

**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ  
Федеральное государственное автономное  
образовательное учреждение высшего образования  
«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»**

**Кафедра инфокоммуникаций**

**Отчет по лабораторной работе № 3.9**

**«Бинарные изображения, основные характеристики бинарных  
изображений»**

**по дисциплине «Технологии распознавания образов»**

Выполнил студент группы

ПИЖ-б-о-21-1

Зиберов Александр

« » мая 2023 г.

Подпись студента \_\_\_\_\_

Работа защищена

« » \_\_\_\_\_ 20\_\_ г.

Проверил Воронкин Р.А. \_\_\_\_\_  
(подпись)

Ставрополь, 2023

## Цель работы:

Изучение методов цифровой обработки бинарных изображений, геометрических характеристик этих изображений, способов получения дополнительных параметров бинарных изображений. Изучение основных функций OpenCV, применяемых для цифровой обработки бинарных изображений.

## Выполнение работы:

Проработать примеры лабораторной работы в отдельном ноутбуке.

### Примеры


```
In [1]: import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline

def imshow(image, conversion=cv2.COLOR_BGR2RGB):
    image = cv2.cvtColor(image, conversion)
    plt.imshow(image)
    plt.xticks([])
    plt.yticks([])
    plt.axis("off")
    plt.show();
```

Вычисления характеристик:

```
In [2]: img = cv2.imread('pictures/bliss.jpg', 0)
imshow(img)

ret, thresh = cv2.threshold(img, 0, 255, 0)
# Все контуры изображения
contours, hierarchy = cv2.findContours(thresh, 5, 5)
# создание массива точек контура
cnt = contours[0]
```



3.1.1. Площадь, ограниченная контуром

```
In [3]: area = cv2.contourArea(cnt)
area
```

```
Out[3]: 2878681.0
```

3.1.2. Длина контурного периметра

```
In [4]: cv2.arcLength(cnt, 1)
```

```
Out[4]: 5996.0
```

Рисунок 1 – Пример 1

### 3.1.3. Моменты

```
In [5]: print(cv2.moments(cnt))

{'m00': 2870681.0, 'm10': 1986741659.5, 'm01': 1117089239.5, 'm20': 2541764829728.333, 'm11': 1871847125300.25, 'm02': 883559526280.3333, 'm30': 3658148676174987.5, 'm21': 1371249755634119.8, 'm12': 771815365465979.8, 'm03': 658288546642359.9, 'mu20': 635426287430.8833, 'mu11': 0.0, 'mu02': 20889881570.88325, 'mu30': -81.5, 'mu21': 0.1875, 'mu12': 0.03125, 'mu03': 0.25, 'nu20': 0.14828621748532587, 'nu11': 0.0, 'nu02': 0.046856081389612614, 'nu30': -1.3218418816782896e-14, 'nu21': 3.89286598934715e-17, 'nu12': 5.865344331557859e-18, 'nu03': 4.852275465246287e-17}
```

### 3.1.4. Отношение ширины к высоте ограничивающего прямоугольника

```
In [6]: x, y, w, h = cv2.boundingRect(cnt)
float(w)/h
```

```
Out[6]: 1.7777777777777777
```

### 3.1.5. Отношение площади контура к площади ограничивающего прямоугольника

```
In [7]: arr = w * h
ar = cv2.contourArea(cnt)
extent = float(arr) / ar
```

```
Out[7]: 1.0014483717529354
```

### 3.1.6. Эквивалентный диаметр

```
In [8]: ar = cv2.contourArea(cnt)
eqdiam = np.sqrt(4*ar / np.pi)
eqdiam
```

```
Out[8]: 1623.69857229762
```

### Задание 3.1.

Вычислить площадь  $s$ , периметр  $p$ , ширину  $w$ , высоту  $h$ , отношение ширины к высоте  $w/h$ , отношение площади изображения к площади описывающего прямоугольника  $s/(wh)$ , эквивалентный диаметр, центр масс, моменты бинарного изображения.

```
In [9]: img = cv2.imread('pictures/bliss.jpg',0)
imag = cv2.imread('pictures/bliss.jpg',0)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis("off")
```

```
Out[9]: (-0.5, 1919.5, 1879.5, -0.5)
```

