SUB-TERRANEAN NAVIGATION PROJECT (V2)

INTRODUCTION

Welcome to the Sensor Fusion Course Final Project! In this exciting challenge, you and your teammate will step into the role of underground explorers racing through mysterious tunnels without GPS. Your mission? Build a state estimation algorithm that keeps track of your robot's position and orientation using onboard sensors and Time-of-Flight (ToF) distance measurements—just like navigating an uncharted sub-terranean world.

OVERVIEW AND OBJECTIVES

- **Team Size:** Pairs (2 students per team)
- **Core Task:** Estimate the robot's 2D position (x, y) and orientation (yaw angle) in real time, inside a motion capture arena.
- **Primary Hardware:**
    - **STM Nucleo IKS02A1 board** featuring:
        - ISM330DHCX (3-axis accelerometer + 3-axis gyroscope)
        - IIS2MDC (3-axis magnetometer)
        - IIS2DLPC (3-axis low-power accelerometer)
    - **X-NUCLEO-53L1A1** with three ToF sensors (one forward-facing, two sideways-facing).
- **Mobile Base carrying the Nucleo board**
    - **The platform can move sideways or rotate with Mecanum wheels**

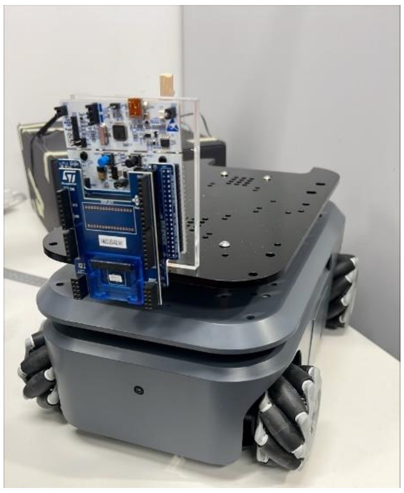

Figure 1 Mobile Base carrying the Nucleo board:

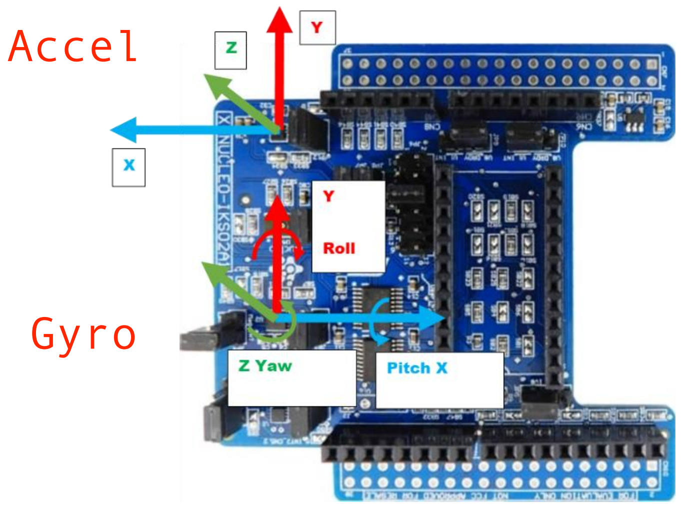

Figure 2 The coordinate system of the IMU and magnetometer. The Z axis is aligned with the forward direction of the robot with installation shown in figure 1 above.

ENVIRONMENT SETUP

- **Arena**: You'll operate the robot inside a motion capture facility with walls. The arena's dimensions and the robot's starting point will be provided beforehand.
- **Walls with Holes**: Some walls contain holes, potentially causing erroneous ToF readings (simulating real-world sensor noise/failures).
- **Ground Truth**: A PhaseSpace camera system with reflective markers on the robot will track the "true" position and orientation. This will be used to assess the accuracy of your algorithm.

