Multi-Session Periodic SLAM for Legged Robots

Hans Kumar

CMU-RI-TR-21-04

The Robotics Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania

Thesis Committee

Prof. Howie Choset (Co-Advisor)
Prof. Matthew Travers (Co-Advisor)
Prof. Michael Kaess
Shuo Yang

In Partial Fulfillment of the Requirements for the
Degree of Master of Science in Robotics

ACKNOWLEDGEMENTS

I would like to first thank my advisors Matthew Travers and Howie Choset for their invaluable support and guidance each week throughout my research journey. They always encouraged me to look at my research through a critical lens and become the best researcher that I could be.

Next, I would like to thank all the members of the Biorobotics Lab for making the lab an enjoyable and constructive place to work. All of the thought-provoking discussions in the lab helped me to think of my best ideas. A list of my peers whom I would especially like to thank includes Shuo Yang, John Payne, Raunaq Bhirangi, Benjamin Freed, Roshan Pradhan, Daniel Vedova, Johnathan Cassady, and Abhimanyu.

Finally, I would like to thank my parents Rakesh and Manpreet for their constant support and guidance. Without them, I would not be where I am today.

ABSTRACT

Methods for state estimation that rely on visual information are challenging on dynamic robots because of rapid changes in the viewing angle of onboard cameras. In this thesis, we show that by leveraging structure in the way that dynamic robots locomote, we can increase the feasibility of performing state estimation despite these challenges. We present a method that takes advantage of the underlying periodic predictability that is often present in the motion of legged robots to improve the performance of the feature tracking module within a visual-inertial SLAM system. Inspired by previous work on coordinated mapping with multiple robots, our method performs multi-session SLAM on a single robot, where each session is responsible for mapping during a distinct portion of the robot's gait cycle. Our method outperforms several state-of-the-art methods for visual and visual-inertial SLAM in both a simulated environment and on data collected from a real-world quadrupedal robot.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

Chapter 1

INTRODUCTION

While there has been tremendous progress in the development of state estimation and SLAM algorithms in the last several decades, dynamic motion can still induce failure on even the most robust systems [2]. More specifically, we find that methods for state estimation and SLAM that rely on visual information experience a significant decrease in the performance of visual feature tracking when there are rapid changes in the viewing angle of cameras on-board a robot. To combat this issue, our approach brings together the often decoupled areas of SLAM and nonlinear analysis of dynamical systems. In this thesis, we present a method that leverages the periodic stability of a dynamical system to introduce beneficial structure within an existing approach to solving SLAM.

Legged robots are of particular interest to us in this work because they are examples of dynamical systems that maintain periodic structure in their motion when executing gait-like behaviors. When locomoting with dynamically stable gaits, such as walking or running on flat ground, legged robots exhibit patterns in their footfall and resulting body orientation. Although periodic, nonlinear impact events and rapid orientation changes have typically been thought of as hindrances to performing SLAM on legged systems, our method proposes using the predictability in these events to improve the performance of estimation compared to an otherwise naive approach.

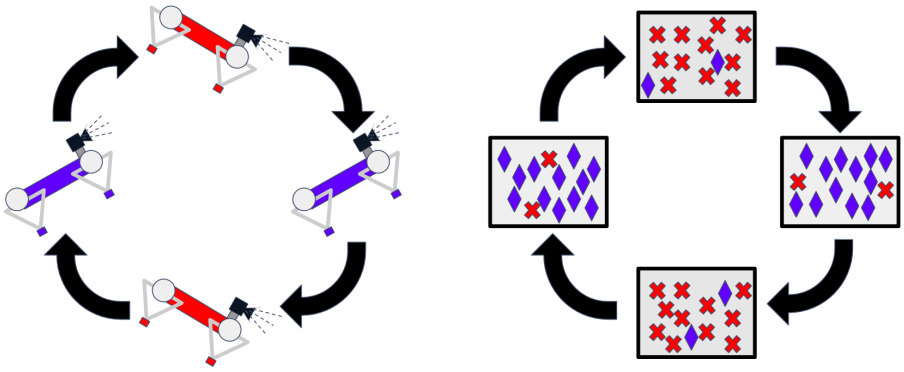

Figure 1.1: Periodicity in legged locomotion. Different colors in this diagram represent different portions of the robot's periodic gait cycle. As the robot moves through different phases of its gait cycle (left), the visual measurements of the world that it receives (right) also follow a periodic pattern.

In this thesis, we present a novel framework for visual-inertial SLAM that exploits the periodic predictability in the visual information obtained by the robot, shown in Figure 1.1, to perform estimation despite rapid changes in the viewpoint of on-board cameras. Our approach explicitly differentiates between visual features detected during each unique section of the robot's gait cycle, when visual information is more likely to be similar. By performing visual SLAM separately on each portion of the robot's gait cycle, we are able to improve the performance of the feature tracking module that is critical to the success of visual SLAM.

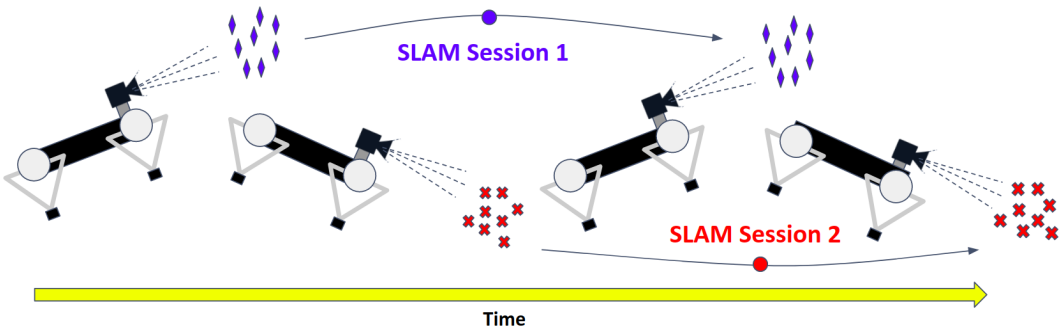

Figure 1.2: An overview of our proposed method that introduces multiple SLAM sessions to track a periodically moving legged robot

In Figure 1.2 we show an example of how our method might introduce two different SLAM sessions on a legged robot: one for when the robot is looking upwards and one for when the robot is looking downwards. Our approach produces a unified SLAM estimate from these individual visual SLAM sessions by incorporating measurements from an IMU sensor attached to the robot. By tightly coupling visual and inertial measurements in a factor graph based optimization framework, our approach achieves a greater combined performance than any individual SLAM session.

We implement our method on a dynamic robot in simulation and on a real-world quadrupedal robot. Furthermore, we compare the performance of our approach against several different state-of-the-art implementations of visual and visual-inertial SLAM. Our experimental results show that when compared to the baseline methods, our approach has clear improvement in estimation accuracy especially when the speed of the robot's motion becomes faster. Lastly, we conclude by noting future directions for using periodicity to improve the performance of SLAM and the applicability of this work to non-periodic systems.

The main contributions of this work are as follows:

1. An evaluation of the shortcomings of several state-of-the-art visual and visual-inertial SLAM methods when dealing with periodic, dynamic motion;
2. a unique formulation of the state estimation problem on dynamic robots as a multi-session SLAM problem, where each session tracks a section of the robot's gait cycle; and,
3. an experimental validation of the above approach on both a simulated and a real-world robot.

Chapter 2

BACKGROUND

In this chapter, we introduce the fundamentals of representing and solving the problem of SLAM. We begin by providing an overview of the SLAM problem. Next, we introduce factor graphs as a popular optimization-based approach for solving SLAM. Then, we explain the front-end of visual methods for SLAM, and we discuss the problem of data association. Finally, we end this chapter with a survey of three state-of-the-art algorithms for visual and visual-inertial SLAM that we reference throughout our work.

2.1 Introduction to SLAM

When mobile robots do not have access to their ground truth position, they must rely on measurements from sensors to build an estimate for where they are in the world. Estimating the position is especially challenging when the robot is operating in an unknown environment. In such cases, the robot must estimate both its position and a map of external landmarks in a process called *simultaneously localization and mapping* (SLAM) [2].

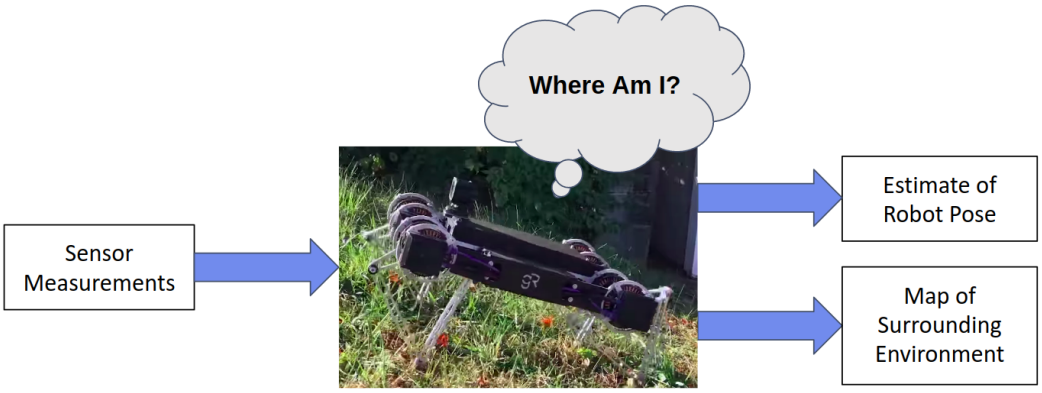

Figure 2.1: An overview of the SLAM problem. In SLAM, the robot must use sensor measurements to deduce its location while simultaneously building a map of the world.

To capture the inherent uncertainty of events in the world, many implementations of SLAM model the robot and landmark locations in the world as a joint probability distribution [20]. We can write the conditional distribution of the unknown robot states (S) and landmark states (L) given a set of known sensor measurements (Z) as

$P(S, L|Z)$. Here, we define $S = [s_1, s_2 \dots s_t]$ to represent the set of all robot poses over time, and we define $L = [l_1, l_2, \dots l_n]$ to represent the positions of mapped landmarks. Likewise, we define $Z = [z_1, z_2 \dots z_t]$ to represent the set of all sensor measurements that the robot receives over time.

The goal of SLAM is to obtain an estimate for the unknown variables (S and L) by performing optimization-based probabilistic inference. Specifically, many SLAM methods look to maximize the conditional probability density of the set of unknown random variables, the robot and map state, given the set of known random variables, the sensor measurements [6]. The values that maximize this probability density and the solution to SLAM are called the *maximum a posteriori* (MAP) estimate:

$$S_{MAP}, L_{MAP} = \underset{S,L}{\operatorname{argmax}}\, P(S, L|Z) \tag{2.1}$$

2.2 Factor Graphs for SLAM

Contemporary approaches for solving the problem of SLAM rely on sparse factor graph based optimization [13]. Factor graphs are a class of graphical models that are useful in representing sparsity within the distribution, $P(S, L|Z)$, in order to enable efficient MAP inference. More precisely, factor graphs are a type of bipartite graph that consists of two types of nodes: factors and variables [6]. In the context of SLAM, variable nodes are used to represent the unknown, latent robot and landmark states we wish to estimate: (S, L), and factors are used to represent constraints (specified by sensor measurements) between states: $\phi(\cdot) : (S, L) \rightarrow \mathbb{R}$. A factor graph for an example SLAM problem in which a robot receives relative odometry measurements and visual measurements to two landmarks in the world is shown in Figure 2.2:

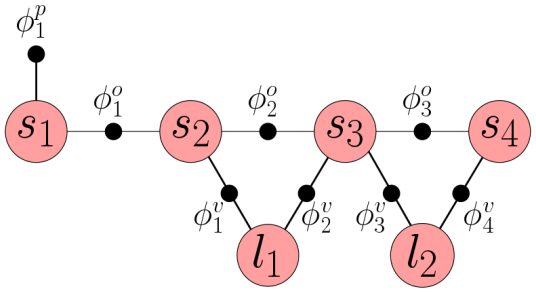

Figure 2.2: Factor graph for an example SLAM problem. Unknown variable nodes are in red and and known factors nodes are smaller black dots. Sequential robot states (s_i) are connected with odometry factors (ϕ_i^o), and visual measurement factors (ϕ_i^v) constraint the robot to two different landmarks (l_i) Lastly, a prior factor (ϕ_1^p) is included to constrain the robot's initial location in the world.

Factor Graphs are useful because they model the sparse structure inherent to SLAM by specifying conditional independence relationships between variables. Using these assumptions of conditional independence, factor graphs succinctly specify the complex joint probability density $P(S, L|Z)$ as the product of more easily computable factors. Each factor represents a smaller probability density that is only a function of the variables to which it is connected in the graphical model. Below, we show the corresponding factorization of the example factor graph in Figure 2.2 after using Bayes rule [6]:

$$\begin{aligned} P(S, L|Z) \propto P(Z|S, L)P(S, L) = p(s_1) \\ \times p(z_1^o|s_1, s_2)p(z_2^o|s_2, s_3)p(z_3^o|s_3, s_4) \\ \times p(z_1^v|s_2, l_1)p(z_2^v|s_3, l_1)p(z_3^v|s_3, l_2)p(z_4^v|s_4, l_2) \end{aligned} \tag{2.2}$$

Here we introduce z_i^v to represent a visual measurement between the robot and a landmark, and we use z_i^o to represent an odometry measurement for the robot. Each of the terms in the factorization of Equation 2.2 correspond to the factors ϕ_i in the graphical model. To make this relationship more explicit, we provide a line by line explanation below:

1. Line 1 contains a prior on the robot state:

   $\phi_1^p \propto p(s_1)$

2. Line 2 contains odometry factors between sequential robot states

   $\phi_1^o \propto p(z_1^o|s_1, s_2)$, $\phi_2^o \propto p(z_2^o|s_2, s_3)$, $\phi_3^o \propto p(z_3^o|s_3, s_4)$

3. Line 3 contains visual measurement factors between robot states and landmarks:

   $\phi_1^v \propto p(z_1^v|s_2, l_1)$, $\phi_2^v \propto p(z_2^v|s_3, l_1)$, $\phi_3^v \propto p(z_3^v|s_3, l_2)$, $\phi_4^v \propto p(z_4^v|s_4, l_2)$

We use $\propto$ instead of $=$ in our factorization above because in general factors do not have to be proper probability densities; they can be any local function of the variables they are connected to. In practice, we usually use factors to represent likelihood functions, which are proportional to but not equal to proper probability density functions. More background on this can be found in [6].

2.2.1 Nonlinear MAP Inference

Using the newly factored joint probability distribution, we can now efficiently solve for the MAP estimate of the robot and landmark states. Below, we adapt Equation 2.1 by using the factorized probability density.

$$S_{MAP}, L_{MAP} = \underset{S,L}{\operatorname{argmax}} \prod_i \phi_i(S, L) \tag{2.3}$$

In order to solve this equation, we must now specify the form for each of our factors. It is convenient to specify that the probability density functions for our factored probabilities be multivariate Gaussian distributions. Using this specification, the visual measurement factor ϕ_1^v from Figure 2.2 with measurement covariance Σ^v would have the following probability density:

$$\begin{aligned} p(z_1^v | s_2, l_1) &\sim \mathcal{N}(h(s_2, l_1), \Sigma^v) \\ \phi_1^z &\propto exp(-\frac{1}{2} \|h(s_2, l_1) - z_1\|^2_{\Sigma^v}) \end{aligned} \tag{2.4}$$

where the Mahalanobis norm is calculated as[6]: $\|e\|^2_\Sigma = e^T \Sigma^{-1} e$

For the visual measurement factor above, we note that the measurement function, h, could be a camera projection function. However, in general h could be a dynamics function for an odometry factor or the identity function for a prior factor. Plugging in Gaussian densities in the form of Equation 2.4 into the MAP optimization problem and simplifying by taking the negative logarithm we get the following:

$$\begin{aligned} S_{MAP}, L_{MAP} &= \underset{S,L}{\operatorname{argmax}} \prod_i exp(-\frac{1}{2} \|h_i(S, L) - z_i\|^2_{\Sigma_i}) \\ S_{MAP}, L_{MAP} &= \underset{S,L}{\operatorname{argmin}} \sum_i \|h_i(S, L) - z_i\|^2_{\Sigma_i} \end{aligned} \tag{2.5}$$

Since the measurement functions h_i are typically nonlinear, we must perform a linearization of them and do iterative optimization to solve for the MAP estimate. Using a Taylor expansion and following the derivation from [6] closely, we are able to arrive at an update equation which is linear with respect to a state update vector, Δ. We can then solve for an optimal update to an initial guess of the robot and landmark states, S^0 and L^0, as follows:

$$\Delta* = \underset{\Delta}{\operatorname{argmin}} \sum_i \left\| H_i \Delta_i + h_i(S^0, L^0) - z_i \right\|^2_{\Sigma_i} \tag{2.6}$$

In the Equation 2.6, we use H_i to denote the Jacobian of the measurement function h_i. With some more simplification of this expression, we are able to arrive at the standard least squares problem:

$$\Delta* = \underset{\Delta}{\operatorname{argmin}} \sum_i \|A\Delta_i - b\|_2^2$$

where:

$$A = \Sigma_i^{-1/2} H_i$$
$$b = \Sigma_i^{-1/2}(z_i - h_i(S^0, L^0)) \tag{2.7}$$

To obtain a MAP estimate, we can solve this least squares problem iteratively for $\Delta*$ and update the initial guess of the robot and landmark state (S^0 and L^0) at each iteration. In practice, performing this nonlinear optimization is done using the Gauss-Newton or Levenberg-Marquardt [23] techniques. Furthermore, because we built up this problem using a factor graph, there exist a number of efficient methods to solve this least-squares problem by exploiting the sparse matrix structure of the measurement Jacobian, H [6].

