Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**ЛАБОРАТОРНАЯ РАБОТА №1**

**Дисциплина: Алгоритмы цифровой обработки мультимедиа**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Д.Е.Гиренко

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Крамаренко

**Цель работы:** Тестирование вывода изображений и видео на экран, запись в файл, формат изображения HSV, определение цвета, построение надписей и дополнительных изображений на рисунке.

**Ход работы:**

Задание 1. Установить библиотеку OpenCV.

С помощью команды pip install cv2 через терминал в VS Code библиотека OpenCV была установлена.

import cv2

Задание 2. Вывести на экран изображение.

Протестировать три возможных расширения, три различных флага для создания окна и три различных флага для чтения изображения.

Для того чтобы вывести изображение на экран необходимо использовать метод cv.imread(), отвечающий за загрузку картинки на экран. Параметры (флаги) этой функции позволяют изменить расширение изображения, способы чтения и т. д. На рисунках 1 и 2 представлены изображения в формате .jpg и в сером цвете, отображенные с помощью данной команды с флагами cv2.IMREAD\_COLOR и cv2.IMREAD\_GRAYSCALE соответственно.



Рисунок 1 – Изображение в формате .jpg.

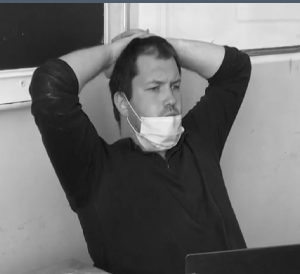


Рисунок 2 – Изображение в сером цвете.

Именованное окно создано с помощью команды сv2.namedWindow() с флагом cv2.WINDOW\_NORMAL, отображающим окно в формате Windows-окна.

Задание 3. Отобразить видео в окне. Рассмотреть методы класса VideoCapture и попробовать отображать видео в разных форматах, в частности размеры и цветовая гамма.

Отобразить заранее записанное видео или видеопоток с IP камеры можно с помощью методов класса VideoCapture. Создаётся экземпляр класса с помощью конструктора cv2.VideoCapture(), в качестве параметров передается название видео с расширением и флаг размерности окна. Далее используется цикл while с командой cv2.isOpened(), т. е. пока видео открыто, цикл выполняется. Кадр из видеопотока читается с помощью функции cap.read(). Эта функция возвращает два значения, первое значение ret – булевское значение, обозначающее, удалось ли выполнить чтение кадра. Сам кадр называем фреймом и сохраняем в формат картинки (двумерная матрица). Если изображение закончилось, ret вернет false и отображение завершится. Далее отобразим полученный фрейм в окне (с помощью команды cv2.imshow() и проверяется условие на закрытие окна. Если нажата клавиша Escape(код 27), отображение завершено.

Отобразить видео в разных размерах можно с помощью команд cv2.resizeWindow() и cv2.cvtColor() соответственно. Флаги первой команды позволяют изменять размер окна, а второй – цветовую гамму. На рисунке 3 представлен результат работы программы.



Рисунок 3 – Отображение видео в окне размером 640x480 в сером цвете.

Задание 4. Записать видео из файла в другой файл.

Для выполнения данного задания код, используемый в предыдущем задании, был изменён. В частности, была добавлена команда для получения размеров кадра (cv2.CAP\_PROP\_FRAME\_WIDTH, cv2.CAP\_PROP\_FRAME\_HEIGHT) и был создан объект класса VideoWriter для записи видео в файл с помощью команды cv2.VideoWriter("./Output/output3.mp4", fourcc, 25, (w, h) , в аргументах которой указано название файла, в который записано видео, код кодека, используемый для сжатия кадров, частота кадров создаваемого видеопотока и размер видеокадров.

Задание 5. Прочитать изображение, перевести его в формат HSV. Вывести на экран два окна, в одном изображение в формате HSV, в другом – исходное изображение.

Перевести изображение в формат HSV с легкостью можно с помощью ранее используемой команды cv2.cvtColor(), указав в аргументах флаг HSV формата (cv2.COLOR\_BGR2HSV).

