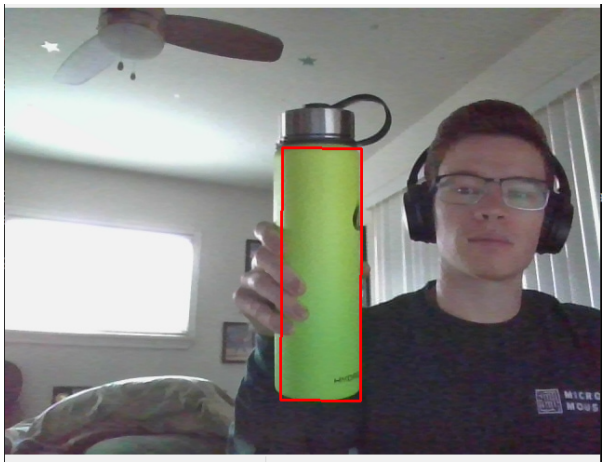
Caleb Terrill
1/13/2022
ECE 180DA

Lab 1 Report

1.
I tracked my yellow/green water bottle using my webcam video feed. The following is a snippet of the code:

```
import numpy as np
import cv2

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, img = cap.read()
    hsv = cv2.cvtColor(img,cv2.COLOR_BGR2HSV)

    yellow_lower = np.array([40, 100, 100])
    yellow_upper = np.array([100, 255, 255])
    mask_yellow = cv2.inRange(hsv, yellow_lower, yellow_upper)

    contours,hierarchy = cv2.findContours(mask_yellow, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    if contours:
        cnt = max(contours, key = cv2.contourArea)

        rect = cv2.minAreaRect(cnt)
        box = cv2.boxPoints(rect)
        box = np.int0(box)
        cv2.drawContours(img,[box],0,(0,0,255),2)

    cv2.imshow('frame',img)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

Here are some screenshots of the feed.



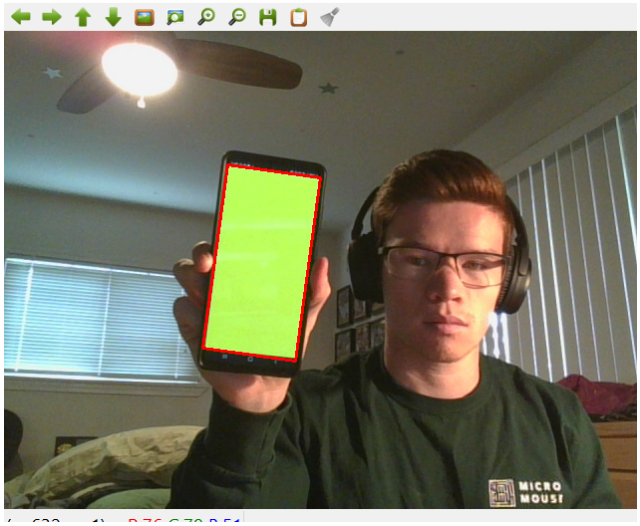
I found that HSV was better at tracking the color, since I could very precisely specify the color I was looking to capture and the amount of wiggle room I was willing to permit.

2.

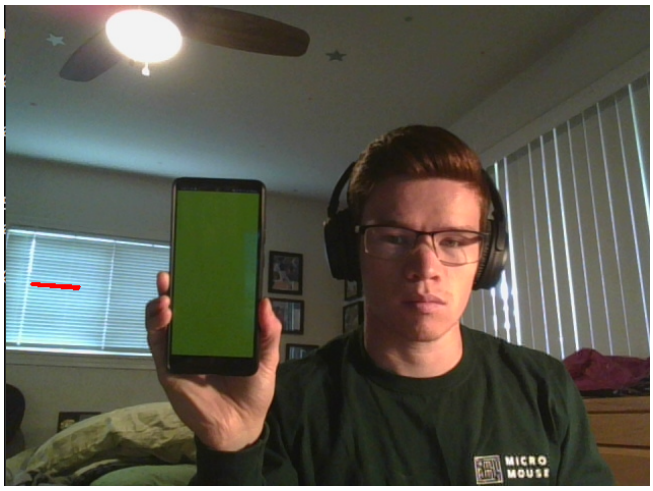
I turned on a bright light and closed the blinds in the back. The algorithm was no longer able to track the bottle as well since the gradient across the bottle now did not all fit within my color range I originally specified. In order to track the bottle, I would need to increase my range of colors that were counted as yellow.

3.

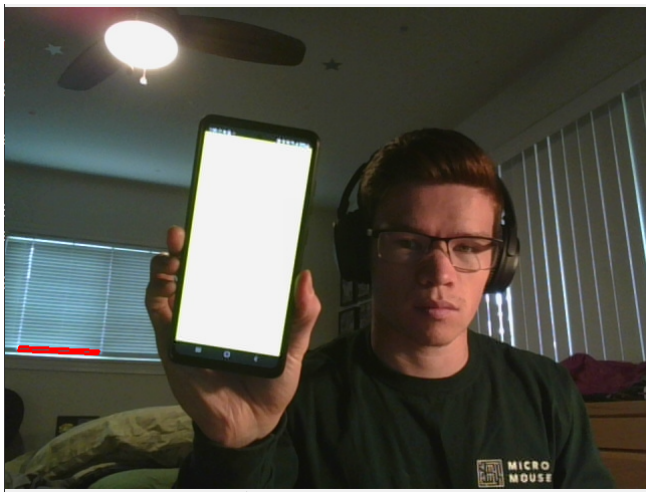
I set my phone to the same yellow color as my bottle. At a medium brightness, the tracker was able to recognise the phone's yellow very well.



When I turned down the brightness, the algorithm was no longer able to track the phone's yellow, and often totally missed it.



