**Ход работы**

1. Я установил библиотеку OpenCV для Python путем ввода команды в консоль:

pip install opencv-python

2. Далее я проверил вывод изображений различных форматов в именованных окнах с различными флагами.



Рисунок 1 – Листинг задания 2

Метод imread считывает изображение по указанному пути и применяет к нему опцию, указанную флагом во втором параметре функции:

* IMREAD\_REDUCED\_COLOR\_8 – считывает изображение в цвете с уменьшением размера в 8 раз,
* IMREAD\_REDUCED\_GRAYSCALE – считывает изображение в чёрно-белом формате,
* IMREAD\_REDUCED\_RGB - считывает изображение в формате RGB.

Метод namedWindow создает новое именованное окно с различными флагами:

* WINDOW\_AUTOSIZE – создает окно автоматически определенного фиксированного размера,
* WINDOW\_KEEPRATIO – создает окно, сохраняющее пропорции изображения при изменении размера,
* WINDOW\_FREERATIO – создает окно со свободными пропорциями изображения.

Метод imshow выводит изображение в указанное именованное окно.

Чтобы окно не закрывалось сразу, с помощью метода waitKey ожидается ввод с любой кнопки.

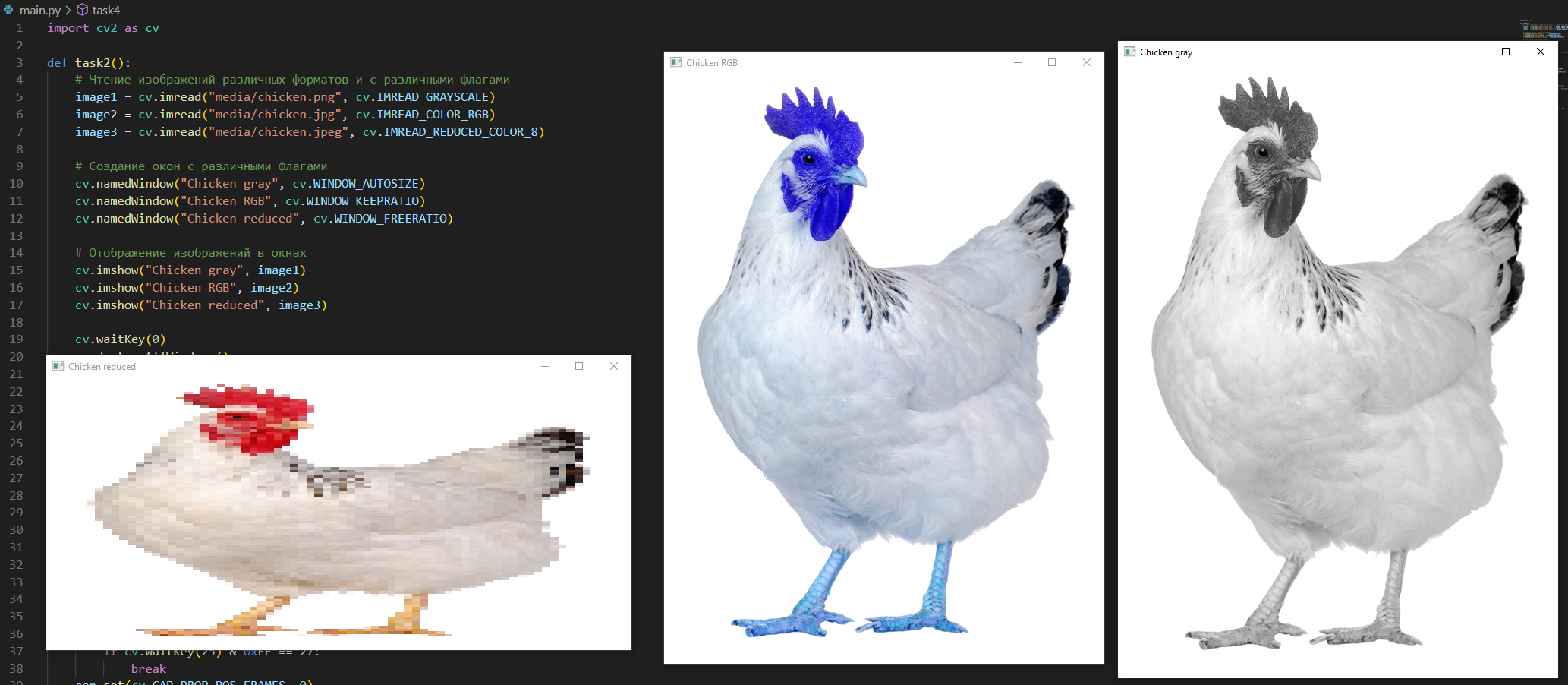


Рисунок 2 – Вывод изображений в различных форматах

3. Для отображения в окне видео в различных форматах я написал функцию, последовательно отображающую кадры считанного видео и применяющую указанные преобразования.



Рисунок 3 – Листинг задания 3

Видео считывается в объект класса VideoCapture, который предоставляет метод read, возвращающий булевое значение ret, обозначающее успешно ли считан кадр, и сам кадр, представляемый, как и обычное изображение, в виде массива NumPy в некотором формате кодирования цветов (по умолчанию BGR).

