ENPM809T Assignment #4

Hamza Shah Khan

UID: 119483152

Email: hamzask@umd.edu

University of Maryland

Course code: ENPM809T Autonomous Robotics

Instructor : Steven E. Mitchell

Date : 2nd March 2023

Semester: Spring 2023

HW#4 Goals

HW#4 explores use of ultrasonic sensor with the Raspberry Pi, and the use of OpenCV to implement corner detection and orientation, including analysis of the Pi's hardware limitations.

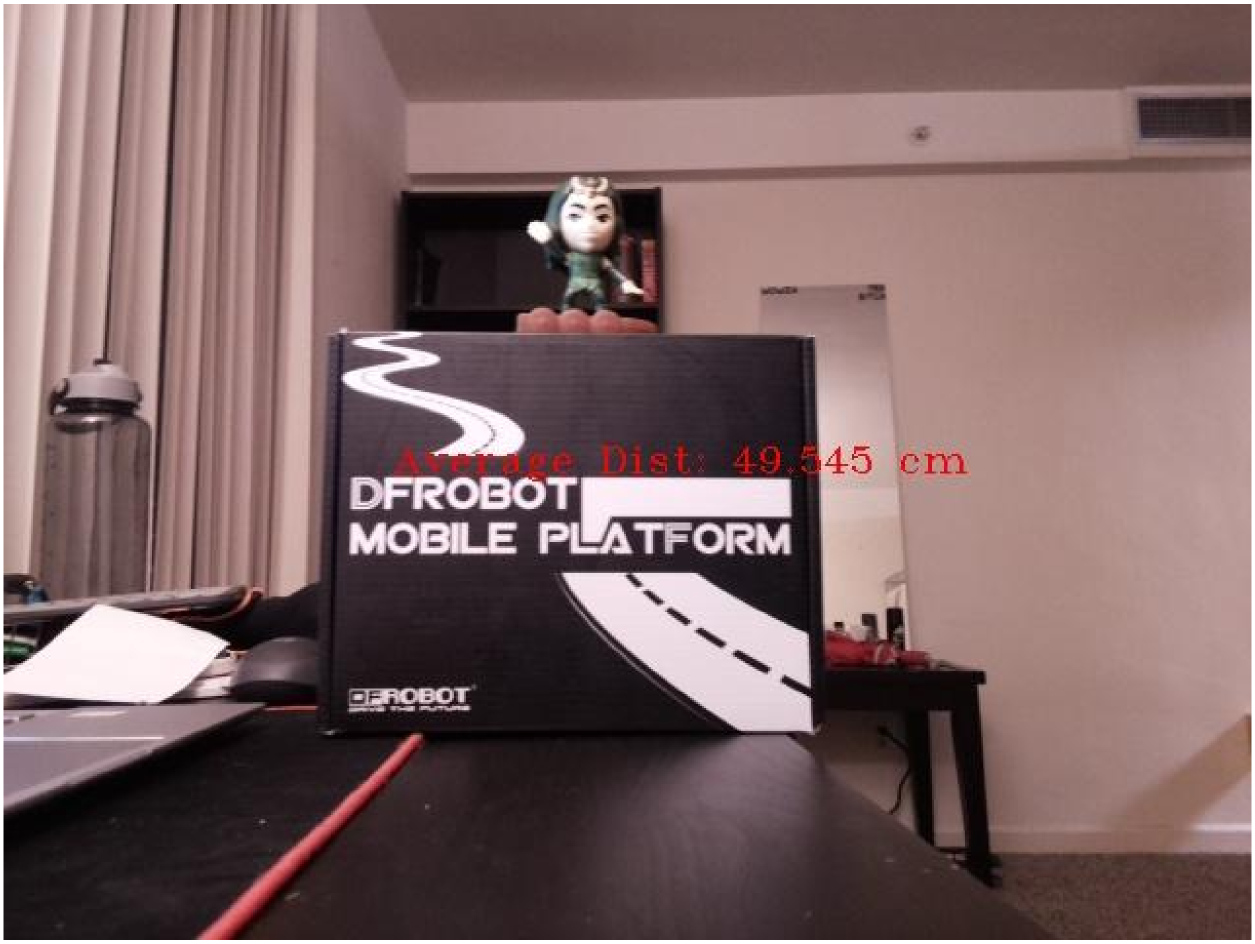
Question 1: (No submission)