RGB и HSV — это два различных цветовых пространства, которые используются для представления цветов. RGB — это аддитивное цветовое пространство, которое используется для представления цветов на экране. Оно состоит из красного, зеленого и синего цветов. HSV — это цветовое пространство, которое используется для представления цветов в виде оттенка, насыщенности и значения. Он также называется HSB (оттенок, насыщенность, яркость). Он позволяет легче управлять цветом, чем RGB. В аддитивном цветовом пространстве цвет создается путем смешивания света разных цветовых компонентов. Например, красный, зеленый и синий свет смешиваются вместе, чтобы создать белый свет. Это отличается от вычитательного цветового пространства, которое используется для представления цветов в печати и других физических процессах. В вычитательном цветовом пространстве цвет создается путем вычитания определенных цветовых компонентов из белого света.

Для перевода цвета из RGB в HSV выполняются следующие действия:

1)Необходимо нормализовать значения R, G и B, чтобы они находились в диапазоне от 0 до 1 – поделить значения на 255.

2)Поиск максимальное и минимальное значение из R, G и B.

3)Вычисляется разность между максимальным и минимальным значением.

4)Вычисляется значение яркости (V) как максимальное значение из R, G и B.

5)Вычисляется насыщенность (S) как разность между максимальным значением и значением яркости (V), деленную на максимальное значение.

6) Оттенок (H) вычисляется следующим образом:

Если максимальным значением является R, то H = 60 \* ((G-B)/delta) (если G < B, то H += 360). Если максимальным значением является G, то H = 60 \* ((B-R)/delta + 2). Если максимальным значением является B, то H = 60 \* ((R-G)/delta + 4). Здесь delta = max(R,G,B) - min(R,G,B). Например, если есть пиксель с RGB-значениями (255, 0, 0), то его HSV-значения будут следующими: H = 0, S = 1,V = 1.

Обратный перевод выполняется следующим образом:

1) Нормализация значения H, S и V, чтобы они находились в диапазоне от 0 до 1.

2) Вычисление следующих значений: C = V \* S, X = C \* (1 - |(H / 60°) mod 2 - 1|) и m = V - C.

3) Вычисление значения R’, G’ и B’ следующим образом:

* + Если 0 ≤ H < 60, то R’ = C, G’ = X, B’ = 0.
  + Если 60 ≤ H < 120, то R’ = X, G’ = C, B’ = 0.
  + Если 120 ≤ H < 180, то R’ = 0, G’ = C, B’ = X.
  + Если 180 ≤ H < 240, то R’ = 0, G’ = X, B’ = C.
  + Если 240 ≤ H < 300, то R’ = X, G’ = 0, B’ = C.
  + Если 300 ≤ H < 360, то R’ = C, G’ = 0, B’ = X.

4) Вычисление значения R, G и B следующим образом: R = (R’ + m) \* 255; G = (G’ + m) \* 255; и B = (B’ + m) \* 255.

Например, если есть пиксель с HSV-значениями (120°, 0.5, 0.5), то его RGB-значения будут следующими: R = 64, G = 128 и B = 64 – красный цвет.

На рисунке 4 представлены два окна с оригинальным и изменённым изображением.



Рисунок 4 – Изображение в формате RGB(справа) и в формате HSV(слева).

Задание 6. (самостоятельно) Прочитать изображение с камеры. Вывести в центре на экране Красный крест в формате, как на изображении. Указать команды, которые позволяют это сделать.

Для каждого из ожидаемых прямоугольников (горизонтального и вертикального) была задана своя ширина и высота в пикселях, а также – цвет и ширина линий отрисовки.

Затем вычислялись координаты прямоугольников таким образом, чтобы они были расположены в центре изображения. Для каждого из углов по оси x и y использовались следующие формулы: левый верхний угол находится в точке (xi\_1, yi\_1), где xi\_1 = (ширина изображения / 2) - (ширина прямоугольника / 2), а yi\_1 = (высота изображения / 2) - (высота прямоугольника / 2). Правый нижний угол прямоугольника находится в точке (xi+1\_1, yi+1\_1), где xi+1\_1 = (ширина изображения / 2) + (ширина прямоугольника / 2), а yi+1\_1 = (высота изображения / 2) + (высота прямоугольника / 2).

Прямоугольники были отрисованы с помощью команды cv2.rectangle(), в которую передавались изображение, координаты углов, цвет и толщина линий.

