ETURoboHub

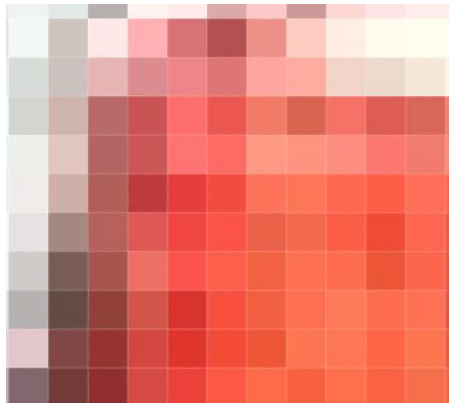
СПбГЭТУ «ЛЭТИ»
ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Введение. Основы работы с OpenCV

Знакомимся с инструментом для работы с роботами

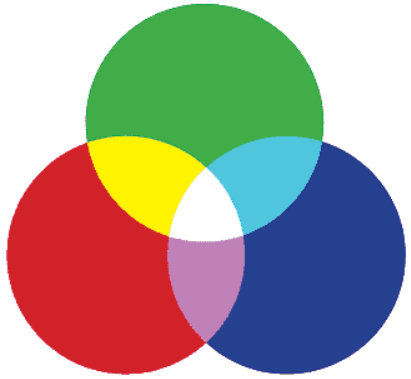
создано с
росмолодёжь
гранты

Введение в техническое зрение



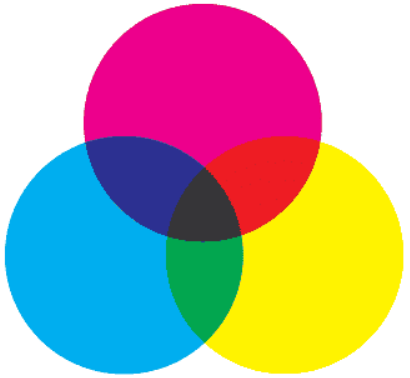
Виды цветовых моделей

RGB

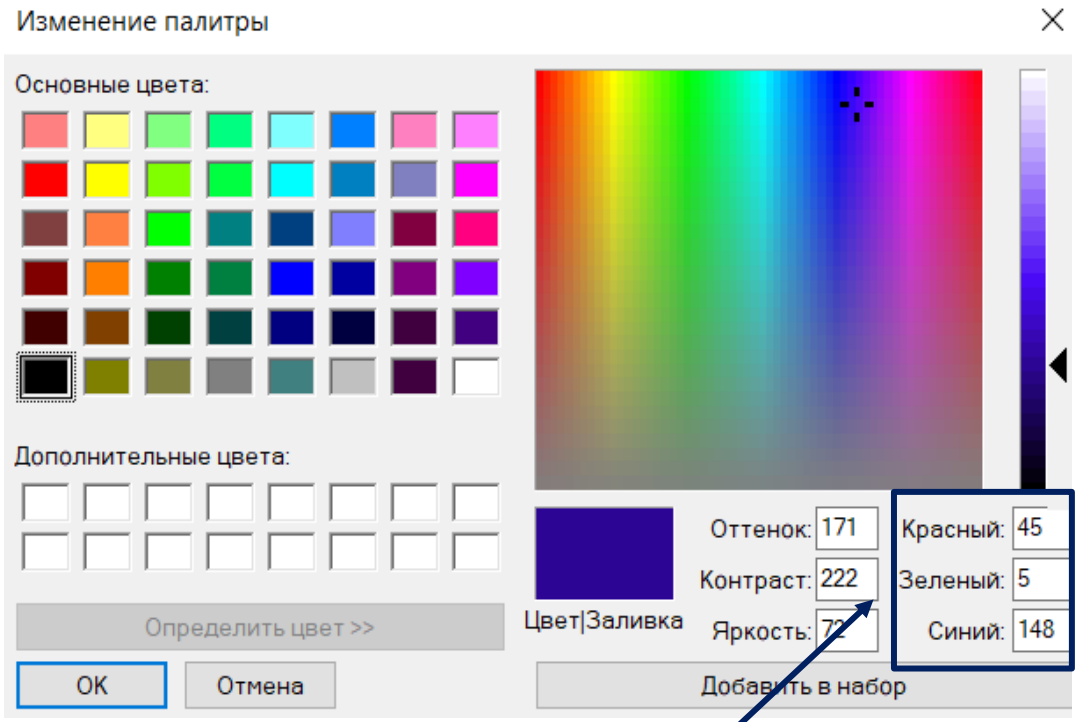


Red, Green, Blue
[Additive Colors]

CMYK



Cyan, Magenta, Yellow
[Subtractive Colors]



Представление цвета в формате RGB

Техническое зрение

Техническое зрение – это дисциплина, являющаяся частью искусственного интеллекта.

OpenCV

Библиотека компьютерного зрения и машинного обучения с открытым исходным кодом. В неё входят более 2500 алгоритмов, в которых есть как классические, так и современные алгоритмы для компьютерного зрения и машинного обучения.


```
pip install opencv-python
import cv2
```

Чтение и запись изображения

Чтение

```
img = cv2.imread(
    path, # путь до
изображения
    flags=cv2.IMREAD_CO
LOR # параметр(ы)
чтения )
```

Запись

```
res = cv2.imwrite(
    filename, # путь до
файла сохранения img, #
переменная, хранящая
изображение
    flags # параметр(ы)
сохранения )
```

Чтение изображения cv2.imread

cv2.IMREAD_COLOR - преобразует изображение в 3-канальное цветное изображение BGR.

cv2.IMREAD_GRAYSCALE - преобразует изображение в одноканальное изображение в оттенках серого (внутреннее преобразование кодека).

cv2.IMREAD_UNCHANGED - возвращает загруженное изображение как есть.

IMREAD_REDUCED_GRAYSCALE_* - преобразовать изображение в одноканальное изображение в оттенках серого и уменьшить размер изображения на **.

IMREAD_REDUCED_COLOR_* - преобразовать изображение в 3-канальное цветное изображение BGR и уменьшить размер изображения на **.