Question 2: Using Distance Sensor

Python Code:

```
import numpy as np
import cv2
import imutils
import RPi.GPIO as gpio
import time
import os

gpio.setwarnings(False)

trig = 16
echo = 18
sum = 0

def distance():

        #GPIO pin setup
        gpio.setmode(gpio.BOARD)
        gpio.setup(trig, gpio.OUT)
        gpio.setup(echo, gpio.IN)

        gpio.output(trig, False)
        time.sleep(0.01)

        #Set trigger pin High for 1Khz
        gpio.output(trig, True)
        time.sleep(0.00001)
        gpio.output(trig, False)

        #Check Echo pin
        while gpio.input(echo) ==0:
                pulse_start = time.time()

        while gpio.input(echo) == 1:
                pulse_end = time.time()

        #Calculate time difference
        pulse_duration = pulse_end - pulse_start

        #Convert to distance
        distance = pulse_duration*17150
        distance = round(distance, 2)

        gpio.cleanup()
        return distance

#Capture Image
name = 'lecture_dist.jpg'
os.system('raspistill -w 640 -h 480 -o ' + name)

#Getting average distance of 10 measurements
for i in range(0,11):
        sum = sum  + distance()
avg_dist = round(sum/10,3)

#Printing distance
print('Avg Distance: ', distance(), 'cm')

#Adding distance text on image
font = cv2.FONT_HERSHEY_COMPLEX_SMALL
text = 'Average Dist: ' + str(avg_dist) + ' cm'
img = cv2.imread(name)
red  = (0,0,255)
cv2.putText(img, text,(200,240),font,1,red,1)

cv2.imshow('Avg Dist',img)
cv2.imwrite('Avg_dist.jpg', img)
cv2.waitKey(0)
```

