CMSC 828T
Vision, Planning and Control in Aerial Robotics
Project 2 Phase 1 (P2Ph1): Search and Rescue
Due on 11:59:59PM on Oct 26, 2017

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

October 26, 2017

1 Introduction

Welcome to the Special Forces of the Disaster Relief team! We hope you enjoyed your short 2 day training program we called "Homework 1". Now that you are ready for the next challenge, your mission is to find the survivors in a power plant that has been destroyed by a recent earthquake. You will be in the intelligence development team. Due to the hazardous materials, you will be using drones (quadcopters) for this mission. There are two parts to this mission. First, since the plant is in disarray, your job is to build the map of the current condition of the plant. You will use a light weight drone with nothing but a single camera and an IMU, that will explore the plant and build the map. The second drone is capable of carrying other needed supplies for the rescue mission. The second drone will use the map from the first drone and go to the rescue target location with the supplies. Remember, there is not much time left. The search and rescue mission relies totally on you. To assist you with this mission, the commanders have provided you with some guidance to help you accomplish this mission. Good luck!

2 The Task

The task has been divided into two "modes", first being the SLAM mode (Refer to `SLAMUsingGTSAM` function) and second, the Localization mode (Refer to `LocalizationUsingiSAM2` function). In the SLAM mode, you will construct a map (using GTSAM) of the world given your AprilTag inputs by optimizing the factor graph. You will simultaneously be localizing yourself in the map. In Localization mode, you will use this constructed map and, based on observations obtained, localize yourself in the map by estimating the 6DOF pose for each AprilTag visible to you in each frame using iSAM2 part of GTSAM package.

Please refer [1] to get ideas about the factor graph needed for both SLAM and Localization modes. You are free to construct any factor graph which you think makes sense for this project phase.

2.1 SLAM Mode

In this step you will build a map based on the observation measurements the robot makes. In this mode, you will use data from `DataMapping.mat` file to build a map and localize your robot on the map (Simultaneous Localization and Mapping) using the GTSAM package. Don't use iSAM2 part of the GTSAM package here to get the best accuracy possible.

2.2 Localization Mode

In this step, you will localize your robot using the map built in the previous step. Once the map has been built, given a new sequence, your robot has to localize itself on the pre-built map using iSAM2 part of the GTSAM package.

For this mode, you will use data from `DataSquare.mat` file which will have the observations the robot makes for a square trajectory. Compute the 6DOF pose and return the values as per the function specifications provided in the next few sections. The ground truth from our iSAM2 and GTSAM output poses are given for your reference.

3 Environment

The provided data for this phase was collected with a hand-held SLAMDunk sensor module [2] (shown in Fig. 1), manufactured by Parrot®, simulating flight patterns over a floor mat of AprilTags [3] each of which has a unique ID. The data files can be downloaded from here. The data contains the rectified left camera images, IMU data, SLAMDunk's pose estimates, AprilTag [3] detections with tag size and camera intrinsics and extrinsics. All units are in m, rad, rads^{-1} and ms^{-2} if not specified.

Camera Intrinsics and Extrinsics are given specifically in `CalibParams.mat` and has the following parameters:

- `K` has the camera intrinsics (assume that the distortion coefficients in the radtan model are zero).
- `TagSize` is in size of each AprilTag in meters.
- `qIMUToC` has the quaternion to transform from IMU to Camera frame `(QuaternionW, QuaternionX, QuaternionY, QuaternionZ)`.
- `TIMUToC` has the translation to transform from IMU to Camera frame `(TransX, TransY, TransZ)`.

The `.mat` file for any of the sequence (in the format `DataNAME_OF_SEQUENCE.mat` for e.g., `DataSquare.mat` where Square is the sequence name) will contain the following data:

- `DetAll` is a cell array with AprilTag detections per frame. For e.g., frame 1 detections can be extracted as `DetAll{1}`. Each cell has multiple rows of data. Each row has the following data format:
  - `[TagID, p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y]`
    Here `p1` is the left bottom corner and points are incremented in counter-clockwise direction, i.e., the `p1x, p1y` are coordinates of the bottom left, `p2` is bottom right, `p3` is top right, and `p4` is top left corners (Refer to Fig. 2). You will use the left bottom corner (`p1`) of Tag 10 as the world frame origin with positive X being direction pointing from `p1` to `p2` in Tag 10 and positive Y being pointing from `p1` to `p4` in Tag 10 and Z axis being pointing out of the plane (upwards) from the Tag.
  - `IMU` is a cell array where each row has the following data
    `[QuaternionW, QuaternionX, QuaternionY, QuaternionZ, AccelX, AccelY, AccelZ, GyroX, GyroY, GyroZ, Timestamp]`
    You do not need the IMU readings for this project, however, if you find a creative way to use it, please feel free to use it.
  - `TLeftImgs` is the Timestamps for Left Camera Frames (Images). For e.g. Frame 1 was collected at time `TLeftImgs`(1).
  - `Pose` is an array with each row in the form
    `[PosX, PosY, PosZ, QuaternionW, QuaternionX, QuaternionY, QuaternionZ, EulerX, EulerY, EulerZ, Timestamp]`
    Here ZYX Euler angle was used which is the default for MATLAB's `quat2eul` function.
  - `PoseGTSAM` is an array of our GTSAM pose outputs from with each row in the form
    `[PosX, PosY, PosZ, QuaternionW, QuaternionX, QuaternionY, QuaternionZ]`
    Here the row number corresponds to frame number, if pose doesn't exist or is erroneous due to missed tag detections then the whole row will be zeros. You can compute valid indexes by using the following command
    `ValidIdxs = ∼any(PoseGTSAM).`
  - `PoseiSAM2` is an array of our iSAM2 pose outputs from with each row in the form
    `[PosX, PosY, PosZ, QuaternionW, QuaternionX, QuaternionY, QuaternionZ]`
    Here the row number corresponds to frame number, if pose doesn't exist or is erroneous due to missed tag detections then the whole row will be zeros.

The Images are given as a `.zip` file with the name `NAME_OF_SEQUENCEFrames.zip` where the image name is `FrameNumber.jpg` and the timestamp for each Frame Number is given in the `DataNAME_OF_SEQUENCE.mat` file's `TLeftImgs` variable, i.e., timestamp for `1.jpg` can be found as `TLeftImgs(1)`.

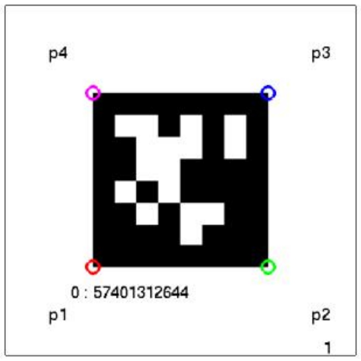

Figure 2: Corners of an AprilTag.

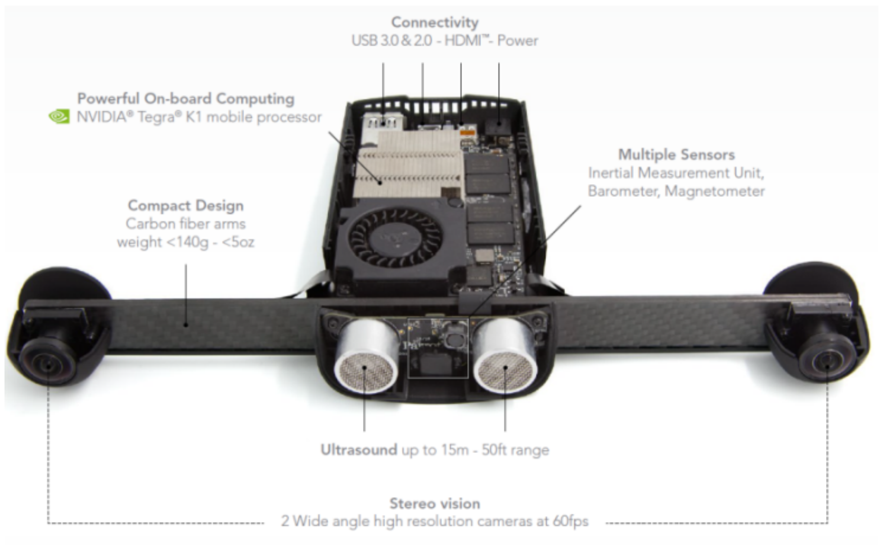
Connectivity
USB 3.0 & 2.0 - HDMI™- Power
Powerful On-board Computing
NVIDIA® Tegra® K1 mobile processor
Multiple Sensors
Inertial Measurement Unit,
Barometer, Magnetometer
Compact Design
Carbon fiber arms
weight <140g - <5oz
Ultrasound up to 15m - 50ft range
Stereo vision
2 Wide angle high resolution cameras at 60fps

Figure 1: Parrot SLAMDunk.

