EC ENGR 180DA, Winter 2022: Kellen Cheng (905155544)
Warm-up Lab (Partners: Ryan Doan, Newton Yee, Grace Zhao)
Due 13 January, 2022 at 6:00 PM

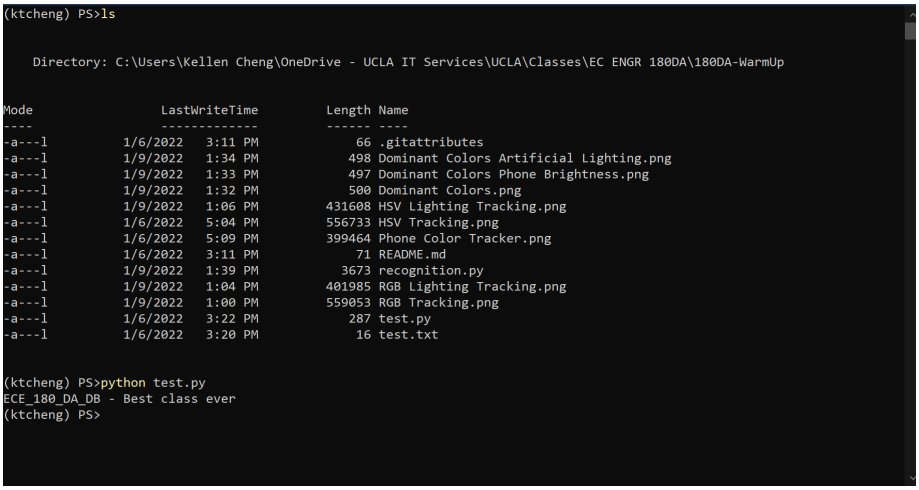
1 WarmUp Lab Tasks

(a) **Solution:** For reference regarding task 1, please use the following URL to access the Github repository (which will also contain files needed to verify completion for the rest of the following tasks). Feel free to let me know if the URL is faulty or does not work. Additionally, you may access this repository by going through the repositories on my Github profile (i.e. ID: ktcheng).

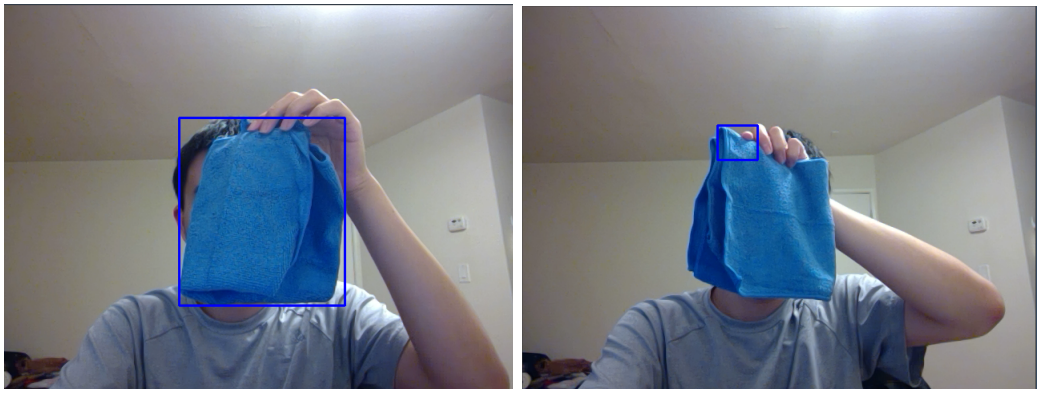
https://github.com/ktcheng/180DA-WarmUp

(b) **Solution:** For reference regarding task 2, please see the file titled "test.txt" in my repository. The text in the file should display the following: "testing 1 2 3!"

