Sensors and Motors Lab

Individual Lab Report 1

Aaditya Saraiya

February 14th, 2019

Team E:

Wholesome Robotics

Teammates:

Aaditya Saraiya

John Macdonald

Dung-Han Lee

Aman Agarwal

Hillel Hochsztein

Individual Progress

Sensors and Motors Lab

For the sensors and motors, I was assigned the task of interfacing the Infrared Distance sensor with the Arduino microcontroller. I was also co-tasked with creating a velocity controller for the DC motor and creating the code interfaces for DC motor code for ease of use with the GUI.

Infrared Sensor Interfacing

Introduction: The IR sensor consists of an IR LED emitter and a photo-sensitive array of detectors. The IR LED transmits rays of infrared wavelength which are reflected from an object. Depending on the position of these rays on the photo-sensitive array, the output voltage of the IR sensor will change.

Wiring: This first part of the Infrared sensor interfacing involved wiring the sensor to the breadboard. The 5V input and ground was provided to the IR sensor directly from the Arduino. The signal pin of the sensor was attached to the A1 (Analog 1) pin.

Explanation of Code: The infrared sensor outputs voltage values proportional to the distance of the sensor from the object. Depending on a transfer function (taken from datasheet and experimentally tested), the voltage of the sensor is converted into distance values. The sensors operating range (with reliable distance values) was obtained between 8 – 59 inches. The code is attached in Appendix 1) of the assignment.

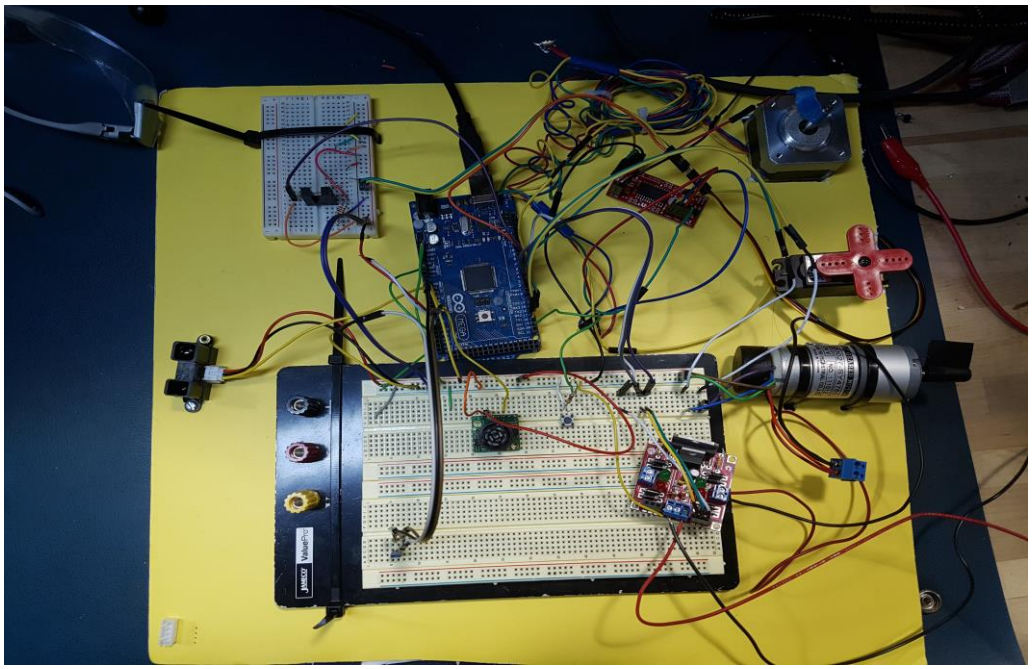

Figure 1: Layout of motor and sensor interfaced to Arduino board

DC Motor Interfacing

For this part of project, my task was implementing velocity control for the DC motor and creating pre-defined code interfaced for the DC motor to be used with the GUI. Aman had previously worked on position control of the DC motor and had interfaced the encoder with the motor.

