Principles of Engineering: Lab 1

Due on Monday, September 23, 2013

Justin Poh and Sophia Seitz

1 Introduction

For this lab, our task was to design and build a LIDAR out of two servos, an arduino, and an infrared range finder.

In order to achieve this, we split the project into two main components. Firstly, we needed a mechanical design that would support sensor such that it was capable of panning (left-right movement) and tilting (up-down movement). This would allow our sensor to scan any point in 3D space within the 180 degree viewing angle it had in front of it.

For our software, we decided to give python ultimate control over the process because we wanted a one-button-push implementation such that once the python programme was run, the programme would control the arduino to do what it needed to do and simply produce the plot on screen at the end of the programme.

2 Mechanical Design

Since we decided to focus less on the mechanical design aspect and more on the software aspect, there will not be too much detail presented here on the mechanical design.

Essentially we knew we needed a pan and tilt head that would ensure that the sensor panned and tilted while remaining centred along the the axes of rotation. Hence, we chose a cantilever design consisting of the panning servo mounted inside a box. An L-bracket cantilever was used to support the tilt servo and the sensor was mounted to a piece of wood that was glued to the tilt servo to achieve tilting ability.

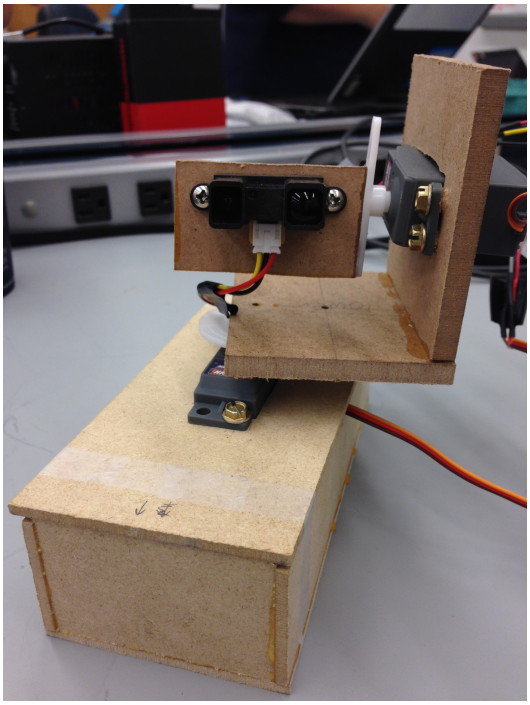

Figure 1: Picture of Mechanical Design

3 Connection Schematic

Below is the schematic showing the pins that we connected the two servos and the infrared range finder to:

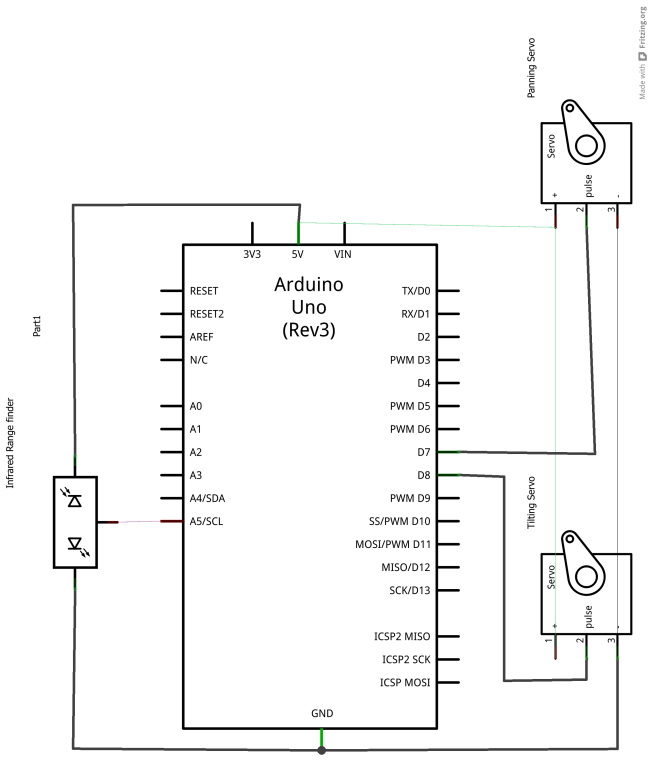

Figure 2: Schematic of the connections of the servos and infrared range finder

4 Sensor Testing

The infrared range finder outputs a voltage that corresponds to the distance it detects. However, according to the data sheet, that relationship is not linear. Hence we had to calibrate our own sensors and figure out that relationship experimentally.

