МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ

ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Кафедра Вычислительной Техники



**Курсовая работа**

по дисциплине: «Архитектура средств вычислительной техники»

на тему: «Программа медианной фильтрации полутонового   
зашумленного изображения»

Выполнил:Проверил:

Студент гр. «*АВТ-819*» *к.т.н. доцент*

*Ванин К.Е. Овчеренко В.А.*

«\_\_» \_\_\_\_\_ 2021 г.«\_\_\_» \_\_\_\_\_\_ 2021 г.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись) (подпись)

Новосибирск

2021

Оглавление

[Цель 3](#_Toc59919076)

[Задача 3](#_Toc59919077)

[Ход работы 3](#_Toc59919078)

[Медианный фильтр 3](#_Toc59919079)

[Медианный фильтр в среде MATLAB 4](#_Toc59919080)

[Медианный фильтр на языке программирования Python 6](#_Toc59919081)

[Вывод 8](#_Toc59919082)

[Список литературы 9](#_Toc59919083)

[Приложение 1 10](#_Toc59919084)

[Приложение 2 11](#_Toc59919085)

# Цель

Разработка программы для прямоугольного медианного фильтра размером 5х5 при помощи высокоуровневых языков программирования.

# Задача

Разработать программу медианной фильтрации полутонового зашумленного изображения. Изображение 256х256х256 градаций серого. Медианный фильтр прямоугольный, размером 5х5. Шум импульсный. Зашумление исходного изображения -50%. Три прохода фильтрации.

# Ход работы

## Медианный фильтр

При применении медианного фильтра происходит последовательная обработка каждой точки кадра, в результате чего образуется последовательность оценок. В идейном отношении обработка в различных точках независима, но в целях ее ускорения целесообразно алгоритмически на каждом шаге использовать ранее выполненные вычисления.

При медианной фильтрации используется двумерное окно (апертура фильтра), обычно имеющее центральную симметрию, при этом его центр располагается в текущей точке фильтрации. На рис. 1 показаны два примера наиболее часто применяемых вариантов окон в виде креста и в виде квадрата. Размеры апертуры принадлежат к числу параметров, оптимизируемых в процессе анализа эффективности алгоритма. Отсчеты изображения, оказавшиеся в пределах окна, образуют рабочую выборку текущего шага.

а)б)

Рисунок 1 - Примеры окон при медианной фильтрации

Двумерный характер окна позволяет выполнять, по существу, двумерную фильтрацию, поскольку для образования оценки привлекаются данные как из текущих строки и столбца, так и из соседних. Обозначим рабочую выборку в виде одномерного массива ; число его элементов равняется размеру окна, а их расположение произвольно. Обычно применяют окна с нечетным числом точек *n* (это автоматически обеспечивается при центральной симметрии апертуры и при вхождении самой центральной точки в ее состав). Если упорядочить последовательность по возрастанию, то ее медианой будет тот элемент выборки, который занимает центральное положение в этой упорядоченной последовательности. Полученное таким образом число и является продуктом фильтрации для текущей точки кадра. Понятно, что результат такой обработки в самом деле не зависит от того, в какой последовательности представлены элементы изображения в рабочей выборке . Введем формальное обозначение описанной процедуры в виде:

x\*=med(y1, y2,…, yn) (1)

Рассмотрим пример. Предположим, что выборка имеет вид: Y={136,110,99,45,250,55,158,104,75}, а элемент 250, расположенный в ее центре, соответствует текущей точке фильтрации (i1, i2) (рис. 1). Большое значение яркости в этой точке кадра может быть результатом воздействия импульсной (точечной) помехи. Упорядоченная по возрастанию выборка имеет при этом вид {45,55,75,99,104,110,136,158,250}, следовательно, в соответствии с процедурой (1), получаем x\*=med(y1, y2,…, y9)=104. Видим, что влияние “соседей” на результат фильтрации в текущей точке привело к “игнорированию” импульсного выброса яркости, что следует рассматривать как эффект фильтрации. Если импульсная помеха не является точечной, а покрывает некоторую локальную область, то она также может быть подавлена. Это произойдет, если размер этой локальной области будет меньше, чем половина размера апертуры медианного фильтра. Поэтому для подавления импульсных помех, поражающих локальные участки изображения, следует увеличивать размеры апертуры медианного фильтра [1].

Из (1) следует, что действие медианного фильтра состоит в “игнорировании” экстремальных значений входной выборки - как положительных, так и отрицательных выбросов. Такой принцип подавления помехи может быть применен и для ослабления шума на изображении.

