Individual Lab Report

Name:	Dung-Han Lee
Team Name:	Team E Wholesome Robotics
Team members:	Aaditya, Aman, John, Hillel
ILR Number:	01
Date:	February 11 2019

February 22, 2019

1 Sensor and Motor Control Lab

1.1 Reading Ultrasonic Distance

My role for the Sensors and Motor Control Lab was to implement the stepper motor and ultrasonic sensor. My deliverable to the team are 2 pack of functions. The first pack has one function, get_ultrasound_ dist(), reads the analog (voltage) from the ultrasonic sensor pin and convert that to distance. More specifically, the relationship was established through measuring the voltage with known distance to obstacle and calibration.

1.2 Smoothing Readings

Also, since ultrasonic wave sometimes travels into longer distance, it can lead to undesirable, sudden jump in sensor readings. This is bad since we are doing stepper motor control directly base on the ultrasonic sensor reading in some mode. To resolve this, I implemented an average smoothing algorithm with a window of size 10 that would give value according to the average number in the past 10 numbers. The result is a significant smoother curve.

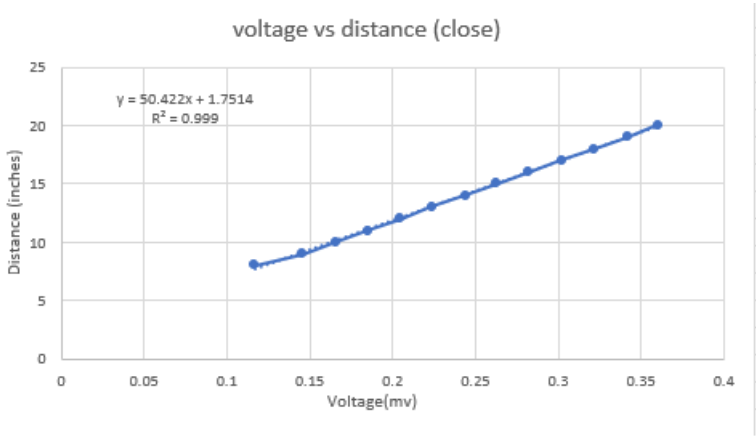

Figure 1: transfer function of ultrasonic

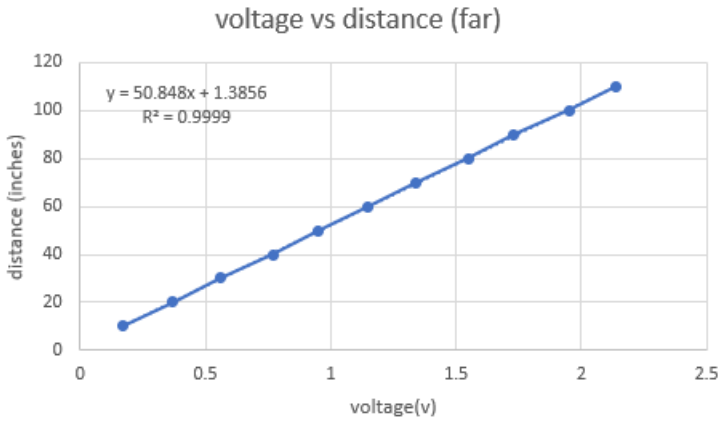

Figure 2: transfer function of ultrasonic (far)

1.2.1 get_ultrasound_dist()

```
#include "ultrasound.h"

float get_ultrasound_dist() {
  float sensor, inches, voltage;
  sensor = analogRead(A0);
  //  according to datasheet: voltage divide in 512 cells
  voltage = sensor * (5.0 / 512.0);
  inches = 50.7*voltage+1.63;  //  calibrated from measurement data
  inches = (inches < 9) ? 9 : inches;  //  dead zone handling

  // update vals in window for smoothing
  total = total - readings[readIndex];  //  get rid of rightmost
     element
  readings[readIndex] = inches;   //  assign new val to rightmost
     element
  total = total + readings[readIndex];  //  increment total with new
     val
  readIndex = readIndex + 1;
  if (readIndex >= numReadings) { readIndex = 0; }
  average = total / numReadings;

  return average;
}
```

1.3 Turn Stepper Motor

As for the second pack of functions, I first implemented a advance(int steps) function that would drive the shaft to go counter-clock-wise or clock-wise given some "steps". It's worth noting that 1600 steps corresponds to 1 revolution in our case, although internet resources indicates that Mercury stepper motor should have a 200 step to 1 revolution ratio.

