**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.**

Студент гр. 6492 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Мурашко А.С.

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Моклева К.А.

Санкт-Петербург

2020

**Лабораторная работа №5**

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

**Ход работы.**

Поиск границ:

Выбираем изображение с четкими границами

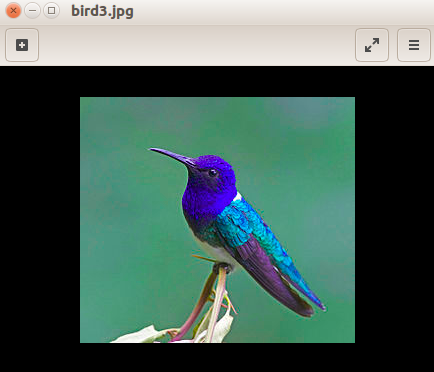


Рисунок 1 — Исходное изображение птички.

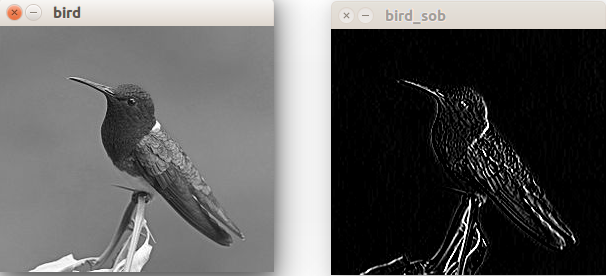
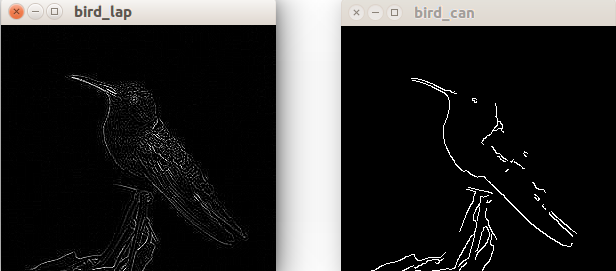
Рисунок 2 — Птичка в сером Рисунок 3 — Оператор Собеля 

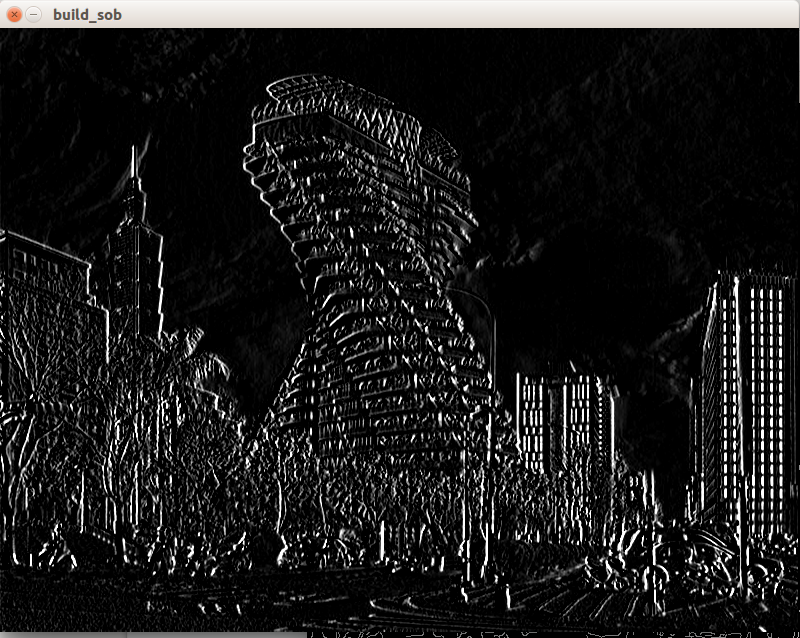
Рисунок 4 — Оператор Лапласса Рисунок 5 — Детектор Кенни

