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 выполнила: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Пинский Д.А.

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:**

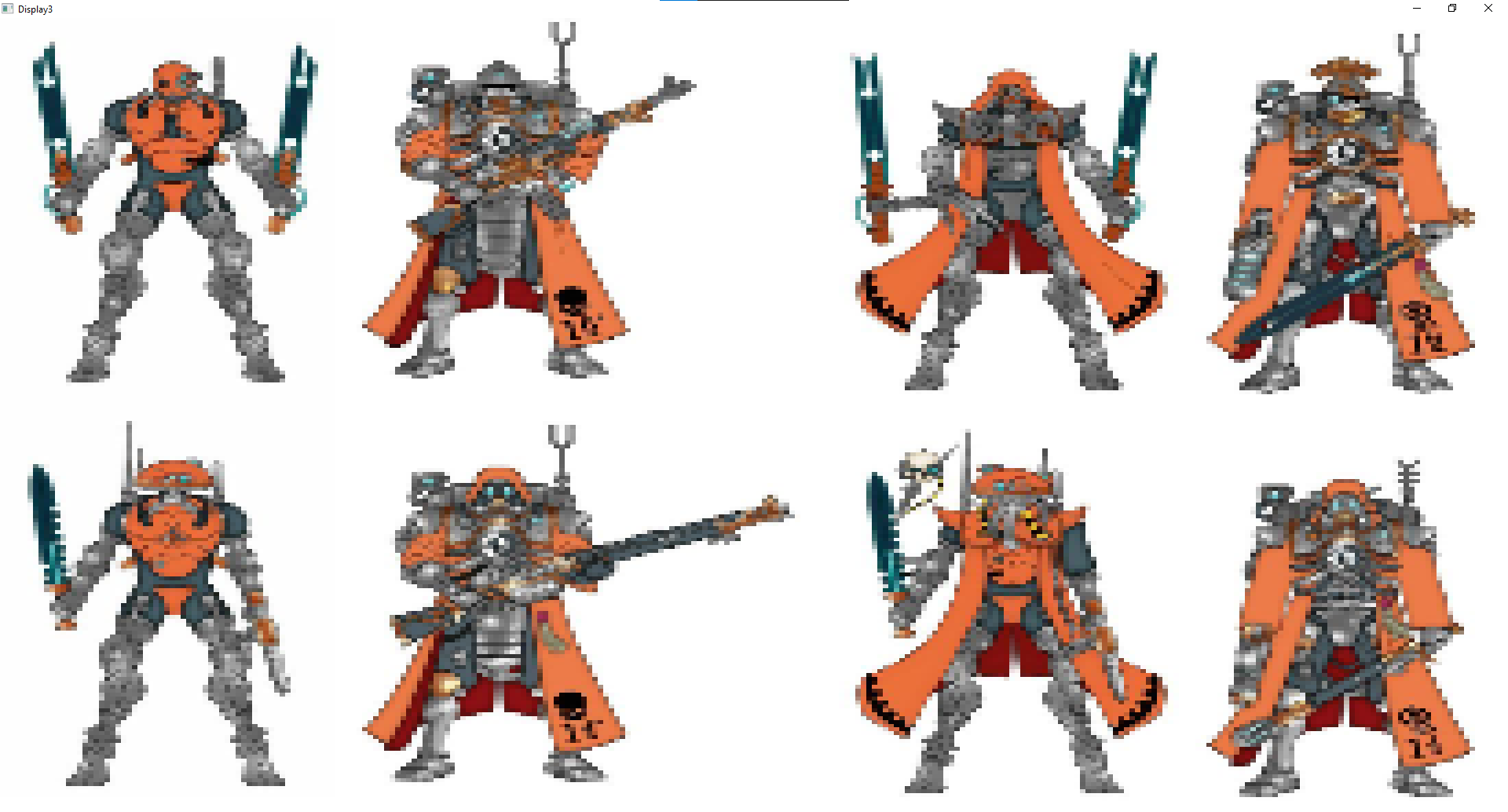
Задание 1. Установить библиотеку OpenCV.