*** - 2, 4, 8**

**** -1/2, ¼, 1/8**

создано с росмолодёжь гранты

Отображение во время выполнения программы

Создание окна для отображения изображения/видео

```
cv2.namedWindow( name, # имя окна
flags=cv2.WINDOW_AUTOSIZE #
параметр(ы) окна )
```

Отображение изображения/видео в окне

```
cv2.imshow( winName, # имя окна
img # переменная, содержащая
изображение )
```

```
key = cv2.waitKey(
time # время ожидания нажатия
)
```

*Используется для того, чтобы выведенное изображение не пропало сразу на экране
**Если time< 1, то изображение будет отображаться сколь угодно долго

Библиотека Numpy

Для работы с массивами в Python будет использоваться библиотека **Numpy**.

Для импортирования библиотеки:

```
import numpy as np
```

Создание массивов, заданных размеров

Создание массива заданного размера, инициализированного нулями

```
res = np.zeros(
shape, # размер массива
dtype=np.float64, # тип данных
order='C', # тип хранения
многомерных данных
)
```

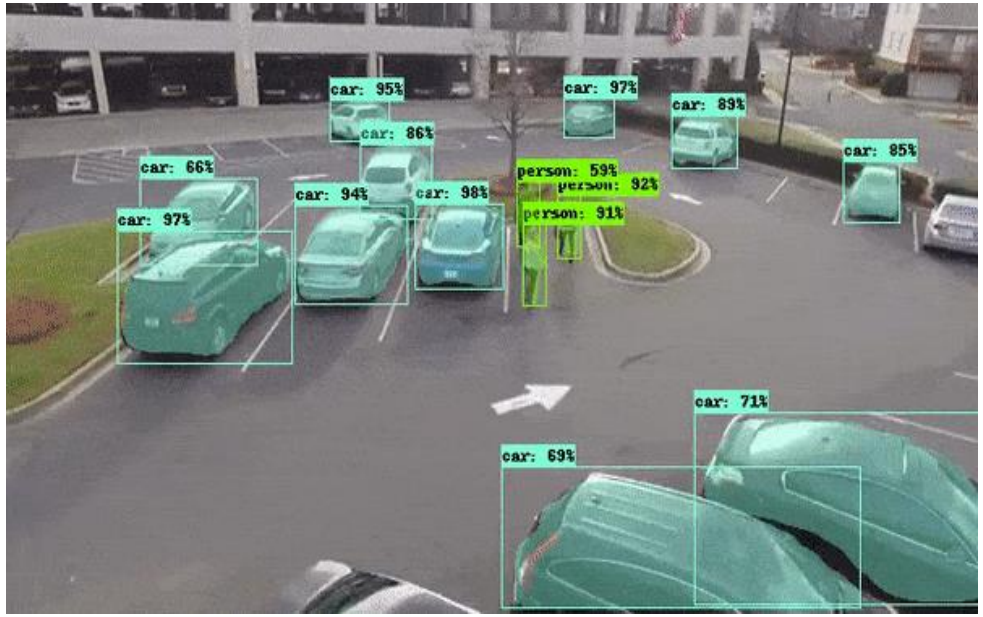
Создание массива заданного размера, инициализированного любыми числами

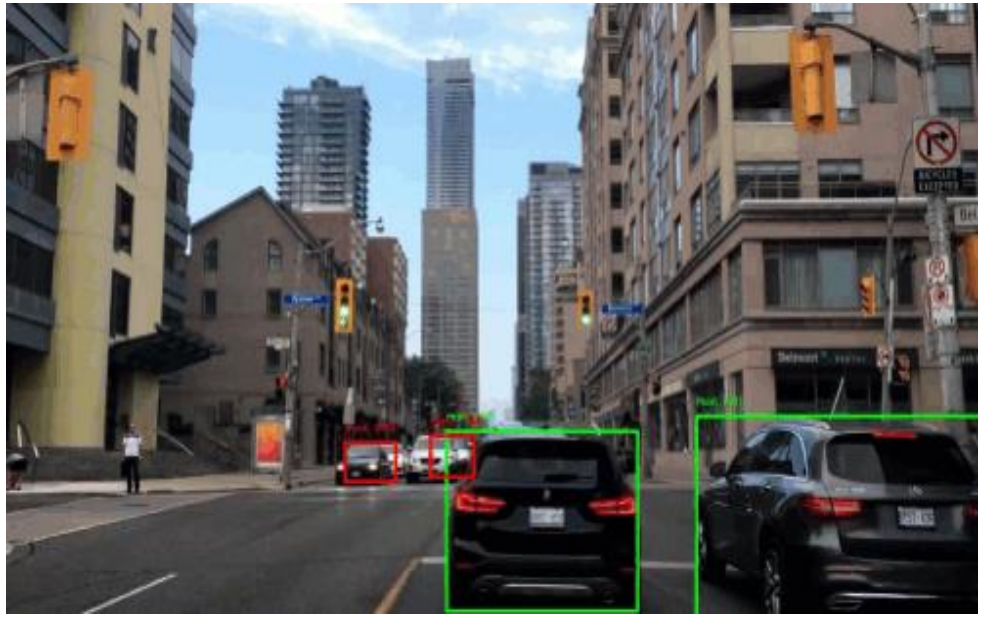
```
res = np.full(
shape, # размер массива
val, # значение для заполнения
dtype, # тип данных
order # тип хранения многомерных
данных
)
```

Функции для рисования и подписи объектов

Линия	• **cv2.line**(переменная, в которой хранится изображение, координаты начала (x1,y1) и конца (x2,y2), (цвет в формате BGR), толщина линии, тип линии)
Стрелка	• **cv2.arrowedLine**(переменная, в которой хранится изображение, координаты начала (x1,y1) и конца (x2,y2), (цвет в формате BGR), толщина линии, тип линии)
Прямоугольник	• **cv2.rectangle**(переменная, в которой хранится изображение,координаты левого верхнего угла (x1,y1),координаты правого нижнего угла (x2,y2),(цвет в формате BGR),толщина линии, тип линии)
Окружность	• **cv2.circle**(переменная, в которой хранится изображение,координаты центра (x,y),радиус,(цвет в формате BGR),толщина линии, тип линии)
Текст	• **cv2.putText**(переменная, в которой хранится изображение,"Ваш текст",координаты (x,y),шрифт,размер шрифта,(цвет в формате BGR), толщина линии, тип линии)