Velocity Control: The main aim of velocity control is to create a PID control loop which allows the DC motor to rotate at a pre-determined velocity. The maximum speed of the motor was experimentally calculated to be 150 RPM. The setpoint provided from the motor was scaled between 0-255 in order to be utilized with the Arduino. The position of the motor was calculated between two time instances (5 milliseconds in this case). The change in position per time was converted into RPM by dividing the values by the gear ratio (27) and the ticks per revolution (112). Depending on the direction in which the motor was to be rotated, the corresponding pin on the DC motor driver was turned on. A term which was proportional to the encoder measurement was added to balance for the error in the model. The PID library was utilised to calculate the PID output.

Converting to pre-defined code interfaces

In order to assist John in the GUI development process, a standardised code interface was pre-decided and implemented to reduce the integration time. This provided significant challenges which have been described further in the Challenges section.

MRSD Capstone Project

For the Capstone project, I have been working on the Simultaneous Localization and Mapping pipeline for the system. Team E is creating an organic monitoring and weeding robot which has to autonomously navigate through the crop rows. My task develop a SLAM pipeline to develop a geometrically registered map of the farm and provide it as an input to the localization algorithm.

At the time of CoDR, a literature survey of various SLAM algorithms had been carried out. Following up on that, the Lidar Odometry and Mapping package was considered apt for initial trials. The LOAM pipeline has been summarised in the flowchart in Figure 2. The LOAM algorithm is optimised to compute edge and plane features from raw point cloud data and utilises the Levenberg- Marquadt algorithm to compute the 6DOF pose transformation between subsequent point clouds.

As initial steps, the package was obtained from code repositories from George Kantor's lab. The time was spent on understanding the feature extraction and Lidar odometry pipeline. The ROS package with its dependencies was built. The Velodyne-16 Lidar was interfaced with the package (with some support from Dun-Han). The raw point cloud data from the Velodyne-16 Puck can be visualised in Figure 3.

