**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра САУ**

**отчет**

**по лабораторной работе № 5**

**по дисциплине «Техническое зрение»**

**Тема: Границы и контуры**

|  |  |  |
| --- | --- | --- |
| Студентки гр. 6491 |  | Ларионов Е.А.  Зверев Д.Ю. |
| Преподаватель |  | Моклева К.А. |

Санкт-Петербург

2020

***Цель работы*:** изучить способы выделения границ на изображении, поиск контуров на границах и получения информации об объектах на основе контуров.

**Задание 1.**

Исследуйте все известные вам способы поиска границ на изображении. Для этого выберите несколько изображений, содержащих как четко отделимые от фона границы, так и нечеткие границы, почти сливающиеся с фоном. При применении методов аргументируйте выбор значений, передаваемых в качестве параметров методов.

**Задание 2.1.**

Исследуйте работу функции findContours() на двух типах бинарных изображений:

1) бинарные изображения, полученные с помощью функции threshold()

2) бинарные изображения границ, полученные детектором границ Кенни

Как отличается количество контуров? Почему?

**Задание 2.2.**

Возьмите изображение окружности с толщиной линии в несколько пикселей (вы можете самостоятельно нарисовать его, например, в paint). Вычислите контуры на этом изображении. Найдите один контур, который описывает окружность с внешней стороны линии, и один контур, который описывает окружность с внутренней стороны линии. Для них вычислите длину, площадь. Почему значения отличаются таким образом? Для каждого контура вычислите ограничивающий прямоугольник и ограничивающую окружность. Сравните значения площадей ограничивающих фигур с площадями контуров. Прокомментируйте результат.

**Дополнительное задание 1.**

Для выполнения этого задания вам понадобится файл 5\_1.png. Для этого

изображения все контуры на нечетном уровне вложенности нарисуйте

красным цветом, все контуры на четном - синим цветом.

**Задание №1:**

**Код программы:**

import cv2

img1 =cv2.imread('C:/Users/greka/greka/technical vision/test2.jpg', cv2.IMREAD\_REDUCED\_COLOR\_2)

#применение собеля, CV\_8U 8 unsigent numpy array(выбирается по изображению, подбирал), следующие за ним цифры это порядок производных по х и у соответсветвенно,

#5 размер ядра собеля

x = cv2.Sobel(img1, cv2.CV\_8U, 1, 0, 5)

#применение метода Щарра, различается видом ядра

x1 = cv2.Scharr(img1, cv2.CV\_8U, 1, 0, 5)

y = cv2.Sobel(img1, cv2.CV\_8U, 0, 1)

xy = cv2.Sobel(img1, cv2.CV\_8U, 1, 1)

# лапласиан хорошо показал себя на четких границах cv\_64f больше глубина выходного изображения (выбирается по изображению, подбирал), потом идут параметры размер ядра, маштаб коэф, смещение

#это все по дефолту установлено вместе с типом границ

laplacian = cv2.Laplacian(img1, cv2.CV\_64F, cv2.BORDER\_DEFAULT)

# размываем изображение для фильтра кенни с помощью гауссовского размытия, в скобка это размер ядра размытия, остальные по дефолту, сглаживаем для уменьшения шума и вычисления градиентов

blur = cv2.GaussianBlur(img1,(1,1), cv2.BORDER\_DEFAULT)

# нижний порог, верхний порог, размер ядра, и выбор точности, тру-точно считает градиент, по формуле корня из суммы квадратов, фалс не точно, просто по суммы модулей

canny = cv2.Canny(blur, threshold1 = 150, threshold2 = 200, apertureSize = 5,

L2gradient = True)

cv2.imshow('sobelx', x)

cv2.imshow('scharr', x1)

cv2.imshow('sobely', y)

cv2.imshow('sobelxy', xy)