С помощью команды pip install opencv-cv через терминал в IDE PyCharm 2022.3 библиотека OpenCV была установлена. Для работы необходимо подключить библиотеку к скрипту прописав строку

import cv2 as cv2

Задание 2. Вывести на экран изображение. Протестировать три возможных расширения, три различных флага для создания окна и три различных флага для чтения изображения.

Загрузить картинку и начать работать с ней необходим метод cv.imread(), отвечающий за загрузку картинки. Параметрами этой функции являются, обязательный аргумент указывающий путь к изображению и флаги. Для данной функции есть три основных флага cv2. IMREAD\_REDUCED\_COLOR\_4, cv2.IMREAD\_GRAYSCALE и cv2. IMREAD\_ANYCOLOR, а также дополнительные значения. На рисунках 1 и 2 представлены изображения, загруженные в сером цвете, отображенные с помощью данной команды с флагами cv2.IMREAD\_GRAYSCALE и cv2.IMREAD\_REDUCED\_ GRAYSCALE \_8 соответственно.

 Рисунок 1 – Изображение в формате IMREAD\_REDUCED\_COLOR\_4

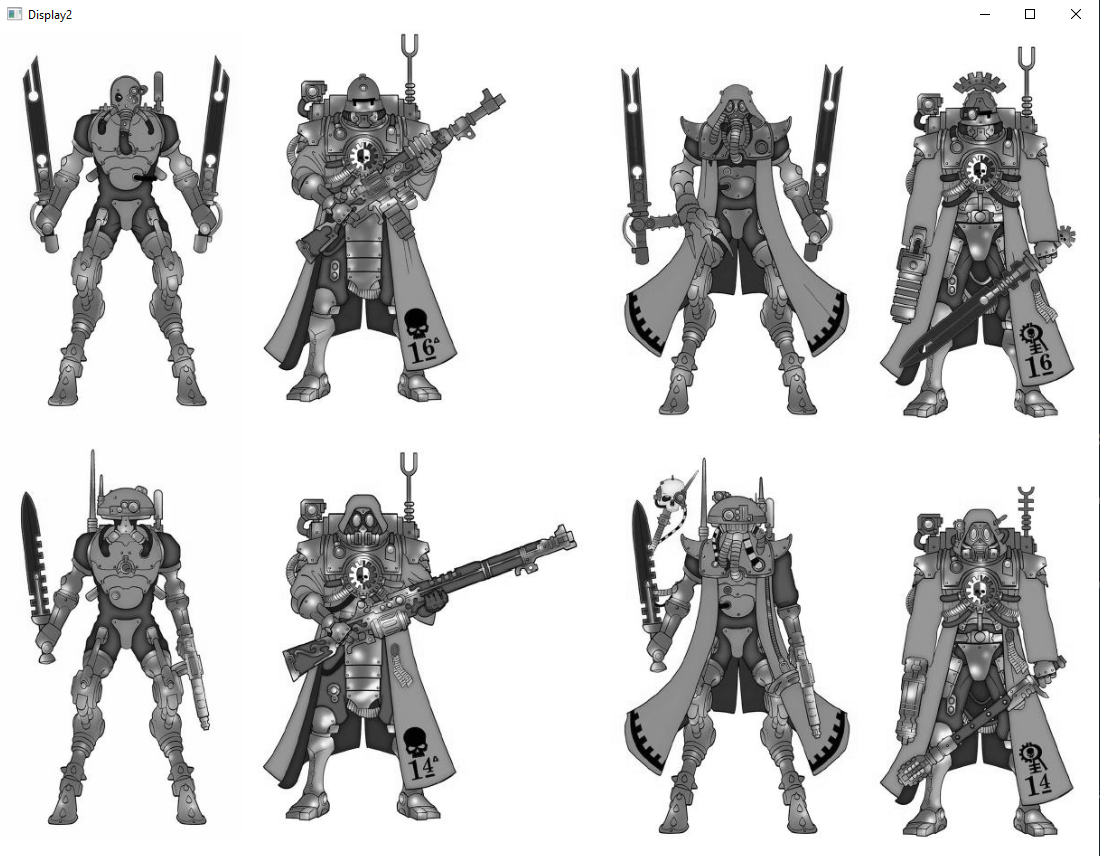


Рисунок 2 – Изображение считано в IMREAD\_GRAYSCALE

Именованные окна созданы с помощью команды сv2.namedWindow() с флагами cv2.WINDOW\_NORMAL, cv2. WINDOW\_FULLSCREEN и cv2. WINDOW\_FREERATIO, отображающим создания Windows-окна , в котором можно отображать изображения и видео.

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 веб или ip камеры можно с помощью методов класса VideoCapture. Создаётся экземпляр класса с помощью конструктора cv2.VideoCapture(), в качестве параметров передается путь к видеофайлу или устройству, с которого он будет считываться. Далее используется цикл while с командой cv2.isOpened(), т. е. пока видео открыто, цикл выполняется. Кадр из видеопотока читается с помощью функции cap.read(), возвращающей 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.resize () 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 сером цвете (снизу) и в обычном формате (сверху).