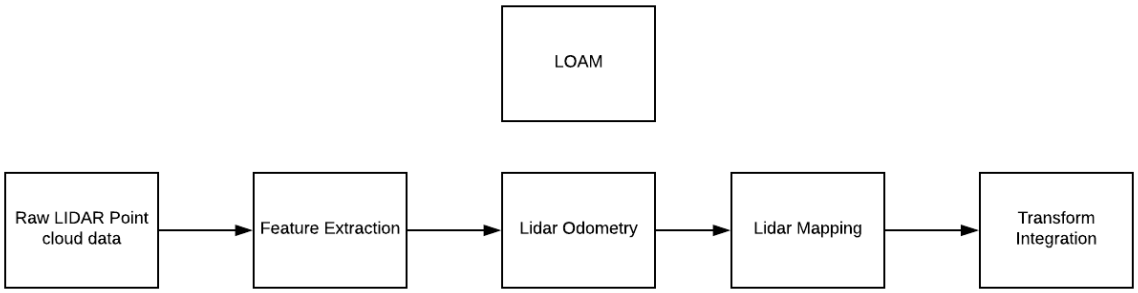

Figure 2: Lidar Odometry and Mapping pipeline

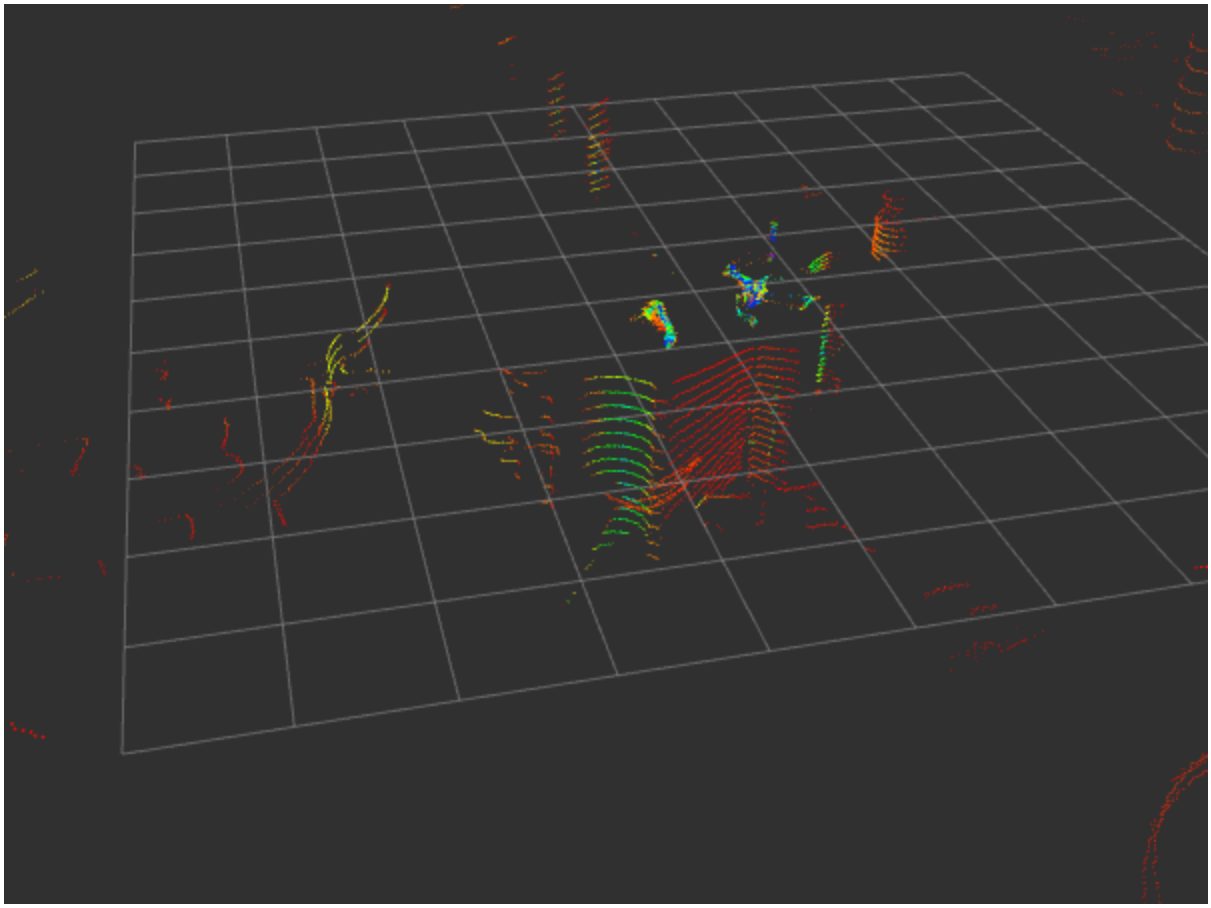

Figure 3: Raw point cloud data interfaced from the Velodyne-16 LIDAR

Challenges

Sensors and Motors Lab

The main challenge I faced during sensor interfacing and DC motor control was developing the code interfaces for the particular sensor. As a team, we had discussed on ways to create modular, easy-to-use code for the GUI. Writing modular software made us focus on good object oriented programming practices and approaches which took some time to get used to it.

A specific challenge was for velocity control of the DC motor. The velocity output was changing in a very short interval of time which was leading to overflow. This part was solved by computing the velocity after 5 milliseconds.

Capstone Project

The main challenge related to the work on the SLAM pipeline was to understand the theoretical concepts behind SLAM. Specifically, it took time to choose a SLAM algorithm for the LIDAR sensor, as many of the SLAM algorithms are not robust for outdoor use. This specific challenge was solved by consulting the research students from Prof. George Kantors lab, who recommended utilising the Lidar Odometry and Mapping (LOAM) pipeline with the Velodyne-16 Puck sensor.

Teamwork

Sensors and Motors Lab

John Macdonald

John was responsible for creating the GUI for creating a real-time control and visualisation platform all the sensors and motors. He was involved with developing well-defined code interfaces and functions for all sensors and motors.

Aman Agarwal

Aman was responsible for interfacing with the temperature sensor. He also co-tasked with developing the velocity and speed controllers for the DC motor. He was involved with the debugging and sensor interfacing throughout the assignment.

Hillel Hochsztein

Hillel was responsible for interfacing with the slot sensor and the servo motor. He also worked on creating a wooden board for mounting the circuits for the presentation.

Dung-Han Lee

Dung-Han Lee was responsible for interfacing the ultrasonic sensors, visualising the transfer function for the ultrasonic sensors. He was also responsible for interfacing and testing the stepper motor.