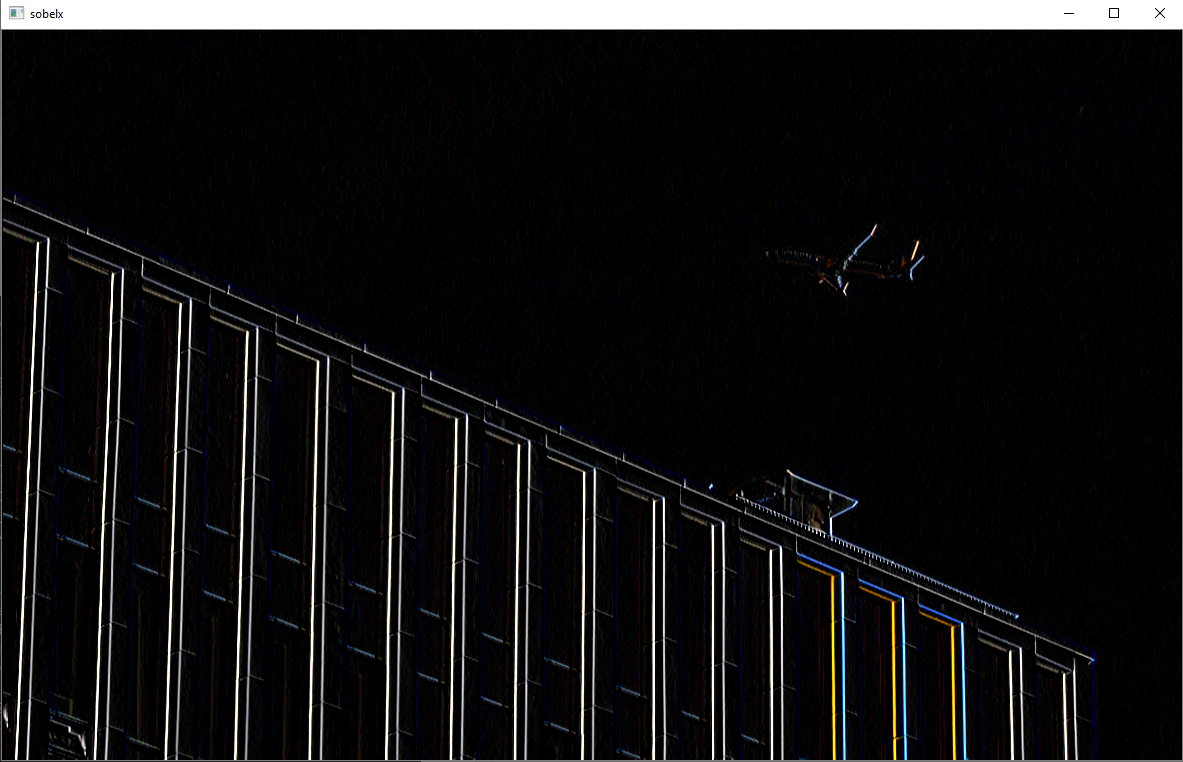
cv2.imshow('laplas', laplacian)

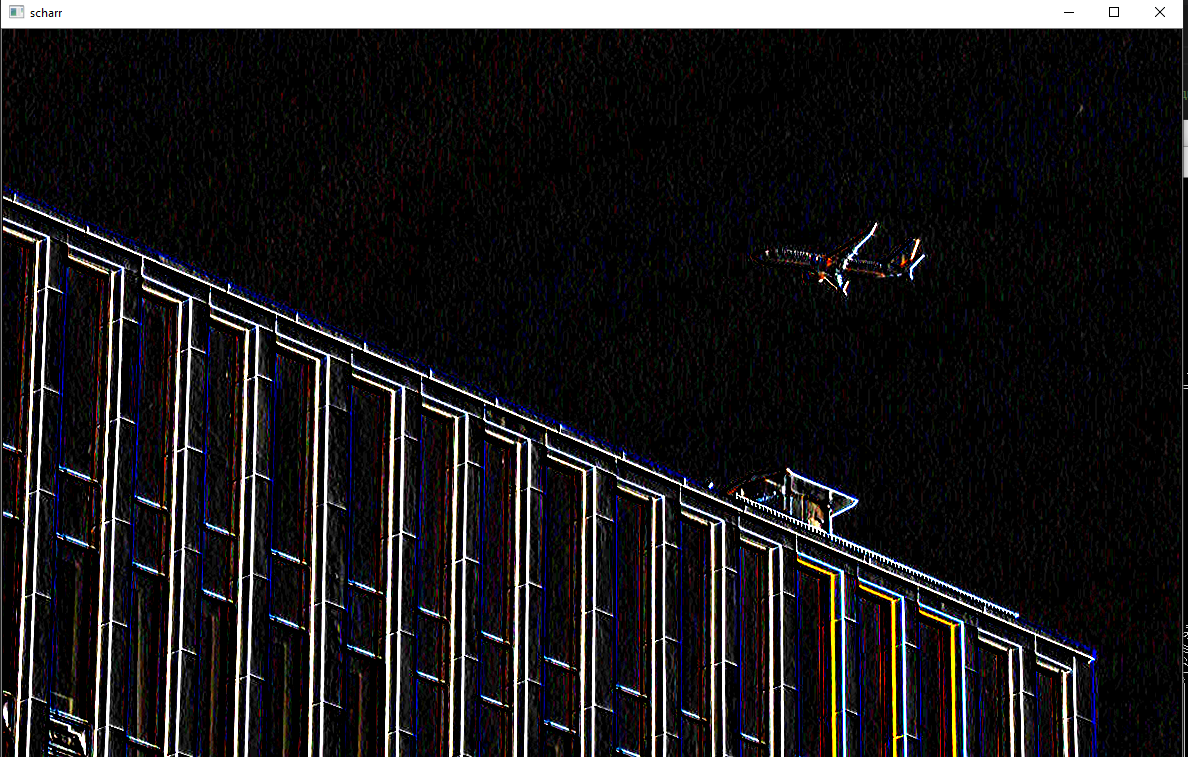
cv2.imshow('canny', canny)