Для размытия центра пересечения прямоугольников и горизонтального прямоугольника был применен метод GaussianBlur() – размытие изображения с помощью фильтра Гаусса. Она принимает следующие параметры: исходное изображение, размер ядра и стандартное отклонение по X и Y. Она вычисляет взвешенное среднее значение пикселей в окне ядра, чтобы создать размытую версию изображения. Чем больше размер ядра, тем сильнее размытие. Чем больше стандартное отклонение, тем меньше влияние дальних пикселей на вычисление нового значения пикселя.

На рисунке 5 отображен получившийся результат.

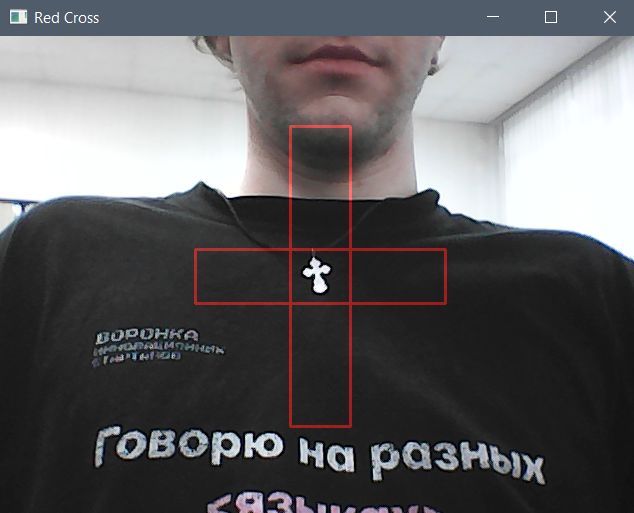


Рисунок 5 – Получившееся изображение.

Задание 7. (самостоятельно) Отобразить информацию с вебкамеры, записать видео в файл, продемонстрировать видео.

В конструкторе cv2.VideoCapture(0) был указан 0, отвечающий за подключение веб-камеры, в отличие от предыдущего задания, где было указано название используемого видео.

На рисунке 6 представлен кадр видео.

