Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»**

**(ФГБОУ ВО «КубГУ»)**

**Факультет компьютерных технологий и прикладной математики**

**Кафедра вычислительных технологий**

**Отчёт**

**по лабораторной работе №3**

**Дисциплина: АЛГОРИТМЫ ЦИФРОВОЙ ОБРАБОТКИ МУЛЬТИМЕДИА**

**Тема: «МЕТОДЫ ВЫДЕЛЕНИЯ ГРАНИЦ. АЛГОРИТМ КАННИ.»**

Работу выполнила \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д. Ш. Пивоварова

Направление подготовки 02.03.02 Фундаментальная информатика и

информационные технологии

Направленность (профиль) Математическое и программное обеспечение

компьютерных технологий

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.С. Жук

**Цель работы**

Цель работы: научиться работать с методами выделения границ, реализовав алгоритм Канни.

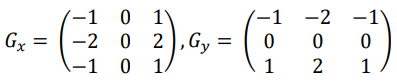
(градиент – вектор частных производных)

**Ход работы**

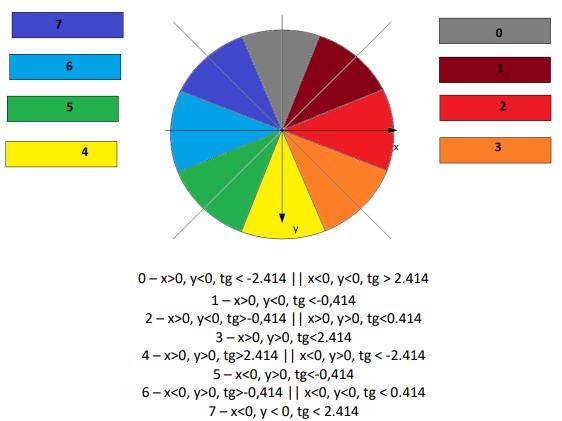
Работа будет осуществляться с помощью средств языка Python 3.10 и IDE PyCharm2021.3.2 с учебной лицензией. Для работы необходимо установить библиотеку opencv.

Алгоритм.

1. Применим к картинке чёрно-белый фильтр.
2. Применим к картинке размытие Гаусса. Для этого воспользуемся конструктором GaussianBlur из пакета opencv.
3. Найдём матрицы для значений частных производных, длины градиента и угла градиента. Для каждого пикселя применим оператор Собеля (матрицы свёртки) для вычисления (Gx, Gy), найдём длину вектора градиента |dI|=sqrt(Gx2+Gy2), найдём tg(f)=Gy/Gx, найдём округлённое значение угла от 0 до 7 (пункт 4).

  
Рисунок 1 – Оператор Собеля

1. Найдём длину вектора для каждого пикселя и сравним пиксель с теми пикселями, по направлению которых значение функции отличается сильнее всего. Для этого вычислим направление и найдём величину угла градиента. Точно знать угол не нужно, важно выбрать 1 из 8 соседних пикселей, поэтому величину угла округлим до 45 градусов. Воспользуемся следующей схемой:

  
Рисунок 2 – Схема округления углов

1. Подавление немаксимумов. Границей считается пиксель, градиент которого максимален в сравнении с пикселями по направлению наибольшего роста функции. Направление задано вдоль оси. Если значение градиента выше, чем у пикселей слева и справа, то данный пиксель – граница, иначе – не граница. В граничные пиксели выставим значение 0, в остальные 255.
2. Двойная пороговая фильтрация. Выбираем 2 пороговых значения для градиента – high и low. Если градиент больше high – точно граница, меньше low – точно не граница. После этого остаются пиксели, значение градиента которых заключено между границами. Для них: если пиксель – граница, то он не может быть отдельной границей, рядом должен быть ещё пиксель с границей. Добавим проверку, что рядом с границей есть другая граница, для чего проверяем 8 пикселей вокруг заданного.

