AUV Localization Using Dead Reckoning Techniques with IMU Sensor

EEL5934 Aerial & Marine Robotics – Course Project

Authors:
Brandon Davis & Ryan Gochar

Overview

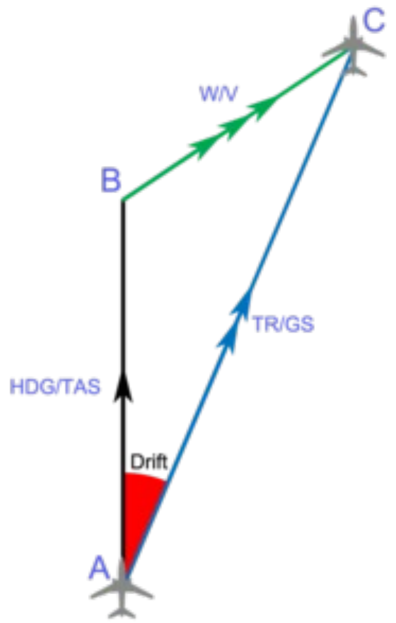
- The purpose of this project is to implement and evaluate a dead reckoning localization algorithm using an inertial measurement unit (IMU) sensor on an autonomous vehicle. The implementation will be applicable for an autonomous underwater vehicle (AUV), but for time constraints of this project will be implemented using a surface vessel.

Background

- Most robust and widely used localization system is Global Navigation Satellite System (GNSS).
- Certain environments deny the use of GNSS: underground, indoors, underwater.
- The need for autonomous vehicles to operate in these GNSS denied environments produce the need for alternative localization methods.

Dead Reckoning Overview

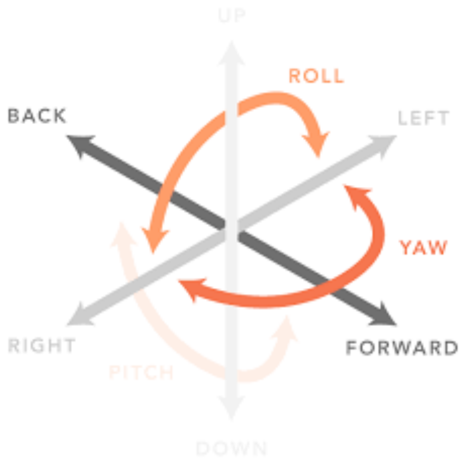
- Method of determining current position based on previous positions and estimates of speed, heading, and elapsed time.

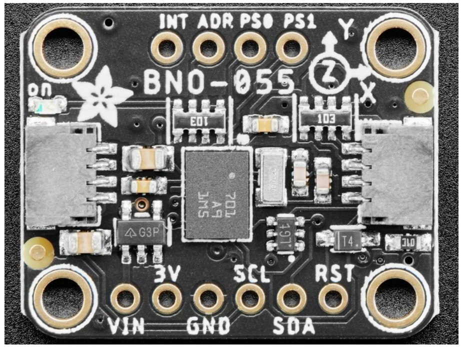


Source: https://en.wikipedia.org/wiki/Dead_reckoning

Internal Measurement Unit (IMU)

- For this project, the Bosch BNO055 9 degrees-of-freedom (DOF) MEMS IMU will be used. This IMU is a combination of these sensors:
  - 3-axis Accelerometer – measures linear velocity along X,Y,Z axis.
  - 3-axis Gyroscope – measures angular velocity around X,Y,Z axis.
  - 3-axis Magnetometer – measures magnetic field along X,Y,Z axis.



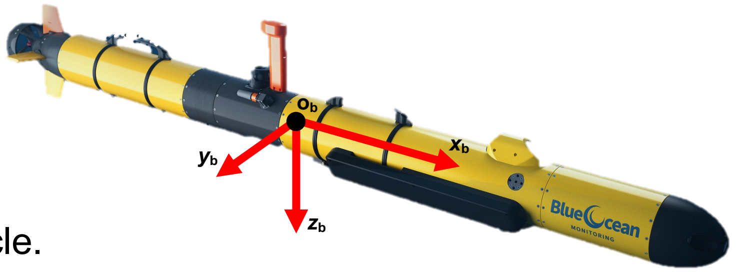


MEMS – Microelectromechanical

Objectives

- Implement a localization scheme using only measured data from an IMU and measure performance compared to ground truth data (absolute position).
- Verify performance of dead reckoning localization by post processing recorded IMU data using a surface vessel as collection platform.
- Verify performance of dead reckoning localization in real-time on surface vessel.

