

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ  
РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное  
образовательное учреждение высшего образования  
«Самарский национальный исследовательский университет  
имени академика С.П. Королева»  
(Самарский университет)

Институт информатики и электроники  
Кафедра технической кибернетики

**Отчет по лабораторной работе №2**

Дисциплина: «Системы обработки изображений»  
Вариант №3

Выполнил: Новицкий Н. В.  
Группа: 6132-010402D

Самара 2022

## **Задание на лабораторную работу**

1. Считать цветное RGB изображение
2. Зашумить изображение аддитивным шумом с вероятностью  $p$  (по вариантам).
3. Написать функцию реализации ранговой фильтрации
4. Отфильтровать зашумленное изображение со всеми возможными рангами (количество рангов зависит от окна по вариантам). Подсчитать СКО для результата фильтрации с каждым рангом.
5. Написать функцию реализации свертки
6. Отфильтровать изображение КИХ фильтром с ядром заданным по вариантам. Подсчитать СКО. Сравнить с результатами пункта 4.

Задание 1. Считать цветное RGB изображение.

Код для считывания RGB изображения можно найти в Jupyter Notebook с выполненной лабораторной работой, ссылка будет указана в конце отчета.

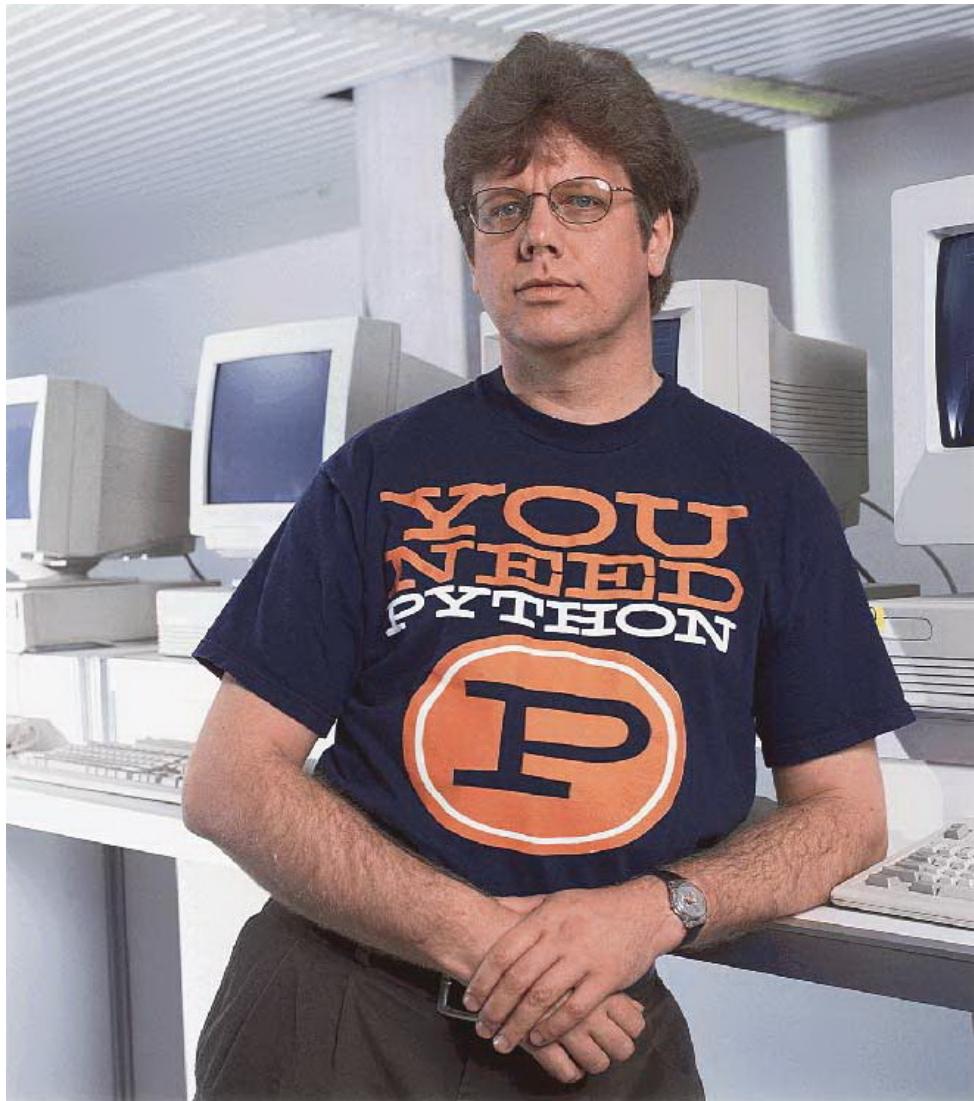


Рисунок 1. Исходное RGB изображение

Задание 2. Зашумить изображение аддитивным шумом с вероятностью  $p$  (по вариантам).

