**БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ**

Лабораторная работа №4

**Случайные процессы в обработке изображений**

Вариант 2

**Выполнили:**

Крученков Евгений Андреевич

Кендысь Алексей Максимович,

студенты 4 курса, 7 группы,

специальность

“прикладная математика”

**Преподаватель:**

Старший преподаватель

кафедры ТВиМС ФПМИ,

Л.А. Хаткевич

Минск, 2023

**Теоретические сведения**

Шум — это бессмысленная информация, добавляемая к данным, которая приводит к их повреждению или искажению. Шум позволяет затруднить распознавание содержимого изображения. В большинстве случаев шумы распространяются случайным образом. Соответственно, шум может обладать статистическими свойствами (распределение, дисперсия и др.). В процессе передачи изображения могут искажаться помехами, возникающими в каналах связи. Т.е. реальные данные часто содержат некоторое количество шума. Соответственно, важно научиться выдерживать шум и изучить его основные закономерности, т.к. модель должна быть способна обобщать зашумлённые обучающие данные.

**Фильтрация изображений**

**Скользящее среднее.** Простейшим видом линейной оконной фильтрации в пространственной области является скользящее среднее в окне. Результатом такой фильтрации является значение математического ожидания, вычисленное по всем пикселям окна. Математически это эквивалентно свертке с маской, все элементы которой равны 1/n, где n– число элементов маски.

**Медианная фильтрация.** Это типичный нелинейный фильтр, представляющий собой технологию нелинейной обработки сигналов, которая может эффективно подавлять шумы на основе статистической теории сортировки. Этот метод может сохранить детали края изображения, устраняя импульсный шум и шум соли и перца. Эти превосходные характеристики недоступны при линейной фильтрации.

**Выполнение задания**

**Подключение библиотек и задание параметров.**

import math

import matplotlib.pyplot as plt

import numpy as np

Скользящий усредняющий фильтр

def SAF(Noiced\_image, N: int):

    size = Noiced\_image.shape

    h = 1/(N\*\*2)

    image = np.ndarray(size, dtype=int)

    for i in range(int(N/2),size[0]-int(N/2)):

        for j in range(int(N/2),size[1]-int(N/2)):

            for k in range(3):

                image[i][j][k] = 0

                for I in range(-int(N/2),int(N/2)+1):

                    for J in range(-int(N/2),int(N/2)+1):

                        image[i][j][k]+= h\*Noiced\_image[i+I][j+J][k]

    return image

Медианный фильтр

def MF(Noiced\_image, N: int):

    size = Noiced\_image.shape

    h = 1/(N\*\*2)

    image = np.ndarray(size, dtype=int)

    for i in range(int(N/2),size[0]-int(N/2)):

        for j in range(int(N/2),size[1]-int(N/2)):

            for k in range(3):

                mas = [0]\*(N\*\*2)

                l = 0

                for I in range(-int(N/2),int(N/2)+1):

                    for J in range(-int(N/2),int(N/2)+1):

                        mas[l] = Noiced\_image[i+I][j+J][k]

                        l+=1

                image[i][j][k] = st.median(mas)

    return image

Применение усредняющего фильтра

fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_noised3, title="Изображение с шумом 3", cmap="gray")

image\_3\_SAF3 = SAF(image\_noised3,3)

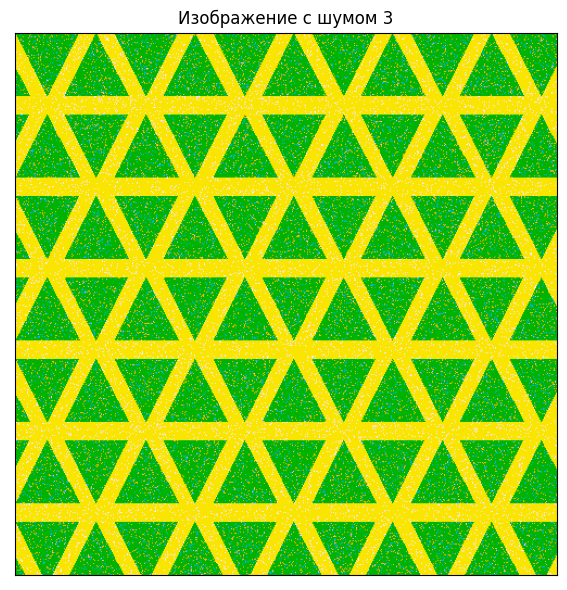
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