Methodology



- Consider these reference frames:
  - Body frame – origin at CO {o_b} of the vehicle.
  - IMU frame – origin of IMU measurement frame.
  - GNSS frame – origin of GNSS sensor.
  - NED - *North-East-Down:*
    - x_n points toward true North.
    - y_n points towards East.
    - z_n points downward normal to Earth's surface.

- Assumptions:
  - For simulated data, we assume for a *strapdown system* that the IMU and GPS frame are aligned with the BODY frame and share an origin. Therefore, all IMU measurements are in the BODY frame.

Methodology (cont.)

- Prior to using the IMU data for any calculations, the static IMU data needs to be evaluated and filtered for noise[1,2].
    - Filtering of IMU data is commonly done using a Kalman filter[1,2]. For this project a Kalman filter will be used. Since the scope of this project is not creating this filter from scratch, an open-source library implementation of the filter will be used. Tuning of the filter will be required and differ from sensor to sensor based on quality of data.
    - Some open-source libraries to be used:
        - https://github.com/CCNYRoboticsLab/imu_tools/tree/noetic
- In addition to filtering, any static transforms from the IMU coordinate frame convention to the body frame convention needs to be calculated.

Methodology (cont.)

- From the filtered IMU data, the *attitude vector* $\Theta_b = [\phi, \theta, \psi]^T$ is calculated[2] where ϕ is the roll, θ is the pitch, and ψ is the yaw.
- The roll and pitch angles are needed to project the 3-axis magnetometer data onto the horizontal plane using known *rotation matrix*[2].
- Then the magnetic compass heading can be calculated using the horizontal components.
- The velocity in 3-axis is calculated by multiplying acceleration measurements by the elapsed time from previous measurement and adding to a previously known velocity value[1].
- The horizontal components of the velocity are used to calculate a new position estimate. The velocity is multiplied by elapsed time and added to the previous position. Doing this for both X and Y axis results in an updated X-Y position estimate.

Methodology – Kalman Filter

- How it works?
- Efficient which makes it good for processing real time data
- Inputs: Acceleration data, Magnetometer data (transformations)
- Implementation: pykalman KalmanFilter
- Define state matrices –
  - F = Transition matrices
  - H = Observation matrices
  - Q = Transition covariance
  - R = Observation covariance
  - X0 = Initial state mean
  - P0 = Initial state covariance
- Output: Filtered Acceleration Data

Methodology – Open-loop IMU Solution

- Direct Filter –
  - Accelerometer and angular rate measurements are *inputs* to *strapdown navigation equations*.
  - Position and velocity are states in the estimator.
- Double integration of Acceleration
  - Velocity
  - Position
- Rotation transformations
- Output: Plot positions against true GPS position

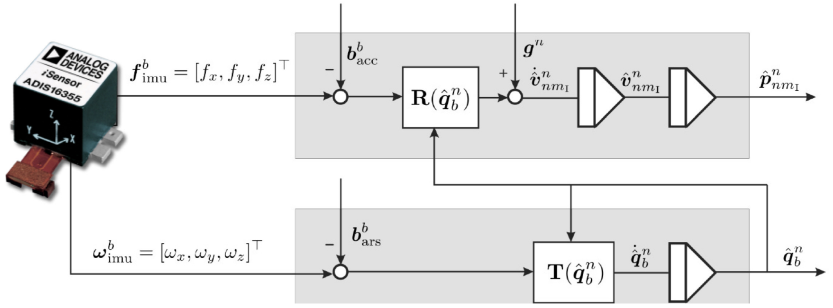

ANALOG DEVICES
iSensor
ADIS16355
$\boldsymbol{f}^b_{\mathrm{imu}} = [f_x, f_y, f_z]^\top$
$\boldsymbol{b}^b_{\mathrm{acc}}$
$-$
$\boldsymbol{g}^n$
$+$
$\mathbf{R}(\hat{\boldsymbol{q}}^n_b)$
$\dot{\hat{\boldsymbol{v}}}^n_{nm_\mathrm{I}}$
$\hat{\boldsymbol{v}}^n_{nm_\mathrm{I}}$
$\hat{\boldsymbol{p}}^n_{nm_\mathrm{I}}$
$\boldsymbol{\omega}^b_{\mathrm{imu}} = [\omega_x, \omega_y, \omega_z]^\top$
$\boldsymbol{b}^b_{\mathrm{ars}}$
$-$
$\mathbf{T}(\hat{\boldsymbol{q}}^n_b)$
$\dot{\hat{\boldsymbol{q}}}^n_b$
$\hat{\boldsymbol{q}}^n_b$