## Рисунок 2 – Пример 2

### Задание 3.1.

Вычислить площадь  $s$ , периметр  $p$ , ширину  $w$ , высоту  $h$ , отношение ширины к высоте  $w/h$ , отношение площади изображения к площади описывающего прямоугольника  $s/(wh)$ , эквивалентный диаметр, центр масс, моменты бинарного изображения.

```
In [9]: img = cv2.imread('pictures/bliss.jpg',0)
imag = cv2.imread('pictures/bliss.jpg',0)
plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis("off")

Out[9]: (-0.5, 1919.5, 1879.5, -0.5)
```



```
In [10]: ret, thresh = cv2.threshold(img, 0, 255, 0)
contours, hierarchy = cv2.findContours(thresh, 5, 5)
# Создание контура
cnt = contours[0]
# Вычисление площади
s = cv2.contourArea(cnt)
# Вычисление периметра
p = cv2.arcLength(cnt, True)
# Вычисление координат
M = cv2.moments(cnt)
# Приблизительный прямоугольник вокруг бинарного изображения (рамка)
x, y, w, h = cv2.boundingRect(cnt)

imag = cv2.rectangle(imag, (x, y), (x+w, y+h), (0, 255, 0), 2)
imshow(imag)
```



## Рисунок 3 – Пример 3

```
In [11]: asprat_ratio = float(w) / h # соотношение сторон
         rectan = w * h
         s_ratio = float(s) / rectan
         eqdiam = np.sqrt(4*an / np.pi)

         print("Площадь s: ", s)
         print("Периметр p: ", p)
         print("Площадки H: ", H)
         print("x, y, w, h: ", x, y, w, h)
         print("Ширина w: (w), Высота h: (h)")
         print("Отношение ширины к высоте w/h: {asprat_ratio}")
         print("Отношение s/(wh): ", s_ratio)
         print("Эквивалентный диаметр: ", eqdiam)

Площадь s: 2070601.0
Периметр p: 5996.0
Площадки H: {'m0': 2070601.0, 'm0': 1986741659.5, 'm1': 1117089239.5, 'm2': 2541704829720.333, 'm11': 1071847125300.25, 'm2': 803559
526280.3333, 'm3': 3658148676174907.5, 'm1': 1371249755634119.0, 'm12': 771015365465979.0, 'm3': 658208546642359.0, 'mu0': 6354262074
30.0833, 'mu11': 0.0, 'mu2': 200889881570.08325, 'mu3': -81.5, 'mu21': 0.1875, 'mu12': 0.03125, 'mu3': 0.25, 'nu0': 0.148208217485325
87, 'nu11': 0.0, 'nu2': 0.046856081389612614, 'nu3': -1.3218418016702896e-14, 'nu21': 3.039206598934715e-17, 'nu12': 5.065344331557859e
-18, 'nu3': 4.052275465246287e-17}
x, y, w, h: 0 0 1920 1080
Ширина w: 1920, Высота h: 1080
Отношение ширины к высоте w/h: 1.7777777777777777
Отношение s/(wh): 0.9985537229938272
Эквивалентный диаметр: 1623.69057229762
```

**3.2.1. Маска и пиксельные точки**

```
In [12]: img = cv2.imread("pictures/bliss.jpg", 0)
         mask = np.zeros(img.shape, np.uint8)

         cv2.drawContours(mask, [cnt], 0, 255, -1)
         pixpoints = np.transpose(np.nonzero(mask))
         pixpoints = cv2.findNonZero(mask)
```

**3.2.2. Максимальное и минимальное значения и их координаты**

```
In [13]: minval, maxval, minloc, maxloc = cv2.minMaxLoc(img, mask=mask)
         print(minval, maxval, minloc, maxloc)

2.0 252.0 (1597, 860) (1783, 582)


3.2.3. Крайние точки



```
In [14]: leftmost = tuple(cnt[cnt[:, :, 0].argmin()][0])
         rightmost = tuple(cnt[cnt[:, :, 0].argmax()][0])
         topmost = tuple(cnt[cnt[:, :, 1].argmin()][0])
         bottommost = tuple(cnt[cnt[:, :, 1].argmax()][0])
         print(leftmost, rightmost, topmost, bottommost)

(0, 0) (1919, 0) (0, 0) (1919, 1079)


3.2.4. Средняя интенсивность



```
In [15]: mean_val = cv2.mean(img, mask=mask)
         mean_val

Out[15]: (123.72932773919753, 0.0, 0.0, 0.0)
```


```


