**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 |  | Ким А.В.  Фещенко М.В. |
| 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.

Задание №1

**Код программы:**

import cv2

#img = cv2.imread('chetko1.jpg', 0)

#img = cv2.imread('chetko2.jpg', 0)

#img = cv2.imread('mutno.jpg', 1)

img = cv2.imread('mutno1.jpg', 1)

sobel\_x = cv2.Sobel(src = img, ddepth = cv2.CV\_8U, dx = 1, dy = 0, ksize = 3)

sobel\_y = cv2.Sobel(img,ddepth = cv2.CV\_8U, dx = 0, dy = 1, ksize = 3)

sobel\_xy = cv2.Sobel(img, ddepth = cv2.CV\_8U, dx = 1, dy = 1, ksize = 3)

laplacian = cv2.Laplacian(img,cv2.CV\_8U, cv2.BORDER\_DEFAULT)

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

canny = cv2.Canny(image = blur, threshold1 = 170, threshold2 = 230, apertureSize = 3, L2gradient = True)

cv2.imshow('Original', img)

cv2.imshow('Sobel\_x', sobel\_x)

cv2.imshow('Sobel\_y', sobel\_y)

cv2.imshow('Sobel\_xy', sobel\_xy)

cv2.imshow('Laplacian', laplacian)

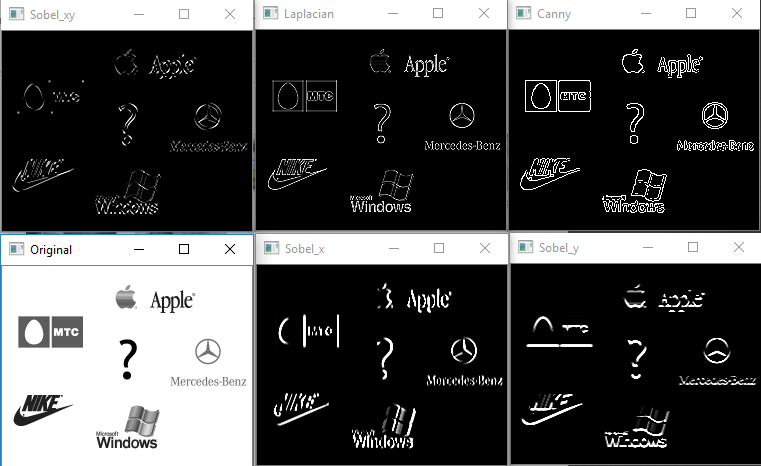
cv2.imshow('Canny', canny)

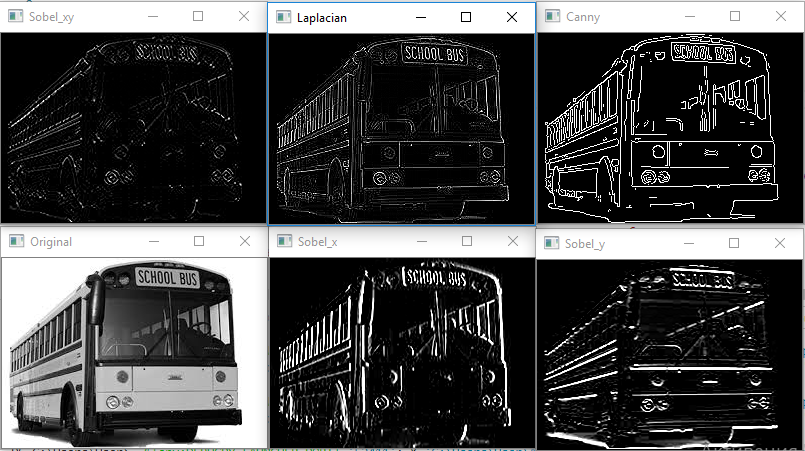
cv2.waitKey(0)

cv2.destroyAllWindows()

