Assignment 2 - State Estimation

Dr. Joseph Conroy (conroyj@umd.edu) 10 points
Due: 26 March 2024

Overview

This assignment is intended to support your understanding of the methods of state estimation, high rate data logging, and coordinate systems used in the drone. Everyone will turn in their own assignment.

1 Understanding the PX4 State Estimation Scheme

The px4 flight stack comes with a state estimation scheme that is, anecdotally speaking, quite robust and accurate considering how brittle state estimation can often be.

- The actual code for state estimation can loosely be considered to be here within the firmware: https://gitlab.com/voxl-public/flight-core-px4/px4-firmware-ci/-/tree/master/src/modules/ekf2/EKF - check this out and simply peruse the codebase for a bit to try to get a handle on the complexity. Read the documents in the “documentation” folder here. No further actions requested.
- Here is a repository with the estimation and control library integrated within the PX4 flight stack: https://github.com/PX4/PX4-ECL - this has been largely developed and maintained by the same person developing the EKF2 module in the flight stack, Dr. Paul Riseborough.
- Overview of the PX4 ECL-EKF estimator: https://docs.px4.io/main/en/advanced_config/tuning_the_ecl_ekf.html - review this
- PX4 Estimation Update 2021 Video: https://www.youtube.com/watch?v=CwehBx2hJcg - watch this
- Significant portions of the state estimation scheme are based on this reference: https://arxiv.org/pdf/1711.02508.pdf - skim and pay close attention to the quaternion basics if you are unfamiliar.

Problem 1 - Having watched the PX4 Estimation Update 2021 youtube video, what are the benefits of the discussed output prediction scheme integrated within the overall state estimation architecture?

2 ULOG/.ulg format flight logging

ULOG format: Sometimes, it’s helpful to have very verbose output of exactly what’s going on within the PX4 flight control system. Separate from ROS1/ROS2, and not limited only to ModalAI hardware, there is a method of logging mavlink messages generated as μORB topics in the flight stack. As an aside, the Ardupilot flight stack has a comparable feature. On older autopilots, such as the Pixhawk Cube, a 4,8, or 16 GB SD card is placed directly in the autopilot to capture such logs. Generally, a log is created every time you arm the system, for both real and simulated setups. Now, with our setups, for simulated vehicles the default log location is:
PX4-Autopilot/build/px4_sitl_default/rootfs/log/<date>/<time>.ulg,
and here for the real modalai vehicles:
/data/px4/log/<date>/<time>.ulg

Read here for an overview: https://docs.px4.io/main/en/dev_log/ulog_file_format.html. There is a method of displaying this collected data as well, known as “Flight Review”: https://github.com/modalai/px4-flight-review. You can break ULOG files out into individual .csv files using pyulog (’pip install pyulog’ on your host machine). Also, here’s the source code: https://github.com/PX4/pyulog.

Problem 2 - Using pyulog and the plotting method of your choice, breakout a .ulog file from either a real or simulated flight (>=30 seconds and <=2 minutes flight time with noticeable motion in each axis is fine) and plot the Euler angles, velocity, and position of the vehicle.

3 VOXL2 Vehicle Parameters

Perform a 'git clone https://gitlab.com/voxl-public/voxl-sdk/utilities/voxl-px4-params.git' somewhere on your laptop, location doesn't matter but know we'll be accessing these parameters later in the semester. Go into voxl-px4-params/params/v1.14/ (we are using version 1.14 of the PX4 flight stack on the drones). Know that from a clean install, I have loaded only three of these .param files: 1.) voxl2_io_helpers/voxl2_io_enable_pwm.params (because we're using the VOXL2 IO board), 2.) platforms/M500_FCV2.params (because the base frame design is an older m500 vehicle), and 3.) EKF2_helpers/indoor_vio.params (because we are flying indoors right now).

Consider also /EKF2_helpers/outdoor_gps.params as well. Lookup and compare all parameters that exist in both the outdoor_gps.params file and indoor_vio.params files.

Problem 3 - Discuss what each parameter setting in both the outdoor_gps.params file and indoor_vio.params file is supposed to do.

4 Coordinate System

Problem 4 - Assuming either ROS1 MAVROS or ROS2 MicroXRCE-DDS framework (please do indicate which), make a diagram (2D "top down" view is fine here) of the drone with world (inertial) coordinates shown, body coordinates shown, and showing the orientation of the drone relative to world coordinates when it starts on the ground (either when the simulation starts, or when you plug in the real drone). Relate this to the heading angle; is the heading angle zero when initialized? When using the real vehicle with indoor parameters, do the "indoor" world coordinates "lock in" from vehicle boot up or when you arm the vehicle? Recommend you remove the props for trying this since you don't need to liftoff to check. For ROS1, you'll probably check the /mavros/local_position_pose topic and for ROS2, you'll need either the /fmu/out/vehicle_local_position or /fmu/out/vehicle_attitude topics.

Problem 5 - Assuming ROS2 (the MicroXRCE-DDS agent setup), what heading (in rad or degrees) corresponds to the quaternion, q={-0.3827, 0, 0, 0.9239}?