```

Рисунок 4 – Пример 4

```
In [11]: asprat_ratio = float(w) / h # соотношение сторон
         rectan = w * h
         s_ratio = float(s) / rectan
         eqdiam = np.sqrt(4*an / np.pi)

         print("Площадь s: ", s)
         print("Периметр p: ", p)
         print("Площадки H: ", H)
         print("x, y, w, h: ", x, y, w, h)
         print("Ширина w: (w), Высота h: (h)")
         print("Отношение ширины к высоте w/h: {asprat_ratio}")
         print("Отношение s/(wh): ", s_ratio)
         print("Эквивалентный диаметр: ", eqdiam)

Площадь s: 2070601.0
Периметр p: 5996.0
Площадки H: {'m0': 2070601.0, 'm0': 1986741659.5, 'm1': 1117089239.5, 'm2': 2541704829720.333, 'm11': 1071847125300.25, 'm2': 803559
526280.3333, 'm3': 3658148676174907.5, 'm1': 1371249755634119.0, 'm12': 771015365465979.0, 'm3': 658208546642359.0, 'mu0': 6354262074
30.0833, 'mu11': 0.0, 'mu2': 200889881570.08325, 'mu3': -81.5, 'mu21': 0.1875, 'mu12': 0.03125, 'mu3': 0.25, 'nu0': 0.148208217485325
87, 'nu11': 0.0, 'nu2': 0.046856081389612614, 'nu3': -1.3218418016702896e-14, 'nu21': 3.039206598934715e-17, 'nu12': 5.065344331557859e
-18, 'nu3': 4.052275465246287e-17}
x, y, w, h: 0 0 1920 1080
Ширина w: 1920, Высота h: 1080
Отношение ширины к высоте w/h: 1.7777777777777777
Отношение s/(wh): 0.9985537229938272
Эквивалентный диаметр: 1623.69057229762
```

**3.2.1. Маска и пиксельные точки**

```
In [12]: img = cv2.imread("pictures/bliss.jpg", 0)
         mask = np.zeros(img.shape, np.uint8)

         cv2.drawContours(mask, [cnt], 0, 255, -1)
         pixpoints = np.transpose(np.nonzero(mask))
         pixpoints = cv2.findNonZero(mask)
```

**3.2.2. Максимальное и минимальное значения и их координаты**

```
In [13]: minval, maxval, minloc, maxloc = cv2.minMaxLoc(img, mask=mask)
         print(minval, maxval, minloc, maxloc)

2.0 252.0 (1597, 860) (1783, 582)


3.2.3. Крайние точки



```
In [14]: leftmost = tuple(cnt[cnt[:, :, 0].argmin()][0])
         rightmost = tuple(cnt[cnt[:, :, 0].argmax()][0])
         topmost = tuple(cnt[cnt[:, :, 1].argmin()][0])
         bottommost = tuple(cnt[cnt[:, :, 1].argmax()][0])
         print(leftmost, rightmost, topmost, bottommost)

(0, 0) (1919, 0) (0, 0) (1919, 1079)


3.2.4. Средняя интенсивность



```
In [15]: mean_val = cv2.mean(img, mask=mask)
         mean_val

Out[15]: (123.72932773919753, 0.0, 0.0, 0.0)
```


```


```

Рисунок 5 – Пример 5

### 3.2.5. Ориентация

```
In [16]: img = cv2.imread('pictures/bliss.jpg', 0)
ret, thresh = cv2.threshold(img, 0, 255, 0)
contours, hierarchy = cv2.findContours(thresh, 5, 5)
cnt = contours[0]
(x, y), (MA, ma), ang = cv2.fitEllipse(cnt)
print(ang)
```

180.0

### Задание 3.2.

Используя изображение маски определить крайние точки, минимальное и максимальное значения и их координаты для бинарного изображения. Найти среднюю интенсивность изображения в градациях серого, ориентацию бинарного изображения с выделенной осью.

```
In [17]: img = cv2.imread('pictures/bliss.jpg', 0)
ret, thresh = cv2.threshold(img, 0, 255, 0)
contours, hierarchy = cv2.findContours(thresh, 5, 5)
cnt = contours[0]

plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.axis("off")
```

