Lab 3

Dead Reckoning W/ and W/O IMU

Objective

The goals are to

- Understand the concept of dead reckoning using odometry
- Learn robot navigation using feedback control
- Experience Arduino programming, Inertia Measurement Unit (IMU) sensor, and linking ROS with Arduino
- Understand advantage and/or disadvantage of IMU

Reference Materials

- ROS Website: wiki.ros.org/ROS/Tutorials: Section 6-8
- Learning ROS for Robotics Programming by Aaron Martinez, Enrique Fernandez: Chapter 3, 4
- ROS By Example by R. Patrick Goebel: Chapter 8
- Introduction to Autonomous Mobile Robots by Siegwart, Nourbakhsh, and Scaramuzza: Chapter 3
- Lecture Notes 7-11
- "Robotics, Vision and Control" by Peter Corke: 4.1.1.4
- Arduino Website: https://www.arduino.cc/en/Tutorial/HomePage
- MPU6050 Tutorial: http://playground.arduino.cc/Main/MPU-6050
  https://diyhacking.com/arduino-mpu-6050-imu-sensor-tutorial/

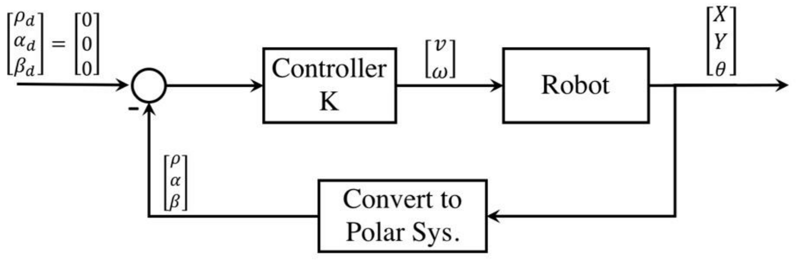
Prelab

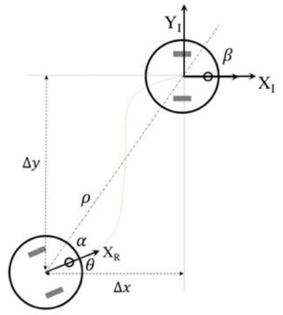
1. Summarize and describe http://wiki.ros.org/roscpp/Overview/Time tutorial.
2. Define Odometry and explain its importance in mobile robots.
3. What are the ROS topics and their related message types to use/monitor for the following tasks:
    a. Move the robot forward
    b. Rotate the robot
    c. Find the current pose of the robot
    d. Retrieve the IMU data

4. Give the sequence of ROS commands to find the message type of the "/odom" topic. What message type did you find? Provide the meaning of the message fields.
5. What is tf broadcaster? Explain (see http://wiki.ros.org/tf2/Tutorials)
6. Explain what is IMU and how it can help us?
7. What are the outputs of MPU6050?
8. Briefly introduce Arduino board and its advantages/capabilities?

Background

Feedback Control: In this lab, you will understand the importance of the feedback gains in a discrete state feedback control for dead reckoning of the robot. The figure below shows the standard full state feedback diagram:


$\begin{bmatrix}\rho_d\\ \alpha_d\\ \beta_d\end{bmatrix}=\begin{bmatrix}0\\0\\0\end{bmatrix}$
-
Controller K
$\begin{bmatrix}v\\ \omega\end{bmatrix}$
Robot
$\begin{bmatrix}X\\Y\\ \theta\end{bmatrix}$
$\begin{bmatrix}\rho\\ \alpha\\ \beta\end{bmatrix}$
Convert to Polar Sys.



where here the states are in polar coordinate $[\rho, \alpha, \beta]^T$ and the desired reference is $[\rho_d, \alpha_d,\ \beta_d]^T = [0, 0, 0]^T$, control signals $[v, \omega]^T$ are the linear and angular velocities and $[K_\rho, K_\alpha, K_\beta]$ are the control gains which should be tuned by experiments. The equations can be derived as:

$$\begin{bmatrix}v\\ \omega\end{bmatrix}=\begin{bmatrix}K_\rho & 0 & 0\\ 0 & K_\alpha & K_\beta\end{bmatrix}\times\begin{bmatrix}\dot\rho\\ \dot\alpha\\ \dot\beta\end{bmatrix}$$

$$\begin{bmatrix}\rho(k+1)\\ \alpha(k+1)\\ \beta(k+1)\end{bmatrix}=\begin{bmatrix}-K_\rho cos\alpha\\ K_\rho sin\alpha - K_\alpha\alpha - K_\beta\beta\\ -K_\rho sin\alpha\end{bmatrix}$$

The output here is the pose of the robot $[X, Y, \theta]$ that should be monitored during the movement of the robot using odometry information. Note that here the system is the robot kinematics in polar coordinate. The conversion from Cartesian to polar coordinate can be found in the reference materials.