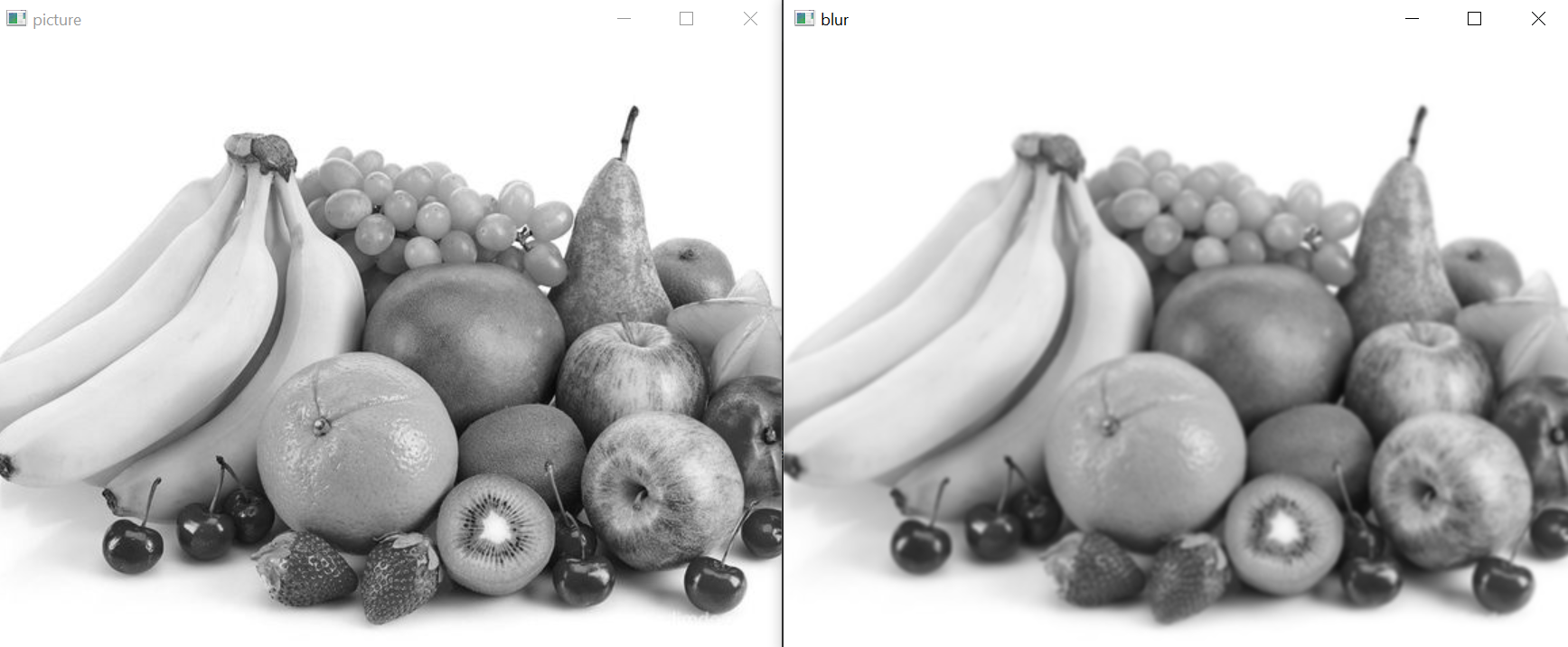


Рисунок 3 – Исходное изображение

Задание 1. Реализовать метод, который принимает в качестве строки полный адрес файла изображения, читает изображение, переводит его в чёрно-белый цвет и выводит его на экран, применяет размытие по Гауссу и выводит полученное изображение на экран.