Вероятность  $p$ , которая задана в варианте равняется 0,11.

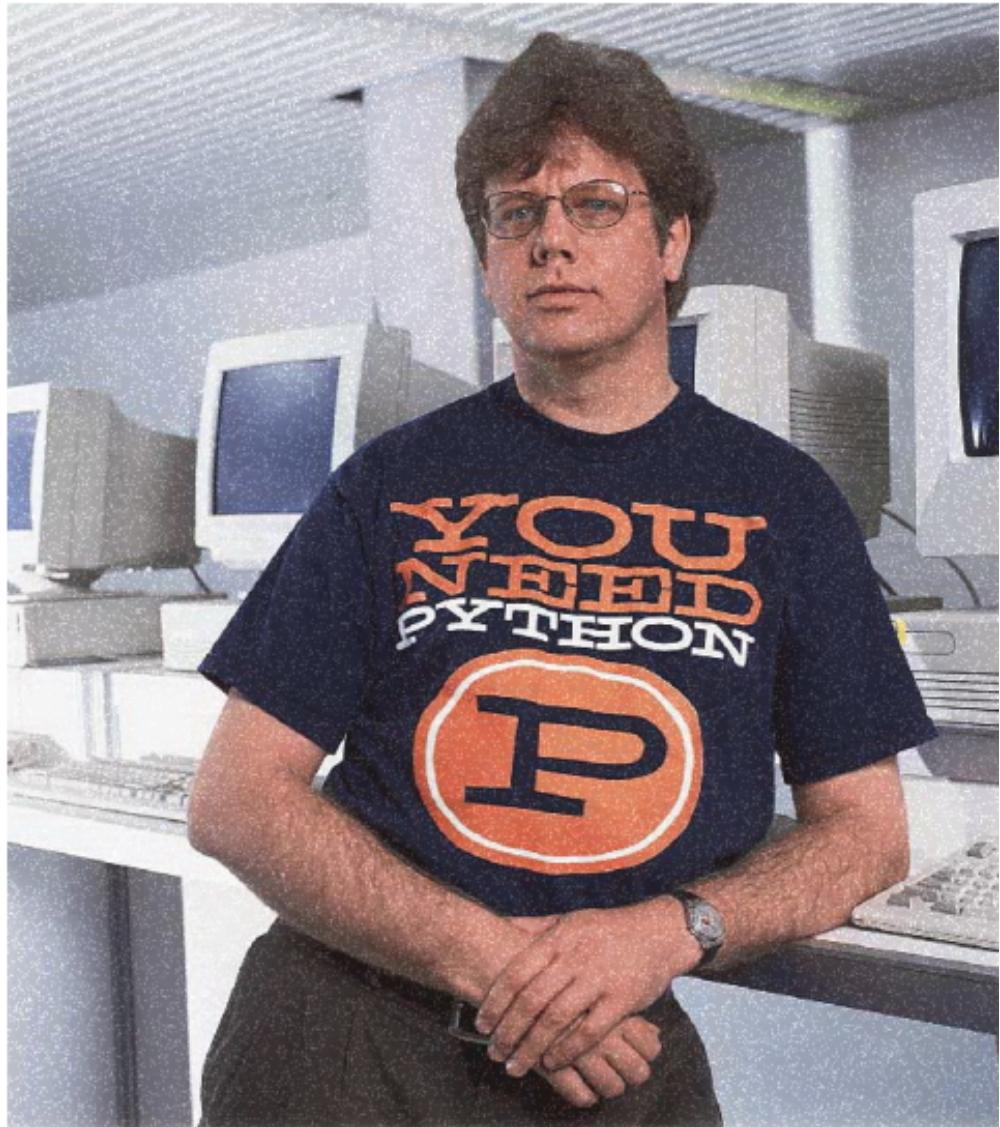


Рисунок 2. Зашумленное изображение с вероятностью  $p$

Задание 3. Написать функцию реализации ранговой фильтрации.