Capstone Project

John Macdonald

John was working on creating a localisation module for the robot. His recent work involved utilising the GPS registered point cloud data collected from the previous farm trials to localise within a crop row using a particle filter.

Aman Agarwal

Aman was recently working on managing and making improvement on 3D CAD Model of the complete robot in order to get the files ready for manufacturing.

Hillel Hochsztein

Hillel was recently developing the 3D CAD designs for the initial versions of the end-effector which will be utilised for weed removal.

Dung-Han Lee

Dung-Han Lee was recently working on creating a training pipeline using Masked R-CNN for detection of holes and diseases in the plant data acquired from previous farm trials.

Future Plans

Sensors and Motors Lab

The future plan includes extending the concepts learnt behind DC motor control and sensor interfacing to the HEBI motors. These motors will be interfaced with the Intel NUC computer and will be utilized to drive the four wheel of the robot. The future challenges in this includes adapting the DC motor code developed in this assignment to motors with heavier load and protecting power electronics equipment.

Capstone Project

For the SLAM pipeline, the next step will be to test the LOAM package on Velodyne-16 point cloud data obtained in real-time. The same pipeline will also be potentially tested with the ROS Bag files which were collected from previous field trials. A major future challenge would be to test the SLAM package under uncertain conditions. A previously recorded map could become redundant with the growth in the plants. These problems could be potentially solved by using visual odometry and loop closures using a forward-facing stereo camera.

Appendix

1) IR Sensor code

Ir.h

```
#ifndef IR_H

void init_ir_dist();
float get_ir_dist();

#define IR_H
#endif /* IR_H */
```

Ir.ino

```
#include "ir.h"

void init_ir_dist()  {}

float get_ir_dist()  {
  int ir_sensor_val = analogRead(A1);
  float dist = (10650.08 * pow(ir_sensor_val,-0.935) - 10)*0.393701;

  if (dist > 59.0) {
    dist = 59.0;
  } else if (dist < 8.0) {
    dist = 8.0;
  }

  return dist;
}
```

2) DC Motor code

Dc_motor.h

```
#ifndef DC_MOTOR_H

#include <Encoder.h>

#include<elapsedMillis.h>

#include <PID_v1.h>

enum DcMotorMode { position = 0, velocity = 1 };

#define DC_ENABLE 7

#define DC_OUT1 2

#define DC_OUT2 3

double oldPosition =  -999;
```

```
double speedMot=         0;
double newP=             0;
double oldP=             0;
double t=                0;
double Output1, Output2;
double Setpoint, Input, Output, OutputNet;
double SetpointPos, InputPos, OutputPos;
float kp_pos= 0.6, kd_pos= 0.01, ki_pos= 0.005, kp_vel= 1;
float ki_vel= 0.7;
float kd_vel= 0.005;
struct dc_motor_data {
DcMotorMode mode;
double setpoint_vel= 30;
double setpoint_pos= 756;
double input;
double output;
};
Encoder dc_motor_encoder(18, 19);
elapsedMillis timeElapsed;
PID PID_pos(&InputPos, &OutputPos, &SetpointPos, kp_pos, ki_pos, kd_pos,
DIRECT);
PID PID_vel(&Input, &Output, &Setpoint, kp_vel, ki_vel, kd_vel, DIRECT);
void init_dc_motor(struct dc_motor_data* data);
// degrees range from 0 to 359, positive is CW
void handle_dc_motor(struct dc_motor_data* data);
#define DC_MOTOR_H
#endif /* DC_MOTOR_H */
```

