ENPM 809T: Autonomous Robotics

HOMEWORK 4

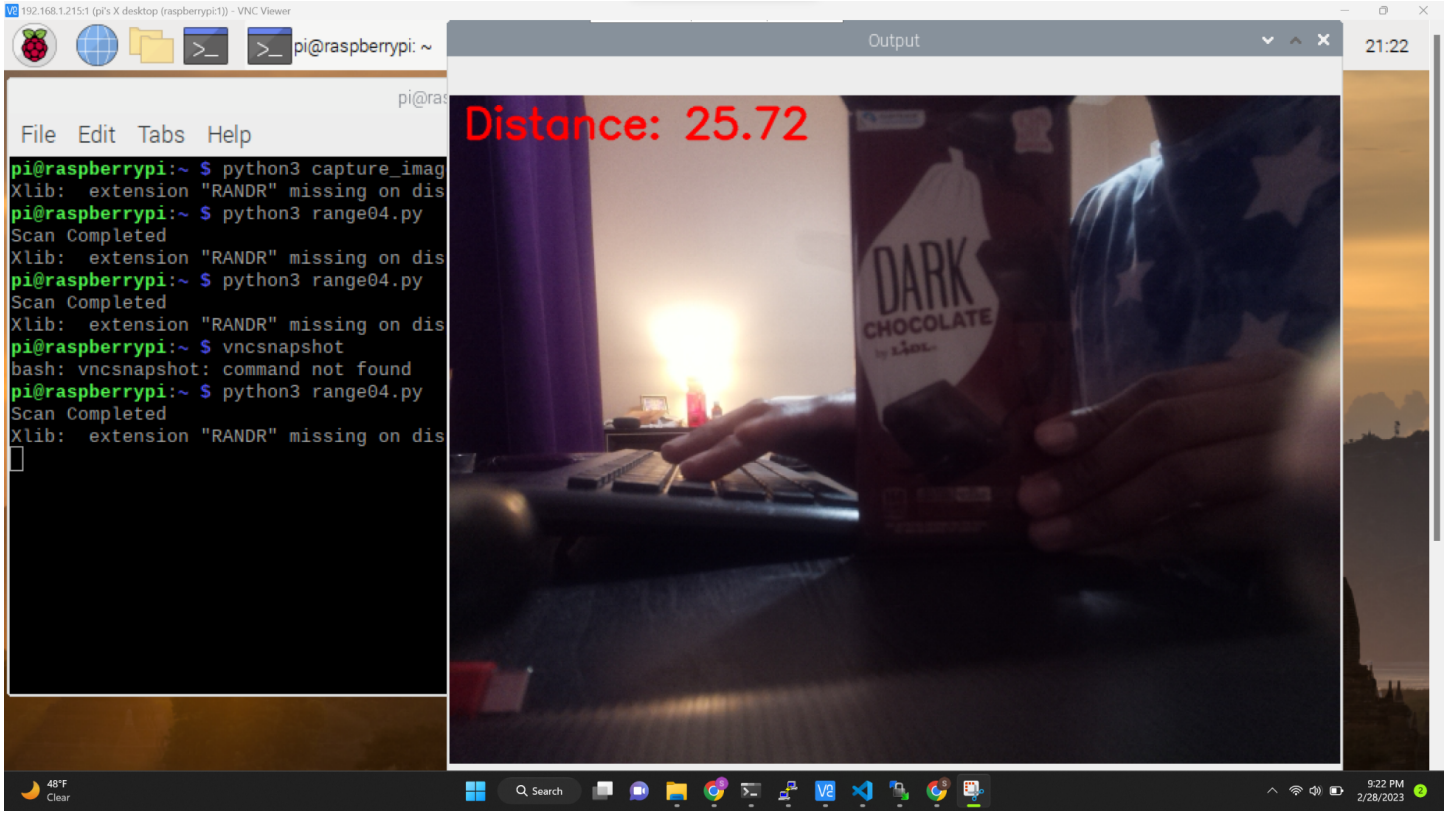
Name: Shantanu Suhas Parab
UID: 119347539
Directory ID: sparab
Date: 03/01/2023

PROBLEM 2:

CODE:

```
import RPi.GPIO as gpio
import time
import cv2


#Define pin allocations
trig=16
echo=18

def distance():
    gpio.setmode(gpio.BOARD)
    gpio.setup(trig,gpio.OUT)
    gpio.setup(echo,gpio.IN)

    #Ensure output has no value
    gpio.output(trig, False)
    time.sleep(0.01)

    #Genereate trigger pulse
    gpio.output(trig,True)
    time.sleep(0.00001)
    gpio.output(trig, False)

    #Generate echo time signal
    while gpio.input(echo) == 0:
        pulse_start = time.time()

    while gpio.input(echo)== 1:
```

```
        pulse_end = time.time()

    pulse_duration = pulse_end-pulse_start

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

    #Cleanup gpio 7 return distance
    gpio.cleanup()
    return distance

# Array to store distance
distance_array=[]

# Averaging the distances
for i in range(10):
    distance_array.append(distance())
    time.sleep(1)
print("Scan Completed")

# Read the image from file
image = cv2.imread('capture.jpg')
image=cv2.flip(image,0)
image=cv2.flip(image,1)

# Add text to the image
distance_string = "Distance: {:.2f}".format(sum(distance_array)/len(distance_array))
cv2.putText(image, distance_string, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 0,
255), 2, cv2.LINE_AA)

# Save the edited image
cv2.imwrite('distance_image.jpg', image)

# Display the image with the text
cv2.imshow('Output', image)
cv2.waitKey(0)
cv2.destroyAllWindows()
```


192.168.1.215:1 (pi's X desktop (raspberrypi:1)) - VNC Viewer
pi@raspberrypi: ~
Output
21:22
File Edit Tabs Help
pi@raspberrypi:~ $ python3 capture_imag
Xlib: extension "RANDR" missing on dis
pi@raspberrypi:~ $ python3 range04.py
Scan Completed
Xlib: extension "RANDR" missing on dis
pi@raspberrypi:~ $ python3 range04.py
Scan Completed
Xlib: extension "RANDR" missing on dis
pi@raspberrypi:~ $ vncsnapshot
bash: vncsnapshot: command not found
pi@raspberrypi:~ $ python3 range04.py
Scan Completed
Xlib: extension "RANDR" missing on dis
Distance: 25.72
DARK
CHOCOLATE
48°F
Clear
Search
9:22 PM
2/28/2023

PROBLEM 3:

Click the below link to view the video.
[Youtube Video for Submission]

CODE:

```
# import the necessary packages
from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import cv2
import numpy as np
import datetime
import math


fl = open('hw4data.txt','a')
# initialize the Raspberry Pi camera
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 25
rawCapture = PiRGBArray(camera, size=(640,480))
# allow the camera to warmup
time.sleep(0.1)

#Detection Parameters
angle_tolerance=30
confidence_threshold=80
# define the codec and create VideoWriter object
fourcc = cv2.VideoWriter_fourcc(*'XVID')
out = cv2.VideoWriter('arrow_final.avi', fourcc, 10, (640*3, 480))


font = cv2.FONT_HERSHEY_SIMPLEX #Setting the Font
textcolorf = (0,255,0)
textcolor = (100,200,0)

# Define the lower and upper bounds of the green color range in HSV
lower_green = np.array([72, 103, 196])
upper_green = np.array([104, 202, 255])
kernel_size = 5

# Define variables for frame rate and status display
fps = 0
```