Ранговой фильтрацией называют преобразование набора отсчётов сигнала в вариационный ряд, в котором отсчёты расставляются по возрастанию значений амплитуды (яркости пикселей) и из элементов которого определённым способом формируется результат обработки.

Таким образом, процесс ранговой обработки состоит из следующих этапов:

- Формирование выборки значений по опорной области;
- Построение вариационного ряда из элементов выборки;
- Выбор элементов вариационного ряда и формирование результата обработки;
- Выполнение дополнительных операций для приведения обработки к заданному виду (например, обнуление отрицательных значений).

Алгоритм ранговой фильтрации в общем виде:

$$y_0(n_1, \dots, n_N) = Fx_k(n_1, \dots, n_N), k \in [0; K],$$

где  $(n_1, \dots, n_N)$  – обработка по N степеням свободы;

$x_k(n_1, \dots, n_N)$  – пиксели локальной окрестности входного изображения;

$y_0(n_1, \dots, n_N)$  – текущий пиксель входного изображения;

$F$ - функциональное представление третьего и четвертого этапов обработки;

$K$  – количество ненулевых элементов маски рангового фильтра.

Реализация ранговой фильтрации средствами языка Python:

```
def get_value(image, kernel, rank):
    """
    Вспомогательная функция для получения значения
    """
    image_rows, image_cols, image_channels = image.shape
    output_image = np.zeros((image_channels,))

    for image_channel in range(image_channels):
        vals = []
        for i in range(image_rows):
            for j in range(image_cols):
                for _ in range(kernel[i, j]):
                    vals.append(image[i, j, image_channel])

        output_image[image_channel] = np.sort(vals, axis=0)[rank]

    return output_image

def rank_filtering(image, kernel, rank):
    """
    Функция ранговой фильтрации
    """
    image_rows, image_cols, image_channels = image.shape
    kernel_cols, kernel_rows = kernel.shape[0] // 2, kernel.shape[1] // 2
    output_image = np.zeros((image_rows, image_cols, image_channels),
                           dtype=np.uint8)

    for i in tqdm(range(kernel_rows, image_rows - kernel_rows)):
        for j in range(kernel_cols, image_cols - kernel_cols):
            current_window = image[
                i - kernel_rows: i + kernel_rows + 1,
                j - kernel_cols: j + kernel_cols + 1,
```

```
]  
output_image[i, j] = get_value(current_window, kernel, rank)  
  
clear_output()  
return output_image
```