Задание 4. Записать видео из файла в другой файл.

Для записи видео из одного файла в другой был создан объект класса VideoWriter для записи видео в файл с помощью команды cv2.VideoWriter('output\_1.mov', fourcc, fps, (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. Для нахождения размера видео и частоты его кадров используем метод cap.get() с аргументами cv2.CAP\_PROP\_FRAME\_WIDTH, cv2.CAP\_PROP\_FRAME\_HEIGHT и cv2.CAP\_PROP\_FPS.

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(слева) и в формате HSV(справа).

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 нахождения правильного расположения креста, находим размеры изображения и делим их на 2.

Затем был записан массив, который будет содержать в себе координаты всех точек. Теперь запишем в массив вычисления помогающие узнать нужные координаты в виде:

    points1 = np.array([[width, height - size],

        [width - size \* np.sqrt(3)/2, height + size/2],

        [width + size \* np.sqrt(3)/2, height + size/2]], np.int32)

    points2 = np.array([[width, height + size],

        [width - size \* np.sqrt(3)/2, height - size/2],

        [width + size \* np.sqrt(3)/2, height - size/2]], np.int32)

cv2.polylines(frame,[points1, points2], isClosed=True, color=(255, 0, 0), thickness=5)

и цикл for для отрисовки всех линий, координаты которых записаны в массиве.

На рисунке 5 отображен получившийся результат.

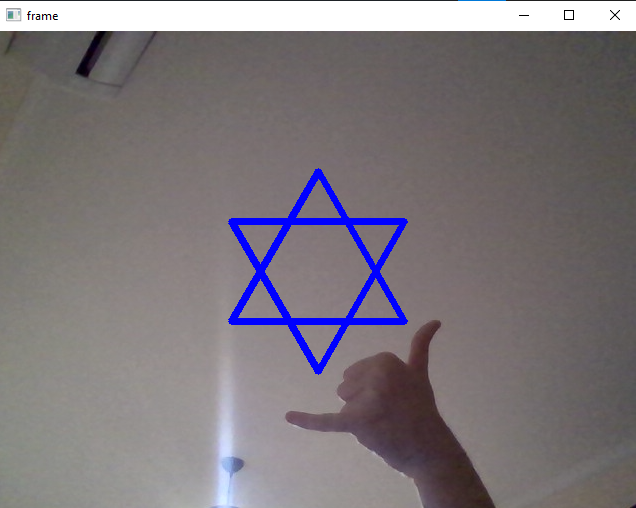


Рисунок 5 – Получившееся изображение.