Метод waitKey ждет указанное число миллисекунд нажатия любой клавиши и возвращает при нажатии код клавиши, которая была нажата. Далее, благодаря побитовому умножению на FF берутся первые 8 бит, которые соответствуют стандартному ASCII-коду клавиши. В данном случае это 27 – Esc.

Таким образом, видео проигрывается в 40 FPS, и кадры могут быть вручную пропущены нажатием клавиши Esc.

4. Для считывания видео и перезаписи его в другой файл необходимо получить его параметры: широту, высоту и количество кадров в секунду, что можно сделать методом cv.get.

Эти параметры применяются классом VideoWriter, который принимает их в конструкторе. В целом, его конструктор принимает следующие параметры:

- filename – путь к выходному файлу

- fourcc (Four Character Code) – кодек, определяющий, как будут сжиматься и храниться кадры видео

- fps – кадры в секунду

- (width, height) – размер кадра в пикселях

- isColor – флаг, определяющий, цветное ли изображение (по умолчанию True)

При создании объекта класса VideoWriter, запись видео в файл происходит аналогично считыванию видео.

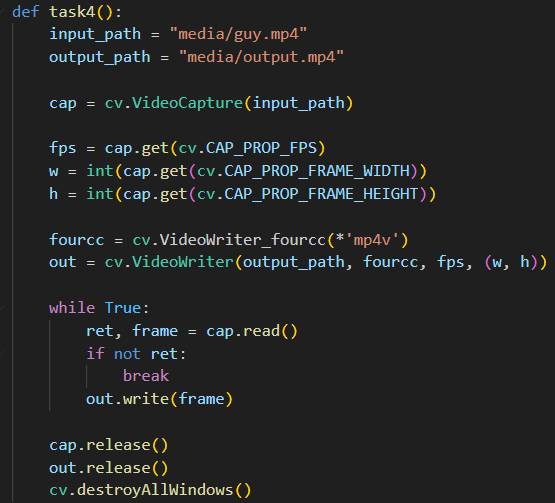


Рисунок 4 – Листинг задания 4

5. Для отображения изображения в стандартном формате и в формате HSV я использовал встроенный в OpenCV метод cvtColor, используемый для изменения цветового пространства, с параметром BGR2HSV.

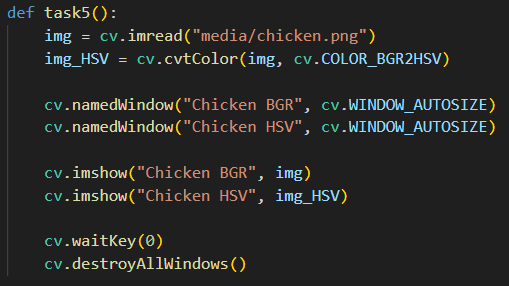


Рисунок 5 – Листинг задания 5

Таким образом, получились следующие изображения:



Рисунок 6 – Вывод изображения в формате BGR и HSV

6. Для получения изображения со встроенной веб-камеры можно воспользоваться конструктором VideoCapture с индексом 0. Таким образом у ОС запрашивается первое доступное устройство для ввода видео.

