Lab 4 Navigation with IMU and Magnetometer

Deadline: As posted to Piazza

Lab learning objectives and tasks:

In this lab, you are going to build a navigation stack using two different sensors – GPS & IMU, understand their relative strengths + drawbacks, and get an introduction to sensor fusion.

Please get an early start on the lab. **You cannot complete this lab in a day!**

Collaboration:

- Data acquisition can be done collectively as a team. One dataset per team is sufficient.
- Plan data collection schedules amongst your teammates. You can talk to your teammates and ask questions on Piazza.

Hardware / Sensors

- GNSS puck - USB based, issued one per team. (use the one issued for Lab 1)
- VectorNav VN-100 IMU - USB based, issued one per team. The user manual for the sensor has already been provided in Piazza readings.

Software

- GPS Driver from LAB 1. It is vital that you use the driver of a person who had written the driver correctly (correct calculations, correct header & timestamp with secs & nsecs from GPS and not system time). Please verify your driver performs the correct calculations/conversions with the LAB1_evaluator.zip
- IMU Driver from LAB 3. Although there were not many calculations to perform in LAB 3 except quaternion conversions, ensure that this driver is also correct. For this to happen, the team should ensure the driver does the following things
    - The header frame ID is correct
    - The timestamp, both seconds & nanoseconds, **should not be** zero (MOST VITAL)
    - The VNYMR string is parsed correctly with variables correctly assigned
    - In LAB 3, the conversion from Euler to Quaternion & then back to Euler from Quaternion gave the exact same Euler angles that the sensor originally provided (No loss of precision with the correct convention for roll, pitch & yaw as in the sensor manual)

Part A: Collect data in a car for dead-reckoning (as a team)

Logistics

1. If any teammate has access to a car, it might be more convenient for data collection so that you can do it on your team's schedule.

2. Alternatively, a team can schedule to use our autonomous car 'NUANCE' for data collection. Please see the spreadsheet for the NUANCE availability.

Setup & Data Collection

Before embarking, test everything by collecting data from both the GPS and IMU sensors. Make sure you can see and record both gps msgs and imu msgs on your machine. Decide on roughly a 10 minute driving loop that starts and ends at Ruggles station. Be ready to share this route with your driver (see "mini Boston tour" for details on a good route).

1. To collect data from both the GPS and IMU sensors, create a single launch file that launches both your GPS & IMU driver nodes, and make sure you are able to see both *gps* and *vectornav* topics on your machine. The two topics should exist & be publishing data together successfully.

- You can verify data is being published by opening multiple terminal windows and doing rostopic echo gps and rostopic echo imu in separate windows. To ease things, we suggest installing terminator by

sudo apt-get install terminator

It will allow you to have multiple terminal screens in one window instead of switching tabs or windows.

2. Before data collection (keep the car turned off until specified)

- Mount the IMU inside your vehicle with electrical tape on the center of the car's dashboard. Make sure that the x–axis is pointed forward and that the IMU is horizontal (verify with a leveling app on your phone).

- Fix the GPS puck to the roof – it has a magnetic back and should just stay there.

- Now connect both sensors to one single laptop and launch the two nodes. These two nodes should never die during this data collection (more explained in 3), so do not turn them off. If a USB unplugs or if one of them crashes, you must collect your entire dataset again. We suggest one member monitors the screen & ensures topics are being published.

3. There will be two datasets in this lab

- The Car Donuts (sort of)
  - Begin a rosbag and call it *data_going_in_circles.bag*
  - Wait 10 - 15 seconds

- Start your car
- Drive the car in circles 4-5 times (the more circular the path, the better). One suggested place is the Ruggles circle near Centennial common.
- Stop recording only the rosbag. **Do not** turn off the car or the driver nodes. If NUANCE's engine stops (it's a hybrid), that's OK.