Класс cv2.VideoCapture

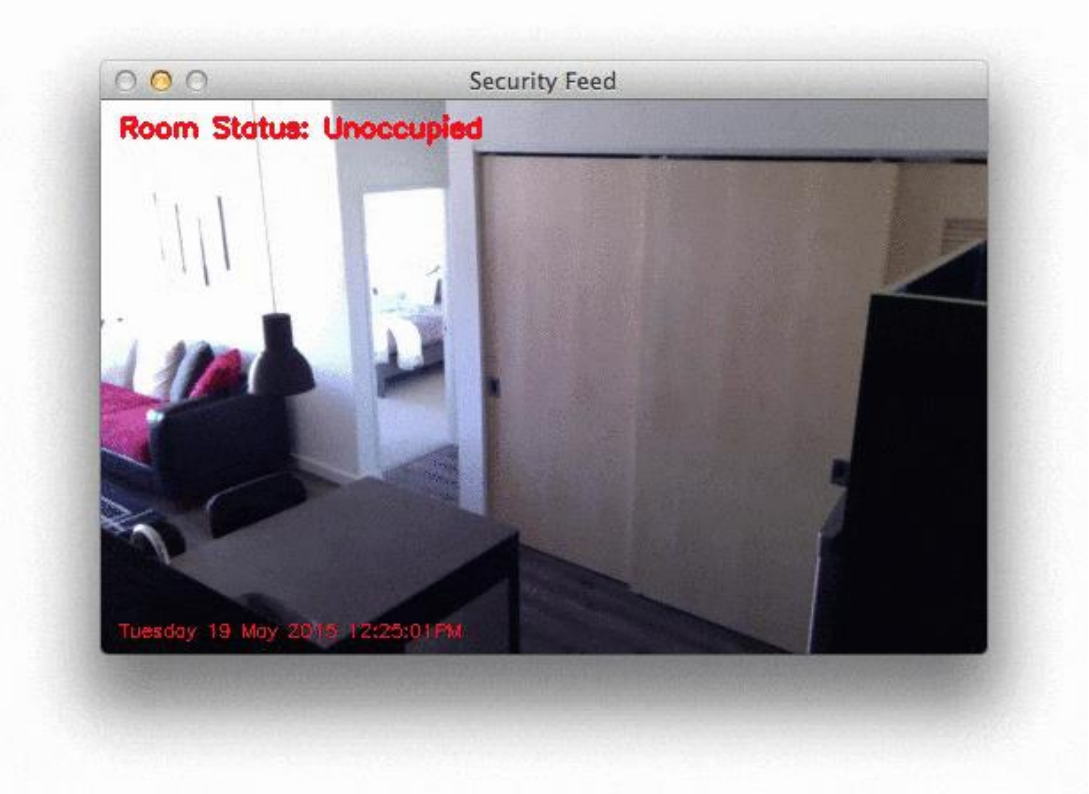




cap.VideoCapture(path), где path – путь для файла
cap.VideoCapture(n), где n – номер камеры, подключенной к компьютеру.

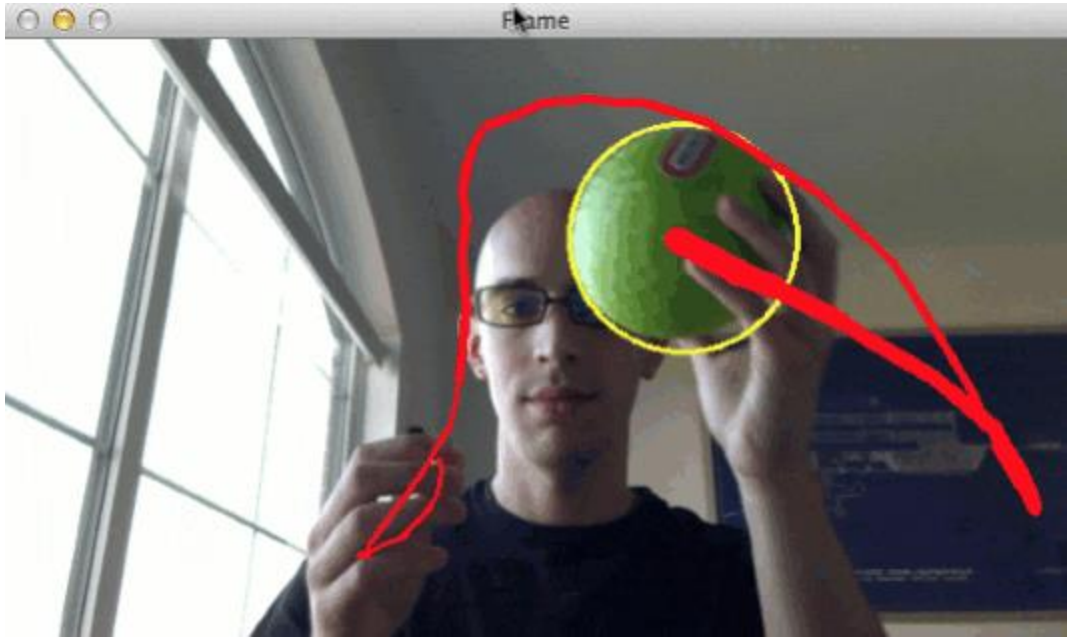
Функция cv2.isOpened()

while (cap.isOpened()):



Функция **isOpened()** объекта класса **VideoCapture()** каждый раз при обращении будет возвращать *True* пока видео не дойдет до конца.

Метод cap.read()



Для чтения видео используется метод cap.read()

Функция .release() и .destroyAllWindows()

После того, как мы закончим использовать камеру, ее нужно «освободить». Если мы этого не сделаем, то в следующий раз при попытке ее использовать, вы получите ошибку.

cap.release()

Эта функция противоположна функции чтения видео cap.read(), что эквивалентно отключению открытого видео.

cap.destroyAllWindows()

Эта функция используется для закрытия всех окон.

Класс cv2.VideoWriter

cv2.VideoWriter(filename, fourcc, fps, frame_size)

- **filename**: имя выходного видео.
- **Fourcc**: 4-значный код кодека, используемый для сжатия кадров.
- **fps**: Частота кадров выходного видео.
- **frame_size**: Размер выходных видеокадров (ширина, высота).

Параметры объекта cv2.VideoCapture

CAP_PROP_POS_MSEC =0, //!< Текущая позиция видеофайла в миллисекундах.

CAP_PROP_POS_FRAMES =1, //!< на основе 0 индекс кадра, который будет декодирован/захвачен следующим.

