Lab 2

Deadline: 11:59 PM on Monday, Feb 10, 2020

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. user_manual_link

Data collection Policy

Plan data collection schedules amongst your teammates. You can talk to your teammates and ask questions on Piazza.

- (Part 1) Everyone in the team needs to write their **own** device driver for IMU. Make sure everyone gets a fair amount of sensor time to test imu driver, and analyse sensor noise.

- (Part 2) Data acquisition can be done collectively in a team. One dataset per team is sufficient.

- (part 3) Analysis on the collected dataset, should be done **individually**.

Setup

1. Imu udev rules: Udev rule to make the VN-100 plug & play and to address Ubuntu USB latency issues.
Connect your imu and do $ dmesg in the terminal, you will see your sensor attributes like product id and vendor id. If they differ from the values below, change them in your 50-VN-100.rules file accordingly.

1. As Sudo create a file under /etc/udev/rules.d called 50-VN-100.rules with the following

```
KERNEL=="ttyUSB[0-9]*", ACTION=="add", ATTRS{idVendor}=="1d6b", ATTRS{idProduct}=="0002",
                                        MODE="0666", GROUP="dialout"
```

2. Ubuntu's default latency of 16ms cause issues with high rate sensors, this can be solved by adding the following UDEV rules as 49-USB-LATENCY.rules

```
ACTION=="add", SUBSYSTEM=="usb-serial", DRIVER=="ftdi_sio", ATTR{latency_timer}="1"
```

3. Once the rules have been added, to get udev to recognize the rule, run the following command:

```
sudo udevadm control --reload-rules && sudo service udev restart && sudo udevadm trigger
```

Finally unplug and replug the VN-100 to have it work with the new rules.

You can verify the applied settings with below command, should report 1 on sucess.
$ cat /sys/bus/usb-serial/devices/<ttyUSB0 your device path>/latency_timer

What to Submit for Lab 2 ?

Part 1: (Do it Individually) Write a device driver for IMU

Open serial port with 115200 baudrate

Parse $vnymr string, to get accel, gyro, orientation(roll,pitch,yaw) and magnetometer data. Refer to sensor user_manual.

Use ROS sensor_msgs/IMU to publish the above data. Convert the above Yaw, Pitch, Roll data into quaternions and publish it as orientation in the same imu msg. Use ROS sensor_msgs/MagneticField to publish magnetometer data.

Begin collecting a time series data(rosbag)of the 3 accelerometers, 3 angular rate gyros, 3- axis magnetometers. Collect at least 1000 points of data with the *instrument stationary and as far away as possible from the computer.*

Plot each time series in your report and figure out the noise characteristics of each of the values reported by the IMU.

Push your IMU device driver code to gitlab, refer to How to Submit Section.

Part 2: (Do in a team) Collect Data in a car for dead-reckoning

Setup

Collect Data from both sensors. Make sure you are able to see both gps msgs and imu msgs on your machine.

Mount the IMU in your vehicle with electrical tape. Make sure that the x –axis is pointed forward and that the imu is as close to horizontal as possible. Fix the GPS puck to the roof – it has a magnetic back and should just stay there. Connect both sensors to your laptop and begin logging. Wait 10 seconds and start the vehicle. Drive course including 360 turns for compass calibration. After finishing the mission, shut down logging.

Ideally in the beginning of your ride, go around in a circle 3 -4 times for compass calibration. One suggested place to do would be roundabout in front of Ruggles station and Ryder hall. Then go for a ride and come back to the starting point. (like a loop).

Logistics

(Recommended) If you / one of your teammates has access to a car, it might be more convenient for data collection. So you can do it at your (teammates) schedule.

Alternatively, a team can schedule to use our autonomous car ‘NUANCE’ for data collection. The availability of the NUANCE car and NEU driver requirements will be updated in a separate document on piazza.

Part 3: (Do it Individually) Analysis on the data collected in Part 2

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.

Submit plots showing the magnetometer data before and after the correction in your report.

Calculate the yaw angle from the corrected magnetometer readings.

Integrate the yaw rate 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.

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

Use a complementary filter to combine the measurements from the magnetometer and yaw rate as described in class to get an improved estimate of 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.

Compare your result to the yaw angle computed by the IMU and write down your observations.

2. **Estimate the forward velocity**

Integrate the forward acceleration to estimate the forward velocity.

Additionally, calculate an estimate of the velocity from your GPS measurements.

Plot both the velocity estimates.

Make observations on the plot. Does the integrated velocity estimate make sense?

Make adjustments to the acceleration measurements to make the velocity plot more reasonable.

Provide the rationale you use for the adjustments and plot the adjusted velocity.

3. **Dead Reckoning with IMU**

Integrate IMU data to obtain displacement and compare with GPS.

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.

1. Assume that $\dot{Y} = 0$ (that is, the vehicle is not skidding sideways) and ignore the offset by setting $x_c = 0$. 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?*

2. Use the heading from the magnetometer to rotate into a fixed (East, North) reference frame.

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

Denote this vector by (v_e,v_n).

Integrate it to estimate the trajectory of the vehicle (x_e,x_n). Compare the estimated trajectory with the GPS track by plotting them on the same plot.

Make sure to adjust the 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.

3. Estimate x_c

NOTE: 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. 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:

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

where we have denoted **v** by $\dot{x}$, and **V** by $\dot{X}$. Taking the x- and y-components of this equation in the vehicle frame gives the two equations quoted above.

Grading Rubric (10 Points)

- 2 points for working device driver (part1)
- 1 point for sensor data collection (part2)
- 4 points for part 3
- 3 Points for data analysis and report.

Late submission policy is mentioned in the syllabus.

How to Submit Lab_2

1. In your class repo 'EECE5554_RoboticsSensing', create a directory called LAB2
2. Copy the ros driver package used for this assignment under LAB2.
3. Inside LAB2, 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. (also upload the report in pdf format on blackboard).

Your repo structure should look similar to

'<Path_to_repo>/EECE5554_RoboticsSensing/LAB2/<your_imu_ros_driver files>'
'<Path_to_repo>/EECE5554_RoboticsSensing/LAB2/analysis/<your analysis files>'
'<Path_to_repo>/EECE5554_RoboticsSensing/LAB2/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.
7. Upload your report in pdf format to Blackboard Assignments under Lab_2