STATE ESTIMATION WITH INCREMENTAL SMOOTHING AND MAPPING USING THE BAYES TREE (ISAM2)

Vidula Kopli

A THESIS

in

Robotics

Presented to the Faculties of the University of Pennsylvania in Partial Fulfillment of the Requirements for the Degree of Master of Science in Engineering

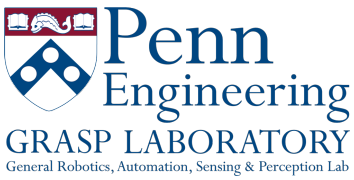
2020

Daniel E. Koditschek, Alfred Fitler Moore Professor, ESE
Supervisor of Thesis

Camillo J. Taylor, Raymond S. Markowitz President's Distinguished Professor, CIS
Graduate Group Chairperson

0.1 Acknowledgements

Many thanks to my mentor Vasileios Vasilopoulos for his guidance with the development process and assistance with operating the robots. Thank you also to Dr. Dan Koditschek for providing resources and an inclusive environment in his lab, allowing me to pursue this study over the summer and during the past few semesters.



0.2 Abstract

There has been increasing interest in using bio-inspired legged robots to explore and accomplish tasks in unfamiliar terrain. These robots must have a reliable way of assessing their pose relative to the environment while mapping their surroundings. Ghost Minitaur^{TM}, a quadruped legged robot, has been shown to be especially resilient to dynamic, real-world environments, able to jump over difficult terrain and climb stairs and chain-link fences. Simultaneous localization and mapping (SLAM) can be difficult for robots like the Minitaur, with contact sensors or visual perception alone being unreliable in such environments. Here, it may be helpful to pose SLAM as a probabilistic inference problem where information from various sensing modalities are combined optimally. In this paper, we apply a graph optimization framework for solving such problems - incremental smoothing and mapping using the Bayes tree (iSAM2) - to solve SLAM for the Minitaur.

Contents

Introduction

1.1 Motivation

Legged robotics present a challenge for state estimation due to slippage of contacts with the ground and complex movements like bounding and jumping [1]. A robust system for localization reliant on multiple sensing modalities is required to accomplish accurate state estimation for legged robotics - a framework that integrates information from various sensors with differences in reliability (noise, covariance) under various conditions. For real-world scenarios, this must be done efficiently with continuous online updates.

1.2 Goals

The goal for this project is to implement visual odometry using iSAM2 on the Turtlebot (Figure 1.1a), integrating measurements from the IMU. This involves 1) understanding machine perception techniques for stereo depth cameras to get initial position estimates of geometric features extracted from the environment, 2) getting iSAM2 running using information from the features and IMU to incrementally solve

for the camera poses and feature positions across timepoints, and 3) creating a visualization of the pose and position estimates to troubleshoot and measure the accuracy of our program.

(a) Turtlebot 2 robot (b) ZED Mini camera

Figure 1.1: The Turtlebot 2 was our main platform for testing. The ZED Mini camera was mounted on the top of the robot; the stereo camera was used to obtain visual features, and the onboard IMU was used to obtain orientation estimates.

Ultimately, we develop this project with the intention to apply it to a legged robotic system. This means considering the addition of information from joint movements and contact with the ground on the Ghost Minitaur^{TM} when discussing the theory.

1.3 Required Hardware and Software Packages

For our vision system we used the ZED Mini stereo camera (Figure 1.1b). The ZED camera has 2 lens for depth sensing and an onboard IMU for orientation estimation. Since our implementation only depends on the ZED camera and its IMU, we used the Turtlebot for testing.

The open-source GTSAM library from Borg Lab was used to implement iSAM2 [2]. The open-source KumarRobotics package “msckf_vio” was incorporated into our package for the purpose of image-based feature detection; the launchable nodelet ”Im-

ageProcessorNodelet" can be ported to any application using a stereo depth cameras [3]. Our project software package is accessible online [4].

The odometry estimates generated by our program are with respect to the "world" coordinate frame, a fixed frame generated by the robot's initial pose. In the Robot Operating System (ROS), we transform the estimated poses and positions of the camera and features, respectively, into the world frame using the provided transform between the ZED onboard IMU and ZED left camera.

Theory

2.1 Problem Setup

The SLAM problem can be posed as a graph optimization problem where the graph represents factored probabilities between observations of the surroundings and the robot itself. An intuitive graph representation for this problem is the factor graph.