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

Для выполнения этого задания код, используемый в задании 3 был преобразован под запись видео с веб камеры. В конструкторе cv2.VideoCapture(0) был указан 0, отвечающий за подключение веб-камеры, в отличие от предыдущего задания, где было указано название используемого видео.

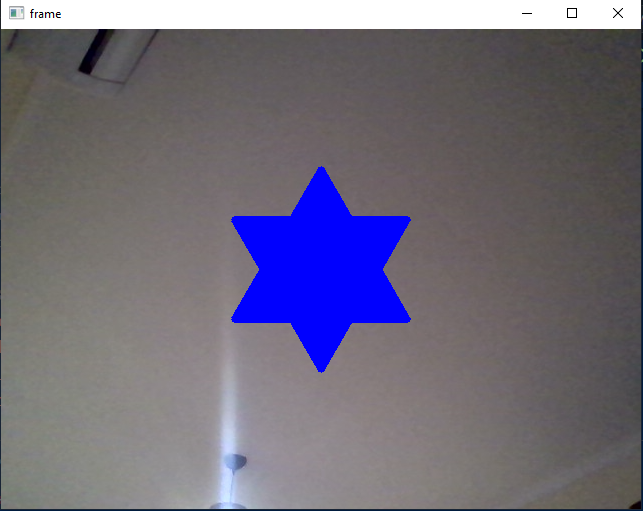
На рисунке 7 представлен кадр видео.



Рисунок 6 – Запись с вебкамеры

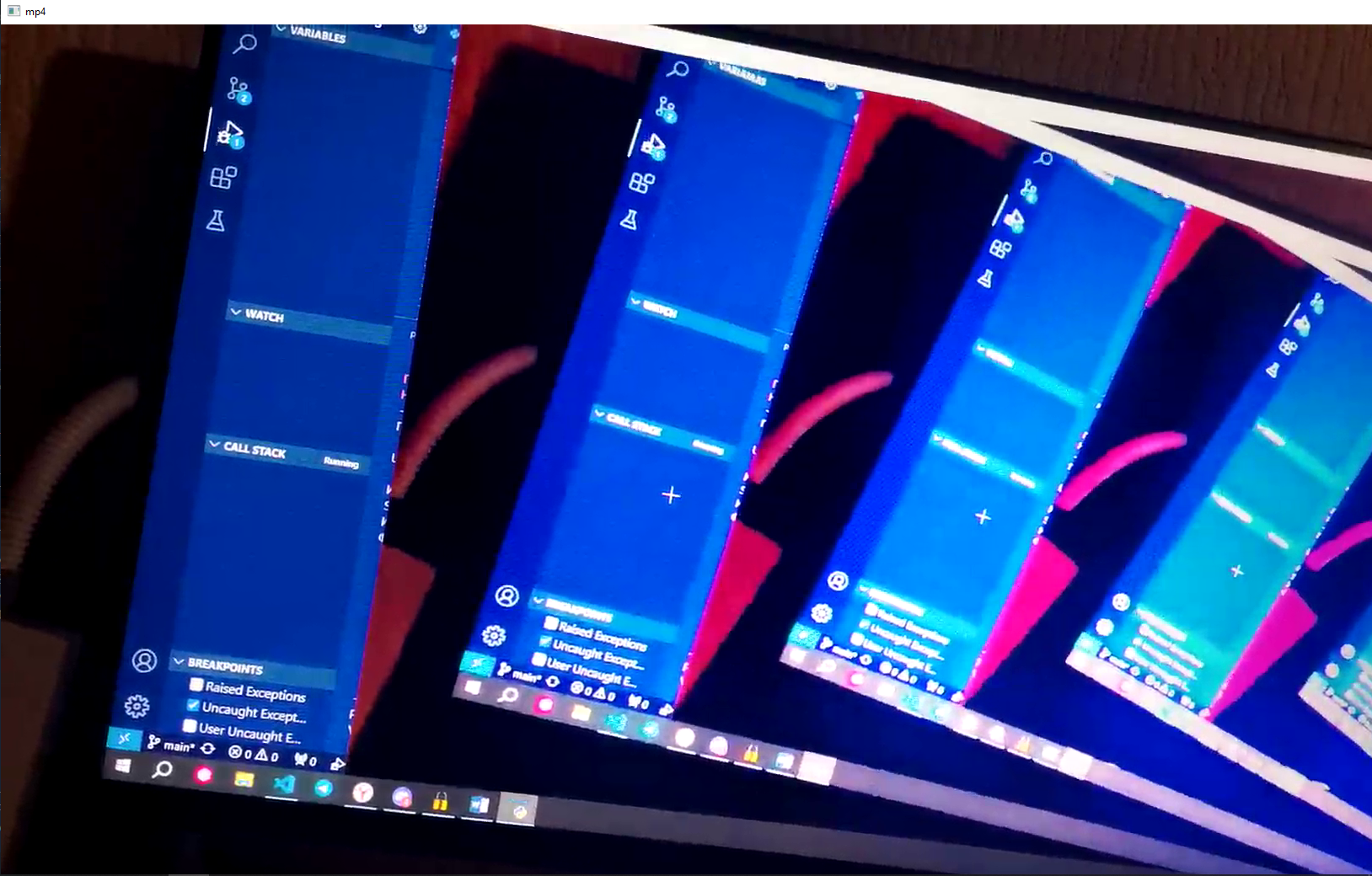
Задание 8. (самостоятельно) Залить крест одним из 3 цветов – красный, зеленый, синий по следующему правилу: НА ОСНОВАНИИ ФОРМАТА RGB определить, центральный пиксель ближе к какому из цветов красный, зеленый, синий и таким цветом заполнить крест.