Чтобы вывести в центр изображения с камеры красный крест, воспользуемся методами библиотеки для рисования геометрического примитива rectangle. Для определения позиции прямоугольника, считаем размер изображения через функцию NumPy shape.

*Изображение выглядит как текст, снимок экрана

Содержимое, созданное искусственным интеллектом, может быть неверным.*

Рисунок 7 – Листинг задания 6

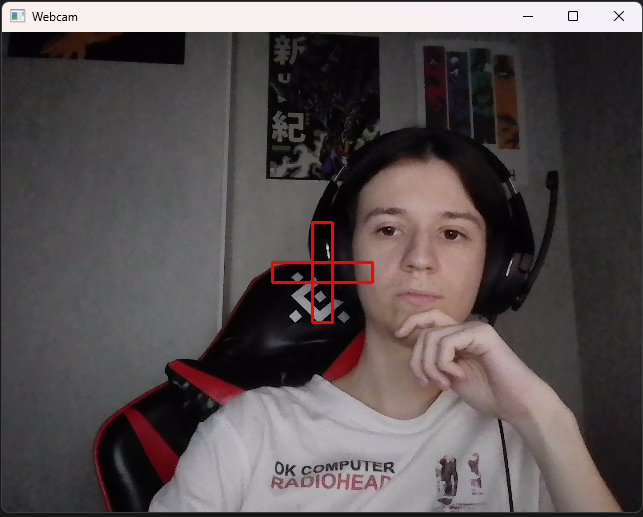


Рисунок 8 – Вывод изображения с камеры и рисование прямоугольников

7. Чтобы записать видео с камеры в файл, воспользуемся ранее использованным классом VideoWriter. Таким образом, при запуске программы изображение с камеры будет выводиться в окно и в файл, пока не будет нажата клавиша Esc.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 9 – Листинг задания 7

8. Чтобы закрасить ранее построенный на изображении крест тем цветом, к которому цвет центрального пикселя ближе, для начала определим местоположение центрального пикселя. Это точка с координатами, равными половине ширины и половине высоты.

Значения цветов (красного, синего и зеленого) далее можно получить, извлекая из кадра массив цветов, находящийся по индексу, равному данным координатам.

Далее достаточно сравнить полученные 3 значения и окрасить рисуемые прямоугольники в тот, чье значение максимально.

Изображение выглядит как текст, снимок экрана, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 10 – Листинг задания 8

Изображение выглядит как текст, Человеческое лицо, человек, Пластиковая бутылка

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как текст, Человеческое лицо, человек, безалкогольный напиток

Содержимое, созданное искусственным интеллектом, может быть неверным.

Изображение выглядит как еда, питьевая вода, напиток, Пластиковая бутылка

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 11 – Определение цвета центрального пикселя изображения

9. Далее, чтобы отобразить видео с камеры телефона, воспользуемся приложением ivCam, предоставляющим доступ к изображению с камеры мобильных устройств в одной локальной сети. Подключив телефон, его камера становится вторым в списке устройств для ввода видео, таким образом к ней можно получить доступ через VideoCapture с параметром 1.

Изображение выглядит как текст, снимок экрана, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок 12 – Листинг задания 9

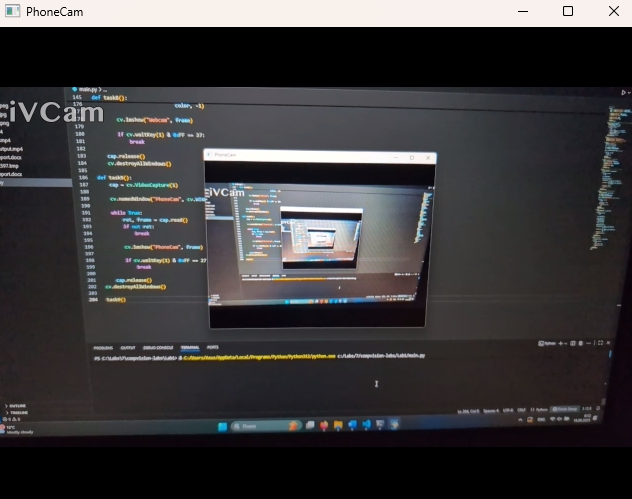


Рисунок 13 – Вывод изображения с камеры телефона