

Вопросы к лабораторной работе №7

1. Какие существуют типы изображений?

- бинарные – изображения, пиксели которого принимают только два значения: 0 и 1, что соответствует черному или белому цвету;
- полутоновые (серые или изображения в градациях серого) – диапазон значений интенсивности пикселей в формате `uint8 [0, 255]` или в формате `double [0,1]` (для языка `python` вещественные числа `float`);
- палитровые – каждому пикселу сопоставляется номер ячейки карты цветов, в карте цветов содержится описание цвета пикселя в некоторой цветовой системе (палитре);
- цветные (RGB) – пиксели непосредственно хранят информацию об интенсивностях цветного изображения, например, об интенсивности красного, зеленого, синего цвета.

По способу хранения описания изображения оно может быть:

- векторным, если изображение создается набором графических примитивов (отрезок прямой, угол, многоугольник, окружность, дуга и т. д.), из которых и формируется изображение;
- растровым, если изображение кодируется двумерным массивом, элементами которого являются интенсивности серого цвета, либо одного из цветов (красного, зеленого, синего).

2. Как осуществить считывание изображения?

Функция чтения изображения из файла `imread(,)`. Первый аргумент в скобках указывает путь к считываемому файлу, второй – флаг. Синтаксис: `img = cv2.imread(" , 1)`. Функция считывает изображение из файла и помещает его в массив `img`. Если второй аргумент равен 0, то цветное изображение в `jpg`-файле трансформируется в полутоновое (серое) изображение и по нему формируется матрица полутонового изображения, если записано 1, то

формируется матрица цветного изображения, если -1 , то изображение загружает изображение как таковое.

3. Какие значения можно присвоить флагу функции `imread(,)`?

`cv2.imread color`: загрузка цветного изображения (по умолчанию);

`cv2.imread grayscale`: загрузка изображения в режиме градаций серого;

`cv2.imread unchanged`: загрузка изображения как такового, включая альфа-канал.

4. Какая функция позволяет вывести изображение на экран?

Функция вывода изображения на экран `imshow`. Синтаксис: `cv2.imshow('image', img)` – вывод изображения на экран с именем 'image'. Можно выводить несколько изображений, но у каждого должно быть свое имя. Первый аргумент в скобках – это имя окна, вторым аргументом является массив, из которого информация выводится на экран.

5. За что отвечают команды `cv.waitKey(0)`, `cv.destroyAllWindows()`.

Первая функция позволяет задерживать изображение после его вывода на экран. В скобках указывается время в миллисекундах. Изображение остается на экране пока не сработает клавиатура. Если нажать какую-либо клавишу, программа продолжится. Если задан 0, то работа программы продолжится после нажатия клавиши.

Вторая функция уничтожает все окна, которые мы создали. Если нужно уничтожить конкретное окно, то в скобках указывается имя окна.

6. Как осуществляется запись изображения в файл?

Для создания изображения из его матрицы в виде файла используется функция `cv.imwrite(,)`. Синтаксис: `imwrite(<имя файла>.<расширение>, img)` – первый аргумент в скобках – это имя сохраняемого файла, второй аргумент

– это название матрицы изображения, с помощью которой создаем файл с выбранным расширением. Функция `imwrite` записывает матрицу бинарного, полутонового или полноцветного изображения на диск и сохраняет изображение в файле с именем

7. Какие основные свойства матрицы, какие команды позволяют их узнать?

`type(img)` – тип класса и класс данных изображения,

`img.shape` – число строк, столбцов и каналов RGB матрицы изображения,

`img.size` – количество пикселей,

`img.dtype` – формат матрицы изображения.

8. Как изменить значение пикселя по его координатам?

Необходимо установить библиотеку `numpy`, затем выполнить обращение к матрице и конкретному пикселю, например, `img[100, 150, (0)]` – обращение к пикселю матрицы `img` с координатами 100, 150 (последний аргумент обращается к интенсивности определенного цвета, 0 – синий, 1 – зеленый, 2 – красный), и присвоить ему значение в формате `[B, G, R]` – где B, R, G – интенсивность синего, красного и зеленого.

9. Как создать бинарное изображения и его негатива из цветного?

Бинарное изображение можно получить из полутонового изображения, если провести его пороговую обработку. Алгоритм бинаризации полутонового изображения таков: если значение пикселя больше порогового значения, то ему присваивается 1, если меньше, то 0.

Выполнить данную операцию позволяет функция `cv2.threshold(gray,128,255,cv2.THRESH_BINARY)`, где `gray` – исходное изображение; 128 – пороговое значение; 255 – значение, которое придаем пикселю, если его значение больше порогового.

Создать негатив из бинарного позволяет функция `cv2.THRESH_BINARY_INV`

10. Как можно выделить область на изображении?

Выделить область можно путем рисования определенной фигуры на изображении.

Рисование круга: `cv2.circle(img, center, radius, color[,thickness [,lineType]])`

Параметры:

`img` — представляет данное изображение;

`center` — центр круга;

`radius` — радиус круга;

`color` — цвет круга;

`thickness` — обозначает толщину контура круга, если она положительна.

А отрицательная толщина означает, что нужно нарисовать закрашенный круг;

`lineType` — определяет тип границы круга;

Рисование прямоугольника: `cv2.rectangle(img, pt1, pt2, color[,thickness[,lineType]])`

Параметры:

`img` — представляет собой изображение;

`pt1` — обозначает вершину прямоугольника;

`pt2` — обозначает вершину прямоугольника напротив `pt1`;

`color` — обозначает цвет прямоугольника яркости (оттенки серого);

`thickness` — представляет толщину линий, составляющих прямоугольник. Отрицательные значения (`CV_FILLED`) означают, что функция должна рисовать прямоугольник с заливкой;

`linetype` — представляет типы линии;

Рисование эллипса: `cv2.ellipse(img, center, axes, angle, startAngle, endAngle, color[, thickness[, lineType]])`

`cv2.ellipse(img, box, color[, thickness[, lineType]])`

Параметры:

`img` — представляет собой изображение;

`box` — представляет собой альтернативное представление эллипса через `RotatedRect` или `CvBox2D`. Это означает, что функция используется для рисования эллипса в изогнутом прямоугольнике;

`color` — обозначает цвет эллипса;

`angle` — обозначает угол поворота;

`startAngle` — обозначает начальный угол эллиптической дуги в градусах;

`endAngle` — обозначает конечный угол эллиптической дуги в градусах;

`thickness` — используется для рисования толщины контура дуги эллипса, если значение положительное. В противном случае это указывает, что должен быть нарисован заполненный эллипс;

`lineType` — обозначает тип границы эллипса;

Рисование линий: `cv2.line(img, pt1, pt2, color[, thickness[, lineType]])`

Параметры:

`img` — представляет собой изображение;

`pt1` — обозначает первую точку отрезка линии;

`pt2` — обозначает вторую точку отрезка;

`color` — представляет цвет линии;

`thickness` — представляет толщину линии;

`lineType` — тип линий:

Рисование полилиний: `cv2.polylines(img, polys, is_closed, color, thickness=1, lineType=8)`

Параметры:

`img` — представляет собой изображение;

`polys` — обозначает массив полигональных кривых;

`is_closed` — это флаг, который указывает, замкнуты ли нарисованные полилинии или нет;

`color` — цвет полилиний;

`thickness` — представляет толщину краев полилиний;

`lineType` — тип сегмента линии;