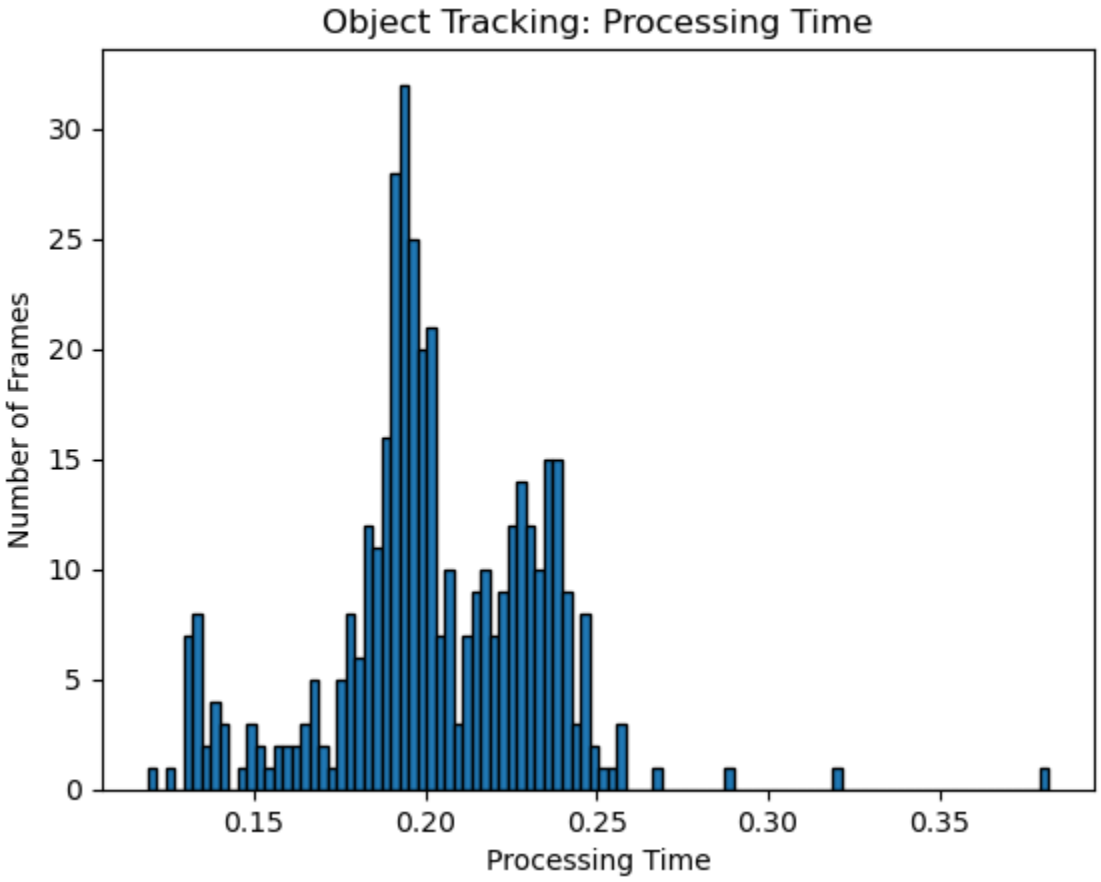
```
status = "Detecting..."
confidence=0
direction=str()
# Get the current time in seconds since the epoch
start_time = time.time()
# Loop through each frame of the video
f=0
for frame in camera.capture_continuous(rawCapture, format="bgr",
use_video_port=False):
    # Start time
    start_time = datetime.datetime.now()

    # grab the current frame
    img = frame.array
    img=cv2.flip(img,0)
    img=cv2.flip(img,1)

    # Convert the image to the HSV color space
    hsv_img = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # Define the lower and upper bounds of the green color range in HSV
    lower_green = np.array([48, 46, 213])
    upper_green = np.array([85, 168, 248])

    # Create a mask to isolate green pixels
    mask = cv2.inRange(hsv_img, lower_green, upper_green)

    # Apply erosion and dilation to remove noise and make corners smooth
    kernel = np.ones((5,5),np.uint8)
    erosion = cv2.erode(mask, kernel, iterations = 1)
    dilation = cv2.dilate(erosion, kernel, iterations = 1)

    # Apply gaussian blur to he mask
    mask_blur = cv2.GaussianBlur(dilation, (5, 5), 0)

    # Find the contours in the binary image
    contours, hierarchy = cv2.findContours(mask_blur, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)

    if len(contours)==0:
        # Default the readings
        status = "Detecting..."
        confidence=0
        direction=""
```

