Enhancing PL-SLAM for Context-aware Mapping

Rajdeep Adak
Arizona State University
radak@asu.edu

Prajjwal Dutta
Arizona State University
pdutta9@asu.edu

Vivek Kulkarni
Arizona State University
vkulka16@asu.edu

Kunal Palasdeokar
Arizona State University
kpalasde@asu.edu

Niyati Vaidya
Arizona State University
nvaidy10@asu.edu

Abstract

In recent years, Simultaneous Localization And Mapping (SLAM) has emerged as an essential component in robot autonomy especially for path planning and intelligent robot navigation in complex environments. The efficiency and performance of SLAM algorithms are therefore critical deciding factors in facilitating the advancement of robot autonomy. This paper documents our engineering efforts and design decisions in implementing a software model of PL-SLAM. The project envisions to identify and incorporate possible enhancements with PL-SLAM.

1 Introduction

Among the available SLAM algorithms, a present state of the art approach, the ORB SLAM [1] has seen profound applications in autonomous systems and reliable path planning. The algorithm has found software implementations on various development tools such as MATLAB [2] and Unreal Engine [3]. Such tools also provide the computational capabilities to handle the associated data overhead and the necessary graphics pipeline to handle point-cloud visualization. Furthermore, they enable re-usability and portability to mobile-computing hardware for processing environment data and developing intelligent control algorithms. PL-SLAM (Point-Line Simultaneous Localization and Mapping), a new purely visual approach builds upon the ORB-SLAM framework, that integrates both point and line feature detection into the SLAM process. This combination enhances the system's robustness and accuracy, especially in environments where point features alone are insufficient for reliable mapping and localization. However, tools implementing the PL-SLAM algorithm are conspicuously absent in the scientific community. Hence, We aim to develop a reusable software implementation of the PL-SLAM algorithm.

2 Related work

In the past, methods like the Extended Kalman Filter (EKF)[4] were used, but they were limited in accuracy and scope. Bundle Adjustment (BA)[5] improved upon this, especially for larger areas. Many methods, including PTAM, help to track objects in the environment and some use more detailed features in images and improving camera tracking[6]. ORB-SLAM is an example of a system that excels at tracking a camera's movement and mapping its surroundings[7]. However, filtering based approaches outperform feature based methods. We explore the performance limitations of ORB-SLAM on the MATLAB based tool [6]. This also enables us to understand the software implementation of the mathematics associated with ORB-SLAM.

3 Baseline results

3.1 PL-SLAM Mapping approach

PL-SLAM optimizes the integration of 3D lines and 2D projections to minimize re-projection errors. It leverages normalized line coefficients and errors from both points and lines through bundle adjustment, refining camera poses with a unified cost function. Initial map estimates use line correspondences for rotation between views. This method enhances SLAM's robustness and accuracy in linear-feature-rich environments by balancing geometric precision with optimization, focusing on efficient and accurate mapping and localization.

3.2 ORB-SLAM Algorithm Overview

The MATLAB ORB-SLAM tool operates in three main stages: feature extraction, initial pose estimation, and optimization. It starts by extracting ORB features from the environment, which are invariant to rotation and scale changes. These features are used to establish correspondences between frames for initial pose estimation. The algorithm then optimizes the pose and map concurrently through local and global bundle adjustments, refining the trajectory and map for accuracy. Loop closure detection corrects accumulated drift by recognizing previously visited places and adjusting the map and trajectory globally.

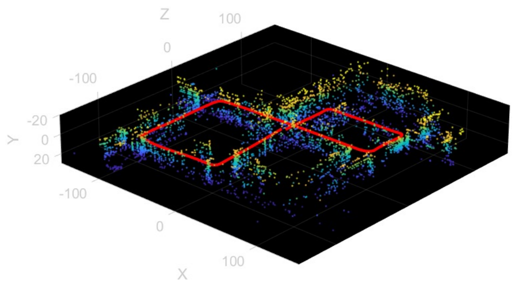

Figure 1: Point cloud generated with ORB-SLAM.

3.3 Implementing PL-SLAM in Unreal Engine

A general-purpose software implementation of PL-SLAM over virtual physical environments can allow simulation of PL-SLAM enabled robot dynamics and interoperability with other data visualization and processing tools. Unreal Engine provides the computational capabilities of implementing a mathematical formulation of the PL-SLAM algorithm. In the following images, various environments in the Unreal Engine have been built where PL-SLAM can be tested. Similar implementations of SLAM based algorithms in the Unreal Engine have been prevalent as the high-fidelity environment provides blue-printing of point-clouds and scripting with Python/C++ APIs. Secondly, the APIs can also be used to develop reusable code similar to the MATLAB ORB-SLAM tool.

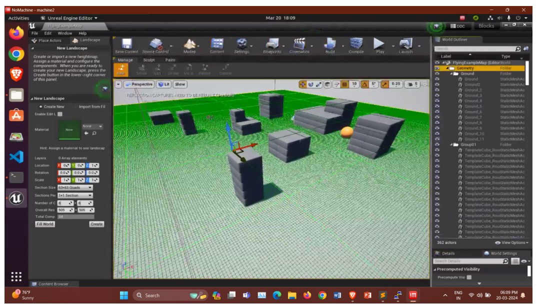

Figure 2: Sample environment for testing PL-SLAM.

4 Future work

While graphically designed 3D environments may be photorealistic, precise results of a PL-SLAM simulator can be best tested on precisely measured 3D-scans of actual environments. Perhaps, a true-to-scale 3D environment scan can be virtualized with PL-SLAM algorithm. For this, an essential step would be to develop modular generalised source code implementing the mathematical formulation of PL-SLAM data to generate a relevant data container. Such data containers can then be visualized over ground-truth reference. This will enable us to validate localization accuracy of PL-SLAM. While re-usability remains our primary goal, control of tun-able algorithm parameters can also be explored to provide flexibility of the PL-SLAM tool via an API. Secondly, API usability can also enable interoperability with other image recognition tools. Most monocular SLAM algorithms are not context aware and PL-SLAM presently lacks segmented point-cloud mapping. Therefore, object identification from point-clouds is challenging. Our research aims to steer towards the larger goal of enhancements on PL-SLAM by first making it reusable in software.

References

[1] R. Mur-Artal & J. D. Tardós, (2017) "ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras," in IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255-1262.

[2] “Monocular visual simultaneous localization and mapping,” Monocular Visual Simultaneous Localization and Mapping - MATLABSimulink, https://www.mathworks.com/help/vision/ug/monocular-visual-simultaneous-localization-and-mapping.html (accessed Mar. 21, 2024).

[3] https://www.mathworks.com/help/driving/ug/develop-visual-slam-algorithm-using-unreal-engine-simulation.html

[4] A. Davison, I. Reid, N. Molton, & and O. Stasse (2007) MonoSLAM: Real-time single camera SLAM. TPAMI, 29(6):1052–1067.

[5] A. Davison, I. Reid, N. Molton, & and O. Stasse (2007) MonoSLAM: Real-time single camera SLAM. TPAMI, 29(6):1052–1067.

[6] G. Klein & D. Murray (2007) Parallel tracking and mapping for small AR workspaces. In ISMAR, pages 225–234. IEEE.

[7] G. Sibley, C. Mei, I. Reid, & P. Newman (2009) Adaptive relative bundle adjustment. In RSS.