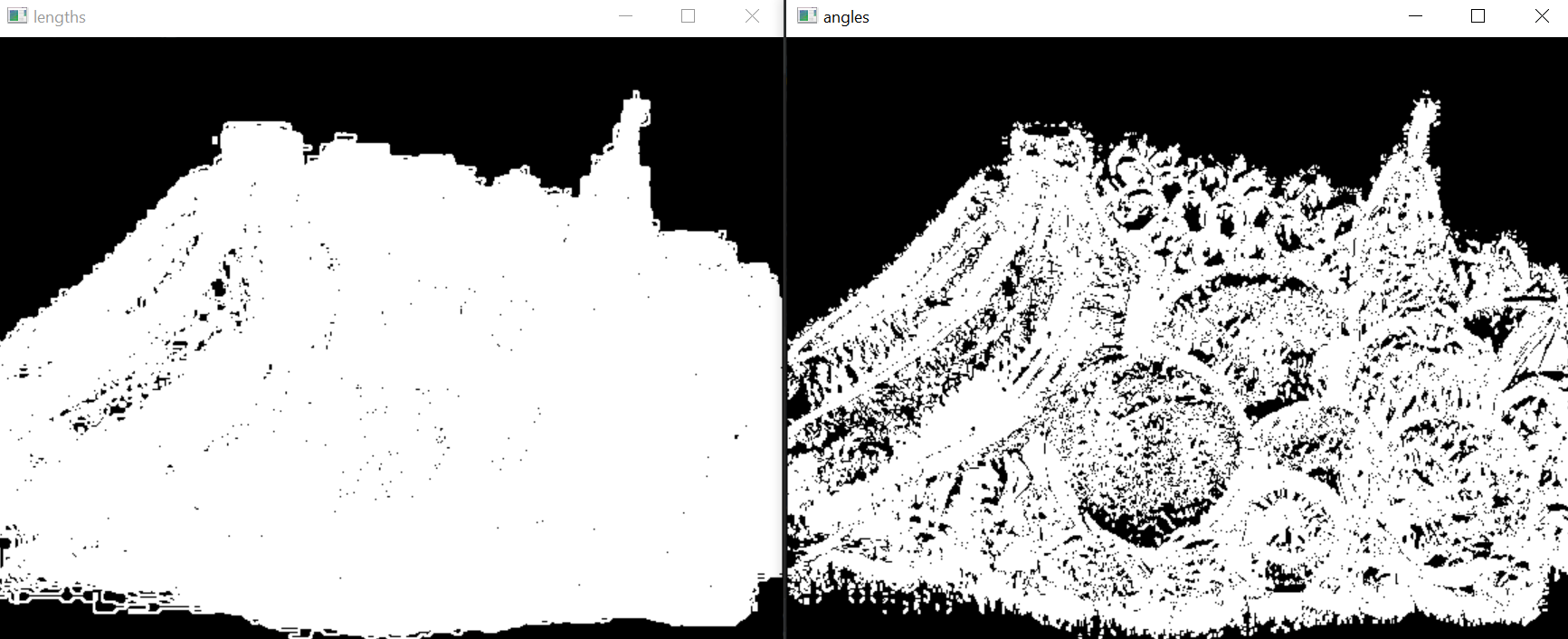
Для чтения картинки используется метод imread(path, flag). Для отображения картинки – метод imshow(name, image). Чтобы прочитать картинку в чёрно-белом цвете, используем flag=cv2.IMREAD\_ANYDEPTH. Чтобы применить размытие по Гауссу, используем функцию cv2.GaussianBlur(picture, (size, size), sigma).

Реализованы пункты 1-2 алгоритма.

  
Рисунок 4 – Результат задания 1

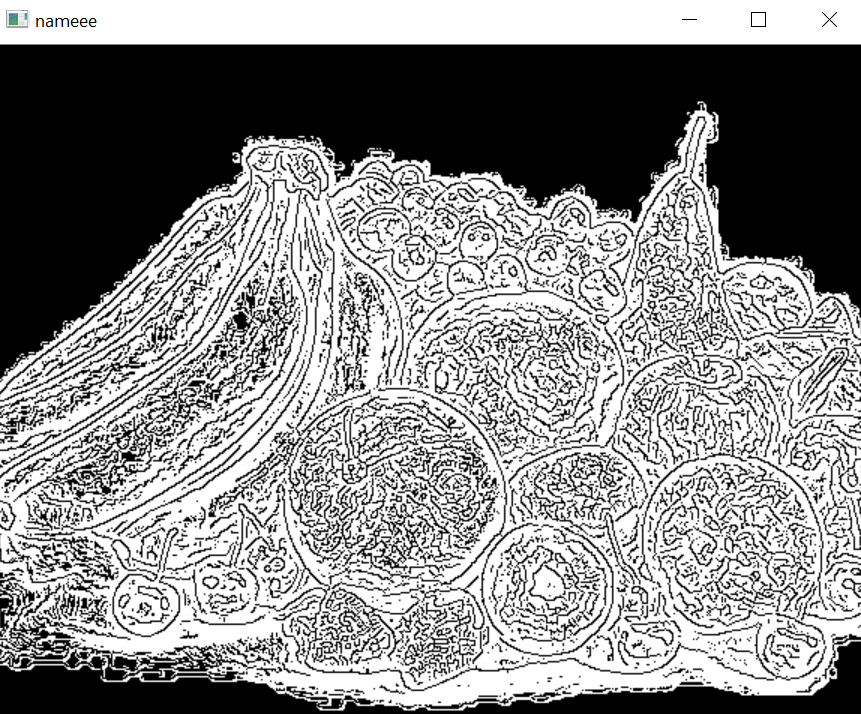
Задание 2. Модифицировать построенный метод так, чтобы в результате вычислялось и выводилось на экран две матрицы – матрица значений длин и матрица значений углов градиентов всех пикселей изображения.

Реализованы пункты 3-4 алгоритма.

  
Рисунок 5 – Результат задания 2

Задание 3. Модифицировать метод так, чтобы он выполнял подавление немаксимумов и выводил полученное изображение на экран.

Реализован пункт 5 алгоритма.