2.3 Front-End of SLAM and Visual Data Association

In the past, SLAM was often performed on mobile robots with laser range sensors, sonar sensors, and wheel encoders [6]. More recently, however, researchers have shown interest in developing camera-only or camera-and-IMU-only algorithms because of their simplicity of sensor configuration [43]. Visual-inertial SLAM methods use images, usually from one (mono) or two (stereo) cameras, as well as gyroscope and accelerometer data from an inertial measurement unit (IMU) sensor to estimate robot motion and maintain a map of the environment. Combining the rich visual information from images with high-frequency IMU measurements has been shown to promise robustness in challenging situations [25]. Because of the simplicity and complementary properties of cameras and IMUs, the rest of our work is focused on performing SLAM specifically with this set of sensors.

In this section, we introduce the front-end of SLAM as it pertains to methods that rely on visual information. We discuss the problem of visual data association, which involves determining how to specify the measurements used in the back-end of SLAM from raw visual data.

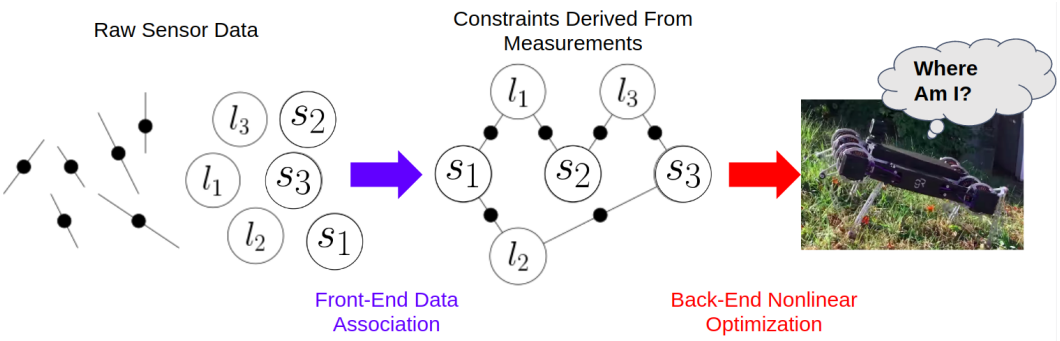

Figure 2.3: Responsibilities of the front-end and the back-end of SLAM. In the case of factor graph based methods, the front-end of SLAM associates factors with different variable nodes. Then, the back-end of SLAM optimizes the graph constructed by the front-end to output a MAP estimate of the robot and map states.

2.3.1 Feature Detection and Tracking

The front-end of visual SLAM methods can broadly be categorized into two categories: direct and indirect [32]. As their name implies, direct methods perform SLAM by directly measuring the change in the pixel intensities from incoming images. In this work, we instead focus on indirect methods for visual SLAM which are are characterized by an image pre-processing step, in which typically hand-crafted features are first detected in images [30]. The first step when doing visual data as-

sociation in indirect methods is to detect salient feature points in an image. Feature detection is typically done by identifying image primitives such as points, lines, and regions that can be used as visual cues [18].

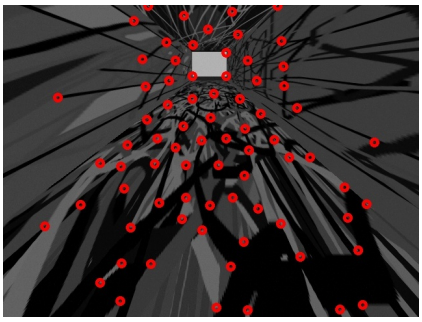

Figure 2.4: Detecting Harris corner features (red) in an image. The image is collected from a simulation environment with lines painted along the walls.

After performing feature detection on a single image, the next part of the data association pipeline is to track these features in another image received from the camera. Feature tracking usually involves performing a local search to track the detected features from the first image in a successive image [20]. The most commonly used method for doing feature tracking, proposed by [19], is the Lucas-Kanade method. This method tries to transform the positions of features in one image towards the positions of the features in the second image. While the Lucas-Kanade method is an efficient method of performing data association, it makes a key assumption that the motion between frames must not be too large. We will see in Chapter 3 that this assumption is one of the reasons visual methods for SLAM are difficult on dynamic robots.

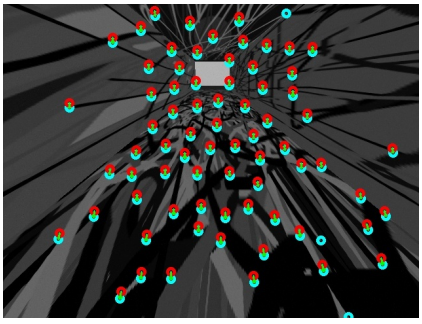

Figure 2.5: Tracking features using Lucas-Kanade Method. Red circles represent features from the previous frame, and blue circles represent tracked features in the current frame.

In this work, we focus on performing the front-end of visual SLAM exclusively with a stereo camera. While monocular camera configurations are cheaper and

more compact, it has been shown that depth is not observable from just one camera [20]. Due to this, monocular methods for SLAM are vulnerable to scale drift over time, and they may fail in the case of pure rotation [25]. Stereo methods for visual SLAM, on the other hand, are able easily to recover the depth of map points through triangulation given a known distance between two cameras. When performing visual SLAM with a stereo camera, in addition to finding feature correspondences between sequential frames, we must also track features across the images received from the left and right cameras.

After successfully executing the front-end visual SLAM by doing feature detection and feature tracking, we are able to add these correspondences as measurements in the back-end of SLAM. We summarize the pipeline for the front-end of visual SLAM in the context of the full SLAM problem in Figure 2.6:

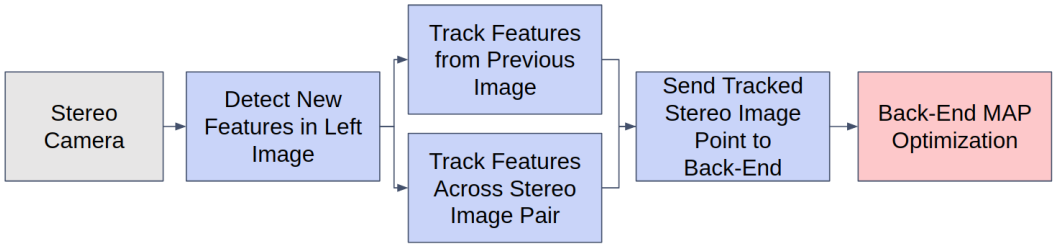

Figure 2.6: The stereo visual SLAM pipeline with a focus on the front-end

2.4 Survey of State-of-the-Art SLAM Implementations

We chose to evaluate the effect of dynamic motion on three different state-of-the-art methods for indirect visual and visual-inertial SLAM. The first method, ORB-SLAM2, provides a robust camera-only solution with multiple levels of data association. The second method, VINS-Fusion, combines camera and IMU information to perform tightly coupled visual-inertial SLAM. The last method we look at, MSCKF-VIO, also performs visual-inertial estimation, but it uses a more classical filtering-based approach as its back-end. In the rest of this section, we describe some of the notable characteristics of each of these three algorithms.

2.4.1 ORB-SLAM2

ORB-SLAM2 is one of the best modern-day implementations of feature-based visual SLAM [25] that uses ORB features for all tasks. Even though ORB-SLAM2 does not use inertial information, we include it as a baseline because of how robust its performance has been shown to be [28] [16]. Moreover, because our work focuses on how dynamic motion affects SLAM systems that rely on using visual features, we believe it is fair to include.

The implementation of ORB-SLAM2 is a sophisticated optimization-based algorithm that contains three main modules working in parallel [12]:

- a **tracking thread** that does frame to frame visual odometry and also adds a new keyframe when needed
- a **local mapping thread** that processes new keyframes and performs local optimization to achieve an optimal space reconstruction
- a **loop closure thread** that searches for large loops when a new keyframe is available

By using three tiers of visual data association, ORB-SLAM2 can maintain robust and accurate tracking even over long distances. Instead of doing optimization over every frame received by the camera, ORB-SLAM2 only uses a subset of frames, called keyframes, to perform optimization over. This technique of keyframing is very common to visual SLAM because it reduces the computational burden of graph-based optimization. Additionally, it should be noted that ORB-SLAM2 can detect loop closure events and re-localize when tracking is lost in real-time.

2.4.2 VINS-Fusion

In situations where there is agile camera movement, visual-inertial SLAM has shown great success [17]. VINS-Fusion is one of the current popular state-of-the-art implementations of stereo visual-inertial SLAM. It tightly couples visual and inertial measurements in an optimization-based estimator and runs a parallel thread for loop closures as well [27].

The authors of VINS-Fusion talk extensively about their method of IMU preintegration that can accurately summarize hundreds of inertial measurements into one single relative motion constraint to be used during optimization [8]. By preintegrating IMU information between camera frames, VINS-Fusion is able to efficiently seed the next estimated camera pose more accurately.

Another key idea used in VINS-Fusion is the use of a sliding optimization window. By only maintaining a small window of states from the past and marginalizing out the rest, VINS-Fusion can perform real-time estimation at the cost of drift due to linearization error [26].

2.4.3 MSCKF-VIO

The work presented in [24] and [29] introduces the Multi-State Constraint Kalman Filter, which performs visual-inertial odometry using an extended Kalman filter (EKF) as its back-end. Filtering-based approaches are how researchers have classically approached the problems of state estimation and SLAM. Unlike VINS-Fusion and ORB-SLAM2 which rely on graph-based optimization over a history of robot states, filtering-based approaches typically work by incrementally updating only the most current state estimate.

The MSCKF-VIO algorithm uses a two-step EKF to combine data from IMU and stereo camera sensors. In the first step, an EKF process model is used to integrate information from the IMU sensor to predict the next pose of the robot. Then, in the second step, an EKF measurement model corrects the estimate for the pose of the robot by tracking visual features from previous frames. Unlike most Kalman Filters, MSCKF-VIO maintains a brief history of previous robot states by appending robot poses to the state matrix. Once the state matrix becomes too large, this method then performs a batch marginalization step over some of the previous poses. While this method is unable to perform SLAM at as large of a scale as the other two methods in this section, we find that it is still useful to include in our comparison.

Chapter 3

WHY DYNAMIC MOTION CAUSES SLAM TO FAIL

In this chapter, we evaluate the performance of the three previously surveyed state-of-the-art algorithms for visual and visual-inertial SLAM in the presence of dynamic motion. We start by introducing a simulation environment that we created to test the performance of each of these algorithms. Then, we look at the limits of each of the surveyed algorithms in our simulation environment and hypothesize reasons for failure.

3.1 Simulation Environment

To evaluate the performance of these different SLAM approaches, we create a simple hallway environment in Gazebo. Lines of different colors are drawn onto the walls of the hallway to ensure that an abundance of visual features is available to perform feature detection and tracking.

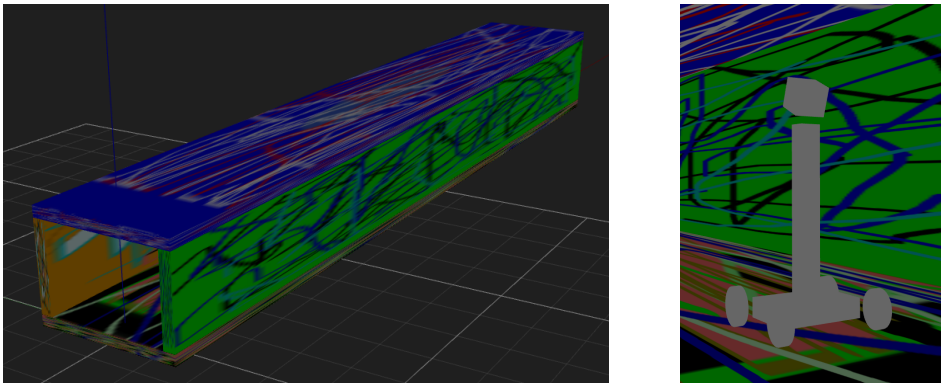

Figure 3.1: Simulated hallway environment in Gazebo. On the left the full hallway is shown with arbitrary drawings on the walls to add visual features. On the right the simulated robot is shown collecting camera and IMU data as it moves down the hallway.

While it might seem counter-intuitive at first, we use a wheeled robot in our simulation instead of a legged robot. We attach a camera and an IMU sensor to the top of this robot via a rotational and a prismatic joint to approximate a hopping and pitching motion. Using these two well-controlled degrees of freedom, the robot is able to approximate the motion of a camera and IMU sensor attached to a legged robot undergoing gait-like motion.

As the robot moves down the simulated hallway, we collect time-synchronized stereo image pairs at 30 Hz and IMU data at 250 Hz from the simulated sensors that are attached to the top of the robot. Additionally, we corrupt all of the measurements with additive Gaussian noise.

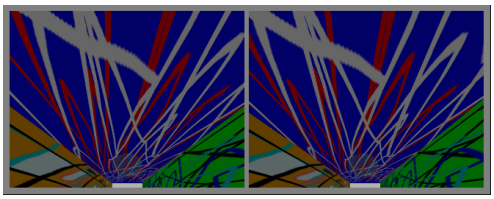

Figure 3.2: Simulated stereo image pair taken from the camera on top of the robot

We describe the motion of the robot with a set of simple periodic functions that take four parameters as input: max pitch angle ($\phi_{\max}$) in radians, max heave distance ($\delta_{\max}$) in meters, frequency of gait cycle (ω) in Hz, and forward velocity of the robot ($\dot{x}$) in meters/second. By doing this, we can modulate different parameters of the "robot's gait" to achieve varying periodic camera trajectories. We include the periodic functions used to describe the robot's motion and a schematic of the simulated robot below:

$$\begin{aligned} \phi_t &= \phi_{\max} \sin(\omega 2\pi t) \\ \delta_t &= \delta_{\max} \sin(\omega 2\pi t) \end{aligned} \tag{3.1}$$

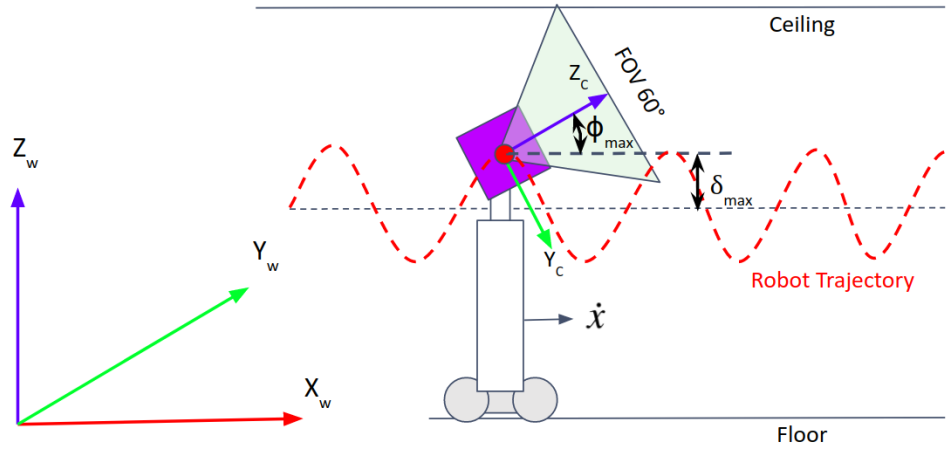

Figure 3.3: Schematic of simulated robot. The subscript "w" represents the world coordinate frame and the subscript "c" represents the camera coordinate frame. The IMU frame is aligned with the camera frame on top of the robot. The red dotted line shows the trajectory that the camera and IMU follow as the robot moves forwards.

3.2 Performance of Different SLAM Implementations in Simulation

To evaluate the performance of the state-of-the-art SLAM methods that we surveyed in Chapter 2, we used open-source implementations of these algorithms on the simulated robot. We tuned each algorithm individually and maintained the same camera and IMU parameters across each algorithm. As expected, we observed that as the simulated robot's motion became faster and more dynamic, each of the surveyed SLAM methods began to experience failure.

We validated this observation by conducting an experiment in which we measured how changing the frequency in which the robot executes a cycle of its gait, ω, affects the performance of each state-of-the-art SLAM method. We ran each SLAM algorithm on the robot as it moved forward 10 meters in the simulated hallway environment, and we modulated the robot's gait frequency from .125 Hz to 2.5 Hz. We selected some nominal parameters for the max pitch angle, max heave height, and forward velocity of the robot that we estimate would roughly correspond to a dynamic legged robot moving in the real world.