In order to do this, we taped a sheet of paper to the wall and placed a tape measure on the ground in order to measure the actual distance our sensor was from the wall. We then proceeded to collect the sensor readings output from the arduino. Although the sensor output is in volts, the arduino converts that to a scale of 0 - 1023 thus the readings we obtain are between 0 and 1023 in value.

Once we had obtained sensor readings from the sensor for a distance range of 20cm - 150cm in steps of 10, we then produced a graph of raw sensor reading against distance:
As shown in the figure above, we used the curve fitting tool to find the equation of the line that would fit the data. We eventually found out that the equation that related our raw sensor output to distance was:

$$distance = 25732.835((sensorreading^{-1.13146}) \quad (1)$$

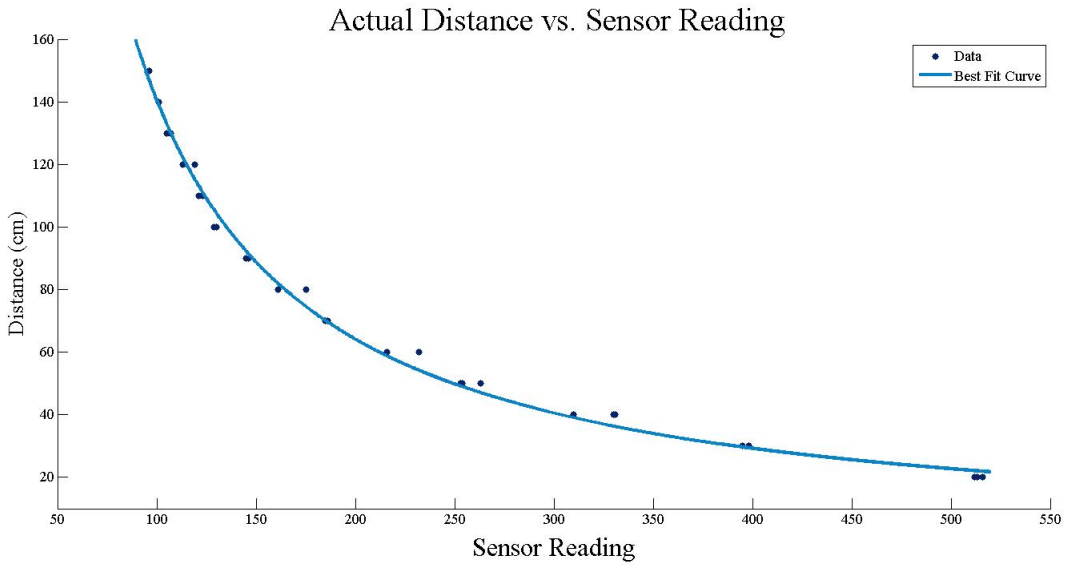

Figure 3: Graph of Actual Distance against Sensor Reading with the best fit curve plotted as well

With this equation now known, we programmed it into the arduino to convert the raw sensor readings into distance. We then repeated the data collection method above but recorded calculated distance data instead of raw sensor data. We then plotted actual distance against distance calculated by the arduino: As can be

