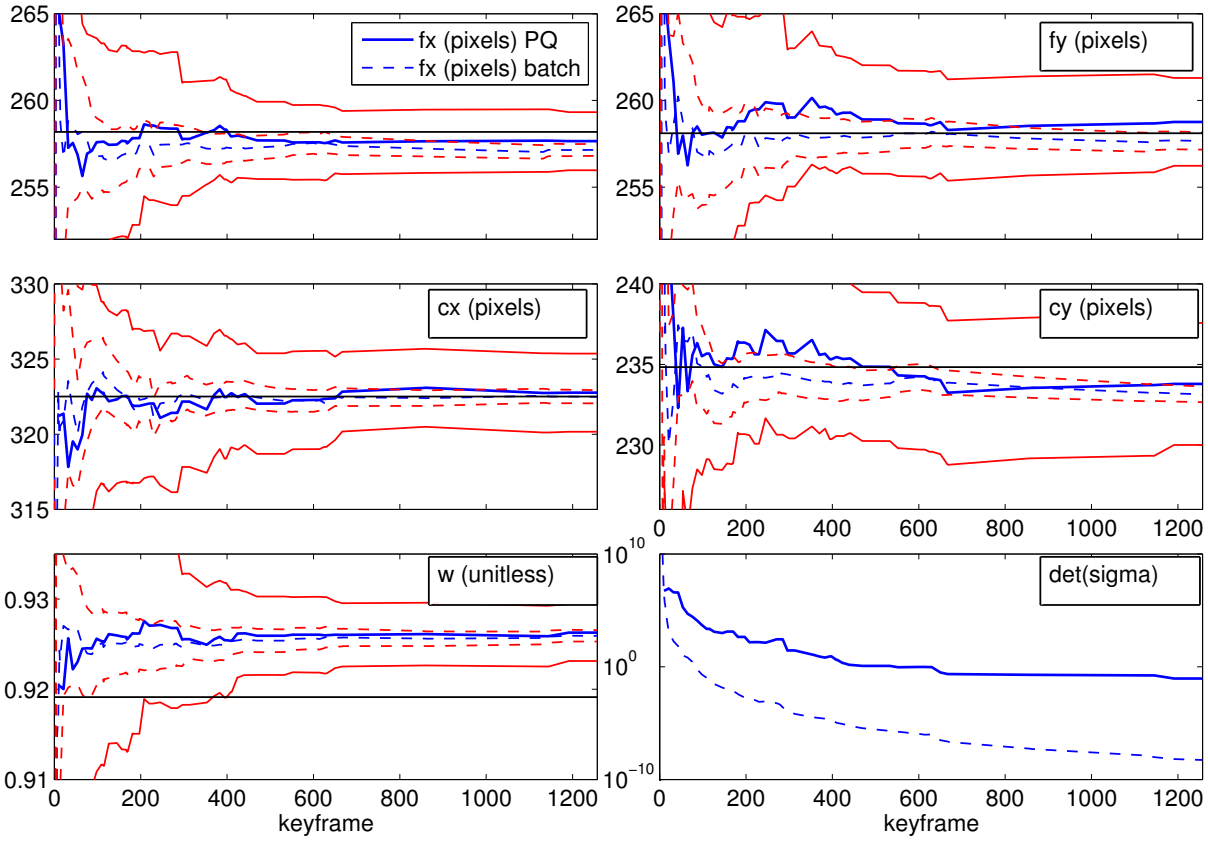


Figure 3.4: Results of a loop dataset spanning 5300 frames and 1200 keyframes, reconstructed by tracking 128 features on average. The priority queue consisted of 10 segments each with 10 keyframes. Solid and dashed red lines represent 3 sigma bounds for the priority queue and batch solutions respectively. f_x, f_y, c_x, and c_y are the x and y focal length and principal point values respectively, and w is the FOV model distortion parameter. $\det(\Sigma)$ plots the determinant of the calibration parameter covariance Σ'_{ij}. Solid black lines represent offline calibration values. The priority queue and batch statistics are only calculated when the update condition is met and a swap takes place in the priority queue.

Fig. 3.4 shows the results of the system running on a loop dataset, captured with a wide-angle lens camera. It can be seen after the first 10 keyframes in which the batch solution is active, the priority queue successfully tracks the mean calibration parameter values obtained by the batch solution, albeit with a higher uncertainty. This is expected for the priority queue, as it is computed using a much smaller subset of the available measurements. The key aspect of the system is that is able to include sections which significantly increase the observability of the parameters, as can be seen in the determinant plot in Fig. 3.4, which plots the determinant of the normalized calibration parameter covariance matrix Σ'_{ij}. At keyframes 300 and 600, the batch solution determinant dips, signaling a reduction in the total uncertainty of the calibration parameters. The priority queue solution is able to mirror these reductions in uncertainty, by swapping in the relevant segments. This is seen more prominently in the indoor dataset results shown in Fig. 3.5, where at keyframe 200 a large reduction in uncertainty is visible in both the batch and priority queue solutions.

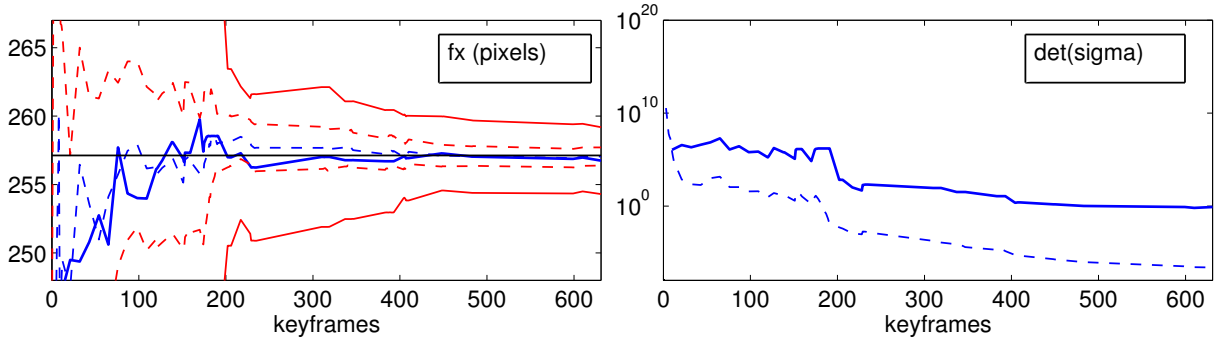


Figure 3.5: Focal length and covariance determinant for an indoor corridor sequence. The capture, sequence and priority queue were performed identical to the results in Fig. 3.4.

The effects of the size of the priority queue are demonstrated in Fig. 3.7, where it is evident that priority queues which consist of more segments tend to better match the batch solution. However, even when using just 5 segments of 10 keyframes each, accurate estimations are observed compared to both the batch and offline estimations. As expected, using a larger number of segments also results in a lower uncertainty for the priority queue estimate. Higher volatility is also observed when fewer segments are present in the priority queue as swapping any individual segment can have

a higher impact on the overall priority queue solution.

Discrepancies are observed between some estimates from the priority queue and the offline calibration values (such as w in Fig. 3.4). This discrepancy can be caused by a number of factors, such as the quality of the offline calibration, feature tracking, the given uncertainty of the visual measurements, and lack of observability over the parameters. However, it is observed that in all cases both the batch and priority queue solution deviate together, and the priority queue closely matches the batch solution. A particular failure case for the system is if each new image adds an equal amount of information to the parameters. In this case, no segment will ever be swapped in the priority queue, as all segments have equal score. However it has been observed that with real and synthetic data, this is an unrealistic case. The determinant based scoring system can also exhibit a failure case, if the uncertainty along a single parameter is exceptionally small compared to the rest due to observability issues. This could artificially inflate the score of a segment, even though it adds little information to most other parameters. It has also been observed that with real and synthetic data that this is not the general case, however depending on the particular camera motion it could transpire, and may warrant a more discerning scoring system.

It must be noted that although the calibration parameters are re-estimated as the priority queue is updated, the past portions of the trajectory are not. This introduces error in the global camera pose estimate, while local estimates remain optimal. Fig. 3.6 shows the estimated camera poses for pre-calibrated and self-calibrating runs for a sample 200m trajectory with a fixed window size of 10 keyframes. It can be seen that as expected, due to scale ambiguity, the loop estimate does not close correctly. However it is also observed that without any specific measure to address global optimality, the two solutions remain close, with a mean error of 1.76% of the distance traveled between them. This signifies that the calibration is estimated accurately and quickly enough to ensure estimation close to the pre-calibrated trajectory. If explicit global optimality is desired, a scheme such as Asynchronous Adaptive Conditioning [44] can be used to match the optimal global solution adaptively in constant-time. A single-threaded, synchronous implementation of the proposed method runs at an average of 27fps over the course of the 200m trajectory in Fig. 3.6 on

a 2.6Ghz Intel i7. An asynchronous implementation would significantly improve this as the sliding window estimation would not be halted by calibration related estimation operations.

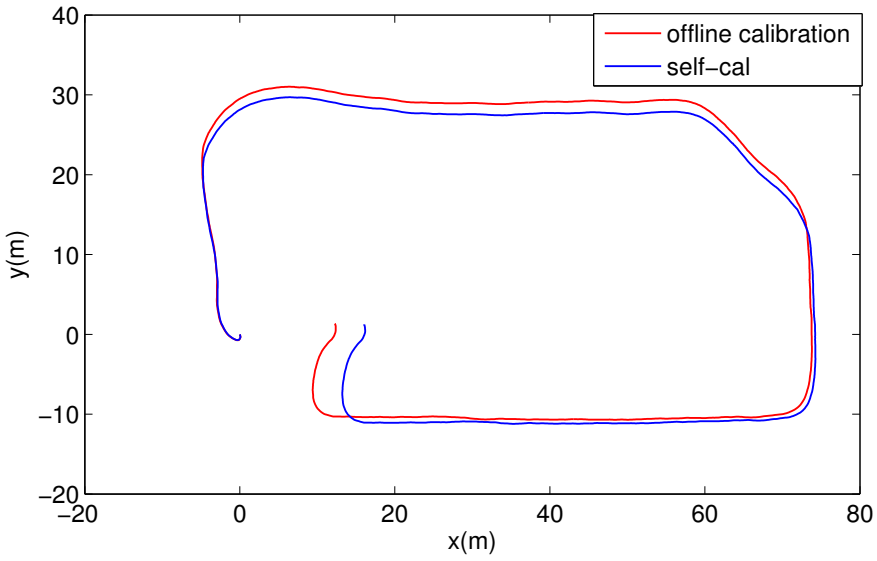


Figure 3.6: Comparison between the estimated poses on a 200m loop trajectory for pre-calibrated and self-calibrating estimators. The loop reconstruction does not close due to monocular scale ambiguity. The mean error between the two trajectories is 1.76% of the traveled distance. The self-calibrating estimator's initial calibration parameters were set as per section 3.3.