Output:



```
Average Dist: 49.545 cm
```

Question 3: Arrow Orientation Detection

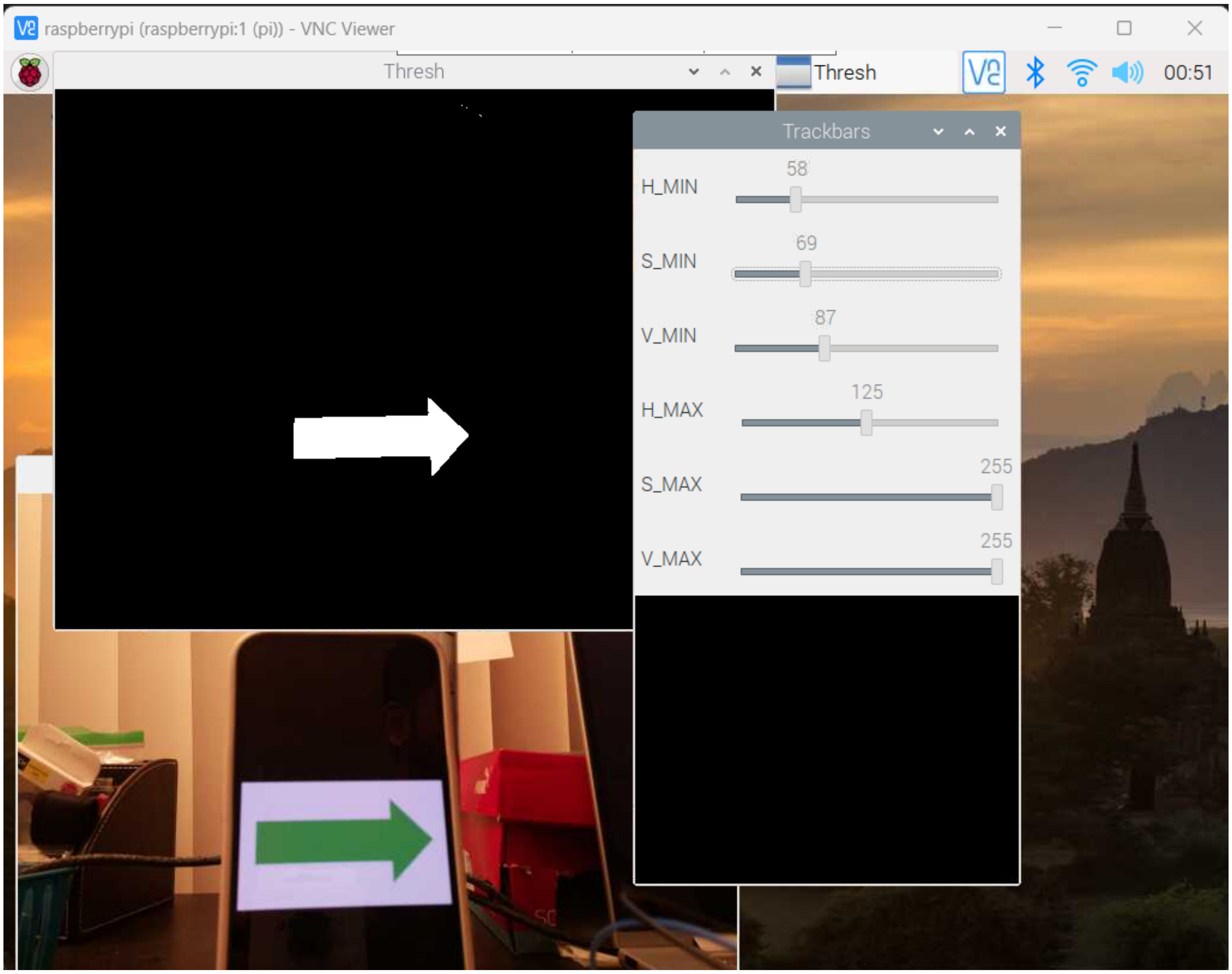
Step 1: Downloading Image

Step 2: HSV Masking

Using the colorpicker.py program, I calculated the upper and lower HSV values, which was used for masking the green arrow using cv2.inRange() function.

Code:

```
    ret,frame=cap.read()

    hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)

    low_green=np.array([60,60,225])
    high_green=np.array([95,255,255])

    mask = cv2.inRange(hsv, low_green,high_green)

    green= cv2.bitwise_and(frame,frame,mask=mask)
```



H_MIN 58

S_MIN 69

V_MIN 87

H_MAX 125

S_MAX 255

V_MAX 255

The HSV Values:

Lower HSV Values = [60,60,65]   Upper HSV Values = [90,255,255]

Step 3: Image Blurring

The masked image was blurred using the inbuilt cv2.GaussianBlur() function

Step 4: Corner Identification

I used the Shi-Tomashi Corner Detection algorithm via the cv2.goodFeaturesToTrack() function in OpenCV, which returns the coordinate values of all the detected corners in the image.

I set the corner limit to 5, to only detect the exterior corners.

corners = cv2.goodFeaturesToTrack(mask,5,0.01,30)

Step 5: Arrow Orientation

I wrote an algorithm from scratch to determine the orientation of the arrow by calculating the average values of all the corners, which basically gives us the center of the arrow. Then searched for the corner which had either a matching x or y coordinate with the center, which is the head of the arrow. I calculated the difference between the head of the arrow and the center of the arrow. This gives us an idea of where the head of the arrow is with respect to the center. If the x coordinate of difference is positive, then the arrow is pointing towards the Right, and if it is negative then it is pointing towards the Left. Similarly, if the y coordinate of the difference is positive, then arrow is pointing DOWN and if it is negative, the arrow is pointing UP.

Printed the direction of the arrow on the frame using the cv2.putText() function. I used this method to not use cv2.findContours().