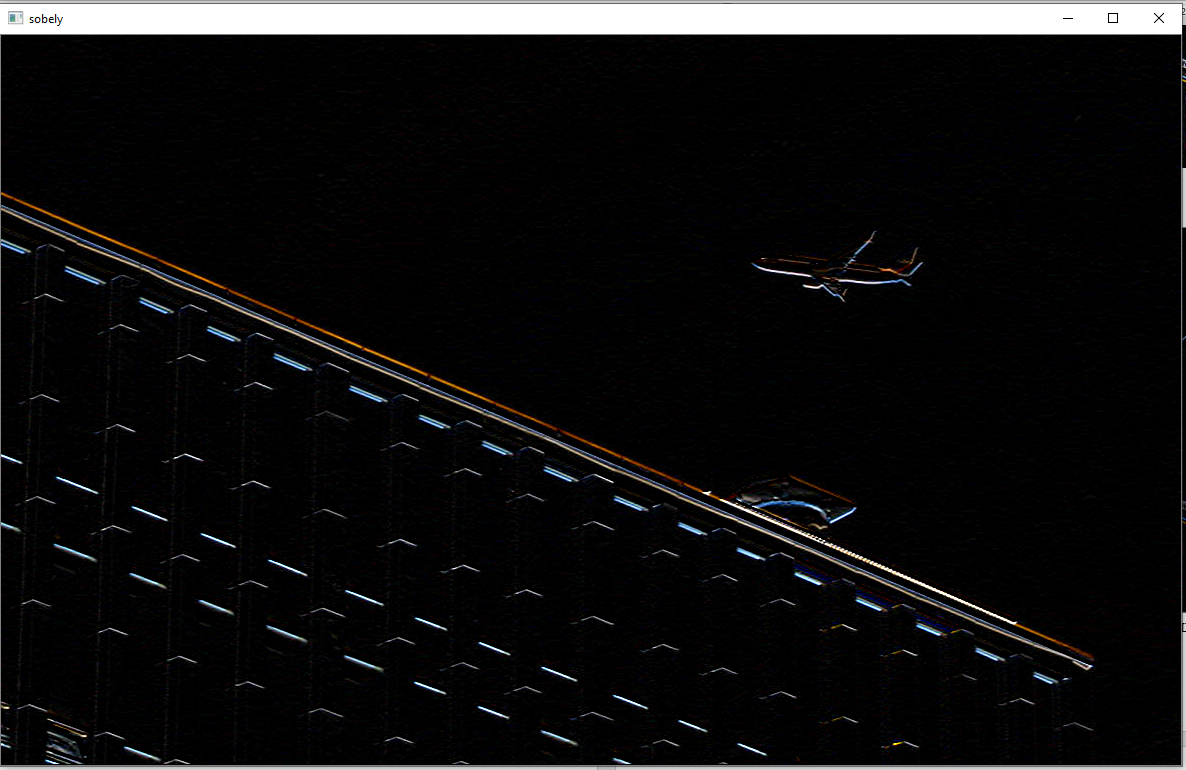
cv2.waitKey()