3.4 Summary

A framework has been presented which enables live and constant-time self-calibration in a SLAM setting. The performance of the system has been experimentally validated, where it has been shown to perform successful SLAM estimation with no initial information about the parameters of the given camera model. The system runs in real-time and is able to closely match the batch calibration solution for the trajectory. Furthermore, as no parametric prior is assumed, no inconsistencies are introduced as a result of early marginalization. Particular attention has been paid to select an update condition which does not artificially bias the priority queue solution towards current estimates. The system automatically identifies measurement sequences that are useful for calibration and saves these sequences in a priority queue. Results show that the mean of the priority queue tracks that of the batch solution, although the uncertainty of the priority

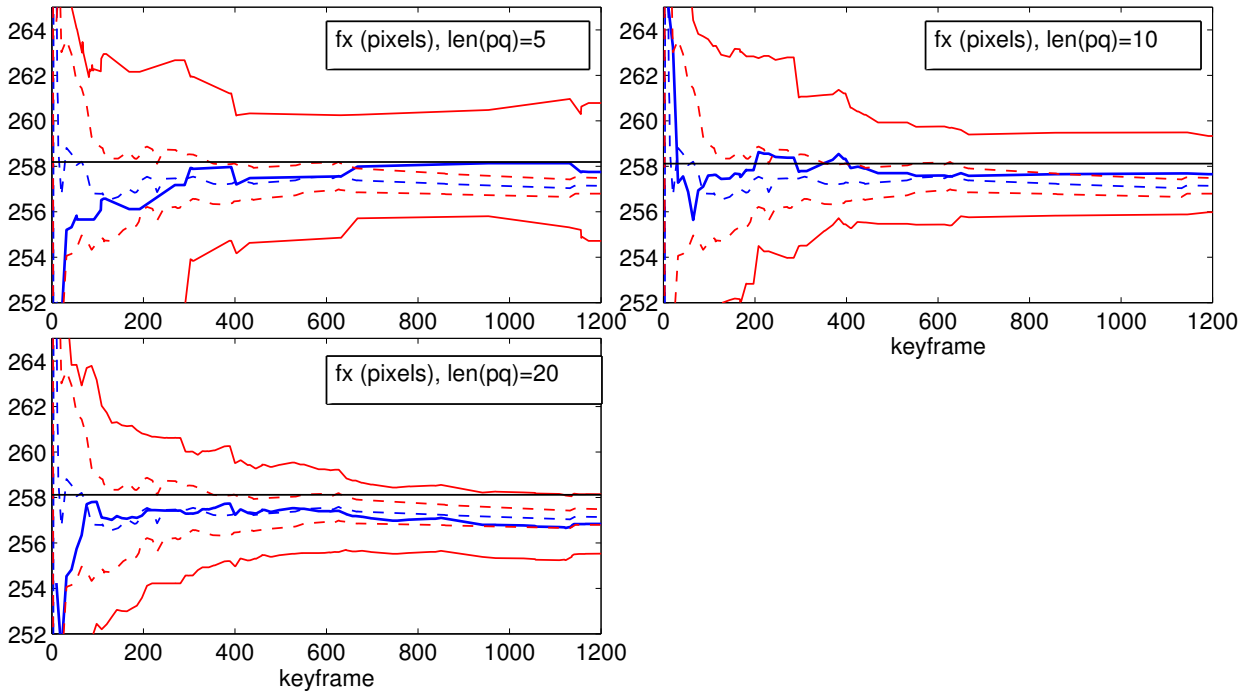


Figure 3.7: x focal length over multiple runs of the loop dataset from Fig. 3.4 with differing priority queue sizes. Larger priority queues tend to more accurately match the evolution of the batch solution (shown as the dashed line).

queue estimates will be understandably higher than that of the batch solution. For future work, the proposed framework presents an ideal platform for change-detection wherein a system reacts to intentional or unintentional changes to the physical calibration parameters. The inclusion of stereo and visual-inertial extrinsics calibration is straightforward given the framework, and would greatly increase the utility of the system. A number of potential failure cases outlined in section 3.3 also warrant further development of the priority queue update condition.

Chapter 4

Change Detection for Self Calibration

4.1 Introduction

In this chapter a novel extension to the self-calibration framework presented in Chapter 3 is introduced which enables handling changes in the calibration parameters. The aforementioned self-calibration framework was specialized to continuously improve the estimates of calibration parameters in constant-time, however the underlying assumption was of constancy in the parameters. That is, it was assumed that the actual ground-truth values of the calibration parameters were constant, therefore all information as to their estimates, regardless of temporal or spacial separation of the point-of-capture of said information, was admissible in the optimization framework used to estimate the parameters. This assumption breaks down if the ground-truth values of the parameters actually change, whether this change is intentional or not. Temporally separated measurements may now estimate different calibration parameters, and can no longer be combined in the same optimization. This necessitates an extension to the framework to actively look for and handle potential changes in the parameters. The presented extension handles this in a probabilistic manner. Similar to the method in Chapter 3, the extension presented in this chapter is derived for the estimation of intrinsic camera calibration parameters, but is able to be adapted to handle any calibration parameter for which constraints can be made relating them to the measurements taken by the sensors.

As discussed in Chapter 3, camera self-calibration is the inference of the intrinsic parameters of a camera without the explicit usage of a known calibration target. The motivation behind

self-calibration in robotics is two-fold: it facilitates the use of computer vision for localization, mapping or scene understanding without requiring arduous calibration procedures, and also in the case of long-term autonomy, robustness is achieved against accidental changes in the calibration parameters. For the first case, a self-calibration methodology which continually estimates a single set of calibration parameters (such as camera intrinsics) will suffice. However, in order to deal with changes in the calibration parameters, the approach must facilitate their re-estimation and detect the onset of the change event. Furthermore, it is desired to include other parameters such as extrinsics between different sensor modalities, time-delays or multiple cameras. For such a method to be useful in the case of long term autonomy however, it must be compatible with current developments in localization and mapping, as well as run in constant time to enable timely re-calibration in case of a change event.

With this motivation in mind, an approach is presented which enables the continuous estimation of camera intrinsics for a monocular setup in constant-time, while also simultaneously estimating the maximum likelihood camera and map parameters. The approach is based on a probabilistic method to detect when significant excitation of the calibration parameters is present in the motion to provide observability for estimation. It is able to deal with degenerate motions and non-linearities introduced due to unknown calibration parameters, which obviates the need for special initialization routines and motions. Probabilistic change detection indicates when the system should re-estimate parameters. Specific attention is paid to ensure past poses and landmarks are well estimated, even in the case of delayed observability of the calibration parameters. The approach is not exclusive to camera intrinsics and can be extended easily to estimate and detect changes in other calibration parameters, such as camera to IMU extrinsics, and time offsets. To the authors' knowledge, this is the first proposed solution to incorporate change detection for the estimation and re-estimation of calibration parameters in the SLAM setting.

The problem of self-calibration with varying camera intrinsics has received much attention in the literature in part due to the benefits outlined in section 4.1. [90] introduced a method to calibrate the varying focal length of a pinhole camera model in batch across images used for 3D

Figure 4.1: Online self-calibration during a zoom event in a $\sim 200m$ sequence. The left image was captured just before the zoom event. The center image was captured during the zoom, where focus is temporarily lost, causing a total loss of feature tracks. The right image shows an image captured just after the zoom. This event is automatically discovered and new accurate camera intrinsics are re-estimated online in real-time.

reconstruction with all other intrinsic parameters known. The method presented in [109] optimizes the focal length and 3D pose of different cameras by incrementally adding them to a batch optimization. [33] presented a method based on bundle adjustment which optimized the focal length and principal point location parameters at each image location for a batch optimization over multiple images. Expanding upon this, [88] introduced a method to detect good portions of the trajectory for self-calibration. Both papers look at self-calibration in the batch setting. [1] presented a method which using the infinite homography constraint, estimates the focal length and principal point for a camera which only rotates, but does not translate. The rotation parameters along with the camera intrinsics are solved in batch using a nonlinear optimizer. [114] introduced a plane-based method for calibrating pinhole camera models in the batch setting. The principal point and focal lengths are estimated for each camera. [60] introduced a method for self-calibration of pinhole cameras using the SVD of the fundamental matrix between two views to derive constraints. These constraints are then solved in a nonlinear batch setting to obtain the focal length and principal point parameters of every camera. [70] presented a method to calibrate the varying intrinsics of a pinhole camera in a batch setting, given the rotation of the camera was known. A solution was also offered to align the rotation sensor and camera data in time.

More recently, simultaneous solutions to the SLAM and self-calibration problem have been

proposed, however to the author's best knowledge, all proposed online solutions assume constant calibration parameters. [13] proposed a method to recursively estimate camera and landmark 3D parameters as well as the intrinsic parameters of a nonlinear camera model in an online framework. To deal with the large non-linearities introduced by the unknown calibration parameters, a Sum of Gaussians (SOG) filter is used in lieu of an EKF. [59] presented a method based on the MSCKF [78] filter which also calibrates the IMU to camera extrinsics. [65] introduced an EKF based method to estimate the calibration between an omnidirectional camera and robot odometry. [57] proposed a filtering solution based on the MSCEKF to estimate both the camera pose and also intrinsics and extrinsics for a non-linear camera model with rolling shutter and a commercial grade IMU in an online framework.

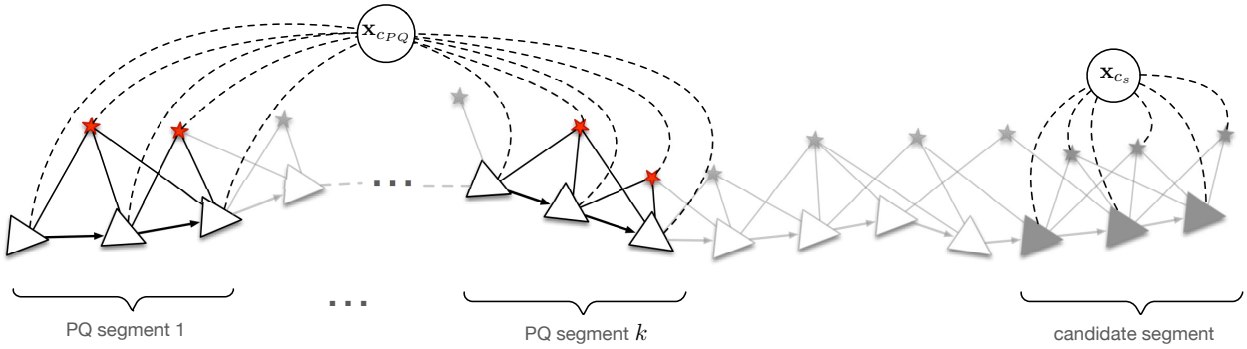


Figure 4.2: Graphical model showing the priority queue and candidate segments. Triangles represent camera pose parameters, while stars represent landmarks. Circles represent calibration parameters. In this figure, the priority queue segment size is 3 (refer to [46]). $\mathbf{x}_{c_{PQ}}$ refers to the calibration parameter estimate computed from the measurements in the priority queue, while $\mathbf{x}_{c_s}$ refers to the calibration parameter estimated computed solely from the measurements in the candidate segment. These two different estimates (and their posterior distributions) are used both to update the priority queue, and to decide if a change event has been detected.