ω (Hz)	ϕ_{max} (deg)	δ_{max} (m)	$\dot{x}$ (m/s)
0.125 - 2.5	25	0.05	0.2

Table 3.1: Simulation Gait Parameters

In our experiment, we ran 50 trials at each gait frequency to account for variance between trials. In each of the trials, we recorded the estimated position of the robot outputted by each SLAM system and the ground truth position of the robot given by the simulation. To measure the performance of each SLAM system, we recorded the distance the robot traveled before losing track of where it was. Since the overall trajectory was 10 meters, we decided that if the difference between the estimated robot position and the ground truth robot position was greater than .5 meters or 5 percent of the entire trajectory, we would deem the trial a failure. While we could have had the simulated robot move over a longer distance (e.g., a 20-meter trajectory), we found that 10 meters were sufficient. We found that in most cases if the robot experienced failure, it did so within the first 5 meters of its trajectory.

Figure 3.4 shows the result of our experiment. We observe a clear trend for each algorithm showing that an increase in robot gait frequency leads to a decrease in SLAM performance. Our results show that once gait frequency exceeds 1.25 Hz, all three methods experience failure. Furthermore, each method ends up failing earlier in the robot's trajectory as gait frequency (ω) is increased.

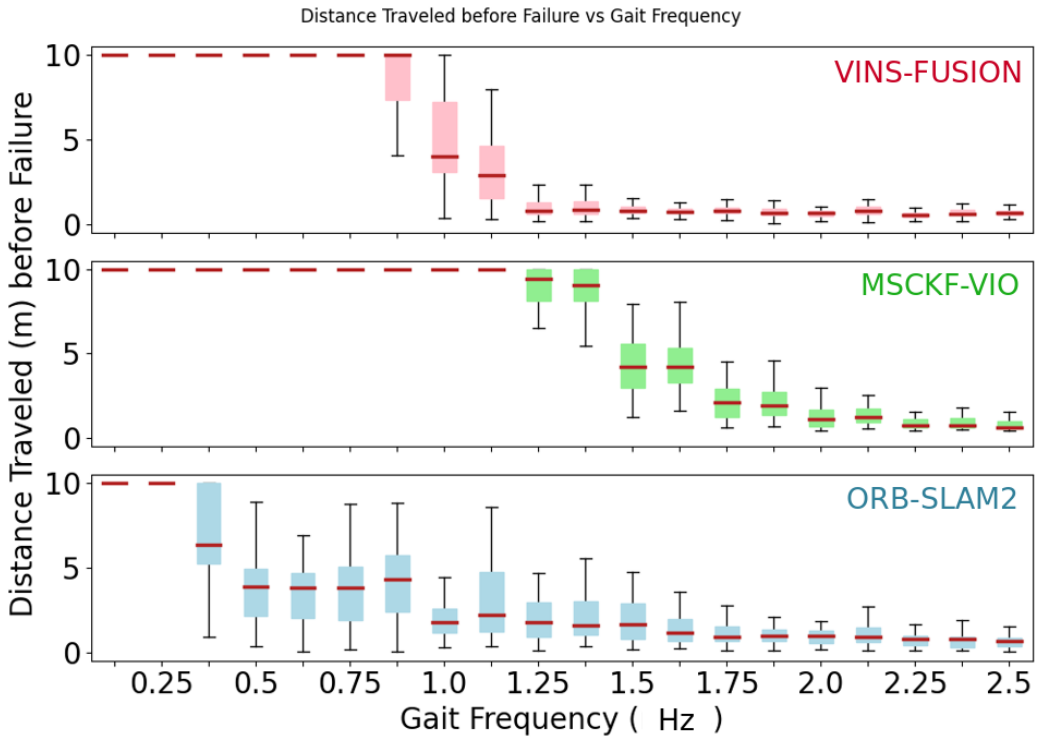

Figure 3.4: Distance Traveled (m) before failure of different SLAM algorithms vs gait frequency of robot. In this box and whiskers plot, each box represents the first to third quartile of the data. The horizontal line inside each box represents the median of the data. Finally, the whiskers show the minimum and maximum of the data.

3.3 Feature Tracking Performance

In the previous section, we established that there was a significant decrease in the performance of all three of the surveyed SLAM implementations as robot motion became faster. In this section, we hypothesize that the front-end of each of the surveyed algorithms is the common failure point. We hypothesize that the feature tracking performance of each of these algorithms directly suffers as a result of rapid motion between sequential camera frames. Furthermore, we believe that without adequate feature tracking, each algorithm is induced to fail. In Figure 3.5 below, we show a visualization of VINS-Fusion's feature tracking for three different gait frequencies.

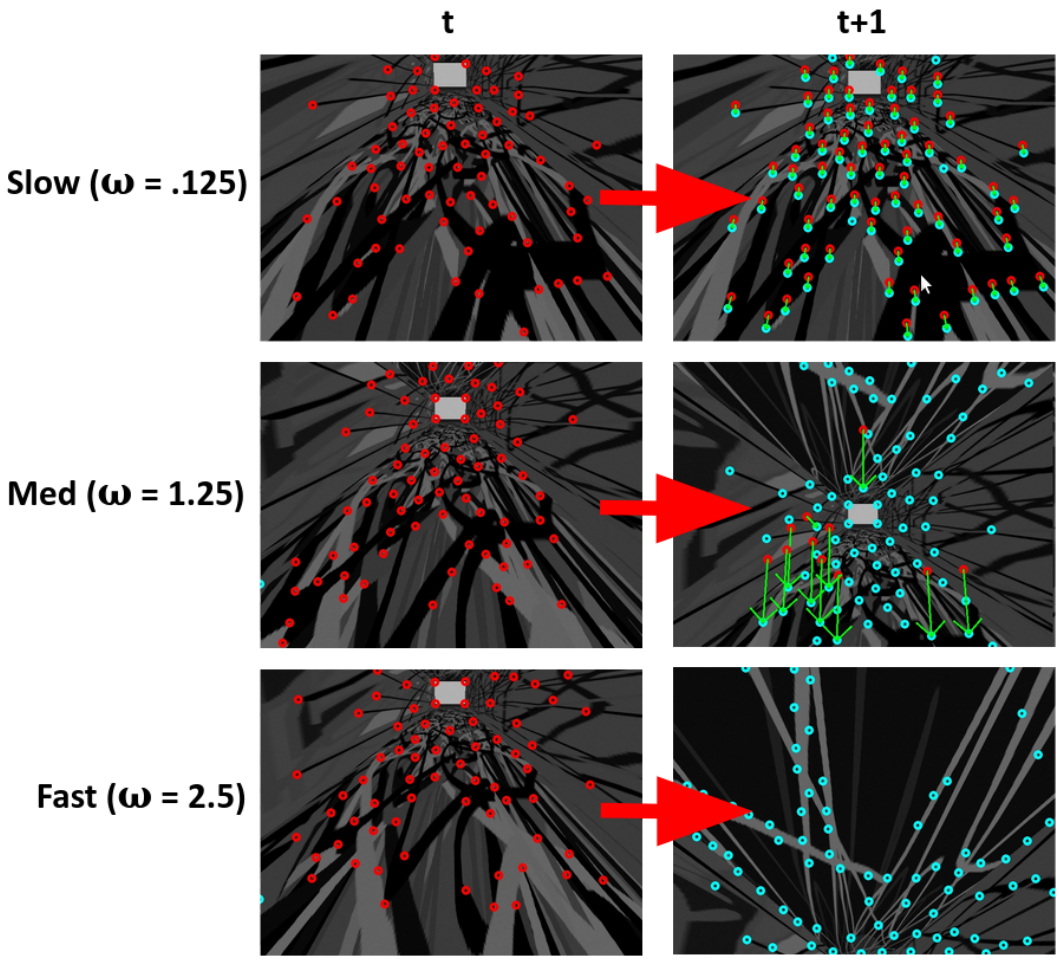

Figure 3.5: Frame-to-frame VINS-Fusion feature tracking output for three different gait frequencies. Red circles represent features detected at the first time step (t), and light blue circles represent features detected at the next time step (t+1). If feature tracking were possible, green arrows are drawn to indicate the change in position from features in one frame to the next. Quantitative results are shown in Figure 3.6.

In Figure 3.5 we can see that that as gait frequency increases, the distance that visual features move from frame to frame becomes larger. This increase in feature displacement naturally makes the problem of feature tracking more difficult, especially in the case of algorithms that track features using local search methods such as Lucas Kanade tracking [19]. Once the change in camera viewpoint from one frame to the next becomes too large, it becomes difficult or in some cases impossible to find accurate feature correspondences between frames.

We conducted another experiment to quantify the relationship between feature tracking performance and gait frequency. We ran the simulated robot in the hallway environment with the same setup as the experiment whose results are shown in Figure 3.4 using the gait parameters from Table 3.1. In this experiment, for each of the 50 trials at each gait frequency, we recorded the number of tracked features present in each image. Since each SLAM method uses a different number of features, we divided this number by the maximum number of potential candidate features that could have been tracked. We show the distribution for this metric across all of the image frames used for each gait frequency in the box plot below:

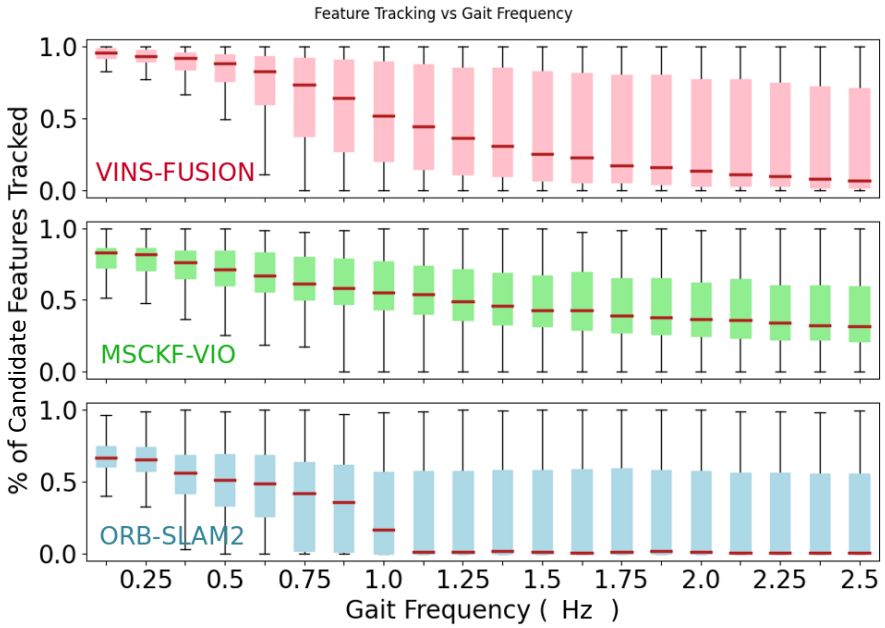

Figure 3.6: Feature tracking performance of different SLAM algorithms vs gait frequency of robot

The above figure validates our hypothesis that the performance of feature tracking decreases as the robot's gait frequency increases for each of the surveyed methods. Intuitively this makes a lot of sense: as camera viewpoint changes more dramat-

ically, it is more difficult to find common visual cues between images taken at successive viewpoints. One caveat about our experiment is that we only look at the performance of short-term data associations. More specifically, we do not evaluate the performance of long-term data associations such as loop closure. Because loop closures generally occur infrequently, we do not believe that including data on them would noticeably change the outcome of any of our feature tracking results.

Reflecting on the results from the two experiments in this chapter, we believe that the decrease in feature tracking performance is one of the primary reasons that visual and visual-inertial methods for SLAM fail when robot motion becomes dynamic. In stereo visual SLAM, it is only possible to estimate the change in the camera's pose, if we can track three features at minimum [11]. This requirement also assumes that the three features that we track and triangulate are not in a degenerate configuration (i.e. a straight line). In practice, however, much more than 3 tracked features are needed to obtain robust visual estimation. While the addition of IMU information allows algorithms such as VINS-Fusion and MSCKF-VIO to avoid complete failure when the minimum number of tracked features is not available, low-cost IMU sensors are generally extremely noisy and lead to large amounts of estimation drift. In the next chapter, motivated to avoid SLAM failure, we describe our method for leveraging the periodicity present in legged locomotion to increase the performance of visual feature tracking.

Chapter 4

SLAM FOR DYNAMIC LEGGED ROBOTS: MULTI-SESSION PERIODIC SLAM

In the previous chapter, we were able to show that sequential feature tracking is not robust against fast, periodic trajectories like those experienced on some legged robots. To combat this, Figure 4.1 highlights the intuition behind the approach of periodic feature tracking:

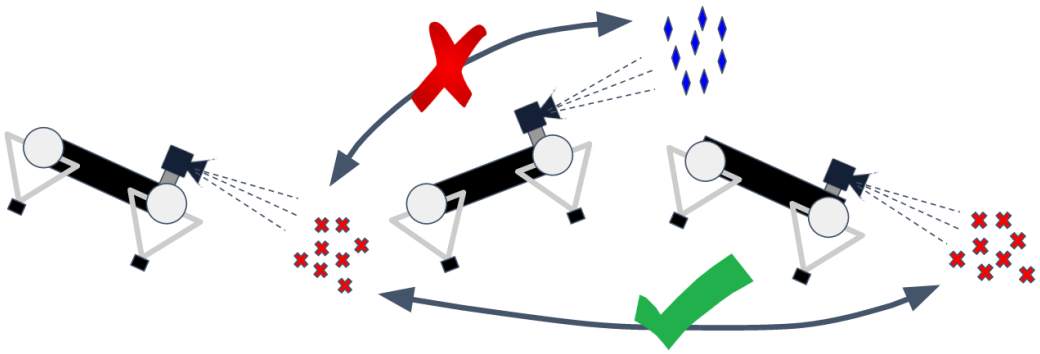

Figure 4.1: In this figure, when the robot shifts its focus from looking downward to upwards, it is unable to track any visual cues from the environment. However, if the robot is able to remember how the floor looks each time it is looking downwards and compare these periodic viewpoints, it can estimate its relative motion.

Because specific visual features seem to persist at certain portions of the gait cycle, tracking features that have similar phase values can lead to improved performance. In this chapter, we present the main contribution of our work: a method for leveraging this intuition by integrating periodic feature tracking into a visual-inertial SLAM system.

4.1 Periodically Mapping Different Portions of the Gait Cycle

In order to perform periodic feature tracking, our method relies on being able to consistently extract and track features from images collected during an interval in which the phase of the robot's gait cycle is similar. Thus, our method makes two key assumptions about the robotic system for which it is being used on:

1. Periodic tracking has a global clock indicating the phase of the robot's gait.

2. Images taken at similar gait phases contain mutual visual features.

Given a set of images with similar gait phases and mutual visual features, feature tracking begins with an initialization step and then a tracking step. After initializing visual features in the first image, the tracking step persists for as long as there are enough features to track. If at any point the system "loses" too many features to track, the approach re-initializes to add new visual features. While this approach is general to any feature detector, keeping efficiency in mind, we use Harris corner detection [22] to initialize features and the Lucas-Kanade method [19] to track them.

Our method initializes multiple periodic feature trackers to track different intervals of the robot's gait cycle. For each periodic feature tracker, our method introduces a visual SLAM session, which is responsible for using the tracked features to build a sparse map of three-dimensional visual landmarks and to estimate the location of the robot during a certain phase interval.

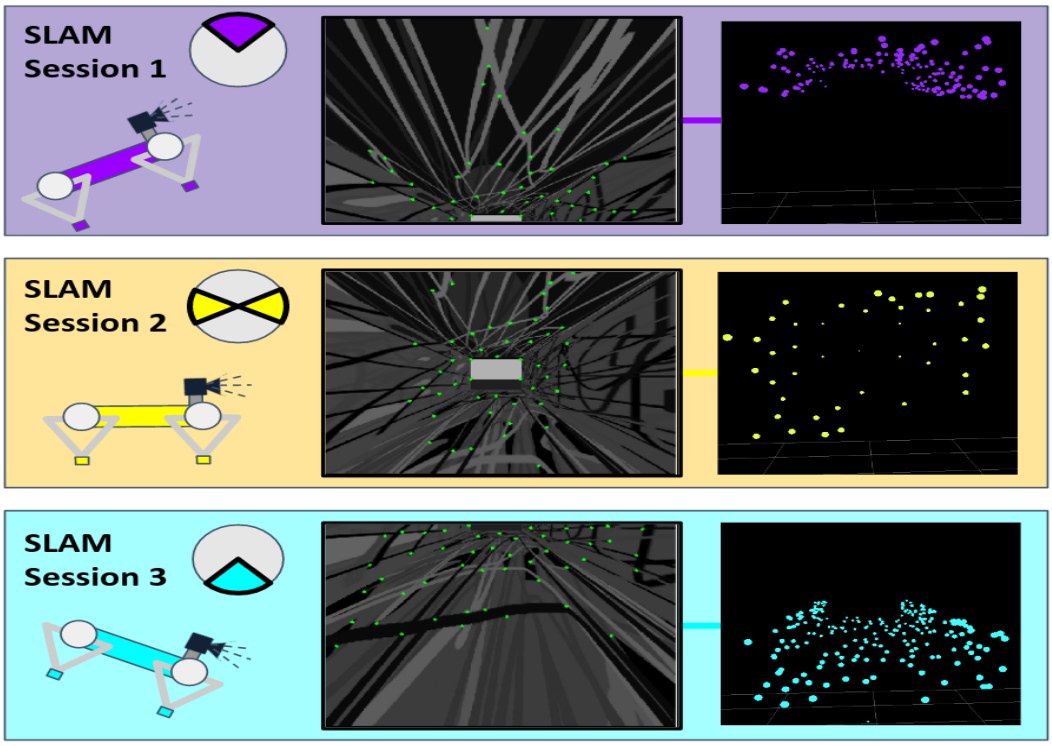

Figure 4.2: Three SLAM sessions performing estimation periodically when a robot is looking upwards, forwards, and downwards. The dial represents which part of the robot's gait cycle images were taken from. The left column of images shows the result of periodic feature tracking in simulation. The right column of images shows the corresponding three-dimensional map of landmarks that each SLAM session maintains using these features.