CAP_PROP_POS_AVI_RATIO =2, //!< Относительное положение видеофайла: 0=начало фильма, 1=конец фильма.

CAP_PROP_FRAME_WIDTH =3, //!< Ширина кадров в видеопотоке.

CAP_PROP_FRAME_HEIGHT =4, //!< Высота кадров в видеопотоке.

CAP_PROP_FPS =5, //!< Частота кадров.

CAP_PROP_FOURCC =6, //!< 4-значный код кодека. см. VideoWriter::fourcc .

CAP_PROP_FRAME_COUNT =7, //!< Количество кадров в видеофайле.

CAP_PROP_BRIGHTNESS =10, //!< Яркость изображения (только для тех камер, которые поддерживают).

CAP_PROP_CONTRAST =11, //!< Контраст изображения (только для камер).

CAP_PROP_SATURATION =12, //!< Насыщенность изображения (только для камер).

CAP_PROP_HUE =13, //!< Оттенок изображения (только для камер).

CAP_PROP_GAIN =14, //!< Усиление изображения (только для тех камер, которые поддерживают).

CAP_PROP_EXPOSURE =15, //!< Экспозиция (только для тех камер, которые поддерживают).

CAP_PROP_CONVERT_RGB =16, //!< Логические флаги, указывающие, следует ли конвертировать изображения в RGB.

Пример

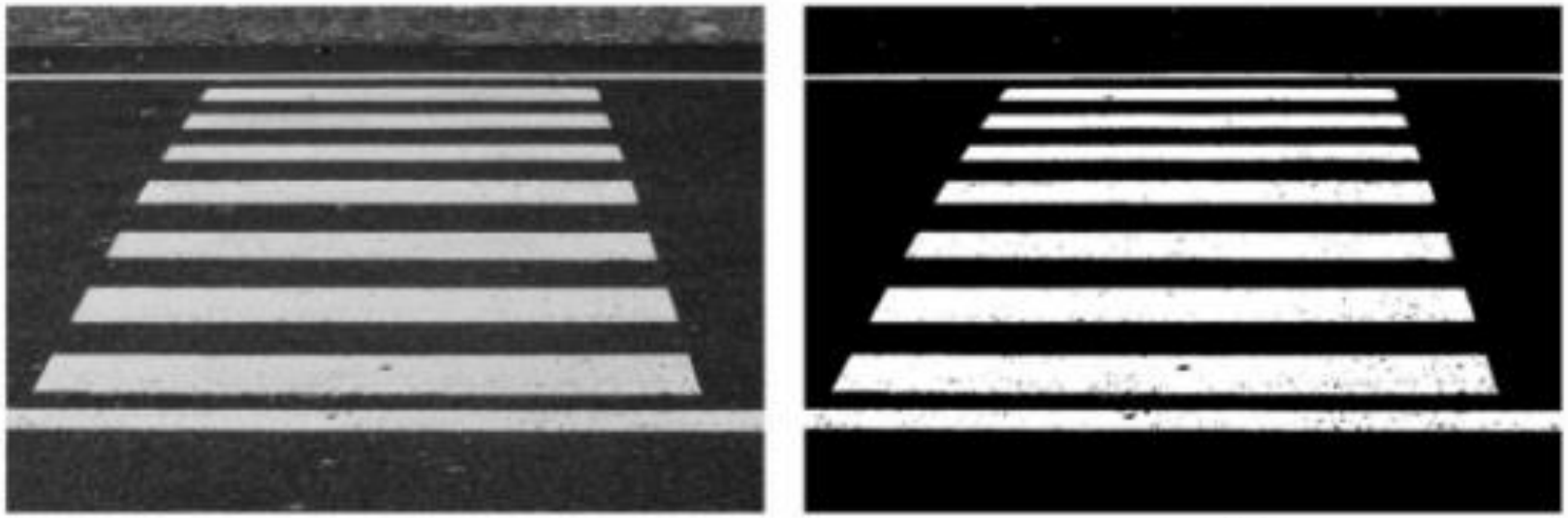
```
import cv2
cap = cv2.VideoCapture(0)
if not cap.isOpened(): #проверяем, успешно ли подключение к камере
    raise IOError("Ошибка") # Если нет, выводим сообщение об ошибке
while True:
    ret, frame = cap.read()#считываем кадр
    cv2.imshow('Video', frame) #отображаем кадр в окне Video
    c = cv2.waitKey(1)
    if c == 27: # Если нажата клавиша Escape
        break #выходим из цикла
cap.release() #освобождаем объект
cv2.destroyAllWindows() #Закрываем все открытые окна
```

Пример

```
import cv2
# Инициализация видео
video_cap = cv2.VideoCapture(0)

# Захват ширины и высоты кадра, частоты
frame_width = int(video_cap.get(3))
print(frame_width)
frame_height = int(video_cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
fps = int(video_cap.get(cv2.CAP_PROP_FPS))

# Инициализация видеокодека и объекта для записи видео
fourcc = cv2.VideoWriter_fourcc(*'XVID')
output = cv2.VideoWriter('output.mp4', fourcc, fps, (frame_width, frame_height))

while True:
    success, frame = video_cap.read()
    cv2.imshow("frame", frame)
    # write the frame to the output file
    output.write(frame)
    if cv2.waitKey(1) == ord('q'):
        break
```

Пороговые преобразования

Можно заметить, что часть пикселей фона получила максимальное значение яркости (белый цвет), а часть пикселей разметки – минимальное (черный цвет).