При вычислении медианы в окне фильтра число операций с данными, например, число операций сортировки, равно n2. При обработке изображения размером MˣN точек (пикселей) число операций с данными будет велико и составит MˣNˣn2.

## Медианный фильтр в среде MATLAB

Полный листинг программы приведен в приложении 1.

Реализуем чтение файла с исходным изображением, а также наложение шума на изображение:

Vanin = imread('F:\256.jpg');

AVTF = imnoise(Vanin,'salt & pepper',0.5);

Фильтрацию будем осуществлять встроенной командой medfilt2.

Реализуем три прохода фильтрации:

K1 = medfilt2(J,[5,5]);

K2 = medfilt2(K1,[5,5]);

K3 = medfilt2(K2,[5,5]);



Рисунок 2 – Исходное изображение

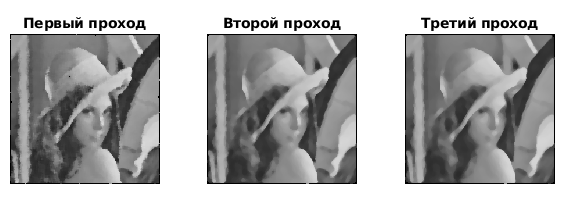


Рисунок 3 – Фильтрация по проходам

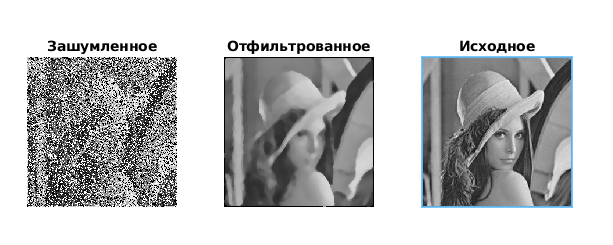


Рисунок 4 – Сравнение результатов

Можно заметить, что изображение восстановлено, но была утеряна детализация изображения, а также, что после первого прохода фильтра на изображении остались артефакты. После третьего прохода фильтра, изображение почти не изменилось.

## Медианный фильтр на языке программирования Python 3.7.

Полный код приведен в приложении 2 и все полученные изображения прилагаются в архиве.

Реализуем функцию добавления шума на изображение:

def sp\_noise(image, prob):

…

return output

Входными параметрами будут изображение и процент зашумления, а выходными зашумленное изображение. Пример работы функции показан на рисунке 5.

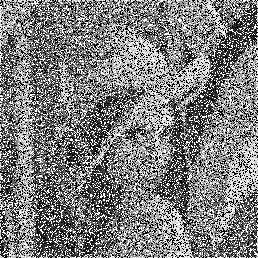


Рисунок 5 – Работа функции зашумления изображения

Теперь реализуем функцию медианной фильтрации:

def median\_filter(data, filter\_size):

…

return data\_final

Входными параметрами будут: зашумленное изображение и размер фильтра. Входным параметров будет отфильтрованное изображение.

В функции реализован перебор всех пикселей и поиск медианного значения.

Пример работы фильтрации представлен на рисунке 6.





Рисунок 6 – Три прохода фильтрации



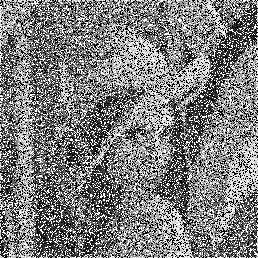


Рисунок 7 – Сравнение результатов

Если сравнивать реализацию в среде MATLAB и на языке Python, то можно сказать, что результаты практически идентичны.

# Вывод

Из результатов, можно сказать, что медианная фильтрация в меньшей степени сглаживает границы изображения. Механизм этого явления очень прост и заключается в следующем. Предположим, что апертура фильтра находится вблизи границы, разделяющей светлый и темный участки изображения, при этом ее центр располагается в области темного участка. Тогда, вероятнее всего, рабочая выборка будет содержать большее количество элементов с малыми значениями яркости, и, следовательно, медиана будет находиться среди тех элементов рабочей выборки, которые соответствуют этой области изображения. Ситуация меняется на противоположную, если центр апертуры смещен в область более высокой яркости. Но это и означает наличие чувствительности у медианного фильтра к перепадам яркости.

При больших размерах маски происходит размытие контуров изображения и, как следствие, снижение четкости изображения. Этот недостаток можно уменьшить до минимума, если воспользоваться медианной фильтрацией с динамическим размером маски (адаптивной медианной фильтрацией).

