МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 44

ОТЧЕТ

ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

доцент		А. М. Сергеев
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №1

РАЗРАБОТКА WINDOWS-ПРИЛОЖЕНИЯ

по курсу: ЦИФРОВАЯ ОБРАБОТКА ИЗОБРАЖЕНИЙ

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4941		В. С Польщиков
	подпись, дата	инициалы, фамилия

Санкт-Петербург
2023

1. **Цель работы:** изучить алгоритм работы и особенности заданного метода и разработать windows-приложения для демонстрации его применения.
2. **Задание**

Вариант 20: Среднегармонический и контргармонический фильтры.

1. Изучить алгоритм работы и особенности заданного метода (методов) обработки изображения в соответствии с полученным вариантом задания.
2. Вычислить результаты обработки типичных фрагментов (импульсы, перепады яркости и т.п.).
3. Разработать алгоритм обработки растрового изображения заданным методом с учетом необходимости решения возможных проблем.
4. Разработать блок-схему, реализующую данный метод (методы), в форме Windows-приложения. Интерфейс приложения должен обеспечивать:
   - ввод монохромного (по желанию студента – полноцветного) растрового исходного изображения формата bmp и его отображение на дисплее в режиме «один пиксель изображения – один пиксель экрана»;
   - возможность настройки параметров обработки (размеры апертуры и тип фильтра, коэффициенты усиления и т.п.);
   - отображение результата обработки на экране дисплея и его сохранение в заданном пользователем файле;
   - отображение на дисплее координат и яркости любого выбранного пользователем пикселя исходного изображения и яркости соответствующего ему пикселя обработанного изображения, а также дополнительные возможности для оценки результата работы, например, гистограммы яркости изображений и/или разрезы функций яркости изображений по строкам (столбцам).
5. Разработать исполняемые модули Windows-приложения, обеспечивающие максимально эффективную программную реализацию обработки заданным методом (методами) исходного изображения.
6. Написать и отладить программу, реализующую разработанное приложение, в среде визуального программирования на одном из объектно-ориентированных языков высокого уровня по выбору студента.

3. **Теория**

Обработка изображения сравнительно редко встречающимся в литературе среднегармоническим фильтром описывается выражением

$$z'(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{z(s,t)}}. \qquad (1.5)$$

Среднегармонический фильтр хорошо выполняет подавление гауссова шума и униполярного импульсного шума в виде белых точек на изображении, но не работает в случае униполярного импульсного шума в виде черных точек. Среднегармонический фильтр, как и среднегеометрический, относится к нелинейным фильтрам, т.к. результат его работы нельзя представить в виде свертки (1.2).

Обработка изображения контргармоническим фильтром описывается выражением

$$z'(x,y) = \frac{\sum_{(s,t)\in S_{xy}} z(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} z(s,t)^{Q}}, \quad (1.6)$$

где Q называется порядком фильтра. Этот фильтр хорошо приспособлен для уменьшения или почти полного устранения импульсного шума, причем при положительных значениях Q фильтр устраняет черную часть импульсного шума, а при отрицательных значениях Q – белую, но обе части шума не могут быть устранены одновременно. При Q = 0 фильтр превращается в среднеарифметический, а при Q = -1 – в среднегармонический.

4. Ход выполнения работы

Приложение разрабатывается на языке программирования Python 3.10, для реализации графического приложения используется библиотека Tkinter.

Так-как учебным пособием рекомендуется не использовать уже готовые методы реализцаии способов обработки изображения, то основные функции реализованы с помощью матриц и математики.