Each individual SLAM session periodically updates its estimate of the robot and the map at a lower rate but is more successful due to periodic rather than sequential feature tracking. While in general this methodology can be extended to any number of SLAM sessions, Figure 4.2 shows an example for a situation in which there are three SLAM sessions to track three different sections of the robot's gait cycle, each characterized by a distinct robot body orientation.

When performing multi-session SLAM, we must address the problem of fusing different state estimates from each visual SLAM session. While a naive approach to this problem might take the average of the output from the different SLAM sessions, it would be advantageous if these sessions tightly shared information with one another to achieve a more robust combined performance. The idea of fusing the result from different SLAM sessions is not new, and our method is inspired by previous work on multi-session visual SLAM for coordinated mapping. In [21], the authors propose connecting different SLAM sessions with visual anchor constraints, which they define to be pose constraints that are created when the visual front-end recognizes that the robot is in the same location.

In our work, we adapt the idea from [21] to the problem of constraining multiple periodically updating SLAM sessions running on a single robot. Because a fundamental challenge we have is that we cannot use visual information to constrain individual SLAM sessions (otherwise we would not need multiple sessions in the first place), we instead choose to take advantage of measurements from an IMU onboard the robot. IMU sensors can be used to provide measurements of the robot's acceleration and angular velocity between different SLAM sessions when feature tracking is not reliable. By performing Euler integration of the IMU measurements (described further in Section 4.3.3), our method is able to introduce relative pose constraints between each of the different SLAM sessions running on the robot.

4.2 Building a Periodic Factor Graph

We choose to solve the periodic multi-session SLAM problem that we have posed using the techniques for factor graph optimization that we described in Chapter 2. The primary reason that we use factor graphs is that they provide an economical representation of the SLAM problem as a system of unknown states, which we wish to solve for, and the relationships or constraints between these states. Furthermore, factor graphs are inherently amendable to adding constraints between sequential and non-sequential states, which is necessary for our method. Figure 4.3 illustrates

a schematic that is useful in thinking about representing this problem as a factor graph:

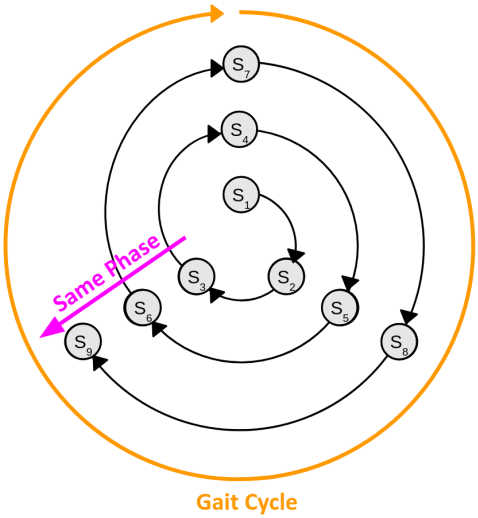

Figure 4.3: Schematic for our method as a factor graph. We use S_i to denote the i^{th} state of the robot in time. Each cycle around the circular graph represents one period of the robot's gait cycle. Robot states positioned along a spoke of the graph have a similar gait cycle phase.

To convert the schematic in Figure 4.3 into an actual factor graph, we must introduce factors between each of the different robot states denoted by S_i. Our method proposes that we add visual factors between the states with similar gait phases positioned along each spoke of the graph. Each spoke of the graph, therefore, represents an individual visual SLAM session performing feature tracking periodically on one portion of the robot's gait cycle. Then, to connect the different visual SLAM sessions running on each spoke, our method introduces inertial factors between every sequential state. Each of the inertial factors uses IMU measurements to estimate the relative change in pose of the robot when feature tracking is not possible. We show a factor graph representation for both the sequential visual-inertial SLAM problem and our novel periodic visual-inertial SLAM problem in Figure 4.4:

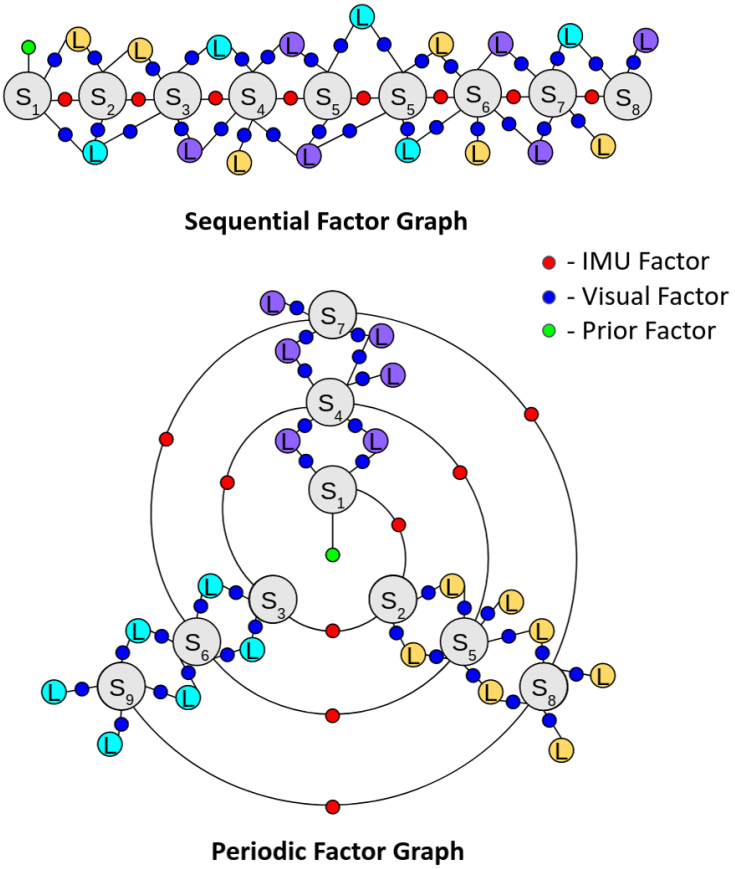

Figure 4.4: Sequential Factor Graph vs. Periodic Factor Graph

In Figure 4.4 blue dots represent factors that provide visual constraints between periodic states, red dots represent factors that provide inertial constraints between sequential states, and green dots represent factors that constrain the robot to a prior location. In this figure, we also introduce landmark nodes, marked with an L, to represent the sparse map that each SLAM session maintains. Each of the landmark nodes in the graph has an associated color that represents the section of the robot's gait cycle in which it was tracked.

We show that compared to the sequential factor graph which includes all landmarks in a single map, our periodic factor graph separates landmarks into individual maps corresponding to their gait phase. While the periodic factor graph in Figure 4.4 only shows SLAM being performed periodically on three portions of the gait cycle, we want to reemphasize that running additional SLAM sessions would simply correspond to adding more spokes to the factor graph. In the next few sections, we show how we can use the periodic factor graph to set up a nonlinear optimization problem to estimate the state of the robot its map.

4.3 Deconstructing the Form of Each Factor

In this section, we rely on the background material from Chapter 2 to explain the mathematics behind the periodic factor graph shown in Figure 4.4. To represent the state of the robot and each visual landmark, we use two types of variable nodes: s_i and l_i. We represent the state of the robot with $s_i = [p_i, R_i, v_i]$, where p_i, R_i, and v_i represent the position, rotation, and velocity of the robot, and we represent the state of a landmark with l_i, where each landmark is a three-dimensional point.

We assume that each of the factors we use is corrupted with zero-mean, additive Gaussian noise. Given this assumption, each Gaussian factor can be written in a form similar to Equation 4.1, where F can be thought of as a constraint or cost function which is dependent on the robot and landmark states as well as a sensor measurement z_i.

$$\phi_i \propto exp(-\frac{1}{2} \|F(s_i, l_i, z_i)\|^2_\Sigma) \tag{4.1}$$

In the rest of this section, we explain how each of the factors in Figure 4.4 can be written similar to this general form. We begin with the prior factor which describes the distribution of the robot's initial state. Then, we explain the visual factor which constrains the robot to visual landmarks in periodic robot states. Finally, we detail the IMU factor which provides odometry between sequential robot states.

4.3.1 The Prior Factor

The prior factor is the simplest factor in the full periodic factor graph shown in Figure 4.4. While all other factors are useful in estimating the robot's relative motion, the prior factor grounds the estimated state of the robot to a global reference frame.

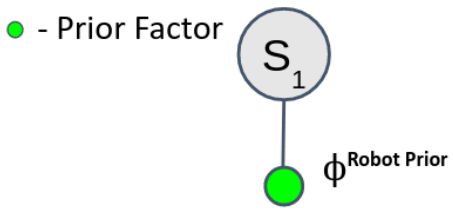

Figure 4.5: A single prior factor on the initial state of the robot

Given a prior measurement of the initial location of the robot, z^p, with covariance Σ^p, we define a Gaussian prior factor on the initial robot state as:

$$\phi^{Prior} \propto exp(-\frac{1}{2}\left\|(h^{Prior}(s_1) - z^p)\right\|^2_{\Sigma^p}) \tag{4.2}$$
where h^{Prior} is trivially the identity function

4.3.2 The Visual Factor

Blue visual factors represent constraints between the state of the robot and every landmark that the robot observes while in that state. We show an example of visual factors along one spoke of the periodic factor graph in Figure 4.6 below:

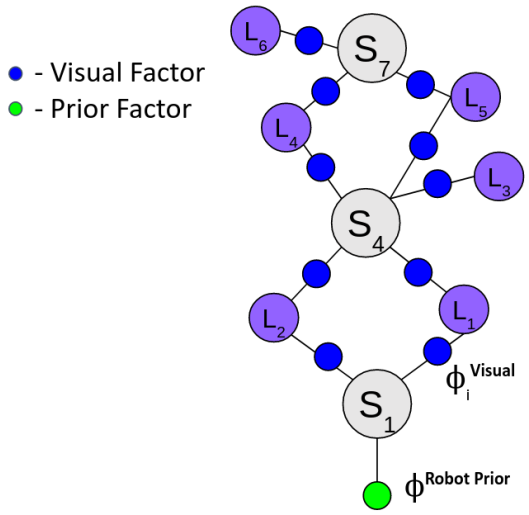

Figure 4.6: Visual factors mapping landmarks at periodic robot states

Each visual factor in Figure 4.6 represents a cost between its connected landmark and robot nodes that is dependent on a visual measurement, z_i^v. Visual measurements in our method are periodically tracked stereo features from the front-end of SLAM with the form $z_i^v = [u_i^L, u_i^R, v_i]$. Here, u_i^L and u_i^R are the x coordinates of the tracked feature in the left and right stereo images and v_i is the y coordinate of the tracked feature in both images.

To calculate the cost for a particular visual measurement, the visual factor transforms a three-dimensional landmark into an estimated stereo feature, $\hat{z}_i^v$, at a corresponding robot state. We introduce the following visual measurement function, h^{visual}, which performs this transformation in two steps: coordinate frame transformation (g) and projection (π):

$$h^{Visual}(s_i, l_i) = \pi(g(s_i, l_i)) = \hat{z}_i^v \tag{4.3}$$

In the first step, the inner function, g, transforms the landmark, l_i, from world coordinates to camera coordinates, $[X_c, Y_c, Z_c]$. In Equation 4.5, g uses the extracted position and rotation components from the robot's state, p_i and R_i, to perform this transformation:

$$g(s_i, l_i) = g(p_i, R_i, l_i) = (R_i)^T(l_i - p_i) = [X_c, Y_c, Z_c] \tag{4.4}$$

Next, the outer function in Equation 4.4, π, is used to project the three-dimensional landmark point from camera coordinates to estimated stereo image coordinates:

$$\begin{gathered}\pi(X_c, Y_c, Z_c) = \begin{bmatrix} \frac{X_c f_x}{Z_c} & \frac{(X_c - b) f_x}{Z_c} & \frac{Y_c f_y}{Z_c} \end{bmatrix} = [\hat{u_L}, \hat{u_R}, \hat{v}] \\ \text{where } f_x \text{ and } f_y \text{ are camera focal lengths} \\ \text{and } b \text{ is the baseline or distance between the camera centers}\end{gathered} \tag{4.5}$$

After transforming the 3D landmark l_i into a stereo feature point, we can calculate a cost, commonly known as re-projection error, by taking the difference between the estimated stereo feature and the measured stereo feature. The visual factor computes the re-projection error for a visual measurement with covariance Σ^v as follows:

$$\phi^{Visual} \propto exp(-\frac{1}{2} \left\| h^{Visual}(s_i, l_i) - z_i^v \right\|^2_{\Sigma^v}) \tag{4.6}$$

Because the visual measurement function is nonlinear, when performing optimization we must calculate its Jacobian with respect to the robot state s_i and the corresponding measured landmark l_i. For the sake of brevity, we omit the derivation for this Jacobian which can be found in [5].

4.3.3 The IMU Factor

The periodic factor graph in Figure 4.4 has IMU factors that provide relative odometry constraints between the two robot states to which the factor is connected.

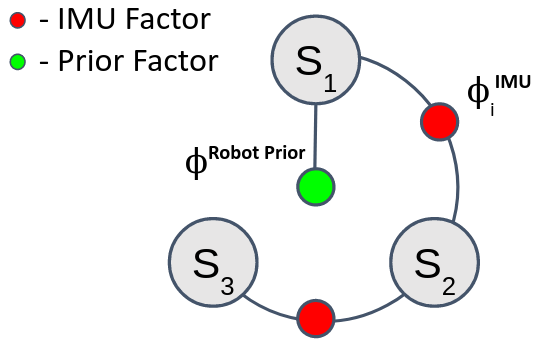

Figure 4.7: IMU factors connecting sequential robot states

Each IMU factor uses measurements from the two sensors that make up the IMU: the gyroscope and the accelerometer. With respect to the robot's inertial frame, the gyroscope sensor outputs the angular velocity of the robot, ω_i, and the accelerometer sensor outputs the acceleration of the robot, α_i. We assume that each sensor's measurements are corrupted with additive white noise given by η^g and η^a.

Using the measurements from the accelerometer and gyroscope, it is possible to describe the dynamics of the robot's position (p_i), orientation (R_i), and velocity (v_i) between two sequential time instances. Equation 4.7 uses the method of Euler integration to describe these dynamics over a fixed time interval Δt [7].

$$\begin{aligned} p_{i+1} &= p_i + \Delta t v_i + \frac{\Delta t^2}{2}\left[g + R_i(\alpha_i - \eta^a)\right] \\ v_{i+1} &= v_i + \Delta t\left[g + R_i(\alpha_i - \eta^a)\right] \\ R_{i+1} &= R_i\, exp\left(\Delta t\left[\omega_t - \eta^g\right]^\wedge\right) \end{aligned} \tag{4.7}$$

In Equation 4.7, we use the $\wedge$ operator to convert an element of $\mathbb{R}^3$ to a skew-symmetric matrix.

$$\begin{bmatrix} \omega_1 \\ \omega_2 \\ \omega_3 \end{bmatrix}^\wedge = \begin{bmatrix} 0 & \omega_3 & \omega_2 \\ -\omega_3 & 0 & -\omega_1 \\ -\omega_2 & \omega_1 & 0 \end{bmatrix} \tag{4.8}$$

To group some of the quantities from the Euler integration step together, we use z_i^{IMU} to represent the angular velocity and acceleration measurements from the IMU, and Σ^{IMU} to represent the covariance of these measurements:

$$\begin{aligned} z_i^{IMU} &= [\omega_i, \alpha_i] \\ \Sigma^{IMU} &= diag(\eta^g, \eta^a) \end{aligned} \tag{4.9}$$

.

