809T Assignment 4

Autonomous Robotics

Prasanna Thirukudanthai Raghavan
UID: 118287546

Department of Robotics
University of Maryland
United States

Contents

Chapter 1

Ultrasonic Sensor:

1.1 Code: range01.py

```
import time
from unicodedata import name
import RPi.GPIO as GPIO
import numpy as np
import cv2
import imutils
import os
```

```
# -----------------------
# Define some functions
# -----------------------

def measure():
  # This function measures a distance
  GPIO.output(GPIO_TRIGGER, True)
  time.sleep(0.00001)
  GPIO.output(GPIO_TRIGGER, False)
  start = time.time()

  while GPIO.input(GPIO_ECHO)==0:
    start = time.time()

  while GPIO.input(GPIO_ECHO)==1:
    stop = time.time()

  elapsed = stop-start
  distance = (elapsed * 34300)/2

  return distance

def measure_average(n):
  # This function takes average of n measurements
  distances = []
  for i in range(n):
      distances.append(measure())
      time.sleep(0.1)
  average = sum(distances)/n
  return average

# -----------------------
# Main Script
# -----------------------

# Use BCM GPIO references
# instead of physical pin numbers
GPIO.setmode(GPIO.BCM)

# Define GPIO to use on Pi
GPIO_TRIGGER = 23
GPIO_ECHO    = 24

print("Ultrasonic Measurement")

# Set pins as output and input
GPIO.setup(GPIO_TRIGGER,GPIO.OUT)  # Trigger
GPIO.setup(GPIO_ECHO,GPIO.IN)      # Echo

# Set trigger to False (Low)
GPIO.output(GPIO_TRIGGER, False)

# Wrap main content in a try block so we can
# catch the user pressing CTRL-C and run the
# GPIO cleanup function. This will also prevent
# the user seeing lots of unnecessary error
# messages.
```

```
65
66 distance = measure_average(10)
67 print("Distance :" ,distance)
68 time.sleep(1)
69
70 dstr = 'Distance:'+str(round(distance,3))
71 #Record Image using RaspiStill
72 os.system('raspistill -w 640 -h 480 -o lecture4inclass.jpg')
73 im = cv2.imread("lecture4inclass.jpg", 1)
74 font = cv2.FONT_HERSHEY_SIMPLEX
75 cv2.putText(im, dstr, (50,50), font, 1, (0, 0, 255), 2, cv2.LINE_AA)
76 cv2.imwrite('lecture4inclass.jpg', im)
77
78
79 GPIO.cleanup()
```

