Project 1 Phase 1

Rahul Subramonian Bama

Abstract— **SLAM using GTSAM was performed in different scenarios**

I. INTRODUCTION

In this project, a map of the ARTags is made and the drone is made to localize itself on the created map.

GTSAM is used to create the initial map, which is built using the detected AR Tags, and then iSAM is used to localize the drone on the created map.

II. GTSAM

For creating the initial map using GTSAM, the following steps were followed:

1. If the same tag has been detected multiple times, the excess tags are removed.

2. Tag 10 is defined as the origin tag and the location of the corners of the tag are defined as (0,0,0), (tagSize,0,0), (tagSize,tagSize,0) and (0,tagSize,0) counter clockwise starting from the bottom left.

3.The relative positions of the corners of the remaining tags are found with tag10 as reference. This is done by first getting the pose (R,T) w.r.t tag10 using RPnP. With the computed R,T, the Projection matrix is defined as P = K*[R,T]. With the projection matrix the lambda is found by taking the mean of the last row in the matrix P*[World coordinates of Tag10]. With this lambda the relative world coordinates of the remaining tags are found as inv(Image coordinates*lambda)*P. The final X,Y,Z are found by dividing the 1st, 2nd and 3rd rows with the last row of inv(Image coordinates*lambda)*P.

4. These world coordinates of these tags are stored in a separate variable so that the next time these tags are detected the pose (R,T) is calculated using these coordinates. (This is done to make it more robust).

5. The calculated Pose (R,T) is inserted as the first factor in the factor graph using the function: `GenericProjectionFactorCal3_S2`.

6. The detected tags are inserted as factors with (measurement) variables w.r.t the first factor.

7. For the next frame, the pose is found out by taking the world coordinates of the tags that are detected which have already been found in the previous frames.

8. The world coordinates of the remaining tags are found similarly (Step 3).

9. The current pose is inserted into the factor graph as a factor which is connected to the previous (pose) factor with an odometry noise.

10. If new tags are detected, then they are added as factors with (measurement) variables for the current factor. For the tags that are already detected a (measurement) variable is specified for the already existing factor.

11. Steps 7 to 10 are repeated till the drone comes to a stop.

12. The constraints for the tags and the initial estimates for all the factors were specified. The prior for the 1st pose was specified.

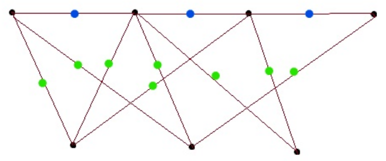

Fig. 1. Example Factor Graph.The small black dots are factors which are either pose or tags. (Note: the tag has 4 corners and each has constraints between them, they are represented as a single unit in the bottom black dots). The top black dots represent poses and they have a conditional probability with a odometry noise which is denoted by the blue dot. The green dots are the (measurement) variables.

III. iSAM

The localization using the map built in GTSAM is done as follows:

1. For the first frame the pose is found out using the world coordinates of tags in the frame.

2. This pose is inserted as the first factor into the graph. The tags that are detected in this frame are also inserted into the factor graph with the corresponding (measurement) variable. (Similar to GTSAM)

3. The initial estimates for the tags and the prior factor for the pose are given.

4. The factor graph is optimized and is updated using `isam.update` and the results are calculated as `isam.calculateEstimate()`.

5.For the next frame, the pose is calculated and then inserted into the factor graph as a factor connected to the previous (pose) factor.

6. The measurement factors for the detected tags in this frame are inserted appropriately (Step 10 in GTSAM).

7. The priors and initial estimates of the tags are initialized. The inital estimate of the pose is also initialized.

8. The factor graph is optimized and the results are calculated similar to step 4.

9. Repeat steps 5 to 8 till the drone comes to a halt.

IV. Results

The results for DataMapping and DataSquare are plotted here. The results for the other cases showed an error because the RPnP returned a NaN value and I couldn't correct it no matter what. When I solved the iSAM for each time step the result it showed was zero for some reason. I tried using different odometry noise and tag noise but the result still showed zero.