Out[17]: (-0.5, 1919.5, 1079.5, -0.5)



Рисунок 6 – Пример 6

```
In [18]: mask = np.zeros(img.shape, np.uint8)

cv2.drawContours(mask, [cnt], 0, 255, -1)
pixpoin = np.transpose(np.nonzero(mask))
minv, maxv, minl, maxl = cv2.minMaxLoc(img, mask=mask)

# Крайние точки
leftmost = tuple(cnt[cnt[:, :, 0].argmin()][0])
rightmost = tuple(cnt[cnt[:, :, 0].argmax()][0])
topmost = tuple(cnt[cnt[:, :, 1].argmin()][0])
bottommost = tuple(cnt[cnt[:, :, 1].argmax()][0])

(x,y),(MA,ma),ang=cv2.fitEllipse(cnt)
meanv = cv2.mean(img,mask = mask)

print(f"Пиксельные точки:\n {pixpoin}")
print(f"Максимальное и минимальное значения и их координаты: {minv}, {maxv}, {minl}, {maxl}")
print(f"Крайние точки: {leftmost}, {rightmost}, {topmost}, {bottommost}")
print(f"Средняя интенсивность: {meanv}")
print(f"Ориентация: {ang}")
```

```
Пиксельные точки:
[[ 0  0]
 [ 0  1]
 [ 0  2]
 ...
 [1079 1917]
 [1079 1918]
 [1079 1919]]
Максимальное и минимальное значения и их координаты: 2.0, 252.0, (1597, 860), (1783, 582)
Крайние точки: (0, 0), (1919, 0), (0, 0), (1919, 1079)
Средняя интенсивность: (123.72932773919753, 0.0, 0.0, 0.0)
Ориентация: 180.0
```

Рисунок 7 – Пример 7

## Индивидуальное задание

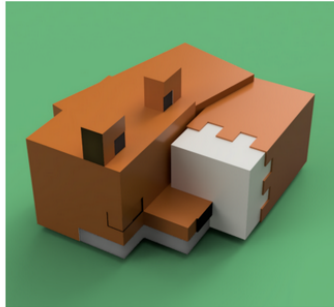
### Задание

1. Вырезать объект с неоднородного фона с использованием маски.
2. Сохранить полученную картинку в формате PNG с альфа-каналом (прозрачностью).

```
In [2]: import cv2
import numpy as np
from matplotlib import pyplot as plt
%matplotlib inline

def imshow(image, conversion=cv2.COLOR_BGR2RGB):
    image = cv2.cvtColor(image, conversion)
    plt.imshow(image)
    plt.xticks([])
    plt.yticks([])
    plt.axis('off')
    plt.show();
```

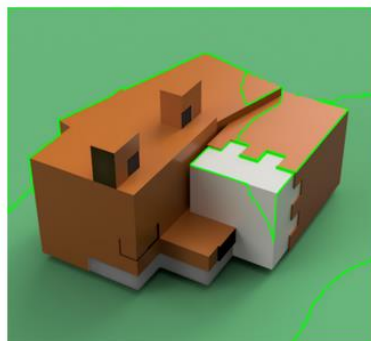
```
In [3]: # Импортируем исходную картинку
img = cv2.imread('pictures/fox.png')
imshow(img)
```



Попробуем найти контуры изображения и отрисовать их при помощи функции `cv2.findContours()`.  
Здесь: `RETR_TREE` означает, что алгоритм извлечет все возможные контуры из двоичного изображения, а `CHAIN_APPROX_SIMPLE` - метод/алгоритм аппроксимации контура (в данном случае `CHAIN_APPROX_SIMPLE`).

```
In [4]: # Переведем изображение в серый
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
# Применяем преобразование
ret, thresh = cv2.threshold(img_gray, 150, 255, cv2.THRESH_BINARY)
imshow(thresh)
# Найдём контуры
contours1, hierarchy1 = cv2.findContours(thresh, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
# Отрисует контуры
img_copy = img.copy()
img_contours = cv2.drawContours(img_copy, contours1, -1, (0, 255, 0), 2)
imshow(img_contours)
```

Рисунок 8 – Индивидуальное задание (1)



Как видим, контуры получились не совсем подходящие для создания маски объекта, поэтому применим другой метод.