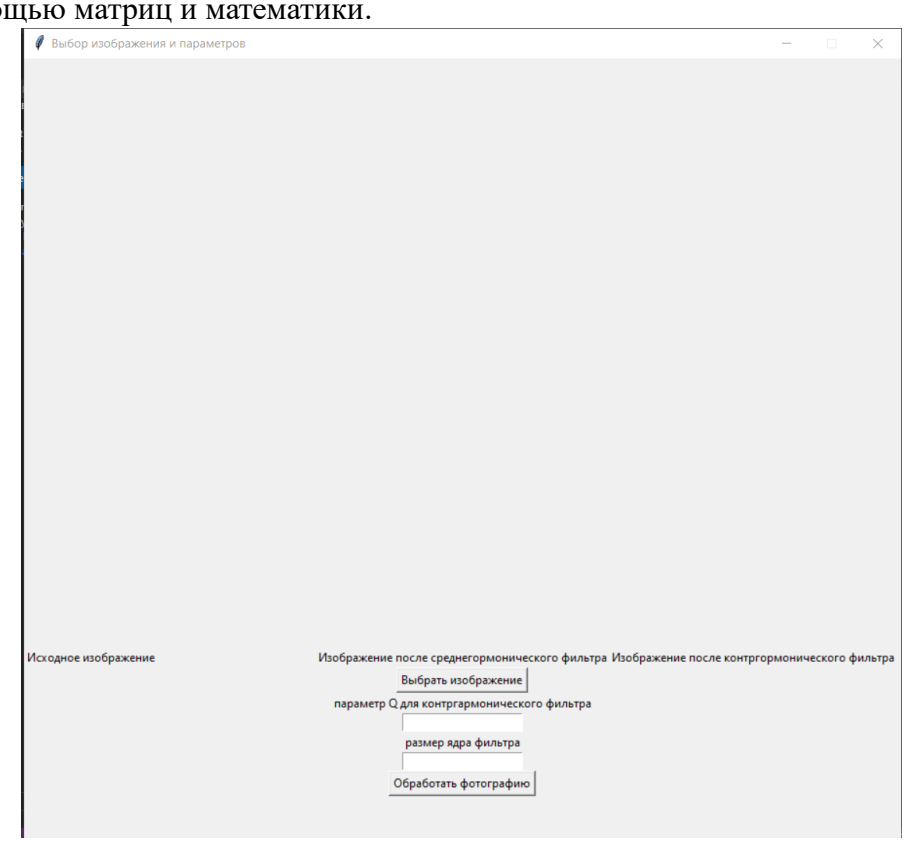

Рисунок 1- Интерфейс приложения

В приложении можно выбрать изображение и задать параметр Q для контргармонического фильтра, а также размер ядра фильтра.

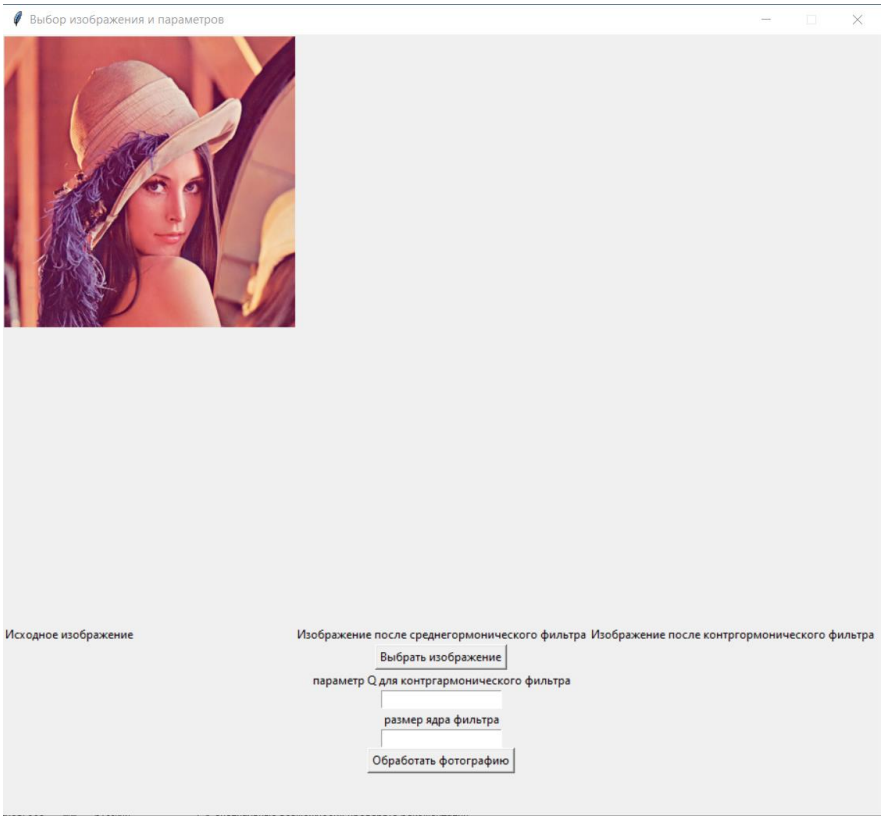

Рисунок 2 – Выбор изображения

После выбора изображения фотография сразу отображается в интерфейсе. После нажатия на кнопку «обработать фотографию». Появляются 2 обработанные фотографии(сначала применяет функция для преобразования в чб изображение, а после применяются фильтры), а также гистограмма распределения яркости изображения.