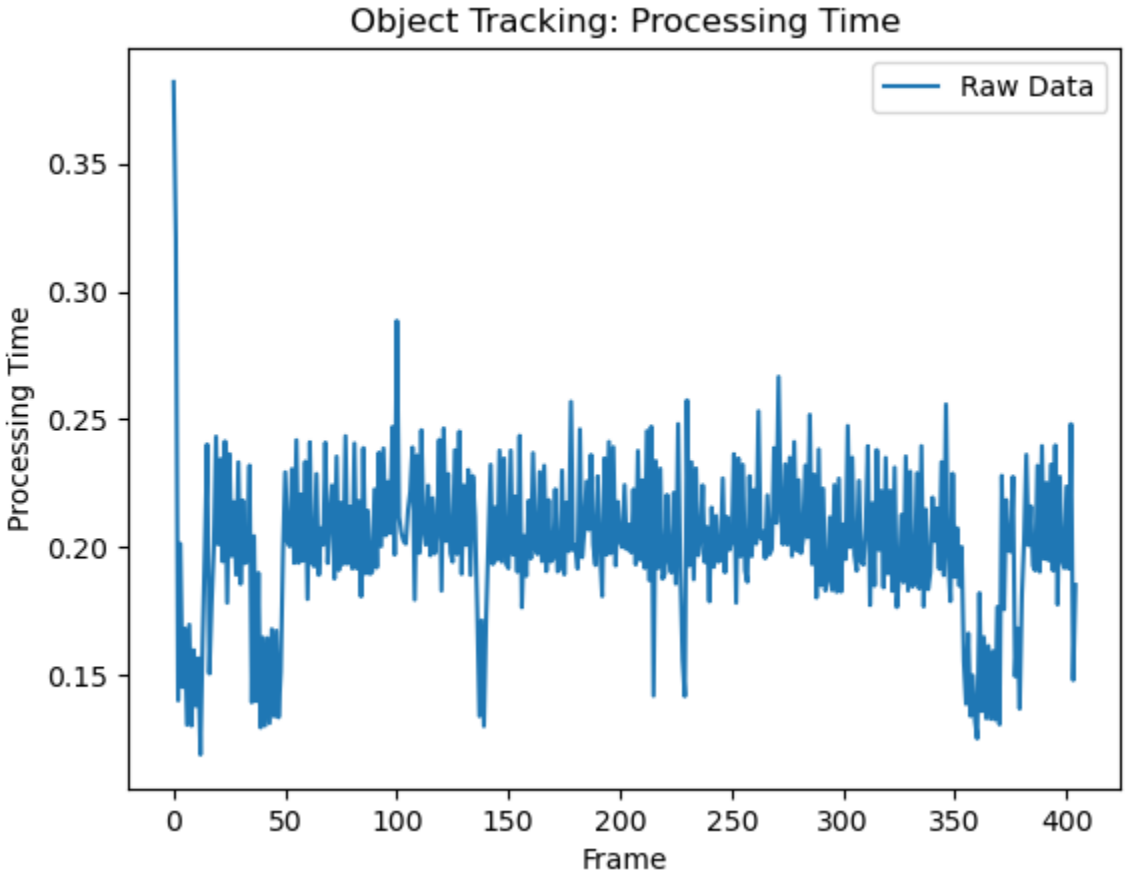
```
    else:
        # Update status
        status = "Green Object Detected"

        # Find the maximum contour
        max_contour = max(contours, key=cv2.contourArea)

        # Draw a bounding box around the max contour and mask everything else
        x,y,w,h = cv2.boundingRect(max_contour)
        tolerance=0
        mask = np.zeros(mask_blur.shape, np.uint8)
        mask[y:y+h+tolerance, x:x+w+tolerance] = mask_blur[y:y+h+tolerance,
x:x+w+tolerance]

        # Use Shi-Thomas feature detection
        corners = cv2.goodFeaturesToTrack(mask,5,0.35,10,15,blockSize=15)

        # Converting to integers
        corners = np.int0(corners)

        # Converting to numpy array for easy operation
        corners=np.array(corners)

        # Drawing the corners
        for corner in corners:
            x,y = corner.ravel()
            cv2.circle(img,(x,y),3,255,-1)

        # Checking for enough corners to make prediction
        if len(corners)>=5:
            # Updating the status
            status = "Arrow Detected"

            # Getting the center of the arrow
            M = cv2.moments(max_contour)
            cX = int(M["m10"] / M["m00"])
            cY = int(M["m01"] / M["m00"])

            # Comparing the width and height of the contour to predict orientation
            if w>h:
                # Count elements greater than and less than center x co ordinate
                greater = np.sum(corners[:, 0, 0] > cX)
                less = np.sum(corners[:, 0, 0] < cX)
                if greater>less:
```

```
                    direction="Right"
                else:
                    direction="Left"
            else:
                # Count elements greater than and less than center y co ordinate
                greater = np.sum(corners[:, 0, 1] > cY)
                less = np.sum(corners[:, 0, 1] < cY)
                if greater>less:
                    direction="Down"
                else:
                    direction="Up"

    # Showing the direction on image
    cv2.putText(img, direction, (10, 30), cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0),
2, cv2.LINE_AA)

    # Display the frame,  and status text
    text = "Status: {} | Frames: {:.2f}".format( status,f)
    cv2.putText(img, text, (10, 460), cv2.FONT_HERSHEY_SIMPLEX,  0.5, (255, 255,
255), 1)

    # Increment frame count
    f=f+1

    # Stack the original image, HSV image, and mask horizontally
    output = np.hstack((img, hsv_img, cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)))

    # Display the output
    cv2.imshow("Arrow Image", output)

    # Write the frame to video
    out.write(output)



    key = cv2.waitKey(1) & 0xFF
    # clear the stream in preparation for the next frame
    rawCapture.truncate(0)

    # End time
    end_time = datetime.datetime.now()

    #Calculating and saving time to file
    now=end_time-start_time
    outstring = str(now.total_seconds())+'\n'
```

```
    # Writing to file
    fl.write(outstring)
    # press the 'q' key to stop the video stream
    if key == ord("q"):
        break
```

PERFORMANCE





The processing time has increased in comparison to the previous assignment. It takes a longer time to detect the corners as it is an iterative process that makes an estimation for each pixel and finalizes a point with the most probable corners. The further computation required to detect the position of the arrow also adds up to the processing time.