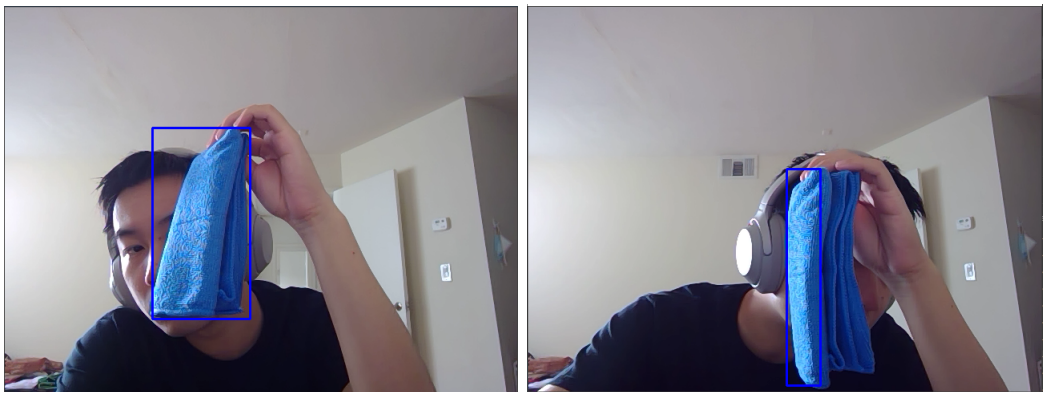
(c) **Solution:** For reference regarding task 3, please see the file titled "test.py" in my repository, which should contain the sample code. Additionally, below is a screenshot attached to show proof of successful completion calling and running the file (note that my virtual environment is "ktcheng").



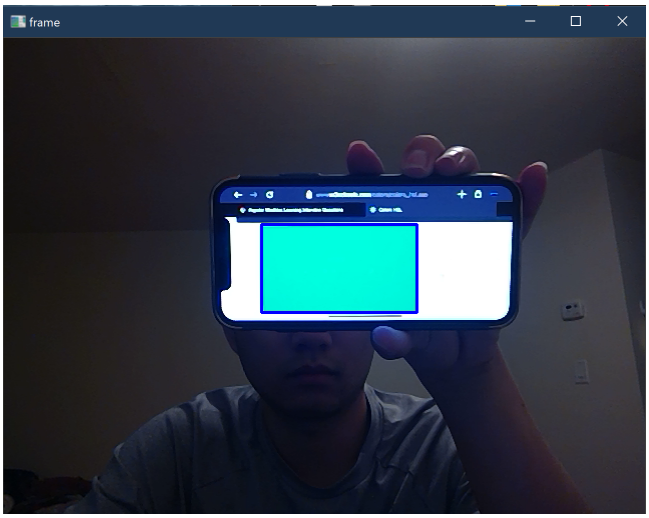
(d) **Solution:** For reference regarding task 4 part 1, below is a screenshot of the bounding box using HSV (left) and RGB (right). Clearly we can see that HSV does a much better job tracking the blue microfiber cloth as opposed to RGB tracking. The threshold range for both of these screenshots were quite large, as it had to encompass the entire range of the color "blue" so as to account for environment differences, such as change in lighting or orientation. Otherwise, these changes would fool the computer into erroneously believing that the cloth was a different color.



(e) **Solution:** For reference regarding task 4 part 2, below is a screenshot of the bounding box performance using HSV (left) and RGB (right), but this time with the blinds open (increases natural lighting conditions). The HSV performance dipped slightly, but performed reasonable within its expectations. On the other hand, the performance of the RGB color scheme improved remarkably, doing a much better job of capturing more of the blue microfiber cloth.



(f) **Solution:** For reference regarding task 4 part 3, we will use HSV tracking, as it had better performance across the previous trials (I attempted this section with the RGB tracking but received overall extremely poor tracking). From the screenshot below, it is evident that we needed high phone brightness to make the tracking relatively acceptable, as at low brightness it began mistaking the blue color as gray or even black.



(g) **Solution:** For reference regarding task 4 part 4, below are three screenshots of the dominant color performance. The top image is with natural lighting conditions, the second is with phone brightness altering lighting conditions, and the third is with artificial lighting (blinds closed to reduce lighting). From the color bars, it appears that phone brightness increases performance. On the other hand, natural lighting conditions actually hurt the performance, as the bright light creates many different shades of white that the Kmeans method mistakenly classifies as very different colors.

