ROBO 599: Visual Intertial State Estimation on Wheeled and Legged Robots

Mitchell Fogelson MFOG@SEAS.UPENN.EDU

Abstract

Visual inertial odometry (VIO) is a method which determines the state and orientation of a robot using sequential images. State estimation is a critical component in allowing robots to navigate and manipulate the environment around them. We define the state of the robot as position, orientation, linear velocity and angular velocity.

$$X = [x, y, z, \phi, \theta, \psi, \dot{x}, \dot{y}, \dot{z}, \dot{\phi}, \dot{\theta}, \dot{\psi}]$$

State estimation, for a typical mobile robot, is usually calculated by fusing onboard sensors such as inertial measurement units (IMUs) and wheel encoders. With the improvements of camera and computer technology in recent years, VIO has become more reliable tool to improve robot state estimation in real time. This paper investigates integrating VIO to improve the robot state estimation. This project was motivated by improvements which could be applied to legged robots. However, the following study was only developed and tested on a TurtleBot, a differential drive robot. We used the Intel Realsense ZR300 RGB-D camera and RTAB-Map ROS package to calculate the odometry estimate. To fuse the sensor outputs of the system an Extended Kalman Filter (EKF) approach using the ROS robot localization package was used. This paper will discuss the motivation for the project, the experiments, results as well as discussion for further study.

1. Introduction

1.1. Motivation

Legged robots offer large benefits for outdoor mobility. They can easily adapt to a multitude of terrains and carry significant payloads (Haldun, 2009). Current systems, like the Minitaur Robot seen in Figure 1, are mostly remotely controlled by an operator and have no vision systems (Ghost, 2016). By improving state estimation of Minitaur it will have greater ability to navigate and manipulate its environment autonomously. This will allow for many useful applications including last mile delivery, warehouse monitoring and search and rescue.

Figure 1. The Minitaur Robot

Figure 2. The TurtleBot

1.2. Goals

The goal of this investigation was to improve robot state estimation by integrating realtime visual odometry methods. The project sought to develop a modular architecture that

University of Pennsylvania, Robo 599 Course Project.

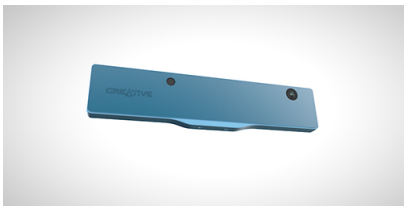

Figure 3. Intel Realsense ZR300

could be applied to any robotic system including but not limited to differential drive and legged robots.

1.3. Hardware and Software

This project took full advantage of the amazing hardware platforms and software packages supporting ROS. The main hardware platforms we used were the TurtleBot2, seen in Figure 2, and the Intel Realsense ZR300 RGB-D camera for our vision system, seen in Figure 3. The ZR300 has 4 cameras, two IR cameras for depth, a fisheye lens and an RGB camera. It also has a pattern emitter for the IR cameras to help with finding features, similar to how the Kinect stereo camera works. Furthermore, the ZR300 also has an onboard IMU for better orientation estimation. Intel also developed the Realsense2 ROS package for their cameras as well to ouput the image data as ROS support topics. To calculate the visual odometry from the camera we used the RTAB-Map SLAM package. Finally, to fuse all of our sensor inputs with an EKF we used the robot localization package (Labbe, 2014) (Moore, 2014). All of the packages mentioned above are open source material.

Bellow are configuration changes from the default settings we used for RTAB-Map:
1) Number of inliers for RANSAC: 10 Default: 20
Reason: This reduces the computational time for the RTAB to find loop closures
2) Optimization Strategy: GTSAM
Reason: This greatly improved the performance of the RTAB-Map odometry

The odometry estimates generated by these packages are with respect to the odom coordinate frame, a fixed frame generated by our initial pose. In ROS we specified all of the transforms between each of the coordinate frames. A transform tree can be seen below in Figure 5.

2. Experiments and Approach

We tested our system in the motion capture area of PERCH. We manually drove the robot and captured data

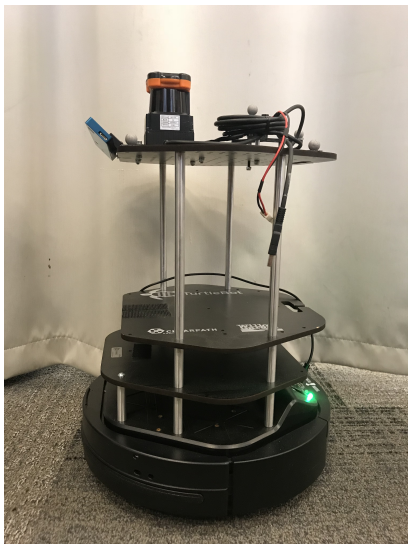

Figure 4. TurtleBot setup as seen from the side with the ZR300 at the top left of the robot.