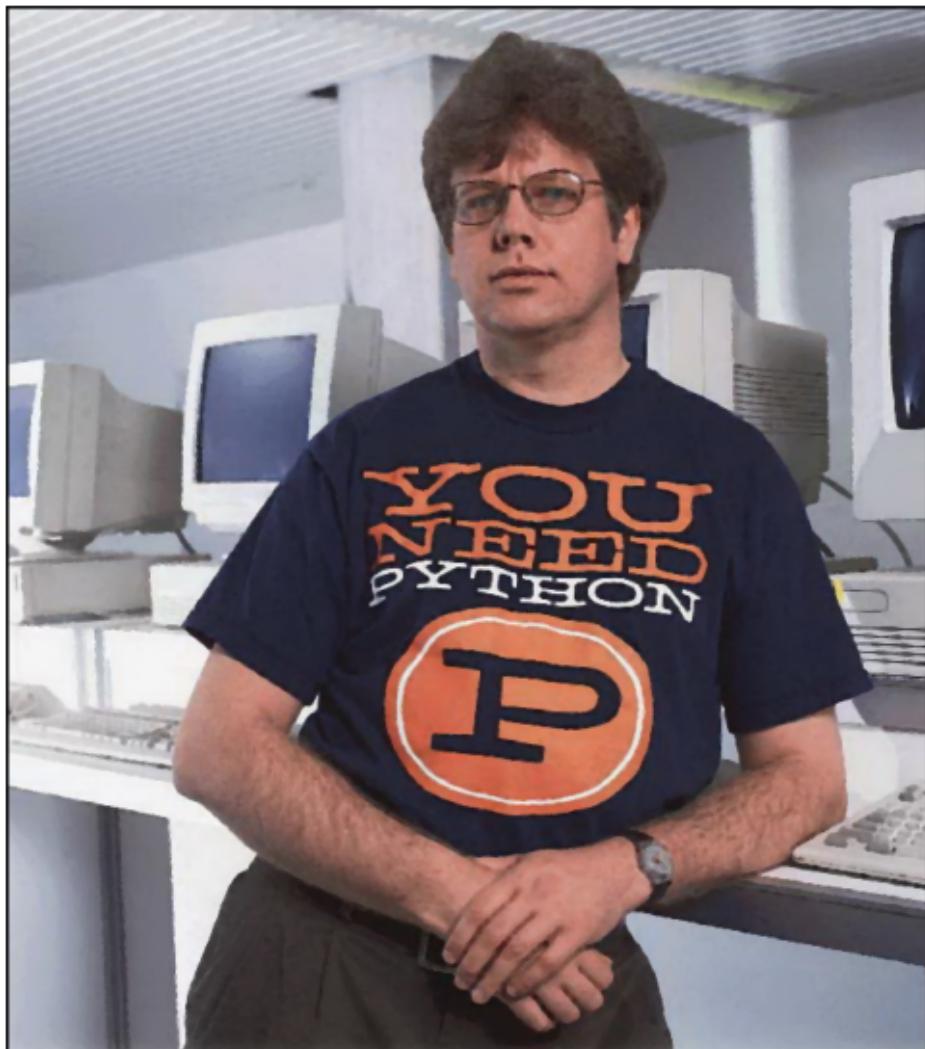


Рисунок 3. Отфильтрованное изображение с рангом 3.

Задание 4. Отфильтровать зашумленное изображение со всеми возможными рангами (кол-во рангов зависит от окна по вариантам). Подсчитать СКО для результата фильтрации с каждым рангом.

СКО (Средняя квадратичная ошибка) определяется следующей формулой:

$$MSE = \frac{1}{N} \sum_{i=1}^N (y_i - \hat{y}_i)^2$$

Окно для варианта 3:

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Далее представлена таблица фильтрации изображения со всеми существующими рангами для данного окна с подсчитанной СКО (сортировка по СКО):

Ранг	СКО
5	21.3519
6	21.8889
4	23.7476
7	26.3779
3	28.9383
8	32.7138
2	35.2295
9	42.0967
1	43.0200
10	57.3225
11	81.5291
12	108.9670