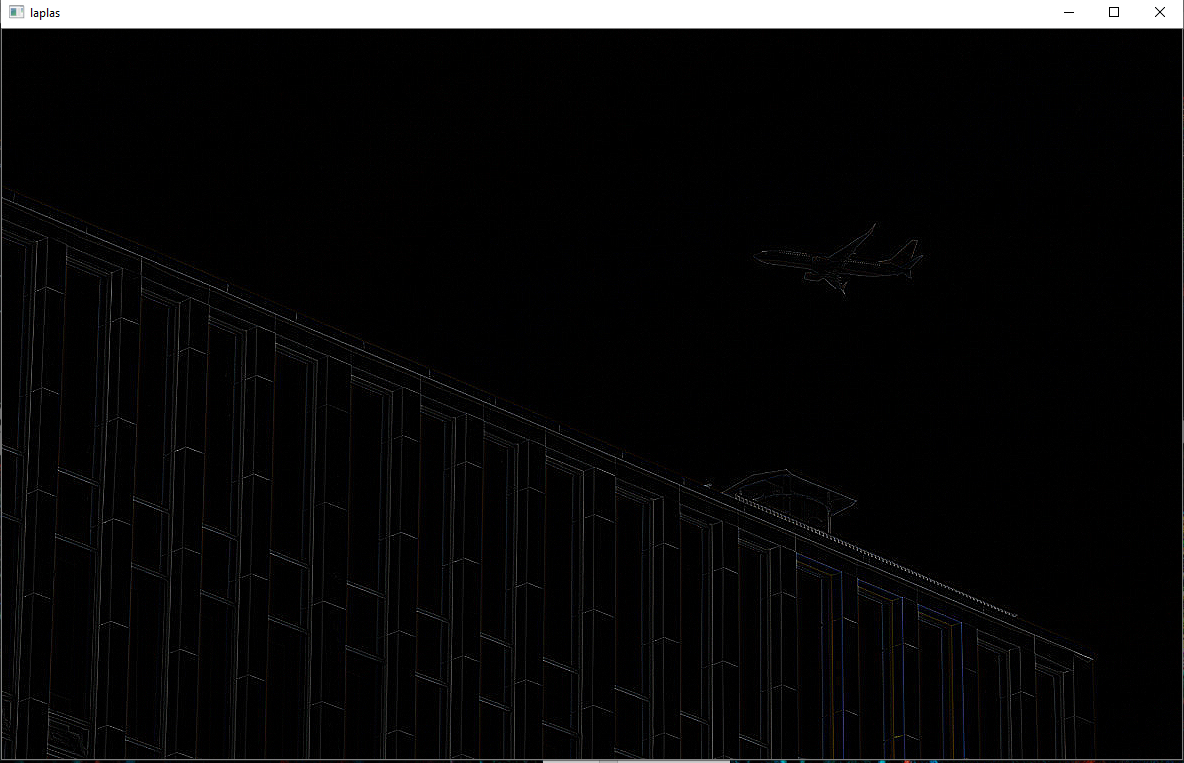
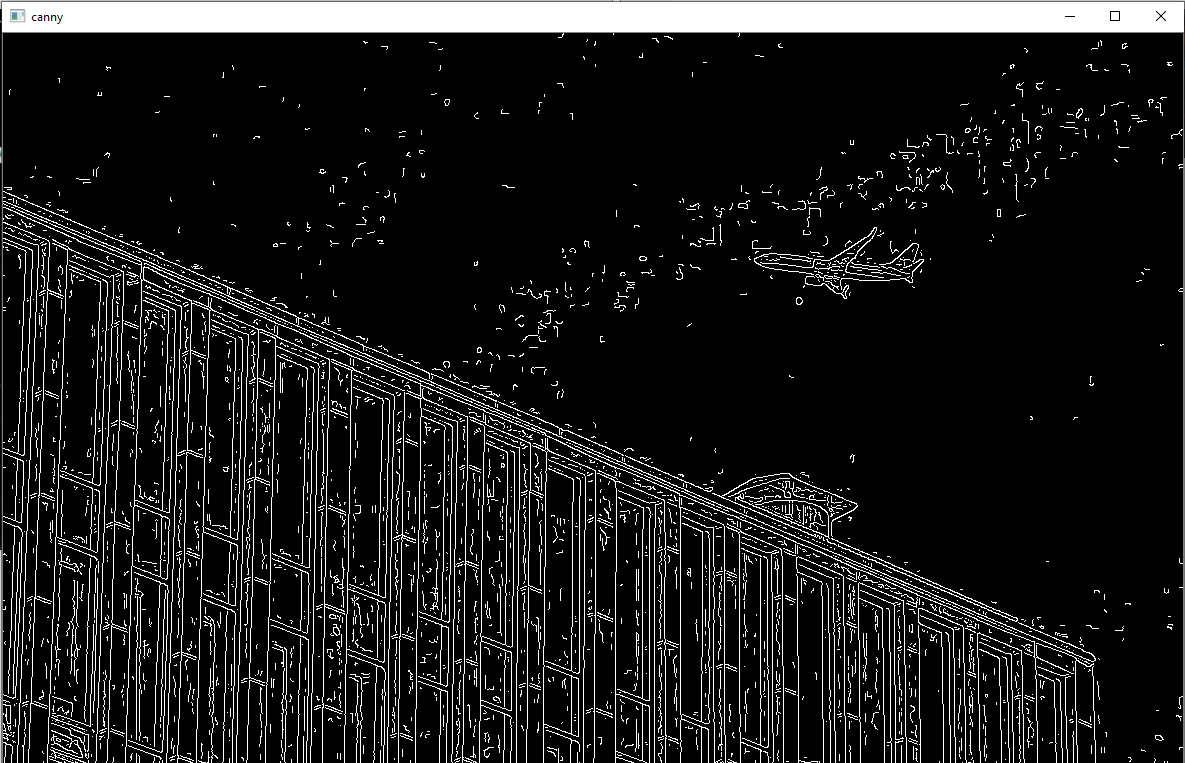
cv2.destroyAllWindows()