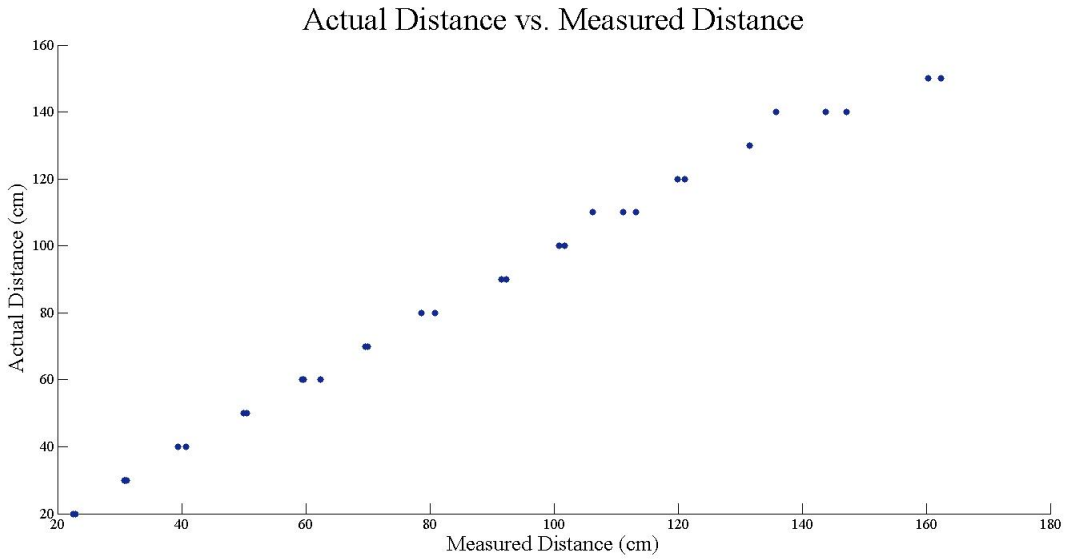

Figure 4: Graph of Actual Distance against calculated distance plotted as a scatter plot

seen in figure 3, the data is approximately linear and the scatter of the data suggests that the sensor readings are pretty accurate. However, the further the actual distance, the larger the scatter and the less accurate the readings.

5 One Servo Sensor

In order to avoid building multiple mechanisms for one servo and then two servos, we designed a mechanism for two servos. In order to take a 2D plot, we set the vertical angle to 90 degrees in order to point the sensor straight ahead. We then perform a full sweep of the horizontal angles once through.

A picture of the sensor taking a horizontal sweep is shown below:

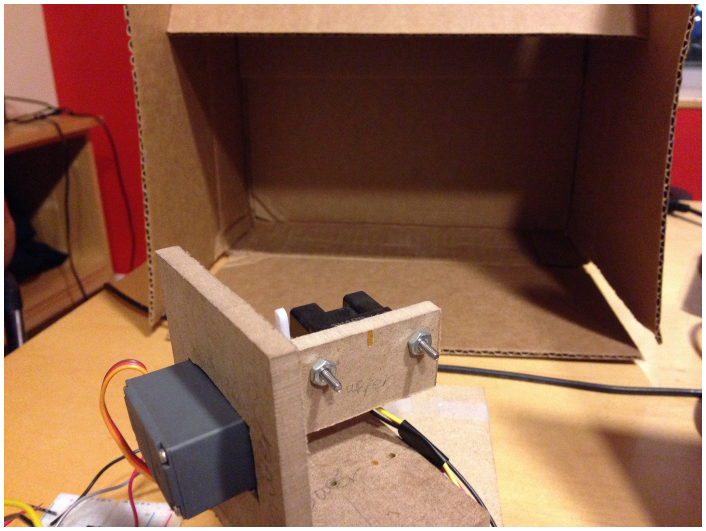

Figure 5: The sensor taking a 2-D scan of the inside of a cardboard box

The output of that scan is now shown below:

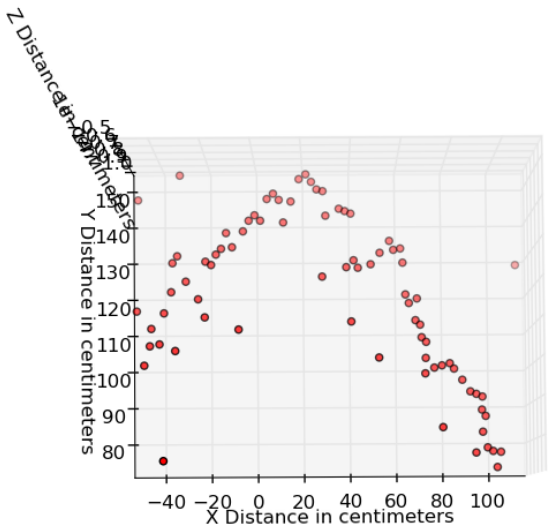

Figure 6: 2-D scan output of the sensor

6 Two Servo Sensor

For this part, we used the same mechanical set up as before. However, we made it perform both a horizontal and vertical sweep instead of just a horizontal sweep. A picture of the sensor taking a 3D sweep of the diorama is shown below:

Figure 7: The LIDAR taking a 3-D scan of the diorama

The 3D output of our python programme is shown on the next page:

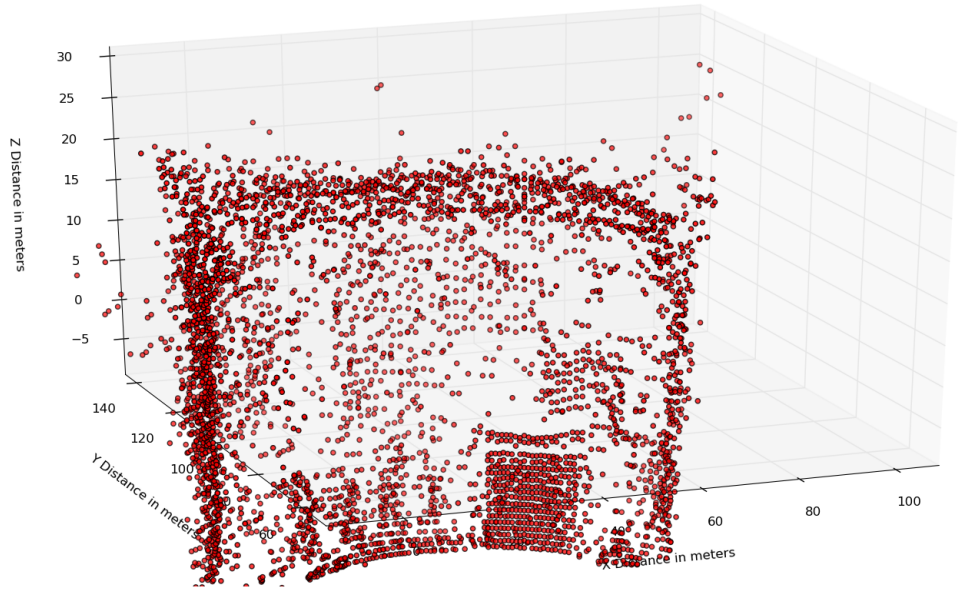

Figure 8: 3D plot of the Diorama used for this class; It has been zoomed in for a better view

The 3D plot looks similar to the diorama, especially towards the front. Both the box and roll in the front of the diorama are identifiable in the picture. However, the further back in the diorama, the less accurate our representation is, hence the lack of detail toward the back of the diarama. In addition, the bottom right corner of the diorama is not very well depicted. This is because we set our LIDAR up at an angle where the box in the front of the diorama blocks that corner.

7 Code

In essence, our code has two main modules to it: one written in python and one written in arduino code.

The arduino code is written to only contain modes of operation. The available modes of operation are as follows:

1. Mode '1': Move panning servo to the right by 1 degree
2. Mode '2': Move panning servo to the left by 1 degree
3. Mode '3': Move tilting servo up by 1 degree
4. Mode '4': Move tilting servo down by 1 degree
5. Mode '5': Move the panning servo to the home position as defined at the beginning of the code

6. Mode '6': Move the tilting servo to the home position as defined at the beginning of the code
7. Mode '7': Take a reading from the infrared range finder and return the distance measured to the serial port for python to read
8. Mode '8': Put the current vertical and horizontal angle together into a string to output to the serial port for python to read

Hence, the arduino simply waits for a command to be passed to it from python through the serial port. Once it receives a command in the form of a mode number, it executes the code corresponding to the mode number received and carries out the action.

The python code is more complex. The python code has full control of the entire operation of scanning the 3D space, collecting the data, performing the trigonometry necessary to convert spherical co-ordinates to cartesian co-ordinates and then finally producing the 3-D plot.

The communication between the python code and the arduino is done entirely through the serial port using the python serial module. The act of scanning the space is done by iterating through a range of horizontal and vertical angles, both in increments of 1 degree. The sensor begins at the pre-defined home position. In the case of the diorama used for this lab, the home position is defined as 135 degrees with respect to the horizontal axis and 74 degrees with respect to the vertical. The reference for these angles is shown below:

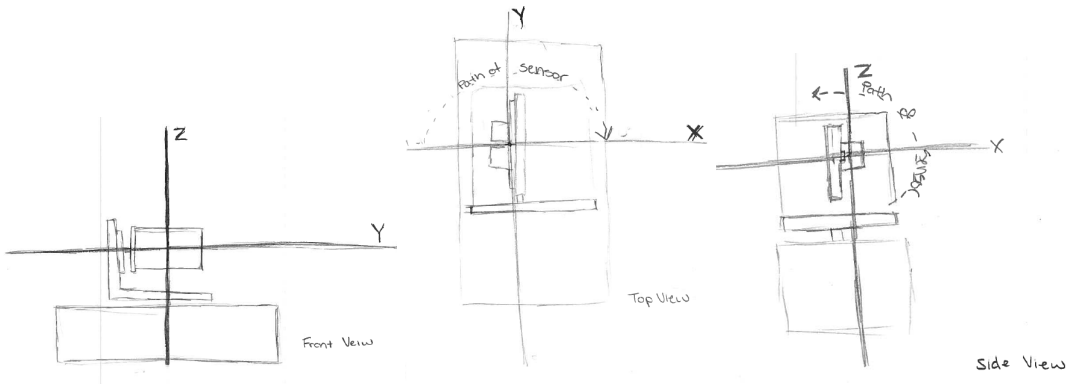

Figure 9: Diagrams showing the reference used to determine the angle of the sensor when converting from spherical to cartesian co-ordinates

From the home position, for each horizontal angle, the python code performs the following steps:

1. Instructs the arduino to take a reading and return the distance measured by the sensor through the serial port
2. Instructs the arduino to read the current vertical and horizontal angle and concatenate it into one string. Then return it through the serial port for python to read
3. Feed the distance reading and the vertical and horizontal angles to a new lidar object as defined in python (see appendix for the full class definition). The initialized lidar object will then take those numbers and convert them from spherical co-ordinates to cartesian co-ordinates. It then stores the cartesian co-ordinates as instance variables.
4. Appends the newly created lidar object to the data list to retrieve later for plotting.

5. Move the sensor 1 degree to the right

Once the arduino has completed a full horizontal sweep at a given tilt angle, it then resets back to the horizontal home position and increases the vertical angle by 1 degree. It then proceeds to sweep through the horizontal angles again as described in the above list.

Once the full 3D sweep of the space is complete, the python code then reads each lidar object in the data list and produces a 3D scatter plot of the data obtained from the scan as a matplotlib figure.

8 Further Testing

To test some more, we took two additional scans; one inside and one outside. Our inside scan was of an armchair. and looked like this.

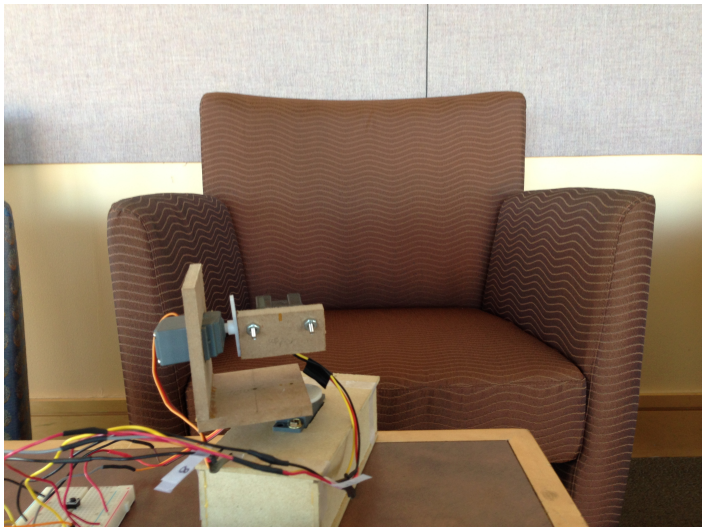

Figure 10: A picture of the armchair we scanned.