4.2 Probabilistic Change-Detection for Self-Calibration

The proposed method aims to continuously estimate the calibration parameters in constant-time, while also detecting the onset of a change event brought forth by perturbations to the sensors. Simultaneously, the maximum likelihood estimates of the camera pose and map parameters are desired. The required functionality can be composed as three sub-components: *Constant Time*

Self-Calibration is required in order to recursively estimate the maximum likelihood calibration parameters at any point in the trajectory, *Change Detection* signals a high probability that the calibration has been perturbed during a change event, and *Adaptive SLAM estimation* is used to ensure maximum likelihood past and current camera and map parameters are estimated.

4.2.1 Constant Time Self-Calibration

To recursively estimate the calibration parameters in constant-time, the approach described in [46] is used. In order to aid the exposition of the overall self-calibration methodology, a brief summary of the method is presented here. The approach aims to obtain maximum likelihood values for the calibration parameters by selecting only the segments of the trajectory which provide the most information. In order to to assess this metric, a score is calculated based on the uncertainty of the calibration parameters as estimated by a particular *candidate segment*. This score is then compared against the score of each segment stored in a fixed-size *priority queue*. If the candidate segment score is better than the worst score in the priority queue, it is swapped into the priority queue. Once this update step takes place, a new estimate for the calibration parameters is obtained by using all segments in the priority queue jointly. As such, the priority queue will always contain the top k most informative segments of the trajectory, where k is a tuning parameter. The segments are of fixed size m, which is set as a constant tuning parameter. Figure 4.2 shows the graphical model representing the priority queue, candidate segment and their respective estimates for the calibration parameters.

The joint probability distribution of the estimator state parameters given the measurements (Z_j) contained in the segment j is given as

$$p\left(\mathbf{X}|Z_j\right) = p\left(\left\{\mathbf{T}_{wc} \in SE3\right\}, \left\{\rho\right\}, \mathbf{x}_c|Z_j\right) \tag{4.1}$$

where $\mathbf{T}_{wc}$ is the transformation from camera to world coordinates, ρ is the landmark parameter given inverse depth parameterization [75] and $\mathbf{x}_c$ is the vector of calibration parameters. Note that

the rotation component of the camera pose parameters $\{\mathbf{R}_{wc} \in SO3\}$ is actually locally parameterized in the $so3$ tangent plane [111]. Using Bayes' Rule, the the joint probability can be factored as follows

$$p\left(\mathbf{X}, Z_j\right) = p\left(Z_j|\mathbf{X}\right) p(\mathbf{X}) = \prod_{i=1}^{n} p\left(z_i|X\right) \tag{4.2}$$

here the likelihood term $p\left(Z_j|X\right)$ is factored due to the conditional independence assumption on individual measurements z_i, and the prior term $P(\mathbf{X})$ is omitted as the approach explicitly avoids a prior in favor of the priority queue. The optimal estimate for the parameter vector $\mathbf{X}$ is then one that would maximize the joint probability

$$\hat{\mathbf{X}} = \arg\max_{\mathbf{X}} p\left(\mathbf{X}, Z_j\right) \tag{4.3}$$

which would also be achieved by maximizing the likelihood term. Assuming a gaussian distribution over the parameter noise, the probability distribution over an individual term can be written as

$$p\left(z_i|\mathbf{X}\right) \propto \exp\left(-\frac{1}{2}\|z_i - h_i(\mathbf{X})\|_{\Sigma}^2\right) \tag{4.4}$$

where $\|\|_{\Sigma}^2$ denotes the squared mahalanobis distance, $z_i \in \mathbb{R}^2$ is the 2D pixel location of the landmark measurement and $h(\mathbf{X}) \in \mathbb{R}^2$ is the measurement model, which predicts the 2D location of the measurement given the current state variables. The measurement model is defined as

$$h_i(\mathbf{X}) = \mathcal{P}\left(\mathbf{p}_r, \mathbf{X}\right) = \mathcal{P}\left(\mathbf{p}_r, \mathbf{T}_{wc_m}, \mathbf{T}_{wc_r}, \rho_l, \mathbf{x}_c\right) \tag{4.5}$$

where $\mathcal{P}$ is a projection function which predicts the 2d pixel location of the projection of a given landmark into the measurement camera, given the 2d pixel location of the initial landmark observation in the reference camera ($\mathbf{p}_2$), the transformation from the coordinates of the reference and

measurement cameras to the world coordinate frame ($\mathbf{T}_{wc_r}$ and $\mathbf{T}_{wc_m}$ respectively), the inverse depth of the landmark in the reference camera ρ_l, and the calibration parameter vector $\mathbf{x}_c$. In the current implementation, $\mathbf{x}_c$ consists of the 5 parameters of the FOV camera model [14]:

$$\mathbf{x}_c = \begin{bmatrix} f_x & f_y & c_x & c_y & w \end{bmatrix}^T \tag{4.6}$$

where f_x, f_y, c_x, and c_y are the x and y focal lengths and principal point coordinates respectively, and w is a radial distortion parameter. Given this parameterization, the estimates for the camera poses $\hat{\mathbf{T}}_{wc_n}$, landmark inverse depths $\hat{\rho}_l$ and calibration parameters $\hat{\mathbf{x}}_c$ can be obtained via maximum likelihood estimation [119]. Furthermore, the normalized covariance matrix for the posterior distribution over the calibration parameters $\Sigma'_{\mathbf{X_c}}$ given the measurements can be obtained by inverting the problem's Fisher information matrix $\mathcal{I}$ at convergence, and extracting the appropriate submatrix. This covariance matrix is normalized so as to remove the effects of the differing units (as described in [46]), and is then used to compute a score which is the priority queue update metric as previously described. Once an update operation takes place on the priority queue, all currently added segments will be used to jointly estimate a new value for the calibration parameters. The newly estimated parameters are then assigned to the frames in the set $\{n_{change}, \ldots, n_{current}\}$ where n_{change} is the frame index of the last detected change event, and $n_{current}$ is the index of the current frame.

Due to the existence of critical motions which render some calibration parameters unobservable, the candidate segment covariance matrix is checked to ensure that it is full rank and well conditioned. If not, the candidate segment is discarded.

4.2.2 Initialization

Similar to the method described in [46], a special initialization phase is used to bootstrap the priority queue and initialize the calibration parameters. A batch optimization is run over all state parameters (camera locations, parameter inverse depths and calibration parameters) from

the most recent change index n_{change} to the current frame $n_{current}$, much like the procedure used over a candidate segment. This joint estimation is run until the score [46] of the batch segment falls below a particular threshold. As the score is calculated from the entropy of the normalized posterior covariance, this threshold is a direct measure of the uncertainty of the posterior, and aims to prolong the batch optimization until the uncertainty over calibration parameters has been sufficiently reduced.

Once this criteria is met, normal operation proceeds, where candidate segments are evaluated and added to the priority queue as necessary.

4.2.3 Change Detection

As shown in Figure 4.2, at each point in the trajectory, two posterior distribution estimates for the calibration parameters are available, each represented by a covariance matrix and mean. One is computed considering only the measurements within a candidate segment which is being evaluated for addition to the priority queue, with covariance Σ'_s, and another considering all measurements contained by the segments in the priority queue, with covariance Σ'_{PQ}.

The priority queue posterior (with covariance Σ'_{PQ}) represents the uncertainty over the calibration parameters considering the top k segments in the trajectory. As these segments can have significant temporal separation, this distribution encodes the long term belief over the calibration parameters. Conversely, the candidate segment posterior (with covariance Σ'_s) is calculated based on the most recent measurements and represents an instantaneous belief over the calibration parameters.

A possible change in the actual calibration parameters would therefore manifest as a difference in the means represented by these two posterior distributions. The hypothesis test that two multivariate normal distributions with unknown and unequal covariance matrices have the same mean is known as the Multivariate Behrens-Fisher problem. The interested reader is directed to reviews for the univariate [52] and multivariate [83] cases. Briefly, the null hypothesis of the test for the change detection case is as follows

$$H_0 = \mu_{PQ} = \mu_s \tag{4.7}$$

where μ_{PQ} is the mean estimated for the posterior distribution considering all the measurements in the priority queue, and μ_s is the mean estimated for the posterior distribution considering only the measurements in the candidate segment. For the purposes of this chapter, the particular solution to the Behrens-Fisher problem used is the one proposed in [120]. Given this method, the null hypothesis has an approximate F distribution which is given by

$$F_{p,v-p+1} \sim T^2 \frac{v-p+1}{vp} \tag{4.8}$$

where the F distribution has degrees of freedom given by $p = \dim(\mathbf{x}_c) = 5$ and $v - p + 1$ with

$$v = \left[\frac{1}{n_{PQ}} \left(\frac{\mu_d^T \tilde{\Sigma}^{-1} \tilde{\Sigma}_{PQ} \tilde{\Sigma}^{-1} \mu_d}{\mu_d^T \tilde{\Sigma}^{-1} \mu_d} \right)^2 + \frac{1}{n_s} \left(\frac{\mu_d^T \tilde{\Sigma}^{-1} \tilde{\Sigma}_s \tilde{\Sigma}^{-1} \mu_d}{\mu_d^T \tilde{\Sigma}^{-1} \mu_d} \right)^2 \right]^{-1}$$

$$\tilde{\Sigma} = \tilde{\Sigma}_{PQ} + \tilde{\Sigma}_s$$

$$\tilde{\Sigma}_{PQ} = \frac{1}{n_{PQ}(n_{PQ}-1)} \Sigma_{PQ}$$

$$\tilde{\Sigma}_s = \frac{1}{n_s(n_s-1)} \Sigma_s$$

$$T^2 = \mu_d^T \tilde{\Sigma}^{-1} \mu_d$$

where n_{PQ} and n_s are the number of measurements in the priority queue and candidate segment respectively, and $\Delta\mu$ is the difference in the mean estimated by the priority queue and candidate segment given by $\mu_d = \mu_{PQ} - \mu_s$. Note that Σ_{PQ} and Σ_s are the un-normalized posterior covariances. Once the approximate F distribution is calculated, a corresponding p-value can be obtained, which can be compared to a level of significance parameter α, and the null hypothesis is rejected if $p \leq \alpha$ where $\alpha = 0.1$ is a tuning parameter which adjusts the sensitivity of the change detector.

Using the previously outlined method for change detection, the uncertainty of both the priority queue and the candidate segment are considered in determining the probability of a change

event. If the candidate segment mean is significantly different than the priority queue mean, but the entropy of its posterior is high relative to the priority queue posterior, it may have a lower probability of being a change event than a posterior distribution with a very low entropy relative to the priority queue, but significantly less deviation from the mean.

Sub-optimal tracking, motion blur and non-static features can all cause potentially misleading posterior distributions to be estimated for the candidate segment. As such, a simple heuristic is implemented to ensure that a single candidate segment which has a test p-value less than the level of significance parameter does not signal a change event. A number n_{test} of candidate segments must consecutively reject the null hypothesis, where $n_{test} = 3$ is a tuning parameter. If such an event does take place, the detected change frame index is set to $n_{change} = n_{current} - n_{test}$. This sets the starting index for a new set of calibration parameter assignments, as described in section 4.2.1.