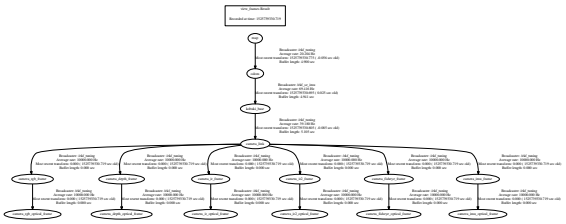

Figure 5. The transform tree for all of the local coordinate systems

from our sensors which we are able to evaluate with the ground truth using the Vicon system. We tested the following systems:

1) ZR300 and RTAB-Map Performance with respect to the camera mounted vertically or tilted towards the floor.
2) ZR300 and RTAB-Map Performance with respect to IR emitter on and off.
3) Kobuki odometry performance over long periods of time.
4) EKF performance

3. Results and Findings

3.1. RTAB-Map Performance WRT. Camera Angle

The ZR300 camera data sheet explains that the depth camera works best within a range of 0.5 and 2.8 meters. It is able to capture features farther than this, but this is the optimal range due to the emitter projection. When initially running ZR300 and RTAB-Map with the camera oriented vertically in the motion capture area, a large open space

with scarce features, the system performed poorly (see Figure 6). Many times the odometry would get lost due to few feature matches. When orienting the camera towards the floor we achieved much better results (see Figure 7). This is due to a combination of the speckle patter on the floor of the motion capture area, providing lots of strong features, as well as the emitter being able to project onto surfaces to create more features to track.

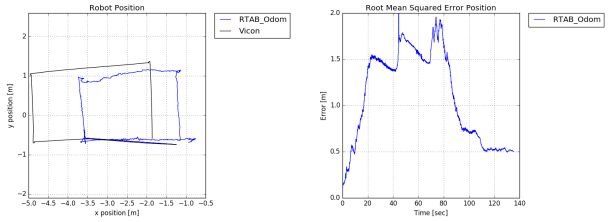

Figure 6. RTAB-Map odometry performance with ZR300 camera oriented vertically

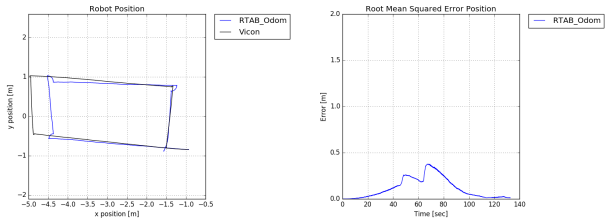

Figure 7. RTAB-Map odometry performance with ZR300 camera oriented tilted 30 degrees off vertical

3.2. RTAB-Map Performance WRT. Emitter

Given the results above about the improved performance of the camera tilted towards the floor, we were interested in validating the effects of the emitter. We tested the RTAB-Map and ZR300 setup with the emitter on and off. The results can be seen in Figure 8 and 9. The emitter has significant improvement to the performance of the system.

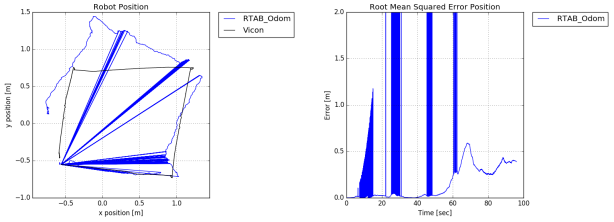

Figure 8. RTAB-Map odometry performance with ZR300 camera emitter off *(Areas with large spikes are due to RTAB-Map loosing its location)

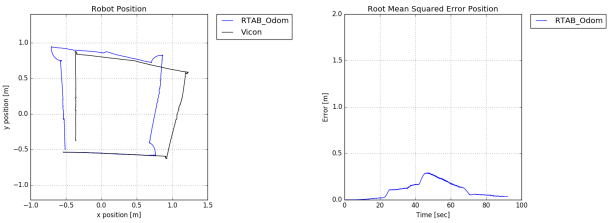

Figure 9. RTAB-Map odometry performance with ZR300 camera emitter on.