IMU Sensor: you will also be using an Arduino UNO (figure b) board to collect data from an IMU sensor, MPU6050, shown in figure (a):

(a) MPU6050

(b) Arduino UNO

The IMU sensor has been mounted on the robot (approximately aligned with **x** pointing forward and **y** pointing left). The Arduino board collects data from sensor through I2C protocol and send them to raspberry pi through USB serial port (see reference material how to use rosserial package). Your task is to control the robot using orientation (θ) coming from IMU sensor and position (x, y) coming from the wheel odometry.

Lab Procedure

Week 1: Implementation of Dead Reckoning

In this part, you will implement the dead reckoning using feedback control and test your program in the simulator (e.g. Stage or Gazebo). You will need odometry information published by the simulator under the /odom topic. Write your program to navigate the robot to the four goal poses as specified in Figure 1 which shows four experiments that need to be performed. For each experiment, report the error measured between the desired goal and the reached goal.

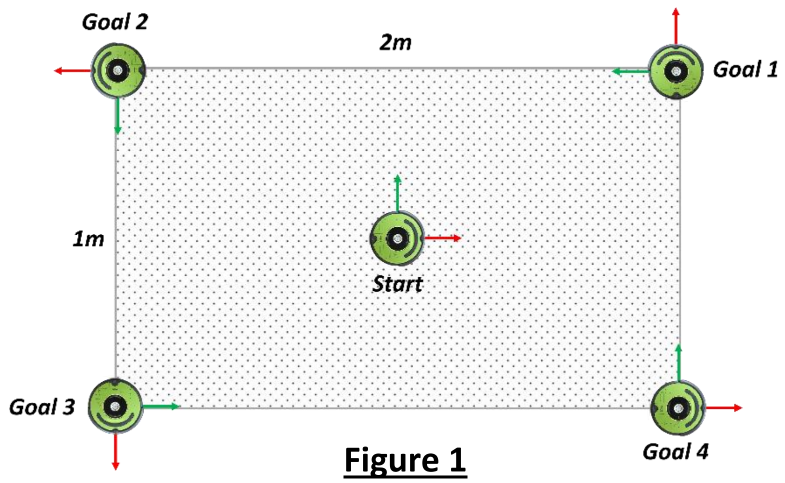


Figure 1

Below are some high-level steps you can take to implement your code:

1. Write the odometry callback function to subscribe to /odom topic and get the pose of the robot. This is the robot pose at current time frame i.
2. Write a function to convert cartesian coordinate to polar coordinate frame.
3. Write a function to transform between coordinate frames and convert the robot pose with respect to Goal frame.
4. Implement a discrete state feedback control using the above equations (look at the lecture notes for detailed equations). That is find the velocity command from feedback controller and send it robot. Try to tune feedback gains and report all your experiments.

Week 2: Dead Reckoning W/O IMU

Part 1: in the simulator

In this part, you will empirically choose the feedback gains of the dead reckoning and test a feedback control program in the simulator.

1. Download the incomplete feedback control program from Canvas.
2. Understand all the functions in the feedback control program.
3. Complete the missing statements in the feedback control program.

Using your implementation from Part 0 or the provided implementation for which you have completed the missing statements:

4. Find the right set of $[K_\rho, K_\alpha, K_\beta]$ feedback gains to reproduce Figures 2 and 3 where the robot needs to navigate to 8 final poses from 2 starting poses.
5. Save and plot the robot trajectories as in Figures 2 and 3 to include in your final report.
6. For each goal, report the error between the desired final pose and the reached pose.
7. Reproduce the experiments from Part 0 (look at Figure 1).

Part 2: in a real environment

Once you have tested your program using the simulator, you can run it on the real robot. You may need to change feedback gains and re-tune them to get best results.

1. Are the feedback gains tuned in the simulator performing the same with the real robot when trying to reproduce Figures 2 and 3? How about for the experiments from Part 0 (Figure 1) performed in the simulator?
2. Save and plot the robot trajectories as in Figures 2 and 3 to include in your final report.
3. For each goal, report the error between the desired final pose and the reached pose.