Then, we introduce an IMU process function, h^{IMU}, to succinctly represent the Euler integration step from Equation 4.7. We note that unlike the previous measurement functions in this section, h^{IMU} is a function of the sensor measurement, and it represents the robot's dynamics.

$$\hat{s}_{i+1} = h^{IMU}(s_i, z_i^{IMU}) \tag{4.10}$$

Using this IMU process function, we can write each IMU factor as an error between the predicted and estimated next state of the robot:

$$\phi^{IMU} \propto exp(-\frac{1}{2} \left\| s_{i+1} - h^{IMU}(s_i, z_i^{IMU}) \right\|^2_{\Sigma^{IMU}}) \tag{4.11}$$

One way to connect consecutive states with an IMU factor would be to create a new factor and robot state node for each measurement from the IMU. To avoid adding states to our graph at such a high rate, in our implementation we instead use the idea of IMU preintegration from [3] and [7], where many IMU measurements are accurately summarized into a single motion constraint between frames of interest. These preintegrated IMU factors also take into account time-varying biases in the measurements from the IMU, which are important in getting our method to work on a real-world robot. Due to complexity, we point the reader interested in details about the preintegrated IMU factors and biases that we use to [7].

4.4 MAP Inference with Robust Cost Functions

After specifying the form for each of the factors, we can maximize their product to calculate a MAP estimate for the robot and landmark states. We adapt the equations from Section 2.4 by plugging in each of the factors to arrive at the following optimization problem:

$$\begin{aligned} S_{MAP}, L_{MAP} &= \underset{s,l}{\operatorname{argmax}} \prod_{i=0}^{T} \phi_i^{Prior} \phi_i^{Visual} \phi_i^{IMU} \\ &= \underset{s,l}{\operatorname{argmin}} \left\| (h^{Prior}(s_1) - z_1^p) \right\|_{\Sigma^p}^2 + \sum_{i=0}^{T} \left\{ \left\| h^{Visual}(s_i, l_i) - z_i^v \right\|_{\Sigma^v}^2 \right. \\ &\qquad \left. + \left\| (s_{i+1} - h^{IMU}(s_i, z_i^{IMU})) \right\|_{\Sigma^{IMU}}^2 \right\} \end{aligned} \tag{4.12}$$

To solve this optimization problem we use the equations from Section 2.4 describing nonlinear MAP inference. The general approach we follow is that we first use a Taylor series expansion to linearize Equation 4.12, and then we iteratively solve this linearized equation using the Gauss-Newton or Levenberg-Marquardt [23] techniques to compute a SLAM solution.

In the current optimization objective, all measurements of a specific type are modeled with the same uncertainty. Moreover, we find that without explicitly pruning erroneous measurements, our method is largely affected by outliers. To combat this issue, we incorporate a robust error model into our optimization to weight each measurement based on its residual value r_i (e.g. for a visual measurement: $r_i = h^{Visual}(s_i, l_i) - z_i^v$). While a number of different robust error models exist [1], in our implementation we choose to use the Geman-McClure cost function (ρ):

$$\rho(r_i) = \frac{\frac{r_i^2}{2}}{1 + r_i^2} \tag{4.13}$$

We choose the Geman-McClure instead of other popular robust cost functions, such as the Huber cost function, because of its strong bias against large outliers. We show a comparison of the standard squared residual cost function along with the Geman-McClure cost function and the popular Huber cost function in Figure 4.8. Compared to the standard least squares cost function, the tails of the Geman-McClure and Huber cost functions neglect the influence of large residuals. We find that by using the Geman-McClure cost function to implicitly remove the effect of outliers, our method has significant improvement.

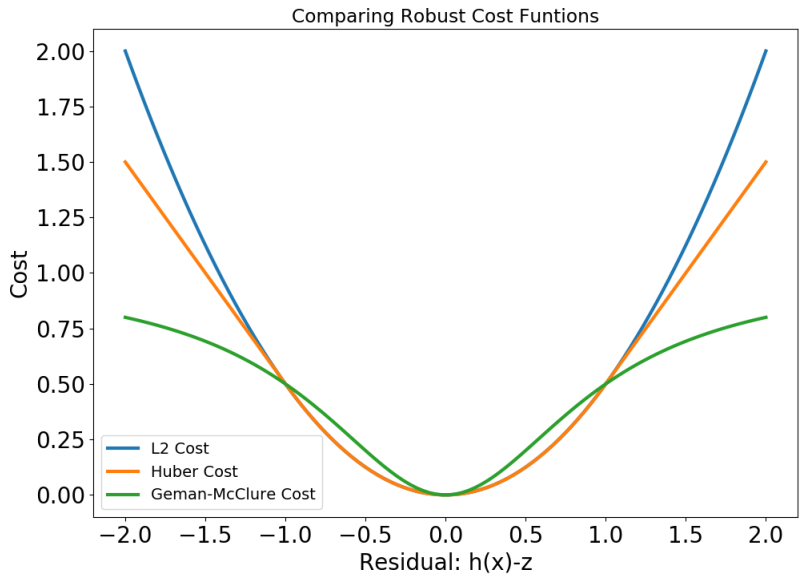

Figure 4.8: The effect of different cost functions on weighting measurement residuals. The Geman-McClure robust cost functions gives the least weight to large residuals when compared to the other cost functions.

4.5 Determining the Number of SLAM Sessions

The number of SLAM sessions and the portion of the gait cycle that each SLAM session tracks are currently hand-tuned based on a few important considerations. In determining which portions of the robot's gait cycle to track, we find that it is important to track parts of the gait cycle in which camera frames are readily available. In Figure 4.9 below, we show that as robot gait frequency increases it is harder to sample images consistently during some portions of the gait cycle.

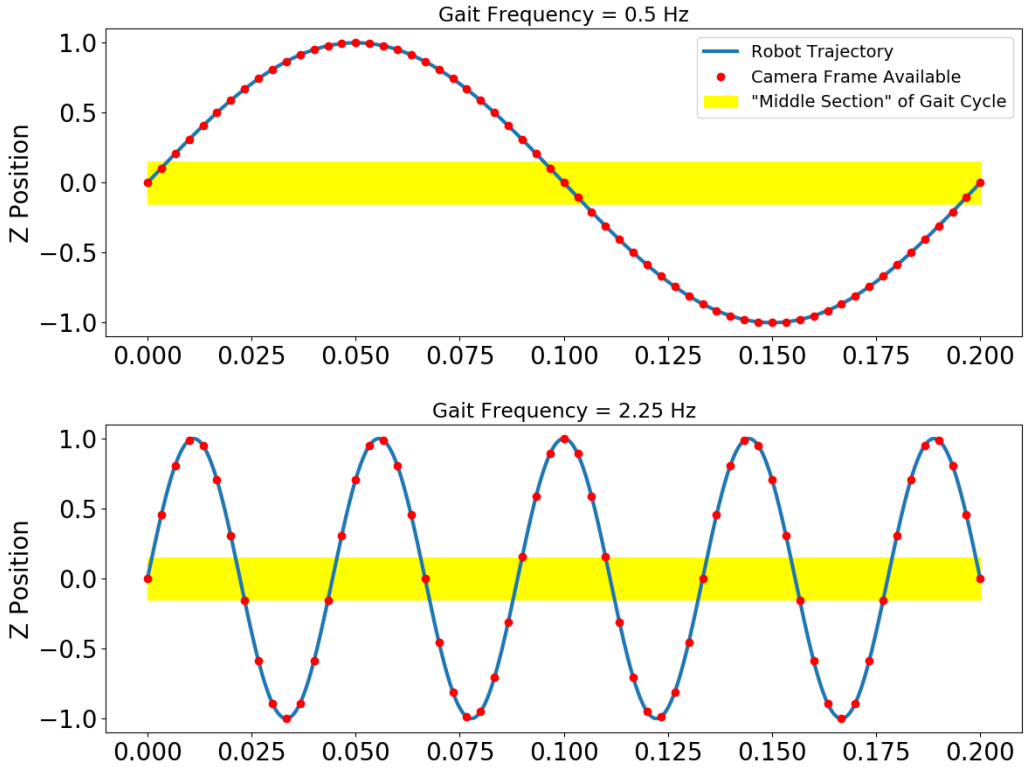

Figure 4.9: The effect of gait frequency on camera frame availability. Using the parameterized gait from Chapter 2, this figure shows a periodic robot's trajectory for two different gait frequencies with blue lines. Red dots along the lines represent instances along the robot's trajectory when an image is available to be tracked. The yellow shaded area in each plot shows a portion of the robot's gait cycle which we argue does not have frames consistently available at higher frequencies.

In Figure 4.9, as the robot's gait frequency increases it becomes more difficult to sample images during the "middle" portion of the robot's gait cycle highlighted in yellow. Because the velocity of the robot is highest when the robot is executing this part of its gait cycle, performing SLAM using images collected from this part of the robot's trajectory is especially inconsistent. Furthermore, if the gait frequency of the robot continues to increase, without a high enough frame-rate, sampling images

during any portion of the robot's gait cycle will become inconsistent and our method will fail.

In picking the number of SLAM sessions, the main consideration that we make is the trade-off between computational efficiency and accuracy. While using more SLAM sessions can lead to higher estimation accuracy by using more information, it comes at the price of maintaining additional maps of the environment. For the case of the simulated robot and the real-world legged robot that we perform experiments on in Chapter 5, we believe that three SLAM sessions evenly spread out across the robot's gait cycle work well. We find that using three SLAM sessions for a periodically pitching robot sufficiently captures features from different viewpoints without being too much of a computational burden. In Figure 4.10, we show how a three-spoke periodic factor graph performs estimation in our simulated environment by maintaining separate maps for when the robot is looking up, forwards, and downwards. We also show that by combining the results from each of these estimators tightly we can calculate unified SLAM output.

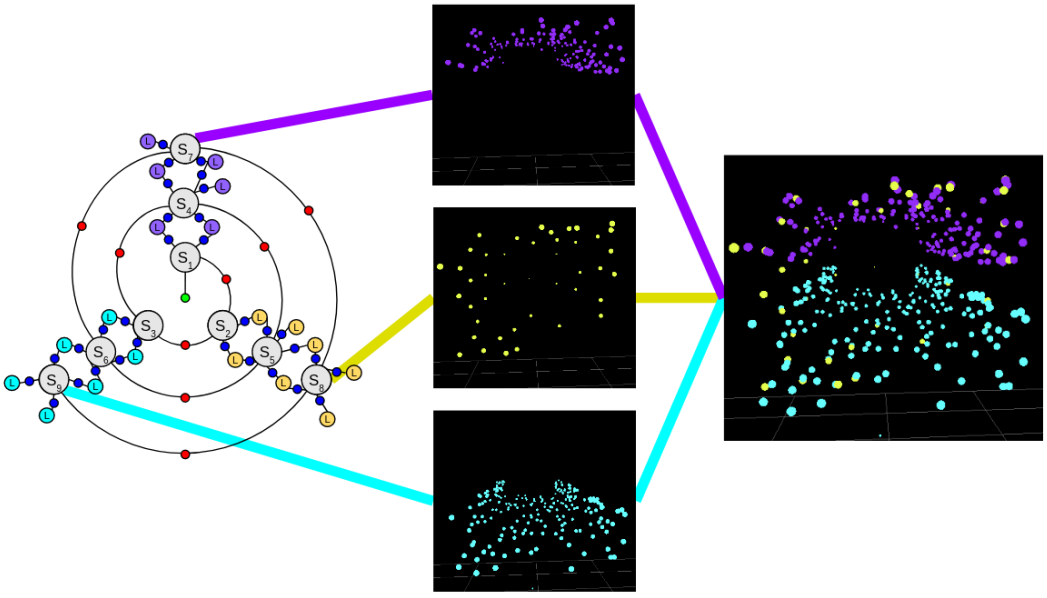

Figure 4.10: Periodic SLAM tracking three different gait phases in simulation. In this figure, we show a three-spoke factor graph (left) combining maps from three different periodic visual SLAM sessions (middle column of images) to get a unified SLAM Result (right).

4.6 Fixed Lag Smoothing and Incremental SLAM

When performing SLAM with a large number of visual landmarks for an extended period of time, we experienced that our method reached its computational perfor-

mance limit. Moreover, optimizing continuously growing factor graphs, was not feasible in real-time. To keep the computational complexity of SLAM bounded, we choose to marginalize over robot states and landmarks that are sufficiently old. Our method uses a technique called fixed lag smoothing which only keeps a fixed lag of previous factors (e.g. last five seconds) and marginalizes out all factors before this time. The process of marginalization effectively summarizes all factors created before the fixed lag into a more succinct marginal factor. We find that fixed lag smoothing improves the real-time performance of our method at the cost of sub-optimal performance due to the inability to relinearize older factors.

To further enable the ability of real-time performance, our method also incorporates an incremental smoothing approach called iSAM2 [14]. Using this incremental approach, our method is able to reuse parts of previously computed SLAM solutions to perform SLAM more efficiently. [14] explains this algorithm in much further detail. We rely on an implementation of this algorithm from the commonly used Georgia Tech smoothing and mapping (GTSAM) library [4].

Chapter 5

RESULTS

In this chapter, we present the results of our proposed method. We begin by briefly introducing some of the details for the implementation of our algorithm. Then, we compare the performance of our algorithm to several state-of-the-art implementations of SLAM in both a simulated and a real-world environment.

5.1 Implementation

We implemented our multi-session periodic SLAM algorithm in C++ relying on different open-source libraries, such as GTSAM, Open CV, and Eigen. All the code for this project can be found at `https://github.com/kumarhans/periodic_slam`. All the experiments were run on an AMD® Ryzen 7 2700x processor at 3.7GHz with 16GB of memory. A high-level block diagram of the main parts of our code is shown in Figure 5.1.

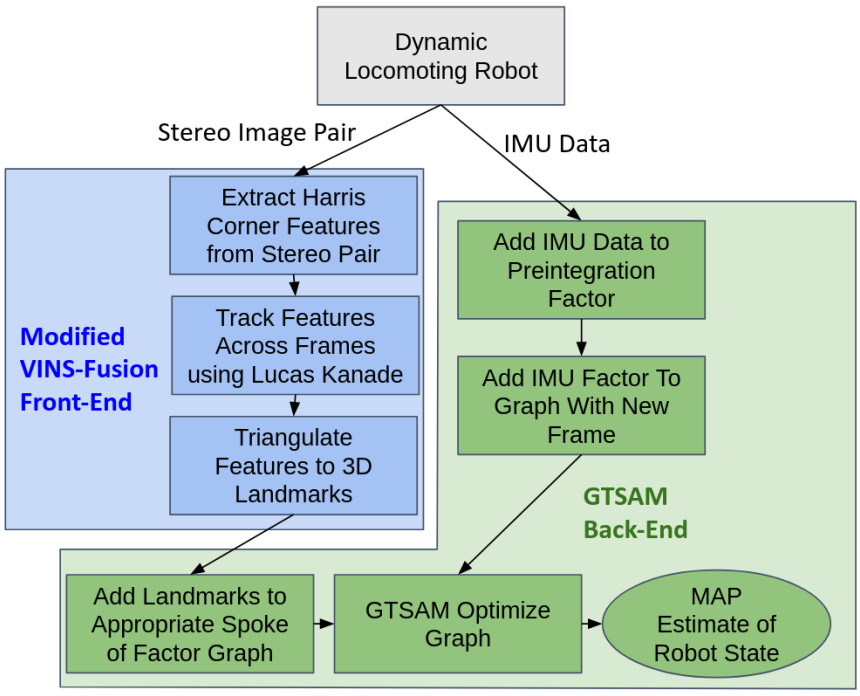

Figure 5.1: Block diagram of major modules in our code. We used a modified version of VINS-Fusion for our SLAM front-end and GTSAM for our SLAM back-end.

While we tried implementing our own SLAM front-end from scratch, for our final implementation we opted to use a modified version of the front-end from VINS-Fusion [26] to achieve more consistent results. Specifically, we found that the front-end of VINS-Fusion did a good job pruning unreliable features during tracking.

5.2 Metric for SLAM Performance

We evaluate the performance of different SLAM systems by comparing the commonly used Absolute Trajectory Error (ATE) metric. We follow the conventions used in [9] and use $P_{est,i}$ and $P_{gt,i}$ to represent the estimated and ground-truth pose of the robot at time step i, respectively. Below we introduce the formula for ATE, and we show how to calculate the Root Mean Square Error (RMSE) as a metric for average error along a trajectory.

1. ATE: This type of error measures the global drift in the estimated poses compared to the ground truth poses. Unlike absolute pose error (APE), this type of error only takes into account the translational component of the drift.

$$\begin{aligned} E_i &= P_{gt,i}^{-1} P_{est,i} \\ ATE_i &= \|trans(E_i)\| \\ RMSE_{ATE} &= \sqrt{\frac{1}{N}\sum_{i=1}^{N} ATE_i^2} \end{aligned} \tag{5.1}$$

5.3 Results in Simulated Environment

Using the simulation environment that we introduced in Chapter 3, we were able to thoroughly evaluate the performance of our method for performing SLAM periodically. For all of our experiments in this section, we compared the performance of our method, which we call Periodic SLAM, against the performance of the same three state-of-the-art algorithms for visual and visual-inertial SLAM that we surveyed in Chapter 2: VINS-Fusion, ORB-SLAM2, and MSCKF-VIO.