Изображение с нечеткими границами

Рисунок 6 — Исходное изображение здания



Рисунок 7 — Здание в сером

Рисунок 8 — Оператор Собеля

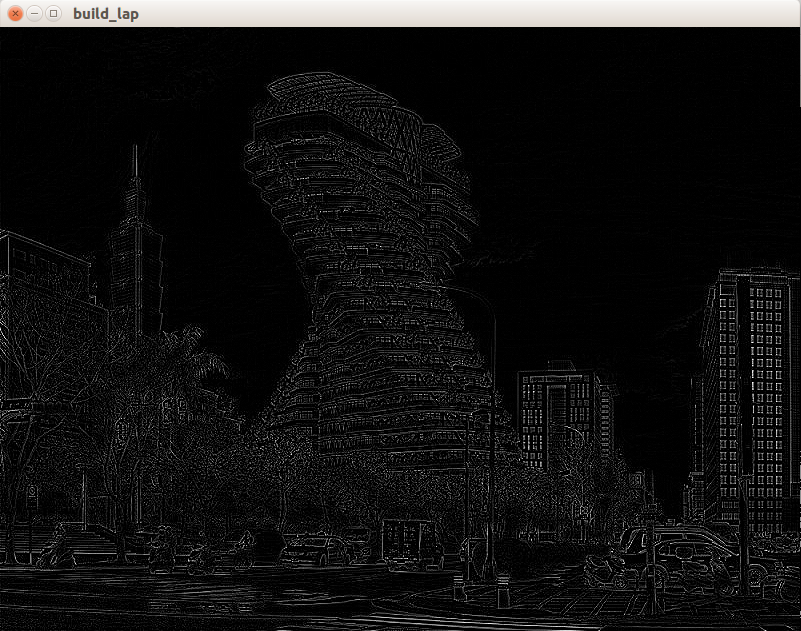


Рисунок 9 — Оператор Лапласа

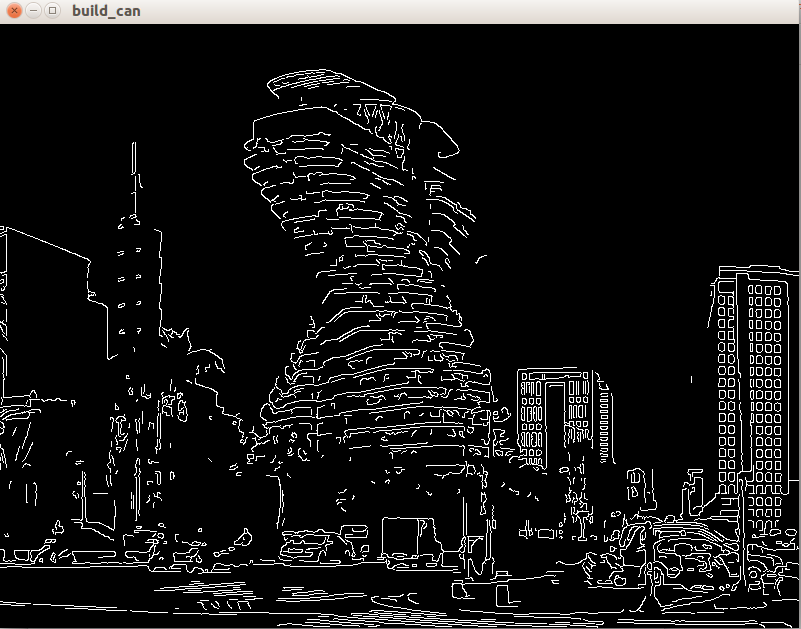


Рисунок 10 — Детектор Кенни

Вывод по границам:

Лучше всего границы видны при использовании оператора Лапласа, но у детектора Кенни границы видны ярче. У оператора Собеля границы видны хуже всего, на мой взгляд.

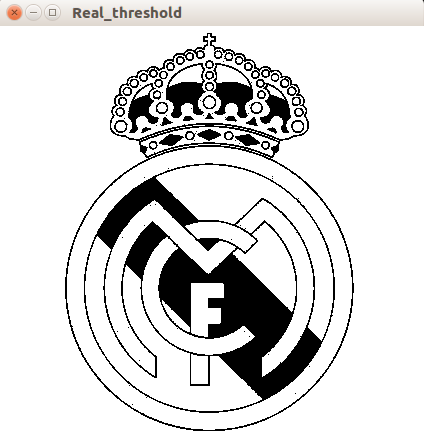
Все зависит от параметров, которые мы передаем в функции, а именно:

Параметр ddepth - глубина выходного изображения. Она может быть следующей: CV\_8U, CV\_16U, CV\_16S, CV\_32F или CV\_64F. Чем больше мы ее берем, тем больше будет видно границ. Почти все изображение будет в границах. Поэтому, чтобы были видны только самые основные границы и самые важные, лучше использовать CV\_8U.

Но у детектора Кенни по-другому. У него мы задаем пороговые значения. Чем больше нижний порог, тем меньше будет границ, так как больше пикселей будет отсеиваться. Тоже самое и с верхним порогом. При высоком пороговом значении границы будут видны только при большой разницы в яркости между пикселями.

Ksize = 3 - это размер ядра Собеля (по умолчанию), порядок производной должен быть меньше, поэтому в работе используется xorder = 1, yorder = 0.