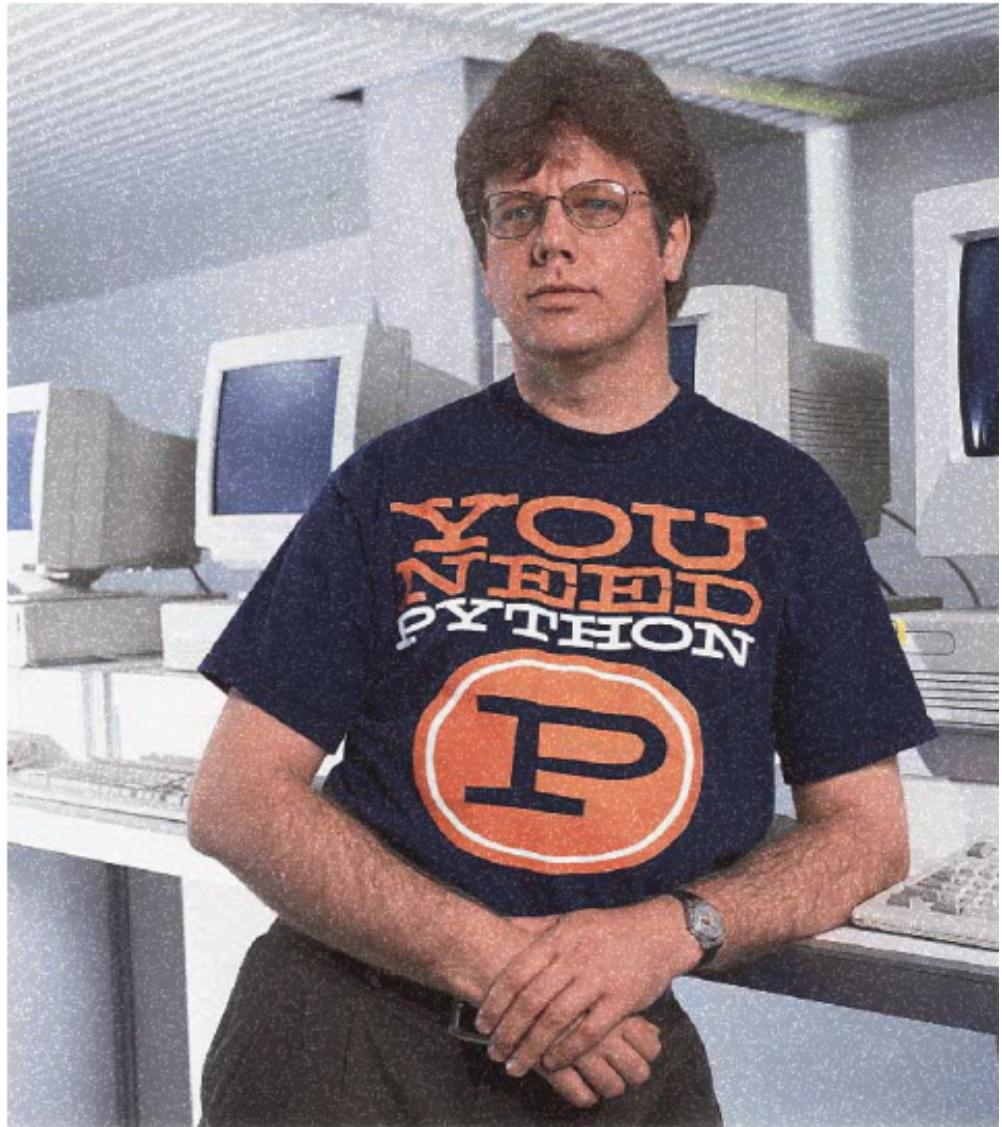


Рисунок 4. Зашумленное изображение с вероятностью  $p$

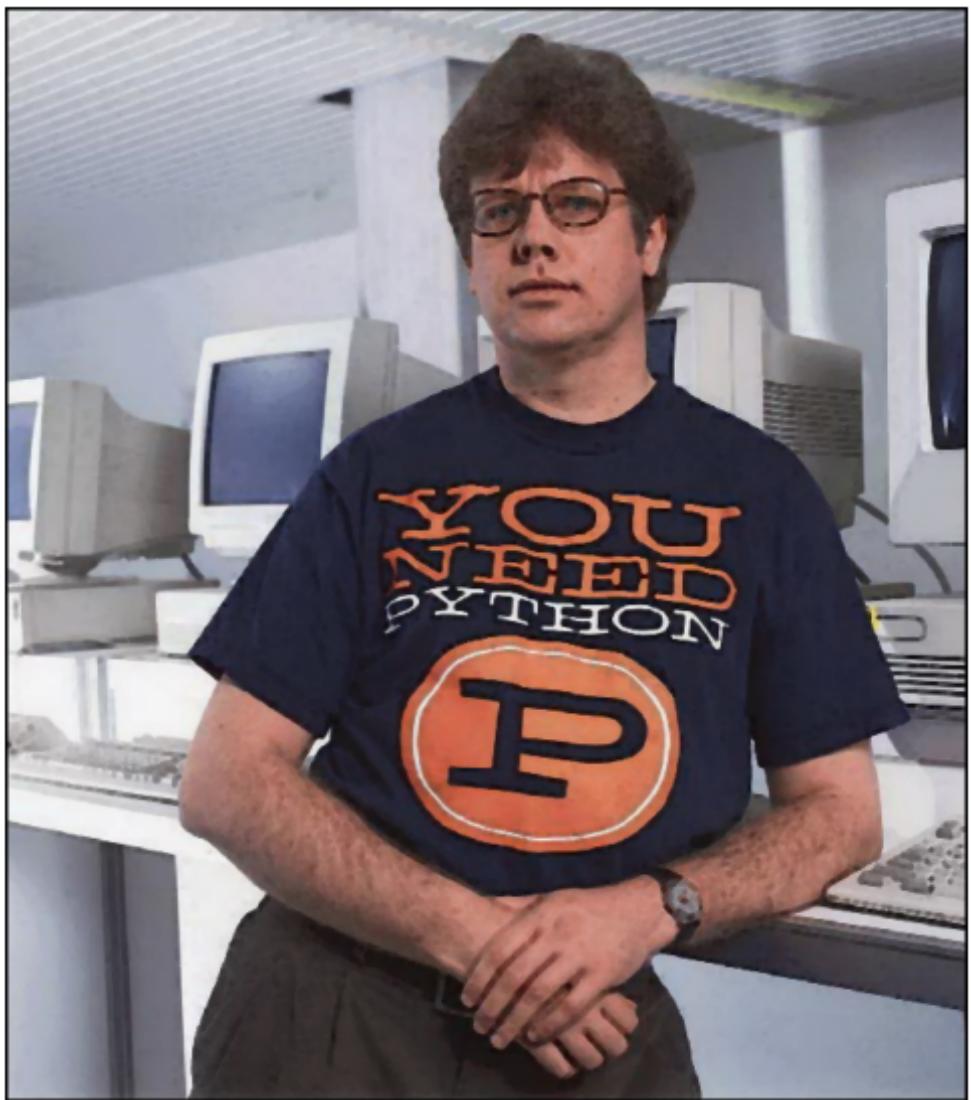


