Project 2: Orientation Tracking

Collaboration in the sense of discussion is allowed, however, the work you turn in should be your own - you should not split parts of the assignments with other students and you should certainly not copy other students' code or papers. See the collaboration and academic integrity statement here: https://natanaso.github.io/ece276a. *Books may be consulted but not copied from.*

Submission

You should submit the following two files by the deadline shown on the top right corner.

1. **FirstnameLastname_P2.pdf** on **Gradescope**: upload your solutions to the theoretical problems (Problems 1-2). You may use latex, scanned handwritten notes (write legibly!), or any other method to prepare a pdf file. Do not just write the final result. Present your work in detail explaining your approach at every step. Also, attach to the **same** pdf the report for Problem 4. You are encouraged but not required to use an IEEE conference template[1] for your report.

2. **FirstnameLastname_P2.zip** on **TritonEd**: upload all code you have written for Problem 3 (do not include the training and test datasets) and a README file with clear instructions for running it. Please try to generate an executable file from your code following the instructions online[2]. While generating an executable is **not required**, it will simplify the task of running your code with different library versions across different platforms.

Problems

In square brackets are the points assigned to each problem.

1. [4 pts] Suppose that the pose of a moving robot with respect to the world frame is given by the following function of time t:

$$T(t) = \begin{bmatrix} \cos\frac{t\pi}{3} & 0 & -\sin\frac{t\pi}{3} & t \\ 0 & 1 & 0 & 0 \\ \sin\frac{t\pi}{3} & 0 & \cos\frac{t\pi}{3} & 2t \\ 0 & 0 & 0 & 1 \end{bmatrix} \in SE(3)$$

   (a) Find the axis-angle representations of the robot orientation at time $t = 1$.

   (b) Find the quaternion representations of the robot orientation at time $t = 1$ and of the inverse of this orientation.

   (c) Compute the linear and the angular velocity with respect to the world frame at time $t = 1$.

   (d) Compute the linear and the angular velocity with respect to the robot frame at time $t = 1$.

   (e) Compute the coordinates of the point $p_W = (9, 0, 0)$ in the robot frame (i.e., find p_R in the robot frame) at time $t = 1$.

2. [4 pts] Consider the discrete-time nonlinear time-invariant system:

$$\begin{aligned} x_{t+1} &= -0.1x_t + \cos(x_t) + w_t, & w_t &\sim \mathcal{N}(0, 1) \\ z_t &= x_t^2 + v_t, & v_t &\sim \mathcal{N}(0, 0.5) \end{aligned}$$

   with a prior distribution $x_0 \sim \mathcal{N}(0, 1)$.

   (a) Derive the extended Kalman filter equations by linearizing around the nominal state $x_t^{nom} \equiv 1$

   (b) Derive the unscented Kalman filter equations around the nominal state $x_t^{nom} \equiv 1$

[1] https://www.ieee.org/conferences_events/conferences/publishing/templates.html
[2] https://natanaso.github.io/ece276a/executable.html

3. [10 pts] Implement an unscented Kalman filter to track the 3-D orientation of a rotating body using inertial measurement unit (IMU) readings. Generate a panoramic image by stitching camera images based on the orientation estimate of your filter. Instructions and tips follow.

   - **Training data**: now available:
     https://drive.google.com/open?id=0B241vEW29598UjlWOUFwaTNnRlE
     The camera sets of images are paired with their respective IMU set using the filename numbers.
   - **Test data**: released at 11:59 pm, 11/11/17:
     https://drive.google.com/open?id=0B241vEW29598Z09xeE5xUExLN2s
   - **Sensor calibration**: The biases and scale factors of the accelerometers and gyroscopes are unknown and should be approximated/learned using the ground truth data from a Vicon motion capture system. The gyros have dual output (1x and 4x). This is done by designers to make the chips versatile in different operating conditions. If you look at the schematic of the board, however, you see that only 4x gyro signals are routed to the microcontroller (therefore you should use the appropriate sensitivity). Make sure you convert sensitivity to mv/rad/sec (not degrees...). The camera axis is aligned with imu x axis. The relative position of the camera is roughly $(0, 0, 0.1)$, i.e., $10cm$ just above the imu. See Platform.jpg and IMU_reference.pdf for details.
   - **Errata in IMU_reference.pdf**:

     $$Vref/1023 * sensitivity \text{ (Incorrect)} \quad \rightarrow \quad Vref/1023/sensitivity \text{ (Correct)}$$

   - **Known glitch**: Some imu data sets (e.g., #4) contain a small glitch in the beginning, where all 3 gyro values jump to a fixed value and then come back to normal operation (typically several seconds after starting). This problem happened during data collection when a reset pin fired, locking the gyros into the nominal zero, rather than the true zero bias gyro level.
   - **Quaternions**: You are not allowed to use built-in libraries to do quaternion manipulations. You should implement your own quaternion functions: multiply, log, exp, conjugate, rotation to quaternion, quaternion to rotation, etc. Also, implement the quaternion average function discussed in class. Test it with some angles represented as quaternions, e.g., what is the average of 170, 270 and -100 degrees? You may try to use a different averaging approach if you prefer but make sure that it works as expected.
   - **Allowed functions** (incompete list): Euler2Rot, Rot2Euler, Euler2Quat, Quat2Euler, Spherical2Cartesian, Cartesian2Spherical, Cholesky. If you are not sure if a function is allowed, ask!
   - **UKF**: Follow Edgar Kraft's paper (https://natanaso.github.io/ece276a/ref/2_Kraft_UKF.pdf) to implement an unscented Kalman filter that uses the gyroscope measurements for prediction and the accelerometer measurements for update (4 dimensional state). Optionally, as Edgar Kraft suggests, you can switch the gyro measurements to observations as well and extend the state of the filter to include angular velocity (7 dimensional state). See which approach works better. Evaluate your performance by comparing it to the ground truth provided by the Vicon system. You may uses the provided plotting function rotplot.py to visualize your results. You can modify it as you see fit. Make sure your program can process new datasets containing only IMU readings without Vicon estimates.
   - **Panorama**: Construct and dispaly a panoramic image by stitching the RGB camera images over time based on your orientation estimates. Don't worry about the image looking perfect, it is very difficult to do. The goal is to make sure that you know how to use the estimated orientation in the context of fusing data coming from other sensors. If you can't get the filter to work well, you can still do the panorama part using the vicon ground truth and show that in your report.

4. [6 pts] Write a project report describing your approach to the orientation tracking and panorama reconstruction problems. Your report should include the following sections:

   - **Introduction**: discuss why the problem is important and present a brief overview of your approach
   - **Problem Formulation**: state the problem you are trying to solve in mathematical terms. This section should be short and clear and should define the quantities you are interested in.

- **Technical Approach**: describe your approach to orientation tracking and panorama reconstruction
- **Results**: present your training results, test results, and discuss them – what worked, what did not, and why. Make sure your results include (a) plots comparing your estimated roll, pitch, and yaw angles to the ground truth on the training sets, (b) plots showing your estimated roll, pitch, and yaw angles on the test sets, (c) panoramic images obtained from the training and test sets.