При использовании детектора Кенни параметр L2gradient задан как True, чтобы градиент яркости вычислялся точнее.

Поиск контуров:

Проверим функцию поиска границ findContours на бинарном изображении, полученном с помощью threshold и с помощью детектора границ Кенни.

Рисунок 11 — Threshold Рисунок 12 — Детектор Кенни

Contours threshold = 192

Contours Canny = 249

Контуров у изображения, полученного с помощью детектора Кенни больше, чем у изображения threshold, так как детектор Кенни выделил больше границ, чем их есть на исходном изображении. Поэтому функция findContours находит больше контуров.

Окружность:

Найдем для окружности внутренний и внешний контур. Вычислим их длину и площадь.

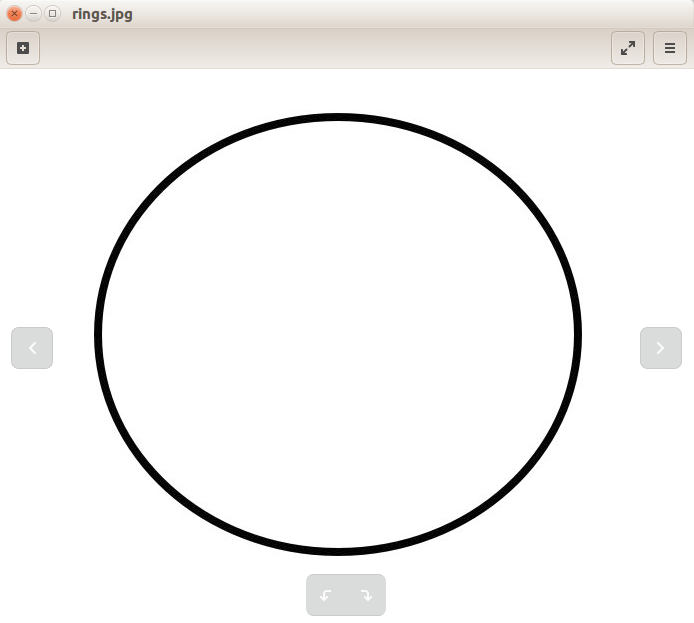


Рисунок 13 — Исходное изображение окружности

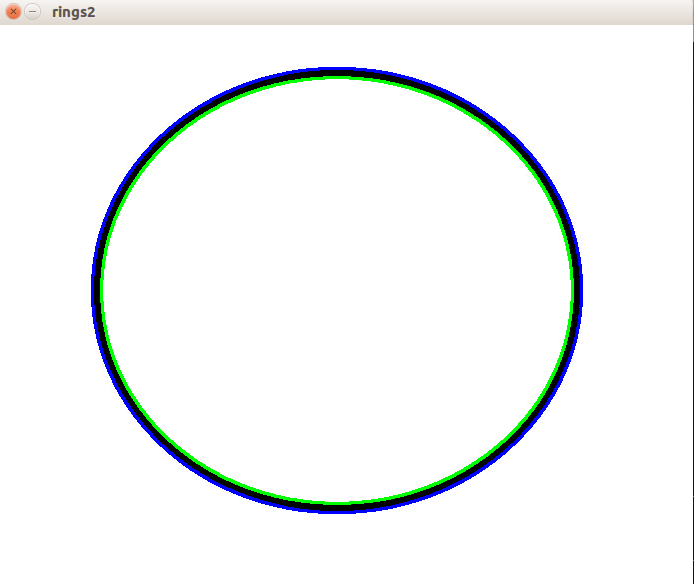


Рисунок 14 — Окружность с выделенным внешним и внутренним контуром

**Результаты:**

Contours Rings = 3

P inside = 1487.04

S inside = 157487.0

P out = 1547.33

S out = 170608.0

Длина и площадь внешнего контура больше, так как внешний контур больше внутреннего и , соответственно, у него больше количество писелей.

Находим ограничивающие прямоугольник и окружность для внешнего и внутреннего контура.

