CMSC 828T
Vision, Planning and Control in Aerial Robotics
Mid Term Exam: Extended Kalman Filter
Due on 11:59:59PM on Nov 14th, 2017

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

November 3, 2017

1 Introduction

In phase 1, you computed the pose of the drone. In phase 2, you estimated the velocity. In this phase you will use Extended Kalman Filter (EKF) with additional sensor data (Inertial Measurement Unit or IMU) to get better estimates of state (more formally defined later).

2 Extended Kalman Filter

Refer to an amazing tutorial on **"Introduction to Filtering"** [2] by Prof. Vijay Kumar from University of Pennsylvania to learn about Bayesian filter, Kalman filter, and Extended Kalman filter. You will use the equations from this reference to smooth our estimates of state (more formally defined later) in Section 4. Read the full tutorial, your implementation will use the state vector defined in Section 4 of this assignment (this is similar to the Model B in Ref. [2] which can be found in Section 5.2 of Ref. [2]).

3 Data and Environment

The raw data provided will be the same as phase 2. It was collected with a hand-held SLAMDunk sensor module [3] (shown in Fig. 1), manufactured by Parrot®, simulating flight patterns over a floor mat of AprilTags [1] 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 [1] detections with tag size and camera intrinsics and extrinsics. All units are in m, rad, rads^{-1} and ms^{-2} if not specified.

3.1 Calibration

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)`.

3.2 The Data

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]`
  Here AccelX, AccelY, and AccelZ are linear accelerations, and GyroX, GyroY, and GyroZ are angular velocities, and Timestamp are the respective timestamps when they were recorded.
- `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.

- `VelGTSAM` is an array of our velocities computed from GTSAM pose outputs with each row in the form `[Vx,Vy,Vz]`
  Here the row number corresponds to frame number, if velocity 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(VelGTSAM).`

- `VeliSAM2` is an array of our velocities computed from iSAM2 pose outputs with each row in the form `[Vx,Vy,Vz]`
  Here the row number corresponds to frame number, if velocity 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(VeliSAM2).`

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).`

4 Task

You will implement EKF that takes in the drone states and outputs filtered estimates of the states. The state vector is defined below (same as Model B in Ref. [2] Section 5.2)

$$x = \begin{bmatrix} p \\ q \\ \dot{p} \\ b_g \\ b_a \end{bmatrix} \tag{1}$$

where p is the position and $\dot{p}$ is the velocity (of camera in the world frame in our case), q is a 3×1 ZXY euler angle, b_g is the gyro bias, and b_a is the accelerometer bias. Use the process model, observation model and filter equations from Ref. [2] Section 5.2.

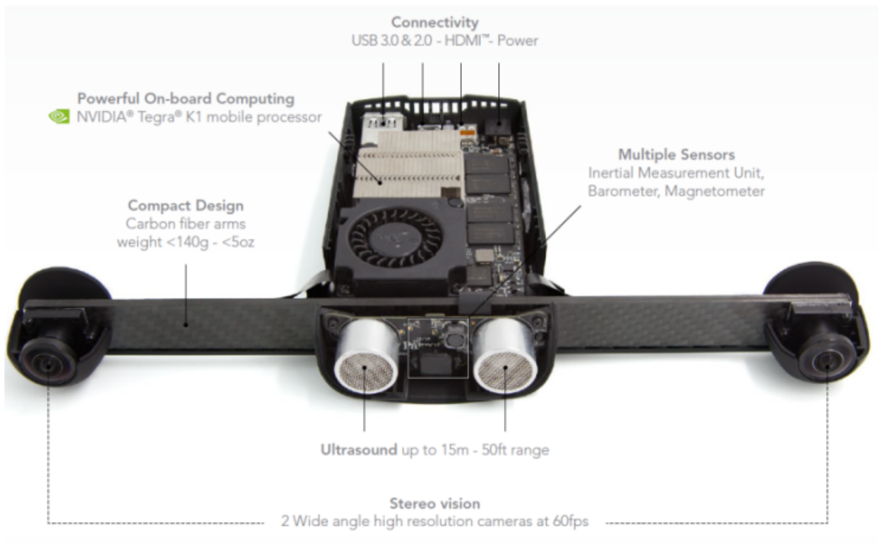
Connectivity
USB 3.0 & 2.0 - HDMI™- Power
Powerful On-board Computing
NVIDIA® Tegra® K1 mobile processor
Multiple Sensors
Inertial Measurement Unit, Barometer, Magnetometer
Compact Design
Carbon fiber arms
weight <140g - <5oz
Ultrasound up to 15m - 50ft range
Stereo vision
2 Wide angle high resolution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

Figure 1: Parrot SLAMDunk.

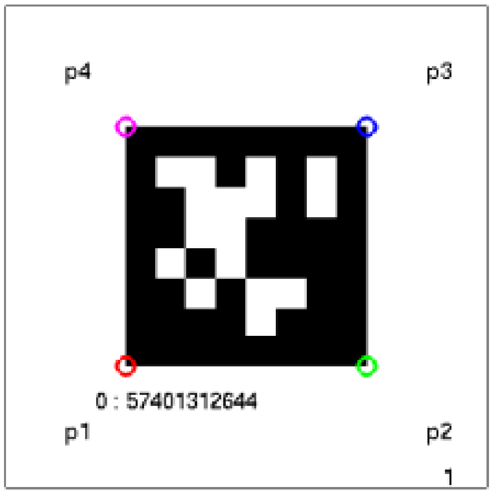

Figure 2: Corners of an AprilTag.

