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: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Романов В.В.

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Крамаренко А.А.

**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 IDE PyCharm была установлена библиотека OpenCV. Для работы необходимо было включать библиотеку в проект, в частности, в каждый скрипт директивой:

**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, они же флаги, этой функции позволяют изменить расширение изображения и способы чтения. На рисунках 1 и 2 представлены изображения в формате .jpg 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 2 значения, первое из них 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, то 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 800х600 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\_1.mov', fourcc, 30.0, (width, height), 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 2 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. 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.

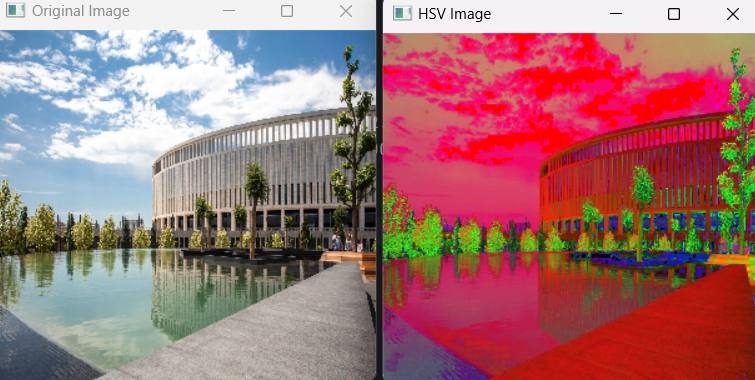


Рисунок 4 – Изображение в формате RGB и HSV соответственно.

Задание 6. 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 нарисованы 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.

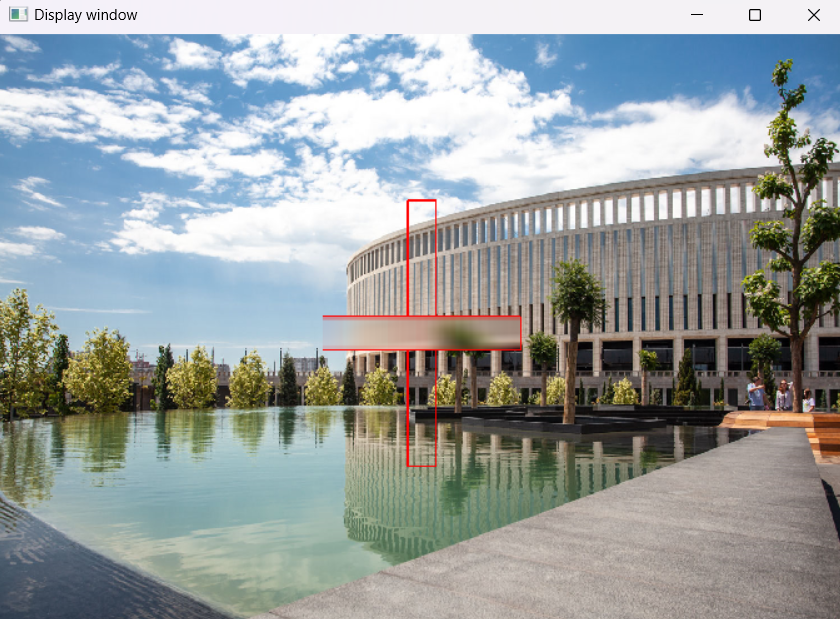


Рисунок 5 – Получившееся изображение.

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

Для выполнения этого задания код, используемый в задании 3 был модифицирован в функцию readIPWriteTOFile(). 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. 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.

Для определения ближайшего цвета из 3 возможных 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 была использована функция cv2.rectangle() с видоизменным списком аргументов. На рисунке 7 изображен результат работы программы.