Рисунок 9 – Индивидуальное задание (2)

Для более удобной работы с отделением фона преобразуем цветовое пространство BGR в HSV при помощи функции `cv2.cvtColor()`. Цветовое пространство HSV (англ. Hue, Saturation, Value — цветовой тон, насыщенность, значение/яркость).

```
In [5]: hsv = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
```

Для отделения яркого объекта нам нужно взять нижний и верхний пороги границы цвета. В нашем случае это оранжевый цвет, запишем эти значения в массив.

```
In [6]: lower_bound = np.array([0, 0, 0])
upper_bound = np.array([64, 255, 255])
```

Создадим маску, в которой отделим объект на основе границ цвета.

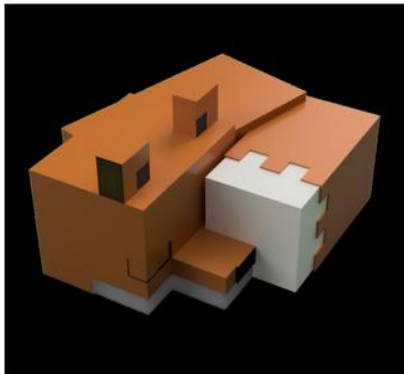
```
In [7]: mask = cv2.inRange(hsv, lower_bound, upper_bound)
imgshow(mask)
```



Рисунок 10 – Индивидуальное задание (3)

Теперь вычтем эту маску из изображения, для этого используем функцию `cv2.bitwise_and()` и получим объект без фона.

```
In [8]: segmented_img = cv2.bitwise_and(img, img, mask=mask)
imgshow(segmented_img)
```



Сохраним полученное изображение в формате PNG с прозрачностью.

Сделать это можно при помощи конвертации изображения в серый, а затем функцией `cv2.threshold()` с последующим спlicing каналов изображения с альфа-каналом.

```
In [9]: gray_img = cv2.cvtColor(segmented_img, cv2.COLOR_BGR2GRAY)
_, alpha = cv2.threshold(gray_img, 0, 255, cv2.THRESH_BINARY)

# Разделяем цветовые каналы
b, g, r = cv2.split(segmented_img)

# Список из каналов, включая альфа-канал (прозрачность)
rgba = [b, g, r, alpha]

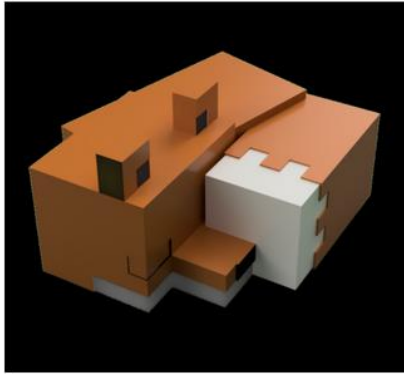
# Соединяем каналы
result_thres = cv2.merge(rgba, 4)

# Записываем изображение
cv2.imwrite("pictures/fox_threshold.png", result_thres)
imgshow(result_thres)
```

Рисунок 11 – Индивидуальное задание (4)

Теперь вычтем эту маску из изображения, для этого используем функцию `cv2.bitwise_and()` и получим объект без фона.

```
In [8]: segmented_img = cv2.bitwise_and(img, img, mask=mask)
imgshow(segmented_img)
```



Сохраним полученное изображение в формате PNG с прозрачностью.

Сделать это можно при помощи конвертации изображения в серый, а затем функцией `cv2.threshold()` с последующим слиянием каналов изображения с альфа-каналом.

```
In [9]: gray_img = cv2.cvtColor(segmented_img, cv2.COLOR_BGR2GRAY)
_, alpha = cv2.threshold(gray_img, 0, 255, cv2.THRESH_BINARY)

# Разделяем цветовые каналы
b, g, r = cv2.split(segmented_img)

# Список из каналов, включая альфа-канал (прозрачность)
rgba = [b, g, r, alpha]

# Собираем каналы
result_thres = cv2.merge(rgba, 4)

# Записываем изображение
cv2.imwrite("pictures/fox_threshold.png", result_thres)
imgshow(result_thres)
```

Рисунок 12 – Индивидуальное задание (5)

Для более лучшего результата, вместо использования функции `threshold` можно добавить маске альфа-канал.

```
In [10]: result_add = segmented_img.copy()
result_add = cv2.cvtColor(result_add, cv2.COLOR_BGR2BGRA)
result_add[:, :, 3] = mask

cv2.imwrite('pictures/fox_add.png', result_add)
imgshow(result_add)
```