Once a change event is detected, all of the current segments in the priority queue are removed, as they represent information contributing to a different set of calibration parameters, and the initialization routine outlined in section 4.2.2 is run once again to obtain an initial estimate over the new parameters. Once the initialization criteria has been met, the new priority queue is populated as per section 4.2.1. Note that while the exact onset of a change in camera parameters is not obtained given the proposed method, the number of keyframes per segment is limited such that an estimate of the new parameters is computed soon after the change event.

4.2.4 Adaptive SLAM

The final component is an adaptive SLAM estimator which is able to incorporate the updates from the calibration estimation and ensure that maximum likelihood current and past poses are re-estimated if calibration parameter estimates are updated. An immediate choice would be a recursive filtering method for SLAM estimation, [56] [58] [78], However, with similar goals to that of the method in [46], a linearized prior distribution is avoided in order to both keep the estimator consistent, and to enable the ease of integration of loop closure and relocalization constraints. Fur-

thermore, as the aim of the system is to obviate the need for particular initialization motions (such as the SLAM wobble), the SLAM estimator must be able to handle potentially degenerate motions, where observability over calibration parameters does not eventuate until some time after initialization. In this case, past camera location and inverse depth parameters must be retroactively updated to ensure maximum likelihood map and trajectory estimates. Such non-linear updates (exacerbated by the non-linearities introduced [13] when intrinsic camera parameters change), motivate the use of an optimization based approach, which avoids marginalization altogether. Since consistency in filtering approaches is achieved by the use of *first estimates* Jacobians [59], a consistent filter would be further susceptible to non-linearities in the parameters and prior distribution. Considering the aforementioned points, the Adaptive Asynchronous Conditioning (AAC) [45] method is used to adaptively adjust the optimization window of an asynchronous bundle adjuster based on the error of the conditioning edge to the inactive part of the trajectory.

When the calibration parameters are updated, the landmarks projecting in both the active and inactive portions of the trajectory will cause an increase in the conditioning error which will be lowered by expanding the optimization window and re-estimating past poses and landmarks. This is done asynchronously along with a constant-size windowed estimator which is conditioned on the inactive part of the trajectory and runs synchronously with the tracker.

The two AAC estimators run alongside the self-calibration estimation and are used to obtain the final estimates over camera position and landmark inverse depth parameters. This is also the case when the self-calibration estimator is in the initialization phase (section 4.2.2). As the AAC estimator may enlarge the optimization window to span across a parameter change frame (indexed by n_{change} described in section 4.2.3), care is taken to ensure that the appropriately assigned calibration parameters are used on either side of the change event.

Note that in section 4.3, the AAC estimator is used with IMU information in order to maintain scale during an experimental trajectory, and also in order to carry the estimation through the loss of tracking caused by de-focus during the zoom operation (center image in Figure 4.1). The formulation for the integration of IMU measurement is as per [44]. IMU measurements are solely

used for scale-correct pose and map estimation. While IMU measurements contribute to the initial pose and landmark estimates which are passed to the self-calibration estimator, they are not used by it in the estimation of poses, landmarks or calibration parameters, which are re-estimated using only visual measurements.

4.2.5 Tracking and Keyframing

To obtain feature correspondences between images, a method inspired by the tracking component of [22] is used, where the photometric error of a re-projected feature patch is directly minimized to obtain the new location of the feature. To initialize features, harris corners are used in a region of the image where not enough active tracks are present. Since the tracking method respects projective geometry, no RANSAC is required for frame-to-frame outlier rejection. Projective outliers however are still rejected in the bundle adjustment based on residuals. An NCC score is computed between the current and original feature patches and is thresholded (at 0.875) to reject feature patches which have changed too much in appearance.

For improved performance, and to solve the stationary camera problem, a keyframing [68] system is implemented based on heuristics on the rotation angle of the camera, traveled distance, and number of successful feature tracks from the last keyframe. If the criteria based on any of these metrics is met, a new keyframe is inserted and the current location of tracked features is used to add projections residuals. As such, if the camera is stationary or hardly moving, the features and consequently the pose of the camera are tracked with respect to the previous keyframe, but no new residuals are added to the problem.

4.3 Experiments

The proposed method was run on real images captured from the visual-inertial rig in Figure 4.5. The rig consist of a grayscale global shutter camera recording at 640x480 pixels resolution and 30 frames per second, and a commercial IMU collecting accelerometer and gyroscope data at 200Hz. A varifocal lens is used with the camera, to allow the zoom and focus to be manually

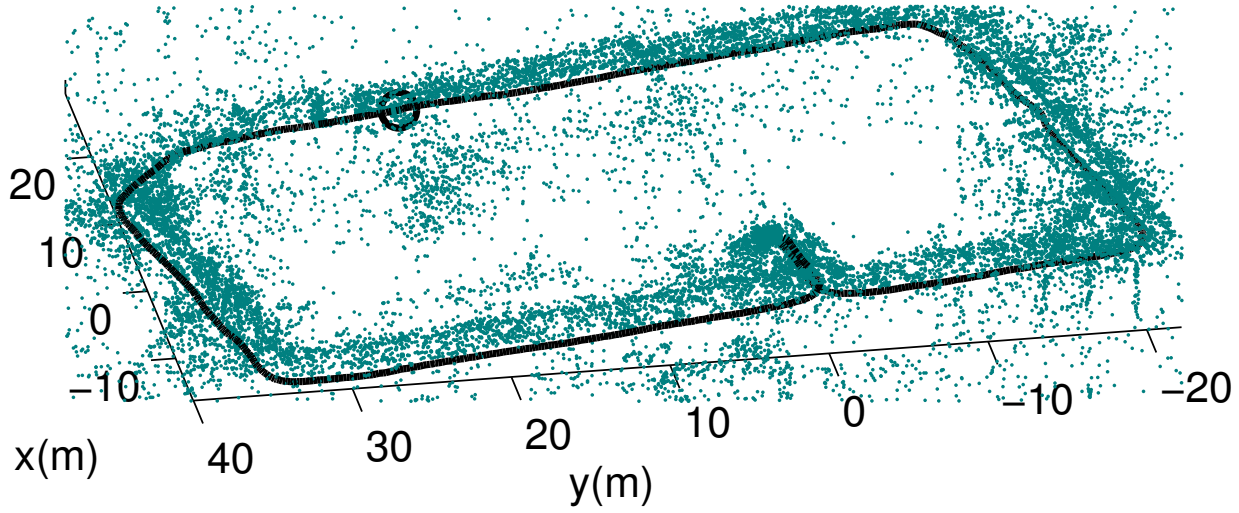


Figure 4.3: Reconstruction and trajectory result from running the online self-calibration and AAC estimators on a 193.5m dataset captured on foot with the rig shown in Figure 4.5. The circle points to the location of the zoom event, where the focal length of the lens was manually adjusted from approximately 4mm to 8mm. The final translation error between the start and end poses is 0.8m or 0.42% of the traveled distance. IMU measurements are used in the AAC estimator to ensure scale consistency in the map and trajectory estimates.

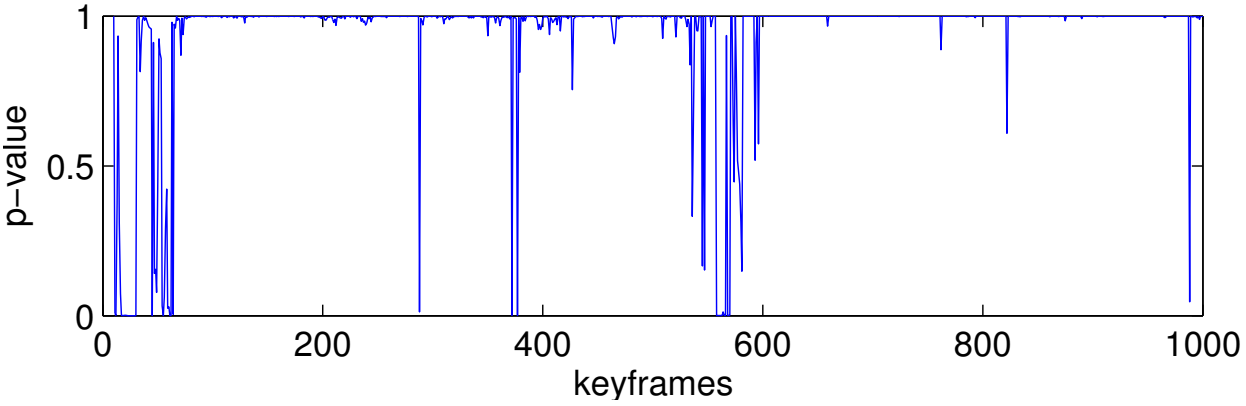


Figure 4.4: p-values for the hypothesis test described in section 4.2.3, proposing that the priority queue and candidate segment window posterior means (corresponding to their best estimates of the calibration parameters) are different. p-values lower than the significance parameter $\alpha = 0.1$ signal that calibration parameters have changed. The initial calibration starting at keyframe 0, as well as the zoom event at approximately keyframe 580 can be seen as numerous consecutive p-values smaller than the significance parameter.

changed while collecting data. The images were used as described in section 4.2.5 to obtain feature tracks. The tracks were then made available to the SLAM and self-calibration estimators via a

shared map structure. IMU information was additionally made available to the AAC estimator for scale consistency. In all experiments, the initial intrinsics estimates for the focal lengths were set to represent a field of view of 90°, and the principal point initialized to be the image center. The distortion parameter w was initialized to 1.

The first experiment was performed on an outdoor dataset captured by walking around a $\sim 200m$ loop. A manual zoom event was performed (marked by the circle in Figure 4.3) which consisted of changing the focal length of the camera from approximately 4mm to 8mm, as per the lens specifications. Due to the manual focus during the zoom, significant blur is introduced (as shown in Figure 4.1 which causes all feature tracks to be lost. In order to verify the accuracy of the reconstruction, IMU information is used in the AAC estimator (as per [44]) to both maintain a consistent scale through the trajectory, and also to carry the estimation through the segments where tracking loss is encountered. Scale loss is also encountered when the zoom event is introduced, as the intrinsics are not instantly known. The AAC estimator is used once again to re-acquire scale and maintain a maximum likelihood map and camera pose estimate, although this does not happen immediately. As the intrinsics are estimated by the self-calibration estimator, the AAC estimator simultaneously attempts to find the maximum likelihood scale, camera pose and map structure given the intrinsics. In order to properly utilize IMU measurements. the IMU to Camera transformation and time delay were calibrated offline in a batch optimization using a known target and set to constant during the experiment. Note that IMU information was not used in the self-calibration estimator, and the intrinsics were estimated solely from visual feature tracks.

Apart from the heuristics outlined previously, the number of segments in the priority queue is set to 5 segments of size 10 keyframes each (for a description of these heuristics, the reader is directed to [46]). The tracker was configured to attempt to maintain 128 active feature tracks at any time. The keyframing heuristics were configured to add a new keyframe if the camera motion exceeds 0.1 radians in rotation, 0.2m in translation, or more than 20% of tracks from the previous keyframe are lost.