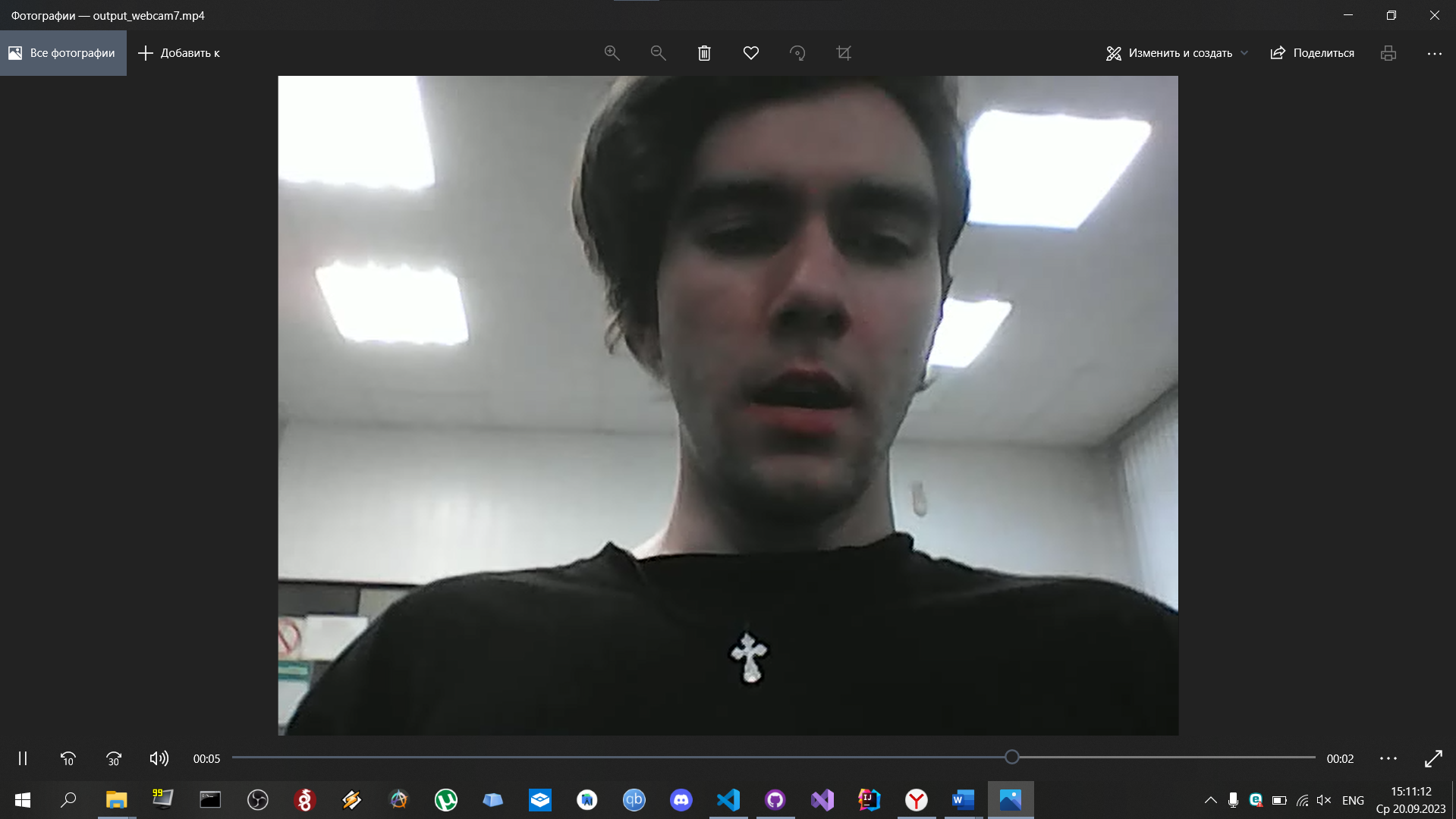


Рисунок 6 – Запись с вебкамеры

Задание 8. (самостоятельно) Залить крест одним из 3 цветов – красный, зеленый, синий по следующему правилу: НА ОСНОВАНИИ ФОРМАТА RGB определить, центральный пиксель ближе к какому из цветов красный, зеленый, синий и таким цветом заполнить крест.

Был использован код из задания 6. Для определения ближайшего цвета из трех возможных вариантов: красный, зеленый или синий, сначала необходимо получить цвет центрального пикселя в формате RGB. Затем вычисляется расстояние между этим цветом и каждым из трех возможных цветов в пространстве RGB, используя евклидово расстояние между координатами цветов. Ближайший цвет из списка возможных цветов по индексу соответствует минимальному расстоянию. После этого вновь была использована функция cv2.rectangle() с видоизменным списком аргументов (необходимо было закрасить оба прямоугольника). На рисунке 7 они отображены.

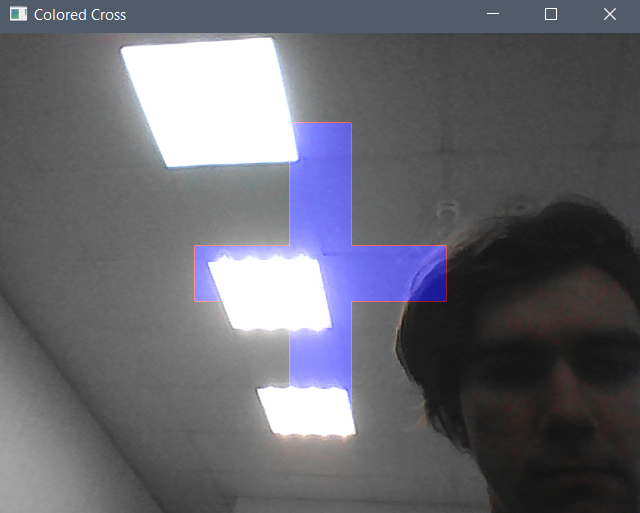


Рисунок 7 – Закрашенный крест.

Задание 9. (самостоятельно). Подключите телефон, подключитесь к его камере, выведете на экран видео с камеры. Продемонстрировать процесс на ноутбуке преподавателя и своем телефоне.

Для выполнения этого задания предварительно на телефон было установлено приложение IPWebcam, позволяющее подключить телефон как IP-камеру. В конструктор cv2.VideoCapture("http://192.168.204.1:8080/shot.jpg ") был передан URL-адрес потока видео. Также этот URL-адрес можно использовать в браузере на компьютере, чтобы увидеть видео с телефона. На рисунке 8 отображен кадр из видеопотока.