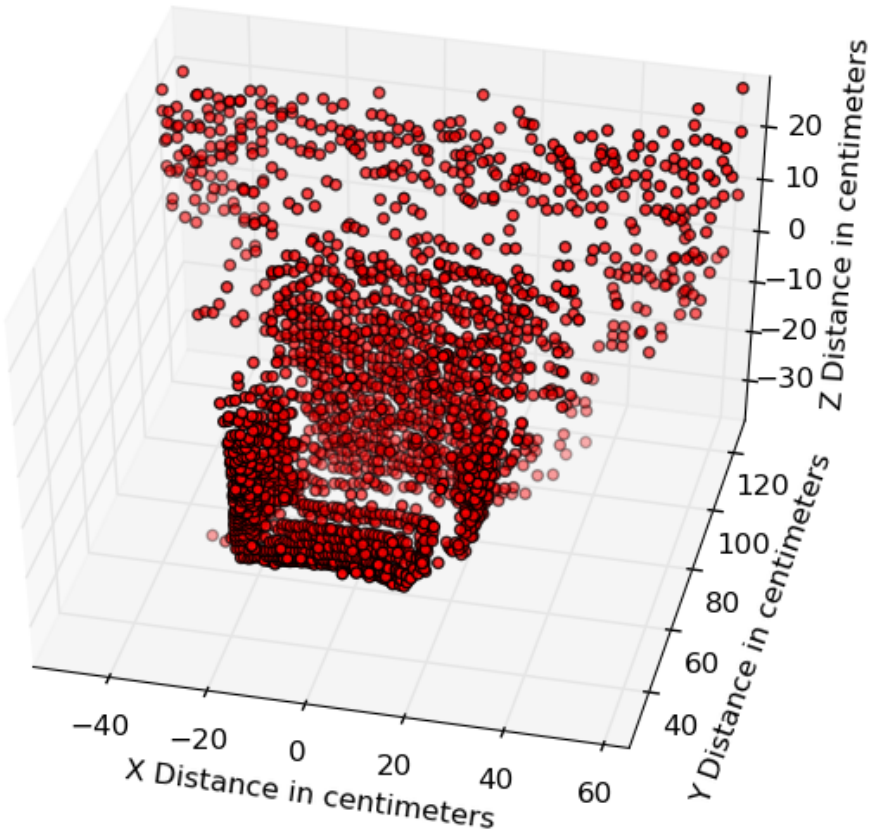

Figure 11: A 3D plot of the armchair we scanned.

Next, we went outside and scanned a cone against a wall.

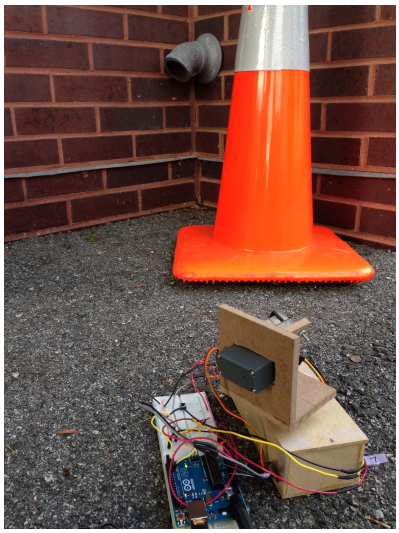

Figure 12: A picture of the cone we scanned.