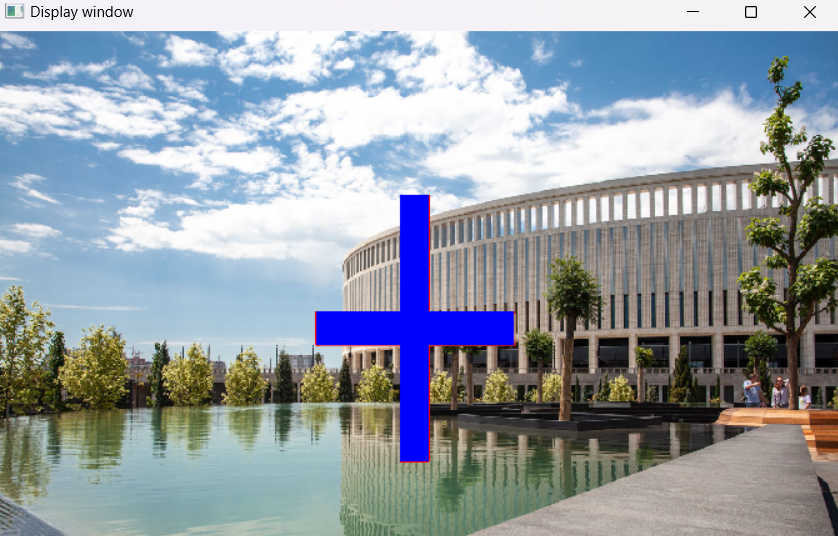


Рисунок 7 – Закрашенный в синий цвет крест.

Задание 9. 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://212.192.147.27:8080/video") 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.

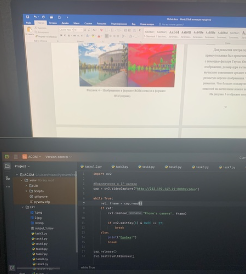


Рисунок 8 – Запись с камеры телефона.

**Листинг программ**

**Файл task1\_2.py**

import cv2  
  
*#Флаги для расширения изображений*  
img1 = cv2.imread('2.jpg', cv2.IMREAD\_COLOR)  
img2 = cv2.imread('3.bmp', cv2.IMREAD\_COLOR)  
  
*#Флаги для чтения изображения*  
img3 = cv2.imread('1.png', cv2.IMREAD\_COLOR) *#Цветное изображение*  
img4 = cv2.imread('1.png', cv2.IMREAD\_GRAYSCALE) *#Серое изображение*  
img5 = cv2.imread('1.png', cv2.IMREAD\_UNCHANGED) *#Без изменений*  
  
*#Флаги для создания окна*  
cv2.namedWindow('Display window', cv2.WINDOW\_NORMAL)  
*#cv2.namedWindow('Display window', cv2.WINDOW\_AUTOSIZE)*  
*#cv2.namedWindow('Display window', cv2.WINDOW\_FULLSCREEN)*  
  
cv2.imshow('Display window', img4)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**Файл task3.py**

import cv2  
  
*#Отображение видео в окне*  
cap = cv2.VideoCapture('Video1.mp4', cv2.WINDOW\_NORMAL)  
cv2.namedWindow('Video', cv2.WINDOW\_NORMAL)  
  
*#Изменение размера окна*  
cv2.resizeWindow('Video', 800, 600)  
*#cv2.resizeWindow('Video', 1024, 1000)*  
*#cv2.resizeWindow('Video', 1800, 800)*  
  
*#Чтение видеофайла кадр за кадром*  
while cap.isOpened():  
 ret, frame = cap.read()  
 if ret:  
 *#Изменение цветовой гаммы кадра*  
gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)  
 vsh = cv2.cvtColor(frame, cv2.COLOR\_BGR2HSV)  
 lab = cv2.cvtColor(frame, cv2.COLOR\_BGR2LAB)  
  
 *#Отображение кадра в окне*  
cv2.imshow('Video', gray)  
  
 *#Выход при нажатии 'esc'*  
if cv2.waitKey(1) & 0xFF == 27:  
 break  
 else:  
 break  
  
cap.release()  
cv2.destroyAllWindows()

**Файл task4.py**

import cv2  
  
*#Отображение видео в окне*  
cap = cv2.VideoCapture(0)  
cv2.namedWindow('Video', cv2.WINDOW\_NORMAL)  
  
*#Получение размеров кадра*  
width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
  
*#Создание объект VideoWriter для записи видео в файл*  
fourcc = cv2.VideoWriter\_fourcc(\*'mp4v')  
out = cv2.VideoWriter('output\_1.mov', fourcc, 30.0, (width, height))  
  