3.3. Kobuki odometry Performance WRT. Time

We expected the Kobuki odometry to have drift overtime due to propagating noise in the encoders. We found that the Kobuki uses it own internal EKF fusing an onboard gyro with the wheel encoders to limit this issue. We ran the robot for over 4 minutes in two trials, one with the robot operating slowly and one with a more aggressive behavior. The Kobuki robot performed surprisingly well in both trials. In the slow trial the Kobuki robot experienced minimal drift and had a maximum RMSE of 0.1m. In the more aggressive trial running the robot up to 0.5 m/s the error does show much stronger indications of drift. The error was still small, with a max RMSE around 0.4m. Given the trend of the error over time, it doest seem that the Kobuki robot is still susceptible to odometry drift for long term operations. See Figure 10 and 11 for results on the Kobuki performance with respect to time.

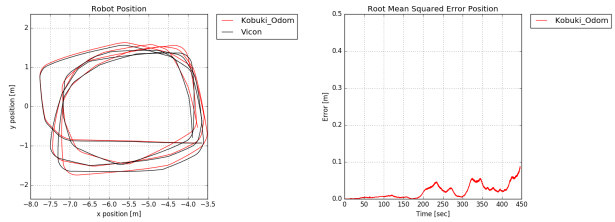

Figure 10. Kobuki odometry performance over 7.5 minutes at slow speeds

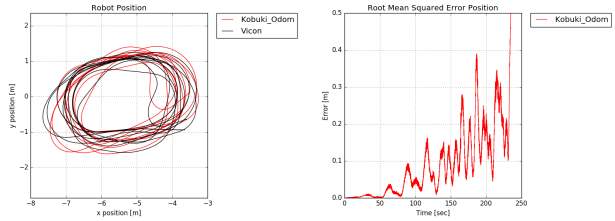

Figure 11. Kobuki odometry performance over 4 minutes at high speeds

3.4. EKF Performance

We used the robot localization ROS package to configure our EKF. We found out earlier today that we had missed a critical part when configuring our EKF. Our odometry from camera, IMU and Kobuki were in different frames. See Figure 12 for images of this critical mistake.

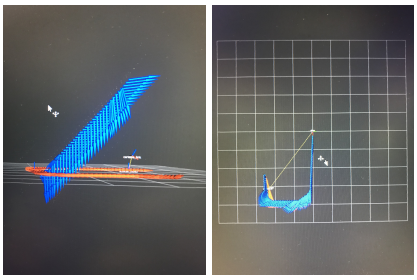

Figure 12. In these images you can see that the odometry for the camera is in the wrong frame, but when projected into the correct frame show that the correct results

4. Discussion

Much of the semester was spent figuring out ROS and how to operate and integrate the various hardware and software packages used. We found a critical error in the final our of configuring this report that would require the need to redo most of the data collection. This research will continue over the Summer under the continued mentorship of Vasileios Vasilopoulos. We intend to confirm these findings with the issue remedied before moving on. We believe this change will provide strong results and match our expectation. After this change, we intend to get a baseline performance of state estimation with the tools we developed over the semester on the Minitaur platform. We then intend to extend the research into using different methods for state estimation for Minitaur robot.

5. Conclusion

In conclusion, we were able to develop a modular architecture for robot state estimation using the ROS framework, regardless of robot platform. From this investigation we have a better understanding of the state of the art vision systems and VIO software. Overall, I learned a lot about implementing software using ROS, depth cameras, novel visual sensors such as event based cameras and SLAM. This investigation plans to continue and correct the errors described above.

6. Acknowledgments

Much thanks to my mentor over the semester Vasileios Vasilopoulos for his extensive help and Dan Koditschek for allowing me to pursue this study.

7. References

[1] Haldun Komsuoglu, "Dynamic Legged Mobility—an Overview", . May 2009.

[2] Ghost Robotics Minitaur, 2016. Available: http://www.ghostrobotics.io/minitaur

[3] Labbe, M. et al., "Online Global Loop Closure Detection for Large-Scale Multi-Session Graph-Based SLAM", Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 2661-2666, September 2014.

[4] Moore, T. et al., A Generalized Extended Kalman Filter Implementation for the Robot Operating System", Proceedings of the 13th International Conference on Intelligent Autonomous Systems, Springer, July 2014.