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**

**ЛАБОРАТОРНАЯ РАБОТА №3**

**Дисциплина: Алгоритмы цифровой обработки мультимедиа**

Работу выполнил: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_C.A. Фролов

Направление подготовки: 02.03.02 Фундаментальная информатика и информационные технологии

Преподаватель: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_А.А. Крамаренко

Лабораторная работа №3

Методы фильтрации и размытия изображения.

Размытие – один из базовых алгоритмов обработки изображения. Размытие строится из предположения, что каждый пиксель картинки по своему значению связан с предыдущим и не должен заметно от него отличатся. Если же так происходит, например, вокруг пикселя яркость 240, а в самом

пикселе 12, то мы предполагаем, что данный пиксель – это шумы. И размытие картинки – один из главных способов нивелировать эти шумы. Базовый подход к фильтрации строится на том, что значение пикселя должно

изменится под влиянием окружающих пикселей.

Для начала переведем изображение в черно-белый формат, в этом случае

каждый пиксель представляет собой только одно целое неотрицательное

значение длиной 8 бит – яркость. Далее рассмотрим изображение как

двумерную матрицу размерности ℎ × 𝑤:

Теперь рассмотрим произвольный пиксель данной матрицы 𝑎𝑖𝑗:

Значение яркости данного пикселя будем менять с учетом всех указанных пикселей вокруг заданного. Для этого необходимо выполнить операцию свертки. Операция свертки заключается в преобразовании исходной матрицы B размерности 𝑛 × 𝑛 в числовое значение с помощью специальной матрицы ker размерности 𝑛 × 𝑛, называемой ядром свертки:

Матрица ker берется с различными значениями для разных типов фильтрации, стандартное условие – данная матрица должна иметь сумму элементов близкую к единице.

Гауссовский фильтр построен на ядре свертки, полученной как значения функции Гаусса (плотность распределения) для двумерной случайной величины:

Здесь 𝑎, 𝑏 – математическое ожидание двумерной случайной величины.

Для построения матрицы в качестве этих значений выбираем координаты

центрального элемента матрицы, например, для матрицы 5 × 5 значения

математического ожидания будут следующими: 𝑎 = 3, 𝑏 = 3. В качестве 𝑥, 𝑦

выбираем все индексы всех элементов матрицы поочередно. Значение 𝜎 будем считать параметром размытия по Гауссу. При каждом новом значении 𝜎 результат размытия будет отличаться.

Размерность матрицы свертки – это другой возможный параметр фильтрации.

Далее применяем операцию свертки ко всем внутренним пикселям

изображения и записываем вычисленное по формуле (1) число в качестве

нового значения яркости пикселя.

Итого алгоритм кратко будет выглядеть следующим образом:

* установить размер матрицы свертки и среднее квадратичное отклонение;
* заполнить матрицу свертки значениями функции Гаусса с мат.
* ожиданием, равным координатам центра матрицы;
* нормировать матрицу так, чтобы сумма элементов равнялась 1;
* создать копию изображения;
* для каждого внутреннего пикселя копии рассчитать новое значение
* насыщенности пикселя по формуле (1) и ЗАПИСАТЬ это значение в пиксель нового изображения

**Задание 1**. Выполнить пункты 1 и 2 алгоритма, то есть построить

матрицу Гаусса. Просмотреть итоговую матрицу для размерностей 3, 5, 7.

**Задание 2.** Нормировать полученную матрицу Гаусса. Протестировать

результаты на матрицах из предыдущего пункта.

**Задание 3**. Реализовать фильтр Гаусса средствами языка python.

**Задание 4.** Применить данный фильтр для двух разных значений

среднего квадратичного отклонения и двух разных размерностей матрицы свертки, сравнить результаты для ОДНОГО изображения.

**Задание 5.** Реализовать размытие Гаусса встроенным методом библиотеки OpenCV, сравнить результаты с Вашей реализацией.  
Итоговые размытия:



Рисунок 1- исходное изображение



Рисунок 2- Размытие от OpenCV



Рисунок 3- размытие 5х5 и стандартным отклонением 100



Рисунок 4- размытие 11х11 и стандартным отклонением 50

**Листинг программы**

import cv2  
import numpy as np  
  
def CVBlur(img, kernel\_size, deviation):  
 return cv2.GaussianBlur(img, (kernel\_size, kernel\_size), deviation)  
  
def GaussBlur(img, kernel\_size, standard\_deviation):  
 kernel = np.ones((kernel\_size, kernel\_size))  
 a = b = (kernel\_size + 1) // 2  
  
 for i in range(kernel\_size):  
 for j in range(kernel\_size):  
 kernel[i, j] = gauss(i, j, standard\_deviation, a, b)  
  
  
 print("//////////")  
 sum = 0  
 for i in range(kernel\_size):  
 for j in range(kernel\_size):  
 sum += kernel[i, j]  
  
 for i in range(kernel\_size):  
 for j in range(kernel\_size):  
 kernel[i, j] /= sum  
  
 print(kernel)  
  
 imgBlur = img.copy()  
 x\_start = kernel\_size // 2  
 y\_start = kernel\_size // 2  
 for i in range(x\_start, imgBlur.shape[0] - x\_start):  
 for j in range(y\_start, imgBlur.shape[1] - y\_start):  
 val = np.sum(img[i - kernel\_size//2: i + kernel\_size//2 + 1, j - kernel\_size//2: j + kernel\_size//2 + 1] \* kernel)  
 imgBlur[i, j] = val  
  
 return imgBlur  
  
  
def gauss(x, y, omega, a, b):  
 omega2 = 2 \* omega \*\* 2  
  
 m1 = 1 / (np.pi \* omega2)  
 m2 = np.exp(-((x-a) \*\* 2 + (y-b) \*\* 2) / omega2)  
  
 return m1 \* m2  
  
  
def Show():  
 orig = cv2.imread('Bochka.jpg',cv2.IMREAD\_GRAYSCALE)  
 img = GaussBlur(orig, 11, 50)  
 cv2.imshow('5x5 Div 100', img)  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()  
  
  
Show()