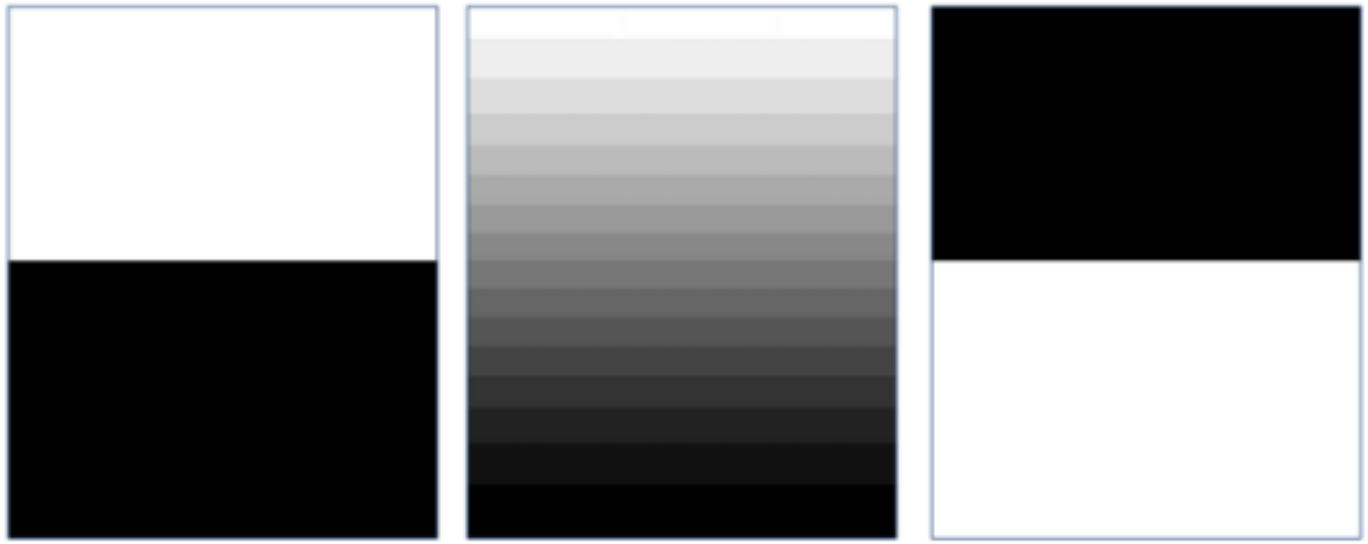
а *б*

Пороговые преобразования

$$I_2(x,y) = \begin{cases} \text{действие 1}, & I_1(x,y) > T \\ \text{действие 2}, & I_1(x,y) \le T \end{cases},$$

$$I_2(x,y) = \begin{cases} 255, & I_1(x,y) > 170 \\ 0, & I_1(x,y) \le 170 \end{cases},$$

Бинаризация

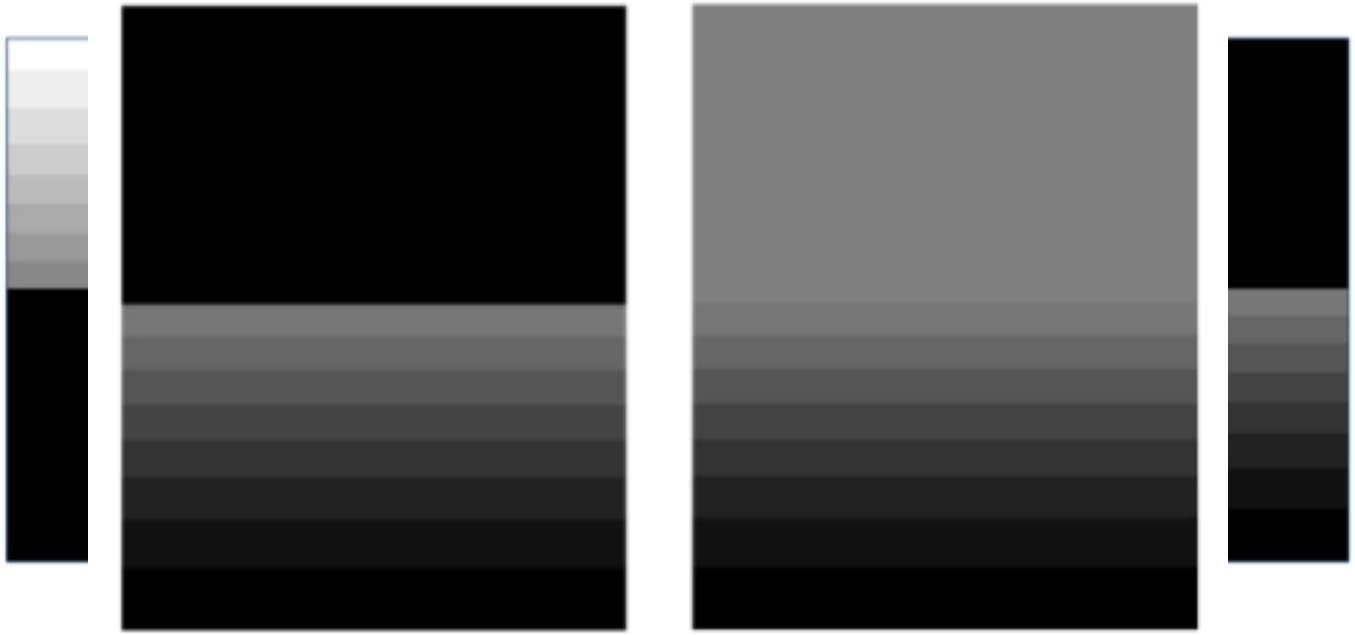


а *б* *в*

```
ЕСЛИ яркость > T ТО
    действие 1
ИНАЧЕ
    действие 2
КОНЕЦ
```

Бинарным будем считать изображение, яркость пикселей которого принимает только два значения – 0 и 255.

Обнуление яркости части пикселей



а *б*

$$I_2(x,y) = \begin{cases} I_1(x,y), & I_1(x,y) > T \\ 0, & I_1(x,y) \le T \end{cases},$$