Для определения ближайшего цвета из трех возможных вариантов: красный, зеленый или синий, сначала необходимо получить цвет центрального пикселя в формате RGB. Далее создаётся массив из 3 нулевых значений. Циклом for проходим по значениям цвета центрального пикселя и то значение, которое окажется максимальным заполняем число 255. Теперь заполняем весь крест полученным цветом при помощи метода cv2.fillPoly(frame, [points], color). На рисунке 7 отображен результат.

Рисунок 7 – Закрашенная звезда.

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://192.168.1.64:8080/ h264\_pcm.sdp") был передан URL-адрес потока видео. Также этот URL-адрес можно использовать в браузере на компьютере, чтобы увидеть видео с телефона. На рисунке 9 отображен кадр из видеопотока.

Рисунок 8 – Запись с камеры телефона.

**Ответы на вопросы:**

1. Опишите формат представления изображений в библиотеке OpenCv

Формат представления изображений в библиотеке OpenCV - это многомерные массивы (например, NumPy массивы) с числовыми значениями, где каждый элемент представляет интенсивность пикселя.

2. Опишите принцип отображения окон в OpenCV

Принцип отображения окон в OpenCV: с помощью функции cv2.imshow() можно создать окно для отображения изображений, а затем с помощью cv2.waitKey() организовать ожидание нажатия клавиши для закрытия окна.

3. Каким образом возможно управлять параметрами отображения окон

в OpenCV

Управление параметрами отображения окон в OpenCV происходит с помощью функций, таких как cv2.namedWindow(), cv2.resizeWindow(), cv2.moveWindow(), cv2.setWindowTitle(), которые позволяют настраивать размер, положение и заголовок окна.

4. Каким класс отвечает за работу с видеопотоком?

Класс, отвечающий за работу с видеопотоком, - это cv2.VideoCapture.

5. Откуда возможно получение видеопотока?

Видеопоток можно получить из разных источников, таких как видеокамера, видеофайл или URL-адрес видеопотока.