The dataset features no special initialization motion, and involves simple forward walking

Figure 4.5: Experimental rig used to capture images (640x480 pixels at 30fps) and IMU data (at 200Hz). Consists of a USB camera attached to a USB commercial-grade IMU. The varifocal lens has adjustments for zoom and focus, with focal length variable between 4mm and 8mm.

from the first frame. The self-calibration and AAC estimators initialize the intrinsics and ensure maximum likelihood map and pose estimates, with consistent scale as obtained from the additional IMU measurements. Figure 4.3 shows the results of running the system on the dataset. As no ground truth measurements were available during the trajectory, the difference between the start and end positions of the camera were used to measure the accuracy. The dataset was captured to ensure that the start and end positions of the camera were the same. The final estimated translation error between the start and end camera positions is 0.8m, equating to a 0.42% error per unit distance traveled.

Figure 4.4 shows the p-value plot (defined in section 4.2.3 from the same dataset. It can be seen that at the beginning and around keyframe 580 the p-value is very small, indicating a large probability that the means of the priority queue and candidate segment posterior distributions are different. In these two cases, the indication is indeed warranted, as when the batch mode for self-calibration is activated (as described in 4.2.2), the means of the priority queue and candidate segment posteriors will often be quite different, as a proper estimate of the calibration parameters

is not yet available in the priority queue. However in several parts of the trajectory, spurious dips in the p-value can be seen, sometimes below the significance level parameter α. These dips are the motivation for the heuristic introduced in section 4.2.3. Since the spurious dips are usually for a single candidate segment only, they do not falsely trigger the change detection system.

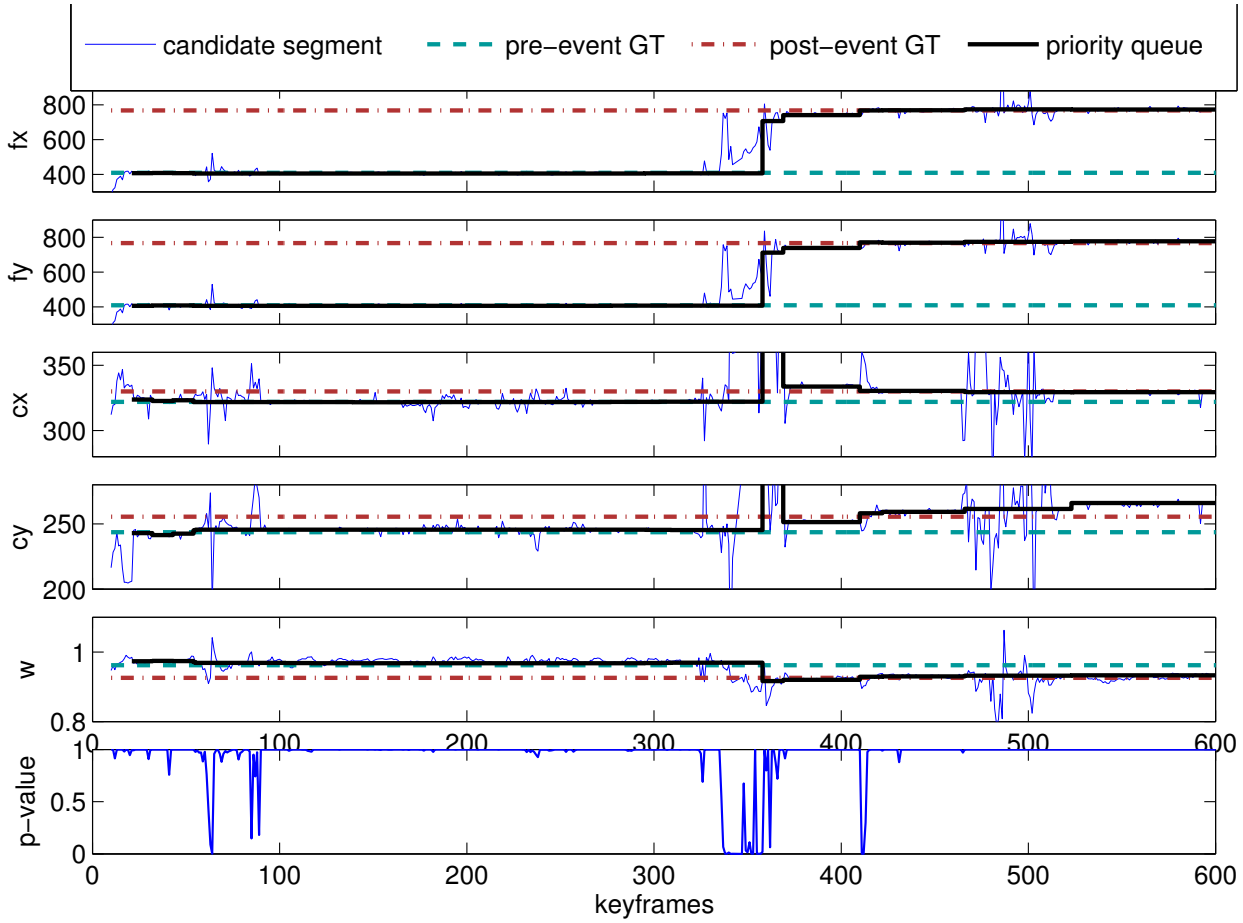


Figure 4.6: Comparisons between the ground truth intrinsics obtained from an offline calibration library [95] and the priority queue and candidate segment estimates. The results are computed over a short outdoor dataset with simple forward walking motion. The plots are of the x and y focal lengths and principal point coordinates respectively, as well as the calibration parameter w and the change detection p-value. The detection of the zoom event is clearly visible in the p-value plot.

The second experiment was undertaken with the aim of comparing the pre and post change event intrinsics with ground truth values obtained from an offline calibration method. The calibration library [95] utilized uses a pre-defined target to estimate the intrinsics. Figure 4.6 shows

the results of the ground truth values compared with the results obtained from running just the self-calibration estimator (no AAC) on images obtained from an outdoor dataset. Similar to the first experiment, the dataset consists of images captured during forward walking motion, with a manual zoom introduced partway through (detected approximately at the 400th keyframe). The figure shows the pre and post-zoom ground truth values, as well as the same parameter as estimated by the priority queue and the candidate segment. The priority queue estimate is shown as a stairstep plot, with steps occurring when a candidate segment is folded in, and a new estimate is computed.

It is notable that while at some points the candidate segment estimates vary wildly, the priority queue is not affected. These bouts of instability in the candidate segment can be caused by a variety of factors, including loss of tracking, motion blur, and outliers. The stability of the priority queue estimate during these portions is due to the high uncertainty in the posterior distribution of the parameters computed for the candidate segment. As such, these segments are not folded into the priority queue.

After the zoom event, instability is also observed in the priority queue estimate, as all the segments in the queue are cleared, and the initialization routine is once again started (section 4.2.2). This also explains the delay between the p-value change, and the first update to the priority queue. In this section, the batch optimization is run to initialize an estimate for the intrinsics. Once the initialization uncertainty reaches below a certain threshold (section 4.2.2), the priority queue estimation is activated. For the most part, the estimates agree to a satisfactory level with the offline calibration values. Although theoretically the estimator should be consistent and unbiased, a more thorough examination of the results (such as with Monte-Carlo simulations) are in order, and will be slated for future work.

The p-value plot shows a similar trend to that of Figure 4.4, where apart from a few spurious dips (which are ignored due to the heuristic introduced in section 4.2.3), the zoom event is clearly detected by a continuous dip in the p-value, suggesting a change event has occurred.

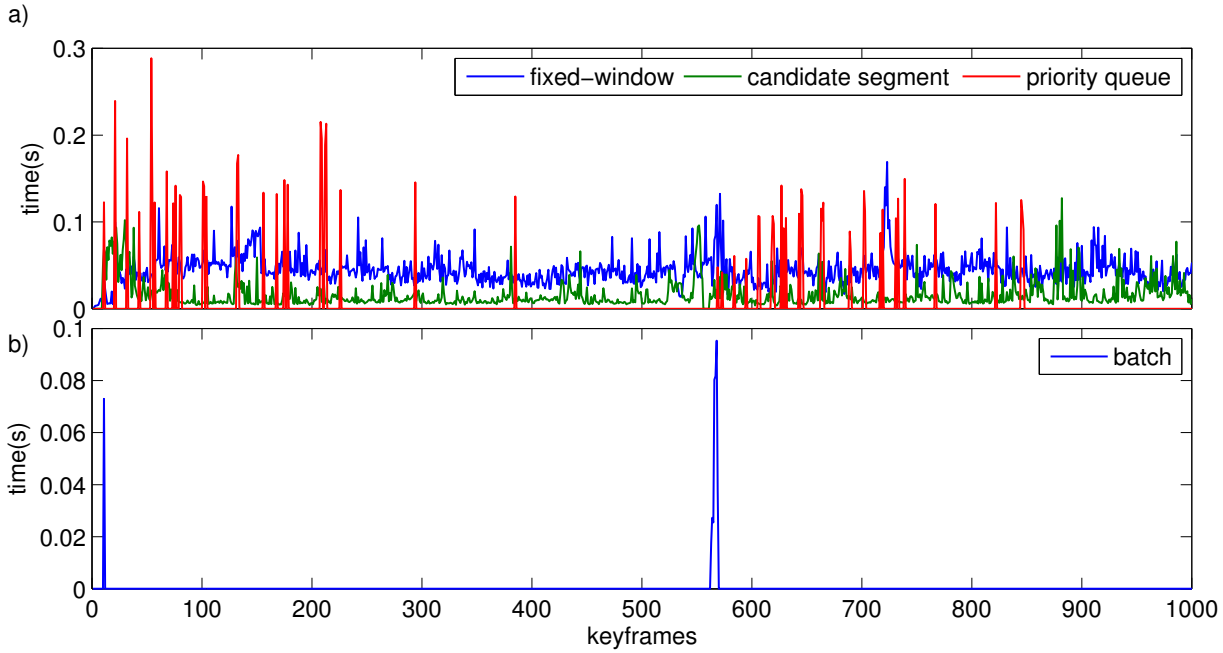


Figure 4.7: a) Time taken per keyframe for the fixed-window SLAM, candidate segment and priority queue estimators. It can be seen that the fixed-window and candidate segment estimators run in constant-time, while the priority queue estimator takes longer, but is only run when an update to the priority queue takes place and the calibration parameters need to be re-estimated. b) Time taken for the batch optimization. It can be seen that the optimization is only invoked at the very beginning of the trajectory, and at the change event. Both times the optimization is run to initialize the calibration parameters as per section 4.2.2.

4.4 Performance

The current implementation runs at approximately 23fps on a Core i7 2.5Ghz processor laptop with a synchronous implementation of the AAC fixed-window and self-cal estimators. Figure 4.7a shows the timing information from the first experiment. It can be seen that the fixed-window estimator runs in constant-time during the entire trajectory. The estimator which obtains the posterior distribution over the candidate segment also runs in constant-time. The priority queue estimator is dormant for most of the trajectory, except when a candidate segment is swapped in, in which case a spike is observed as new intrinsics are estimated. Figure 4.7b shows the timings for the batch estimator. It can be seen that as stated during section 4.2.2, the batch estimator is only run during initialization or when a change event is detected, in order to initialize the estimate

for the calibration parameters. Currently, all estimators in Figure 4.7 are run synchronously. Run independently, the front-end tracker and fixed-window optimization achieves greater than 60fps. A more optimized approach would be to run the front-end, candidate segment, and AAC solver asynchronously. This would both increase the average frame-rate of the system as well keep the front-end running at the fastest rate.

