CMSC828T P2Ph1: Mapping and Localization

Yash Manian

I. INTRODUCTION

This project was divided into two parts, i.e. the Localization and Mapping parts. For the Mapping portion of the project, we are supposed to take in the detected corner points from the AprilTags in the scene per frame, and construct a map using the tag IDs as landmarks. We are given the intrinsic camera matrix and the transform from the camera to the IMU frame. The second part of the project requires the use of the map created in the first, and recreate the trajectory of the camera around the map using the map generated in the first part. It requires a non linear graph to optimize. This project was done in collaboration with Steve Gambino, and Homework 1 was heavily referenced along with the examples in the gtsam toolbox.

II. MAPPING

A. Factor Graph

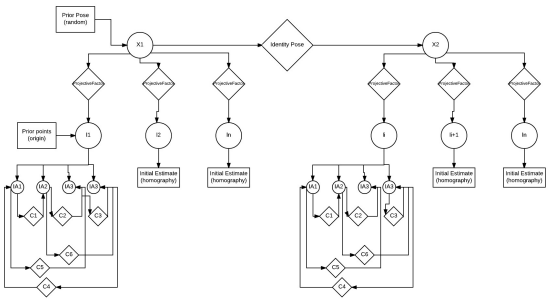

Fig. 1. Factor graph

The factor graph treats each frame as one state, Xi. The corner points for tags detected in each frame are considered as landmarks li. There is a projection factor in between the landmarks and the state, which takes in the camera matrix K. The prior for the first state is a random pose, and the priors for the origin tag, if detected in the first frame, is the origin. If the origin tag isn't detected in the first frame, the first detection is chosen, and is then made the origin tag. Later, when the origin tag comes into frame, a linear transform is performed and the origin is shifted. There are six constraints applied between the points of the detected tag. The constraints are basically the ideal distances between the tags. Initial Estimates for the landmarks and states are applied. The initial estimate for the state is a random pose. Whereas, the initial estimates for the landmarks are computed through homography.

B. Code

The main challenge for this project was coming up with the factor graph and figuring out the initial estimates. The initial estimates are calculated using homography per frame. The reference tag, i.e. tag 10 has its homography calculated with respect to its actual position:

$$Ref = \begin{bmatrix} 0 & 0 & 0 \\ TagSize & 0 & 0 \\ TagSize & TagSize & 0 \\ 0 & TagSize & 0 \end{bmatrix} \quad (1)$$

The toolbox used to compute homography was Peter Kovesi's computer vision toolbox. The homography2D function was used. Once the homography is calculated, each point in the frame is then multiplied by the homography matrix to attain their position in the world frame. The rotation and translation of the camera are also calculated. These calculated world frame values are then passed into the initial estimates.

C. Optimizer

The factor graph is passed into the optimizer along with the initial estimates. This returns an optimized value of the landmarks along with the estimated pose for each frame. The optimizer used here is the Dogleg optimizer.

D. Results

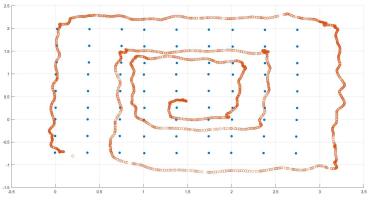

Fig. 2. XY

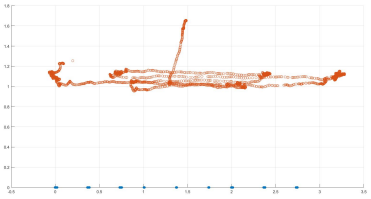

Fig. 3. XZ

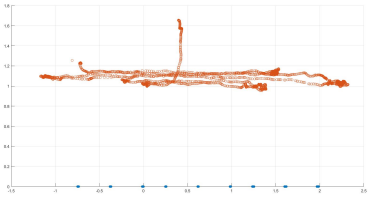

Fig. 4. YZ

III. Localization

For Localization, I used the ISAM2 function from the GTSAM toolbox. Even though the factor graph remains the same, the major difference between GTSAM and ISAM2 is that the process is incremental, and can therefore be run in real time. Unlike GTSAM, which creates a whole interconnected graph as a continuous entity, ISAM2 creates a new graph every state and uses priors to optimize.

A. Factor graph

The factor graph remains the same, except that the initial estimates for the landmarks come from the Mapping code results i.e GTSAM. Also, it is recreated every frame.

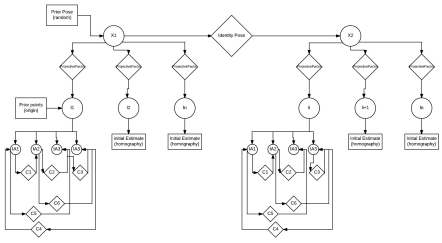

Fig. 5. Factor graph for iSAM

The first two frames have priors in this case. The priors are the same as the initial estimates. Pose estimates are calculated using the Computer Vision toolbox's PnP solver.