- Mini Boston Tour (again… sort of)
  - Begin a rosbag and call it *data_driving.bag*.
  - You might want to make a video of your route as you drive, it may help understand any data anomalies.
  - Wait 10 - 15 seconds.
  - Go for a drive around Boston **AND** return to the spot where you started. We suggest a total distance of **AT LEAST** 2 – 3 kilometers with a **MINIMUM** of 10 turns.
  - Do not go underground/in tunnels (e.g., eastbound on Columbus to go underneath Mass Ave). Enjoy the view (whether outdoors or on the linux screen).
  - Once you return to the spot you started from, stop the rosbag recording
  - You can now turn off your car & your ROS drivers.

Note: Please follow all local & state traffic laws. Always yield to pedestrians. And remember that not everyone likes robots.

Part B: Analysis of the data collected in Part A (Individually)

1. **Estimate the heading (yaw)**

Magnetometer Calibration:

- Correct magnetometer readings for "hard-iron" and "soft-iron" effects using the data collected when going around in circles. **Do not** use *magcal* (a built-in function of MATLAB). Write the code yourself. The *magcal* function is not optimized for high-grade sensors.

- *Submit a plot showing the magnetometer data before and after the correction in your report.*

Sensor Fusion:

All further analysis will now be performed with data from data_driving.bag and the previously obtained calibration will be used for the magnetometer.

- Calculate the yaw angle from the magnetometer calibration & plot the raw magnetometer yaw with the corrected yaw for comparison. Going forward, only use the corrected magnetometer yaw.

- Integrate the yaw rate/gyro sensor to get yaw angle. One way to do this is to use '*trapz'* or '*cumtrapz'* commands in MATLAB and treat your time series as a function that is being integrated. You are free to use your own function if the results are better.

- Compare the yaw angle from above two methods. (Magnetometer vs. Yaw Integrated from Gyro).

- Use a complementary filter to combine the yaw measurements from the magnetometer and yaw rate/gyro as described in class to get an improved estimate of the yaw angle (filter the magnetometer estimate using a low pass filter and gyro estimate using a high pass filter). You might also find the '*unwrap'* or '*wraptopi'* commands in MATLAB useful.

- Plot the results of the low pass filter, high pass filter & complementary filter together.

- *Compare your sensor fusion yaw result with the yaw angle computed by the IMU and write down your observations.*
  Note : you **cannot** use the original yaw from the IMU in any analysis except this last part.

Plots to submit:

- The magnetometer X-Y plot before and after hard and soft iron calibration
- The time series magnetometer data before and after the correction in your report.
- LPF, HPF, and CF plots together
- Compare the yaw angle between four methods: Magnetometer, Yaw Integrated from Gyro, Complementary filter, Yaw angle computed by the IMU

2. Estimate the forward velocity

1. Integrate the forward acceleration to estimate the forward velocity. Your initial values for velocity may be too low or high. You may need to make some adjustments on these data (e.g., is there velocity in another axis? Do the values make sense given your knowledge of what the car was doing?)

2. Calculate an estimate of the velocity from your GPS measurements.

Plots to submit:

- Velocity estimate from the GPS
- Velocity estimate from accelerometer before and after adjustment

3. Dead Reckoning with IMU

- Integrate the forward velocity to obtain displacement and compare with GPS displacement.

- We simplify the description of the motion by assuming that the vehicle is moving in a two-dimensional plane.

  Denote the position of the center-of-mass (CM) of the vehicle by (X,Y,0) and its rotation rate about the CM by (0,0,ω). We denote the position of the inertial sensor in space by (x,y,0) and its position in the vehicle frame by (x_c,0,0).

  Then the acceleration measured by the inertial sensor (i.e. its acceleration as sensed in the vehicle frame) is

$$\ddot{x}_{obs} = \ddot{X} - \omega\dot{Y} - \omega^2 x_c$$
$$\ddot{y}_{obs} = \ddot{Y} + \omega\dot{X} + \dot{\omega}x_c$$

where all of the quantities in these equations are evaluated in the vehicle frame. Assume that $\dot{Y} = 0$ (that is, the vehicle is not skidding sideways) and ignore the offset by setting $x_c = 0$ (meaning that the IMU is on the center of mass of the vehicle, i.e. the point about which the car rotates). Then the first equation above reduces to $\ddot{X} = \ddot{x}_{obs}$.