1.3.1 init_stepper_motor()

```
void init_stepper_motor()  {
  pinMode(STEP, OUTPUT);
  pinMode(DIR, OUTPUT);
  digitalWrite(STEP, LOW);
  digitalWrite(DIR, LOW);
  }
```

1.3.2 advance(int steps)

```
void advance(int steps)
{
  if(steps>0){
    digitalWrite(DIR, LOW);  // Set clockwise direction
    stride = 0.225; // 360 degree = 1600 steps thus 1 step = 0.225
        degree
  }
  else{
    digitalWrite(DIR, HIGH);  // Set counter-clockwise direction.
    steps=abs(steps);
    stride = -0.225;
  }
  //each square pulse moves shaft by 0.225 degree
  for (int i = 0; i<steps; i++){
    digitalWrite(STEP, HIGH);
    delayMicroseconds(700);
    digitalWrite(STEP, LOW);
    delayMicroseconds(700);
    stepper_status += stride;
  }
}
```

1.4 Set Desire Goal Angle

In order to achieve control base on ultrasonic readings, I also implemented a function set_goal(float ultra_dist), which maps distance from get_ultrasound_ dist() to desire goal angles in degrees. The range is [9,19] inches to [0,360] degrees. Why [9,19]? less than 9 is dead zone for ultrasonic, more than 19 is hard to get a steady reading. See Figure 1 and Figure 2 for transfer function.

1.4.1 set_goal(float ultra_dist)

```
float set_goal(float ultra_dist){
  //  input: distance in inches [9,19]
  //  output: degrees [0,360]
  float goal = 36*(ultra_dist-9);
  //change nothing if distance is too far
  if(ultra_dist >19){
    goal =  stepper_status;
  }
  return goal;
}
```

1.4.2 Interfacing and Controlling

Both stepper_with_ultrasonic() and void GUI_control(int degree, bool CW) functions take care of the actual interfacing with GUI. The former would control steppers' angle based on ultrasonic distance while the latter will take command from the GUI user directly.

1.5 stepper_with_ultrasonic()

```
void stepper_with_ultrasonic(){
  //  1 degree = 4.44 steps i.e 1 rev = 1600 steps
  float diff = 4.44*(set_goal(ultrasonic())-stepper_status);
  advance(diff);
}
```

1.5.1 GUI_control(int degree, bool CW)

```
void GUI_control(int degree, bool CW)
{
  if(CW>0){
    digitalWrite(DIR, LOW); // Set clockwise direction.
    stride = 0.225;
  }
  else{
    digitalWrite(DIR, HIGH);          // Set counter-clockwise direction
        .
    stride = -0.225;
  }

  for (int i = 0; i<4.44*degree; i++){ // Rotate n steps.
    digitalWrite(STEP, HIGH);
    delayMicroseconds(200);            // Minimum STEP Pulse Width 1.0uS
    digitalWrite(STEP, LOW);          // Step pulse (rising edge).
    delayMicroseconds(200);
    stepper_status += stride;
    if (stepper_status>359) stepper_status = 0;
    else if (stepper_status<0) stepper_status = 359;
  }
}
```

2 Individual Progress

2.1 Getting LiDAR work

For the Wholesome Robotics(Agricultural robot) I have been mostly focusing on getting sensor done and deep learning for visual analytic in plant health. In order to navigate in the field reliably I sourced an VLP-16 puck LiDAR sensor, fixed issues in communication with computer (recognizing the sensor), solving environment parameters (ROS had conflict wit Anoconda). As a result I successfully connected it to the ROS environment.

2.2 Mask-RCNN and Labeling

As for the visual analytic, I modified the original plan: instead of using cascaded system of faster-RCNN and conditional - GAN, I decided to go for Mask-RCNN. The reason being it provides the desirable high level features (area of disease/holes and their absolute numbers) from the farmers at once, so I only need to train 1 neural net instead of 2. Also, in order to provide training data, I manually labeled 500 bit-mask images for the training set, and trained the neural net on tensorflow. See Figure 3 for reference.