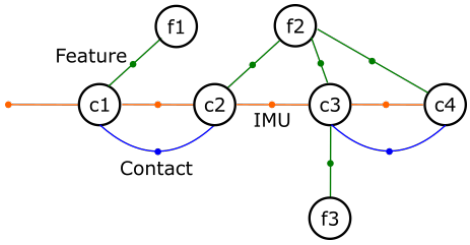


Figure 2.1: Example factor graph for SLAM on the Ghost Minitaur^{TM}. The camera variables (c1, c2, c3, c4) represent the camera at a sequence of timepoints. A prior is placed on c1 to indicate that the pose of c1 is a known constant (the robot starts at c1). The feature variables (f1, f2, f3) represent the corners/distinct features found in the camera image by the image processing library. The factors represent the relationship between the variables. For example, f1 is seen from the camera during the 1st timepoint (c1), and an initial estimate for the position of f1 with relation to c1 is generated. The "Contact" factors in blue connect the camera variables at timepoints at which the Minitaur maintains contact with the ground. An initial estimate for the position of the camera with respect to the previous position is generated from the relative pose measurements taken from the IMU at each timepoint.

Figure 2.1 shows an example of how a factor graph would represent a sample scenario for the Ghost Minotaur^{TM}. Observations or "values" (large open-circled nodes in the graph) represent the state of the robot of objects in the environment at certain timepoints. Factors (small closed-circle nodes in the graph) define constraints between different observations over time.

Observations that are grounded in reality, such as the first pose of the robot, have a prior or unary factor on them to indicate a rigid constraint; these are factors that only connect to one variable like the one that connects to c1 in Figure 2.1. When a new observation is taken from the environment, we can obtain an initial estimate of its location with reference to another observation and add a binary factor connecting those observations.

2.2 Graph Optimization

We used the factor graph to set up the SLAM probability inference problem. However, the initial estimates and priors on the poses and positions of the various observations may not all agree with one another. We can relax the constraints on the estimated pose/position of each observation by considering the noise or covariance of the sensor measurements used to calculate the initial estimate. For instance, the contact sensors may slip, the detected features that are consistent across camera observations may be off by a pixel in one of the camera images, and the IMU may be slightly faulty; these errors may propagate down to the pose and position estimates at later timepoints.

We may also consider that even if the camera has seen many features across timepoints, some are more reliable in estimating the camera's location than others. This is the value in adding probabilistic relationships between observations and using them to continually update the pose and position estimates on the variables as new

information comes in. On each new timepoint, we are updating our best guess at the true relationship between observations and thus the true state of every observation in the graph.

In this way, a factor graph represents a factorization of a joint probability distribution.

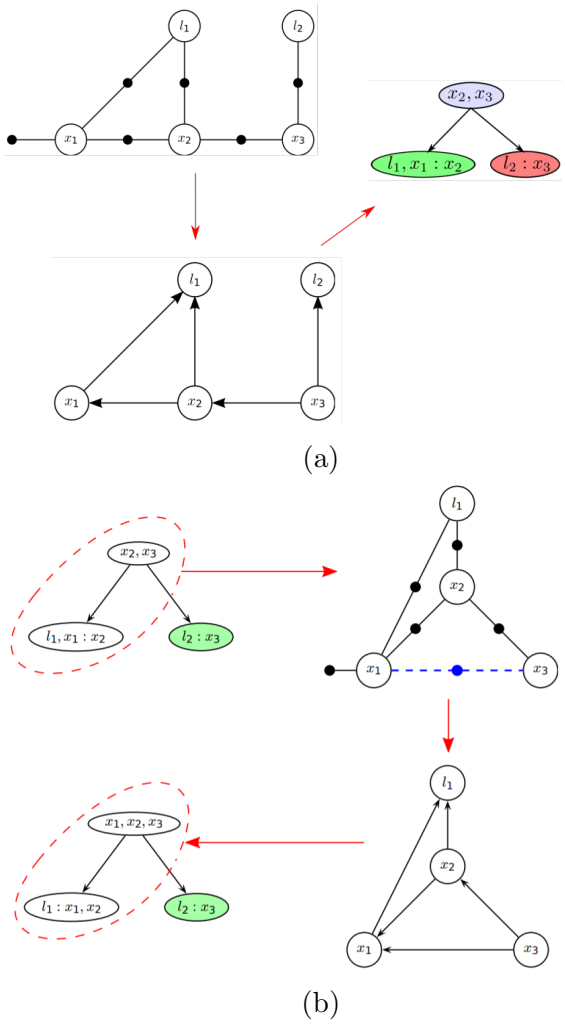


Figure 2.2: (a) Factor graph is eliminated into a chordal Bayes Net, then cliques are collapsed into leaves to form a Bayes tree. (b) When a factor is added, only the nodes leading up to the root of the Bayes tree must be updated [5].

