Project 4 Report

Yizhan Ao(Email: josephao@umd.edu) Yingqiao Gou(ygou@terpmail.umd.edu)

UID:116022064 UID: 115979000

1. Work Distribution

We have distributed the work equally throughout the project so we both learned from the course material to complete the project.

2. Introduction

The goal of this project is to determine the drone camera's 3-Dimensional position and posture in the global frame using **AprilTag** recognition from the given frames. We're also attempting to map the environment. The key of simultaneous localization and mapping (SLAM) is that we must deal with these two variables simultaneously and can only make guesses. Fortunately, we have limitations that we can apply with some precision to guide and modify our estimations.

3. Tasks

In general there are two main tasks in this project.

1. SLAM mode

2. Localization mode

From SLAM, the map generated by (GTSAM-toolbox) was given the April-Tag results by optimizing the factor graphs.

In localization mode, the use of constructed map was based on the observation measurements the drone makes.

4. Steps

a. Initialize the camera pose and homography

The world coordinates must be properly setup in order for GTSAM programs to optimize them appropriately. Knowing that the origin in world coordinates will be the bottom-left corner of tag 10, the poses of the camera frames including tag 10 may be calculated using the compute Homography helper function. We know the drone will be moving slightly so the common corners which also is landmarks are existed.

b. Initialize Frame landmark position and build Factor graph

We initialize world coordinates point estimates with the tags' world coordinates we computed pre-GTSAM.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & r_3 & T \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

c. Camera pose estimation

We added the numbers we generated based on the first section to the initial homography estimation. This contained the estimated distances as well as the real-world locations of the landmarks. We encountered an issue in this step that we created the projection factor, but literally forgot to add it to the graph. The rest of the camera position just goes to the origin point.

d. GTSAM and Optimizer

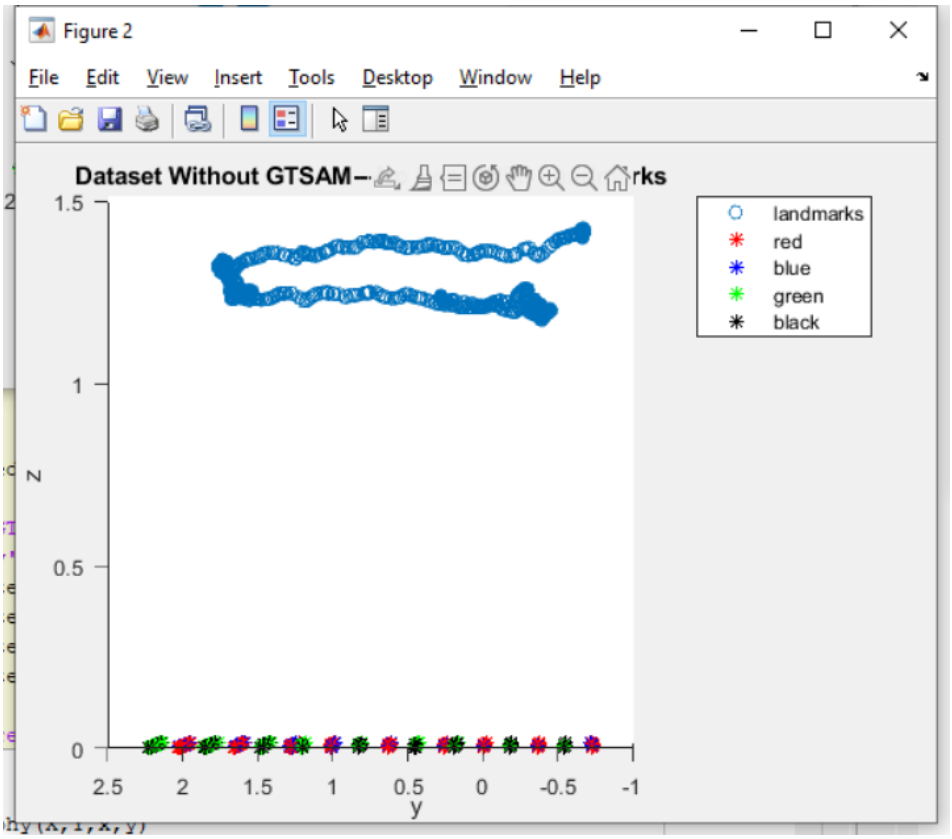
We have used the Levenberg Marquadt non-linear optimizer from the GTSAM library since the first thought dog-leg optimizer didn't work properly. We have separated graphs and compiling issues which are confusing to solve.

e. Plot the new poses and corner points.