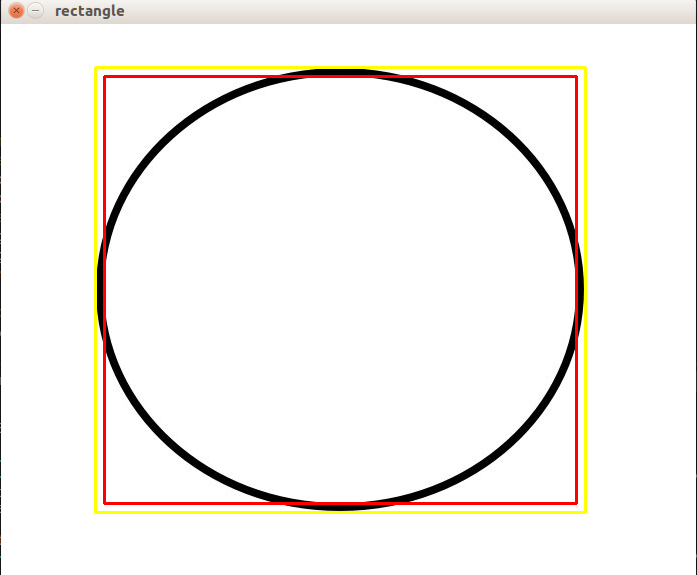


Рисунок 15 — Ограничивающие прямоугольники

S inside rectangle = 201544

S outside rectangle = 218050

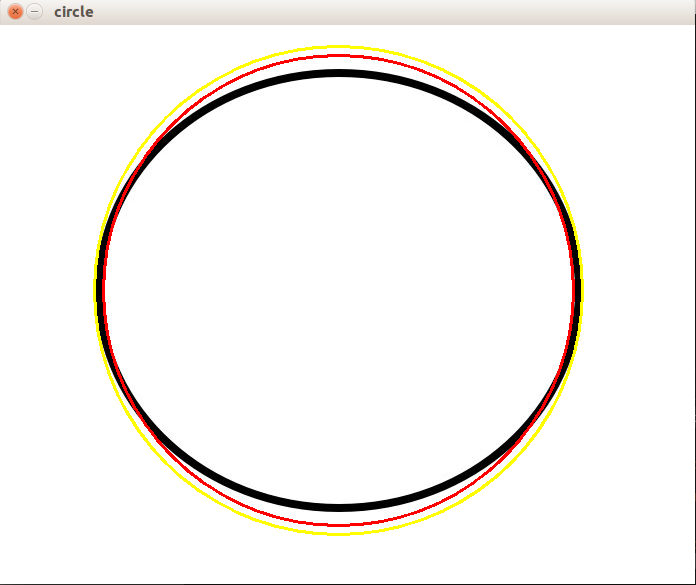


Рисунок 16 — Ограничивающие окружности

S inside circle = 173406.5

S outside circle = 186943.04

Площади ограничивающих фигур больше площадей контуров, так как они больше в размере, чем контура.

**Код программы.**

import cv2

# ------------------------------- ГРАНИЦЫ -------------------------------------------

# читаем исходное изображение с четкими границами

img1 = cv2.imread('/home/andrew/foto/bird3.jpg', cv2.IMREAD\_GRAYSCALE)

cv2.imshow('bird', img1)

# поиск границ оператором Собеля

sob1 = cv2.Sobel(img1, cv2.CV\_8U , 1, 0, ksize = 3, scale = 1, delta = 0, borderType = cv2.BORDER\_DEFAULT)

cv2.imshow('bird\_sob', sob1)

# поиск границ оператором Лапласа

lap1 = cv2.Laplacian(img1, cv2.CV\_8U, 3)

cv2.imshow('bird\_lap', lap1)

# поиск границ детектором Кенни

# для этого предварительно размыв изображение с помощью фильтра Гаусса

gauss1 = cv2.GaussianBlur(img1, (3, 3), sigmaX = 0, sigmaY = 0, borderType = cv2.BORDER\_DEFAULT)

can1 = cv2.Canny(gauss1, threshold1 = 150, threshold2 = 220, apertureSize = 3, L2gradient = True)

cv2.imshow('bird\_can', can1)

cv2.waitKey(0)

# читаем исходное изображение с нечеткими границами

img2 = cv2.imread('/home/andrew/foto/build2.jpg', cv2.IMREAD\_GRAYSCALE)

cv2.imshow('build', img2)

# поиск границ оператором Собеля

sob2 = cv2.Sobel(img2, cv2.CV\_8U , 1, 0, ksize = 3, scale = 1, delta = 0, borderType = cv2.BORDER\_DEFAULT)

cv2.imshow('build\_sob', sob2)

# поиск границ оператором Лапласа

lap2 = cv2.Laplacian(img2, cv2.CV\_8U, 3)

cv2.imshow('build\_lap', lap2)

# поиск границ детектором Кенни

# для этого предварительно размыв изображение с помощью фильтра Гаусса

gauss2 = cv2.GaussianBlur(img2, (3, 3), sigmaX = 0, sigmaY = 0, borderType = cv2.BORDER\_DEFAULT)

can2 = cv2.Canny(gauss2, threshold1 = 150, threshold2 = 220, apertureSize = 3, L2gradient = True)

cv2.imshow('build\_can', can2)

cv2.waitKey(0)

