809T Assignment 3

Autonomous Robotics

Prasanna Thirukudanthai Raghavan
UID: 118287546

Department of Robotics
University of Maryland
United States

Contents

Chapter 1

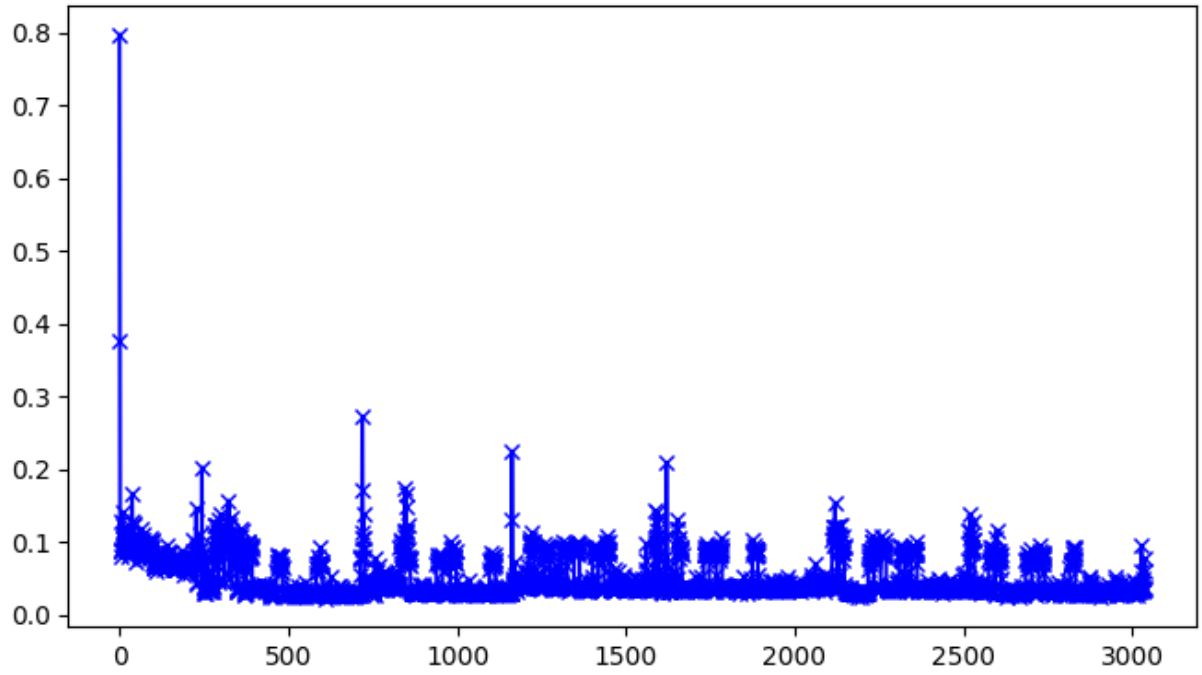
Object Tracking

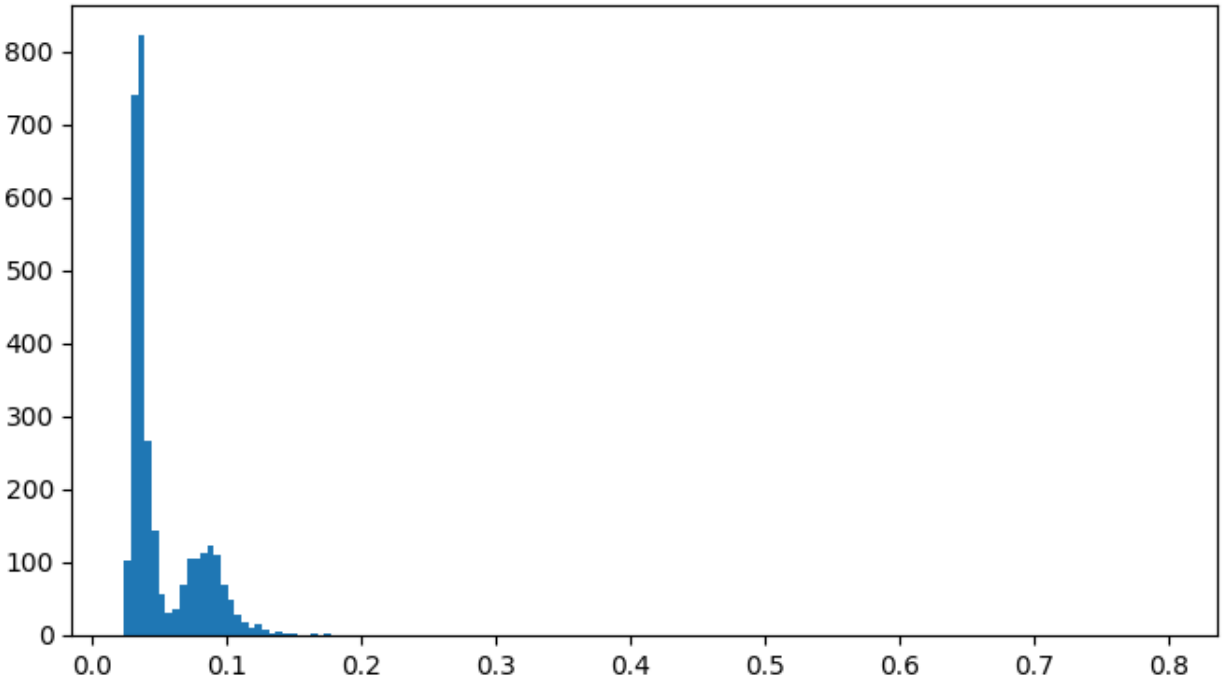
1.1 Video Link:

The Link: https://youtu.be/dKq_qguZSL0
All the requirements were checked prior to making this video.

1.2 Tracking Performance:

In this implementation of the problem we have reduced the size of the processed image as we can process more information and more number of frames instead of a single huge frame. But we see that there are a certain frills that form around other colors while doing this as the lessened number of frames mean that the quality of information that we process is also reduced. We essentially sacrifice the speed for time and as we have been informed about the grad challenge where we would not have much time to complete an assigned task we proceed to process the information faster as error correction can be done sooner than processing a large chunk of information together. The extra frills that appear around other colors is due to the lighting as the thresholding of the values tends to get hazy in extremely exposed scenarios. These are the constraints that we have worked with in this implementation.
We then try and plot the values of the time taken for each loop to run and after the first time, the loop seems to run in a faster time as supported by the histogram data also.


0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
0
500
1000
1500
2000
2500
3000


800
700
600
500
400
300
200
100
0
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

Chapter 2

Code:

2.1 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, (36, 25, 25), (86, 255,255))
39      #mask = cv2.inRange(hsv, (36, 25, 25), (70, 255,255))
40
41      ## slice the green
42      imask = mask>0
43      green = np.zeros_like(image, np.uint8)
44      green[imask] = image[imask]
45      imgray = cv2.cvtColor(green,cv2.COLOR_BGR2GRAY)
46      ret,thresh = cv2.threshold(imgray,100,255,0)
47      # find contours in the thresholded image
48      cnts = cv2.findContours(thresh.copy(), cv2.RETR_EXTERNAL,
49          cv2.CHAIN_APPROX_SIMPLE)
50      cnts = imutils.grab_contours(cnts)
51      # loop over the contours
52      for c in cnts:
53          # calculate moments for each contour
54          M = cv2.moments(c)
55          # compute the center of the contour
56          if M["m00"] != 0:
57              cX = int(M["m10"] / M["m00"])
58              cY = int(M["m01"] / M["m00"])
59          else:
60              cX, cY = 0, 0
61          # draw the contour and center of the shape on the image
62          cv2.drawContours(image, [c], -1, (128, 0, 128), 2)
63          cv2.circle(image, (cX, cY), 7, (128, 0, 128), -1)
64          # show the frame
65      cv2.imshow("Frame",image)
66      key = cv2.waitKey(1) & 0xFF
67
68      #save the frame to a file
69      out.write(image)
70
71      # clear the stream in preparation for the next frame
72      rawCapture.truncate(0)
73      # if the ‘q‘ key was pressed, break from the loop
74      if key == ord("q"):
75          break
76      stop = datetime.now()
77      # open .txt file to save data
78      f = open('hw3data.txt','a')
79      # print time to run through loop to the screen & save to file
80      now = stop - start
81      outstring = str(now.total_seconds()) + '\n'
82      f.write(outstring)
83      print(now.total_seconds())
```

2.2 Plotting Code:

```
1 import numpy as np
2 import matplotlib.pyplot as plt
3 data = np.loadtxt("hw3data.txt", dtype=float)
4 plt.subplot(1, 2, 1)
```

```
plt.plot(data,'xb-')
plt.subplot(1, 2, 2)
plt.hist(data,bins='auto')
plt.show()
```