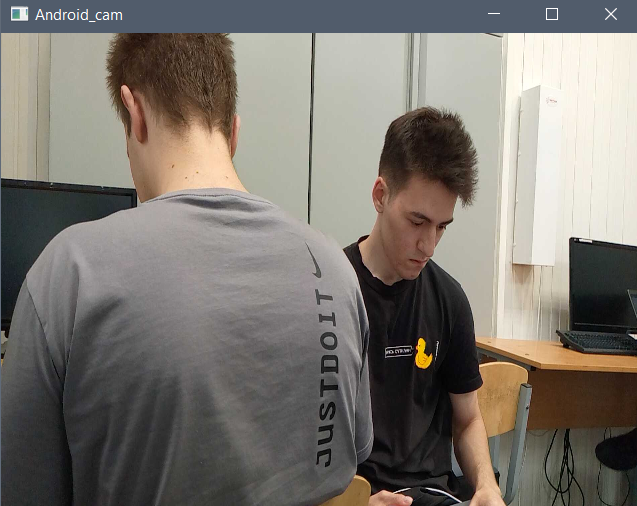


Рисунок 8 – Запись с камеры телефона.

**Листинг программы**

import requests

import cv2

import numpy as np

def task2():

    img1 = cv2.imread(r'.\source\img1.jpg', cv2.IMREAD\_GRAYSCALE)  # ЧБ

    img2 = cv2.imread(r'.\source\img2.png',

                      cv2.IMREAD\_UNCHANGED)  # без изменений

    # с учетом  глубины цвета

    img3 = cv2.imread(r'.\source\img3.bmp', cv2.IMREAD\_ANYDEPTH)

    cv2.namedWindow('lol', cv2.WINDOW\_NORMAL)

    cv2.namedWindow('test', cv2.WINDOW\_NORMAL)

    cv2.namedWindow('kek', cv2.WINDOW\_NORMAL)

    cv2.imshow('lol', img1)

    cv2.imshow('test', img2)

    cv2.imshow('kek', img3)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

def task3():

    cap = cv2.VideoCapture(r'.\source\video.mp4', cv2.CAP\_ANY)

    new\_width = 640

    new\_height = 480

    while True:

        ret, frame = cap.read()

        if not ret:

            exit()

        frame = cv2.resize(frame, (new\_width, new\_height))

        gray\_frame = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

        cv2.imshow('Video', gray\_frame)

        if cv2.waitKey(1) & 0xFF == 27:

            exit()

def task4():

    video = cv2.VideoCapture(r'.\source\video.mp4', cv2.CAP\_ANY)

    ok, vid = video.read()

    w = int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

    h = int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

    fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')

    video\_writer = cv2.VideoWriter("./Output/output3.mp4", fourcc, 25, (w, h))

    while (True):

        ok, vid = video.read()

        if (not ok):

            break

        cv2.imshow('Video', vid)

        video\_writer.write(vid)

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

            break

    video.release()

    video\_writer.release()

    cv2.destroyAllWindows()

def task5():

    img1 = cv2.imread(r'.\source\img2.png')

    img2 = cv2.imread(r'.\source\img2.png')

    cv2.namedWindow('test', cv2.WINDOW\_NORMAL)

    cv2.namedWindow('test\_hsv', cv2.WINDOW\_NORMAL)

    cv2.imshow('test', img1)

    hsv = cv2.cvtColor(img2, cv2.COLOR\_BGR2HSV)

    cv2.imshow('test\_hsv', hsv)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

def task6():

    cap = cv2.VideoCapture(0)

    while True:

        ret, frame = cap.read()

        if not ret:

            break

        height, width, \_ = frame.shape

        cross\_image = np.zeros((height, width, 3), dtype=np.uint8)

        vertical\_line\_width = 60

        vertical\_line\_height = 300

        cv2.rectangle(cross\_image,

                      (width // 2 - vertical\_line\_width // 2,

                       height // 2 - vertical\_line\_height // 2),

                      (width // 2 + vertical\_line\_width // 2,

                       height // 2 + vertical\_line\_height // 2),

                      (0, 0, 255), 2)

        horizontal\_line\_width = 250

        horizontal\_line\_height = 55

        cv2.rectangle(cross\_image,

                      (width // 2 - horizontal\_line\_width // 2,

                       height // 2 - horizontal\_line\_height // 2),

                      (width // 2 + horizontal\_line\_width // 2,

                       height // 2 + horizontal\_line\_height // 2),

                      (0, 0, 255), 2)

        result\_frame = cv2.addWeighted(frame, 1, cross\_image, 0.5, 0)

        cv2.imshow("Red Cross", result\_frame)