4 Code

The starter code can be found in the `code` folder. There are 3 `.m` files in the folder, namely, `SLAMUsingGTSAM.m`, `LocalizationUsingiSAM2.m` and `Wrapper.m`. You are required to submit the functions `SLAMUsingGTSAM` and `LocalizationUsingiSAM2` along with any other helper functions you would need. The script `Wrapper.m` is given for your debugging only. Also, the Input and Output for the `SLAMUsingGTSAM` and `LocalizationUsingiSAM2` functions are given in the Environment section. The `SLAMUsingGTSAM` function has two return arguments, `LandMarksComputed` and `AllPosesComputed`. `LandMarksComputed` is an array of landmark locations where each row is `[TagID, p1x, p1y, p2x, p2y, p3x, p3y, p4x, p4y]`. Note that the rows have to be sorted in ascending order by TagIDs. `AllPosesComputed` is an array of 6DOF pose where each row is pose at each camera timestep, i.e., you have 1 pose measurement per camera step. Each row is `[PosX, PosY, PosZ, Quaternion,`

`QuaternionX, QuaternionY, QuaternionZ]`. The reason we are using quaternions is because euler angles flip near singularity conditions - similar to the ones we are operating in. There is only a single output argument for the function `LocalizationUsingiSAM2`, namely, `AllPosesComputed` which has the same specifications as described before.
The folder `QuaternionToolbox` has the quaternion manipulation functions is included for your reference in case you do not have the aerospace toolbox.

4.1 Submission Guidelines

Note that this time you'll have to submit a 6 (upto) page report (PDF) typeset in double column format in LaTeXtalking about results and the factor graph used with any observations and analysis along with code this time.

Submit your code via CMSC 828T submit section. You should create a folder called `code` and copy `SLAMusingGTSAM.m` and `LocalizationUsingiSAM2.m` into it along with functions needed, zip it as `code.zip`, and submit `code.zip`. Any other file 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 other supplementary m-files you might have created.
- Do NOT submit any other files that are not necessary.

Your submission should contain:

- A `README.txt` detailing anything specific regarding your code.
- A `LateDays.txt` containing only one number, specifying how many late days you have used for this submission.
- A 6 (upto) page report (PDF) typeset in double column format in LaTeXtalking about results and the factor graph used with any observations and analysis. Your report should **HAVE** the following content:
  1. Factor graphs used for both GTSAM and iSAM2 parts.
  2. A plot for each sequence which includes GTSAM, iSAM2 and ground truth GTSAM and iSAM2 outputs along with the Tag locations with legends. Plot all three views XY, YZ and XZ views.
  3. Plot position and orientation error of your GTSAM and iSAM2 output as compared to our GTSAM and iSAM2 outputs respectively (compare your GTSAM output with our GTSAM output and your iSAM2 output with our iSAM2 output) using the `ComputePosError.m` and `ComputeAngError.m` for each sequence.

4. A detailed analysis of why something works better than the other or the other things you tried which did not work out.
5. Problems you faced and how you solved them.
6. Anything extra is good!

- All necessary files inside one folder `code` such that we can just run the autograder correctly.
- Folder structure for submission:

```
code
├── SLAMusingGTSAM.m
├── LocalizationUsingiSAM2.m
├── SupplementaryCodes.m
├── README.txt
├── Report.pdf
└── LateDays.txt
```

5 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] Bernd Pfrommer, Nitin Sanket, Kostas Daniilidis, and Jonas Cleveland. Penncosyvio: A challenging visual inertial odometry benchmark. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3847–3854, May 2017.

[2] Parrot. SLAMDunk. http://developer.parrot.com/docs/slamdunk/, 2016.

[3] Edwin Olson. AprilTag: A robust and flexible visual fiducial system. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3400–3407. IEEE, May 2011.