Рисунок 3 – Обработка изображения

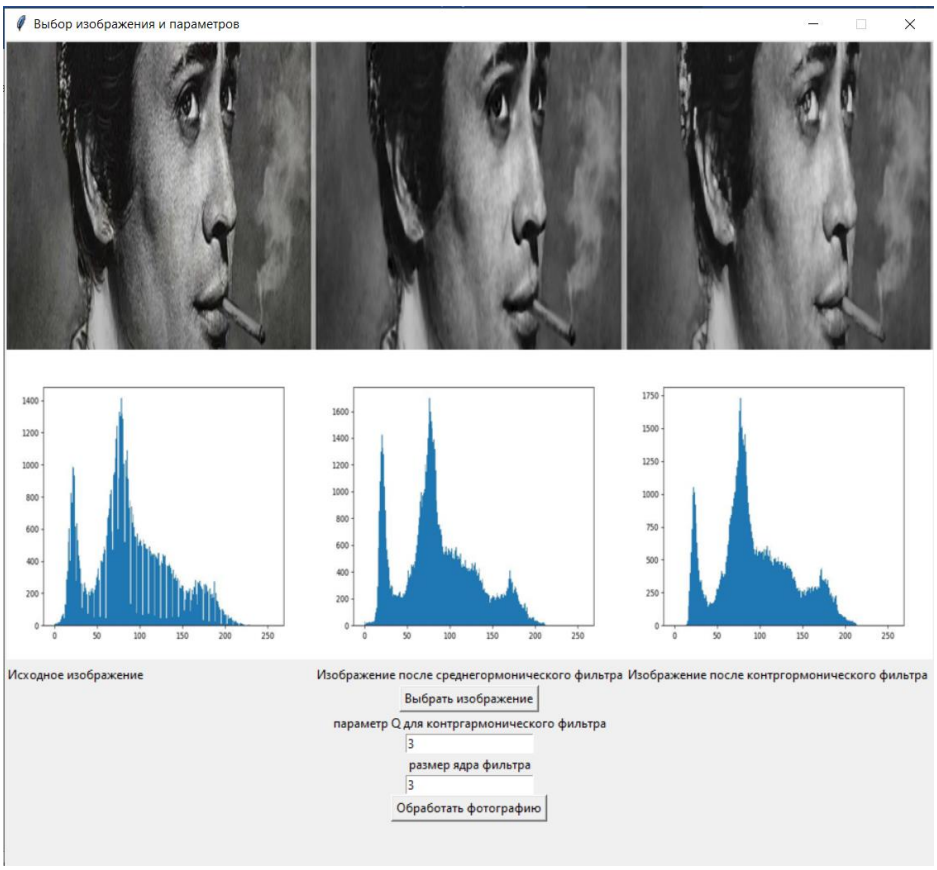

Рисунок 4 – Пример обработки изображения

5. Вывод

В результате выполнения работы изучил алгоритм работы и особенности заданного метода и разработать windows-приложения для демонстрации его применения.