4.1 Align IMU timestamps

You will need to first synchronize your data with the IMU timestamps. Here you will implement a way to filter out IMU readings that are not aligned with your pose and velocity estimates (based on your image timestamp `TLeftImgs`). Please use

$$\min\left(|\mathtt{t_{image}} - \mathtt{t_{IMU}}|\right) \leq \tau$$

to find the relevant timestamps. Where $\mathtt{t_{image}}$ is `TLeftImgs` and $\mathtt{t_{IMU}}$ is the eleventh column in your `IMU` data. You will additionally need to filter out the timestamps that are not within a time threshold (τ) to filter out the readings that are not in sync. Ignore entries with many to one mapping. Ignore all frames that do not have synchronized timestamps between IMU and velocity and pose estimates (based on your image timestamp `TLeftImgs`).

4.2 Transform IMU to camera

Use the camera extrinsic with respect to IMU (obtained from `CalibParams.mat`) so that they are both in the same coordinate of reference. **Think how the transformation will be applied to and where.**

4.3 EKF Implementation

You will use pose and velocity estimates from your iSAM2 (based on your image timestamp `TLeftImgs`) to compute the new filtered pose and velocity of the drone using EKF. For this, we suggest that you use the `jacobian` function to calculate the required Jacobian matrices and then use the `simplify` function to shorten the symbolic representation. Please note that you should precompute the Jacobians and use `matlabFunction` to generate a function for them. The process update and measurement update steps may be run at different rates; It is up to you to decide what works best. Refer to Section 4 of Ref. [2].

5 Requirements

5.1 Runtime Requirement

In this phase, you will need to run your vision-based pose estimator (from phase 1 - iSAM2), optical flow based velocity estimator (from phase 2) and then the EKF in should run in realtime. So you should carefully monitor the computation time of all your algorithms (tag-based pose estimator, velocity estimator, and EKF). A 15ms runtime bound per iteration can be considered as safe for EKF.

6 Submission

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

- Your maximum upload size will be limited to 10 MB. If your file size exceeds, please upload a link to an online storage instead.
- 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 detailed and comprehensive `report.pdf` detailing your process and approach, as well as plots and diagrams for the same.
- A README.txt detailing anything we should be aware of. Since we are not providing strict guidelines on input/output specifications, please mention what we need to do to get your code running.
- **There is NO late days for this submission.**
- **There will be NO extensions.**
- All necessary files inside one folder `code` such that we can just run the automated testing script to see your results.

6.1 Project Report

Primarily, your assignment will be graded on the quality and content of your submitted `report.pdf`. We expect a PDF of at most 16 pages in double-column format, complete with figures, plots and all other necessary explanations and details regarding your process.

Your output must include, at the very least, the following points of information:

- Describe the EKF pipeline and the algorithm (use algorithm environment from the package algorithm2e in LaTeX) used along with all the equations and any assumptions made.
- The following comparison plots are expected in your report for each trajectory (`Mapping`, `StraightLine`, `SlowCircle`, `FastCircle`, `Mountain` and `Square`) with clear legends:
  - Position (`x, y, z`) plot of your iSAM2 output vs. EKF vs. GTSAM Ground Truth vs. Your GTSAM output
    - X component plot
    - Y component plot
    - Z component plot
  - Orientation (ϕ, θ, ψ) plot of your iSAM2 output vs. EKF vs. GTSAM Ground Truth vs. Your GTSAM output
    - ϕ component plot
    - θ component plot
    - ψ component plot
  - Velocity $(\dot{x}, \dot{y}, \dot{z})$ plot of your iSAM2 output vs. EKF vs. GTSAM Ground Truth vs. Your GTSAM output
    - $\dot{x}$ component plot
    - $\dot{y}$ component plot
    - $\dot{z}$ component plot

- Give a detailed analysis of why you think something works or not.
- Mention the noise covariance noise values used (R and Q matrices)

7 Acknowledgement

The project has been adapted from the University of Pennsylvania MEAM 620 course.

8 Collaboration Policy

Since this is a midterm, you are expected to complete the exam on your own. You are allowed to utilize Piazza for any clarifications. Any questions related to actually solving the problem(s) will be deleted immediately and the questioner may be penalized. This is a take home mid term exam; please treat it as such. Plagiarism is strictly prohibited. Plagiarism checker will be used to check your submission. Please make sure to **cite** any references from papers, websites you might have referred. You will be provided with one in-class tutorial by your awesome TA's and will have the opportunity to clarify your doubts regarding the exam. No further office hours will be entertained.

References

[1] AprilTags. https://april.eecs.umich.edu/software/apriltag.html.

[2] Vijay Kumar. Introduction to Filtering. https://alliance.seas.upenn.edu/~meam620/wiki/index.php?n=Main.Schedule2015?action=download&upname=2015_filtering.pdf, 2015.

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