*#Чтение видеофайла кадр за кадром и запись его в другой файл*  
while cap.isOpened():  
 ret, frame = cap.read()  
 if ret:  
 out.write(frame)  
 cv2.imshow('Video', frame)  
  
 if cv2.waitKey(1) & 0xFF == 27:  
 break  
 else:  
 break  
  
cap.release()  
out.release()  
cv2.destroyAllWindows()

**Файл task5.py**

import cv2  
  
img = cv2.imread('2.jpg')  
  
cv2.namedWindow('Original Image', cv2.WINDOW\_NORMAL)  
cv2.namedWindow('HSV Image', cv2.WINDOW\_NORMAL)  
  
cv2.imshow('Original Image', img)  
  
hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)  
  
cv2.imshow('HSV Image', hsv)  
  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**Файл task6.py**

import cv2  
  
img = cv2.imread('1.png')  
cv2.namedWindow('Display window', cv2.WINDOW\_NORMAL)  
  
*#Цвет и толщина фигуры*  
color = (0, 0, 255)  
thickness = 2  
  
height, width, \_, = img.shape  
  
*#Ширина и высота вертикального прямоугольника*  
rect\_width\_1 = 50  
rect\_height\_1 = 400  
  
*#Координаты углов*  
x1\_1 = width // 2 - rect\_width\_1 // 2 *#Левый верхний угол по оси x*  
y1\_1 = height // 2 - rect\_height\_1 // 2 *#Левый верхний угол по оси y*  
x2\_1 = width // 2 + rect\_width\_1 // 2 *#Правый нижний угол по оси x*  
y2\_1 = height // 2 + rect\_height\_1 // 2 *#Правый нижний угол по оси y*  
  
*#Ширина и высота горизонтального прямоугольника*  
rect\_width\_2 = 50  
rect\_height\_2 = 350  
  
*#Координаты углов*  
x1\_2 = width // 2 - rect\_height\_2 // 2 *#Левый верхний угол по оси x*  
y1\_2 = height // 2 - rect\_width\_2 // 2 *#Левый верхний угол по оси y*  
x2\_2 = width // 2 + rect\_height\_2 // 2 *#Правый нижний угол по оси x*  
y2\_2 = height // 2 + rect\_width\_2 // 2 *#Правый нижний угол по оси y*  
  
*#Отрисовка*  
cv2.rectangle(img, (x1\_1, y1\_1), (x2\_1, y2\_1), color, thickness)  
cv2.rectangle(img, (x1\_2, y1\_2), (x2\_2, y2\_2), color, thickness)  
  
*#Для размытия центра креста используется GaussianBlur*  
*#Ширина и высота ядра в px для размытия*  
kernel\_size = (71, 11)  
  
*#Часть изображения, соответствующая горизонтальному прямоугольнику*  
img\_part = img[y1\_2:y2\_2, x1\_2:x2\_2]  
  
img\_part\_blur = cv2.GaussianBlur(img\_part, kernel\_size, 30)  
  
*#Замена части изображения размытой версией*  
img[y1\_2:y2\_2, x1\_2:x2\_2] = img\_part\_blur  
  
cv2.imshow('Display window', img)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**Файл task7.py**

import cv2  
  
def readIPWriteTOFile():  
 video = cv2.VideoCapture(0)  
 ok, img = video.read()  
 w = int(video.get(cv2.CAP\_PROP\_FRAME\_WIDTH))  
 h = int(video.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))  
 fourcc = cv2.VideoWriter\_fourcc(\*'XVID')  
 video\_writer = cv2.VideoWriter("output\_2.mov", fourcc, 25, (w, h))  
 while (True):  
 ok, img = video.read()  
 cv2.imshow('Video from webcam', img)  
 video\_writer.write(img)  
  
 if cv2.waitKey(1) & 0xFF == 27:  
 break  
 video.release()  
 cv2.destroyAllWindows()  
  
readIPWriteTOFile()

**Файл task8.py**

import cv2  
import numpy as np  
  
img = cv2.imread('1.png')  
cv2.namedWindow('Display window', cv2.WINDOW\_NORMAL)  
  