Dc_motor.ino

```
#include "dc_motor.h"
void init_dc_motor(struct dc_motor_data* data)  {
    pinMode(DC_ENABLE, OUTPUT);
    pinMode(DC_OUT1, OUTPUT);
```

```
    pinMode(DC_OUT2, OUTPUT);
    PID_pos.SetMode(AUTOMATIC);
    PID_pos.SetOutputLimits(-255, 255);
    //Setpoint= data -> setpoint_vel;
    PID_vel.SetMode(AUTOMATIC);
}
// degrees range from 0 to 359, positive is CW
void handle_dc_motor(struct dc_motor_data* data)  {
 if(data-> mode == 0)      // Doing position control
    {
      double newPosition = dc_motor_encoder.read();
      InputPos= newPosition;
      PID_pos.Compute();
      analogWrite(DC_ENABLE,255);
      if (OutputPos>=0){
        Output2=OutputPos;
        Output1=0;
      }
      else{
        Output2=0;
        Output1=-OutputPos;
      }
      analogWrite(DC_OUT1, Output1);
      analogWrite(DC_OUT2, Output2);
    }
    if(data-> mode == 1)     // Doing velocity control
    {
      double baseline=Setpoint*255/150;
      double newPosition = dc_motor_encoder.read();
      if (newPosition != oldPosition && timeElapsed>5){
          newP=newPosition;
          oldP=oldPosition;
          t=timeElapsed;
```

```
            oldPosition = newPosition;
            speedMot=((oldP-newP)*1000*60)/(7*16*27*t); //RPM
            //speedMot=((oldP-newP)*1000)/(7*16*27*t); //RPS
            Input = speedMot;
            //analogWrite(motor_out1, Output);
            /* Serial.print("Motor speed "); */
            /* Serial.println(speedMot,6); */
            timeElapsed=0;
        }
        PID_vel.Compute();
        OutputNet=baseline+Output+0.03*speedMot;
        if (OutputNet > 255.) { OutputNet = 255.0; }
        analogWrite(DC_ENABLE,255);
        analogWrite(DC_OUT1, OutputNet);
        analogWrite(DC_OUT2, 0);
    }
}
```

Task 4 (Sensors and Motor Control Lab) Quiz

1. Reading a datasheet. Refer to the ADXL335 accelerometer datasheet (https://www.sparkfun.com/datasheets/Components/SMD/adxl335.pdf) to answer the below questions.
   - What is the sensor's range?

   A: ADXL335 has a measurement range of ± 3g (where g is 9.8 ms-2)

   - What is the sensor's dynamic range?

   A: Dynamic range: 6g (Taking the typical value of the sensor in question)

   - What is the purpose of the capacitor C_{DC} on the LHS of the functional block diagram on p. 1? How does it achieve this?

     A: C_{DC} allows to decouple noise from the sensor values. It is a decoupling capacitor. It does this by suppressing high frequency noise in the voltage supply.
     Case 1: If frequency of signal is high, the capacitance is low. Hence, the path from power to ground is closed and the high frequency signal goes to the ground.
     Case 2: If the frequency of the signal is low, the capacitance is high. Hence, the capacitor acts as an open circuit and the low frequency signal is allowed to pass through the circuit. This is how high frequency voltage ripples are removed using a capacitor.

   - Write an equation for the sensor's transfer function.

   A: Vout/a = 1.5V + (300 mV/g)

   where Vout is the output voltage, a is the input acceleration

What is the largest expected nonlinearity error in g?

A: ± 0.3 % of full scaled output, where full scaled output is 3V, which is equal to 0.009 V.

- How much noise do you expect in the X- and Y-axis sensor signals when the sensor is excited at 25 Hz?

  A: As noise density for Xout and Yout can be given using the following formula.
  Noise density= 150 µg* $\sqrt{25 * 1.6}$ = 30 µg
  RMS Noise= Noise Density x ($\sqrt{Hz * 1.6}$)
  Using 25 Hz, noise density is 948.68 µg

- How about at 0 Hz? If you can't get this from the datasheet, how would you determine it experimentally?

  A: Using the previous formula, 150 µg* $\sqrt{0 * 1.6}$ = 0 µg.
  This value maybe mathematically correct, however, in reality, the RMS value of noise is not zero. Hence, the sensor should be kept on stationary, to simulate 0 Hz. Using the output data from the stationary sensor (at 1.5 V which is the bias), the RMS value of the sensor can be experimentally determined.

2. Signal conditioning
   - Filtering
     - What problem(s) might you have in applying a moving average?

       A: Disadvantages of applying moving average filter can be summarized as follows:
       - Moving average filter is unable to model the uncertainty of complex noise models.
       - There is a time delay in responding to a variation in signal trends.

     - What problem(s) might you have in applying a median filter?

     A:

     1) In order to apply a median filter successfully, you will have to sort sensor data which will require additional memory and processing (N log N) for the best sorting algorithms.
     2) Median filter is biased against extreme values. Hence, it may be biased to remove values which are at the end of the transfer function.