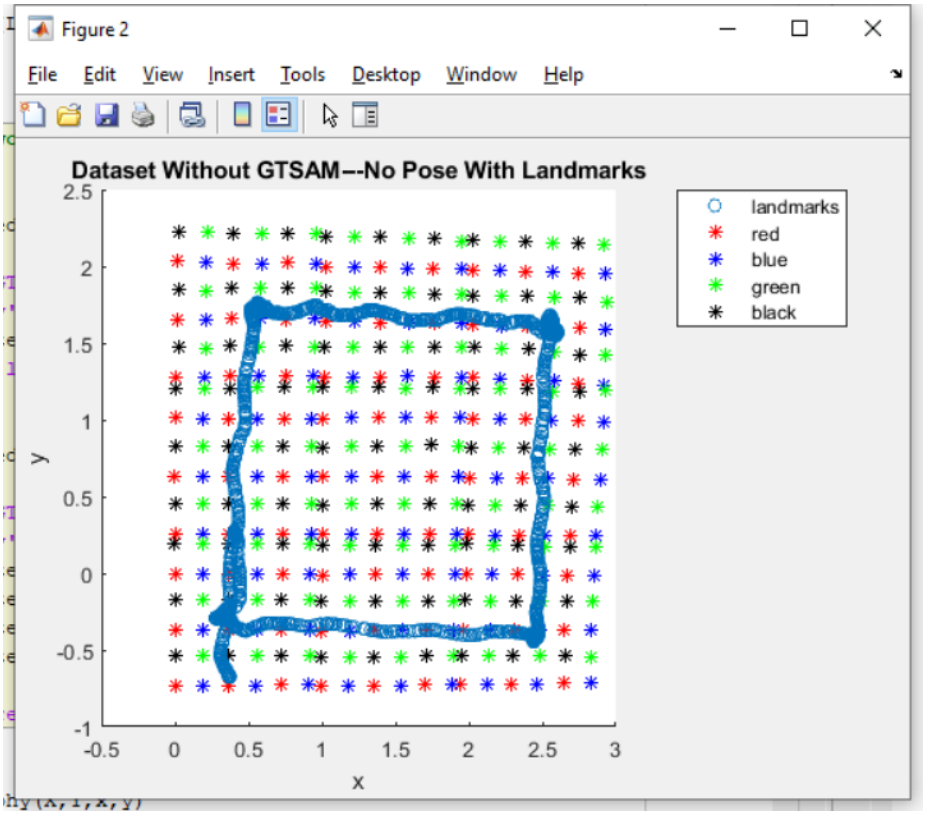
5. Results

We adjusted the picture outputs manually to show results of the GTSAM compared to without GTSAM. The pictures from the dataset 1 worked well in our optimizer. From the figures below, we could tell the GTSAM applications were a lot better than without. There are slight differences in pre and post GTSAM however, when we take a look at the squared pre and post square the differences are huge this could be resulted from our bad gaps and errors while computing the optimizer lambda from GTSAM. The differences in data mapping pre and post GTSAM is quite similar also hard to be spotted differences. Our pre-gtsam model yields a too good response that it may seem no improvements to the post-gtsam. We concluded that the factor graph created following was nearly identical. The sensors are too precise for GTSAM to make a meaningful effect.