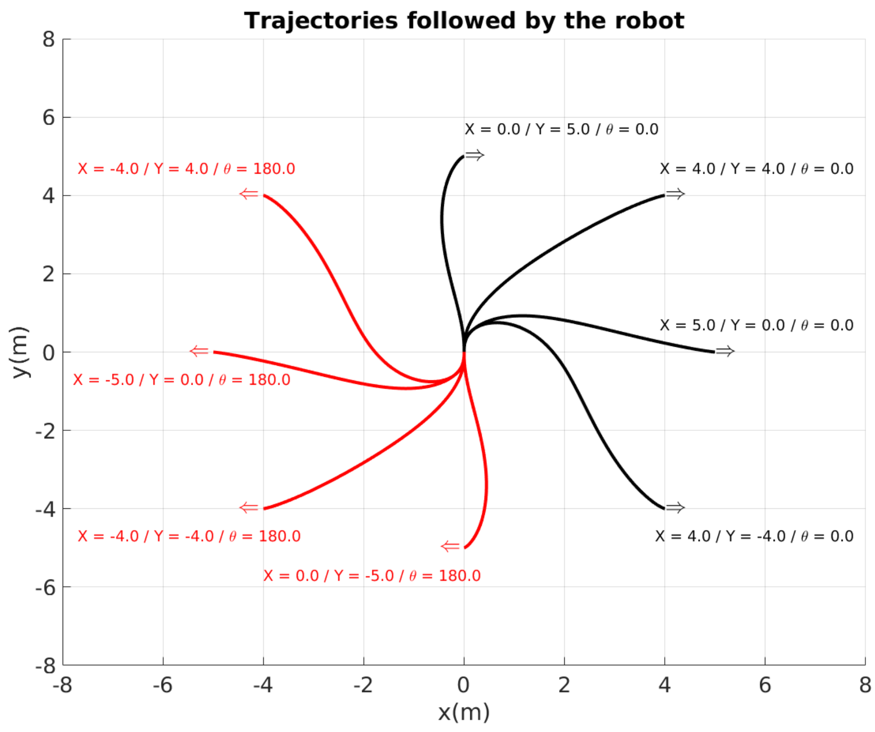


Figure 2

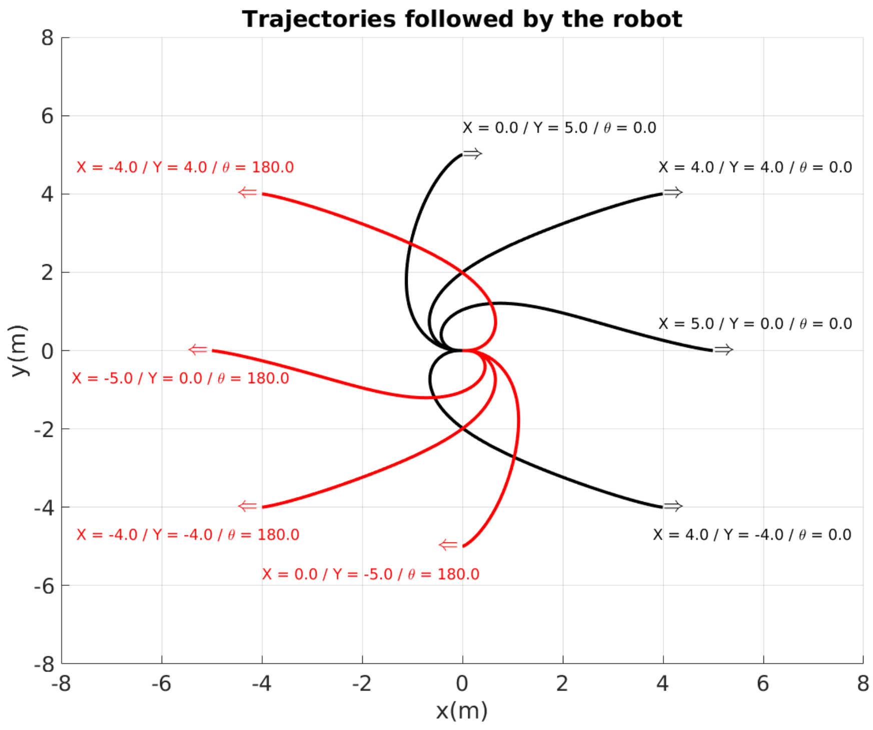


Figure 3

Week 3: Moving the robot using the IMU

Part 1: Getting Started with Arduino and IMU (MPU-6050)

In this part, you will start using the Arduino board to collect data from IMU sensor. Make sure you have reviewed the reference materials related to the Arduino board and the MPU-6050 sensor. Below are the steps for this part:

1. **Arduino board:** Arduino UNO is assembled on Create 2 robots and connected to raspberry pi through USB port. Log in to your raspberry pi ($slogin –Y pi@10.14.1.23x) and run $arduino, you will see the Arduino IDE. You can write your program and compile by pressing tick button and upload to the board by pressing arrow button. You can use Arduino library where language reference can be found from reference material. Write a simple hello world program and try to learn Arduino IDE environment, analog/digital ports and communication over serial port.