Figure XX. Principle integration of IMU data.[1]

$$\mathbf{p}[\mathbf{k}] = \mathbf{p}[\mathbf{k}-\mathbf{1}] + \mathbf{v} * dt$$

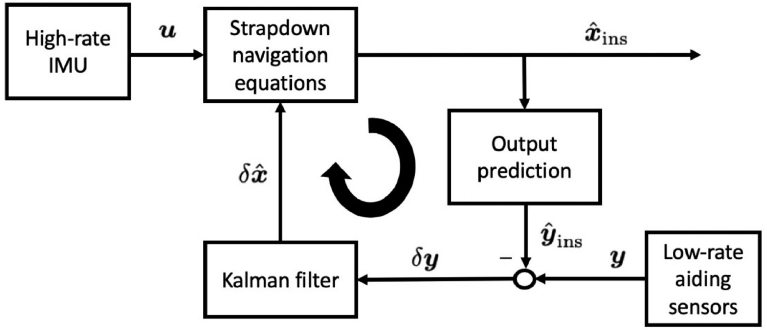
$$\mathbf{v}[\mathbf{k}] = \mathbf{v}[\mathbf{k}-\mathbf{1}] + \mathbf{a} * dt$$

Methodology – Indirect Feedback Kalman Filter

- Indirect Filter –
  - Accelerometer and angular rate measurements are *inputs* to *strapdown navigation equations*.
  - Error estimates are used to update the INS estimates directly to avoid drift.[1]

True state: $\chi[k] = \chi_{\mathbf{ins}}[k] + \delta\chi[k]$

Error state: $\delta\chi[k]$


High-rate IMU
$\boldsymbol{u}$
Strapdown navigation equations
$\hat{\boldsymbol{x}}_{\text{ins}}$
Output prediction
$\delta\hat{\boldsymbol{x}}$
$\hat{\boldsymbol{y}}_{\text{ins}}$
$-$
$\delta\boldsymbol{y}$
$\boldsymbol{y}$
Kalman filter
Low-rate aiding sensors

Feedback filter (reset)

Figure XX. Indirect feedback filter for INS.[1]

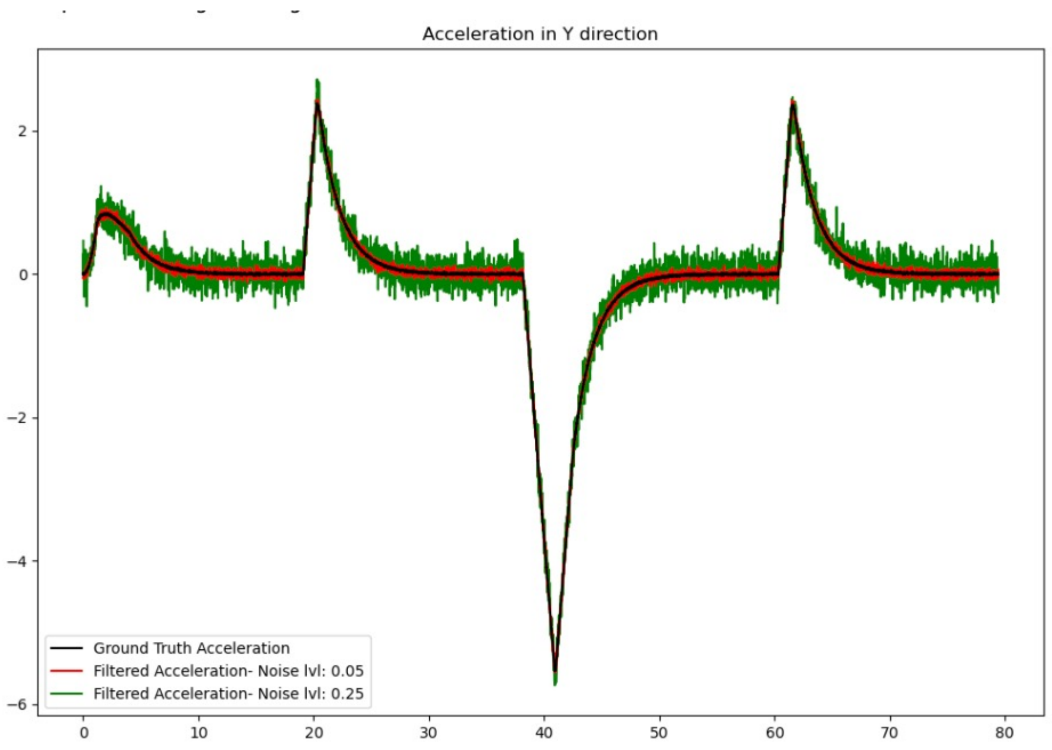
Methodology (cont.)