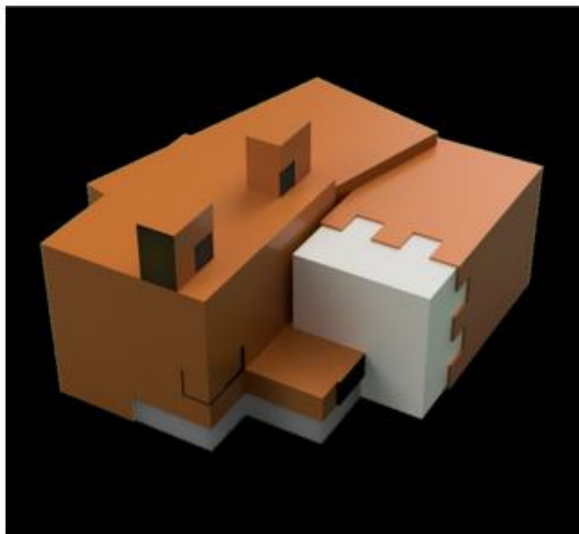


Рисунок 13 – Индивидуальное задание (6)



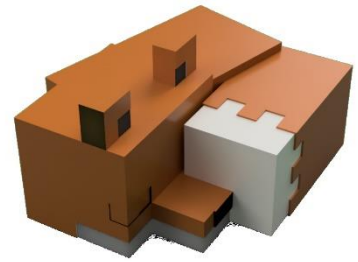
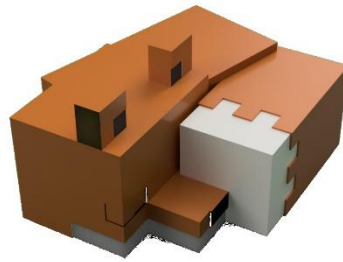
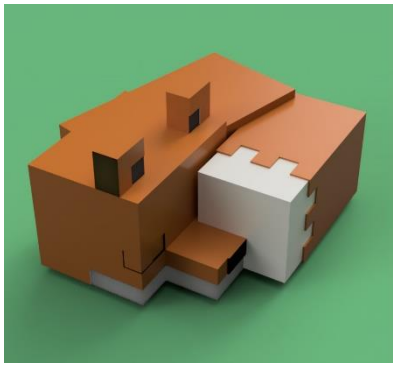


Рисунок 14 – Индивидуальное задание, результат

**Вывод:** В результате выполнения работы были изучены методы цифровой обработки бинарных изображений, их геометрические характеристики и основные функции для работы с ними.

### **1. Что такое бинарное изображение?**

Бинарные изображения – изображения, пиксели которого принимают только два значения: 0 и 1, что соответствует черному или белому цвету.

### **2. Какие существуют характеристики бинарного изображения?**

Площадь, периметр, объем, ширина, высота, отношение ширины к высоте, отношение площади изображения к площади описывающего прямоугольника, эквивалентный диаметр, моменты, определяющие площадь, центр масс объекта, и другие моменты более высокого порядка, положение в пространстве и ориентация.

### **3. Что используют для описания бинарного изображения?**

Характеристическая функция  $b(x, y)$ .

### **4. Как можно получить бинарное изображение?**

Бинарное изображение можно получить после пороговой обработки полутонового изображения.

$$b(x, y) = \begin{cases} 1, & \text{если } f(x, y) \geq a, \\ 0, & \text{если } f(x, y) < a \end{cases}$$

### **5. Чему равна площадь бинарного изображения?**

$$S = \sum_{x,y} b(x, y)$$

### **6. Что такое маска изображения?**

Маска изображения – массив всех точек изображения.

### **7. Как вычислить все моменты?**

Вычислив моменты объектов, можно использовать их в качестве характерных признаков для классификации этих объектов. Функция `cv2.moments ()` дает список всех вычисленных значений моментов.

### **8. Что такое эквивалентный диаметр?**

Эквивалентный диаметр – это диаметр круга, площадь которого совпадает с площадью контура.

### **9. Каковы характерные параметры бинарных изображений?**

Максимальное и минимальное значения и их координаты, крайние точки, средняя интенсивность, ориентация.