**Результат:**

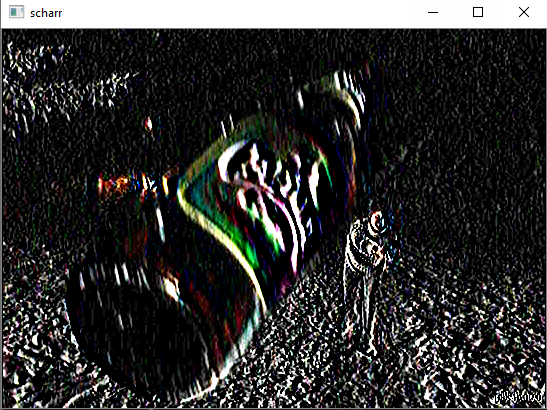
Изображения с четко отделимыми от фона границами:

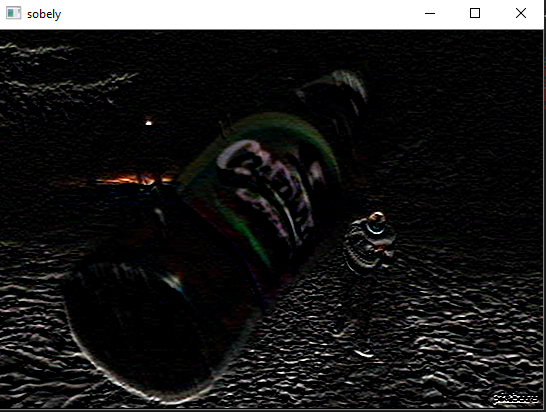
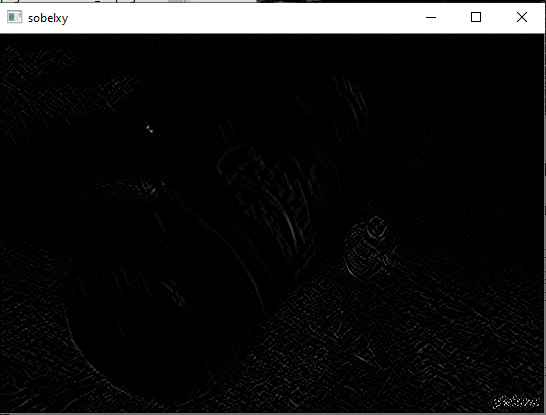




Изображения с нечеткими границами, почти сливающимися с фоном:



Чем порядок производной выше, тем границы на изображении более  размыты;

Порядок производной должен быть меньше размера ядра Собеля(ksize). Ksize установлен 5

Оператор Собеля 'ksize' вычисляет градиент яркости изображения в каждой  точке, участки с большой величиной градиента видны как белые линии;

Для изображений с четкими границами больше всего подходит оператор Лапласа.

**Задание №2.1**

import cv2

img = cv2.imread('C:/Users/greka/greka/technical vision/test2.jpg',

cv2.IMREAD\_REDUCED\_COLOR\_2)

img1 = cv2.imread('C:/Users/greka/greka/technical vision/test2.jpg',

cv2.IMREAD\_REDUCED\_GRAYSCALE\_2)

#применяем пороговый фильтр

ret, thresh = cv2.threshold(img1, 150, 255, cv2.THRESH\_BINARY)

#находим контура, retr\_tree, полная иерархия вложенности контуров, chain\_approx\_none сохраняет все точки контура

contours1, hierarchy = cv2.findContours(thresh,cv2.RETR\_TREE,

cv2.CHAIN\_APPROX\_NONE)

print('Количество контуров функции threshold:'+ str(len(contours1)))

blur = cv2.GaussianBlur(img1, (1,1), cv2.BORDER\_DEFAULT)

canny = cv2.Canny(blur, threshold1 = 200, threshold2 = 225, apertureSize = 3,

L2gradient = True)

contours2, hierarchy = cv2.findContours(canny, cv2.RETR\_TREE,

cv2.CHAIN\_APPROX\_NONE)

print('Количество контуров функции Canny():' + str(len(contours2)))

#разница в количестве строк из-за того что кенни определяет еще края как белые линии

cv2.drawContours(img, contours2, -1, (25, 150, 230), 2)

#а это подрисовал контур на основном изображении

cv2.imshow('Original', img)

cv2.imshow('Gray', img1)

cv2.imshow('Threshold', thresh)