Рисунок 5. Отфильтрованное изображение с рангом 1

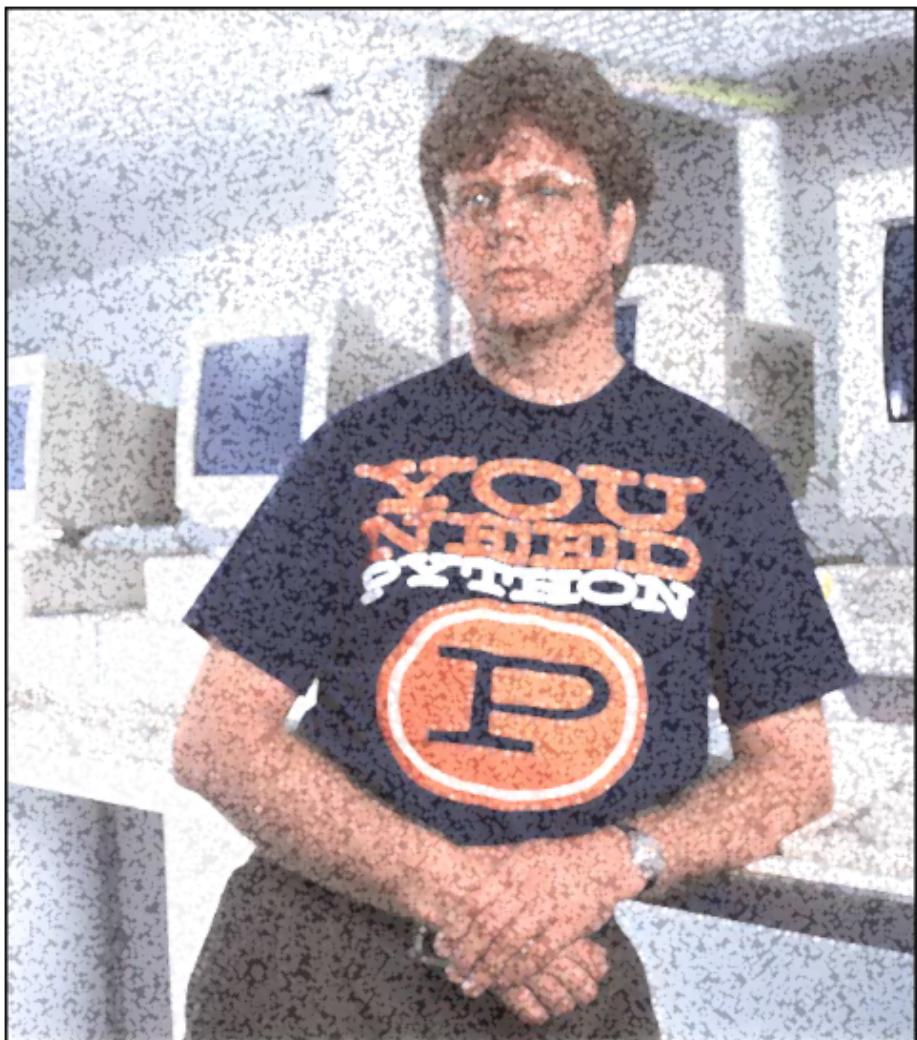


Рисунок 6. Отфильтрованное изображение с рангом 12.