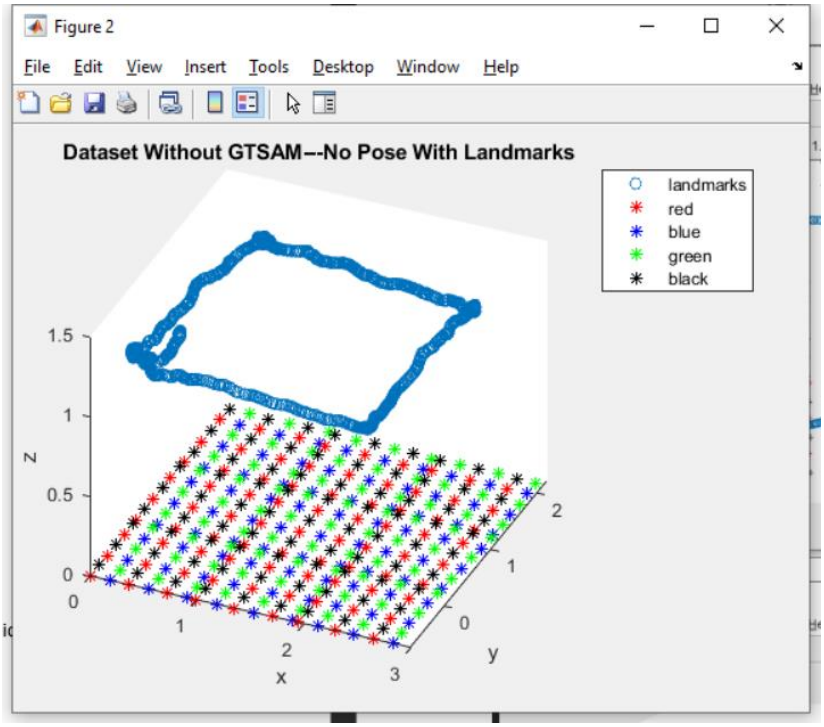
DataSquareFall2020.mat



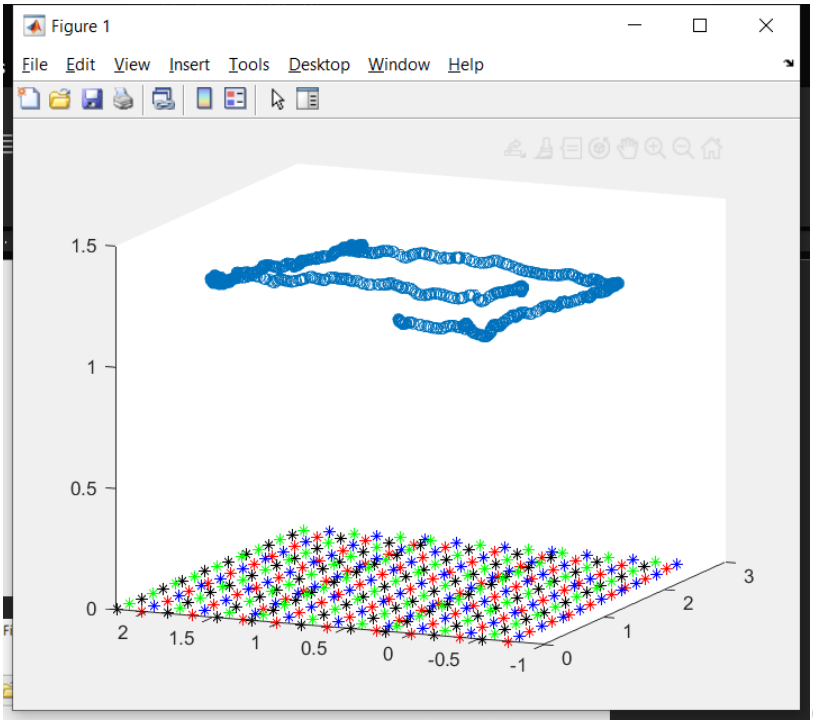
(Side View)



(Top View)