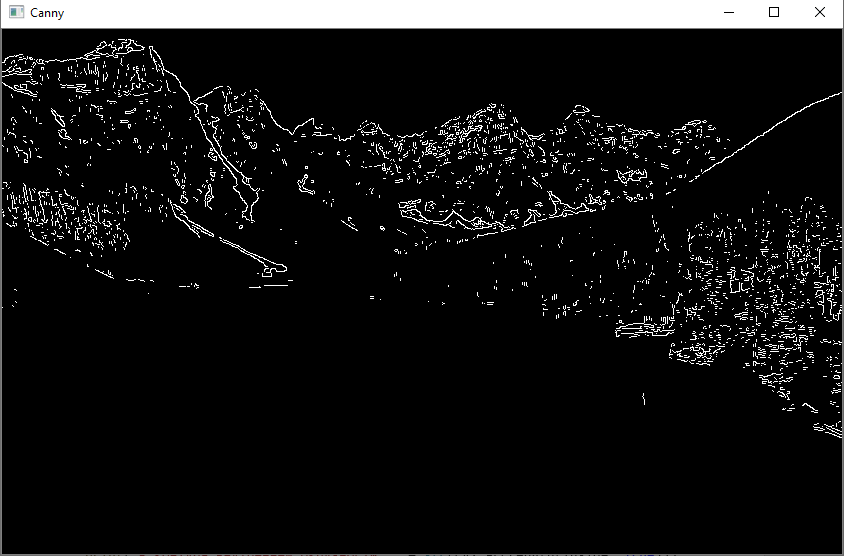
cv2.imshow('Canny', canny)

cv2.waitKey()

cv2.destroyAllWindows()

**Результат:**



Число контуров функции threshold: 1253

Число контуров функции Canny: 1987

Различное число контуров объясняется применением детектора края Кенни к изображению. В результате получаются белые линии, которые тоже являются контурами.

**Задание №2.2**

**Код программмы:**

import cv2

from math import pi as pi

img1 = cv2.imread('C:/Users/greka/greka/technical vision/lab5\_21.png',

cv2.IMREAD\_COLOR)

img2 = cv2.imread('C:/Users/greka/greka/technical vision/lab5\_21.png',

cv2.IMREAD\_GRAYSCALE)

ret, tresh = cv2.threshold(img2, 150, 255, cv2.THRESH\_BINARY)

contours, hierarchy = cv2.findContours(tresh, cv2.RETR\_TREE,

cv2.CHAIN\_APPROX\_NONE)

print('Количество контуров функции threshold:'+ str(len(contours)))

cv2.drawContours(img1, contours, 1, (255, 0, 0), 2)

cv2.drawContours(img1, contours, 2, (0, 255, 0), 2)

outside = contours[1]

inside = contours[2]

#arcLength-вычисляет длину контура, contourArea-площадь контура

print('Периметр контура внешней окружности: ' + str(cv2.arcLength(outside, True)))

print('Площадь контура внешней окружности: ' + str(cv2.contourArea(outside)))

#координаты внешнего ограничивающего прямоугольника

x, y, w, h = cv2.boundingRect(outside)

cv2.rectangle(img1, (x, y), (x + w, y + h), (0, 0, 255), 2)

print('Площадь описывающего прямоугольник: ' + str(w\*h))

#координаты внешней ограничивающей окружности

(x,y), radius = cv2.minEnclosingCircle(outside)

#переводим их в int из float, для построения окружности

center = (int(x), int(y))

radius = int(radius)

cv2.circle(img1, center, radius, (255, 0, 255),  2)

print('Площадь внешней ограничивающей окружности: ' + str(pi\*(radius\*\*2)) + '\n')

print('Периметр контура внутренней окружности: ' + str(cv2.arcLength(inside, True)))

print('Площадь контура внутренней окружности: ' + str(cv2.contourArea(inside)))

# координаты внутреннего огр.прямоугольник:

x, y, w, h = cv2.boundingRect(inside)

cv2.rectangle(img1, (x, y), (x + w, y + h), (255, 255, 0), 2)

print('Площадь внутреннего ограничивающего прямоугольника: '+ str(w\*h))

#  координаты внутренней ограничивающей окружности:

(x,y), radius = cv2.minEnclosingCircle(inside)

center = (int(x), int(y))

radius = int(radius)

cv2.circle(img1, center, radius,(0, 255, 255), 2)

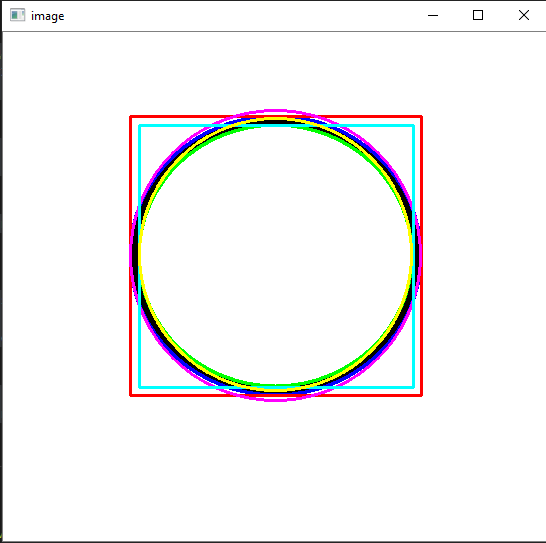
print('Площадь внутренней ограничивающей окружности: ' + str(pi\*(radius\*\*2)) + '\n')

cv2.imshow('image',img1)

cv2.waitKey()

cv2.destroyAllWindows()