2.2.1 Initial results

The first training result achieved 60% accuracy in detecting holes and 55% in disease-like features. Also the model is being a bit conservative as we have far more false negative than false positive.

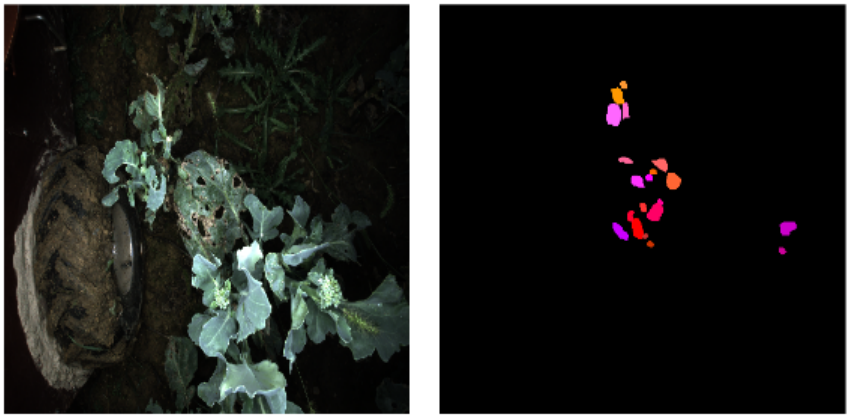

Figure 3: sample data and bit-mask label

Figure 4: Sample output

Holes Accuracy: 61.2%		
	Label Positive	Label Negative
Actual Positive	True Positive	False Negative
	30	16
Actual Negative	False Positive	X
	3	

Disease Accuracy: 55%		
	Label Positive	Label Negative
Actual Positive	True Positive	False Negative
	14	8
Actual Negative	False Positive	X
	5	

Figure 5: Confusion Mat of 2 categories

3 Challenges

3.1 Motor and Sensors Lab

The challenge I had was getting reliable and steady value from the ultrasonic sensor, which in effect turned into choosing a best range that works for our application and an average smoothing window. Also, I need to implement a stepper motor control code with minimum blocking, so that my code does not affect the functionality of other team members' works. This was achieved by setting higher motor speed and control the motor step by step.

Also we received a broken driver at the first place, and I tried all possible ways to nail down the root cause: testing my code on other team's hardware, switching specific hardware components and examine all the wiring etc. Identifying the root cause from a embedded system is very challenging as there are multiple pieces that can go wrong: wiring, soldering, software, component malfunction. The lesson is try to test the system piece by piece, and isolate single component during the development.

3.2 Individual Progress

The biggest challenge I had was dealing with big amount of data efficiently. To this end, I wrote auto-annotating codes to generate necessary annotations for the neural net. To be more specific, given a bit-mask image (Figure 3) the code would assign category (holes or disease) based on the color, and generate a set of coordinates that describe the contour of the patches. Furthermore, I developed standardized process for me and teammate to follow in labeling data in the future.

4 Teamwork Distribution

4.1 sensor motor lab

1. John: integration and GUI
2. Aaditya: DC motor and infrared sensor
3. Hillel: servo motor and slot sensor
4. Dung-Han: stepper and ultrasonic
5. Aman: DC motor and temperature

4.2 MRSD Project

1. John: Localization in the field
2. Aaditya: Building map base on sensory data
3. Hillel: Assemble machine part, design end-effector for weeding
4. Dung-Han: Monitor plant health base on deep learning
5. Aman: Design machine parts, setup motors

5 Future Plan

Since the stereo camera take pictures from different height, two holes having same pixel area may actually have very different size in reality. To workaround this issue, I need a dimensionless metric. To achieve that, I plan to change the metric from area of holes/disease to relative area: $\frac{area\_holes}{area\_leaves}$. To this end, I will look into depth disparity functions, to segment leaves out from the ground.

Meanwhile, I will also be tweaking the parameters in neural net to hit 80% accuracy in both categories.