МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Федеральное государственное автономное

образовательное учреждение высшего образования

«СЕВЕРО-КАВКАЗСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Кафедра инфокоммуникаций

«Процессы дискретизации и квантования изображения»

Отчет по лабораторной работе № 9 (3)

по дисциплине «Технологии распознавания образов»

Выполнил студент группы ПИЖ-б-о-21-1

Кучеренко С. Ю. « » 2023г.

Подпись студента_____

Работа защищена « »_____2023г.

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:

Примеры лабораторной работы

In [1]:
```
import cv2
import numpy as np
```

In [6]:
```
img = cv2.imread('cat.jpeg', 0)
```

In [7]:
```
ret, thresh = cv2.threshold(img, 0, 255, 0)
contours, hierarchy = cv2.findContours(thresh, 5, 5)
cnt = contours[0]
```

3.1.1. Площадь, ограниченная контуром

In [8]:
```
area = cv2.contourArea(cnt)
area
```

Out[8]:
```
3682561.0
```

3.1.2. Длина контурного периметра

In [9]:
```
cv2.arcLength(cnt, 1)
```

Out[9]:
```
7676.0
```

3.1.3. Моменты

In [29]:
```
cv2.moments(cnt)
```

Out[29]:
```
{'m00': 3682561.0,
 'm10': 3533417279.5,
 'm01': 3533417279.5,
 'm20': 4520418506240.333,
 'm11': 3390313879680.25,
 'm02': 4520418506240.333,
 'm30': 6506012335106149.0,
 'm21': 4337341556737600.0,
 'm12': 4337341556737600.0,
 'm03': 6506012335106400.0,
 'mu20': 1130104626560.0825,
 'mu11': -0.00048828125,
 'mu02': 1130104626560.0825,
 'mu30': -250.0,
 'mu21': 0.75,
 'mu12': 0.75,
 'mu03': 1.0,
 'nu20': 0.08333333333333329,
 'nu11': -3.6005607985627815e-17,
 'nu02': 0.08333333333333329,
 'nu30': -9.60649884765057e-15,
 'nu21': 2.881949654295171e-17,
 'nu12': 2.881949654295171e-17,
 'nu03': 3.842599539060228e-17}
```

3.1.4. Отношение ширины к высоте ограничивающего прямоугольника

In [30]:
```
x, y, w, h = cv2.boundingRect(cnt)
```

In [31]:
```
float(w)/h
```

Out[31]:
```
1.0
```

3.1.5. Отношение площади контура к площади ограничивающего прямоугольника

In [35]:
```
arr = w * h
ar = cv2.contourArea(cnt)
```

In [37]:
```
extent = float(arr) / ar
extent
```

Out[37]:
```
1.0010424810342584
```

3.1.6. Эквивалентный диаметр

In [39]:
```
ar = cv2.contourArea(cnt)
```

In [40]:
```
eqdiam = np.sqrt(4*ar / np.pi)
eqdiam
```

Out[40]:
```
2165.3596216562887
```

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 [3]:
```
import cv2
import numpy as np
```

In [4]:
```
img = cv2.imread('cat.jpeg', 0)
imag = cv2.imread('cat.jpeg', 0)
```

In [5]:
```
ret, thresh = cv2.threshold(img, 0, 255, 0)
contours, hierarchy = cv2.findContours(thresh, 5, 5)
cnt = contours[0]
```

In [6]:
```
ar = cv2.contourArea(cnt)
print(ar)
```

```
3682561.0
```

In [7]:
```
prm = cv2.arcLength(cnt, 1)
print(prm)
```

```
7676.0
```

In [8]:
```
M = cv2.moments(cnt)
print(M)
```

```
{'m00': 3682561.0, 'm10': 3533417279.5, 'm01': 3533417279.5, 'm20': 4520418506240.333,
'm11': 3390313879680.25, 'm02': 4520418506240.333, 'm30': 6506012335106149.0, 'm21': 4
337341556737600.0, 'm12': 4337341556737600.0, 'm03': 6506012335106400.0, 'mu20': 11301
04626560.0825, 'mu11': -0.00048828125, 'mu02': 1130104626560.0825, 'mu30': -250.0, 'mu
21': 0.75, 'mu12': 0.75, 'mu03': 1.0, 'nu20': 0.08333333333333329, 'nu11': -3.60056079
85627815e-17, 'nu02': 0.08333333333333329, 'nu30': -9.60649884765057e-15, 'nu21': 2.88
1949654295171e-17, 'nu12': 2.881949654295171e-17, 'nu03': 3.842599539060228e-17}
```

