Avinash Sookram

Professor Avanzato

Lab #7: Red and Blue Color Detection with OpenCV and Python

Contents

[Objective: 1](#_Toc121769575)

[Test Case #1: 4 red objects, Unsuccessful 1](#_Toc121769576)

[Test Case #2: Successful, Combination of Blue and Red 3](#_Toc121769577)

[Test Case #3: One Red, Two Blue objects, Unsuccessful 4](#_Toc121769578)

[Test Case #4: All Blue, Success 4](#_Toc121769579)

[Test Case #5: Mix of Red and Blue, Success 5](#_Toc121769580)

[Test Case #6: Two Red Objects, Success 6](#_Toc121769581)

[Test Case #7: Two Blue Objects, Success 6](#_Toc121769582)

[Test Case #8: Two Blue Objects, Success 7](#_Toc121769583)

[Test Case #10: One Red Object, One Blue Object, Unsuccessful 8](#_Toc121769584)

[Conclusion: 8](#_Toc121769585)

[OpenCV Python Code: 9](#_Toc121769586)

[References: 10](#_Toc121769587)

Objective:   
To use OpenCV and Python to showcase blue and red objects in a live video feed.

Test Case #1: 4 red objects, Unsuccessful   
(Can be interpreted as the program thinking the tiny body wash container on the top left was part of the red folder)

Graphical user interface

Description automatically generated

Test Case #2: Successful, Combination of Blue and Red  
(Band-AID box and Wet Ones, red box around Wet Ones wording)

A picture containing text, indoor, electronics, screenshot

Description automatically generated

Test Case #3: One Red, Two Blue objects, Unsuccessful  
(Giant red folder, mouth wash (boxed), Q-tip container, unsuccessful since folder should’ve been highlighted.)

A picture containing text, indoor, screenshot

Description automatically generated

Test Case #4: All Blue, Success   
(Nintendo 3DS(boxed), mouth wash, Q-tip container

A picture containing text, indoor, electronics, display

Description automatically generated

# Test Case #5: Mix of Red and Blue, Success

(Nintendo 3DS(boxed), deck box, body wash, mouthwash, Wet Ones, Q-Tip container)

A screenshot of a computer

Description automatically generated with medium confidence

# Test Case #6: Two Red Objects, Success

(Body wash container (boxed) and deck box)

A picture containing text, indoor, screenshot

Description automatically generated

# Test Case #7: Two Blue Objects, Success

(Glow-mouse(boxed), mouth wash)

A picture containing text, monitor, indoor, electronics

Description automatically generated

Test Case #8: Two Blue Objects, Success  
 (Pen, Mouthwas(boxed))

A picture containing text, indoor, desktop

Description automatically generated

Test Case #9: One Red, One Blue Object, Success   
(Mouth wash(boxed), scissors)

A picture containing text, indoor, monitor, electronics

Description automatically generated

Test Case #10: One Red Object, One Blue Object, Unsuccessful  
(Folder, mouth wash(boxed), unsuccessful as folder should be boxed)

A computer screen capture

Description automatically generated with low confidence

Conclusion:  
This lab was pretty cool to do being able to understand how the camera processes two different colors at once for analysis. I did notice that some colors showed up so much more in the masks than others even though I would think they are the same shade (such as in Test Case #6 where I was expecting the box to stand out more in the combined blue and red mask). Overall I feel like most of the test cases were failures though, as it seems like neither the most vibrant nor the biggest of the colors were highlighted the most in the bounding box, and I am concerned that these sought after traits did not appear in the masks.

# OpenCV Python Code:

import cv2 as cv

import numpy as np

cap = cv.VideoCapture(0)

while(1):

    # Take each frame

    \_, frame = cap.read()

    # Convert BGR to HSV

    hsv = cv.cvtColor(frame, cv.COLOR\_BGR2HSV)

    # define range of blue color in HSV

    lower\_blue = np.array([110,50,50])

    upper\_blue = np.array([130,255,255])

    lower\_red = np.array([-20, 100, 100])

    upper\_red = np.array([13, 255, 255])

    # Threshold the HSV image to get only blue colors

    mask\_red = cv.inRange(hsv, lower\_red, upper\_red)

    mask\_blue = cv.inRange(hsv, lower\_blue, upper\_blue)

    # Bitwise-AND mask and original image

    mask = cv.bitwise\_or(mask\_red, mask\_blue)

    contours, hierarchy = cv.findContours(mask, cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_SIMPLE) #CHAIN APPROX SIMPLE -  It removes all redundant points and compresses the contour, thereby saving memory

     #first one is source image, second is contour retrieval mode, third is contour approximation

     # RETR\_EXTERNAL If you use this flag, it returns only extreme outer flags. All child contours are left behind.

    if(len(contours)) != 0:

        for contour in contours: #go through all contours in feed

            if cv.contourArea(contour) > 500:  #pick out high contour areas

                max\_contour = max(contours, key = cv.contourArea) #get max contour

                x, y, w, h = cv.boundingRect(max\_contour) #bounding box around max contour

                cv.rectangle(frame, (x,y), (x+w, y+h), (0,0,255)) #dimensions for bounding box

    res = cv.bitwise\_and(frame, frame, mask=mask)

    cv.imshow('frame', frame)

    cv.imshow('mask', mask)

    cv.imshow('res', res)

    k = cv.waitKey(5) & 0xFF

    if k == 27:                             # escape key is 27;  space is 32, etc

        break

cv.destroyAllWindows()

# References:

<https://docs.opencv.org/4.x/d9/d8b/tutorial_py_contours_hierarchy.html>

<https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html>