  
Рисунок 6 – Результат задания 3

Задание 4. Модифицировать метод так, чтобы он выполнял двойную пороговую фильтрацию и выводил полученное изображение на экран.

Реализован пункт 6 алгоритма.

  
Рисунок 7 – Результат задания 4

Задание 5. Провести опыты для различных параметров размытия и различных пороговых значений градиента, определить наилучшие параметры для вашего изображения.

  
Рисунок 8 – Ядро свёртки 3х3

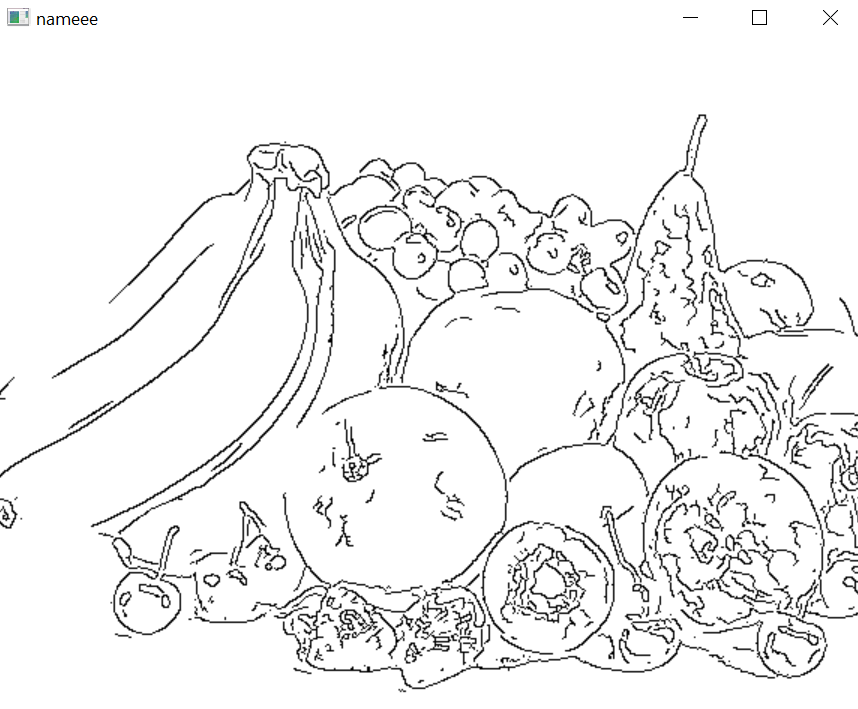
  
Рисунок 9 – Ядро свёртки 5х5

  
Рисунок 10 – Ядро свёртки 9х9

  
Рисунок 11 – Ядро свёртки 21х21

Как видно из рисунков, при увеличении размерности ядра свёртки детализация падает, однако более чётко виден общий контур. При этом, разницы между 9х9 и 21х21 практически не видно. Для своего изображения я выбрала 5х5.

  
Рисунок 12 – Нижний порог величины градиента 0, верхний 100

  
Рисунок 13 – Нижний порог величины градиента 100, верхний 150

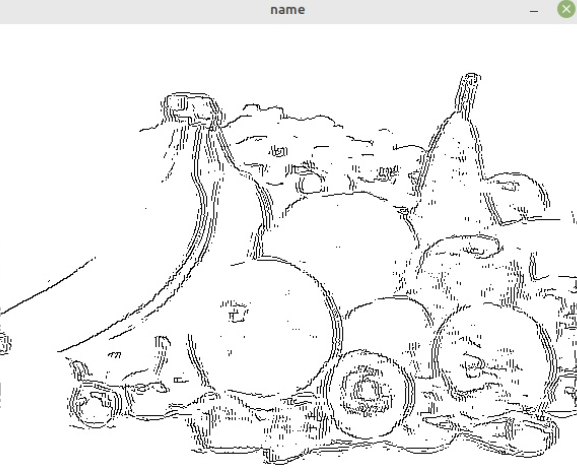
  
Рисунок 14 – Нижний порог величины градиента 200, верхний 250

  
Рисунок 15 – Нижний порог величины градиента 0, верхний 50

Из рисунков можно увидеть, что самыми удачными параметрами оказались нижний порог 0 и верхний 50. Это означает, что значения выше 50 отсекать не стоит. Если брать верхний порог выше 100, то общие границы видны более чётко, но при этом куда-то делся один банан.

Задание 6. Реализовать алгоритм Канни на другом языке программирования.

Задание выполнено на языке C++.

  
Рисунок 16 – Результат задания 6

**Вывод**

Была изучена работа с методами выделения границ на примере алгоритма Канни.