Code:

```
for X in corners:
      if ((math.isclose(X[0],x_avg,abs_tol=2)) or (math.isclose(X[1],y_avg,abs_tol=2))):
        print('Found centre')
        print(X)
        D=X-center
        print('Difference is: \n')
        print(D)
        if (D[0]>0 and (math.isclose(D[1],0,abs_tol=10))):
          print('R')
          cv2.putText(frame,'Right',(50,50),font,2,text_color,2)
        if (D[0]<0 and (math.isclose(D[1],0,abs_tol=10))):
          print('L')
          cv2.putText(frame,'Left',(50,50),font,2,text_color,2)
        if (D[1]>0 and (math.isclose(D[0],0,abs_tol=10))):
          print('D')
          cv2.putText(frame,'Down',(50,50),font,2,text_color,2)
        if (D[1]<0 and (math.isclose(D[0],0,abs_tol=10))):
          print('U')
          cv2.putText(frame,'Up',(50,50),font,2,text_color,2)
```

Step 6: Applying to picamera video feed

Code: **(This is the complete code for the project)**

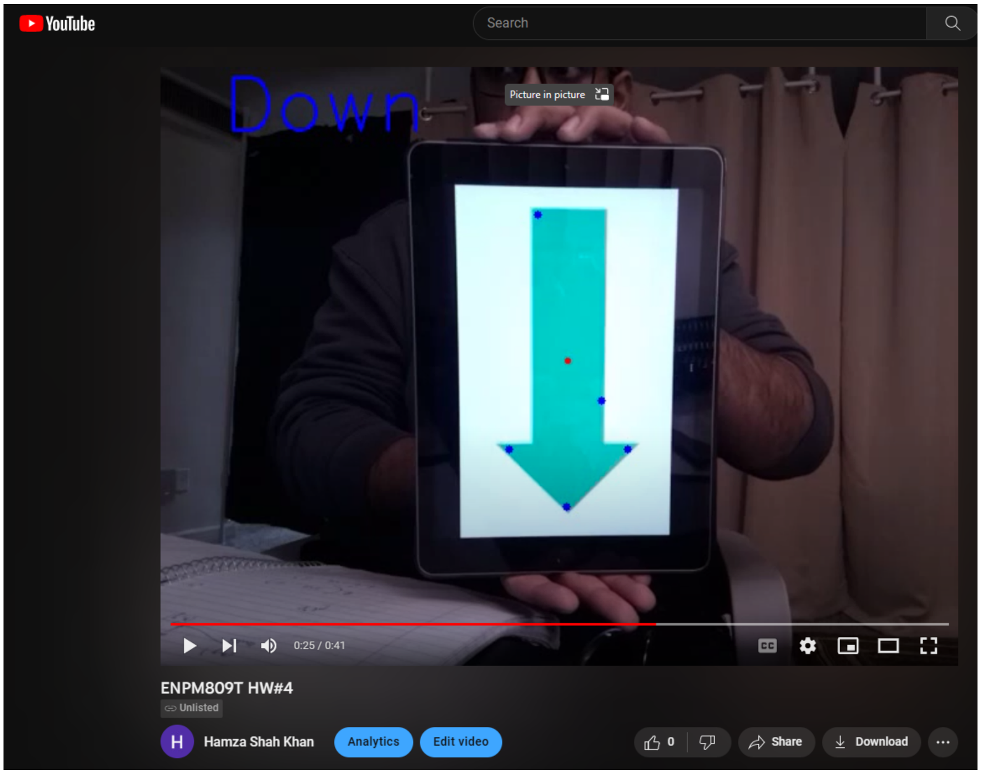
```
#for live video
import cv2
import os
from matplotlib import pyplot as plt
import numpy as np
import datetime
import math

cap = cv2.VideoCapture(0)
font = cv2.FONT_HERSHEY_SIMPLEX
text_color = (255, 0, 0)

frame_width=int(cap.get(3))
frame_height=int(cap.get(4))

f=open('hw4data.txt','a')

out=cv2.VideoWriter('hw3outputvideo.avi',cv2.VideoWriter_fourcc('M','J','P','G'),10,(frame_width,frame_height))

while True:
    start_time=datetime.datetime.now()

    ret,frame=cap.read()

    hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)

    low_green=np.array([60,60,225])
    high_green=np.array([95,255,255])

    mask = cv2.inRange(hsv, low_green,high_green)

    green= cv2.bitwise_and(frame,frame,mask=mask)

    frame=frame.copy()

    mask = cv2.GaussianBlur(mask, (11, 11), 0)