2.3 Online Updates

We could collect all observations then fully solve the graph to obtain the optimal poses and positions across observations; this would solve the full SLAM problem for these observations. However, in real-world scenarios, we would want to be able to collect new observations and optimize the graph online so as to get a "current pose" estimate of the robot and its environment at any given time. Fully solving the graph for every new timepoint would be too slow for these real-world scenarios. This is where iSAM2 becomes helpful.

2.4 iSAM2 and Bayes Trees

Rather than storing the graph in factor graph form, iSAM2 stores the graph as a Bayes Tree. The process by which a factor graph can be converted into a bayes tree is shown in Figure 2.2a [5]. This process is called graph elimination and is equivalent to the factorization of the matrix representation. The matrix form of the bayes tree, the square root information matrix, is much sparser than the matrix form of the factor graph, allowing for more efficient computation. Additionally, as seen in Figure 2.2b updating the bayes tree only requires some nodes to be updated, corresponding to a sparser square root information matrix which requires less memory and computation. Finally, variable reordering can be done while updating the bayes tree, allowing the tree to maintain an optimal structure with smaller cliques, leading to more efficient computations [5].

Experimentation

3.1 Methods

To test the algorithm on the Turtlebot and ZED camera, we launched our program in ROS with the command: "roslaunch gtsam_vio isam2_minitaur_zed.launch". We launched our visualization of the program estimates in RVIZ by running "rviz rviz -d /catkin_ws/src/gtsam_vio/rviz/rviz_tf_features_config.rviz".

The Turtlebot was teleop-ed down the narrow aisles of an around an indoor lab environment with various objects within the field of vision. By directing the robot back to its starting point, we determined the precision of the estimates in the visualization (if the robot returned to the initial position in the visualization, the program created an internally consistent model of the robot's position and surroundings). By directing the robot through turns, we determined the ability of the program to keep up with a rapidly changing field of view and its accuracy in estimating the angle of turns. Finally, by directing the Turtlebot down long halls, we could determine the accuracy of program in estimating the distance traveled by the robot.