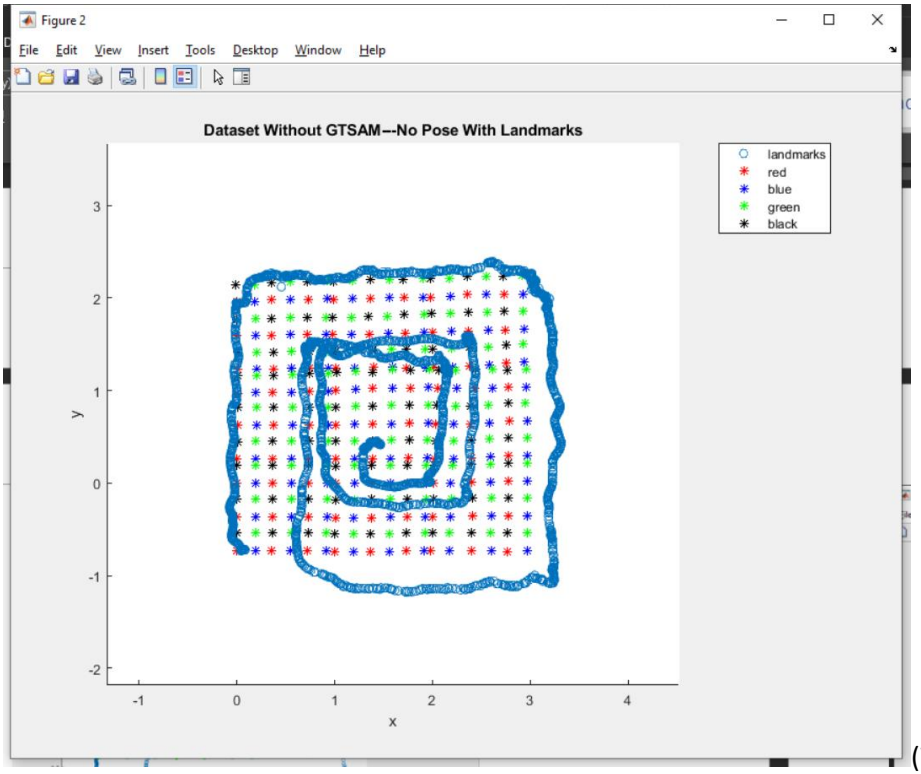


(From down to upwards)

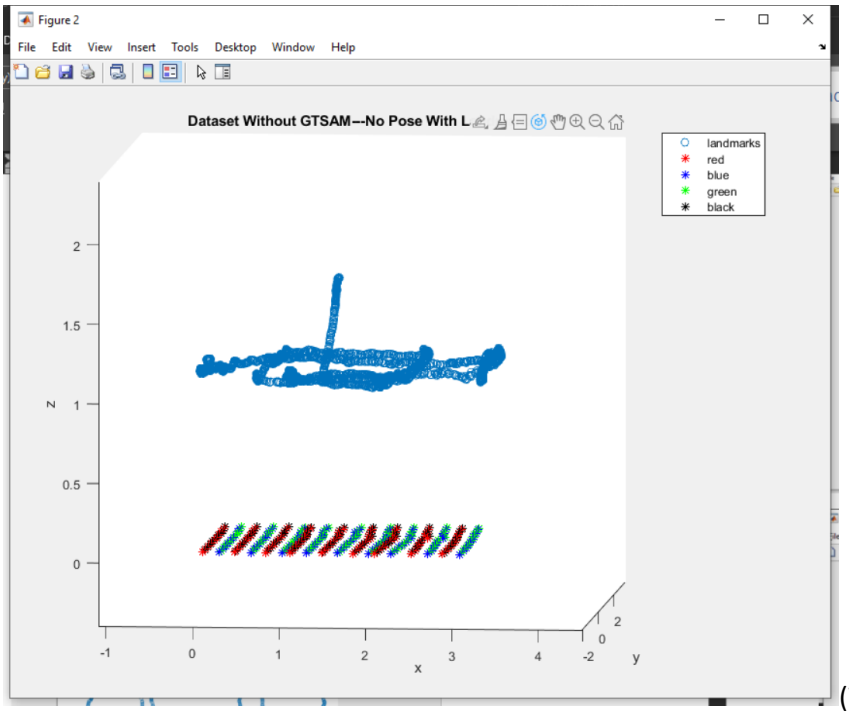


(The view from above)