Integrate this to obtain $\dot{X}$. Compute $\omega\dot{X}$ and compare it to $\ddot{y}_{obs}$. How well do they agree? If there is a difference, what is it due to? Can you fix it? Plot all results.

- With the previous assumptions, use the heading from the magnetometer (or your complementary filter if you have better results) to rotate your previously estimated forward velocity.

  Denote this vector by (v_e,v_n) , where 'e' means Easting & 'n' means Northing.

  Integrate (v_e,v_n) to estimate the trajectory of the vehicle (x_e,x_n).

  Compare this estimated trajectory with the GPS track by plotting them on the same plot. Make sure to adjust starting point, so that both the tracks start at the same point and same heading (adjust heading so that the first straight line from both are oriented in the same direction).
  Report any scaling factor used for comparing the tracks.

- **(Bonus up to 100% credit for this lab)** Estimate x_c
  NOTES/HINTS: Denote the position and velocity of the center-of-mass (CM) of the vehicle by **R** and **V**, respectively. The inertial sensor is displaced from the CM by **r** = (xc,0,0) - note that this vector is constant in the vehicle frame & assumes that the displacement of the IMU sensor is only along the x-axis. Let **ω** = (0,0,ω) denote the rotation rate of the vehicle about the CM. The velocity of the inertial sensor is **v** = **V** + **ω** x **r**, and its corresponding acceleration is:

  $$\ddot{x} = \dot{v} + \omega \times v = \ddot{X} + \dot{\omega} \times r + \omega \times \dot{X} + \omega \times (\omega \times r)$$

  Taking the x- and y-components of this equation in the vehicle frame gives the two equations quoted above.

Estimate the value of x_c . Show your working/approach used to find the value (no code snippets).

Questions to answer in your report

- How did you calibrate the magnetometer from the data you collected? What were the sources of distortion present, and how do you know?
- How did you use a complementary filter to develop a combined estimate of yaw? What components of the filter were present, and what cutoff frequency(ies) did you use?
- Which estimate or estimates for yaw would you trust for navigation? Why?
- What adjustments did you make to the forward velocity estimate, and why?
- What discrepancies are present in the velocity estimate between accel and GPS. Why?
- Compute $\omega\dot{X}$ and compare it to $\ddot{y}_{obs}$. How well do they agree? If there is a difference, what is it due to?
- Estimate the trajectory of the vehicle (x_e,x_n) from inertial data and compare with GPS. (adjust heading so that the first straight line from both are oriented in the same direction). Report any scaling factor used for comparing the tracks.
- Estimate x_c and explain your calculations (bonus up to 100%)
- Given the specifications of the VectorNav, how long would you expect that it is able to navigate without a position fix? For what period of time did your GPS and IMU estimates of position match closely? (within 2 m) Did the stated performance for dead reckoning match actual measurements? Why or why not?

Grading (will be re-scaled to 100 when posted to Canvas)

Plot of magnetometer calibration	1
Plot of magnetometer yaw estimate before and after calibration	1
Plot of yaw estimate from VectorNav	1
Plot of complementary filter estimate (these should all be on one plot, but each point is for each analysis type)	1
Plot of fwd velocity from accel (before and after adjustment)	2
Plot of fwd velocity from gps (again plot together, but points are separate)	1
Questions 0-8	9
Professional engineering norms in writing (style guide)	2

How to Submit Lab_4

1. In your class repo ‘EECE5554’, create a directory called LAB4
2. Copy the ROS driver package used for this assignment under LAB4.
3. Inside LAB4, create sub-directory ‘analysis’
4. Copy the MATLAB/python code used for data analysis and dead reckoning under ‘analysis’
5. Place your report in pdf format also in analysis directory.

Your repo structure should look similar to

‘<Path_to_repo>/EECE5554 /LAB2/src/<your_driver_files>’
‘<Path_to_repo>/EECE5554 /LAB2/src/analysis/<your analysis files>’
‘<Path_to_repo>/EECE5554 /LAB2/src/analysis/report.pdf’

6. Push your local commits to (remote) gitlab server. You can verify this by visiting gitlab.com and making sure you can see the commit there.