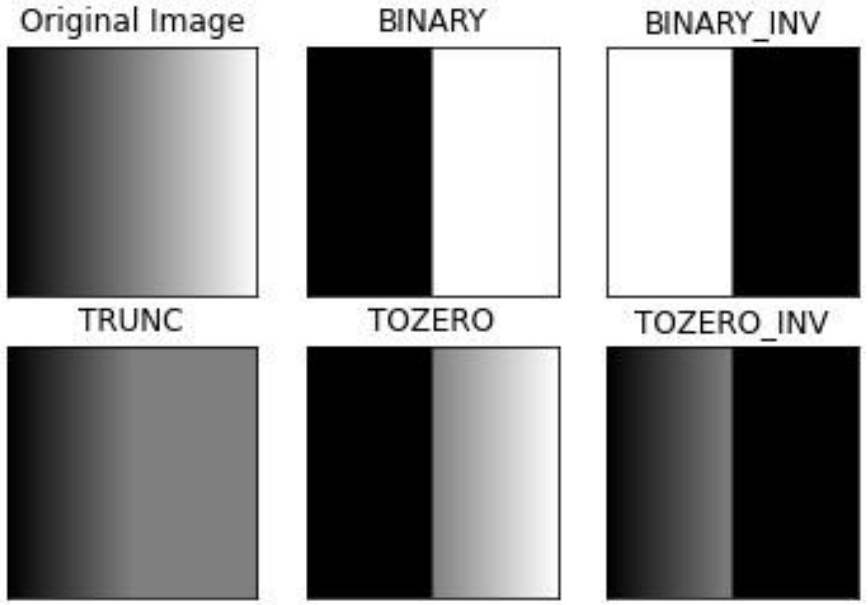
Пороговые преобразования в OpenCV

Thresholding – это **метод сегментации изображений, в общем он используется для создания бинарных изображений**. Thresholding бывает двух типов, а именно: простой порог и адаптивный порог.

```
res = cv2.threshold(
        img,          # входное изображение
        threshold,    # порог
        maxVal,       # максимально возможное значение яркости
        type          # тип преобразования
    )
res = threshold_new, new_img
```

Пороговые преобразования в OpenCV

1) **cv2.THRESH_BINARI** – бинаризация изображения
2) **cv2.THRESH_BINARY_INV** – инверсная бинаризация изображения
3) **cv2.THRESH_TOZERO** – обнуление яркости части пикселей изображения
4) **cv2.THRESH_TOZERO_INV** – инверсное обнуление яркости части пикселей изображения
5) **cv2.THRESH_TRUNC** – ограничение яркости части пикселей изображения



Адаптивное вычисление порога в OpenCV

```
img_new = cv2.adaptiveThreshold(
    img,                # входное изображение
    maxVal,             # максимально возможное значение яркости
    adaptiveMethod,     # метод расчета порога
    type,               # тип преобразования
    blockSize,          # размер окна
    C,                  # константа
    )
```

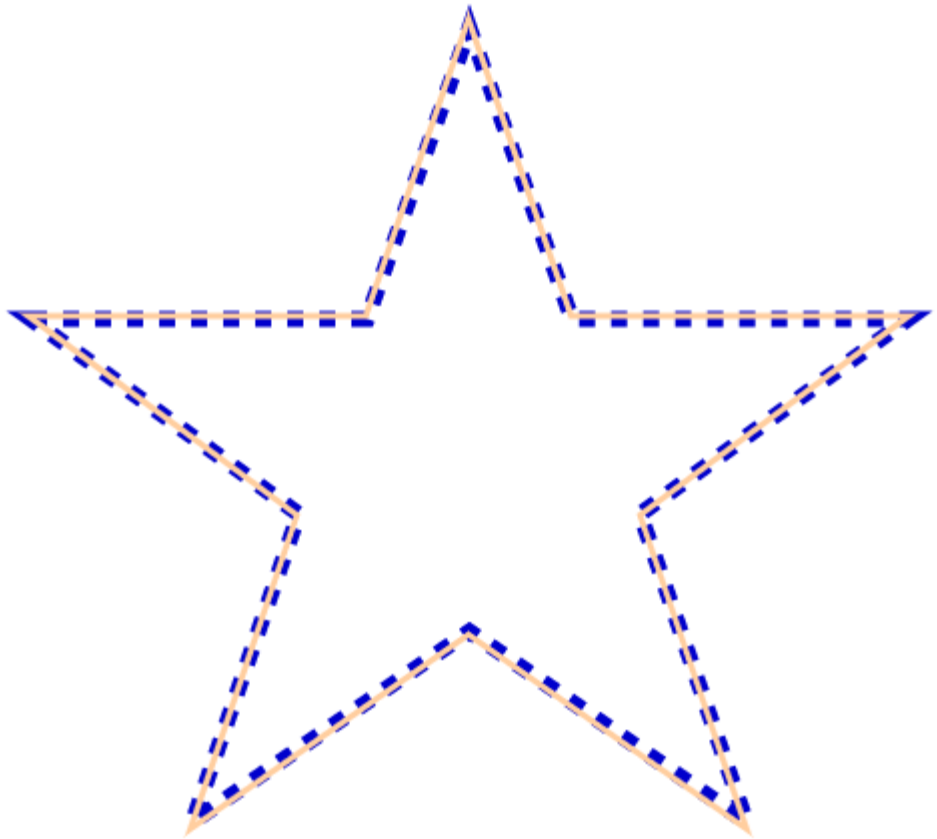
Принцип расчета очень похож на то, как меняется значение пикселя при наложении фильтра. Для расчета в аргумент **adaptiveMethod** можно передать два значения:

1) cv2. ADAPTIVE_THRESH_MEAN_C;

2) cv2.ADAPTIVE_THRESH_GAUSSIAN_C.

Что такое контур?

Контур - кривая, последовательно соединяющая все точки (вдоль границы), имеющие одинаковый цвет или яркость.