- For this project, since we are using a surface vessel we only care about measurements in the X-Y plane. We only care about a pose estimate in the X-Y plane.
- In the application of an underwater or aerial vehicle, the pose estimate in three dimensions is important. This often calls for fusing additional sensor data such as depth from a depth sensor or altitude from a barometer.
- The ability to calculate pose estimate in 3-axis is planned to be implemented in software but will be difficult to test without an underwater or aerial platform.

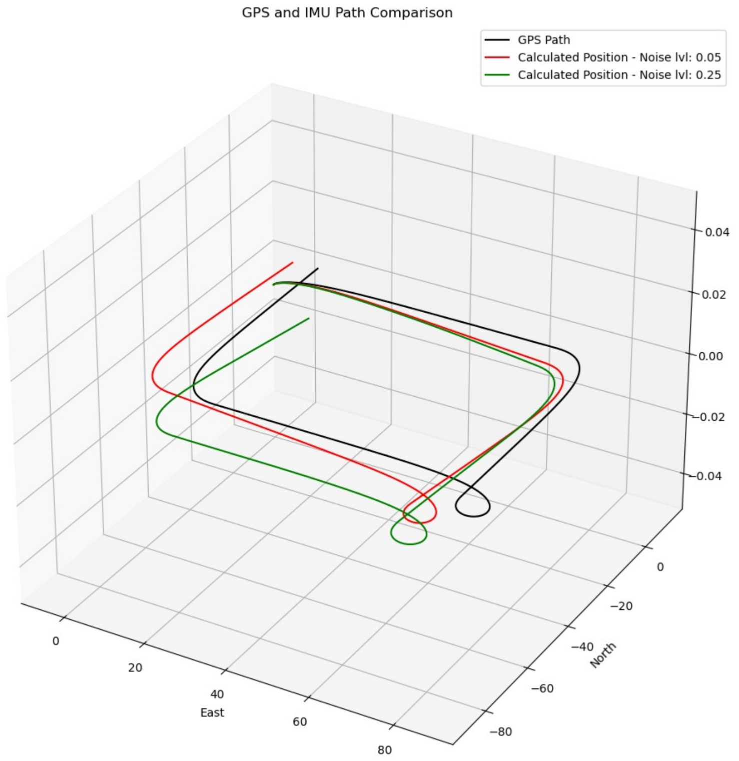
Methodology (cont.)

- We want to compare how introducing an additional sensor measurement will affect the localization solution.
- The process is repeated but instead of using the calculated velocity from IMU data, measured velocity data is used.
- For the purposes of this project, the measured velocity data will come from the GPS data. In a real GNSS denied scenario an alternative sensor type might be used to measure velocity and will depend on the environment/application.