4.5 Summary

In this chpater, a method for online, constant-time self-calibration and automatic change detection and re-calibration was presented. Experiments demonstrate that the system can estimate accurate calibration parameters, camera poses and landmark depths without any prior information. A filtering framework is explicitly avoided in favor of an adaptive asynchronous optimization [44] which provides the maximum likelihood current and past camera pose and landmark estimates. Rather than roll past information about the calibration parameters into a linearized prior distribution prone to inconsistency, a priority queue [46] is used to store the most-observable segments in the trajectory to estimate the calibration parameters. The approach enables "hands-off" initialization, where no specialized motion is needed. As parameters become observable over time the relevant segments are automatically included in the priority queue, and past camera pose and landmark parameters are updated when necessary.

A novel application of multivariate probabilistic change detection spurs the re-estimation of parameters if a significant change occurs. This causes re-calibration in situations where the physical sensor rig is perturbed. To the authors' knowledge, this work is the first instance of online self-calibrating SLAM which can handle a significant change in the calibration parameters while still estimating the full maximum likelihood map and trajectory.

With additional IMU measurements to aid scale estimation, the system is able to achieve a distance-traveled error of 0.42% even in the presence of a significant calibration change and total loss of tracking. Experiments show that parameter estimates from before and after a calibration-change event closely match values obtained via manual estimation with a calibration target.

While this paper has demonstrated real-time methods for two long-standing challenge-problems in robotics, namely 1) power-on-and-go self-calibration and 2) robust long-term SLAM in the face of model perturbation, the application of **probabilistic change detection** is perhaps the most compelling result; it is a powerful tool for introspection and model verification more generally.

Chapter 5

Conclusions

In this thesis, the problem of simultaneous localization and mapping (SLAM) using visual and inertial measurements has been explored with a particular focus towards challenges encountered when pursuing long term autonomy. Specifically, the focus has been on ensuring the accuracy of localization and the consistency of mapping given an open-ended runtime requirement. The challenges motivated the design of several novel frameworks which aid in both the core estimation of the SLAM problem, as well as the continued estimation and maintenance of calibration parameters. The main contributions of this thesis are:

- A method for visual-inertial SLAM which avoids marginalizing and instead uses an asynchronous, adaptive form of conditioning in order to maintain constant-time computationally complexity while simultaneously correcting errors in past parameters as new information becomes available. The use of a relative representation for the map allows asynchronous optimization of the graph at different scales. Crucially for the problem of long term autonomy, the use of conditioning rather than marginalization has the potential to simplify the treatment of loop-closures when using both visual and inertial constraint. The framework has shown to work in real-time with bounded computational complexity, and to result in accurate pose estimates.

- A framework for self-calibration with bounded complexity which uses a probabilistic algorithm to pluck the best portions of input measurements for the purpose of calibration as

they become available, thereby avoiding marginalization while closely approximating the optimal batch solution. Due to the constant time complexity, the self-calibration framework can run indefinitely in the background, continually improving the calibration estimates as new data becomes available. In the long term autonomy setting, this can be a crucial advantage, especially for hard to calibrate parameters where the initial, offline estimates are made with limited data and may not be optimal The framework is shown to produce accurate results when using visual measurements for the purposes of calibrating the extrinsics. However, it can easily be extended to produce estimates for any calibration parameters as long as input measurements can be used to form constraints in order to estimate them.

- An extension to the self-calibration framework previously presented in order to address a shortcoming in the method borne out of an assumption that calibration parameters do not change throughout the life of the robot. This assumption can be particularly problematic in the case of long term autonomy, where physical changes to the robot may suddenly change the calibration at any time. The extension provides a probabilistic method for detecting the change, and uses the adaptive asynchronous conditioning framework also described in this thesis to seamlessly attempt to recalibrate new estimates for the parameters while maintaining a constant-time SLAM solution. The extension is shown to produce accurate results in the present of significant and sudden changes to the camera intrinsic calibration (in the form of a zoom event), when using both visual and inertial measurements. Similar to the self-calibration framework it extends, the extension is able to be adapted to estimate other calibration parameters.

These contributions work together to enable more robust localization using inertial data to supplement visual measurements, while also advancing the goal of life-long continuous self calibration. An ambitious but in this author's opinion worthwhile goal for robotic systems is for them to never need calibration during their lifetimes even in the presence of physical alterations due to extraneous influences, and to also be able to self-calibrate from the instance they are turned on, a

true power-on-and-go solution. This is due to the inevitability of extraneous factors that influence calibration except for the most tightly controlled circumstances. A true power-on-and-go solution is desirable as it reduces barriers to operation, increasing ease of use and adoption potential for robotic systems.

The contributions presented in the thesis loosely follow a central theme: that of eschewing global constraints and instead focusing locally. This theme is apparent in the decision to forgo marginalization in favor of conditioning in Chapter 2. It can also be observed again in Chapter 3, where a local window is used to make an estimate of calibration parameters. This local focus is rooted in practicality: robotics applications generally involve interaction with the local environment, and therefore require high accuracy in that space. Global accuracy and consistency can be useful for applications such as distance measurement, or the creation of consistent reconstructions of the environment. However, interactions often occur in the local space, where accuracy is key. For example, a ground robot tasked with retrieving an object from a room in a building generally requires some global representation of the spaces within it. This global representation needs to be consistent to the degree which allows navigation within the building from the starting location to the room containing the object. This representation can be topographical and therefore globally inconsistent, while still allowing navigation to the room. However, regardless of the global accuracy and consistency of the representation, a highly accurate local representation is nonetheless needed in order to navigate through the space without colliding with obstacles, and avoiding non-traversable terrain. Ultimately, the task of retrieving the object requires accurate local representations of the manipulator used to retrieve the object and the object itself. The global consistency of the representation of the object, the robot and the manipulator can be factored out. These interactions do not and should not consider the global representation, as it simply does not matter. Instead, a local representation will factor out any global accuracies, and instead focus on relative relationships which are required for the task at hand. A similar condition can be observed in applications involving Augmented Reality. The objective of overlaying 3D information over the real scene requires accurate localization and tracking to the scene. However, as in the case of the navigation

task previously mentioned, local accuracy is required in order to believably overlay 3D imagery over real scenes, but global accuracy and consistency can be less important. These conditions are also applicable to estimating uncertainties. When navigating or interacting with the local environment, the *local* uncertainty is directly applicable for the task at hand, and can be a used to assess the likelihood of success or even to initiate actions in order to better localize the robot or the space it's interacting with. Global uncertainty on the other hand is less useful. For example, in the case of manipulating an object, the pose of both the robot and the scene can be largely uncertain with respect to the global origin, but manipulation would still be possible if the *relative* uncertainty of the robot manipulator and the object in the local scene is low. These practical requirements fundamentally motivate the theme of local methods in this thesis.

5.1 Future Work

Each topic explored in this thesis leaves a number of follow-on topics for research and development. In this section, a few examples of these topics are listed for each of the subjects discussed in the preceding chapters:

- The visual-inertial SLAM framework presented in Chapter 2 eschewed marginalization in favor of an asynchronous adaptive variant of conditioning. This presents some benefits, specifically regarding loop closure incidents or when attempting to localize against a prior map. However, an actual implementation would have to overcome many hurdles, such as the treatment of the gravity vector and alignment of the entire map with respect to it. Of further interest is the treatment of inertial measurements when localizing again a map that contains both visual and inertial measurements, as the traversal velocity between different incidences of localization will not be the same. Finally, the adaptive nature of the algorithm creates a challenge when localizing against the map, specifically for the case where the localization adds new information to the prior map and an adaptive visual-inertial optimziation has to take place in order to obtain the maximum likelihood estimate

for both the prior map as well as the current location of the visual-inertial rig.

- The self-calibration framework in Chapter 3 enabled life-long calibration by selecting relevant segments of input data. These segments were then stored in a fixed-size priority queue, ordered by the amount of information they contributed to the calibration parameters. The framework thus allowed constant-time, lifelong self-calibration. While the framework is able to be extended to cover any calibration parameter for which optimziation constraints can be made against data, the specific implementation and experiments focused on camera intrinsics. Each camera parameter will bring with it a bespoke requirement for constraints against input data, as well as initialization. Extending the framework to cover visual-inertial calibration will thus require research to develop these constraints as well as a fast initialization routine that quickly approximates the visual-inertial calibration parameters to bootstrap the priority-queue self calibrator.

- The extension to the previous mentioned self-calibration framework allowing it to handle changes in the calibration parameters was introduced in Chapter 4. This extension employed a probabilistic method to determine if the ground-truth values of the calibration parameters had changed, and if so provided a framework for detecting the exact time of the change and handling the re-initialization and subsequent self-calibration required to ensure continued operation. In this work, the calibration parameters were modeled as a 0^{th} degree polynomial, and the changes detected were assumed to be in the constant value of the parameter. However it would be possible to extend the method to use other methods such as linear or nonlinear change, and to detect when the parameter no longer fit a particular model. Doing so would introduce the challenge of constraint formation and initialization to match the new model, but would enable handling a wider range of change possibilities. The presented extension was also only tested on the problem of camera intrinsic calibration (similar to the method in Chapter 3 which it extends). However, it would be possible to adapt it to the problem of visual-inertial calibration and change detection.

This would be similar to the adaptation required by the self-calibration framework itself to work with visual-inertial data, as the change-detection method itself would be agnostic to the particular calibration parameters involved. However, specific challenges involving the relative uncertainties of the additional calibration parameters would have to be overcome, as well as deciding whether or not to group all of the calibration parameters together or separate them for the purposes of change detection.

While the methods presented in this thesis were done so with the motivation of enabling long-term autonomy, the simultaneous localization and mapping frameworks outlined are better placed under the *visual odometry* category. Even though these frameworks build maps of the environment while also estimating the maximum likelihood pose of the camera, the maps and representations that are built are not subsequently used for localization. This has the side effect that errors in the odometry system will grow unbounded. For true long-term autonomy, long-lasting representations are required. There is a considerable challenge in going from visual odometry systems like the ones detailed in this thesis, to long-term representations of the environment that can be used for autonomy. These representations would require robustness not only to non-static scene components and changes in calibration, but also to systemic changes in the environment which would render parts of the map potentially obsolete. Such a system would not only build maps, but also *maintain* them, in order to support localization over long time durations. While the local methods for visual-inertial SLAM presented in this thesis can enable such long-term mapping due to the ease with which constraints against an incumbent map can be incorporated into the estimation framework, there still remains a large body of future work in designing and proving out such a framework. Furthermore, the local methods detailed in this thesis are only *spatially* local. However, temporal locality must also be considered when dealing with long-term autonomy, as the appearance - and by consequence, the representation - of a scene may change depending on the time of day or time of year. Disregarding this temporal locality can lead to maps which are continuously expanded or modified in order to represent a bounded environment. These topics comprise some of the many

potential directions for research surrounding long-term autonomy and the creation and maintenance of supporting representations of the environment.

Bibliography

[1] L. De Agapito, E. Hayman, and I. Reid. Self-calibration of a rotating camera with varying intrinsic parameters. In In Proc 9th British Machine Vision Conf, Southampton, pages 105–114, 1998.