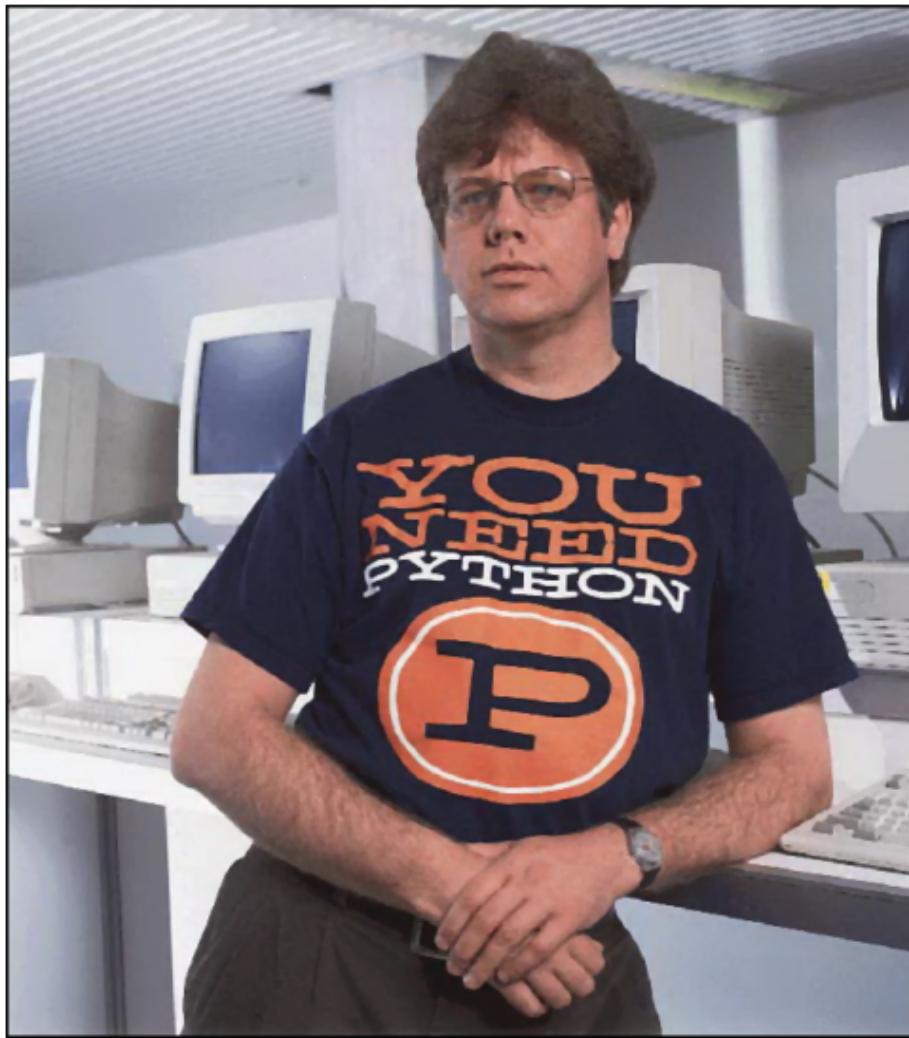


Рисунок 7. Отфильтрованное изображение с рангом 5 (лучший результат!)

Задание 5. Написать функцию реализации свертки.

Свертка - это операция вычисления нового значения выбранного пикселя, учитывающая значения окружающих его пикселей. Для вычисления значения используется матрица, называемая *ядром свертки*. Обычно ядро свертки является квадратной матрицей  $n \times n$ , где  $n$  — нечетное.

Реализованная функция свертки на языке Python:

```
def convolution(image, kernel):
    """
    Функция свертки
    """
    kernel = np.flipud(np.fliplr(kernel))
    image_rows, image_cols, image_channels = image.shape
    kernel_rows, kernel_cols = kernel.shape
    output_image = np.zeros_like(image)

    if kernel_rows % 2 == 0:
```

```

output_image_rows = image_rows + (kernel_rows // 2)
copy_rows = (0, image_rows)
else:
    output_image_rows = image_rows + 2 * (kernel_rows // 2)
    copy_rows = (kernel_rows // 2, output_image_rows - (kernel_rows // 2))

if kernel_cols % 2 == 0:
    output_image_cols = image_cols + (kernel_cols // 2)
    copy_cols = (0, image_cols)
else:
    output_image_cols = image_cols + 2 * (kernel_cols // 2)
    copy_cols = (kernel_cols // 2, output_image_cols - (kernel_cols // 2))

img = np.zeros((output_image_rows, output_image_cols, image_channels))
img[copy_rows[0]: copy_rows[1], copy_rows[0]: copy_cols[1], :] = image

for i, j, k in tqdm(np.ndindex((image_rows, image_cols, image_channels))):
    output_image[i, j, k] = np.sum(
        kernel * img[i: i + kernel_rows, j: j + kernel_cols, k]
    )

clear_output()
return output_image

```

Задание 6. Отфильтровать изображение КИХ фильтром с ядром заданным по вариантам. Подсчитать СКО. Сравнить с результатами пункта 4.