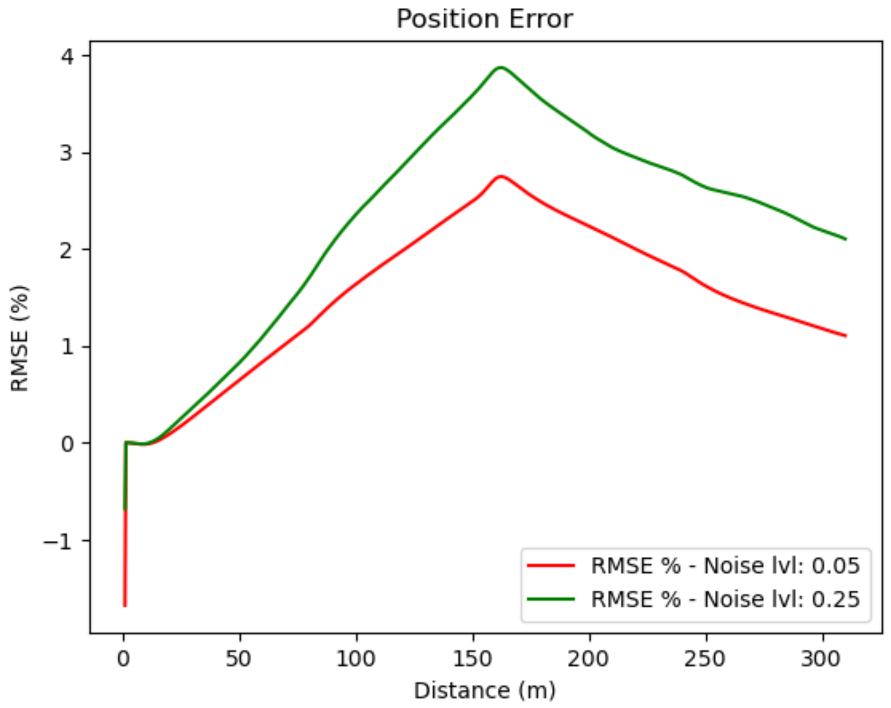
Results – Filtering Acceleration



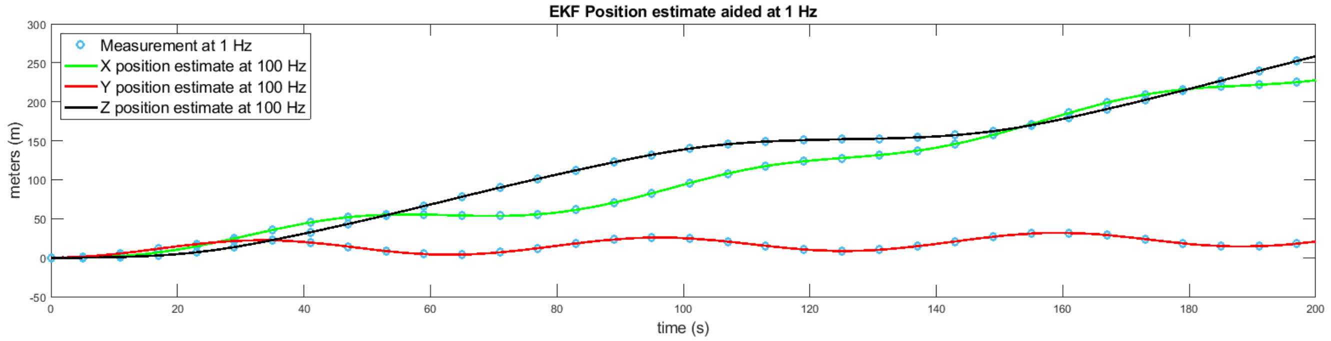
Results – Open-loop IMU Solution

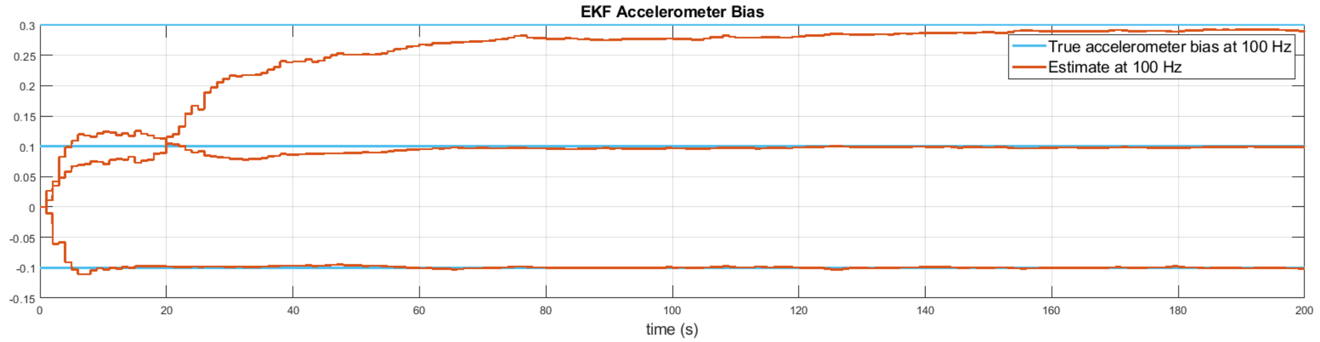