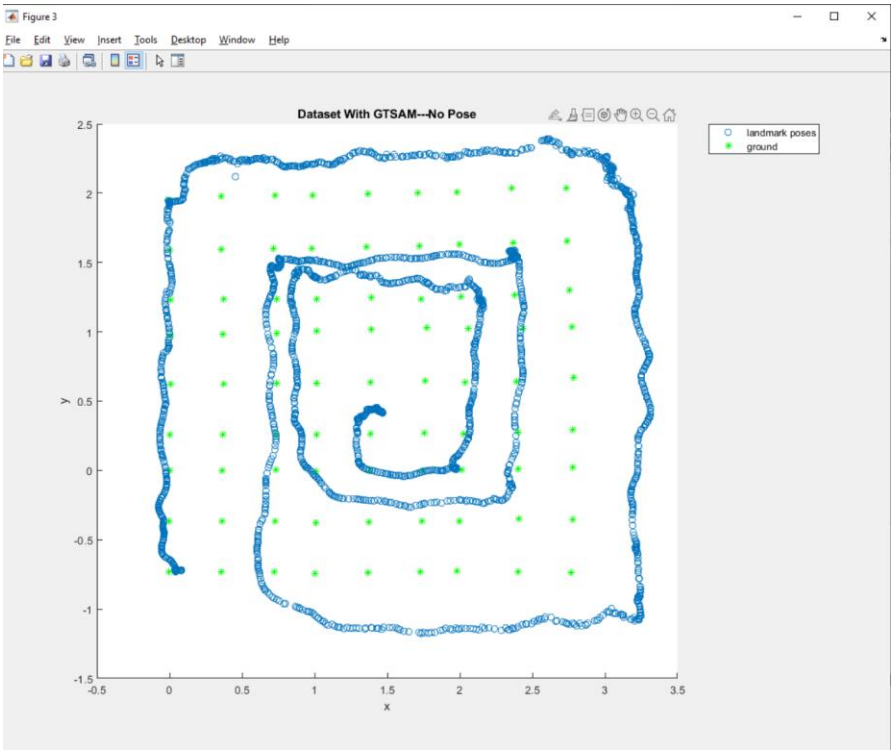
a. DataMappingFall2020.mat



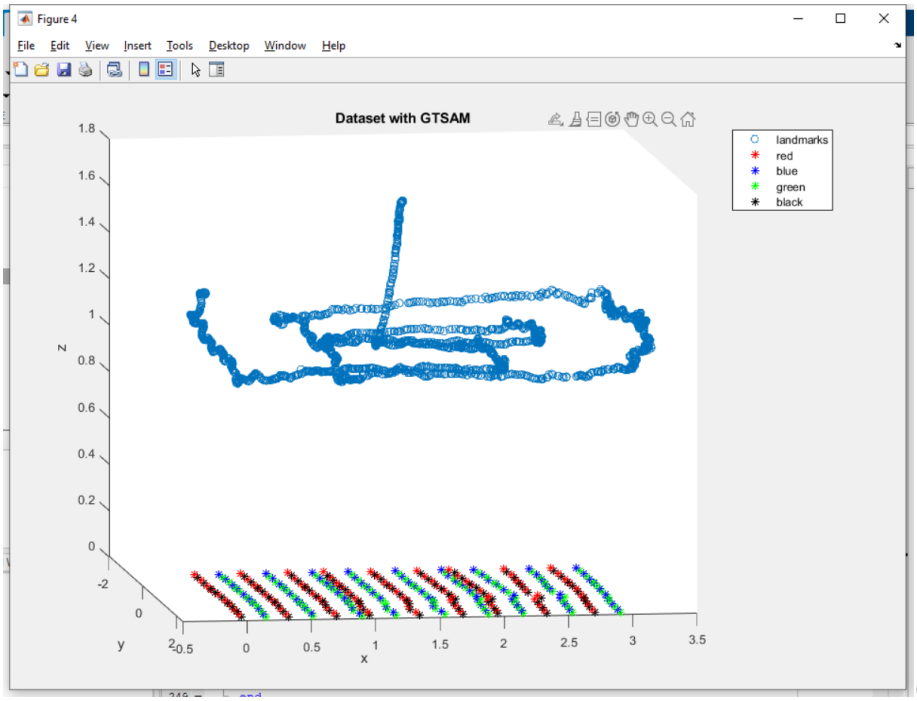
(Without GTSAM top view)