*#Цвет и толщина прямоугольников и линии*  
color = (0, 0, 255)  
thickness = 2  
  
height, width, \_, = img.shape  
  
*#Ширина и высота вертикального прямоугольника*  
rect\_width\_1 = 50  
rect\_height\_1 = 400  
  
*#Координаты углов*  
x1\_1 = width // 2 - rect\_width\_1 // 2 *#Левый верхний угол по оси x*  
y1\_1 = height // 2 - rect\_height\_1 // 2 *#Левый верхний угол по оси y*  
x2\_1 = width // 2 + rect\_width\_1 // 2 *#Правый нижний угол по оси x*  
y2\_1 = height // 2 + rect\_height\_1 // 2 *#Правый нижний угол по оси y*  
  
*#Ширина и высота горизонтального прямоугольника*  
rect\_width\_2 = 50  
rect\_height\_2 = 350  
  
*#Координаты углов*  
x1\_2 = width // 2 - rect\_height\_2 // 2 *#Левый верхний угол по оси x*  
y1\_2 = height // 2 - rect\_width\_2 // 2 *#Левый верхний угол по оси y*  
x2\_2 = width // 2 + rect\_height\_2 // 2 *#Правый нижний угол по оси x*  
y2\_2 = height // 2 + rect\_width\_2 // 2 *#Правый нижний угол по оси y*  
  
*#Отрисовка*  
cv2.rectangle(img, (x1\_1-5, y1\_1-3), (x2\_1-5, y2\_1-3), color, thickness)  
cv2.rectangle(img, (x1\_2-5, y1\_2-3), (x2\_2-5, y2\_2-3), color, thickness)  
  
*#Для размытия центра креста используется GaussianBlur*  
*#Ширина и высота ядра в px для размытия*  
kernel\_size = (71, 11)  
  
*#Часть изображения, соответствующая горизонтальному прямоугольнику*  
img\_part = img[y1\_2:y2\_2, x1\_2:x2\_2]  
  
img\_part\_blur = cv2.GaussianBlur(img\_part, kernel\_size, 30)  
  
*#Замена части изображения размытой версией*  
img[y1\_2:y2\_2, x1\_2:x2\_2] = img\_part\_blur  
  
*#Получение цвета центрального пикселя в формате RGB*  
cx = width // 2 *#Координата x центра изображения*  
cy = height // 2 *#Координата y центра изображения*  
r, g, b = img[cy][cx] *#Компоненты цвета в RGB*  
  
*#Определение ближайшего цвета*  
colors = [(255, 0, 0), (0, 255, 0), (0, 0, 255)] *#Список возможных цветов в формате RGB*  
distances = [] *#Список расстояний от центрального пикселя до каждого цвета*  
  
*#Расстояние между 2-мя цветами в RGB вычисляем по Евклидовому расстоянию между их координатами*  
for color in colors:  
 distance = np.sqrt((r - color[0])\*\*2 + (g - color[1])\*\*2 + (b - color[2])\*\*2)  
 distances.append(distance)  
  
*#Индекс ближайшего цвета в списке colors соответствует минимальному расстоянию в списке distances*  
min\_index = distances.index(min(distances))  
  
*#Выбор ближайшего цвета из списка colors по индексу*  
nearest\_color = colors[min\_index]  
  
*#Закрашивание креста ближайшим цветом*  
cv2.rectangle(img, (x1\_1-5, y1\_1-3), (x2\_1-5, y2\_1-3), nearest\_color, -1) *#-1 заполнение всей области*  
cv2.rectangle(img, (x1\_2-5, y1\_2-3), (x2\_2-5, y2\_2-3), nearest\_color, -1)  
  
cv2.imshow('Display window', img)  
cv2.waitKey(0)  
cv2.destroyAllWindows()

**Файл task9.py**

import cv2  
  
# объект VideoCapture для подключения к IP-камере  
cap = cv2.VideoCapture("http://212.192.147.27:8080/video")  
  
while True:  
 ret, frame = cap.read()  
 if ret:  
 cv2.imshow("Phone's camera", frame)  
 # Ждем нажатия клавиши q для выхода из цикла  
 if cv2.waitKey(1) & 0xFF == 27:  
 break  
 else:  
 print("Ошибка чтения видео")  
 break  
  
cap.release()  
cv2.destroyAllWindows()