3.2 Results

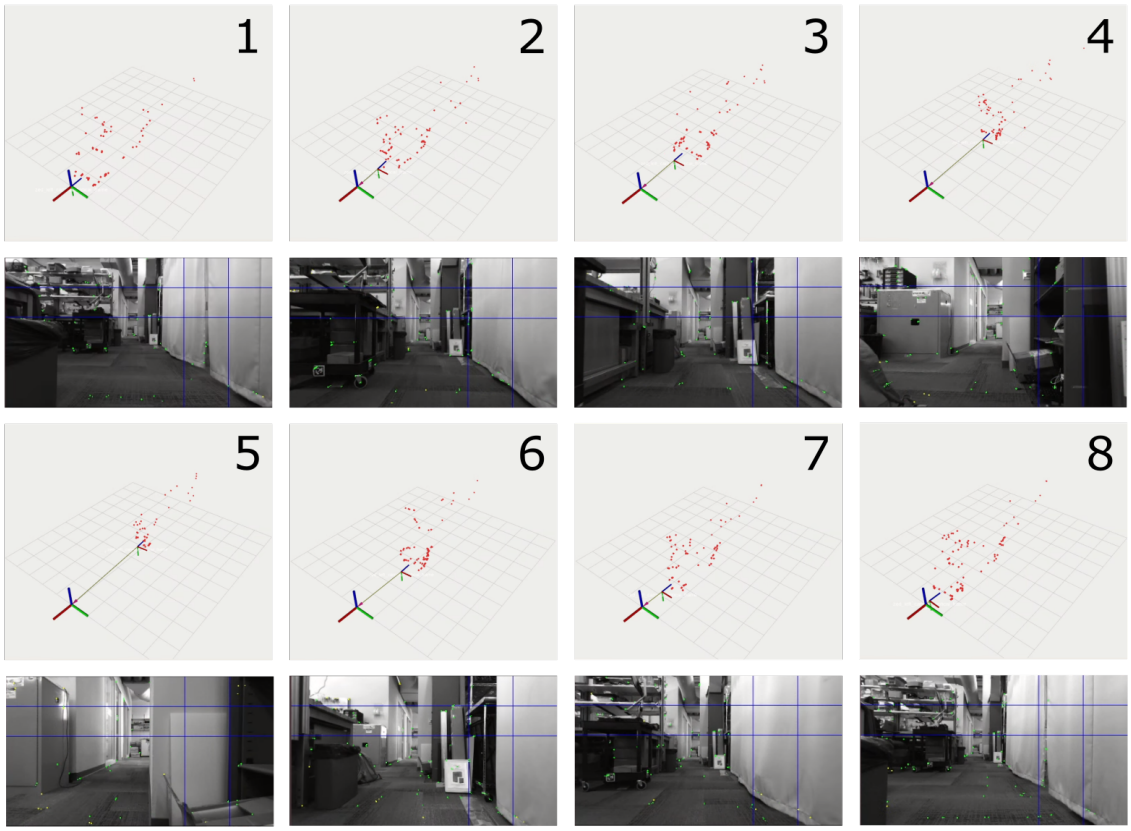


Figure 3.1: Snapshots of pose and position estimates of camera and detected features as the Turtlebot completes a path forwards and backwards to the starting position without turning. In the 3D visualization, the large motionless axes represents the global frame and the smaller moving axes represents the estimated pose of the ZED camera, where red, green, and blue represents the x, y, and z axis, respectively. The red points in the 3D visualization represent the features detected by the image processor and their estimated positions. In the camera images, the features from the image processor are green/yellow.

Since we only added factors for the features detected by the image processor and the camera's IMU, we tested on the Turtlebot. We plotted the estimated camera poses and feature locations while driving the Turtlebot along predetermined paths through the lab. The performance was robust to changes in the environment such as