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

            break

    cap.release()

    cv2.destroyAllWindows()

def task7():

    video = cv2.VideoCapture(0)

    ok, vid = video.read()

    w = int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

    h = int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

    fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')

    video\_writer = cv2.VideoWriter(

        "./Output/output\_webcam7.mp4", fourcc, 25, (w, h))

    while (True):

        ok, vid = video.read()

        cv2.imshow('Video', vid)

        video\_writer.write(vid)

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

            break

    video.release()

    video\_writer.release()

    cv2.destroyAllWindows()

def task8():

    cap = cv2.VideoCapture(0)

    while True:

        ret, frame = cap.read()

        if not ret:

            break

        height, width, \_ = frame.shape

        cross\_image = np.zeros((height, width, 3), dtype=np.uint8)

        vertical\_line\_width = 60

        vertical\_line\_height = 300

        cv2.rectangle(cross\_image,

                      (width // 2 - vertical\_line\_width // 2,

                       height // 2 - vertical\_line\_height // 2),

                      (width // 2 + vertical\_line\_width // 2,

                       height // 2 + vertical\_line\_height // 2),

                      (0, 0, 255), 2)

        rect\_start\_v = (width // 2 - vertical\_line\_width // 2,

                        height // 2 - vertical\_line\_height // 2)

        rect\_end\_v = (width // 2 + vertical\_line\_width // 2,

                      height // 2 + vertical\_line\_height // 2)

        horizontal\_line\_width = 250

        horizontal\_line\_height = 55

        cv2.rectangle(cross\_image,

                      (width // 2 - horizontal\_line\_width // 2,

                       height // 2 - horizontal\_line\_height // 2),

                      (width // 2 + horizontal\_line\_width // 2,

                       height // 2 + horizontal\_line\_height // 2),

                      (0, 0, 255), 2)

        rect\_start\_h = (width // 2 - horizontal\_line\_width // 2,

                        height // 2 - horizontal\_line\_height // 2)

        rect\_end\_h = (width // 2 + horizontal\_line\_width // 2,

                      height // 2 + horizontal\_line\_height // 2)

        central\_pixel\_color = frame[height // 2, width // 2]

        color\_distances = [

            np.linalg.norm(central\_pixel\_color - np.array([0, 0, 255])),

            np.linalg.norm(central\_pixel\_color - np.array([0, 255, 0])),

            np.linalg.norm(central\_pixel\_color - np.array([255, 0, 0]))

        ]

        closest\_color\_index = np.argmin(color\_distances)

        if closest\_color\_index == 0:

            cv2.rectangle(cross\_image, rect\_start\_h,

                          rect\_end\_h, (0, 0, 255), -1)

        elif closest\_color\_index == 1:

            cv2.rectangle(cross\_image, rect\_start\_h,

                          rect\_end\_h, (0, 255, 0), -1)

        else:

            cv2.rectangle(cross\_image, rect\_start\_h,

                          rect\_end\_h, (255, 0, 0), -1)

        if closest\_color\_index == 0:

            cv2.rectangle(cross\_image, rect\_start\_v,

                          rect\_end\_v, (0, 0, 255), -1)

        elif closest\_color\_index == 1:

            cv2.rectangle(cross\_image, rect\_start\_v,

                          rect\_end\_v, (0, 255, 0), -1)

        else:

            cv2.rectangle(cross\_image, rect\_start\_v,

                          rect\_end\_v, (255, 0, 0), -1)

        result\_frame = cv2.addWeighted(frame, 1, cross\_image, 0.5, 0)

        cv2.imshow("Colored Cross", result\_frame)

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

            break

    cap.release()

    cv2.destroyAllWindows()

def task9():

    url = "http://192.168.204.1:8080/shot.jpg"

    while True:

        img\_resp = requests.get(url)

        img\_arr = np.array(bytearray(img\_resp.content), dtype=np.uint8)

        img = cv2.imdecode(img\_arr, -1)

        img = cv2.resize(img, (640, 480))

        cv2.imshow("Android\_cam", img)

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

            break

    cv2.destroyAllWindows()

task9()