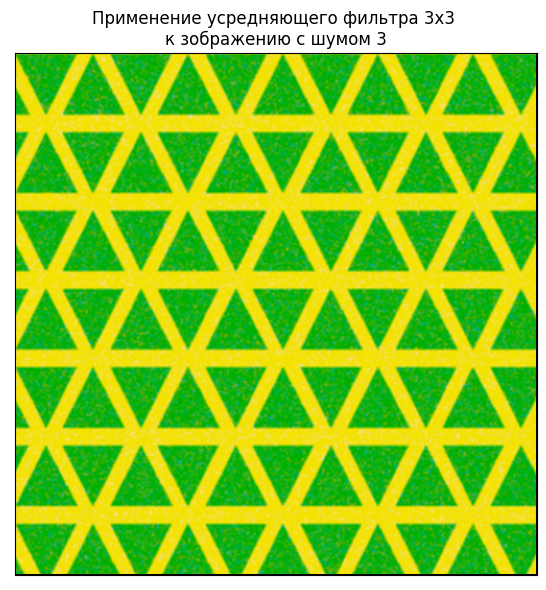
show\_image(ax, image\_3\_SAF3, title="Применение усредняющего фильтра 3х3 \nк зображению с шумом 3", cmap="gray")

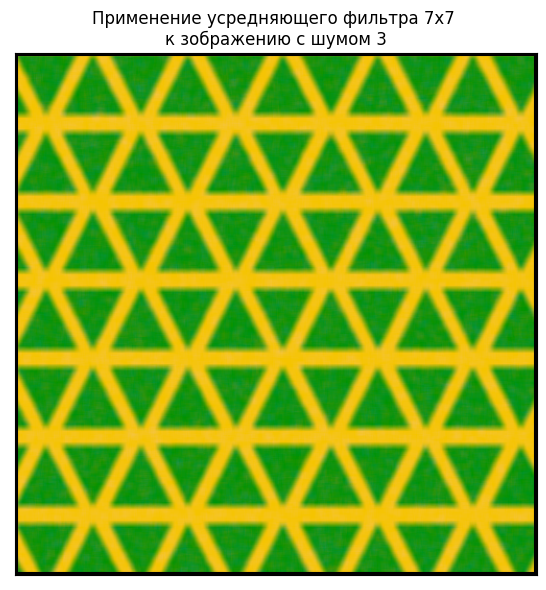
image\_3\_SAF7 = SAF(image\_noised3,7)

fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_3\_SAF7, title="Применение усредняющего фильтра 7х7 \nк зображению с шумом 3", cmap="gray")







Применение медианного фильтра

fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_noised4, title="Изображение с шумом 4", cmap="gray")

image\_3\_MF3 = MF(image\_noised4,3)

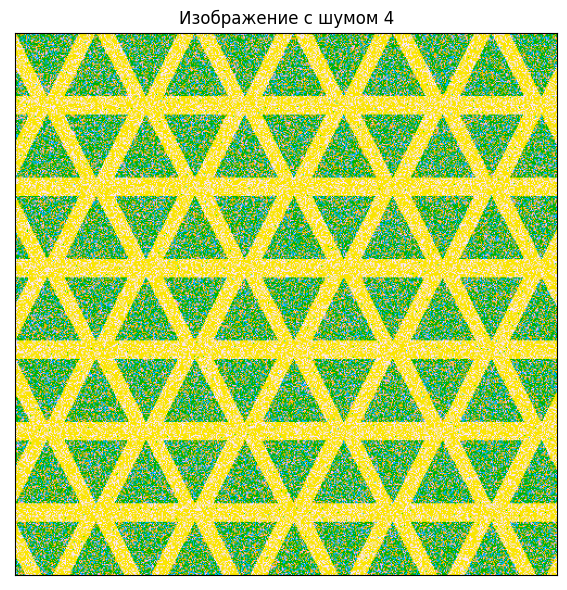
fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