# ------------------- ПОИСК КОНТУРОВ ------------------------------------

# поиск контуров с помощью функции threshold

img = cv2.imread('/home/andrew/foto/real.jpg', cv2.IMREAD\_GRAYSCALE)

threshold, img\_thr = cv2.threshold(img, 170, 255, cv2.THRESH\_BINARY)

cv2.imshow('Real\_threshold', img\_thr)

cv2.waitKey(0)

contours\_thr, hierarchy\_thr = cv2.findContours(img\_thr, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

print('Contours threshold =', len(contours\_thr))

# поиск контуров с помощью детектора границ Кенни

gauss = cv2.GaussianBlur(img, (3, 3), sigmaX = 0, sigmaY = 0, borderType = cv2.BORDER\_DEFAULT)

img\_can = cv2.Canny(gauss, threshold1 = 150, threshold2 = 250, apertureSize = 3, L2gradient = True)

cv2.imshow('Real\_Canny', img\_can)

cv2.waitKey(0)

contours\_can, hierarchy\_can = cv2.findContours(img\_can, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

print('Contours Canny =', len(contours\_can))

# ------------------------- ОКРУЖНОСТИ -------------------------------------

# читаем изображение с окружностью

rings = cv2.imread('/home/andrew/foto/rings.jpg')

rings\_gray = cv2.cvtColor(rings, cv2.COLOR\_BGR2GRAY)

# используем пороговый фильтр BINARY

threshold\_rings, rings\_thr = cv2.threshold(rings\_gray, 170, 250, cv2.THRESH\_BINARY)

# находим контура

contours, hierarchy = cv2.findContours(rings\_thr, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

outside = contours[1]

inside = contours[2]

print('Contours Rings =', len(contours))

# рисуем внутреннюю(зеленая) и внешнюю(синяя) окружность

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

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

cv2.imshow('rings2', rings)

cv2.waitKey(0)

# Выводим значение длины окружности и ее площадь

print('P inside = ', cv2.arcLength(inside, True))

print('S inside = ', cv2.contourArea(inside))

print('P out = ', cv2.arcLength(outside, True))

print('S out = ', cv2.contourArea(outside))

# ------------- ограничивающий прямоугольник ----------------

# Находим координаты вершины левого верхнего угла ограничивающего прямоугольника, его ширину и высоту

x1, y1, w1, h1 = cv2.boundingRect(inside)

x2, y2, w2, h2 = cv2.boundingRect(outside)

# Рисуем ограничивающий внешний и внутренний прямоугольник

cv2.rectangle(rings, (x1, y1), (x1+w1, y1+h1), (0, 0, 255), 2)

cv2.rectangle(rings, (x2, y2), (x2+w2, y2+h2), (0, 255, 255), 2)

cv2.imshow('rectangle', rings)

cv2.waitKey(0)

# Рассчитываем площадь ограничивающих прямоугольников

S\_in\_rectangle = w1\*h1

S\_out\_rectangle = w2\*h2

print('S inside rectangle = ', S\_in\_rectangle)

print('S outside rectangle = ', S\_out\_rectangle)

# ------------- ограничивающая окружность ----------------

# Находим координаты центра и радиус ограничивающих окружностей

(x1, y1), r1 = cv2.minEnclosingCircle(inside)

(x2, y2), r2 = cv2.minEnclosingCircle(outside)

# Преобразуем полученные значения в int

center1 = (int(x1), int(y1))

radius1 = int(r1)

center2 = (int(x2), int(y2))

radius2 = int(r2)

# Рисуем внешнюю(красный) и внутреннюю(желтый) ограничивающие окружности

cv2.circle(rings, center1, radius1, (0, 0, 255), 2)

cv2.circle(rings, center2, radius2, (0, 255, 255), 2)

cv2.imshow('circle', rings)

cv2.waitKey(0)

# Вычисляем площадь ограничивающих окружностей

S\_in\_circle = 3.14\*(radius1\*\*2)

S\_out\_circle = 3.14\*(radius2\*\*2)

print('S inside circle = ', S\_in\_circle)

print('S outside circle = ', S\_out\_circle)

**Вывод:** в ходе работы была изучена работа с границами и контурами изображений. Были исследованы различные методы определения границ на изображении, изучены способы определения и рисования контуров. Также разобраны различные операции с контурами, такие как вычисление площади, длины контуров.