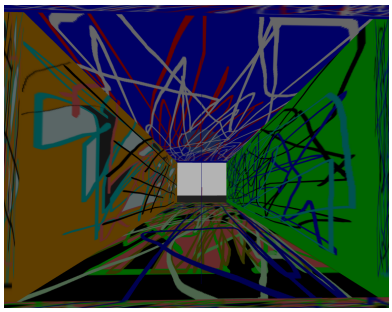

Figure 5.2: Simulated hallway environment

We implemented our method to use three different SLAM sessions to track three distinct portions of the robot's gait cycle. Our method used one session for when

the robot was looking upwards, one session for when the robot was looking down the middle, and one session for when the robot was looking downwards.

We conducted a set of experiments in which we used the same set of robot gait parameters from Table 3.1, and we once again ran the robot forward in the simulated hallway for 10 meters. We modulated the robot's gait frequency parameter, ω, from .125 Hz to 2.5 Hz to see how increasingly dynamic motion would affect the performance of our method. We also performed 50 trials at each gait frequency to account for variance between trials. We show the median values over these trials for all of the performance metrics in this chapter.

5.3.1 Comparing Front-End Performance

Before evaluating the performance of our full periodic SLAM system, we chose to compare the performance of the front-end (feature tracking) of our method against the surveyed methods in simulation. We perform this experiment first because the front-end of SLAM is the core module that we claim our method improves upon. We came up with two metrics that we believed were useful in evaluating visual front-end performance:

1. Average Percent of candidate features tracked = $\frac{1}{F-1} \sum_{i=1}^{F-1} \frac{C_i}{R_i}$
    - F-1 = total number of camera frames we perform tracking on (impossible to track features in first frame)
    - C_i = number of tracked features in current frame
    - R_i = number of detected features in reference frame
2. Average Feature lifetime = $\frac{1}{N} \sum_{i=1}^{N} (t_{i,last} - t_{i,first})$
    - N = total number of features tracked over robot's trajectory
    - $t_{i,last}$ = timestamp in seconds that we last tracked feature
    - $t_{i,first}$ = timestamp in seconds that we first detected feature

Unfortunately, because of ORB-SLAM2's complicated, multi-tier data association scheme, we were unable to extract a meaningful metric for its feature lifetime, and for that reason we do not include it. Aside from that, we show a performance comparison between the front-end of each method below in Figure 5.3:

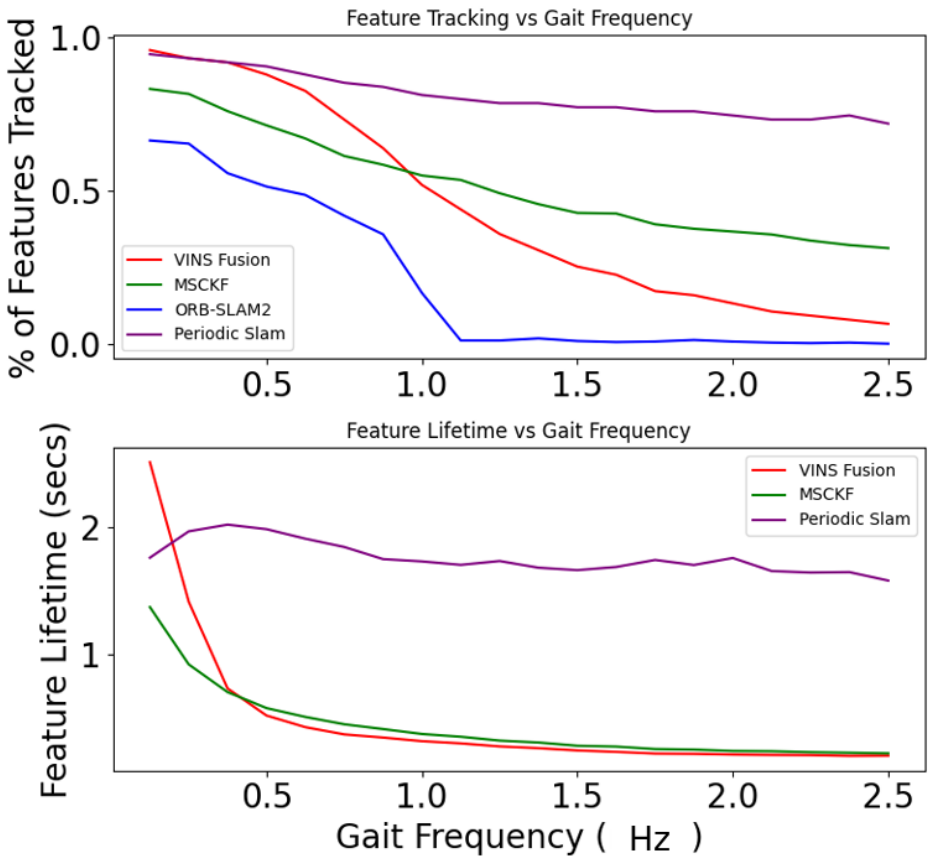

Figure 5.3: Comparing the performance of SLAM front-ends in simulation

From the above plots in Figure 5.3, we see that our method shows a significant increase in the performance of feature tracking and feature lifetime in the simulation environment. Looking at the first plot above, we see that by performing feature tracking periodically rather than sequentially, our method can consistently track over 75% of candidate features even as robot motion becomes more dynamic. Notably, our method has a similar performance to VINS-Fusion at low gait frequencies because our implementation uses a modified version of VINS-Fusion's front-end.

In the second plot of Figure 5.3 showing feature lifetime, we see an even bigger difference in performance. We see that our system can remember features for roughly 4x longer than the other two methods when the robot's gait frequency exceeds 1.0 Hz. Making longer-term data associations is beneficial in reducing overall drift in SLAM. These results serve as a great validation for our previous hypothesis that tracking features periodically on a periodic, dynamic system is more effective than tracking features sequentially.

5.3.2 Comparing Full SLAM Performance Against State-of-the-Art Methods

To begin evaluating the accuracy of our full SLAM system, we first show the estimated XZ trajectories of the robot outputted by our SLAM method compared to the state-of-the-art methods. Below we include the trajectories from one of our trials at three different gait frequencies:

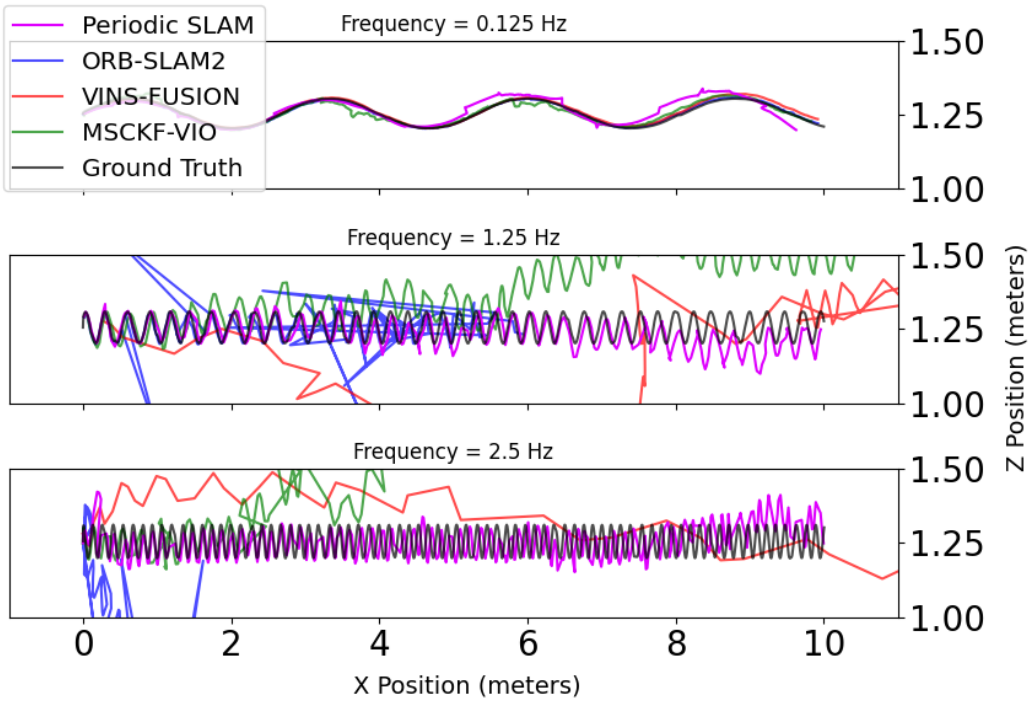

Figure 5.4: XZ trajectories for simulated robot trials

In Figure 5.4 above, the qualitative performance of each method for three different "levels" of dynamic motion can be observed. When the simulated robot is moving at .125 Hz per second, the output from our method (purple) and the three surveyed methods all follow the ground truth trajectory (black) closely. However, in the middle and bottom plots of Figure 5.4, when the robot executes larger gait frequencies, our method follows the ground truth trajectory much more closely than the surveyed methods, which end up diverging.

To quantitatively summarize the performance of each of the simulation methods, we show plots for the median RMSE of ATE over the set of 50 trials in Figure 5.5. The shaded regions for each plot represent the 25th and 75th percentiles for the RMSE values. We cut off the maximum error for each curve at .5 for ATE to make the plot more readable.

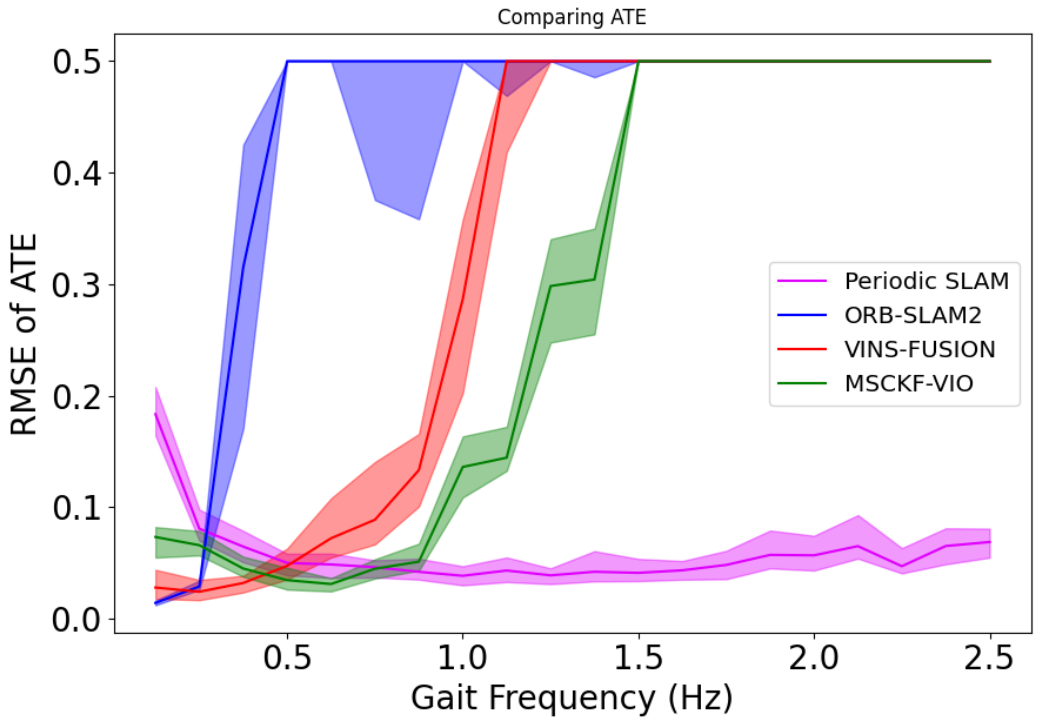

Figure 5.5: RMSE of ATE for different SLAM methods as a function of gait frequency in the simulated environment

Figure 5.5 shows that at the lowest gait frequencies (.5 Hz and below) the state-of-the-art methods have lower global drift (ATE) than our proposed method. This result shows that when the robot is moving slowly, sequential feature tracking can achieve higher accuracy than periodic feature tracking because it uses information from all of the images received. We believe that, at the cost of computational resources, our method can perform better at these lower gait frequencies if it uses more than three periodic SLAM sessions.

However, as the gait frequency of the simulated robot increases, Figure 5.5 shows that our method outperforms the surveyed methods in terms of ATE. Because of a decrease in the performance of feature tracking, all of the state-of-the-art methods end up diverging as gait frequency increases, while our method maintains accurate state estimation.

5.3.3 Comparing Our Approach Against Modified State-of-the-Art Methods

We decided to additionally compare our approach to a modified version of each of the state-of-the-art methods surveyed. In this section we make a simple modification to each of state-of-the-art methods that removes all camera frames besides frames taken

when the robot is pitching upwards. By doing this, each of the surveyed methods can be thought of as an periodically updating upwards facing SLAM session. We perform the same experiment as before in which we vary the gait frequency of the simulated robot, and we show the results in Figure 5.6.

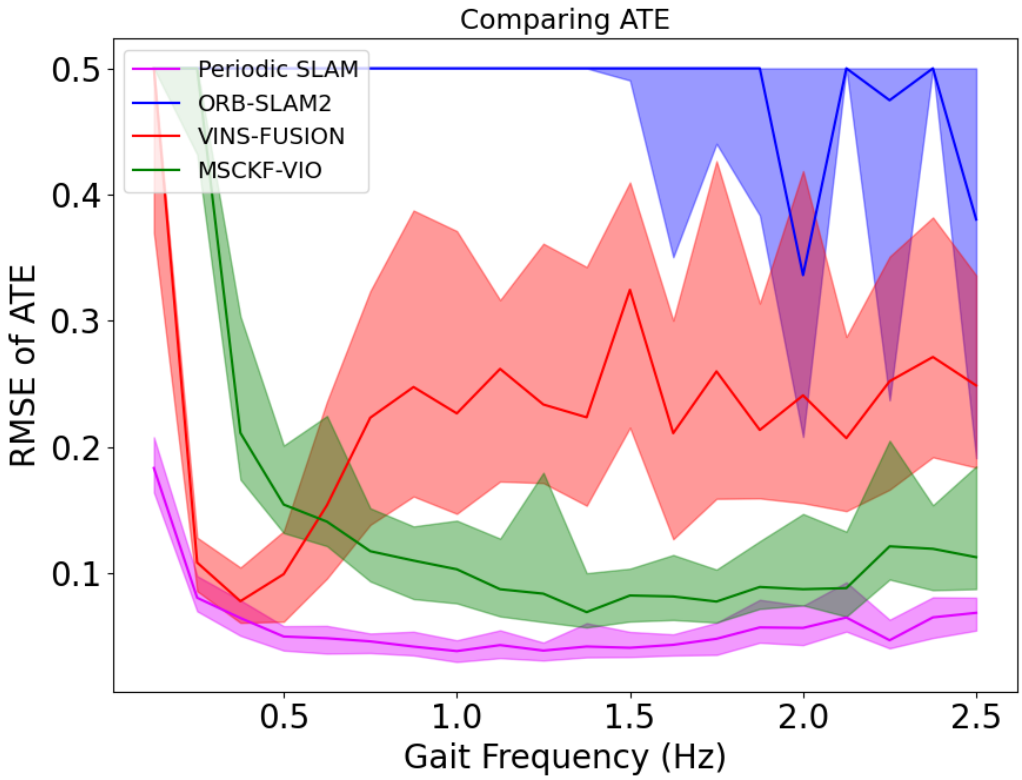

Figure 5.6: RMSE of ATE of state-of-the-art methods given only upwards frames vs. full Periodic SLAM in a simulated environment

Figure 5.6 shows that by only using frames from when the robot is looking upwards, VINS-Fusion and MSCKF-VIO are both able to avoid divergence even as the robot's gait frequency is increased. ORB-SLAM2, however, still experiences divergence because it does not utilize IMU measurements. In addition to the median RMSE, the variances of VINS-Fusion and MSCKF-VIO are also both consistently higher than our approach. We believe that by only using information from one camera viewpoint, these modified approaches are vulnerable to outlier measurements. Outlier measurements can cause small local jumps in the estimated robot's pose that end up having large effects on the robot's global drift. This result further validates our approach to combine multiple different periodically updating SLAM sessions.

5.3.4 Comparing Our Approach Against Individual SLAM Sessions

To further analyze the performance of our method, we decided to compare the performance of each individual SLAM session against our full combined multi-session Periodic SLAM method. In Figure 5.7 below, we show the performance for individual SLAM sessions tracking the robot's motion when it is looking upwards, down the middle, and downwards along with our combined method.