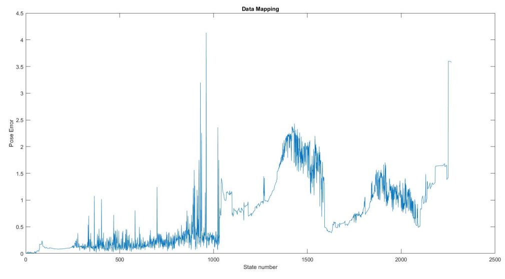

Fig. 2. Error in Pose of DataMapping

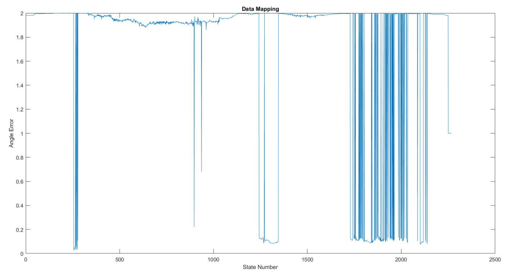

Fig. 3. Error in Angle of DataMapping

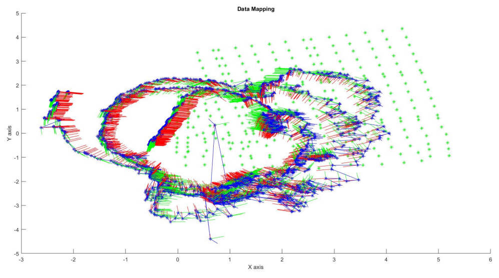

Fig. 4. DataMapping: XY Estimated Position

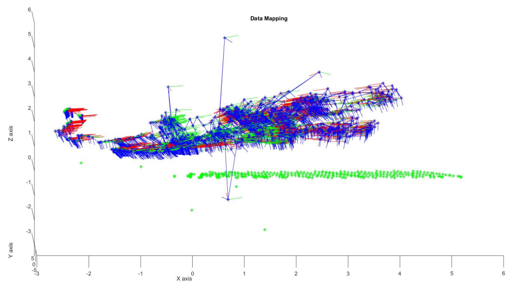

Fig. 5. DataMapping: XZ Estimated Position

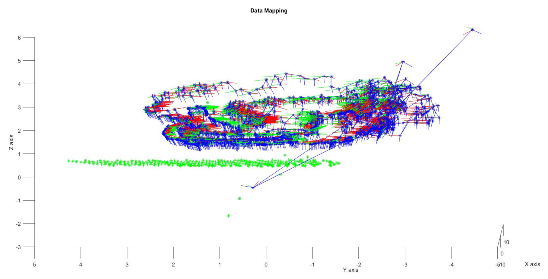

Fig. 6. DataMapping: YZ Estimated Position

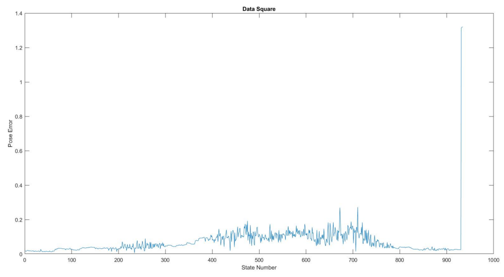

Fig. 7. Error in Pose of DataSquare

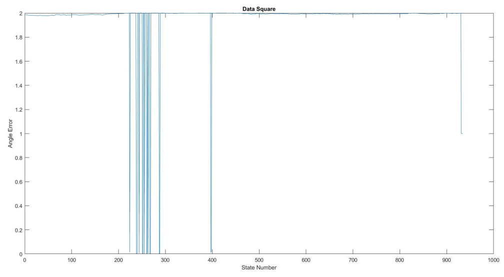

Fig. 8. Error in Angle of DataSquare

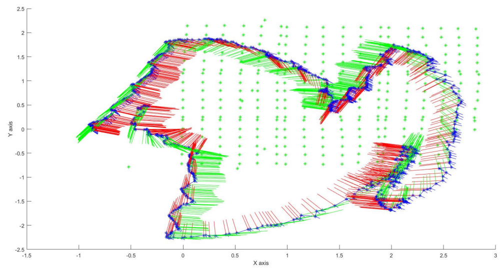

Fig. 9. DataSquare: XY Estimated Position

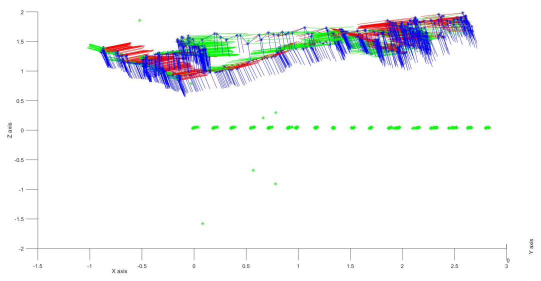

Fig. 10. DataSquare: XZ Estimated Position

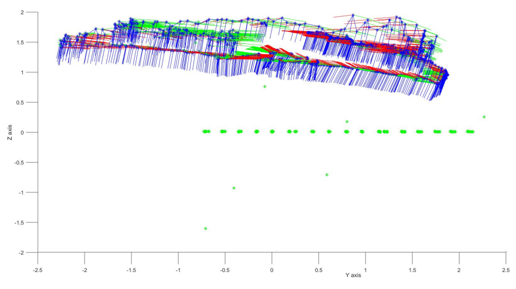

Fig. 11. DataSquare: YZ Estimated Position