[2] L. De Agapito, E. Hayman, and I. Reid. Self-calibration of a rotating camera with varying intrinsic parameters. In In Proc 9th British Machine Vision Conf, Southampton, pages 105–114, 1998.

[3] Sameer Agarwal, Noah Snavely, Steven M. Seitz, and Richard Szeliski. Bundle adjustment in the large. In Proceedings of the 11th European Conference on Computer Vision: Part II, ECCV'10, pages 29–42, Berlin, Heidelberg, 2010. Springer-Verlag.

[4] Hatem Said Alismail, Brett Browning , and Simon Lucey. Robust tracking in low light and sudden illumination changes. In International Conference on 3D Vision, October 2016.

[5] Nancy M. Amato and Yan Wu. A randomized roadmap method for path and manipulation planning. In In IEEE Int. Conf. Robot. & Autom, pages 113–120, 1996.

[6] Kai Oliver Arras. An introduction to error propagation:derivation, meaning and examples of equation cy=fxfxfxt̂. 1998.

[7] Anil Aswani, Patrick Bouffard, and Claire Tomlin. Extensions of learning-based model predictive control for real-time application to a quadrotor helicopter. In ACC 2012, to appear, Montreal, Canada, June 2012.

[8] Cedric Audras, Andrew I. Comport, Maxime Meilland, and Patrick Rives. Real-time dense RGB-D localisation and mapping. In Australian Conference on Robotics and Automation, Monash University, Australia, 2011.

[9] Jerome Barraquand, Lydia Kavraki, Jean-Claude Latombe, Tsai-Yen Li, Rajeev Motwani, and Prabhakar Raghavan. A random sampling scheme for path planning. International Journal of Robotics Research, 16:759–774, 1996.

[10] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Speeded-up robust features (surf). Comput. Vis. Image Underst., 110(3):346–359, June 2008.

[11] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In Proceedings of the 11th European Conference on Computer Vision: Part IV, ECCV'10, pages 778–792, Berlin, Heidelberg, 2010. Springer-Verlag.

[12] J.-W. Choi, Ren Curry, and Gabriel H Elkaim. Continuous curvature path generation based on bézier curves for autonomous vehicles. IAENG International Journal of Applied Mathematics, 40, 2010.

[13] Javier Civera, Diana R Bueno, Andrew J Davison, and JMM Montiel. Camera self-calibration for sequential bayesian structure from motion. In Proc. IEEE International Conference on Robotics and Automation (ICRA) 2009, pages 403–408. IEEE, 2009.

[14] Frederic Devernay and Olivier Faugeras. Straight lines have to be straight. In SPIE, volume 2567, 2001.

[15] A.W. Divelbiss and J.T. Wen. Trajectory tracking control of a car-trailer system. Control Systems Technology, IEEE Transactions on, 5(3):269 –278, may 1997.

[16] J. Engel, T. Schöps, and D. Cremers. LSD-SLAM: Large-scale direct monocular SLAM. In European Conference on Computer Vision (ECCV), September 2014.

[17] J. Engel, J. Sturm, and D. Cremers. Semi-dense visual odometry for a monocular camera. In IEEE International Conference on Computer Vision (ICCV), Sydney, Australia, December 2013.

[18] C. Engels, H. Stewenius, and D. Nister. Bundle adjustment rules. In Photogrammetric Computer Vision, 2006.

[19] Ryan Eustice, Hanumant Singh, John Leonard, Matthew Walter, and Robert Ballard. Visually navigating the rms titanic with slam information filters. In Proceedings of Robotics: Science and Systems, Cambridge, USA, June 2005.

[20] Ryan Eustice, Hanumant Singh, John Leonard, Matthew Walter, and Robert Ballard. Visually navigating the rms titanic with slam information filters. In Proceedings of Robotics: Science and Systems, Cambridge, USA, June 2005.

[21] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM, 24(6):381–395, 1981.

[22] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. Svo: Fast semi-direct monocular visual odometry. In Proc. IEEE Intl. Conf. on Robotics and Automation, 2014.

[23] Christian Forster, Matia Pizzoli, and Davide Scaramuzza. SVO: Fast semi-direct monocular visual odometry. In IEEE International Conference on Robotics and Automation (ICRA), 2014.

[24] U. Frese. A proof for the approximate sparsity of slam information matrices. In Proceedings of the 2005 IEEE International Conference on Robotics and Automation, pages 329–335, April 2005.

[25] P. Furgale, T.D. Barfoot, and G. Sibley. Continuous-time batch estimation using temporal basis functions. In Robotics and Automation (ICRA), 2012 IEEE International Conference on, pages 2088 –2095, may 2012.

[26] A. Gelb. Applied Optimal Estimation. MIT Press, Cambridge, MA, 1974.

[27] G. Genta. Motor Vehicle Dynamics: Modeling and Simulation. Advances in Fuzzy Systems. World Scientific Publishing Company, Incorporated, 1997.

[28] P. Grieder, F. Borrelli, F. Torrisi, and M. Morari. Computation of the constrained infinite time linear quadratic regulator. In American Control Conference, 2003. Proceedings of the 2003, volume 6, pages 4711–4716 vol.6, 2003.

[29] Chris Harris and Mike Stephens. A combined corner and edge detector. In Alvey vision conference, volume 15, page 50. Manchester, UK, 1988.

[30] R. I. Hartley and A. Zisserman. Multiple View Geometry in Computer Vision. Cambridge University Press, ISBN: 0521540518, second edition, 2004.

[31] J. A. Hesch, D. G. Kottas, S. L. Bowman, and S. I. Roumeliotis. Camera-imu-based localization: Observability analysis and consistency improvement. International Journal of Robotics Research, 33:182–201, 2013.

[32] Joel A. Hesch, Dimitrios G. Kottas, Sean L. Bowman, and Stergios I. Roumeliotis. Towards consistent vision-aided inertial navigation. In Workshop on Algorithmic Foundations of Robotics, volume 86 of Springer Tracts in Advanced Robotics, pages 559–574. Springer, 2012.

[33] Anders Heyden and Kalle Astrom. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 1997, pages 438–443. IEEE, 1997.

[34] Anders Heyden and Kalle Astrom. Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 1997, pages 438–443. IEEE, 1997.

[35] Thomas Howard, Colin Green, and Alonzo Kelly. Receding horizon model-predictive control for mobile robot navigation of intricate paths. In Proceedings of the 7th International Conferences on Field and Service Robotics, July 2009.

[36] Thomas M. Howard and Alonzo Kelly. Optimal rough terrain trajectory generation for wheeled mobile robots. Int. J. Rob. Res., 26(2):141–166, February 2007.

[37] Tor A. Johansen and N-Trondheim Norway. Computation of lyapunov functions for smooth nonlinear systems using convex optimization. Automatica, 36:1617–1626, 1999.

[38] E. Jones. Large Scale Visual Navigation and Community Map Building. PhD thesis, University of California at Los Angeles, June 2009.

[39] E. Jones, A. Vedaldi, and S. Soatto. Inertial structure from motion with autocalibration. In ICCV Workshop on Dynamical Vision, 2007.

[40] M. Kaess. Incremental Smoothing and Mapping. Ph.D., Georgia Institute of Technology, December 2008.

[41] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J.J. Leonard, and F. Dellaert. iSAM2: Incremental smoothing and mapping using the Bayes tree. Intl. J. of Robotics Research, IJRR, 31(2):217–236, Feb 2012.

[42] Y. Kanayama, Y. Kimura, F. Miyazaki, and T. Noguchi. A stable tracking control method for an autonomous mobile robot. In Robotics and Automation, 1990. Proceedings., 1990 IEEE International Conference on, pages 384 –389 vol.1, may 1990.

[43] S. Karaman and E. Frazzoli. Incremental sampling-based algorithms for optimal motion planning. In Robotics: Science and Systems (RSS), Zaragoza, Spain, June 2010.

[44] Nima Keivan and Gabe Sibley. Asynchronous adaptive conditioning for visual-inertial slam. In Proceedings of the International Symposium on Experimental Robotics (ISER), 2014.

[45] Nima Keivan and Gabe Sibley. Asynchronous adaptive conditioning for visual-inertial slam. In Proceedings of the International Symposium on Experimental Robotics (ISER), 2014.

[46] Nima Keivan and Gabe Sibley. Constant-time monocular self-calibration. In Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO), 2014.

[47] Nima Keivan and Gabe Sibley. Asynchronous adaptive conditioning for visualinertial slam. The International Journal of Robotics Research, 34(13):1573–1589, 2015.

[48] Alonzo Kelly. Trajectory generation for car-like robots using cubic curvature polynomials. Pittsburgh, PA, June 2001.

[49] Alonzo Kelly. A vector algebra formulation of kinematics of wheeled mobile robots. Technical Report CMU-RI-TR-10-33, Robotics Institute, Pittsburgh, PA, August 2010.

[50] J. Kelly and G. S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration". International Journal of Robotics Research, 2010.

[51] Jonathan Kelly and Gaurav S. Sukhatme. Visual-inertial sensor fusion: Localization, mapping and sensor-to-sensor self-calibration. International Journal of Robotics Research, pages 56–79, 2011.

[52] Seock-Ho Kim and Allan S Cohen. On the behrens-fisher problem: A review. Journal of Educational and Behavioral Statistics, 23(4):356–377, 1998.

[53] G. Klein and D. Murray. Parallel tracking and mapping for small ar workspaces. In International Symposium on Mixed and Augmented Reality, 2007.

[54] Georg Klein and David Murray. Parallel tracking and mapping for small ar workspaces. In Proceedings of the 2007 6th IEEE and ACM International Symposium on Mixed and Augmented Reality, ISMAR '07, pages 1–10, Washington, DC, USA, 2007. IEEE Computer Society.

[55] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

[56] Stefan Leutenegger, Paul Timothy Furgale, Vincent Rabaud, Margarita Chli, Kurt Konolige, and Roland Siegwart. Keyframe-based visual-inertial slam using nonlinear optimization. In Robotics: Science and Systems, 2013.

[57] M. Li, H. Yu, X. Zheng, and A. I. Mourikis. High-fidelity sensor modeling and calibration in vision-aided inertial navigation. In Proceedings of the IEEE International Conference on Robotics and Automation, pages 409–416, Hong Kong, May 2014.

[58] Mingyang Li and Anastasios I Mourikis. 3-d motion estimation and online temporal calibration for camera-imu systems. In Proc. IEEE International Conference on Robotics and Automation (ICRA), 2013.

[59] Mingyang Li and Anastasios I Mourikis. High-precision, consistent ekf-based visual–inertial odometry. The International Journal of Robotics Research, 32(6):690–711, 2013.

[60] Manolis I. A. Lourakis and Rachid Deriche. Camera self-calibration using the kruppa equations and the svd of the fundamental matrix: The case of varying intrinsic parameters. In Research Report, INRIA SOPHIA-ANTIPOLIS, 2000.

[61] Manolis I. A. Lourakis and Rachid Deriche. Camera self-calibration using the kruppa equations and the svd of the fundamental matrix: The case of varying intrinsic parameters. In Research Report, INRIA SOPHIA-ANTIPOLIS, 2000.