IV. Results

The results are relatively accurate as opposed to the PnP solver estimates. The incremental SAM creates a new graph every frame, and the initial estimates are from the PnP solver. The test cases were run where for a given trajectory, GTSAM and ISAM2 were run and the error in position and orientation were observed.

A. Straight line

This test case consists of the camera moving in a straight line with respect to the landmarks on the floor. The GTSAM isn't quite as accurate because all the landmarks aren't observed during course of the trajectory

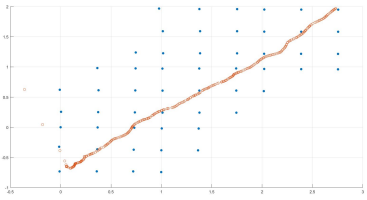

Fig. 6. GTSAM optimization XY view for Straight line trajectory

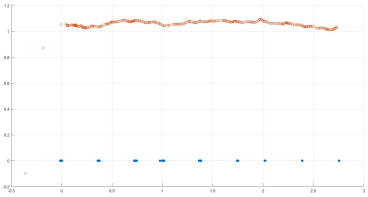

Fig. 7. GTSAM optimization XZ view for Straight line trajectory

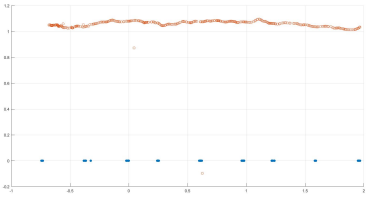

Fig. 8. GTSAM optimization YZ view for Straight line trajectory

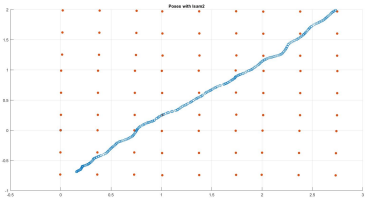

Fig. 9. iSAM optimization XY view for Straight line trajectory

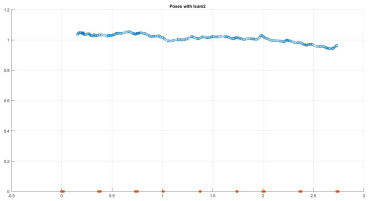

Fig. 10. iSAM optimization XZ view for Straight line trajectory

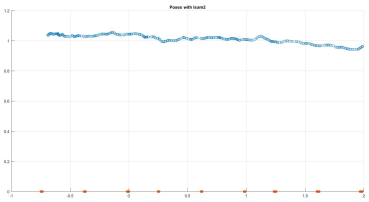

Fig. 11. iSAM optimization YZ view for Straight line trajectory

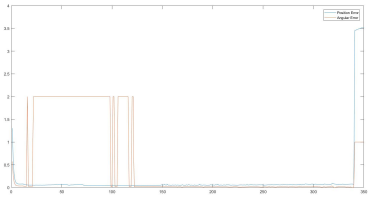

Fig. 12. Straight Line Position and Angular error with GTSAM

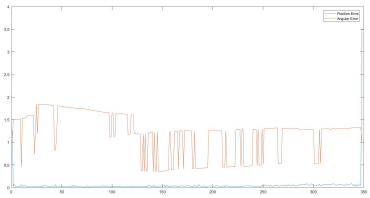

Fig. 13. Straight Line Position and Angular error with iSAM

B. Fast Circle

This test case consists of the camera moving in a straight line with respect to the landmarks on the floor. The GTSAM isn't quite as accurate because all the landmarks aren't observed during course of the trajectory