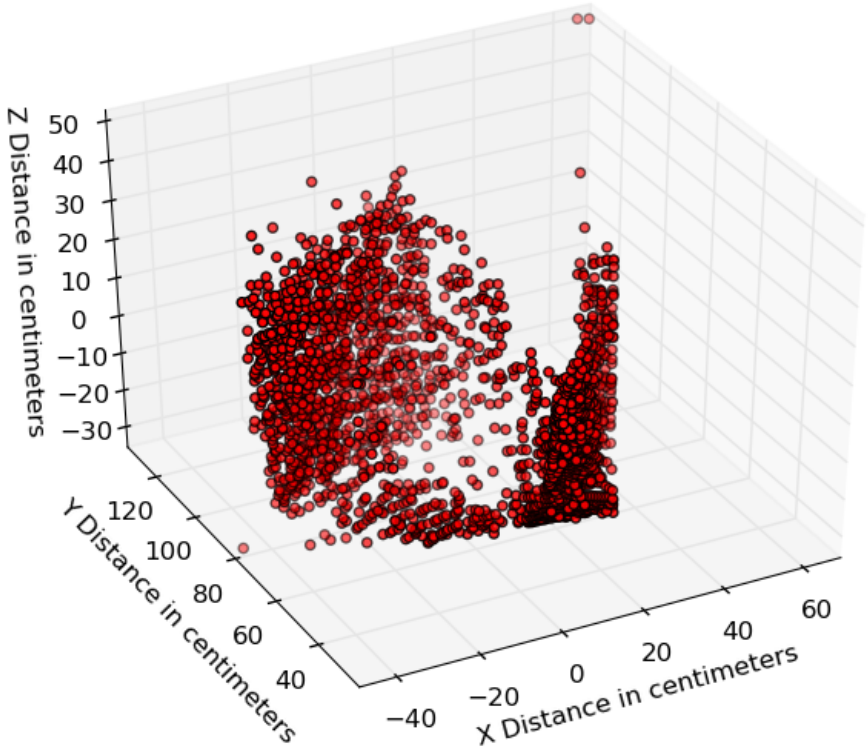

Figure 13: A 3D plot of the cone we scanned.

```
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import serial
import time
from lidar_classdef import *

ser=serial.Serial('/dev/ttyACM0', 9600) #Defines the serial port to use

data=[] #List that stores the data from the arduino

horzBegin=115 #Sets the home horizontal angle
horzEnd=60 #Sets the end horizontal angle

vertBegin=64 #sets the home vertical angle
vertEnd=103 #Sets the end vertical angle

print "Waiting for arduino to be ready....." #Indicates that the programme is
waiting for the arduino to initialize

time.sleep(1) #Waits for the arduino to initialize

print "Programme beginning now" #Indicates that the python programme is running

#Moves the servo to the starting position

ser.write('5') #Sets the servo over to its home position on the left
time.sleep(0.5) #Waits half a second to ensure the arduino is ready for next command
ser.write('6') #Sets the servo to it's starting vertical angle

horzAngle=range(horzBegin, horzEnd, -1) #creates the horizontal range of angles that
the servo should sweep through
vertAngle=range(vertBegin,vertEnd,1) #Creates the vertical range of angles that the
servo should sweep through

for eachVerticalAngle in vertAngle:


        for eachHorizontalAngle in horzAngle:

                ser.write('7') #Tells the arduino to send back a distance reading

                distance_response=ser.readline() #Receives the distance reading from
the arduino
                cleanReading=distance_response[0:-2] #Removes the last 2 characters
("\r\n") from the arduino output
                distance=float(cleanReading) #Type-casts it as an float so that it
can be plotted

                ser.write('8') #Tells the arduino to send back the current vertical
and horizontal angles

                angle_response=ser.readline() #Receives the vertical and horizontal
angles from the arduino as a string

                cleanReading=angle_response[0:-2] #Removes the last 2 characters
("\r\n") from the arduino output
                seperator_index=cleanReading.index(',') #Figures out at what index
the comma appears (which is how we delineate vertical and horizontal angle)
                vertical_angle=cleanReading[0:seperator_index] #Retreives the
vertical angle from the string that is received from the arduino
                vertical_angle=int(vertical_angle) #Converts the angle from string
to integer
```

```
                horizontal_angle=cleanReading[(seperator_index+1): ] #Retreives the
horizontal angle from the string received from the arduino
                horizontal_angle=int(horizontal_angle) #Converts the angle from a
string to a integer

                processed_full_data_holder=lidar(horizontal_angle, vertical_angle,
distance) #Passes the distance reading and angles into a lidar object (defined
seperately)

                data.append(processed_full_data_holder) #Appends the new lidar
object to the data list

                ser.write('1') #Moves the servo one degree to the right after taking
a reading

        ser.write('3') #At the end of the horizontal sweep, increase the vertical
angle by 1 degree
        time.sleep(0.5) #Waits to allow the arduino to execute the command and be
ready for another

        ser.write('5') #Resets the arduino back to the left to perform a full
horizontal sweep
        time.sleep(0.5) #Waits for the arduino to execute the command and be ready
for another

#def randrange(n, vmin, vmax):
#    return (vmax-vmin)*np.random.rand(n) + vmin

fig = plt.figure() #Creates a new pyplot figure
ax = fig.add_subplot(111, projection='3d') #adds a 3d plot axis to it

xs = [] #Creates a new list for x axis values
ys = [] #Creates a new list for y axis values
zs = [] #Creates a new list for z axis values

for eachPoint in data:
        xs.append(eachPoint.x_pos) #Adds each x axis value to the x axis data list
        ys.append(eachPoint.y_pos) #Adds each y axis value to the y axis data list
        zs.append(eachPoint.z_pos) #Adds each z axis value to the z axis data list

ax.scatter(xs,ys,zs, c ='r', marker = 'o') #Creates a scatter plot of all data

ax.set_xlabel('X Distance in centimeters') #X axis label
ax.set_ylabel('Y Distance in centimeters') #Y axis label
ax.set_zlabel('Z Distance in centimeters') #Z axis label

plt.show() #show the pyplot figure
```

```
import math #Imports math to do trig calculations

class lidar: #Definition of a new class called lidar
        def __init__(self, input_horz_ang=0, input_vert_ang=0, radius=0):
                horz_ang=math.radians(input_horz_ang) #converts the horizontal angle
from degrees to radians