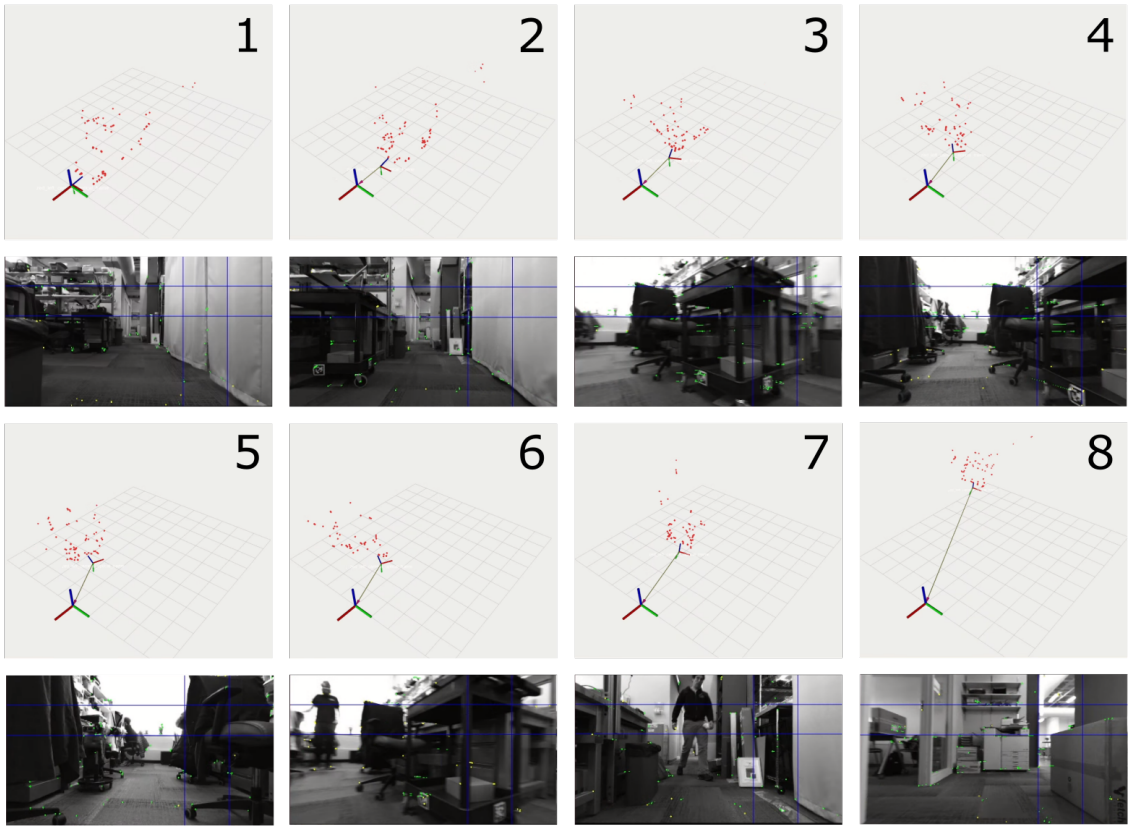


Figure 3.2: Snapshots of pose and position estimates of camera and detected features as the Turtlebot completes slow 90°turns. The robot navigates a straight path, turns 90°to the left, moves forward and backwards, then turns 90°to the right to continue along the original path. See Figure **??** for a complete description of the markings in the images.

people walking by, as shown in Figure 3.2, since the features were mainly immobile objects in the environment with distinct corners.

The program performed best when moving in a straight line forwards and backwards (Figure 3.1). Quick turns caused issues for feature detection, even when optimizing the number of grid cells and features per grid cell for accuracy and solvability. This is partially due to the frame rate of the camera limiting the number of frames that occur during turns and blurring; the number of features surviving between image frames is reduced during turns, providing very little visual information from the en-

vironment. This causes the program to underestimate the degree to which the robot turns, as can be seen in Figure 3.2.

3.3 Discussion

Overall, the use of sensor fusion in solving the SLAM problem is useful in its own right. Particularly, the efficiency of the matrix/graph structure in iSAM2 is well-suited for real-world scenarios, as can be seen in the success of using iSAM2 to solve online SLAM for the Turtlebot.

Though there is room for improvement, the goal of implementing visual odometry using iSAM2 was ultimately achieved. There is also value in exploring the addition of more sensors/factors to our algorithm.

3.4 Future Work

To further test the accuracy of our SLAM program, we plan to record the movements of the Turtlebot on VICON cameras in the motion capture area in PERCH. The VICON cameras will produce a ground truth path of the robot to compare with the robot location predicted by our algorithm.

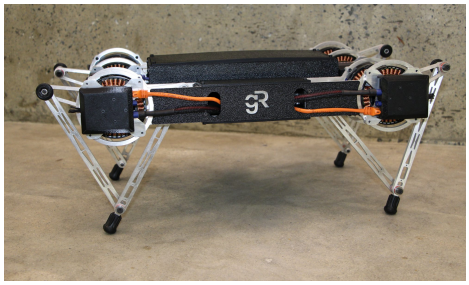


Figure 3.3: Ghost Minitaur^{TM}, the robot to which we ultimately hope to apply this algorithm. We plan to mount the ZED Mini stereo camera (Figure 1.1b) on top of the base and attach contact sensors to the grippers on the legs.

We also plan to integrate additional factors for legged robots into our algorithm to increase accuracy, such as forward kinematic and contact factors based on the leg kinematics and contact sensors, respectively. Then, we would need to move forward to testing the entire pipeline on the Ghost Minitaur$^{\text{TM}}$ (Figure 3.3).

Bibliography

[1] Ross Hartley, Josh Mangelson, Lu Gan, Maani Jadidi, Jeffrey Walls, Ryan Eustice, and Jessy Grizzle. Legged robot state-estimation through combined forward kinematic and preintegrated contact factors. pages 1–8, 05 2018.

[2] Frank Dellaert and Varun Agrawal. Gtsam 3.2.1. 2019. URL https://borg.cc.gatech.edu/download.htmldownload.

[3] Ke Sun, Kartik Mohta, and Christopher Choi. msckf_vio. 2018. URL https://github.com/KumarRobotics/msckf$_v io$.

[4] Vidula Kopli and Vasileios Vasilopoulos. Gtsam vio. 2019. URL https://github.com/vkopli/gtsam$_v io$.

[5] Michael Kaess, Hordur Johannsson, Richard Roberts, Viorela Ila, John J Leonard, and Frank Dellaert. isam2: Incremental smoothing and mapping using the bayes tree. *The International Journal of Robotics Research*, 31(2):216–235, 2012.