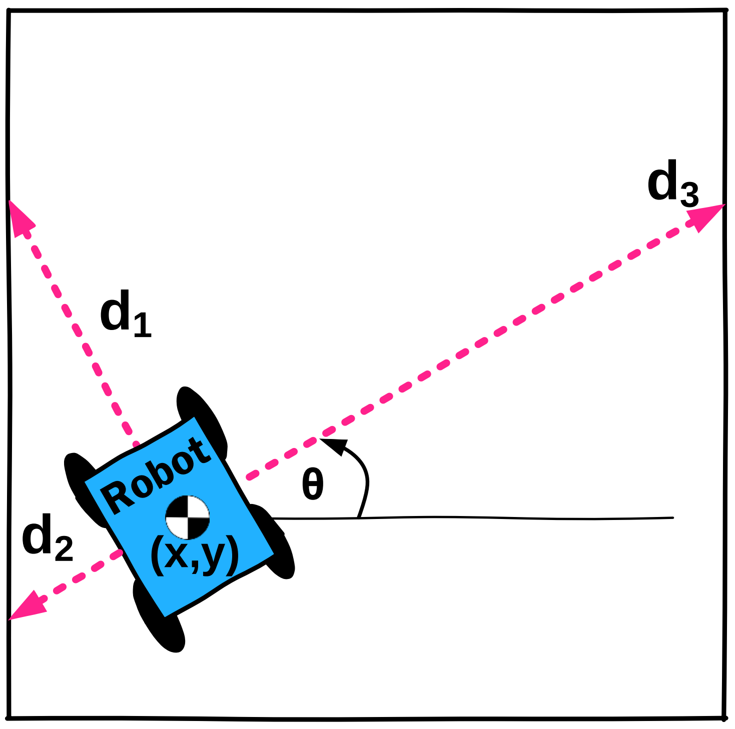

Figure 3: The robot will be equipped with three ToF sensors measuring the distances to the walls. The robot will also have IMU sensors.

The walls are of square dimension around 2.4m x 2.4m

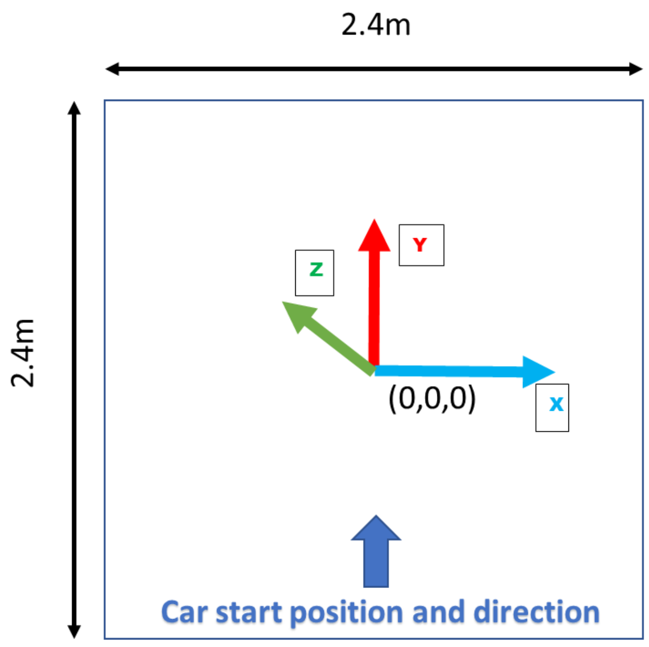

Figure 4 Dimension and coordinate system of the robot space

UPDATE: Instead of the running the project on your own hardware, we will provide collected datasets for you to run your estimator on. These will be:

1. **Training data 1:** We will provide a .mat file with logged data of the robot doing task 1's motion. This will include all the sensor data (IMU, magnetometer, temperature and 3x ToF) as well as the ground truth phase space data for you to use for developing your estimator.
2. **Training data 2:** We will provide a .mat file with logged data of the robot doing task 2's motion. This will include all the sensor data (IMU, magnetometer, temperature and 3x ToF) as well as the ground truth phase space data for you to use for developing your estimator.
3. **Calibration data 1:** We will provide a .mat file with logged data of the robot standing stationary for 1 minute, driving completely straight forwards and backward
4. **Calibration data 2:** We will provide a .mat file with logged data of the robot continuous rotating for 5 complete cycles.

The dataset also has ground truth position and orientation which is useful for aligning your estimation result and ground truth. You will need to use the first few points for starting position. Do not assume the robot start at (0,0,0).

The ground truth rotation is in quaternion (4 numbers). More detail can be found at: https://uk.mathworks.com/help/nav/ref/quaternion.html

The data format and structure are as follows (6362 is the example length of the dataset):