Results – Open-loop IMU Solution

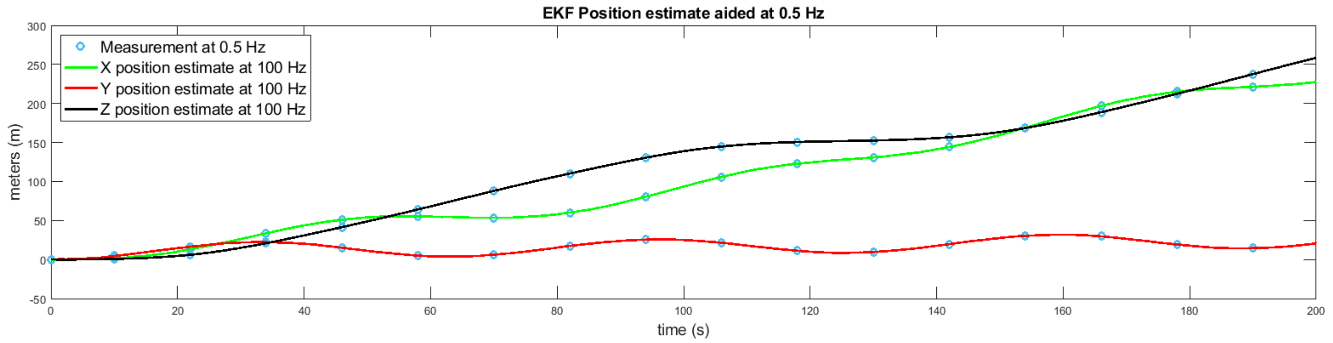


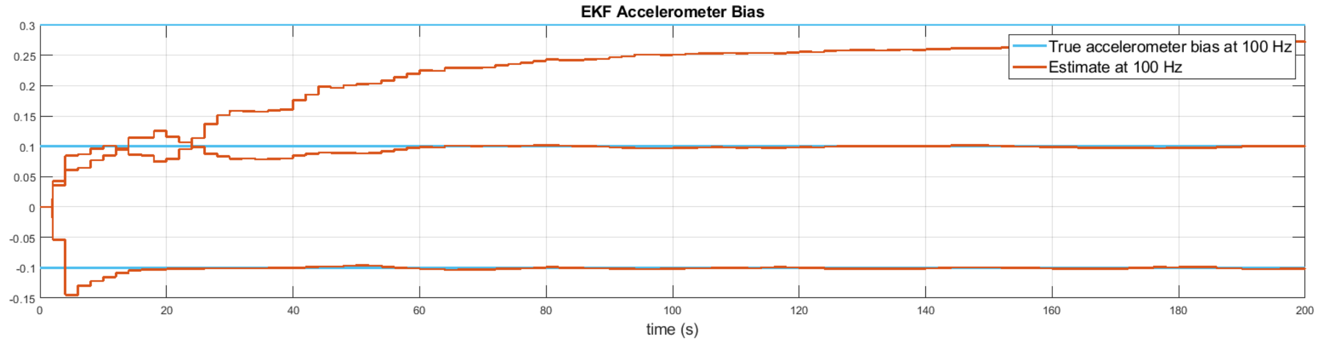
Results – Indirect Feedback Kalman Filter