    corners = cv2.goodFeaturesToTrack(mask,5,0.01,30)
```

```
try:

    corners = np.int_(corners)
    corners = np.reshape(corners,(5,2))

    for i in corners:
        x,y = i.ravel()
        cv2.circle(frame,(x,y),3,255,-1)


    xmin = corners[:,0].min()
    ymin = corners[:,1].min()
    xmax = corners[:,0].max()
    ymax = corners[:,1].max()

    x_avg = round(((xmin + xmax)/2),0)
    y_avg = round(((ymin+ ymax)/2),0)

    center = [x_avg,y_avg]

    cv2.circle(frame,(int(x_avg),int(y_avg)),1,(0,0,255),2)

    for X in corners:
        if ((math.isclose(X[0],x_avg,abs_tol=2)) or (math.isclose(X[1],y_avg,abs_tol=2))):
            print('Found centre')
            print(X)
            D=X-center
            print('Difference is: \n')
            print(D)
            if (D[0]>0 and (math.isclose(D[1],0,abs_tol=10))):
                print('R')
                cv2.putText(frame,'Right',(50,50),font,2,text_color,2)
            if (D[0]<0 and (math.isclose(D[1],0,abs_tol=10))):
                print('L')
                cv2.putText(frame,'Left',(50,50),font,2,text_color,2)
            if (D[1]>0 and (math.isclose(D[0],0,abs_tol=10))):
                print('D')
                cv2.putText(frame,'Down',(50,50),font,2,text_color,2)
            if (D[1]<0 and (math.isclose(D[0],0,abs_tol=10))):
                print('U')
                cv2.putText(frame,'Up',(50,50),font,2,text_color,2)

except:
    continue

out.write(frame)
```

```
    stackedImg = np.hstack((green,frame))
    cv2.imshow("Image",stackedImg)

    end_time=datetime.datetime.now()
    now=end_time-start_time
    outputstr=str(now.total_seconds())+'\n'
    f.write(outputstr)

    if cv2.waitKey(1)== ord('q'):
        break


f.close()
cap.release()
out.release()
cv2.destroyAllWindows()
```

