CMSC 828T
Vision, Planning and Control in Aerial Robotics
Homework 1: SLAM using GTSAM
Due on 12:29:59PM on Oct 6, 2017

Prof. Yiannis Aloimonos, Nitin J. Sanket,
Kanishka Ganguly, Snehesh Shrestha

October 3, 2017

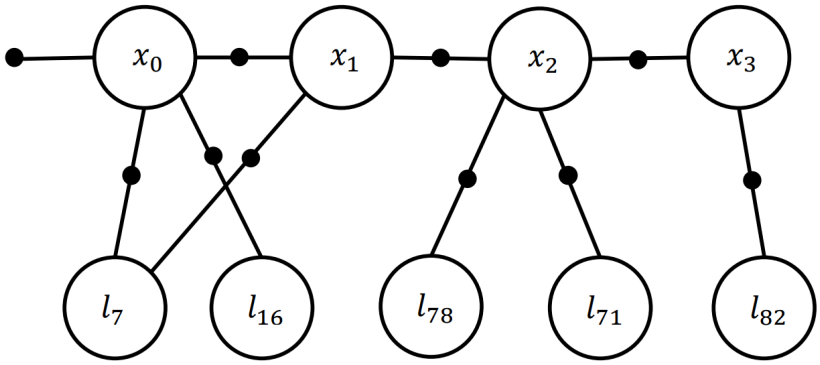
1 Introduction

Factor graphs are graphical models that are well suited to modeling complex estimation problems such as Simultaneous Localization and Mapping (SLAM) or Structure from Motion (SfM). You might be familiar with another often used graphical model, Bayes Networks [1]. Unlike Bayes Networks, which are directed acyclic graphs (DAG), factor graphs are bipartite graphs consisting of factors connected to variables. Variables represent the unknown random variables and factors represent probabilistic information on those variables, derived from measurement or prior knowledge.
In the first phase of the upcoming project, we will try to solve the most commonly used estimation problem in robots: SLAM. Before digging deep into the problem, let us solve a 'toy problem' for the same. For both Homework 1 and Project 2 Phase 1, we will be using GTSAM[2] toolbox ("Georgia Tech Smoothing and Mapping") which can be downloaded from [3]. It's surprising how well the factor graph frameworks can solve a complex problem (like SLAM) effortlessly, providing state-of-the-art solutions to the SLAM and SfM problems. *Excited already??*
For more details about the homework refer to the class slides [4] and Lecture video [5].

2 Task

Given a set of noisy observation measurements of landmarks and odometry data, we have to simultaneously estimate the robot pose and landmark locations in the world frame.
For inputs and outputs, check the comments in `SLAMusingGTSAM.m` function. The inputs can be found the `HW1.mat` file. **Note:** In GTSAM toolbox, the bearing angle is defined as the angle between the vector made by `robot` to `landmark` and the robot heading angle (robot pose).
In MATLAB, `BearingAngle = atan2(LandMarkY, LandMarkX) - Pose(3);`

x_i: Robot Pose



l_k: Landmarks

Figure 1: Factor Graph for a single sensor system

Before calculating error in pose using GTSAM, compute the error on the basis of dead reckoning. It means simply computing the error based on path integration. The error in pose from GTSAM should be less than dead reckoning error else we are doing something wrong with GTSAM. Note that the noise in robot's translational movement is 30% and in rotational movement is 10% of the movement. Moreover, the camera/distance sensors are not perfect in nature. The error in estimating the distance from the robot to the landmark can be assumed to be about 0.1m in both X and Y direction. Assume all noise to be modelled by *Gaussian* distribution of zero mean.

3 Grading Scheme

GTSAM is supported on Mac OS, Windows, and Linux but we are officially supporting Linux only! However, you are free to use any OS that best suits you.

- The homework will be graded on the basis of the error in your computed poses.
- Since you will be running the GTSAM toolbox directly, which contains MEX files, timing is not going to be a criterion for grading. However, to prevent extinction of the grader, there will be a hard cutoff at 30 seconds.
- Your function must return a matrix where the landmarks are in ascending order. If any of your landmarks are missing, it will result in an immediate failure of the test.
- As always, ensure that your all your code exists inside the `code` folder, which will be zipped into a `code.zip` and submitted.
  - Please ensure no extra folders are created or uploaded at submission time.

  - Please ensure no visualization related code is uploaded at submission time. It is fine to use them for your debugging.

- Since this is, essentially, an optimization problem, we will run each test case 5 times and take the best score from them.

- We will scale your pose errors for each landmark and provide a final score out of 100 points.

3.1 Submission Guidelines

Please submit your code via CMSC 828T submit section. You should create a folder called `code` and copy `SLAMusingGTSAM.m` into it, zip it, and submit `code.zip`. Please note the zip file needs to be `.zip` format. Any other format is not valid.

Please note:

- Do NOT add or submit any sub-folders.

- Do NOT submit any visualization code. If you have any either remove them or comment them out.

- Do NOT print out any outputs. If you have any debug code printing outputs to the console, please remove them or comment them out.

- Only include the files that are listed below and any new dependent m-files you might have created.

- Do NOT submit any other files that are not necessary and was created only for your testing/ debugging.

Your submission should contain:

- A README.txt detailing anything we should be aware of.

- A LateDays.txt with just the number of late days you would like to exercise for this submission.

- All necessary files inside one folder `code` such that we can just run the automated testing script to see your results. The expected files for this assignments are:

  - `SLAMusingGTSAM.m`
  - And any other new m-files you might have created that is necessary for running your code.

4 Collaboration Policy

You can discuss with any number of people. But the solution you turn in MUST be your own. Plagiarism is strictly prohibited. Plagiarism checker will be used to check your submission. Please make sure to **cite** any references from papers, websites, or any other student's work you might have referred.

References

[1] Bayes Networks. https://en.wikipedia.org/wiki/Bayesian_network.

[2] GTSAM Documentation. https://research.cc.gatech.edu/borg/sites/edu.borg/files/downloads/gtsam.pdf.

[3] GTSAM Download. https://research.cc.gatech.edu/borg/gtsam.

[4] CMSC 828T Lecture 10 Slides. https://cmsc828t.cs.umd.edu/slides/Class10%20GTSAM.pdf.

[5] CMSC 828T Lecture 10: Factor Graph Based Filtering Using GTSAM. https://www.youtube.com/watch?v=o6jEKbnqvTU.