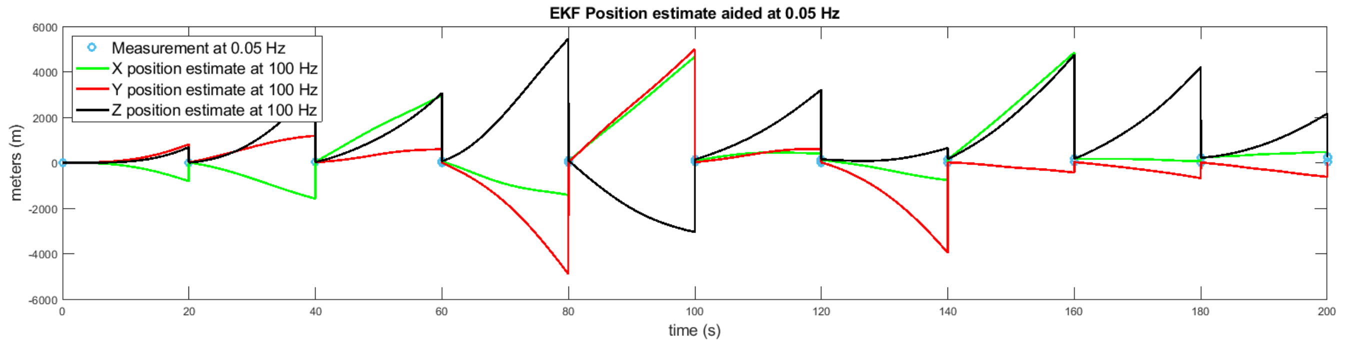


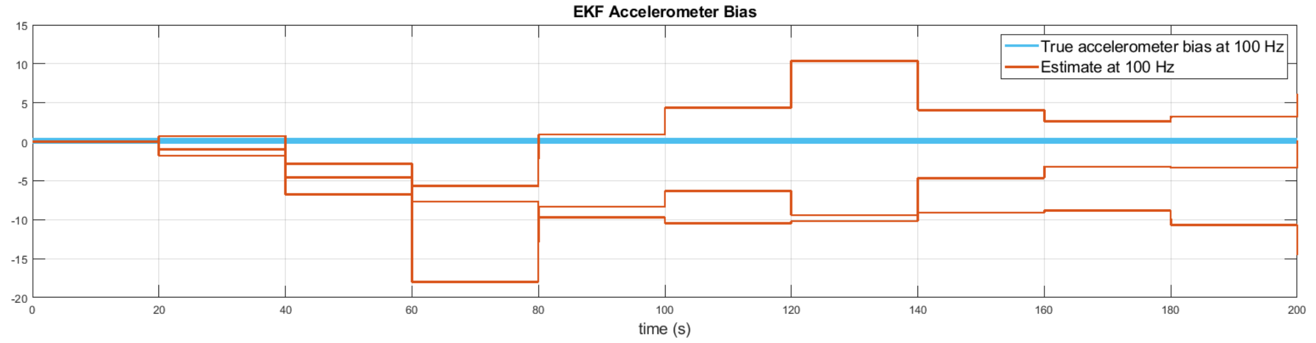
Results – Indirect Feedback Kalman Filter



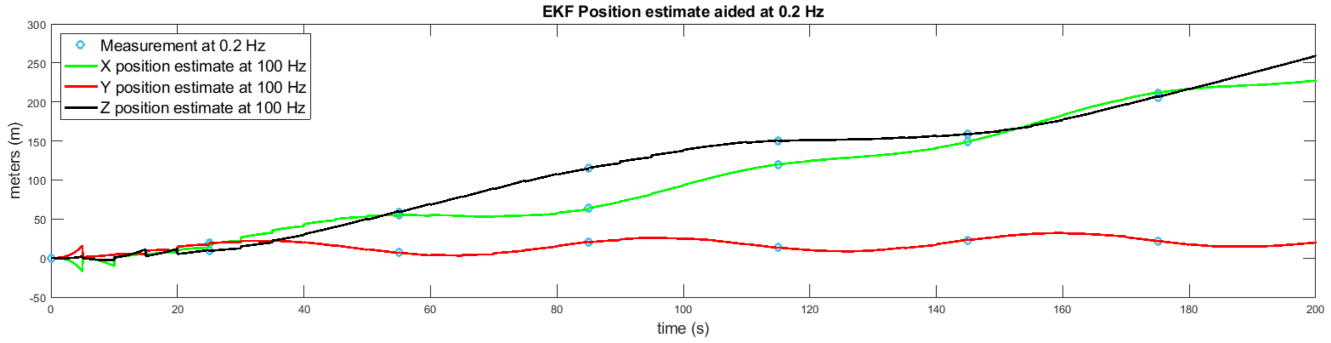


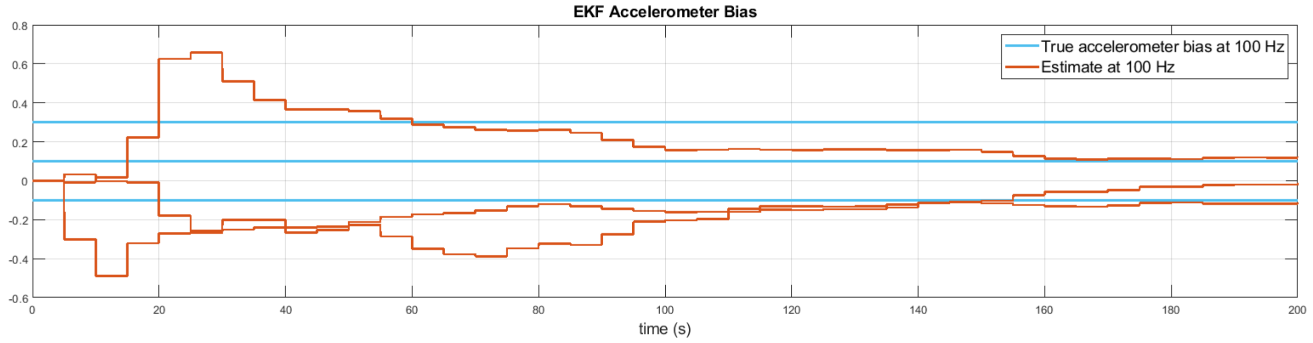
Results – Indirect Feedback Kalman Filter