6. Каковы общие принципы работы с изображениями в потоке? Опишите понятие frame и принцип работы метода read()

Общие принципы работы с изображениями в потоке:

Frame: Кадр (frame) - это отдельное изображение из видеопотока.

Метод read(): Используется для чтения следующего кадра из видеопотока.

7. Что такое fourcc? Зачем применяется?

FourCC - это четырехбайтный код, используемый для определения формата видеокодирования в видеофайлах.

Он задает тип компрессии и кодек, используемый для сжатия видео.

8. Опишите основные особенности класса video\_writer?

Основные особенности класса cv2.VideoWriter:

Позволяет записывать видеофайлы.

Настраиваемые параметры, такие как формат, частота кадров и размер кадра.

Поддерживает различные кодеки для сжатия видео.

9. Что же значит эта проверка «cv2.waitKey(1) & 0xFF == 27»?

Проверка cv2.waitKey(1) & 0xFF == 27 используется для ожидания нажатия клавиши в течение 1 миллисекунды и проверки, равен ли код клавиши 27 (ESC). Если это условие выполняется, программа завершается.

10. Объяснить, зачем применяется формат HSV, рассказать значения каждого из параметров, найти и указать формулы перевода из HSV в RGB и обратно. Объяснить геометрический смысл таких преобразований.

Формат HSV (оттенок, насыщенность, значение) используется для описания цвета. Значения параметров:

Оттенок (H): Задает цвет, измеряется в градусах от 0 до 360.

Насыщенность (S): Определяет насыщенность цвета, измеряется в процентах от 0% (серый) до 100% (насыщенный).

Значение (V): Определяет яркость цвета, измеряется в процентах от 0% (черный) до 100% (белый).

Преобразование из HSV в RGB:

R = V - V \* S \* max(0, min(1, (H / 60) % 6 - 3))

G = V - V \* S \* max(0, min(1, 2 - (H / 60) % 6))

B = V - V \* S \* max(0, min(1, 2 + (H / 60) % 6 - 3))

Обратное преобразование из RGB в HSV:

V = max(R, G, B)

S = (V - min(R, G, B)) / V (если V не равно нулю)

H = 60 \* (V - R) / (V - min(R, G, B)) (для R максимального значения)

или H = 60 \* (V - G) / (V - min(R, G, B)) + 120 (для G максимального значения)

или H = 60 \* (V - B) / (V - min(R, G, B)) + 240 (для B максимального значения)

Геометрический смысл: HSV позволяет представлять цвета на цилиндрической координатной системе, где оттенок определяет угол, насыщенность - радиус, а значение - высоту.

Это более естественное представление для работы с цветами, чем RGB, что облегчает манипуляции с цветовой информацией.