In [9]:
```
x, y, w, h = cv2.boundingRect(cnt)
print(x, y, w, h)
```

```
0 0 1920 1920
```

In [59]:
```
imag = cv2.rectangle(imag, (x, y), (x+w, y+h), (0, 255, 0), 2)
cv2.imshow('Rectan', imag)
```

In [ ]:
```
asprat = float(w) / h
rectar = w * h
extent = float(ar) / rectar
eqdiam = np.sqrt(4*ar / np.pi)

print(asprat, extent)
print(eqdiam)
cv2.waitKey(0)
```

```
1.0 0.9989586046006944
2165.3596216562887
```

3.2.1. Маска и пиксельные точки

In [29]:
```
import cv2
import numpy as np
```

In [30]:
```
img = cv2.imread("cat.jpeg", 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 [38]:
```
minval, maxval, minloc, maxloc = cv2.minMaxLoc(img, mask=mask)
print(minval, maxval, minloc, maxloc)
```

```
0.0 255.0 (877, 0) (695, 402)
```

3.2.3. Крайние точки

In [37]:
```
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, 1919)
```

3.2.4. Средняя интенсивность

In [40]:
```
mean_val = cv2.mean(img, mask=mask)
mean_val
```

Out[40]:
```
(76.24194200303819, 0.0, 0.0, 0.0)
```

3.2.5. Ориентация

In [41]:
```
import cv2
import numpy as np
```

In [42]:
```
img = cv2.imread('cat.jpeg', 0)
```

In [44]:
```
ret,thresh = cv2.threshold(img,0,255,0)
contours, hierarchy = cv2.findContours(thresh, 5, 5)
cnt = contours[0]
```

In [45]:
```
(x, y), (MA, ma), ang = cv2.fitEllipse(cnt)
print(ang)
```

```
179.98919677734375
```

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 [2]:
```
import cv2
import numpy as np
```

In [3]:
```
img = cv2.imread('cat.jpeg', 0)
```

In [4]:
```
ret, thresh = cv2.threshold(img, 0, 255, 0)
contours, hierarchy = cv2.findContours(thresh, 5, 5)
cnt = contours[0]
```

In [5]:
```
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)
```

In [6]:
```
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])
```

In [7]:
```
(x,y),(MA,ma),ang=cv2.fitEllipse(cnt)
```

In [8]:
```
meanv = cv2.mean(img,mask = mask)
```

In [ ]:
```
print(pixpoin)
print(minv, maxv, minl, maxl)
print(leftmost, rightmost, topmost, bottommost)
print(meanv)
print(ang)
cv2.waitKey(0)
```

```
[[   0    0]
 [   0    1]
 [   0    2]
 ...
 [1919 1917]
 [1919 1918]
 [1919 1919]]
0.0 255.0 (877, 0) (695, 402)
(0, 0) (1919, 0) (0, 0) (1919, 1919)
(76.24194200303819, 0.0, 0.0, 0.0)
179.98919677734375
```

Самостоятельное задание

Загружаем изображение с белым фоном и черными линиями на нем, создаем бинарное изображение и выводим на экран количество линий на изображении. Затем находим центральную точку каждой линии и выводим координаты этих точек на экран.

In [60]:
```
import cv2
```

Загружаем изображение и преобразуем в градации серого.

In [70]:
```
img = cv2.imread('bin.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

Находим контуры на бинарном изображении.

In [67]:
```
ret, thresh = cv2.threshold(gray, 200, 255, cv2.THRESH_BINARY_INV)
contours, hierarchy = cv2.findContours(thresh, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMP
```

In [68]:
```
print('Number of lines:', len(contours))
```

```
Number of lines: 11
```

Находим центральные точки каждой линии.

Формула для нахождения координат центра масс контура в двумерном пространстве:

$$cX = \frac{M_{10}}{M_{00}}$$

$$cY = \frac{M_{01}}{M_{00}}$$

где M_{00} M_{10} и M_{01} - это нормализованные моменты контура первого и нулевого порядка.

In [69]:
```
for i, contour in enumerate(contours):
    moments = cv2.moments(contour)
    cX = int(moments["m10"] / moments["m00"])
    cY = int(moments["m01"] / moments["m00"])
    print(f'Line {i+1}: ({cX}, {cY})')
```

```
Line 1: (299, 81)
Line 2: (271, 81)
Line 3: (242, 81)
Line 4: (213, 81)
Line 5: (183, 81)
Line 6: (154, 81)
Line 7: (124, 81)
Line 8: (95, 81)
Line 9: (66, 81)
Line 10: (36, 81)
Line 11: (7, 81)
```