**Результат работы:**



**Итог:**

Количество контуров функции threshold:3

Периметр контура внешней окружности: 941.5188946723938

Площадь контура внешней окружности: 63625.0

Площадь описывающего прямоугольник: 81189

Площадь внешней ограничивающей окружности: 66051.9855417254

Периметр контура внутренней окружности: 885.2346239089966

Площадь контура внутренней окружности: 55921.0

Площадь внутреннего ограничивающего прямоугольника: 71788

Площадь внутренней ограничивающей окружности: 58106.89772079681

**Дополнительное задание №1**

import cv2

import numpy as np

img1 = cv2.imread('C:/Users/greka/greka/technical vision/lab5\_1.png', cv2.IMREAD\_REDUCED\_COLOR\_2)

img2 = cv2.imread('C:/Users/greka/greka/technical vision/lab5\_1.png', cv2.IMREAD\_REDUCED\_GRAYSCALE\_2)

ret, tresh= cv2.threshold(img2, 150, 255, cv2.THRESH\_BINARY)

#ищем контуры

contours, hierarchy = cv2.findContours(tresh, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

print('Количество контуров функции threshold:'+ str(len(contours)))

index = 0

layer = 0

#функции для отслеживания слоев с помощью ползунков

def update():

    img3 = tresh.copy()

    cv2.drawContours(img3, contours, index, (255,0,0), 2, cv2.LINE\_AA, hierarchy, layer )

    cv2.imshow('contours', img3)

def update\_index(v):

    global index

    index = v-1

    update()

def update\_layer(v):

    global layer

    layer = v

    update()

update\_index(0)

update\_layer(0)

#сами ползунки, с которых получаю значение для индексов контура и слоя в иерархии

cv2.createTrackbar( "contour", "contours", 0, int(len(contours)), update\_index )

cv2.createTrackbar( "layers", "contours", 0, 15, update\_layer )

img4 = img1.copy()

#закрашивание контуров по четным, нечетным, беру по 2 рядом: внешний/внутренний контур

for i in  range(15,0,-4):

    cv2.drawContours(img4, contours,-1, (0,0,255),1, cv2.LINE\_AA, hierarchy, i)

    cv2.drawContours(img4, contours,-1, (0,0,255),1, cv2.LINE\_AA, hierarchy, i-1)

    cv2.drawContours(img4, contours,-1, (255,0,0),1, cv2.LINE\_AA, hierarchy, i-2)