Step 7: Record Video File

Link to youtube video: https://youtu.be/TkS-p0Xqvk8



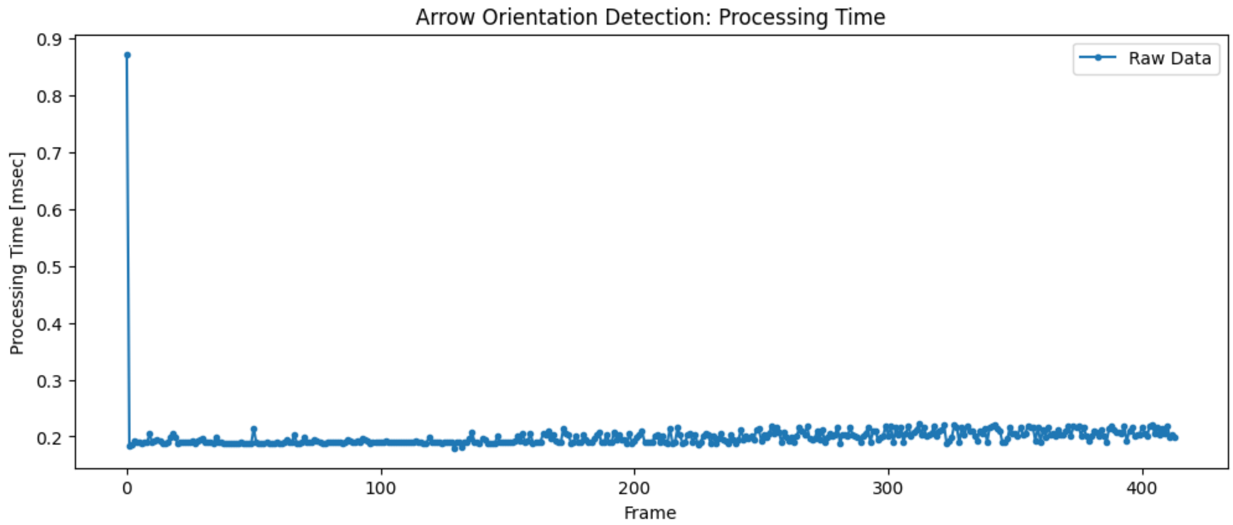
Step 8: Analysis

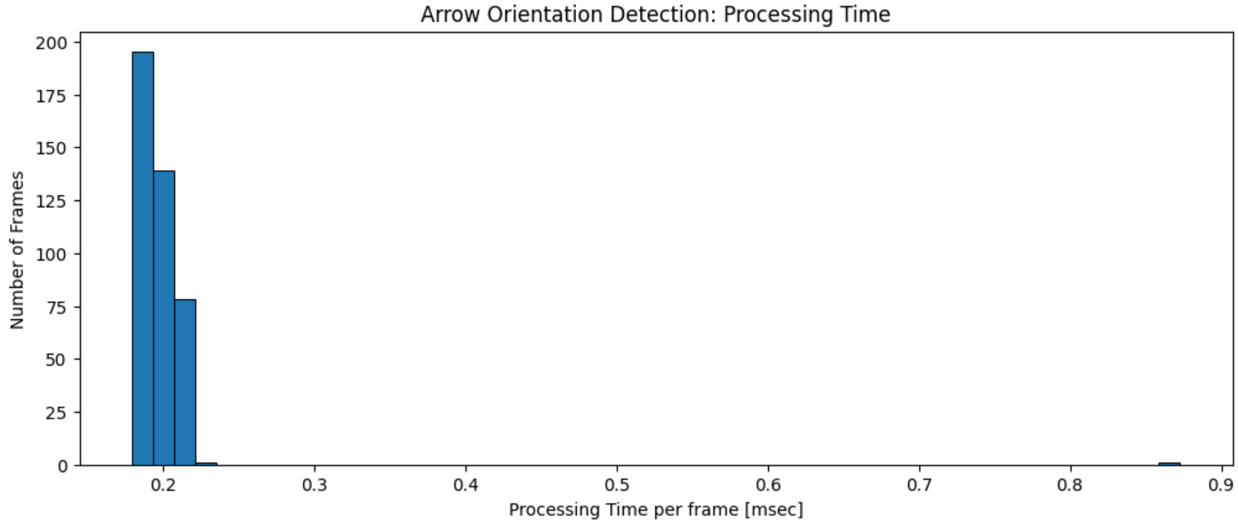
Using the datetime module, I calculated the time required to process each frame by using the following code:

```
f=open('hw4data.txt','a')
.
.
.
start_time=datetime.datetime.now()
.
.
.
end_time=datetime.datetime.now()
now=end_time-start_time
outputstr=str(now.total_seconds())+'\n'
f.write(outputstr)
```

Code for plotting the data:

```
#Importing packages
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#Reading delta values from .txt file from the Raspberry Pi
delta_values = np.loadtxt("HW4/From Rpi/hw4data.txt", dtype=float)
y=delta_values[:]
x=list(range(len(y)))
plt.subplots(figsize=(12, 10))

#Plotting x/y plot of the data
plt.subplot(2,1,1)
plt.title("Arrow Orientation Detection: Processing Time")
plt.xlabel("Frame")
plt.ylabel("Processing Time [msec]")
_= plt.plot(x,y,marker='.')
plt.legend(['Raw Data'])

#Plotting histogram of the data
plt.subplot(2,1,2)
plt.title("Arrow Orientation Detection: Processing Time")
plt.ylabel("Number of Frames")
plt.xlabel("Processing Time per frame [msec]")
_= plt.hist(y, bins=50, linewidth=.7, edgecolor="black")
```





The RaspberryPi worked pretty well for its limited hardware. The output video while running the program was a bit choppy which shows that the Pi's performance was hampered due to the load. The performance could be increased by optimizing the code, or by resizing the video to a lower resolution.