Struct Name	Length	Dimension	Update Frequency	Elements in Value
GT_position	[6362]	[3]	200 Hz	X, Y, Z [m]
GT_rotation	[6362]	[4]	200 Hz	W, X, Y, Z (Quaternion)
GT_time	[6362]	[1]	200 Hz	Time [s]
Sensor_ACCEL	[6362]	[3]	104 Hz	ISM330DHCX Acceleration X, Y, Z [m/s²]
Sensor_GYRO	[6362]	[3]	104 Hz	ISM330DHCX Gyroscope X, Y, Z [rad/s]
Sensor_LP_ACCEL	[6362]	[3]	100 Hz	IIS2DLPC Acceleration X, Y, Z [m/s²]
Sensor_MAG	[6362]	[3]	50 Hz	IIS2MDC Magnetometer X, Y, Z [T]
Sensor_Temp	[6362]	[1]	100 Hz	Temperature [°C]
Sensor_Time	[6362]	[1]	200 Hz	Time [s]
Sensor_ToF1	[6362]	[4]	10 Hz	Distance[m], Ambient[kcps/spad], Signal[kcps/spad], Status
			10 Hz	Status*:(0=Probably too close but OK, 2=Long OK, 4=Short OK) (for all ToF)*
Sensor_ToF2	[6362]	[4]	10 Hz	Distance[m], Ambient[kcps/spad], Signal[kcps/spad], Status
Sensor_ToF3	[6362]	[4]	10 Hz	Distance[m], Ambient[kcps/spad], Signal[kcps/spad], Status

PROJECT TASKS

1. **Straight-Line Drive**
   a. The robot moves forward and backward in a straight line at known intervals.
   b. Your algorithm must accurately estimate position and yaw during these simple movements.
2. **Circuit Course Drive**
   a. The robot drives a more complex path around the arena with varying velocity.

b. Exact path is unknown to you ahead of time (and differs across teams).
c. Your estimator must remain robust to changing velocity and direction.

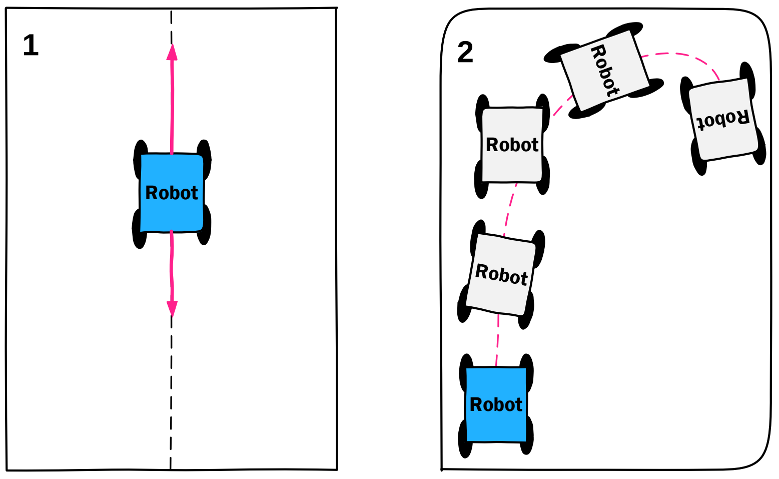

Figure 5: The two tasks for the project that need to be completed by your team. Task 3 is optional.

DELIVERABLES

- **Code:** Your finalized code for the state estimation and control interface.
- You will need to run your estimation using on the data and submit a .mat file with your state vector which we will use for the grading.
- **6-Page Report:** Explaining the design of your state estimator, sensor fusion methodology, and any relevant algorithms.
    - A detailed template will be released in the coming weeks.
    - Make sure to detail how you designed your estimator to tackle sensor noise, calibration, distance anomalies, real-time constraints, etc..
- **Submission**: The report and code should be submitted on to Moodle as a .zip file.

EVALUATION CRITERIA ON THE TRAJECTORY

- Use the compareTrajectories function in MATLAB ***R2024b*** *or newer* to compare the trajectory of your method and the ground truth.
- To achieve this, you will need to format your solution pose into rigidtform3d objects and compare them in compareTrajectories.
- rigidtform3d has two elements: position and rotation. Construct your solution to position and rotation first, and then form rigidtform3d object.
- The rotation in ground truth are represented as quaternions, you could use quat2eul to convert it to the Euler angle representation of rotations.
- Aside from **AbsoluteRMSE** (see the compareTrajectories page)error, what else should you consider for evaluating your solution. Do you need to consider the delay of your filter and fusion method?

Function requirement:

```
%Your function should take in the logged struct and output your state
estimate as well your state covariance and the ground truth (X,Y,Z).
[X_Est, P_Est, GT] = myEKF(out)
```

DELIVERABLES

- **Practical Performance (75%)**
    - Performance in Task 1, Task 2 (each weighted proportionally to difficulty).
    - Accuracy of your estimator (comparison to motion capture ground truth).
- **Report (25%)**
    - Clarity and depth of your explanation.
    - Quality of experimental results and analysis.
    - Insights into design trade-offs and lessons learned.