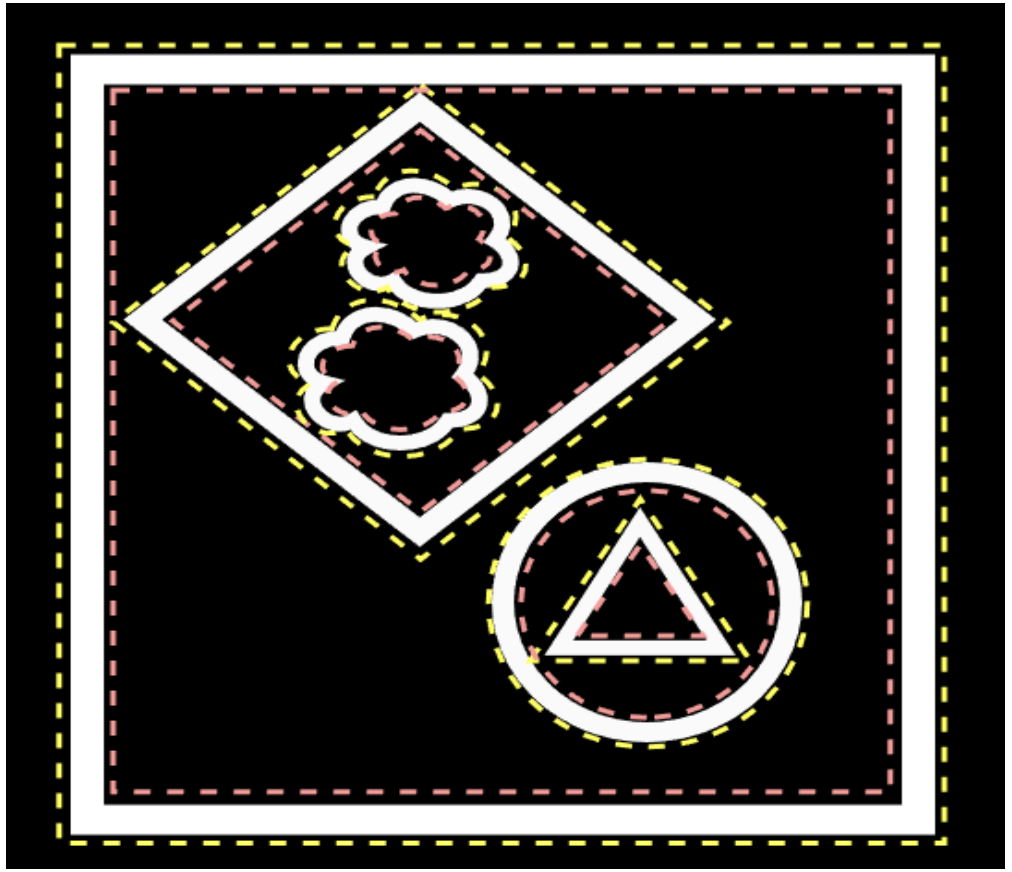
Нахождение контуров в OpenCV

```
contours, hierarchy = cv.findContours(
        img, # исходное изображение
        mode, # взаимное расположение контуров
        method # метод аппроксимации точек контур
)
```

FindContours возвращает две переменные – список всех контуров с координатами точек, по которым он строится (contours) и взаимное расположение контуров (hierarcy).

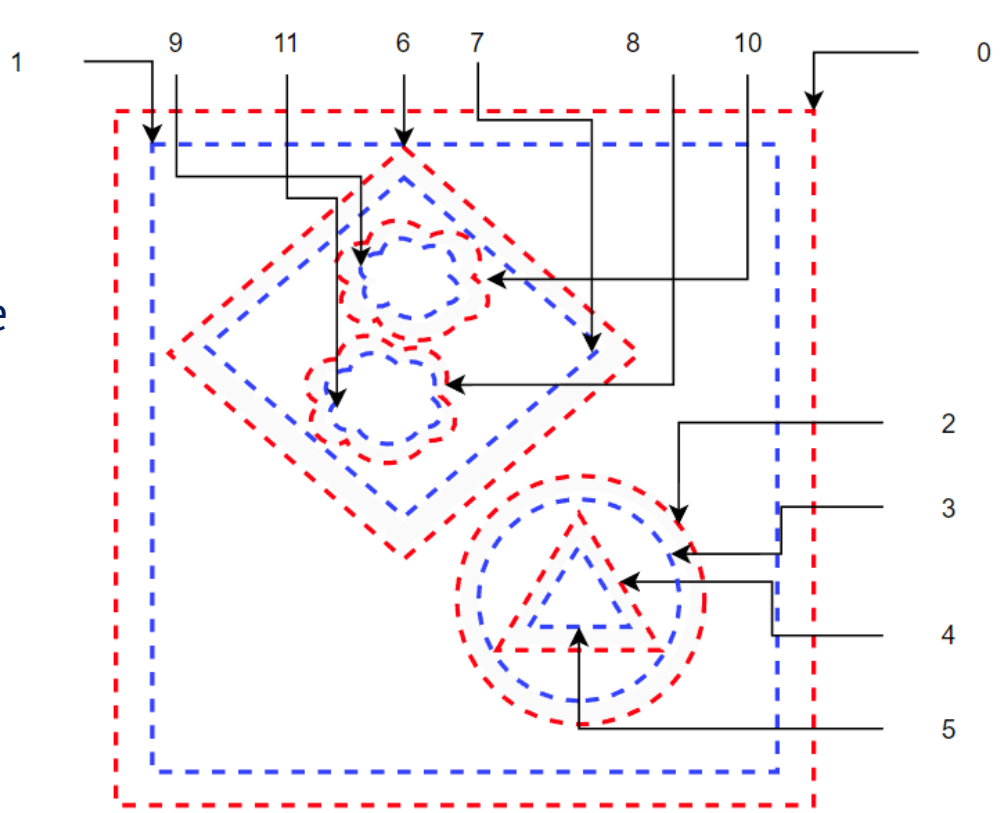
Особенности работы функции

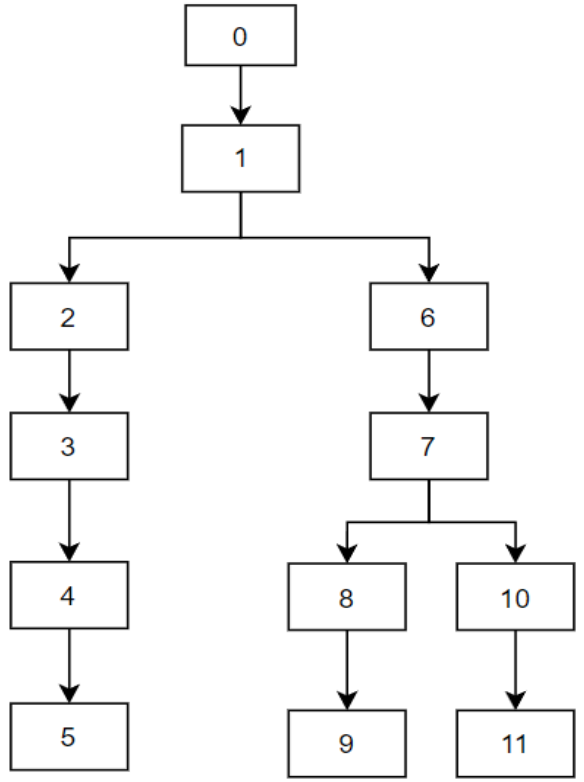
Передаваемое изображение должно быть бинарным. Значение всех пикселей, не являющимися границей, должно быть равно **0**, всех пикселей, являющимися границей – **255**.