[62] David G. Lowe. Object recognition from local scale-invariant features. In Proceedings of the International Conference on Computer Vision-Volume 2 - Volume 2, ICCV '99, pages 1150–, Washington, DC, USA, 1999. IEEE Computer Society.

[63] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc.

[64] Todd Lupton and Salah Sukkarieh. Visual-inertial-aided navigation for high-dynamic motion in built environments without initial conditions. Trans. Rob., 28(1):61–76, February 2012.

[65] Agostino Martinelli, Davide Scaramuzza, and Roland Siegwart. Automatic self-calibration of a vision system during robot motion. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 43–48. IEEE, 2006.

[66] Agostino Martinelli, Davide Scaramuzza, and Roland Siegwart. Automatic self-calibration of a vision system during robot motion. In Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on, pages 43–48. IEEE, 2006.

[67] P. S. Maybeck. Stochastic Models, Estimation, and Control, volume 141 of Mathematics in Science and Engineering. Academic Press, Inc, Boston, 1979.

[68] Christopher Mei, Gabe Sibley, Mark Cummins, Paul M. Newman, and Ian D. Reid. Rslam: A system for large-scale mapping in constant-time using stereo. International Journal of Computer Vision, 94(2):198–214, 2011.

[69] Jean-François Menudet, Jean-Marie Becker, Thierry Fournel, and Catherine Mennessier. Plane-based camera self-calibration by metric rectification of images. Image and Vision Computing, 26(7):913–934, 2008.

[70] Jan michael Frahm and Reinhard Koch. Camera calibration with known rotation. In In Proceedings of IEEE Int. Conf. Computer Vision ICCV, pages 1418–1425, 2003.

[71] Jan michael Frahm and Reinhard Koch. Camera calibration with known rotation. In In Proceedings of IEEE Int. Conf. Computer Vision ICCV, pages 1418–1425, 2003.

[72] Isaac Miller, Sergei Lupashin, Noah Zych, Pete Moran, Brian Schimpf, Aaron Nathan, and Ephrahim Garcia. Cornell university's 2005 darpa grand challenge entry. J. Field Robotics, 23(8):625–652, 2006.

[73] Faraz M. Mirzaei and Stergios I. Roumeliotis. A kalman filter-based algorithm for imu-camera calibration: Observability analysis and performance evaluation. IEEE Transactions on Robotics, 24(5):1143–1156, 2008.

[74] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit. Fastslam: A factored solution to the simultaneous localization and mapping problem. In Proceedings of the 18th National Conference on Artificial Intelligence (AAAI), pages 593–598, July 2002. Stresses the conditional independence of observations (z) if the state of the robot is known from time 1:t. As in, given the state, the measurement model can be calculated independently for each landmark. (This assumes that the correspondence between landmarks and measurements is also known).

[75] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular slam. In Proceedings of Robotics: Science and Systems, Philadelphia, USA, August 2006.

[76] J. Montiel, J. Civera, and A. Davison. Unified inverse depth parametrization for monocular slam. In Proceedings of Robotics: Science and Systems, Philadelphia, USA, August 2006.

[77] E. Mouragnon, M. Lhuillier, M. Dhome, F. Dekeyse, and P. Sayd. Real time localization and 3d reconstruction. In Proceedings of Computer Vision and Pattern Recognition, New York, New York, June 2006.

[78] Anastasios I Mourikis and Stergios I Roumeliotis. A multi-state constraint kalman filter for vision-aided inertial navigation. In Proc. IEEE International Conference on Robotics and Automation, pages 3565–3572. IEEE, 2007.

[79] E. D. Nerurkar, Kejian J. Wu, and S. I. Roumeliotis. C-klam: Constrained keyframe localization and mapping for long-term navigation. In IEEE International Conference on Robotics and Automation Workshop on Long-term Autonomy, 2013.

[80] Richard A. Newcombe, Steven J. Lovegrove, and Andrew J. Davison. Dtam: Dense tracking and mapping in real-time. In Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, pages 2320–2327, Washington, DC, USA, 2011. IEEE Computer Society.

[81] Steven Lovegrove Nima Keivan and Gabe Sibley. A holistic framework for planning, real-time control and model learning for high-speed ground vehicle navigation over rough 3d terrain. IROS, 2012.

[82] H.B. Pacejka and Society of Automotive Engineers. Tire and Vehicle Dynamics. SAE-R. Society of Automotive Engineers, Incorporated, 2006.

[83] Junyong Park and Bimal Sinha. Some aspects of multivariate behrens-fisher problem. Bulletin of the Calcutta Statistical Association, 61(241):125, 2009.

[84] Jon W. Tollea2 Paul T. Boggsa. Sequential quadratic programming. Acta Numerica, 4:1–51, January 1995.

[85] T. Pietzsch. Efficient feature parameterisation for visual SLAM using inverse depth bundles. In British Machine Vision Conference, 2008.

[86] Mihail Pivtoraiko and Alonzo Kelly. Kinodynamic motion planning with state lattice motion primitives. In Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, 2011.

[87] Matia Pizzoli, Christian Forster, and Davide Scaramuzza. REMODE: Probabilistic, monocular dense reconstruction in real time. In IEEE International Conference on Robotics and Automation (ICRA), 2014.

[88] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In International Journal of Computer Vision, pages 7–25, 1999.

[89] Marc Pollefeys, Reinhard Koch, and Luc Van Gool. Self-calibration and metric reconstruction in spite of varying and unknown internal camera parameters. In International Journal of Computer Vision, pages 7–25, 1999.

[90] Marc Pollefeys, Luc Van Gool, and Marc Proesmans. Euclidean 3d reconstruction from image sequences with variable focal lengths. In Computer VisionECCV'96, pages 31–42. Springer, 1996.

[91] Marc Pollefeys, Luc Van Gool, and Marc Proesmans. Euclidean 3d reconstruction from image sequences with variable focal lengths. In Computer VisionECCV'96, pages 31–42. Springer, 1996.

[92] M. Powell. Numerical Methods for Nonlinear Algebraic Equations. Gordon and Breach Science, 1970.

[93] M. Powell. Numerical Methods for Nonlinear Algebraic Equations. Gordon and Breach Science, 1970.

[94] R. Rajamani. Vehicle Dynamics and Control. Mechanical Engineering Series. Springer, 2011.

[95] Autonomous Robotics and Perception Group. Calibu camera calibration library. `http://github.com/arpg/calibu`. [Online; accessed September-2014].

[96] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In Proceedings of the 9th European Conference on Computer Vision - Volume Part I, ECCV'06, pages 430–443, Berlin, Heidelberg, 2006. Springer-Verlag.

[97] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: An efficient alternative to sift or surf. In Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, pages 2564–2571, Washington, DC, USA, 2011. IEEE Computer Society.

[98] Keir Mierle Sameer Agarwal. Ceres Solver: Tutorial & Reference. Google, 2012 November.

[99] T. W. Sederberg. Computer Aided Geometric Design. Brigham Young University, April 2007.

[100] Neal Seegmiller, Forrest Rogers-Marcovitz, and Alonzo Kelly. Online calibration of vehicle powertrain and pose estimation parameters using integrated dynamics. In IEEE International Conference on Robotics and Automation, May 2012.

[101] Jianbo Shi and Carlo Tomasi. Good features to track. In 1994 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'94), pages 593 – 600, 1994.

[102] Ken Shoemake. Animating rotation with quaternion curves. SIGGRAPH Comput. Graph., 19(3):245–254, July 1985.

[103] G. Sibley. Sliding window filters for SLAM. Technical report, University of Southern California, Center for Robotics and Embedded Systems, CRES-06-004, 2006.

[104] G. Sibley, L. Matthies, and G. Sukhatme. Sliding window filter with applications to planetary landing. Journal of Field Robotics, 27(5):587–608, September/October 2010.

[105] G. Sibley, C. Mei, I. Ried, and P. Newman. Adaptive relative bundle adjustment. In Robotics: Science and Systems, 2009.

[106] Gabe Sibley. Relative visual inertial integration and calibration. Technical report, University of Oxford, Parks Road, OX13PJ, August 2011.

[107] Gabe Sibley, Larry Matthies, and Gaurav Sukhatme. Sliding window filter with application to planetary landing. Journal of Field Robotics, 27(5):587–608, 2010.

[108] R. Smith, M. Self, and P. Cheeseman. Autonomous robot vehicles. chapter Estimating uncertain spatial relationships in robotics, pages 167–193. Springer-Verlag New York, Inc., New York, NY, USA, 1990.

[109] Noah Snavely, Steven M. Seitz, and Richard Szeliski. Modeling the world from internet photo collections. Int. J. Comput. Vision, 80(2):189–210, November 2008.

[110] Alex Stewart and Paul Newman. Laps - localisation using appearance of prior structure: 6-dof monocular camera localisation using prior pointclouds. In Proc. IEEE International Conference on Robotics and Automation (ICRA), Minnesota, USA, May 2012.

[111] Hauke Strasdat. Local Accuracy and Global Consistency for Efficient Visual SLAM. PhD thesis, Imperial College London, 2012.

[112] Peter Sturm. On Focal Length Calibration from Two Views. In IEEE International Conference on Computer Vision and Pattern Recognition, volume 2, pages 145–150. IEEE Computer Society, 2001.

[113] Peter Sturm. On Focal Length Calibration from Two Views. In IEEE International Conference on Computer Vision and Pattern Recognition, volume 2, pages 145–150. IEEE Computer Society, 2001.

[114] Peter F. Sturm and Stephen J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In Computer Vision and Pattern Recognition (CVPR), pages 432–437, 1999.

[115] Peter F. Sturm and Stephen J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In Computer Vision and Pattern Recognition (CVPR), pages 432–437, 1999.

[116] R. Tedrake. LQR-trees: Feedback motion planning on sparse randomized trees. In Proceedings of Robotics: Science and Systems, Seattle, USA, June 2009.

[117] Sebastian Thrun. A probabilistic online mapping algorithm for teams of mobile robots. International Journal of Robotics Research, 20:2001, 2001.

[118] Carlo Tomasi and Takeo Kanade. Detection and tracking of point features. Technical report, International Journal of Computer Vision, 1991.

[119] Bill Triggs, Philip McLauchlan, Richard Hartley, and Andrew Fitzgibbon. Bundle adjustment – a modern synthesis. In Vision Algorithms: Theory and Practice, LNCS, pages 298–375. Springer Verlag, 2000.

[120] Ying Yao. An approximate degrees of freedom solution to the multivariate behrens fisher problem. Biometrika, pages 139–147, 1965.

[121] C. Zach, T. Pock, and H. Bischof. A duality based approach for realtime tv-l1 optical flow. In Proceedings of the 29th DAGM conference on Pattern recognition, pages 214–223, Berlin, Heidelberg, 2007. Springer-Verlag.

[122] Matthew Zucker, Nathan Ratliff, Anca Dragan, Mihail Pivtoraiko, Matthew Klingensmith, Christopher Dellin, J. Andrew (Drew) Bagnell, and Siddhartha Srinivasa. Chomp: Covariant hamiltonian optimization for motion planning. International Journal of Robotics Research, May 2013.