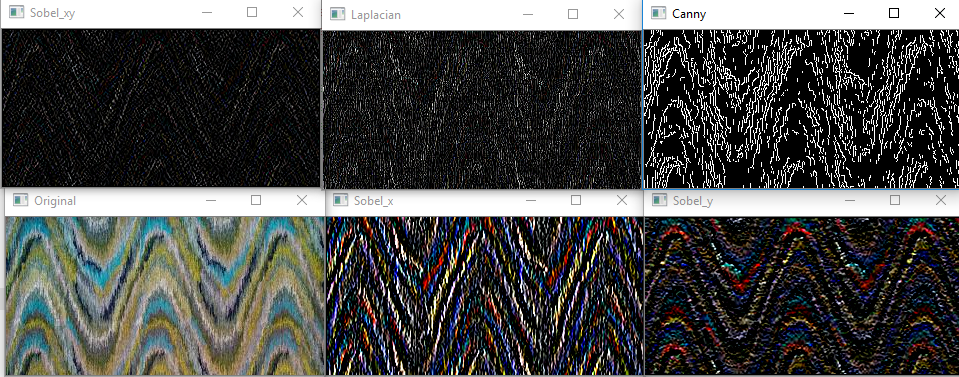
**Результат:**

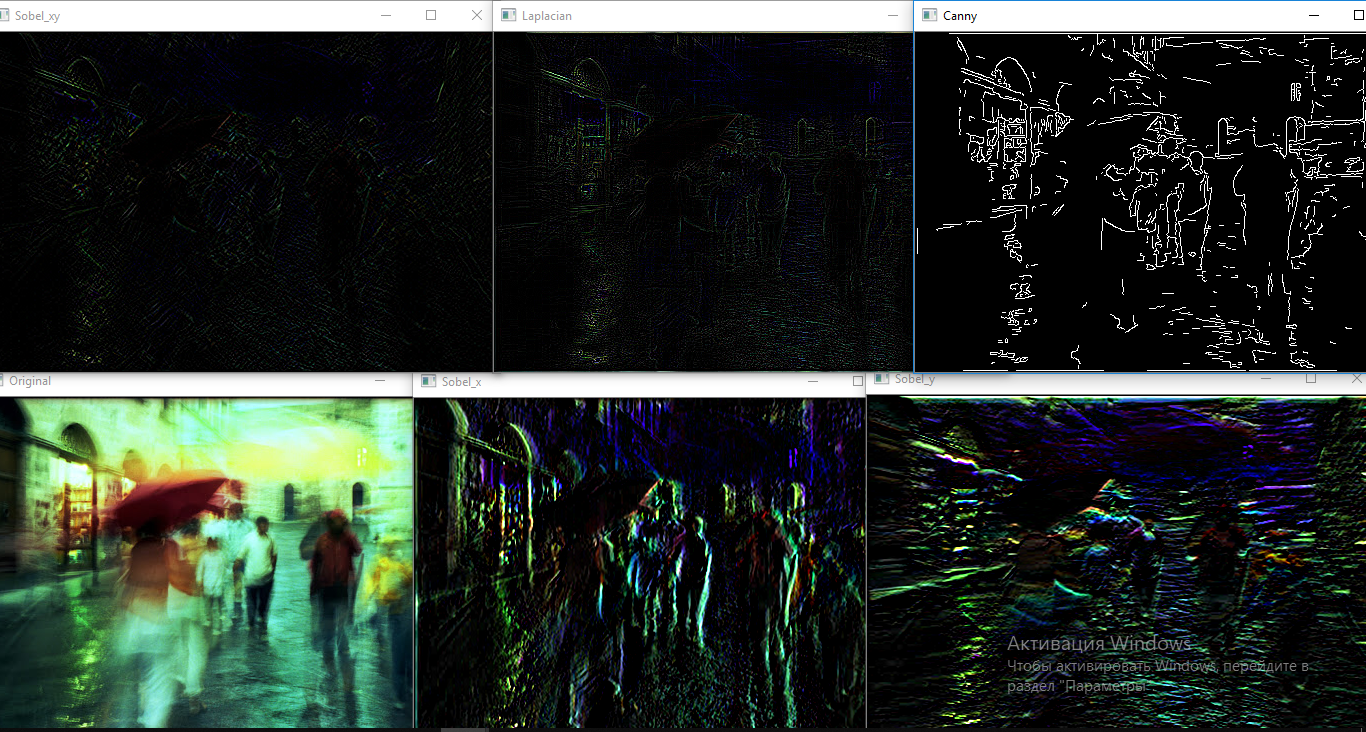
Изображения с четко отделимыми от фона границами:





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). По  умолчанию ksize = 3;

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

**Код программы:**

import cv2

img = cv2.imread('ubuntu.png')

img1 = cv2.imread('ubuntu.png', 0)

ret, thresh = cv2.threshold(img1, 170, 255, cv2.THRESH\_BINARY)