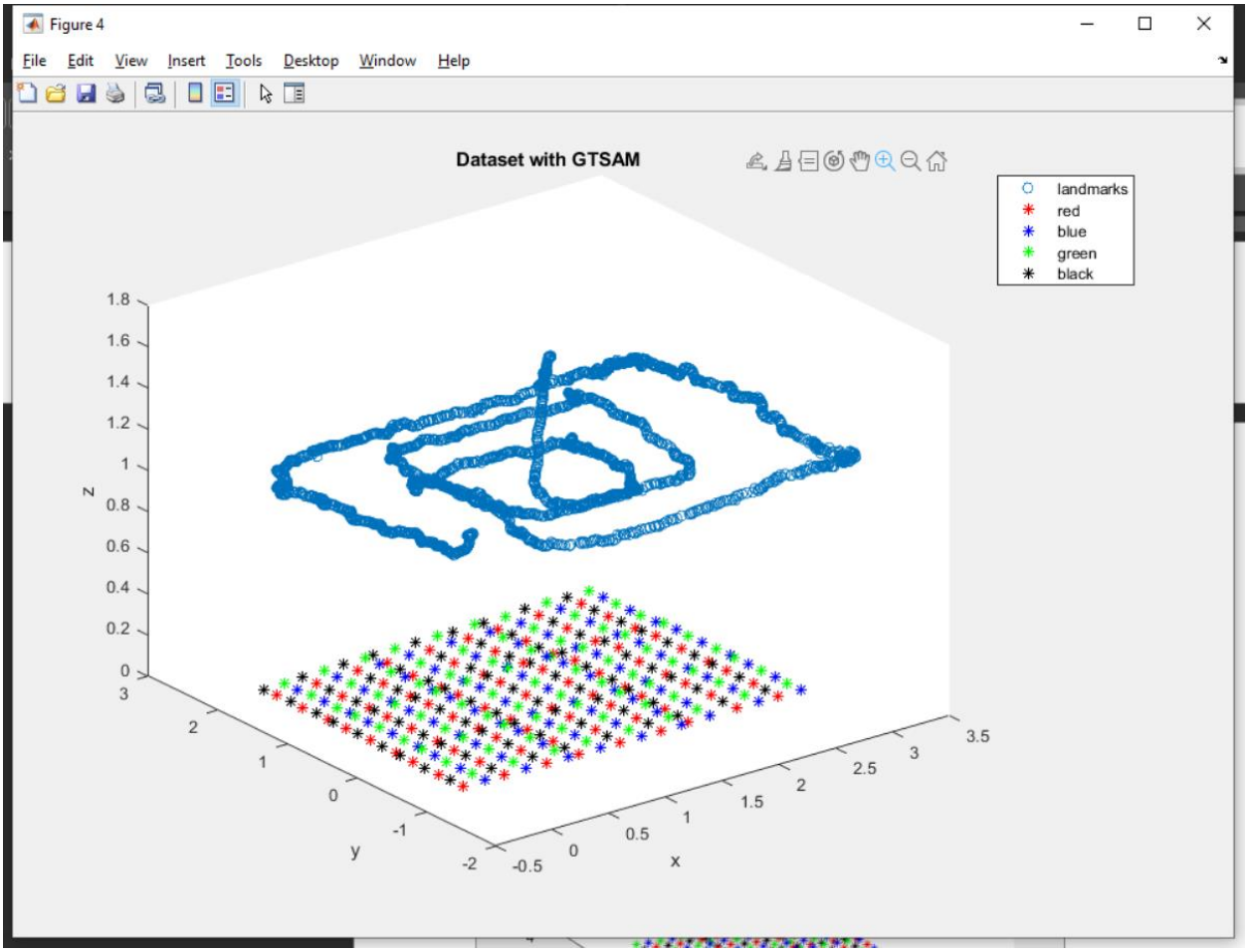
(Without GTSAM side vie)



(Top View)



(Side View)



(Pose trackingGeneral view)