

## Вопросы к лабораторной работе №10

1. С помощью какой функции можно совершить изменение размера изображения?

`cv.resize (img, dim, interpolation=...)`

Первый аргумент – матрица изображения, второй `dim` либо `width`, `height` – размер изображения, третий – метод интерполяции

2. Какие существуют способы изменения размера?

– Размер нового изображения указывается в процентах (например: 50%): `scale_percent = 50`.

– Размер изображения задается вручную: `width=58, height=71`.

– Размер изображения задается с помощью коэффициента масштабирования.

3. Перечислите основные методы интерполяции.

`cv.INTER_AREA` – для сжатия,

`cv.INTER_CUBIC` и `cv.INTER_LINEAR` – для масштабирования.

По умолчанию используется метод интерполяции `cv.INTER_LINEAR`.

4. С помощью какой функции можно осуществить сдвиг изображения?

`cv2.warpAffine(src, M, dsize[, dst[, flags[, borderMode[, borderValue]]]])`

`src` - изображение.

Матрица `M` - преобразования.

`dsize` - размер выходного изображения.

`flags`-комбинация методов интерполяции (тип `int`!)

`borderMode` - режим пикселей границы (тип `int`!)

`borderValue` - (выделение) Значение заполнения границы; по умолчанию это 0.

5. С помощью какой функции можно осуществить вращение изображения?

`cv2.getRotationMatrix2D(center, angle, scale)`

center: Центр вращения

angle( $\theta$ ): угол поворота.

scale: коэффициент масштабирования

6. Что происходит при аффинной трансформации изображения?

При аффинном преобразовании все параллельные линии исходного изображения остаются параллельными и в выходном изображении.

7. Какие функции позволяют выполнить охват объекта?

Функция `cv2.drawContours()` возвращает структуру `box`, которая содержит 37 следующие аргументы: верхний левый угол (x, y), ширину, высоту, угол поворота. Чтобы нарисовать прямоугольник, нужны 4 угла прямоугольника, которые задаются функцией `cv2.boxPoints()`.

Окружность с минимальной площадью, охватывающей объект, можно нарисовать с помощью функции `cv2.minEnclosingCircle()`.

Используя функцию `cv2.ellipse()`, можно вписать изображение в эллипс с минимальной площадью.

8. Опишите процесс создания выпуклой оболочки вокруг контура

Чтобы нарисовать выпуклую оболочку вокруг контура некоторого изображения, выделяем все его крайние точки и соединяем их ломанной прямой линией. Ни одна точка изображения не должна выходить за пределы выпуклой оболочки.

Импортируем цветное изображение и трансформируем его в полутоновое изображение. Функция `Canny` выделяет контуры, а с помощью функции `cv2.findContours()` создаем иерархию контуров. Выделяем только внешние контуры изображения. Затем, используя цикл `for`, проходим по

каждому из контуров изображения. С помощью переменной `hull` создаем выпуклую оболочку сначала для первого контура, затем для каждого другого контура. В результате получим контур, охватывающий изображение.

9. Какая функция позволяет аппроксимировать контур?

Функция `cv2.approxPolyDP(cnt,epsilon,True)`.

Первый аргумент `cnt = contours [i]` – массив с координатами пикселей контура, аргумент `epsilon` задается в процентах, с уменьшением `epsilon` максимальное расстояние между ломаной прямой, аппроксимирующей контур, и самим контуром также уменьшается. Значение этого аргумента вычисляется функцией `epsilon = 0.1*cv2.arcLength(cnt,True)`.

10. Как осуществить выделение на изображении интересующей области, создание для нее отдельного изображения

Выделим на изображении интересующую нас область, заключив ее в прямоугольную рамку с помощью функции рисования `cv2.rectangle`. Фрагмент изображения, заключенный в рамке, выведем на экран. Используя функцию `.shape`, получим размер изображения и изменим его с помощью функции `cv2.resize`. Функция `cv2.getRotationMatrix2D` предназначена для поворота изображения, а функция `cv2.warpAffine` – для аффинного преобразования.