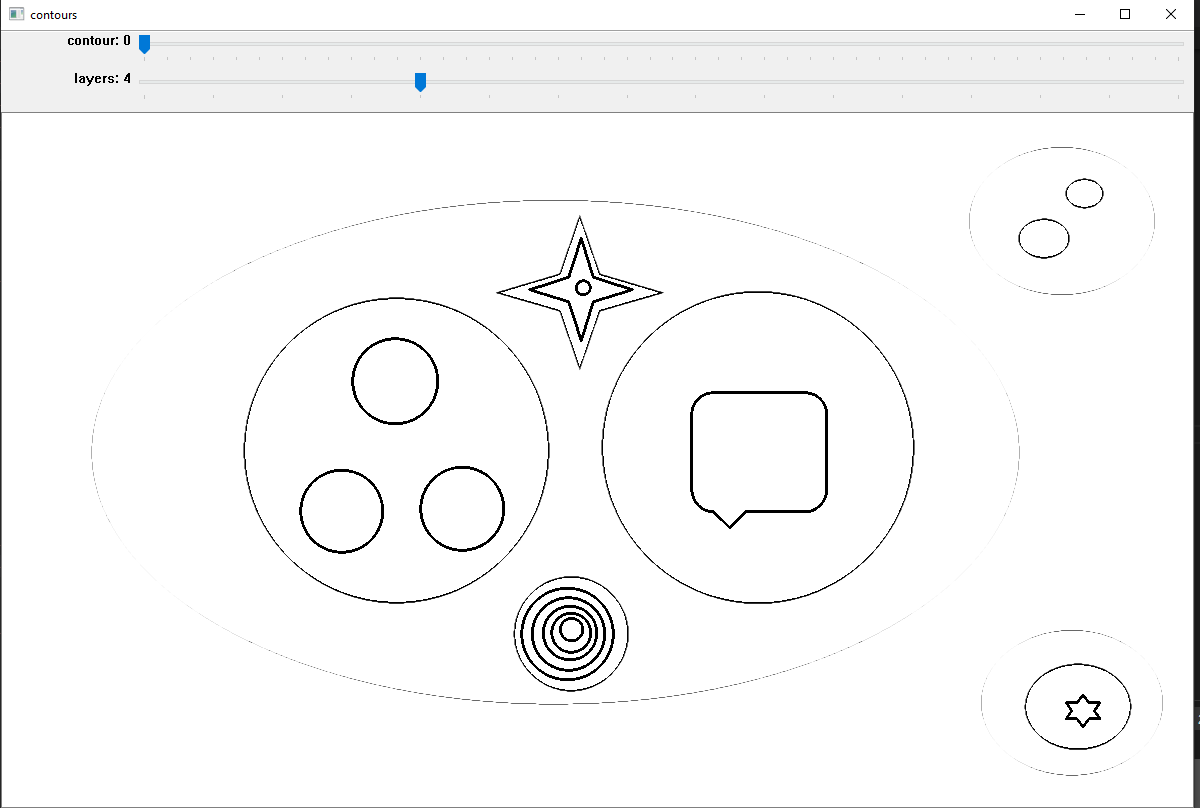
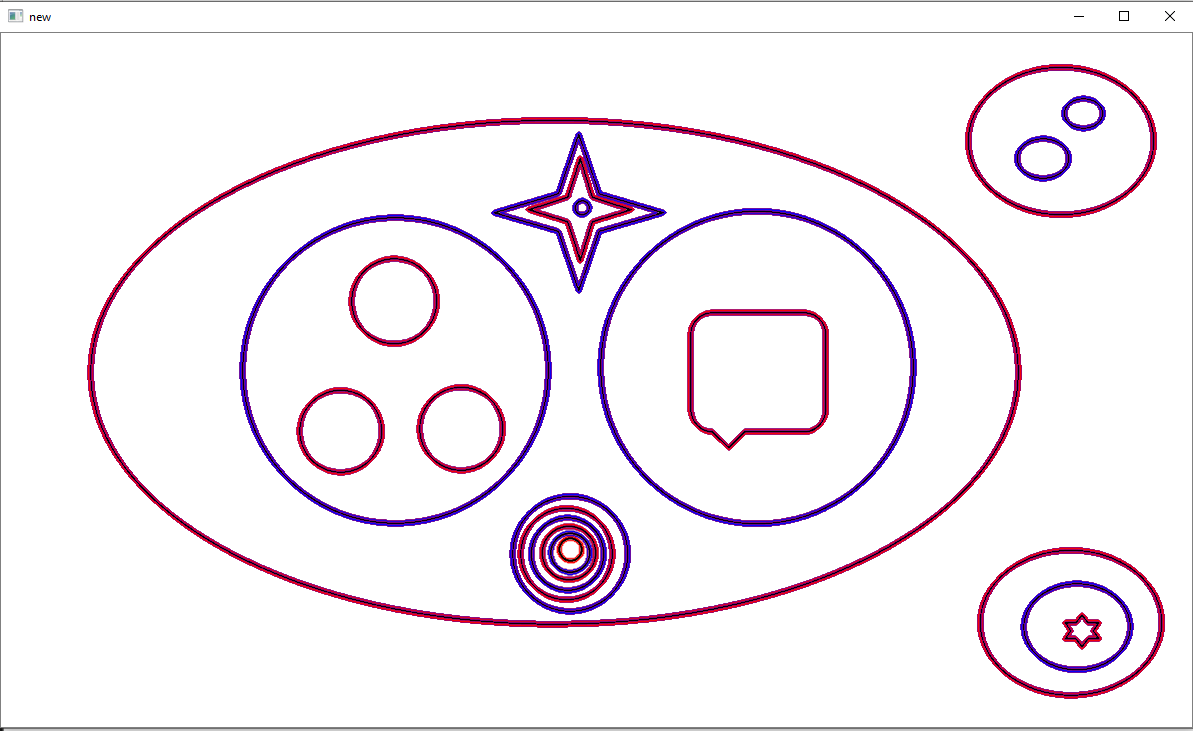
    cv2.drawContours(img4, contours,-1, (255,0,0),1, cv2.LINE\_AA, hierarchy, i-3)

cv2.imshow('new',img4)

cv2.waitKey()

cv2.destroyAllWindows()

**Результат работы:**

***Вывод*:** в ходе лабораторной работы были изучены способы выделения границ на изображении, поиск контуров на границах и получения информации об объектах на основе контуров. Так же работа с контурами и иерархией контуров