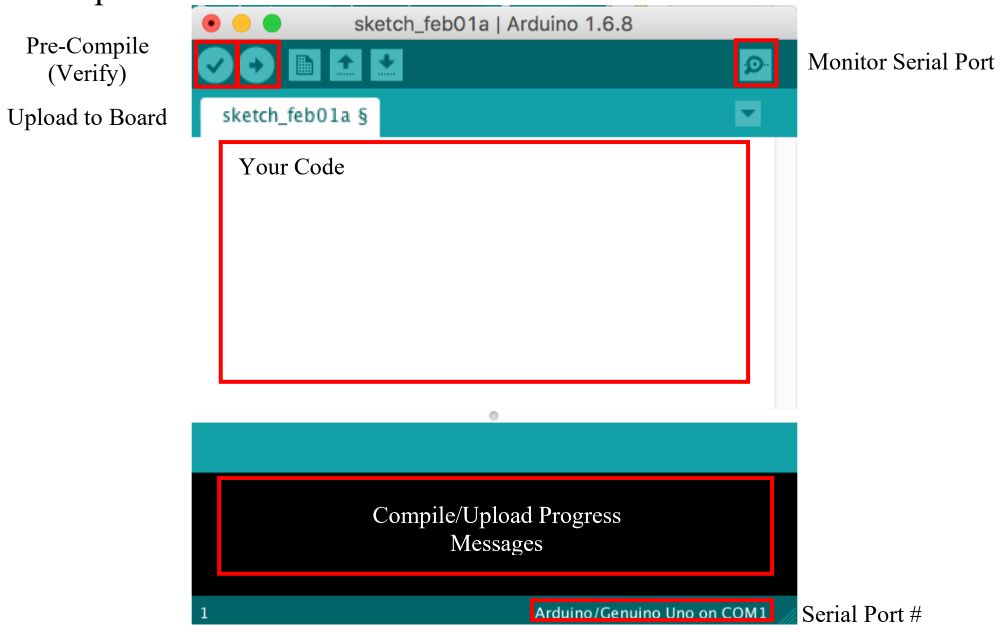


Figure 4

2. **MPU-6050:** An IMU sensor, MPU-6050 is also assembled on the robot and connected to Arduino board. Check all wiring and make sure they are connected to the right pins (see reference material). The sensor sends data through I2C protocol where two wires are required (SDA: pin A4, SCL: pin A5). All libraries required for this sensor can be found under ~/sketchbook /libraries directory on the raspberry pi. You can open an example in IDE by File>Examples>MPU6050>MPU6050_DMP6. Try to go through the examples and understand the sample codes, upload to the board and monitor the data. **Is there any consideration that should be taken? Interpret the data obtained by sensor when you move the robot.**

Part 2: Making Arduino as a ROS node

In this part, you will make the Arduino to act as a node communicating with the master running on your raspberry pi. That is, using the rosserial library (see reference material); program the Arduino board to publish IMU data under a topic name so that your navigation node can subscribe to that topic and read the data. Follow the steps below to finish this part:

1. Write a talker node and upload it to the Arduino board. Use the listener node you wrote in lab2 to subscriber to the Arduino talker and print the received messages. Go to ~/sketchbook/libraries directory and run:
   ```
   $ rm –rf ros_lib
   $ rosrun rosserial_arduino make_libraries.py .
   ```
   This will create ros libraries required to upload your talker code to Arduino board. Now you can run your talker and listener node by:
   ```
   $ roscore
   $ rosrun rosserial_python serial_node.py _port:=/dev/ttyACM0
   ```
   make sure you are using right serial port (see figure above). Now you can run your listener node to print messages sent by talker.
2. Combine the Arduino example code (MPU6050) with above to write a node collecting data from IMU sensor and publishing to an appropriate topic (e.g. /imu).
3. Show the data sent by your program using $rostopic echo /topic_name. Use rqt_plot node to illustrate your results.
4. Finally, modify your week 1 codes to get the position from the Odometry and the orientation from the IMU sensor. Use Rviz to visualize the movement of the robot and plot the robot’s path.

Extra Credit (30 pts):

5. Modify your feedback implementation to get the position and orientation from IMU sensor. Use Rviz to visualize the movement of the robot and plot the robot’s path.

Post Lab Questions:

1. Derive the two equations used for feedback control? Did you locate both equations in the feedback control program?
2. How varying each feedback gain $K_\rho, K_\alpha, K_\beta$ affects robot motion? Explain with your experiments.
3. Is there any value for feedback gains that make the robot unstable? Explain with your experiments.
4. What are the outputs of IMU sensor? How are they correlated with the robot? Plot all possible outputs of the sensor and describe each of them?
5. Is there any consideration for the IMU sensor? Are you getting accurate data from the IMU sensor? Explain by your experiments.
6. What message type did you use to publish IMU data, and why? Is there any limitation for a specific message type?