                vert_ang=math.radians(input_vert_ang) #converts the vertical angle
from degrees to radians

                self.z_pos = -radius*math.cos(vert_ang) #Calculates the length of
the position vector projected on the z-axis
                self.y_pos = radius*math.sin(vert_ang)*math.sin(horz_ang)
#Calculates the length of the position vector projected onto the y axis
                self.x_pos = radius*math.sin(vert_ang)*math.cos(horz_ang)
#Calculates the length of the position vector projected onto the x axis

        def display_data(self): #Instance method to display the data encapsulated in
the lidar object

                print "X Position = ", self.x_pos
                print "Y Position = ", self.y_pos
                print "Z Position = ", self.z_pos
```

```
#include <Servo.h>
//variable that receives the command from the python code
int  incomingMessage=0;
//Sets the left side home position
int  leftHomePosition=115;
//Keeps track of the horizontal angle
int horizontalAngleTrack=leftHomePosition;

int downHomePosition=90;
//Keeps track of the vertical angle
int verticalAngleTrack=downHomePosition;
//Sets the increment of the horizontal angle every time the
//move command is called
int  horizontalIncrement=1;
//Sets the increment of the vertical angle every time the move
//command is called
int verticalIncrement=1;

//Declares a servo called horizontal_servo
Servo horizontalServo;
//Declares a servo called vertical_servo
Servo  verticalServo;

void setup()                 {
   //Sets up the sensor in the analog pin A5
  pinMode(A5, INPUT);
  //Initializes the serial port
  Serial.begin(9600);
  //Declares the pin that the servo controlling horizontal
  //angle is connected to
 horizontalServo.attach(7);
  //Declares the pin that the servo controlling vertical
  //angle is connected to
 verticalServo.attach(8);
}

// the loop routine runs over and over again forever:
void loop() {

  //Sends data only when it is received
  if (Serial.available()>0){
    //Waits for a message from the serial port (in this case,
    // from python code)
   incomingMessage=Serial.read();

    //Moves to the right
```

```
if (incomingMessage=='1'){
  //Decrements the horizontal angle by the specified
  //increment to
  //move it to the right
 horizontalAngleTrack=horizontalAngleTrack-horizontalIncrement;
  //writes the new angle to the servo to move it there
  horizontalServo.write(horizontalAngleTrack);
}

//Moves to the left
if (incomingMessage=='2'){
  //Increases the horizontal angle by the specified
  //increment to move
  //it to the left
horizontalAngleTrack=horizontalAngleTrack+horizontalIncrement;
 //writes the new angle to the servo to move it there
  horizontalServo.write(horizontalAngleTrack);
}

//Moves the servo up
if (incomingMessage=='3'){
  //Increases the vertical angle by the specified
  //increment to move it up
 verticalAngleTrack=verticalAngleTrack+verticalIncrement;
 //Writes that angle to the servo to move it there
  verticalServo.write(verticalAngleTrack);
}

//Moves the servo down
if (incomingMessage=='4'){
  //Decreases the vertical angle by the specified increment
  //to move it down
  verticalAngleTrack=verticalAngleTrack-verticalIncrement;
  //Writes that angle to the servo to move it there
  verticalServo.write(verticalAngleTrack);
}

//Moves sensor all the way left
if (incomingMessage=='5'){
  //Sets the horizontal angle to the left_home_position
 horizontalAngleTrack=leftHomePosition;
 //writes that angle to the servo to move it there
  horizontalServo.write(horizontalAngleTrack);
}

//Moves sensor to the lowest possible vertical angle setting
```

```
    if (incomingMessage=='6'){
      //Sets the vertical angle to the down_home_position
      verticalAngleTrack=downHomePosition;
      //Writes that angle to the servo to move it there
      verticalServo.write(verticalAngleTrack);
    }

    //Takes a reading from the sensor
    if (incomingMessage=='7'){
      //Takes a reading from the sensor connected to port A5
      int reading =analogRead(A5);
      //Uses the model we derived to calculate distance from
      //the sensor reading obtained
      float distance=25732.834527*pow(reading,-1.1314581);
      //Prints that distance to the serial port for the python
      //code to receive
      Serial.println(distance);
    }

    //Sends back the current horizontal and vertical angles
    if (incomingMessage=='8'){
      //Packs the current vertical and horizontal angle into a
      //concatenated string to print to serial port
      String anglePacked=String(verticalAngleTrack)+','+String(horizontalAngleT
      Serial.println(anglePacked);//Prints that concatenated
      //string to the serial port for python to receive
    }
}
}
```