The Complementary Filter

MEAM 620 Project 2, Phase 1

March 28, 2024

1 Introduction

In this assignment you are being asked to implement the complementary filter for estimating the attitude of a flying quadrotor based on data from a 6 axis IMU which provides acceleration and angular rate measurements.

In order to further ground the assignment we are having you apply the algorithms you write to real data acquired by a quadrotor platform. For this we are using portions of the EuRoc dataset collected at ETH Zurich. We are providing a paper describing this dataset and the platform on which it was collected as one of the documents associated with this assignment. I would definitely reccomend taking a quick look at this paper.

2 Code Organization

The code packet that you are being provided with as part of this assignment is organized in the usual manner. In the top level directory there is a file entitled `setup.py` which you should run in order to install all of the needed packages. The `proj2_1` package is divided into 3 subdirectories.The `util` directory contains a set of tests you can run on your code via the unit test script `test.py`. The `dataset` directory contains a small portion of the EuRoc dataset containing some IMU data.

The code directory contains a set of code files and sandbox files which constitute the coding assignment. The file `complementary_filter.py` contains stubs for the implementation of the complementary filter algorithm described in class. The `sandbox_complementary_filter.py` script tests your complementary filter implementation by running it against a complete IMU dataset and plotting the result. We show you what the final plot should look like in the file `util/MachineHall01_imu0.png`. Note that running this script can take 20 seconds or more since it runs against several thousand IMU measurements.

3 Task : Complementary Filter

Your task is to provide an implementation for the complementary_filter_update function found in the file `complementary_filter.py`.

This function takes as input an initial rotation estimate expressed as a scipy.spatial.transform.Rotation object. It also takes two 3×1 vectors which denote the angular velocity during the interval, the measured acceleration at the end of the interval and the duration of the interval. The units are specified in the comments. The output from this function should be the new estimate for the rotation at the end of the interval again returned as a rotation object. You should use the implementation provided in the course lecture slides as inspiration.

A few things to be aware of, firstly the Rotation class that we want you to use expresses quaternions in a different format than the one used in the lecture slides, namely (x, y, z, w) as opposed to (w, x, y, z). Another important difference is that for the dataset we are testing on the x axis of the IMU is roughly

aligned with the gravity direction, not the z axis. So we desire $R_{1k}g$ to be e_x not e_z which means that you will need to use a slightly different quaternion construction to come up with the rotation correction q_{acc}. Lastly the units of the acceleration measurements are expressed in meters per second squared, not in units of g's so you must remember to normalize correctly.

4 Slide Elements

We would like you to include a single slide with your final submission that contains the following elements:

1. Please include the plot that the sandbox code produced when you ran it. Not the image that we gave you.
2. Please include a derivation showing how you modified the equations presented in the lecture slides to account for the difference in the gravity direction, e_x vs e_z.
3. Provide a derivation showing that the if (u_0, u) is a unit quaternion which can be written as $(\cos\frac{\theta}{2}, \sin\frac{\theta}{2}\omega)$ where θ is an angle in radians and $\omega \in \mathbb{R}^3$ is a unit vector indicating the axis of rotation. Then

   $$H(u_0, u) = (u_0^2 - u^T u)I + 2u_0\hat{u} + 2uu^T$$

   is actually equivalent to the Rodrigues Formula.
4. Any other details we should be aware of (collaborations, references, etc.) if any.

5 Submission

When you are finished, submit your code via Gradescope which can be accessed via the course Canvas page. Your submission should contain:

1. The `complementary_filter.py` .
2. The one page slide.

Shortly after submitting you should receive an e-mail from `no-reply@gradescope.com` stating that your submission has been received. Once the automated tests finish running the results will be available on the Gradescope submission page. There is no limit on the number of times you can submit your code.

Please do not attempt to determine what the automated tests are or otherwise try to game the automated test system. Any attempt to do so will be considered cheating, resulting in a 0 on this assignment and possible disciplinary action.