Приложение А.

```
from tkinter import *
from tkinter import filedialog
from tkinter import messagebox
from PIL import Image, ImageTk
import cv2
import numpy as np

import matplotlib.pyplot as plt

# Создаем окно приложения
root = Tk()
root.title("Выбор изображения и параметров")
root.geometry("900x800")
root.resizable(False,False)

# Функция для выбора изображения
def choose_image():
    global img_path,  img_preview
    img_path = filedialog.askopenfilename(title="Выберите изображение", filetypes=[("JPEG
files", "*.jpg")])
    img = Image.open(img_path)
    img = img.resize((300, 300))
    img_preview = ImageTk.PhotoImage(img)
    canvas.create_image(0, 0, anchor=NW, image=img_preview)

# Функция для вывода изображений
def image_choose():
    global img1_path,img1_preview
    global img2_path,img2_preview
    img1_path = "C:\\Users\\Whatislav\\Desktop\\mean_harmonic_filtered.jpg"
    img1 = Image.open(img1_path)
    img1 = img1.resize((300, 300))
    img1_preview = ImageTk.PhotoImage(img1)
    img2_path = "C:\\Users\\Whatislav\\Desktop\\contraharmonic_filtered.jpg"
    img2 = Image.open(img2_path)
    img2 = img2.resize((300, 300))
    img2_preview = ImageTk.PhotoImage(img2)
    canvas.create_image(300, 0, anchor=NW, image=img1_preview)
    canvas.create_image(600, 0, anchor=NW, image=img2_preview)
    hist(img1, 1)
    hist(img2, 2)

# Функция для гармонического фильтра
def mean_harmonic_filter(img, ksize):
    img_arr = np.array(img)
    img_float = img_arr.astype(np.float32)
    epsilon = 1e-8
    inverted_img = 1 / (img_float + epsilon)
    mean_harmonic = cv2.blur(inverted_img, ksize)
```

```
    filtered_img = 1 / (mean_harmonic + epsilon)
    return filtered_img.clip(0, 255).astype(np.uint8)

# Функция для контргармонического фильтра
def contraharmonic_filter(img, ksize, Q):
    img_arr = np.array(img)
    img_float = img_arr.astype(np.float32)
    epsilon = 1e-8
    numerator = np.power(img_float + epsilon, Q + 1)
    denominator = np.power(img_float + epsilon, Q)
    numerator_blur = cv2.blur(numerator, ksize)
    denominator_blur = cv2.blur(denominator, ksize)
    filtered_img = numerator_blur / (denominator_blur + epsilon)
    return filtered_img.clip(0, 255).astype(np.uint8)

# Функция для построения гистограмм
def hist(img, i):
    gray_img = np.array(img)
    plt.hist(gray_img.ravel(), bins=256, range=(0, 256))
    hist="histogram"+str(i)+".jpg"
    plt.savefig(hist)
    plt.close()

# Функция для вывода гистограммы
def hist_view():
    global hist_path,hist_preview,hist_preview2,hist_preview3
    hist_path = "histogram0.jpg"
    hist= Image.open(hist_path)
    hist = hist.resize((300, 300))
    hist_preview = ImageTk.PhotoImage(hist)
    canvas.create_image(0, 300, anchor=NW, image=hist_preview)
    hist_path = "histogram1.jpg"
    hist= Image.open(hist_path)
    hist = hist.resize((300, 300))
    hist_preview2 = ImageTk.PhotoImage(hist)
    canvas.create_image(300, 300, anchor=NW, image=hist_preview2)
    hist_path = "histogram2.jpg"
    hist= Image.open(hist_path)
    hist = hist.resize((300, 300))
    hist_preview3 = ImageTk.PhotoImage(hist)
    canvas.create_image(600, 300, anchor=NW, image=hist_preview3)

# Функция для обработки
def click_button():
    global Q_entry, size_entry
    try:
        Q =float(Q_entry.get())
        size = int(size_entry.get())
        input_image_path = img_path.replace("/","\\")
        output_mean_harmonic_path =
'C:\\Users\\Whatislav\\Desktop\\mean_harmonic_filtered.jpg'
```

```
        output_contraharmonic_path =
'C:\\Users\\Whatislav\\Desktop\\contraharmonic_filtered.jpg'
        img1 = Image.open(input_image_path)
        img1= img1.convert('L')
        ksize = (size, size)
        mean_harmonic_filtered_img = mean_harmonic_filter(img1, ksize)
        cv2.imwrite(output_mean_harmonic_path, mean_harmonic_filtered_img)
        contraharmonic_filtered_img = contraharmonic_filter(img1, ksize, Q)
        cv2.imwrite(output_contraharmonic_path, contraharmonic_filtered_img)
        hist(img1, 0)
        image_choose()
        hist_view()
    except ValueError:
        messagebox.showerror("Ошибка", "Параметры должны быть числами")

def validate_input(input_str):
    if input_str.isdigit():
        return True
    else:
        return False

# Создаем элементы интерфейса
canvas = Canvas(root, width=900, height=600)
canvas.pack()
label = Label(root, text="Исходное изображение")
label.place(x=0, y=604)
label = Label(root, text="Изображение после среднегормонического фильтра")
label.pack()
label = Label(root, text="Изображение после контргормонического фильтра")
label.place(x=600, y=604)
choose_image_button = Button(root, text="Выбрать изображение", command=choose_image)
choose_image_button.pack()
Q_label = Label(root, text="параметр Q для контргармонического фильтра")
Q_label.pack()
Q_entry = Entry(root)
Q_entry.pack()
size_label = Label(root, text="размер ядра фильтра")
size_label.pack()
size_entry = Entry(root)
size_entry.pack()
process_button = Button(root, text="Обработать фотографию", command=click_button)
process_button.pack()
# Запускаем главный цикл приложения
root.mainloop()
```