Chapter 2

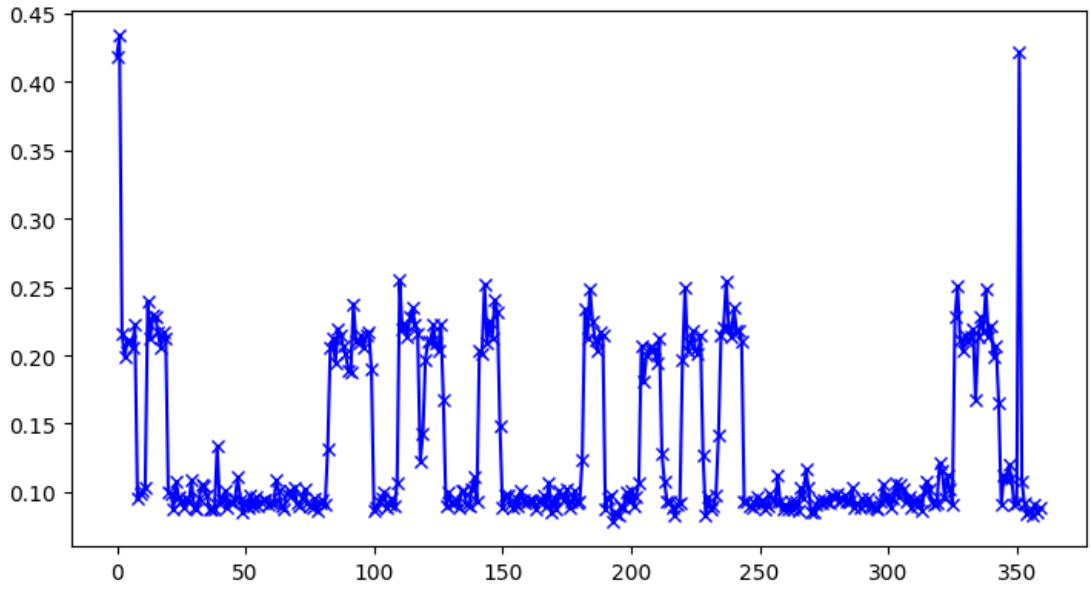
Arrow Tracking and Orientation

2.1 Video Link:

The Link: https://youtu.be/C1OmTF1YxAA
All the requirements were checked prior to making this video.

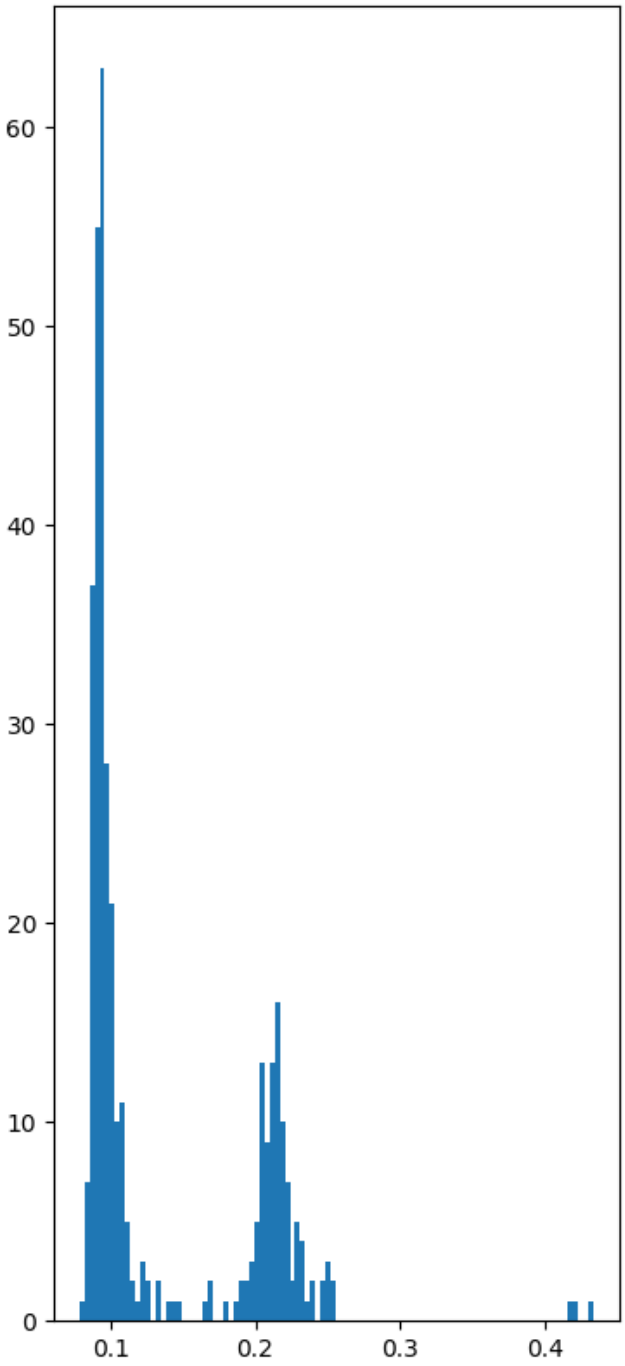
2.2 Tracking Performance:

We can see from the video above that the Object despite being solitarily in the frame due to lighting and video coloring, the arrow orientation often points in the opposite direction due to the code having an else condition. The perfect orientation tracking happens when the corners are aligned along the axis and we present it in the center of the screen.