   - Opamps
     - In the following questions, you want to calibrate a linear sensor using the circuit in Fig. 1 so that its output range is 0 to 5V.

     Identify: 1) which of V1 and V2 will be the input voltage and which the reference voltage;
     2) the value of the reference voltage; and
     3) the value of Rf/Ri in each case. If the calibration can't be done with this circuit, explain why.
       - Your uncalibrated sensor has a range of -1.5 to 1.0V (-1.5V should give a 0V output and 1.0V should give a 5V output).
       - Your uncalibrated sensor has a range of -2.5 to 2.5V (-2.5V should give a 0V output and 2.5V should give a 5V output).

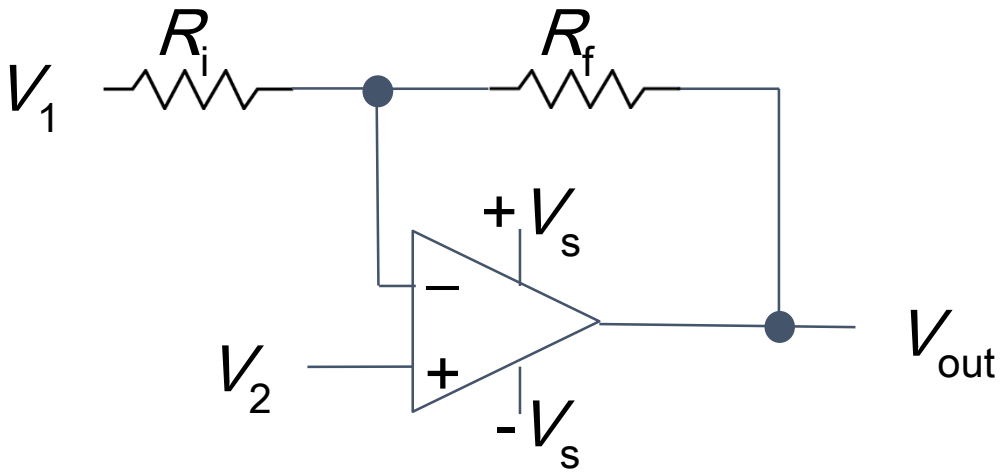

Fig. 1 Opamp gain and offset circuit

A:

$$\frac{V1 - V2}{R_i} = \frac{V2 - Vout}{R_f}$$

Simplifying the equation with the gain K= $\frac{R_f}{R_i}$, the following equation can be derived

$$Vout = V2(1 + K) - KV1$$

For the sensor, taking V2 as input

Case 1: (-1.5 to 1 V)

Taking (Vin = -1.5V , Vout= 0)
0= -1.5(1+K)- KV1

Taking (Vin = 1V , Vout= 5)
5= 1(1+K) – KV1
Solving simultaneously, K= 1 and V1 is -3 V

Case 2: (-2.5 to 2.5 V)

Taking (Vin = -2.5V , Vout= 0)
0= -2.5(1+K)- K (-2.5)

Taking (Vin = 2.5V , Vout= 5)
5= 2.5(1+K) – K(2.5)
Solving simultaneously, we can find that the solution cannot be found.

The calibration cannot be done as the gain obtained from the following calculation is negative which does not make sense for an amplifier.

3. Control
   - If you want to control a DC motor to go to a desired position, describe how to form a digital input for each of the PID (Proportional, Integral, Derivative) terms.
     A:

   - If the system you want to control is sluggish, which PID term(s) will you use and why?
     A: If the system is sluggish, you will apply P control to system as this will increase the output by a term that is proportional to the error.

   - After applying the control in the previous question, if the system still has significant steady-state error, which PID term(s) will you use and why?
     A: If the system has significant steady state error, the integral (I) control is utilized. This method acts on the accumulated steady state error over time and increases the output

accordingly.

- After applying the control in the previous question, if the system still has overshoot, which PID term(s) will you apply and why?
  A: If the system has significant overshoot, the D control is utilized. D control acts as a damper, which tries to reduce the rate of change of error. The major reason for overshoot is that the error has some 'inertia' when it is reaching the target value.