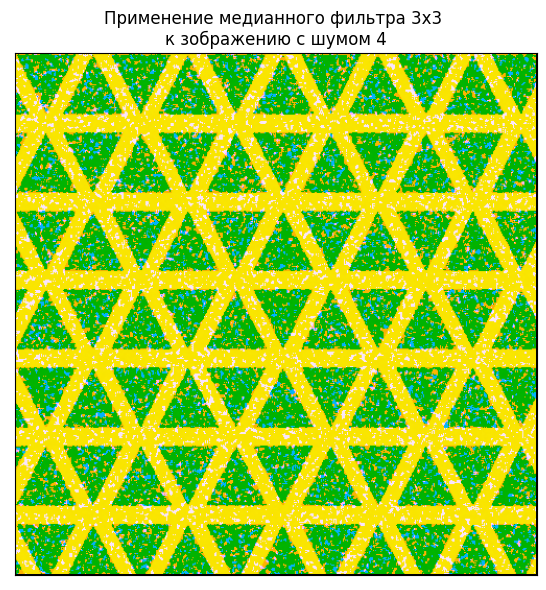
show\_image(ax, image\_3\_MF3, title="Применение медианного фильтра 3х3 \nк зображению с шумом 4", cmap="gray")

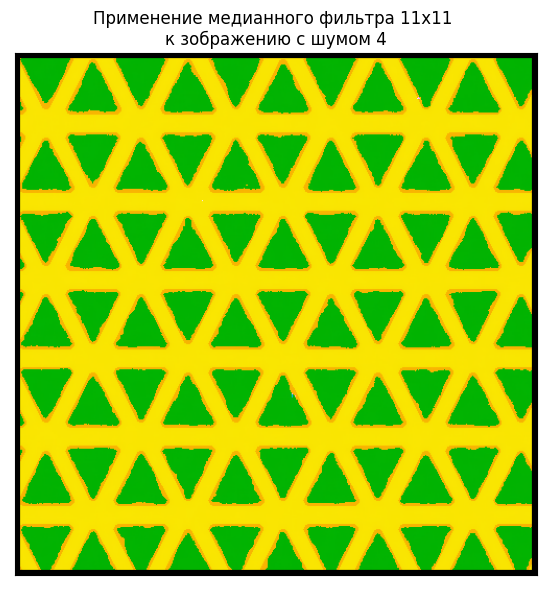
image\_3\_MF7 = MF(image\_noised4,11)

fig, ax = plt.subplots(figsize=(6, 6), layout="tight")

show\_image(ax, image\_3\_MF7, title="Применение медианного фильтра 11х11 \nк зображению с шумом 4", cmap="gray")







Приближенный фрагмент изображения:



**Выводы**

Что касается шума, то можно заметить, что с увеличением дисперсии растёт и зашумлённость исходного изображения. Так, с дисперсией 0.1 шум едва заметен, а с ростом дисперсии он начинает проявляться всё более явно, при дисперсии 5 изображение заметно портится. Это имеет смысл, т.к. при увеличении дисперсии растёт разброс значений в гауссовском распределении. Также стоит отметить, что шум получился цветным, т.к. генерировался для всех трёх цветовых каналов.

Усредняющий фильтр является линейным фильтром и одним из самых простых для понимания фильтров. Результатом применения этого фильтра является более размытое, нежели исходное, изображение. Рассмотрим полученные результаты. При фильтрации изображения фильтром 3х3 с дисперсией 0.5 этот фильтр довольно хорошо показал себя: края изображения размыты не слишком сильно, в то время как шум на изображении практически не заметен. Для фильтра 7х7 зашумленности изображения нет, но размытие слишком сильное. Можно сделать вывод о том, что при увеличении маски увеличивается размытие изображения, что очевидно. Следовательно, не имеет смысла использовать шаблон слишком большого размера.

Медианный фильтр хорошо показал себя при обработке изображений с сильной зашумленностью (было рассмотрено изображение с дисперсией 5). При фильтрации фильтром 11х11 мы наблюдаем полное отсутствие шума, хоть он и был очень сильным, и некоторые неровности краёв изображения и присутствие другого цвета на границе, что всё же нельзя назвать размытием в прямом смысле данного слова, что является очевидным. Полученные результаты подтверждают целесообразность применения этого фильтра при устранении шума.