КИХ фильтр заданный вариантом:

$$\frac{1}{13} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 1 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

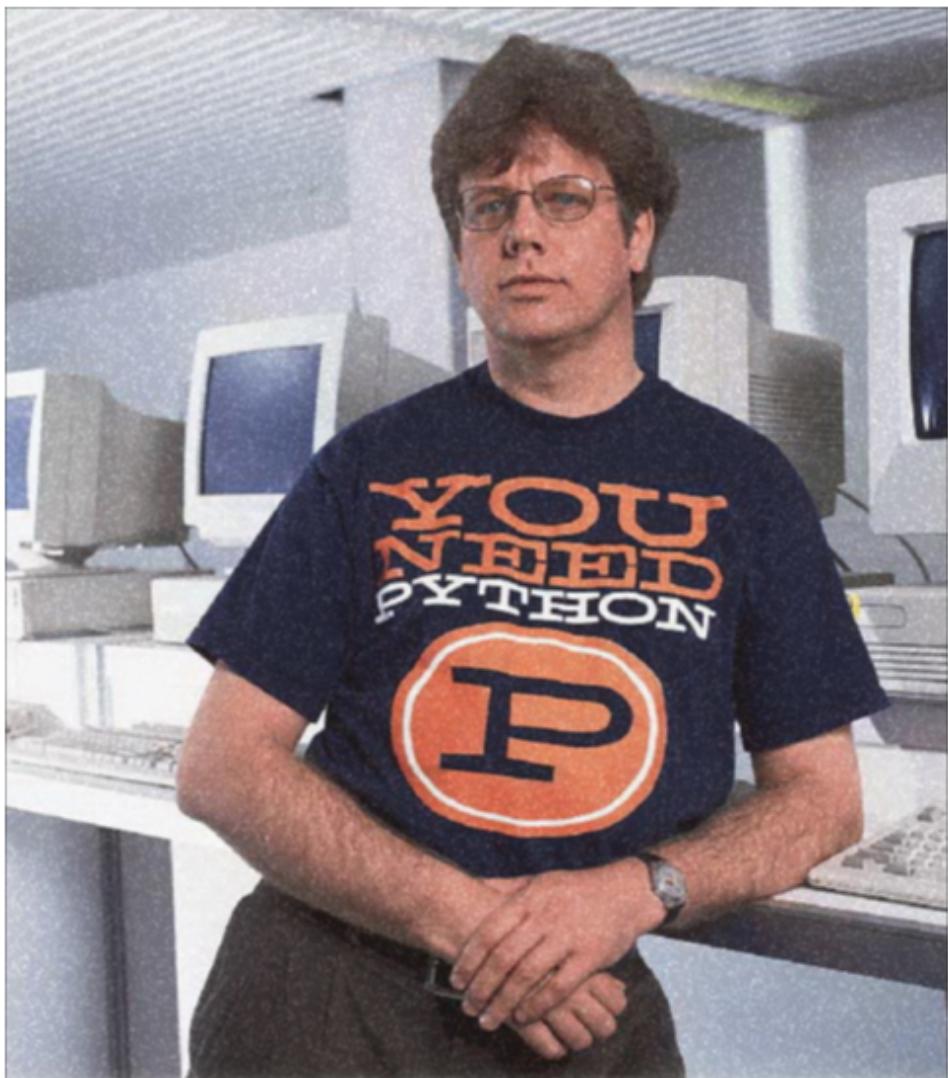


Рисунок 8. Отфильтрованное изображение КИХ фильтром

СКО для отфильтрованного изображения с КИХ фильтром составило: 49.6034. Исходя из этого можно сказать, что ранговая фильтрация с рангом 5 справилась с очисткой изображения намного лучше, нежели фильтрация с КИХ фильтром.

## **Заключение**

В данной лабораторной работе был рассмотрен аддитивный шум, накладываемый на изображение, после была осуществлена фильтрация с помощью ранговой фильтрацией и различными рангами окна, была посчитана СКО для каждого ранга и в конце изображение было отфильтровано КИХ фильтром, который показал себя хуже, чем лучший ранг (равен 5) ранговой фильтрации.

Исходный код можно найти в репозитории GitHub автора лабораторной работы: [ссылка](#)