**Листинг программ**

import cv2 as cv2

import numpy as np

# Задание 1-2

# img1 = cv2.imread('AdMech.png',cv2.IMREAD\_GRAYSCALE)

# img2 = cv2.imread('AdMech.png',cv2.IMREAD\_REDUCED\_COLOR\_4)

# img3 = cv2.imread('AdMech.png',cv2.IMREAD\_ANYCOLOR)

# cv2.namedWindow('Display1',cv2.WINDOW\_NORMAL)

# cv2.namedWindow('Display2',cv2.WINDOW\_FULLSCREEN)

# cv2.namedWindow('Display3',cv2.WINDOW\_FREERATIO)

# cv2.imshow('Display3',img2)

# cv2.waitKey(0)

# cv2.destroyAllWindows()

# Задание 3

# cap = cv2.VideoCapture('o-velikiy-sup-navarili.mp4',cv2.CAP\_ANY)

# while True:

#     ret, frame = cap.read()

#     if not(ret):

#         break

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

#     cv2.imshow('frame',gray)

#     #cv2.imshow('frame', frame)

#     if cv2.waitKey(1) & 0xFF == 27:

#         break

# cap.release()

# cv2.destroyAllWindows()

# Задание 4

# cap = cv2.VideoCapture("video.mp4",cv2.CAP\_ANY)

# width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))

# height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))

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

# video\_writer = cv2.VideoWriter("output.mp4", fourcc, 25, (width, height))

# while True:

#     ret, frame = cap.read()

#     if not(ret):

#         break

#     cv2.imshow('frame',frame)

#     video\_writer.write(frame)

#     if cv2.waitKey(1) & 0xFF == 27:

#         break

# cap.release()

# cv2.destroyAllWindows()

# Задание 5

# img = cv2.imread("AdMech.png",cv2.IMREAD\_COLOR)

# hsv = cv2.cvtColor(img,cv2.COLOR\_BGR2HSV)

# cv2.imshow('original\_frame',img)

# cv2.imshow('hsv\_frame',hsv)

# cv2.waitKey(0)

# cv2.destroyAllWindows()

# Задание 6

# cap=cv2.VideoCapture(0)

# width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))/2

# height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))/2

# size = 100

# while True:

#     ret, frame = cap.read()

#     if not ret:

#         break

#     points1 = np.array([[width, height - size],

#         [width - size \* np.sqrt(3)/2, height + size/2],

#         [width + size \* np.sqrt(3)/2, height + size/2]], np.int32)

#     points2 = np.array([[width, height + size],

#         [width - size \* np.sqrt(3)/2, height - size/2],

#         [width + size \* np.sqrt(3)/2, height - size/2]], np.int32)

#     cv2.polylines(frame,[points1, points2], isClosed=True, color=(255, 0, 0), thickness=5)

#     cv2.imshow('frame', frame)

#     if cv2.waitKey(1) & 0xFF == 27:

#         break

# cap.release()

# cv2.destroyAllWindows()

# Задание 7

# cap = cv2.VideoCapture(0)

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

# out = cv2.VideoWriter('output2.avi', fourcc, 20.0, (640, 480))

# while True:

#     ret, frame = cap.read()

#     if not ret:

#         break

#     out.write(frame)

#     cv2.imshow('Webcam', frame)

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

#         break

# cap.release()

# out.release()

# cv2.destroyAllWindows()

# Задание 8

# cap=cv2.VideoCapture(0)

# width = int(cap.get(cv2.CAP\_PROP\_FRAME\_WIDTH))//2

# height = int(cap.get(cv2.CAP\_PROP\_FRAME\_HEIGHT))//2

# size = 100

# while True:

#     ret, frame = cap.read()

#     if not ret:

#         break

#     points1 = np.array([[width, height - size],

#         [width - size \* np.sqrt(3)/2, height + size/2],

#         [width + size \* np.sqrt(3)/2, height + size/2]], np.int32)

#     points2 = np.array([[width, height + size],

#         [width - size \* np.sqrt(3)/2, height - size/2],

#         [width + size \* np.sqrt(3)/2, height - size/2]], np.int32)

#     cv2.polylines(frame,[points1, points2], isClosed=True, color=(255, 0, 0), thickness=5)

#     center\_pixel = frame[height, width]

#     r, g, b = center\_pixel

#     if r > g and r > b:

#         star\_color = (0, 0, 255)

#     elif g > r and g > b:

#         star\_color = (0, 255, 0)

#     else:

#         star\_color = (255, 0, 0)

#     cv2.fillPoly(frame, [points1], star\_color)

#     cv2.fillPoly(frame, [points2], star\_color)

#     cv2.imshow('frame', frame)

#     if cv2.waitKey(1) & 0xFF == 27:

#         break

# cap.release()

# cv2.destroyAllWindows()

#Задание 9

# cap=cv2.VideoCapture("rtsp://192.168.1.64:8080/h264\_pcm.sdp")

# while True:

#     ret, frame = cap.read()

#     if not(ret):

#         break

#     cv2.imshow('mp4',frame)

#     if cv2.waitKey(1) & 0xFF == 27:

#         break