# Список литературы

1. Медианная фильтрация // [Электронный ресурс]// URL: http://sernam.ru/book\_kir.php?id=25 (дата обращения 02.06.2021)
2. Ярославский Л.П. Цифровая обработка сигналов в оптике и голографии: введение в цифровую оптику. - М.: Радио и связь, 1987. - 296 с.
3. Упакова А.Г. ФИЛЬТРАЦИЯ ИЗОБРАЖЕНИЙ С ИМПУЛЬСНЫМИ ПОМЕХАМИ НА ОСНОВЕ ТЕОРИИ ДИСКРЕТНЫХ СИГНАЛОВ НА КОНЕЧНЫХ ИНТЕРВАЛАХ // Современные проблемы науки и образования. – 2014. – № 2.; URL: [http://www.science-education.ru/ru/article/view?id=12304] (дата обращения: 02.06.2021).
4. Виды шумов Фильтрация изображений. Источники шума – неидеальное оборудование для захвата изображения (ТВ-тюнер, видеокамера, сканер и т.п.); – презентация. Электронный источник. [http://www.myshared.ru/slide/685707/] (дата обращения: 02.06.2021).
5. Введение в цифровую обработку изображений. Ярославский Л.П. М.: Сов. радио, 2007. – 312 с.
6. Цифровая обработка изображений. Книга 2. Прэтт У. цифровая обработка изображений: Пер. с англ. — М.: Мир, 1982. — Кн. 2 — 480 с. Предназначена для специалистов, имеющих дело с обработкой изображений, а также для студентов, изучающих методы цифровой обработки изображений.

# Приложение 1

Vanin = imread('256.jpg');

AVTF = imnoise(Vanin,'salt & pepper',0.5);

K1 = medfilt2(AVTF,[7,7]);

K2 = medfilt2(K1,[7,7]);

K3 = medfilt2(K2,[7,7]);

figure(1)

subplot(1,3,1)

imshow(AVTF), title('Зашумленное');

subplot(1,3,2)

imshow(K3), title('Отфильтрованное');

subplot(1,3,3)

imshow(Vanin), title('Исходное');

figure(2)

subplot(2,3,1)

imshow(Vanin), title('Исходное');

subplot(2,3,2)

imshow(AVTF), title('Зашумленное');

subplot(2,3,4)

imshow(K1), title('Первый проход');

subplot(2,3,5)

imshow(K2), title('Второй проход');

subplot(2,3,6)

imshow(K3), title('Третий проход');

# Приложение 2

import numpy as np

import random

from PIL import Image, ImageFilter

def median\_filter(data, filter\_size):

temp = []

indexer = filter\_size // 2

data\_final = []

data\_final = np.zeros((len(data),len(data[0])))

for i in range(len(data)):

for j in range(len(data[0])):

for z in range(filter\_size):

if i + z - indexer < 0 or i + z - indexer > len(data) - 1:

for c in range(filter\_size):

temp.append(0)

else:

if j + z - indexer < 0 or j + indexer > len(data[0]) - 1:

temp.append(0)

else:

for k in range(filter\_size):

temp.append(data[i + z - indexer][j + k - indexer])

temp.sort()

data\_final[i][j] = temp[len(temp) // 2]

temp = []

return data\_final

def sp\_noise(image,prob):

output = np.zeros(image.shape,np.uint8)

thres = prob / 2

for i in range(image.shape[0]):

for j in range(image.shape[1]):

rdn = random.random()

if rdn < thres:

output[i][j] = 0

elif rdn < prob:

output[i][j] = 255

else:

output[i][j] = image[i][j]

return output

imgStock = Image.open("F:\\256.jpg").convert("L")

arImg = np.array(imgStock)

arImgNoised = sp\_noise(arImg, 0.5);

imgNoised = Image.fromarray(arImgNoised)

imgNoised = imgNoised.convert("L")

imgNoised.save("F:\\noised.jpg")

arImgFiltered1 = median\_filter(arImgNoised, 5)

imgFiltered1 = Image.fromarray(arImgFiltered1)

imgFiltered1=imgFiltered1.convert("L")

imgFiltered1.save("F:\\filtered1.jpg")

arImgFiltered2 = median\_filter(arImgFiltered1, 5)

imgFiltered2 = Image.fromarray(arImgFiltered2)

imgFiltered2 = imgFiltered2.convert("L")

imgFiltered2.save("F:\\filtered2.jpg")

arImgFiltered3 = median\_filter(arImgFiltered2, 5)

imgFiltered3 = Image.fromarray(arImgFiltered3)

imgFiltered3 = imgFiltered3.convert("L")

imgFiltered3.save("F:\\filtered3.jpg")