As we can see except the first measurement, The rest of the measurements are faster and hence we can conclude that the pi processes data faster after the first iteration of

the loop.
The general performance of the pi depends upon the processing power of the pi and the maximum of processing from the camera feed is done per frame and that is shown in the performance graph.


60
50
40
30
20
10
0
0.1
0.2
0.3
0.4

Chapter 3

Code:

3.1 PiCam Code:

```
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import numpy as np
import time
import datetime
import cv2
import imutils
from imutils.video import VideoStream
import datetime
from datetime import datetime,timedelta
# initialize the camera and grab a reference to the raw camera
    capture
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640, 480))

#Define the codec
today = time.strftime("%Y%m%d-%H%M%S")
fps_out = 32
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter(today + ".avi", fourcc, fps_out, (640, 480))

# allow the camera to warmup
time.sleep(0.1)
# capture frames from the camera
for frame in camera.capture_continuous(rawCapture, format="bgr",
    use_video_port=True):
     # grab the raw NumPy array representing the image, then
    initialize the timestamp
     start = datetime.now()
     # and occupied/unoccupied text
     image = frame.array
     hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

```
36
37      ## mask of green (36,25,25) ~ (86, 255,255)
38      mask = cv2.inRange(hsv, (29, 73, 99), (133, 255,208))
39      ## slice the green
40      imask = mask>0
41      green = np.zeros_like(image, np.uint8)
42      green[imask] = image[imask]
43      imgray = cv2.cvtColor(green,cv2.COLOR_BGR2GRAY)
44      ret,thresh = cv2.threshold(imgray,100,255,0)
45
46      blur = cv2.GaussianBlur(thresh,(15,15),cv2.BORDER_DEFAULT)
47      corners = cv2.goodFeaturesToTrack(blur,7,0.01,10)
48      if(isinstance(corners,np.ndarray)):
49          corners = np.int0(corners)
50          for i in corners:
51              x,y = i.ravel()
52              cv2.circle(image,(x,y),3,(0,0,255),-3)
53
54          xmax, ymax = (np.max(corners, axis = 0)).ravel()
55          xmin, ymin = (np.min(corners, axis = 0)).ravel()
56          font = cv2.FONT_HERSHEY_SIMPLEX
57          if( abs(xmax-xmin) > abs(ymax-ymin)):
58              if(np.count_nonzero(corners[:,0,0] == xmax) == 2):
59                  cv2.putText(image,"LEFT", (50,50), font, 1, (0, 255,
    0), 2, cv2.LINE_AA)
60              else:
61                  cv2.putText(image, "RIGHT", (50,50), font, 1, (0,
    255, 0), 2, cv2.LINE_AA)
62          else:
63              if(np.count_nonzero(corners[:,0,1] == ymax) == 2):
64                  cv2.putText(image, "UP", (50,50), font, 1, (0, 255,
    0), 2, cv2.LINE_AA)
65              else:
66                  cv2.putText(image, "DOWN", (50,50), font, 1, (0, 255,
     0), 2, cv2.LINE_AA)
67      # show the frame
68      cv2.imshow("Frame",image)
69      key = cv2.waitKey(1) & 0xFF
70
71      #save the frame to a file
72      out.write(image)
73
74      # clear the stream in preparation for the next frame
75      rawCapture.truncate(0)
76      # if the ‘q‘ key was pressed, break from the loop
77      if key == ord("q"):
78          break
79      stop = datetime.now()
80      # open .txt file to save data
81      f = open('hw4data.txt','a')
82      # print time to run through loop to the screen & save to file
83      now = stop - start
84      outstring = str(now.total_seconds()) + '\n'
85      f.write(outstring)
86      print(now.total_seconds())
```

3.2 Plotting Code:

```
import numpy as np
import matplotlib.pyplot as plt
data = np.loadtxt("hw4data.txt", dtype=float)
sample = data[0:360]
plt.plot(sample,'xb-')
plt.show()
plt.hist(sample,bins=100)
plt.show()
```