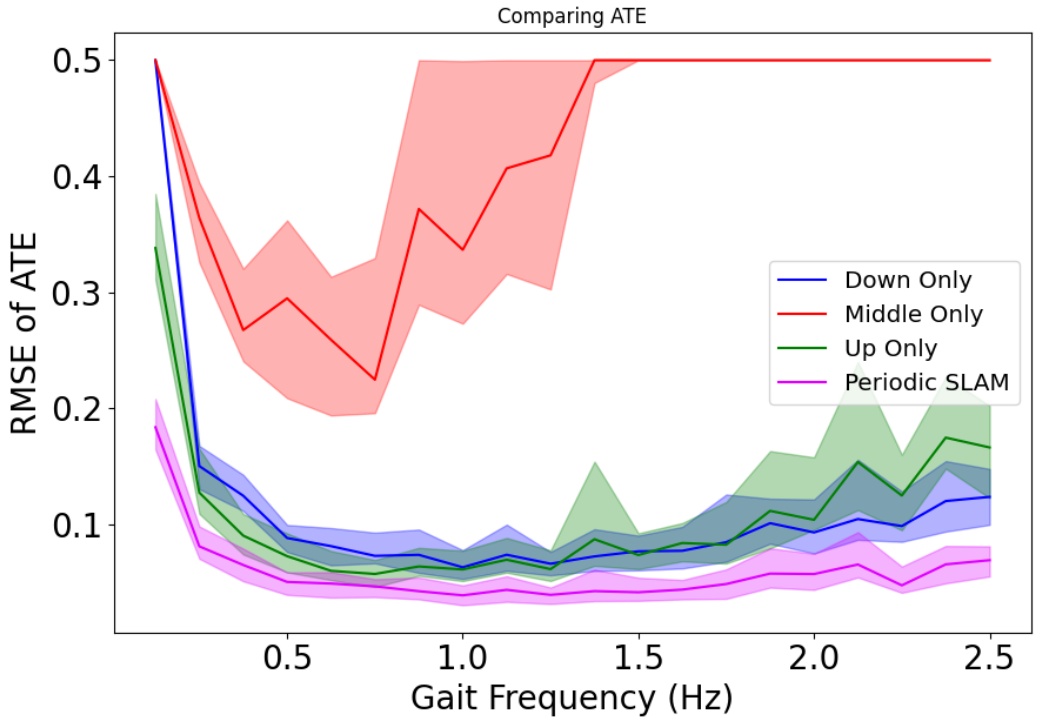

Figure 5.7: RMSE of ATE of individual SLAM sessions vs. full Periodic SLAM in a simulated environment

In Figure 5.7, it can be seen that using three combined SLAM sessions strictly has a lower ATE than using any individual SLAM session at all robot gait frequencies that were tested. We find that by incorporating visual measurements at a higher rate, our combined method can decrease the amount of drift due to noisy IMU measurement propagation. More specifically, at faster gait frequencies above 1.5 Hz our combined approach has at most 65% of the global drift (ATE) of any individual SLAM session.

Our results show that the middle SLAM session performs poorly at each gait frequency. This is because when the robot is moving through the middle portion of its gait cycle, it is moving at its highest angular velocity. The high angular velocity in the middle portion of the gait cycle leads to increased difficulty in sampling images at a consistent viewpoint, and it results in low feature tracking performance.

Without a sufficient camera frame-rate, if the gait frequency of the robot were to keep increasing, the other SLAM sessions would also perform worse for the same reasoning.

5.4 Results in Real World

The platform which we selected for our real-world experiments is the Minitaur robot made by Ghost Robotics [10]. The robot is propelled by two direct-drive actuators [15] on each of its four legs. We chose this platform because of its ability to execute highly dynamic gaits such as bounding. To perform visual-inertial SLAM, we mounted an Intel Realsense d435i camera on top of the Minitaur robot. The d435i camera captured stereo image pairs from two global shutter cameras at 30 Hz and IMU data at 300 Hz.

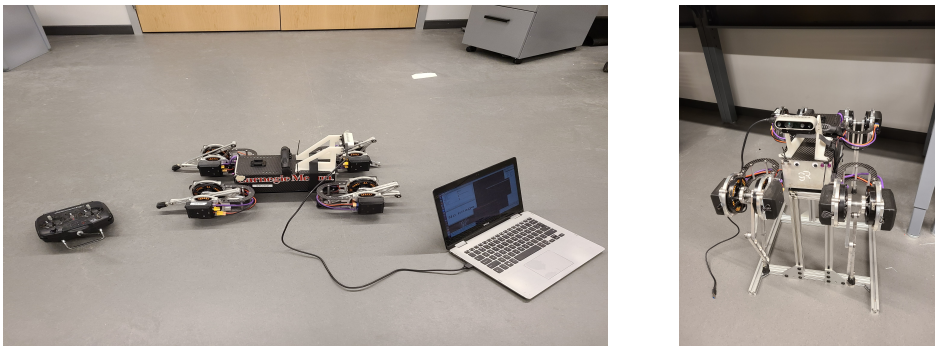

Figure 5.8: Minitaur robot with the Realsense D435i visual-inertial camera mounted to its body

We collected data from the Minitaur robot in an indoor lab space. We relied on a system of 20 Optitack Prime 41 cameras arranged across the room to provide the robot's ground truth pose at 180 Hz.

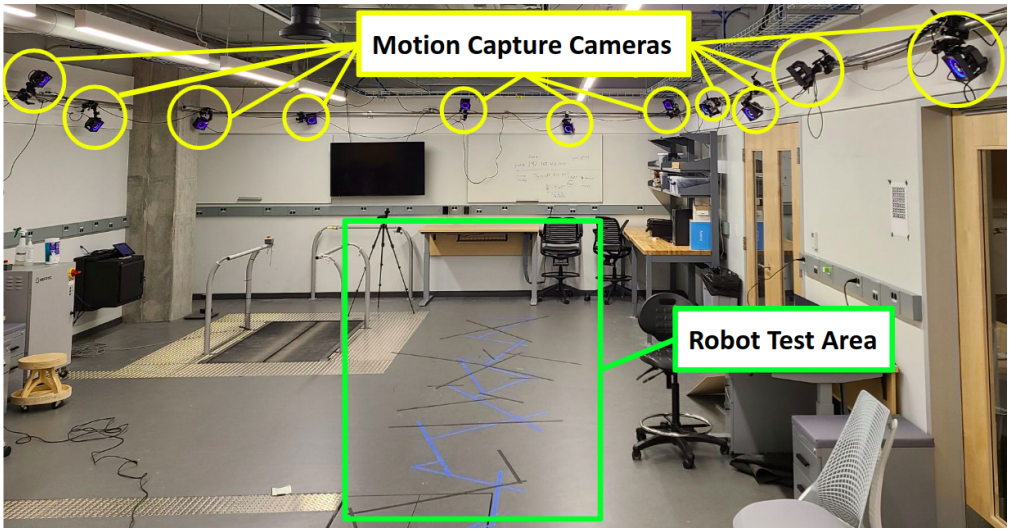

Figure 5.9: Testing area with motion capture setup. We added some strips of black and blue tape to the otherwise featureless floor.

We tested the performance of our Periodic SLAM algorithm against the same three state-of-the-art algorithms for SLAM that we used in our simulated environment: VINS-Fusion, ORB-SLAM2, and MSCKF-VIO. We collected data from the Minitaur robot in three different situations. In the first situation, which we refer to as Easy Gait, the robot executed a slow walking gait forwards for about 7.5 meters. In the second situation, which we refer to as Hard Gait 1, the robot executed a dynamic bounding gait forwards for about 7.5 meters. In the last situation, which we refer to as Hard Gait 2, the robot executed the same gait as in Hard Gait 1 moving forwards for about 7.5 meters, but it was facing the opposite direction. We make this choice because we find that in the opposite direction the robot can track more visual features. For each of these three situations, we recorded 7 trials of the robot moving forward, and we recorded the SLAM output from each algorithm as well the ground truth trajectory from the Mocap setup. Our method once again tracked three portions of the robot's gait cycle: when the robot was looking upwards, when the robot was looking down the middle, and when the robot was looking downwards.

To easily tune the parameters for each of the algorithms we used, we decided to only record raw data from the Minitaur robot and perform SLAM offline. Additionally, in the data we collected we were unable to record phase information from the robot's onboard controller, so we instead hand labeled the phase information for the data that we collected. We realize that this might not be possible outside of the lab setting, but we think that our results still provide a good proof of concept for our method. Lastly, we used Umeyama method's [31] to align the SLAM output from each surveyed SLAM system to best match the ground truth trajectory. This algorithm rotated and translated the output trajectories from each SLAM system to be aligned with the ground truth trajectory for easier comparison.

5.4.1 Comparing Full SLAM Performance Against State-of-the-Art Methods

To begin evaluating the accuracy of our SLAM system on real-world data, we first show the estimated XZ trajectories of the robot outputted from our SLAM method as well as from each of the state-of-the-art methods. Below we include the trajectories from one of our 7 trials at three different gait frequencies:

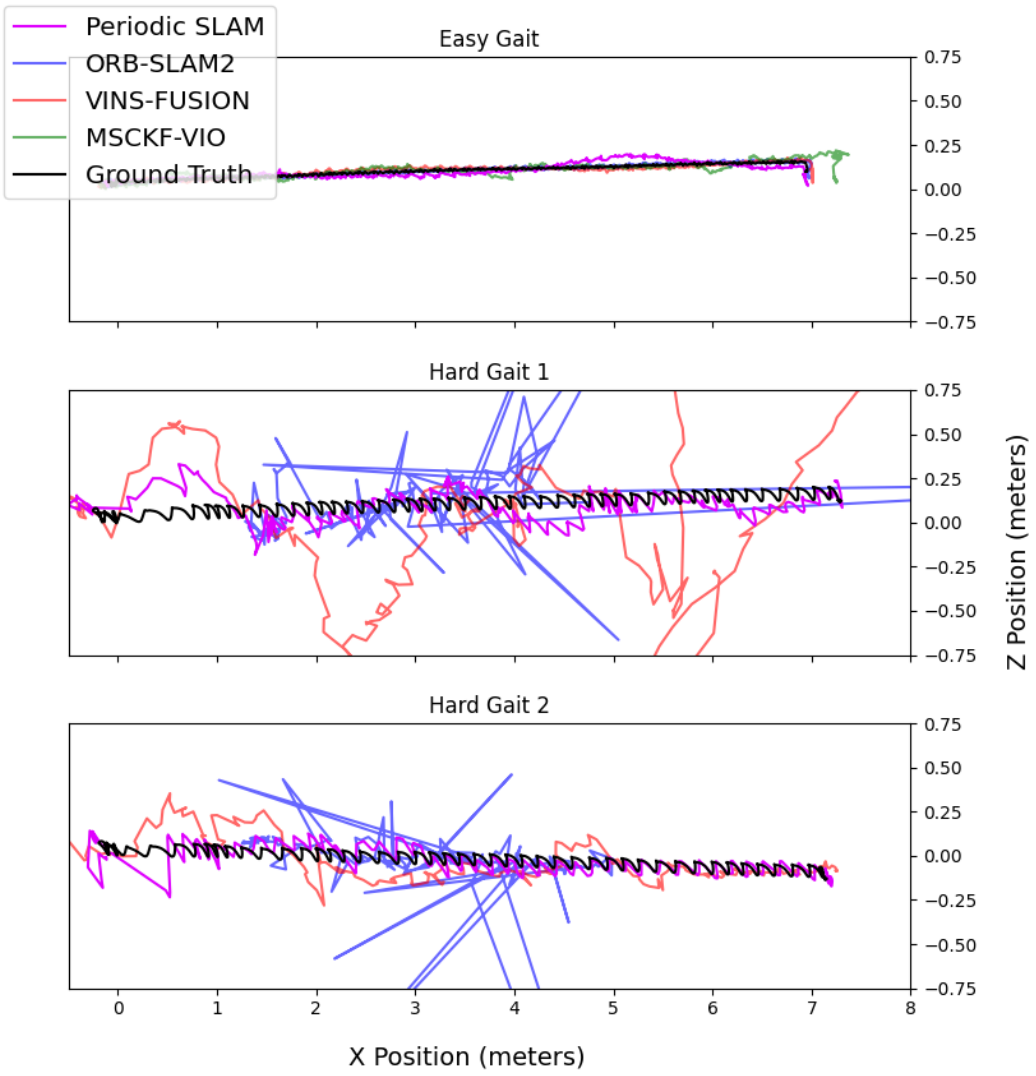

Figure 5.10: XZ trajectories for Minitaur robot trials

In Figure 5.10 above, we are able to see the qualitative performance of each method in three different scenarios. In the easy gait scenario, when the robot is walking slowly, each of the SLAM algorithms outputs a trajectory that is very close to the ground truth trajectory (black). This result shows that each of the SLAM algorithms works on the legged robot when motion is not dynamic. However, for both of the

hard gaits, the performance of all of the SLAM systems severely decreases. While the performance of our method, Periodic SLAM, also decreases it follows the ground truth trajectory most closely.

To quantitatively summarize the performance of each of the methods on the Minitaur robot, we show plots for the RMSE of ATE over the set of 7 trials below in Figure 5.11. We use box plots to show the first to third quartile of RMSE from the distribution of 7 trials. We cut off the maximum error for each plot at 1.0 for ATE to make the figures more readable.

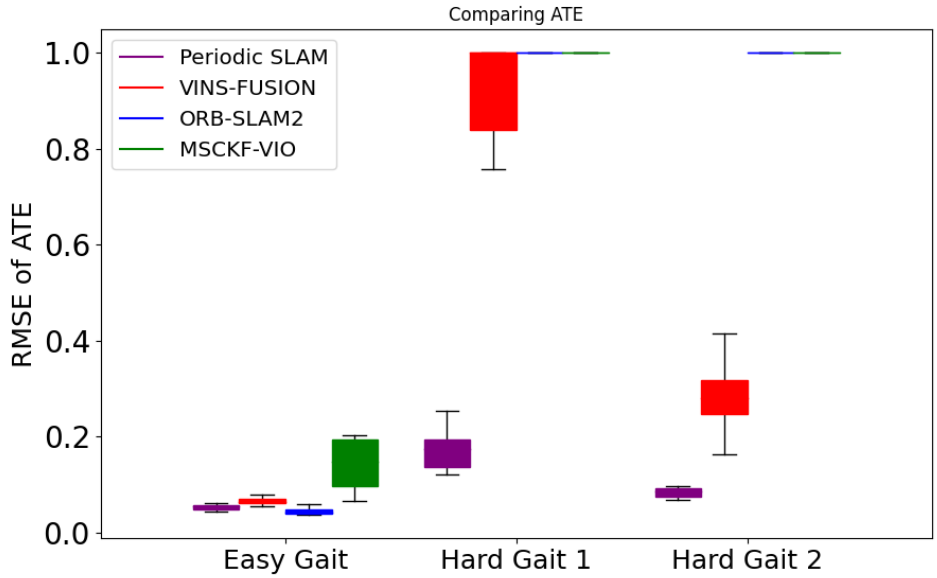

Figure 5.11: Comparing the RMSE of ATE of different state-of-the-art methods vs. Periodic SLAM on data collected from on the Minitaur robot

SLAM Method	Easy Gait	Hard Gait 1	Hard Gait 2
Periodic SLAM	0.053	**0.173**	**0.084**
VINS-Fusion	0.062	1.061	.280
ORB-SLAM2	**0.042**	1.750	1.627
MSCKF-VIO	0.147	53.931	105.258

Table 5.1: Median of the RMSE of ATE of different SLAM systems on Minitaur robot

In Figure 5.11 above, we were surprised to see that ORB-SLAM2, a visual-only method, performed the best on the easy gait scenario. We believe that because of ORB-SLAM2's multi-tier data association scheme, it was able to produce the most accurate results when motion was slow. However, for the two difficult gaits,

periodic SLAM outperforms all three of the state-of-the-art SLAM implementations. In terms of ATE, our method performs roughly 6x better than the next best method (VINS-Fusion) on hard gait 1 and roughly 3x better than the next best method (VINS-Fusion again) on hard gait 2.

While these results show that periodic feature tracking helps improve the performance of our visual-inertial SLAM method, we believe that there is still considerable room to go. Our method still has a high average ATE of .173 meters for hard gait 1. We believe that one of the main reasons for this drift is additional sensor noise that we have not accounted for in this work. As the robot makes large impacts with the ground, the onboard IMU sensor sometimes becomes saturated and unreliable. Moreover, even though the robot is equipped with global shutter cameras, we observe that a small amount of motion blur is still present in the images captured from the camera. We believe that unaccounted IMU saturation and motion blur are two phenomena that negatively affect the performance of our method and still make the problem of SLAM on dynamic systems difficult.

5.4.2 Comparing Our Approach on Difficult Bounding Gaits Against Modified State-of-the-Art Methods

Similar to our experiments in simulation, we additionally compare our approach to a modified version of each of the state-of-the-art methods surveyed on real-world data. By making a simple modification to each of state-of-the-art methods that removes all camera frames besides frames taken when the robot is pitching upwards, we are able to improve the performance of the baseline methods. We only compare to the two difficult bounding gaits in this section and the next section because these are the only gaits in which the robot's camera has a distinct upwards facing viewpoint. We show a comparison between the ATE of our approach and the modified state-of-the-art methods in Figure 5.12.