Similarly, turning the brightness up too high caused the yellow to get washed out and not be recognised. The yellow was too bright with respect to the surroundings for the camera to pick it up well.



4.

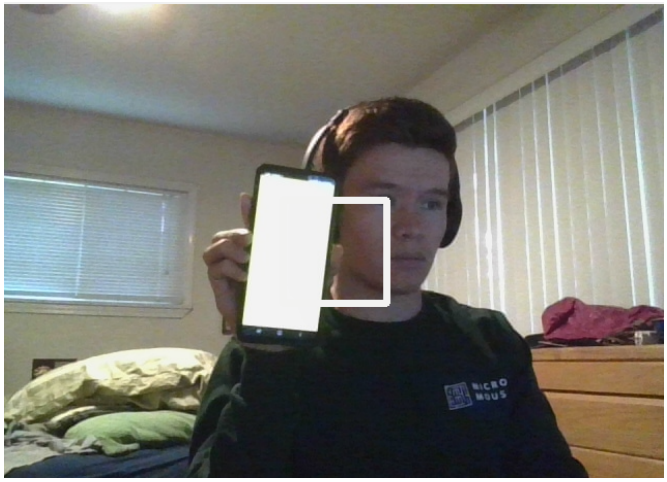
Next I determined the dominant color in the video feeds. The following code pulled pixels from a 100x100 rectangle in the center of the frames, determined the dominant color using kmeans with k = 3, and painted the rectangle in the output video feed in that dominant color.

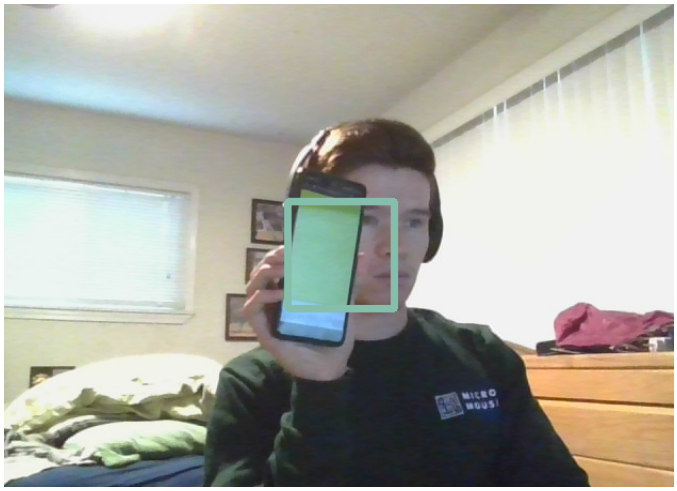
```
import numpy as np
import cv2
from sklearn.cluster import KMeans
from collections import Counter

cap = cv2.VideoCapture(0)

while(True):
    # Capture frame-by-frame
    ret, img = cap.read()
    img_small = img[190:190+100,270:270+100]
    img_small = img_small.reshape((img_small.shape[0] * img_small.shape[1],3)) #represent as row*column,channel number

    clt = KMeans(n_clusters=3) #cluster number
    clt.fit(img_small)

    count = Counter(clt.labels_)
    common_label = count.most_common(1)[0][0]
    common_color = clt.cluster_centers_[common_label]


    cv2.rectangle(img,(270,190),(270+100,190+100),common_color,5)
    cv2.imshow('frame',img)

    if cv2.waitKey(1) & 0xFF == ord('q'):
        break


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

It worked pretty well, here are some examples.


It seemed robust to different lighting conditions. The only issue is the dominant color might not be the dominant color of the actual object since the camera picks up the light differently in the dark. For example in the following picture the dominant picture is a muddy green since the bottle is a mix of light and rand yellow/green.

When detecting the dominant color on the phone under various brightnesses, it seemed to do as well as the camera allowed it to, getting the right color as shown on the images.





So it seems like this technique is more sensitive to external lighting conditions than the brightness of the object.