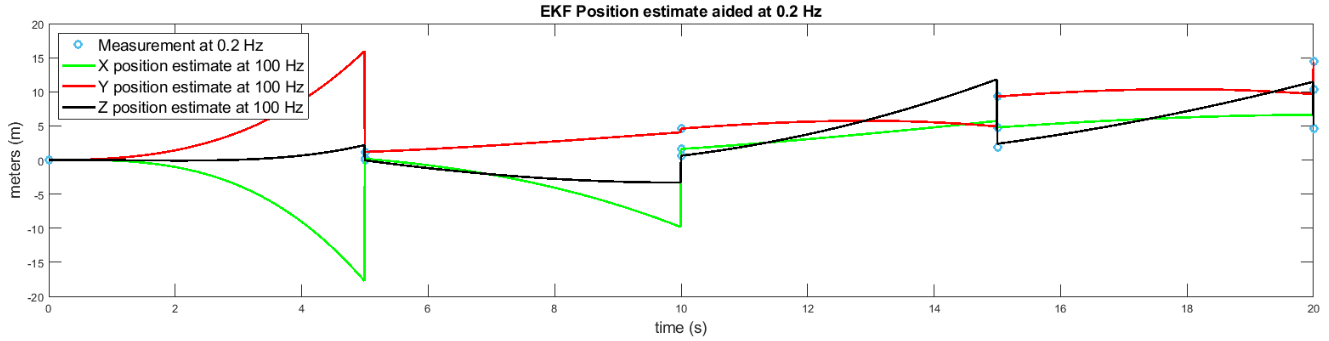


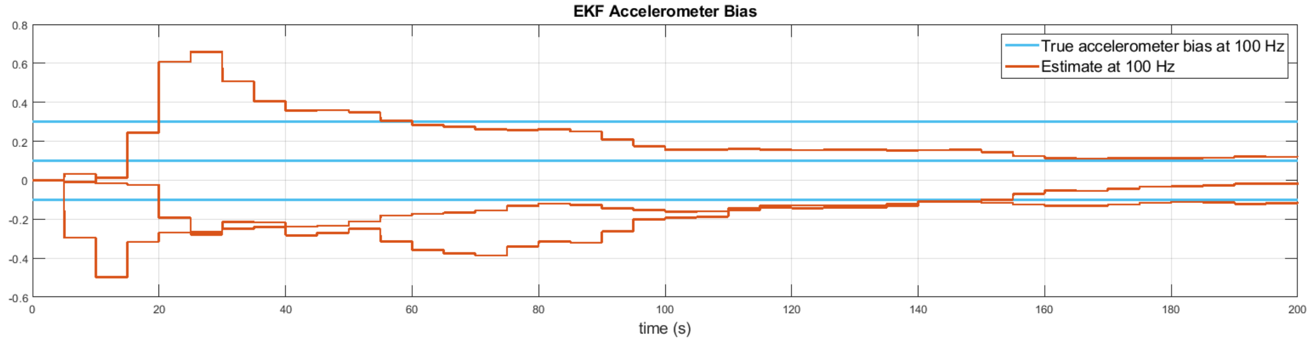
Results – Indirect Feedback Kalman Filter





Results – Indirect Feedback Kalman Filter





References

[1] Toy, A. Durdu and A. Yusefi, "Improved Dead Reckoning Localization using IMU Sensor," 2022 International Symposium on Electronics and Telecommunications (ISETC), Timisoara, Romania, 2022, pp. 1-5, doi: 10.1109/ISETC56213.2022.10010239.

[2] T. I. Fossen, *Handbook of Marine Craft Hydrodynamics and Motion Control,* 2nd ed. Hoboken, NJ, USA: Wiley, 2021.

Open-Source Software:

- Robot Operating System (ROS): https://docs.ros.org/
  - Various tools to be used for visualizing and working with data:
    - RVIZ
    - Gazebo
  - Various open-source libraries for implantation of filters.
  - Various open-source libraries aiding in static and dynamic transforms.
- UUV Simulator using ROS: https://uuvsimulator.github.io/

Questions?