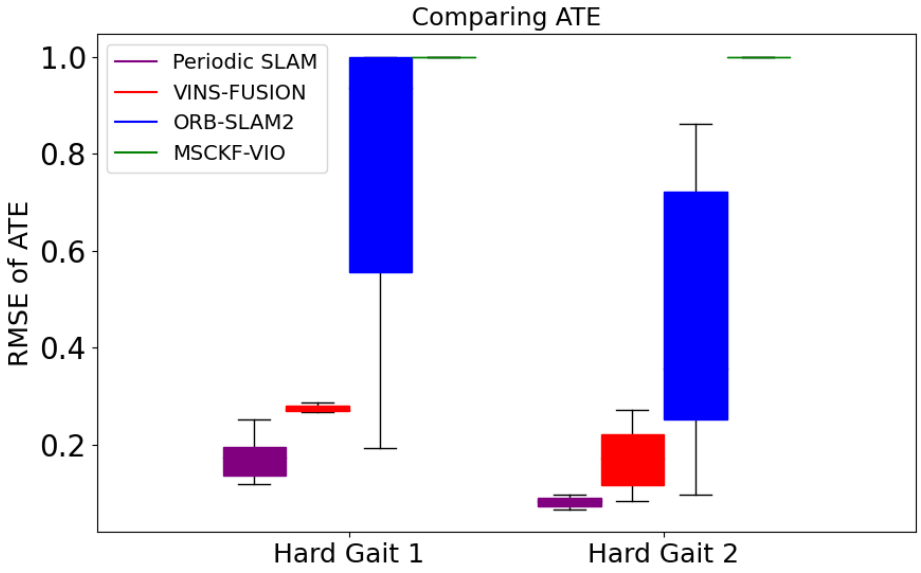

Figure 5.12: Comparing the RMSE of ATE of different state-of-the-art methods given only upwards frames vs. Periodic SLAM on data collected on the Minitaur robot

SLAM Method	Hard Gait 1	Hard Gait 2
Periodic SLAM	**.173**	**.084**
VINS-Fusion	.273	.170
ORB-SLAM2	.936	.358
MSCKF-VIO	71.648	2.270

Table 5.2: Median of the RMSE of ATE of modified state-of-the-art SLAM systems vs. Periodic SLAM on Minitaur robot

Figure 5.12 shows that by only using frames from when the robot is looking upwards, VINS-Fusion and ORB-SLAM2 both have much better performance than when using all available camera frames. Although our modification improved the performance of MSCKF-VIO in simulation, this method diverges on real-world data. We believe that because of its filtering-based back-end, MSCKF-VIO is particularly susceptible to linearization errors caused by the robot's nonlinear impact events. For both of the difficult bounding gaits, our Periodic SLAM approach still performs better in terms of ATE than all of the modified state-of-the-art methods. We believe that on real-world data, with an abundance of sensor measurement noise, incorporating visual information from more than one viewpoint allows our approach to achieve more robust estimation.

5.4.3 Comparing Our Approach on Difficult Bounding Gaits Against Individual SLAM Sessions

In addition to comparing the performance of our Periodic SLAM implementation to each of the state-of-the-art SLAM implementations, we also compared it against each of the different visual SLAM sessions that make up our method. In the graphs below we compare the performance of using three combined SLAM sessions against using only an individual session on the data that we collected from the Minitaur robot.

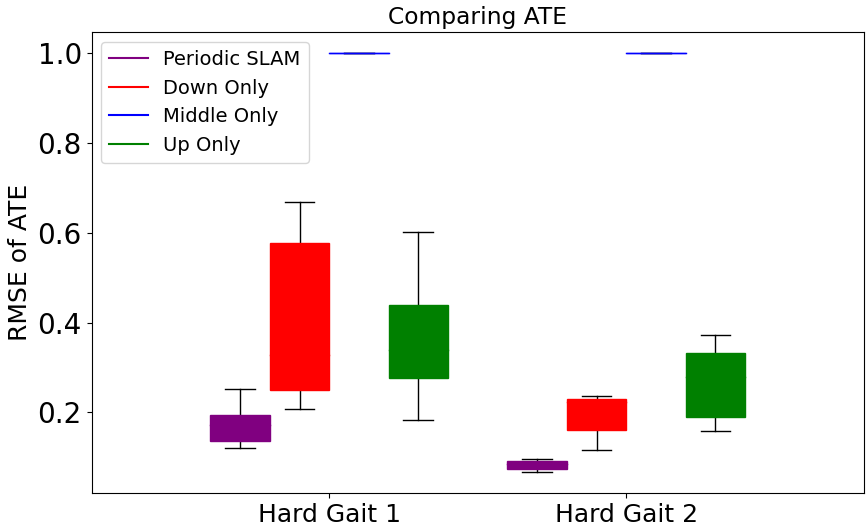

Figure 5.13: Comparing the RMSE of ATE of Individual SLAM Sessions vs. full Periodic SLAM on data collected on the Minitaur robot

SLAM Method	Hard Gait 1	Hard Gait 2
Periodic SLAM	**0.173**	**0.084**
Down Only	0.327	0.223
Middle Only	2.421	2.224
Up Only	0.338	0.280

Table 5.3: Median of the RMSE of ATE of Individual SLAM Sessions vs. Full Periodic SLAM on Minitaur robot

In Figure 5.13, it can be observed that in the two difficult gaits our combined multi-session SLAM method has a lower ATE when compared to any individual SLAM session. Moreover, by connecting the different SLAM sessions our approach is able to achieve 55% less global drift (ATE) than any individual SLAM session. We believe that by using multiple SLAM sessions our method is more resistant to outliers because it has more visual measurements. Similar to the simulated environment, the middle SLAM session once again performs poorly because of difficulty sampling images consistently when the robot is pitching at its highest velocity. We also observe that the middle SLAM session receives images that are affected by motion blur, making estimation even more of a challenge.

Chapter 6

CONCLUSIONS AND FUTURE WORK

In this thesis, we present a method for performing visual-inertial SLAM on a robot with highly dynamic motion. We show that on legged systems when motion is periodic, we can perform the front-end of visual SLAM, feature tracking, with higher fidelity if we track features periodically rather than sequentially. Using this knowledge, we present an algorithm that maintains multiple different visual SLAM sessions that each track features periodically across separate sections of the robot's gait cycle. We also show that if we combine these estimators with an IMU, we can achieve a better combined result than using any individual SLAM session.

In getting our approach to work robustly on a real, legged robot, dealing with large impacts between the robot and the ground is a challenge that we do not address. These impacts can cause IMU sensors to saturate and cause images to become blurry. Fortunately, these impacts are periodic and predictable. Future work includes leveraging the periodicity in legged robot gaits to increase the covariance of measurements received during predictable impact events. Another future direction could be assigning a different weight to each SLAM session in our approach to account for known sections of the robot's gait cycle that are more or less noisy.

Another limitation of our work is that it assumes that the phase of a legged robot's gaits is correlated to the viewpoint of an onboard camera. While this correlation might be reasonable to assume on flat ground, on rough terrain it would definitely not exist. To combat this issue, future work includes incorporating multiple hypotheses for each landmark observed by the robot. By doing this, the visual SLAM session that each landmark is assigned to could be a variable that is optimized over. Moreover, exact phase information may not be an explicit requirement.

Currently, our method is limited to only periodically moving robots that experience dynamic changes in camera viewpoint. However, the general idea of using multiple SLAM sessions connected with an IMU on a single robot can be applied to any scenario in which there is a predictable change in viewpoint. This work could potentially be extended to non-legged robots maintaining multiple maps of an environment in which there is predictability in when to use which map.

BIBLIOGRAPHY

[1] Michael J Black and Paul Anandan. "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields". In: *Computer vision and image understanding* 63.1 (1996), pp. 75–104.

[2] Cesar Cadena et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332.

[3] Luca Carlone et al. "Eliminating conditionally independent sets in factor graphs: A unifying perspective based on smart factors". In: *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2014, pp. 4290–4297.

[4] Frank Dellaert. *Factor graphs and GTSAM: A hands-on introduction*. Tech. rep. Georgia Institute of Technology, 2012.

[5] Frank Dellaert. *Visual SLAM Tutorial: Bundle Adjustment*. 2017. URL: http://www.cs.cmu.edu/~kaess/vslam_cvpr14/media/VSLAM-Tutorial-CVPR14-A13-BundleAdjustment-handout.pdf.

[6] Frank Dellaert, Michael Kaess, et al. "Factor graphs for robot perception". In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[7] Christian Forster et al. "IMU preintegration on manifold for efficient visual-inertial maximum-a-posteriori estimation". In: Georgia Institute of Technology. 2015.

[8] Christian Forster et al. "On-Manifold Preintegration for Real-Time Visual–Inertial Odometry". In: *IEEE Transactions on Robotics* 33.1 (2016), pp. 1–21.

[9] Michael Grupp. *evo: Python package for the evaluation of odometry and SLAM.* https://github.com/MichaelGrupp/evo. 2017.

[10] *Home: Philadelphia: Ghost Robotics*. URL: https://www.ghostrobotics.io/.

[11] Andrew Howard. "Real-time stereo visual odometry for autonomous ground vehicles". In: *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2008, pp. 3946–3952.

[12] Arthur Huletski, Dmitriy Kartashov, and Kirill Krinkin. "Evaluation of the modern visual slam methods". In: *2015 Artificial Intelligence and Natural Language and Information Extraction, Social Media and Web Search FRUCT Conference (AINL-ISMW FRUCT)*. IEEE. 2015, pp. 19–25.

[13] Vadim Indelman et al. “Factor graph based incremental smoothing in inertial navigation systems”. In: *2012 15th International Conference on Information Fusion*. IEEE. 2012, pp. 2154–2161.

[14] Michael Kaess et al. “iSAM2: Incremental smoothing and mapping using the Bayes tree”. In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[15] Gavin Kenneally, Avik De, and Daniel E Koditschek. “Design principles for a family of direct-drive legged robots”. In: *IEEE Robotics and Automation Letters* 1.2 (2016), pp. 900–907.

[16] Patrick Latcham and Clark N Taylor. “Comparison of Visual Simultaneous Localization and Mapping Methods for Fixed-Wing Aircraft Using SLAM-Bench2”. In: *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2020, pp. 1578–1586.

[17] Stefan Leutenegger et al. “Keyframe-based visual–inertial odometry using nonlinear optimization”. In: *The International Journal of Robotics Research* 34.3 (2015), pp. 314–334.

[18] Yali Li et al. “A survey of recent advances in visual feature detection”. In: *Neurocomputing* 149 (2015), pp. 736–751.

[19] Bruce D Lucas, Takeo Kanade, et al. “An iterative image registration technique with an application to stereo vision”. In: (1981).

[20] John McDonald. “Multi-session Visual Simultaneous Localisation and Mapping”. PhD thesis. National University of Ireland Maynooth, 2013.

[21] John Mcdonald et al. “6-DOF Multi-session Visual SLAM using Anchor Nodes”. In: *Proc. European Conference on Mobile Robots* (Jan. 2011).

[22] Hans P Moravec. *Obstacle avoidance and navigation in the real world by a seeing robot rover.* Tech. rep. Stanford Univ CA Dept of Computer Science, 1980.

[23] Jorge J Moré. “The Levenberg-Marquardt algorithm: implementation and theory”. In: *Numerical analysis*. Springer, 1978, pp. 105–116.

[24] Anastasios I Mourikis and Stergios I Roumeliotis. “A multi-state constraint Kalman filter for vision-aided inertial navigation”. In: *Proceedings 2007 IEEE International Conference on Robotics and Automation*. IEEE. 2007, pp. 3565–3572.

[25] Raul Mur-Artal and Juan D Tardós. “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras”. In: *IEEE Transactions on Robotics* 33.5 (2017), pp. 1255–1262.

[26] Tong Qin, Peiliang Li, and Shaojie Shen. “Vins-mono: A robust and versatile monocular visual-inertial state estimator”. In: *IEEE Transactions on Robotics* 34.4 (2018), pp. 1004–1020.

[27] Tong Qin et al. *A General Optimization-based Framework for Global Pose Estimation with Multiple Sensors*. 2019. eprint: `arXiv:1901.03642`.

[28] Nicolas Ragot et al. “Benchmark of visual slam algorithms: Orb-slam2 vs rtab-map”. In: *2019 Eighth International Conference on Emerging Security Technologies (EST)*. IEEE. 2019, pp. 1–6.

[29] Ke Sun et al. “Robust stereo visual inertial odometry for fast autonomous flight”. In: *IEEE Robotics and Automation Letters* 3.2 (2018), pp. 965–972.

[30] Takafumi Taketomi, Hideaki Uchiyama, and Sei Ikeda. “Visual SLAM algorithms: a survey from 2010 to 2016”. In: *IPSJ Transactions on Computer Vision and Applications* 9.1 (2017), p. 16.

[31] Shinji Umeyama. “Least-squares estimation of transformation parameters between two point patterns”. In: *IEEE Computer Architecture Letters* 13.04 (1991), pp. 376–380.

[32] Georges Younes, Daniel Asmar, and Elie Shammas. “A survey on non-filter-based monocular visual SLAM systems”. In: *arXiv preprint arXiv:1607.00470* 413 (2016), p. 414.

Appendix A

PROOF THAT USING MULTIPLE ESTIMATORS IS BENEFICIAL

In state estimation, assuming that we have accurately modeled our measurements and their noise, more measurements can only be beneficial. At the worst case, if a measurement gives us no information, our belief of the state will remain the same. We show a proof that the use of measurements can only improve our estimate below:

$$P(\hat{X} = x) \leq \mathop{\mathbb{E}}_{z \sim P(Z|\hat{X}=x)} P(\hat{X} = x|Z = z)$$
$$ln(P(\hat{X} = x)) \leq ln(\mathop{\mathbb{E}}_{z \sim P(Z|\hat{X}=x)} P(\hat{X} = x|Z = z))$$

Using Jenson's Inequality we can pull the expectation outside of our log:

$$ln(P(\hat{X} = x)) \leq \mathop{\mathbb{E}}_{z \sim P(Z|\hat{X}=x)} ln(P(\hat{X} = x|Z = z))$$

We can then apply Bayes rule to the right side:

$$ln(P(\hat{X} = x)) \leq \mathop{\mathbb{E}}_{z \sim P(Z|\hat{X}=x)} ln(\frac{P(Z = z|\hat{X} = x)P(\hat{X} = x)}{P(Z = z)})$$
$$ln(P(\hat{X} = x)) \leq \mathop{\mathbb{E}}_{z \sim P(Z|\hat{X}=x)} \left[ln(\frac{P(Z = z|\hat{X} = x)}{P(Z = z)})\right] + ln(P(\hat{X} = x))$$

Now the right side of our equation has become a KL divergence expression added to the log-likelihood of our original distribution.

$$\mathop{\mathbb{E}}_{z \sim P(Z|\hat{X}=x)} \left[ln(\frac{P(Z = z|\hat{X} = x)}{P(Z = z)})\right] = D_{KL}\left(P(Z = z|\hat{X} = x)||P(Z = z)\right)$$
$$D_{KL}\left(P(Z = z|\hat{X} = x)||P(Z = z)\right) \geq 0$$

Since we know that KL divergence is always non-negative, we can finish our proof.

$$ln(P(\hat{X} = x)) \leq D_{KL}\left(P(Z = z|\hat{X} = x)||P(Z = z)\right) + ln(P(\hat{X} = x))$$

For this reason, in the case of our method for SLAM, the use of "multiple sessions" of Visual SLAM should outperform the performance of a single session only looking at one part of the robot's gait cycle. In addition to this, having more frequent measurement updates reduces the average uncertainty of our robot along its path. Although the figure below illustrates this idea within the context of Bayesian filtering, the same idea applies to our optimization-based framework.

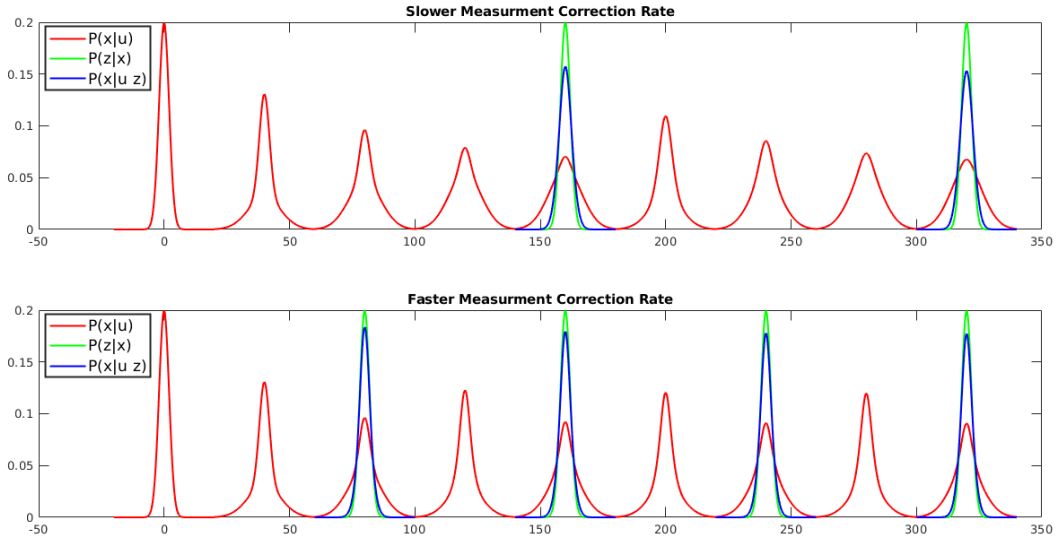

Figure A.1: Visualizing Different Measurement Rates