Вложенность контуров

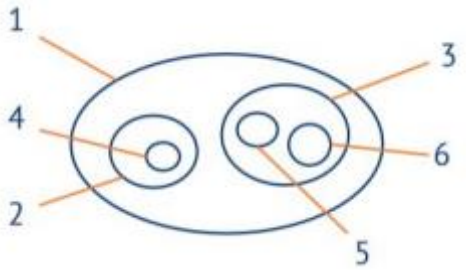
OpenCV может собрать найденные контуры в дерево контуров, которое отражает отношение вложенности контуров в своей структуре.

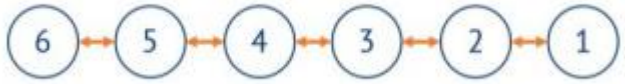




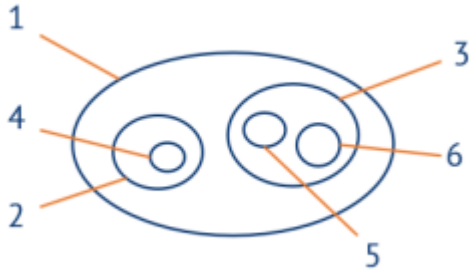
Режимы возврата контуров

1. RETR_LIST

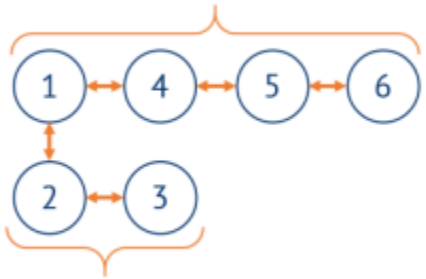




2. RETR_CCOMP

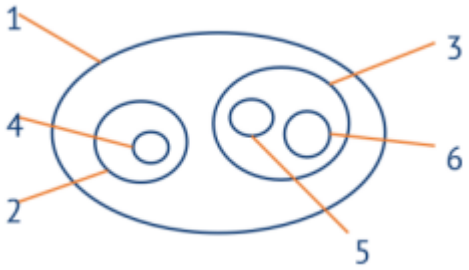


внешние контуры

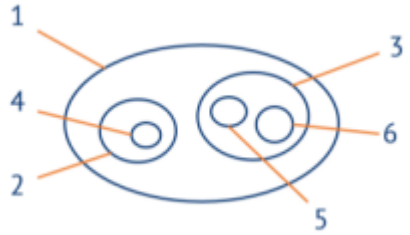


внутренние контуры

3. RETR_EXTERNAL



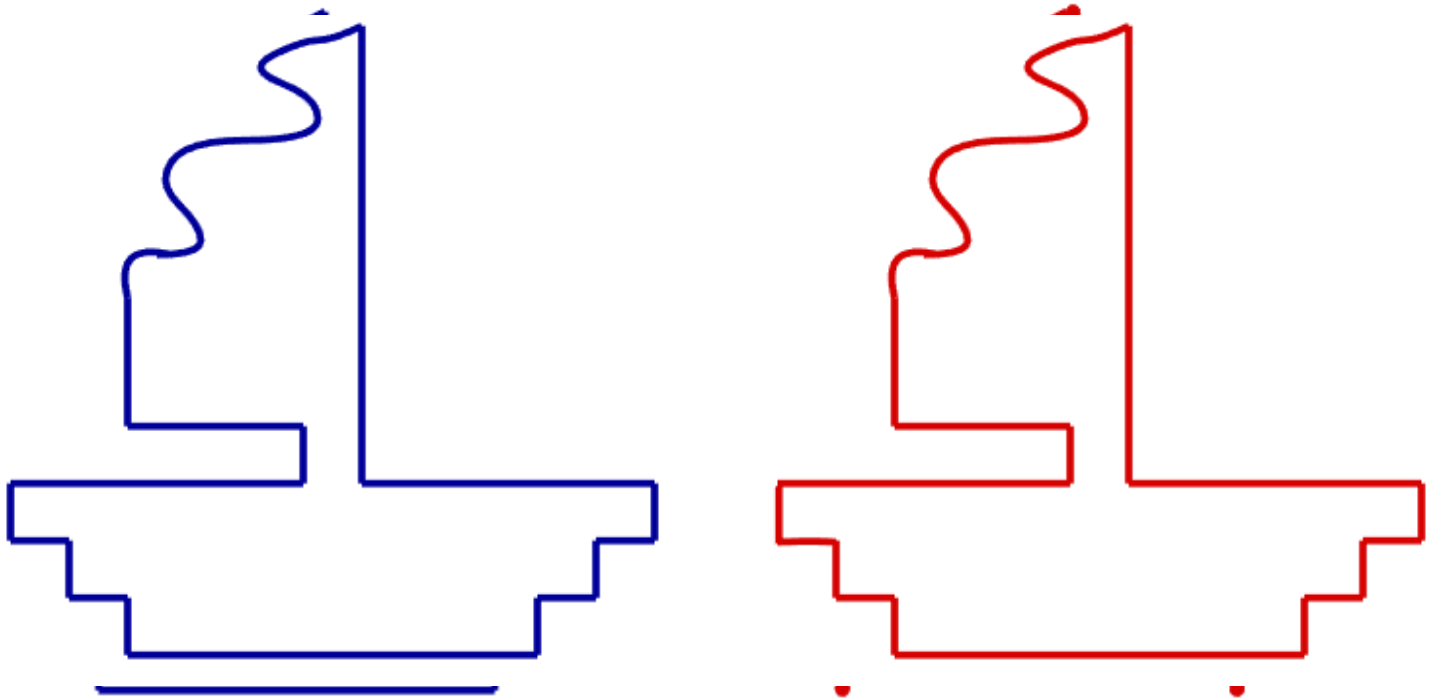
4. RETR_TREE



создано с
росмолодёжь
гранты

Способы хранения точек контуров

1. **CHAIN_APPROX_SIMPLE**
2. **CHAIN_APPROX_NONE**



Рисование контуров

```
cv2.drawContours(
img,       # исходное изображение
contours,    # список контуров
contourIdx, # индекс контура для рисования
color,            # цвет для рисования
thickness, # толщина линии для рисования
lineType    # тип линии для рисования
)
```

Свойства контуров

Функция	Описание
area = cv2.contourArea(contour**)**	Площадь
perimeter = cv2.arcLength(contour**,True)**	Длина контура
(x,y),radius = cv2.minEnclosingCircle(contour**)**	Описанная окружность
x,y,w,h = cv2.boundingRect(contour**)**	Ограничивающий прямоугольник