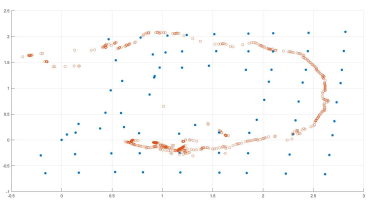

Fig. 14. GTSAM optimization XY view for Fast Circle trajectory

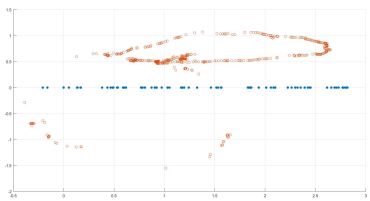

Fig. 15. GTSAM optimization XZ view for Fast Circle trajectory

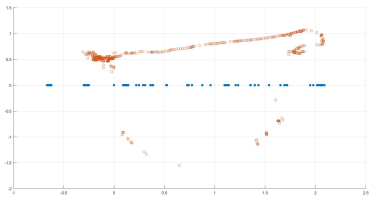

Fig. 16. GTSAM optimization YZ view for Fast Circle trajectory

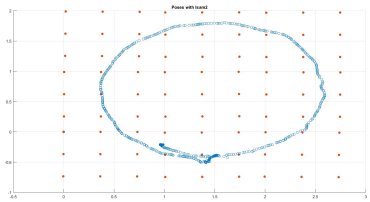

Fig. 17. iSAM optimization XY view for Fast Circle trajectory

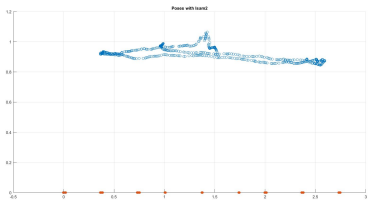

Fig. 18. iSAM optimization XZ view for Fast Circle trajectory

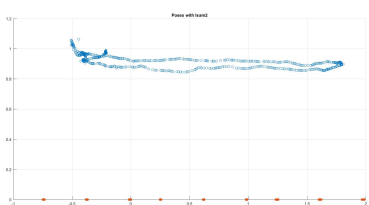

Fig. 19. iSAM optimization YZ view for Fast Circle trajectory

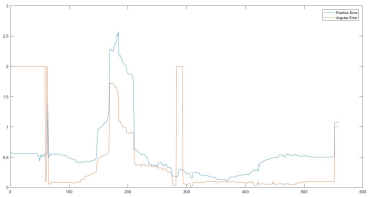

Fig. 20. Fast Circle Position and Angular error with GTSAM

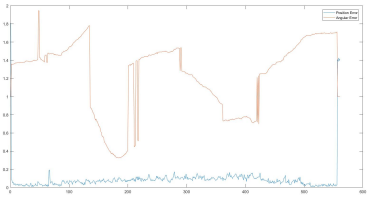

Fig. 21. Fast Circle Position and Angular error with iSAM

C. Square

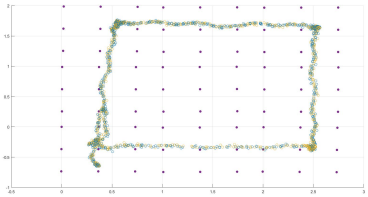

Fig. 22. iSAM optimization XY view for Square trajectory

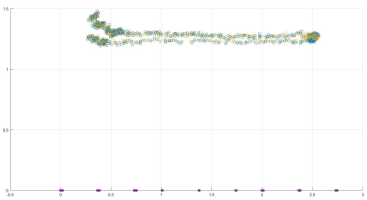

Fig. 23. iSAM optimization XZ view for Square trajectory

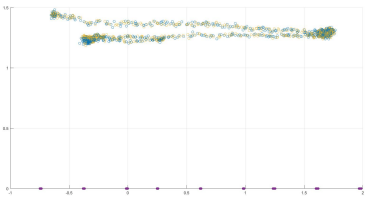

Fig. 24. iSAM optimization YZ view for Square trajectory

D. Mountain

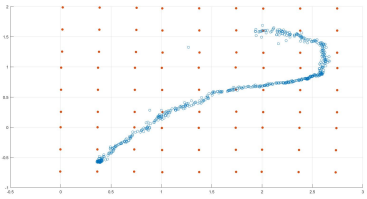

Fig. 25. iSAM optimization XY view for the Mountain trajectory

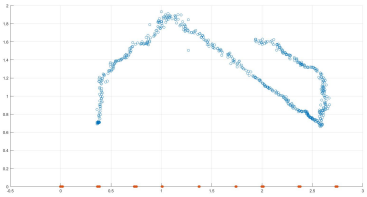

Fig. 26. iSAM optimization XZ view for the Mountain trajectory

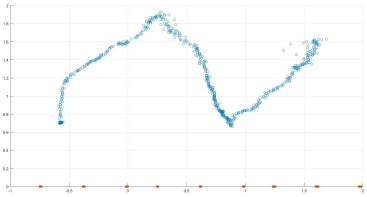

Fig. 27. iSAM optimization YZ view for the Mountain trajectory

V. Analysis

A. GTSAM

For the mapping portion of the project, there is no ground truth given for the landmark positions, however, operating on the assumption that the landmarks form a uniform grid, the output from the optimizer with respect to the initial estimates from homography show a clear improvement. As for the Localization, the accuracy of the trajectory depends on how many tags have been observed. This leads to a fairly high location error in both the Fast Circle and Straight Line trajectories. However, angular error is relatively low compared to iSAM. I am unsure as to why this is. The possible hypothesis being that the homography2D is better at estimating the initial estimates for rotation when compared to the Computer Vision toolbox.

B. ISAM2

The incremental nature of ISAM intuitively should trade off accuracy for speed.However, over both test cases i.e. Fast Circle and Straight Line, the location error is lower than the GTSAM estimation

by a fair margin. This leads me to conclude that the factor affecting this might be the pre-mapped landmarks which are fed as initial estimates to the factor graph in iSAM. The angular error is however larger than that of GTSAM. This might be attributed to the use of the getCamerPose from the Computer Vision toolbox as opposed to the homography2D function from Peter Kovesi's toolbox. As for the other trajectories, the shapes appear consistent with what was expected from them. There are a few outliers which can be explained by the noise parameters chosen.