2 Python Code

(a) **Solution:** Below is the Python code required for the entirety of task 4. This can also be found on the Github repository under the file "recognition.py".

```
# -*- coding: utf-8 -*-
"""
Created on Thu Jan  6 15:24:15 2022

@author: Kellen Cheng

Dominant color finding code can be credited towards the following:
https://code.likeagirl.io/finding-dominant-colour-on-an-image-b4e075f98097

- Functions that are utilized are the same and are pulled from the source
- Altered the pipeline regarding image conversion before Kmeans analysis, such
  that image channels are manually reversed for intuitive sense (line 95)

Baseline video feed control loop code can be credited towards the following:
https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_gui/py_video_display/
py_video_display.html

- Overall structure is retained
- Color segmentation capacity is added to track the color range of Blue
- Additionally, the above line method is utilized for both HSV and RGB ranges
- Other changes include adding code that can draw real-time contour boxes around
  the tracked object
"""

import cv2
import sys
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
import matplotlib.patches as patches

# %% Dominant Color Tutorial Functions
def find_histogram(clt):
    """
    create a histogram with k clusters
    :param: clt
    :return:hist
    """
    numLabels = np.arange(0, len(np.unique(clt.labels_)) + 1)
    (hist, _) = np.histogram(clt.labels_, bins=numLabels)

    hist = hist.astype("float")
    hist /= hist.sum()
```

```
    return hist

def plot_colors2(hist, centroids):
    bar = np.zeros((50, 300, 3), dtype="uint8")
    startX = 0

    for (percent, color) in zip(hist, centroids):
        # plot the relative percentage of each cluster
        endX = startX + (percent * 300)
        cv2.rectangle(bar, (int(startX), 0), (int(endX), 50),
                      color.astype("uint8").tolist(), -1)
        startX = endX

    # return the bar chart
    return bar

# %% Video Feed Control Loop
cap = cv2.VideoCapture(0)
first = True
test_frame = None
test_mask = None

while(True):
    # Frame-by-frame
    ret, frame = cap.read()

    # Find blue objects in our video feed
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV) # HSV
    # hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB) # RGB

    # HSV Color Range for Blue
    lower_blue = np.array([100,150,20]) # RGB = 17, 51, 0
    upper_blue = np.array([140,255,255]) # RGB = 0, 255, 85
    mask = cv2.inRange(hsv, lower_blue, upper_blue)

    # RGB Color Range for Blue
    # lower_blue = np.array([17,51,0]) # RGB = 17, 51, 0
    # upper_blue = np.array([65,105,225]) # RGB = 0, 255, 85
    # mask = cv2.inRange(hsv, np.array([0,0,190]), np.array([140,255,255]))

    # Find the maximum contour and draw a rectangle over it
    contour, _ = cv2.findContours(mask, cv2.RETR_TREE, cv2.CHAIN_APPROX_NONE)

    # Draw contours on the live frames before displaying
    if len(contour) != 0:
        c = max(contour, key = cv2.contourArea)
        x,y,w,h = cv2.boundingRect(c)
```

```
        frame = cv2.rectangle(frame, (x, y), (x+w, y+h), (255,0,0), 2)

    # Display the resulting frame
    cv2.imshow('frame', frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

# When everything done, release the capture
cap.release()
cv2.destroyAllWindows()

# %% Kmeans Dominant Colors
img = frame[..., ::-1] # Reverse the channels to get to RGB

img = img.reshape((img.shape[0] * img.shape[1],3)) #represent as
↪  row*column,channel number
clt = KMeans(n_clusters=5) #cluster number
clt.fit(img)

hist = find_histogram(clt)
bar = plot_colors2(hist, clt.cluster_centers_)

plt.axis("off")
plt.imshow(bar)
plt.show()
```