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

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

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

canny = cv2.Canny(image = blur, threshold1 = 170, threshold2 = 230, apertureSize = 3, L2gradient = True)

contours1, hierarchy = cv2.findContours(canny, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

print('Число контуров функции Canny:'+ str(len(contours1)) +'\n')

cv2.drawContours(img, contours1, -1, (30, 170, 230), 2)

cv2.imshow('Original', img)

cv2.imshow('Grayscale', img1)

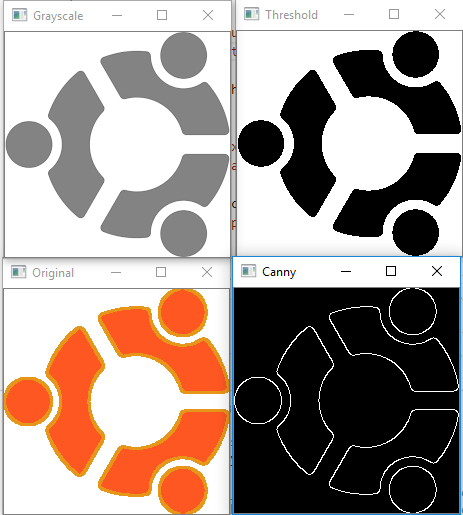
cv2.imshow('Threshold', thresh)

cv2.imshow('Canny', canny)

cv2.waitKey(0)

cv2.destroyAllWindows()

**Результат:**



Число контуров функции threshold: 5

Число контуров функции Canny: 10

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

circle = cv2.imread('circle.jpg')

circle1 = cv2.cvtColor(circle, cv2.COLOR\_BGR2GRAY)

ret, thresh = cv2.threshold(circle1, 150, 255, cv2.THRESH\_BINARY)

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

cv2.drawContours(circle, contours, 1, (30, 170, 230), 2)

cv2.drawContours(circle, contours, 2, (255, 170, 0), 2)

outside = contours[1]

inside = contours[2]

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

cv2.rectangle(circle, (x, y), (x + w, y + h), (0, 0, 255), 1)

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

center = (int(x),int(y))

radius = int(radius)

cv2.circle(circle,center,radius,(0, 255, 0), thickness = 1)

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

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

print('S внешнего прямоугольника: '+ str(w\*h))

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

x, y, w, h = cv2.boundingRect(inside)

cv2.rectangle(circle, (x, y), (x + w, y + h), (255, 0, 0),1)

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

center = (int(x),int(y))

radius = int(radius)

cv2.circle(circle,center,radius,(0,255,0),1)

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

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

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

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

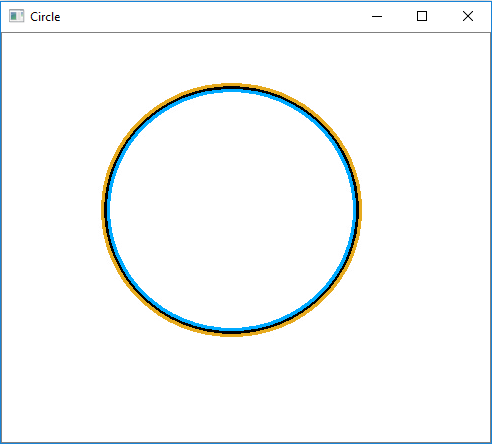
cv2.imshow('Circle', circle)

cv2.waitKey(0)

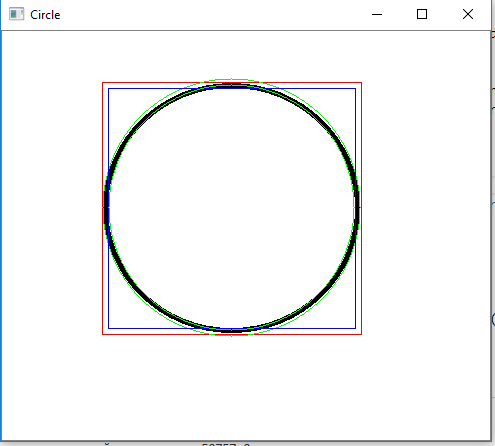
cv2.destroyAllWindows()

**Результат:**

Внутренний и внешний контуры окружности:



Итог:



P контура внешней окружности: 842.2640614509583

S контура внешней окружности: 50757.0

S внешнего прямоугольника: 65268

S внешней ограничивающей окружности: 52279.24334838775

P контура внутренней окружности: 803.6366448402405

S контура внутренней окружности: 46182.0

S внутреннего прямоугольника: 59280

S внутренней ограничивающей окружности: 47529.15525615998

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