Министерство образования Республики Беларусь
Учреждение образования
«Брестский государственный технический университет»
Кафедра ИИТ

ОТЧЕТ
Лабораторная работа №1

Тема: «Фильтрация изображений»

Выполнил:
Студент 4 курса
Группа АС-50
Сабо Е. О.
Проверила:
Давидюк Ю.И.

Брест 2020

Лабораторная работа №1

Вариант – 38

Цель работы: Изучить и применить на практике методы фильтрации изображений от импульсных помех.

Задание:

1. Изучить алгоритмы фильтрации изображений от импульсных помех.
2. В программе необходимо добавить возможность импульсного зашумления открытого пользователем изображения (количество шума на изображении должно регулироваться).
3. Запрограммировать метод фильтрации согласно варианту, обеспечить возможность просмотра и сохранения отфильтрованного изображения.

Вариант задания:

38.	Двумерный медианный фильтр с квадратным окном 3x3

Алгоритм зашумления:

```
void AddNoise(double probability)
{
    int size = width*height;
    int count = (int)(size*probability)/100;
    int x,y;
    long pos;
    for (int i = 0; i < count; i++)
    {
        x = rand()%width;
        y = rand()%height;
        pos = y*width+x;
        src_image[pos].blue = rand()%256;// = 255 -
white noise
        src_image[pos].green = rand() % 256;// = 255 -
white noise
        src_image[pos].red = rand() % 256;// = 255 -
white noise
    }
    cout<<"Point was added: "<<count<<endl;
}
```

Алгоритм фильтрации изображения:

```
//Размер сетки медианного метода
int med_size = 3;

BYTE* medium_blue = new BYTE[med_size * med_size];
BYTE* medium_green = new BYTE[med_size * med_size];
BYTE* medium_red = new BYTE[med_size * med_size];
```

```
void TwoDementionFilter() {
    int size = width * height;
    int count = size;
    long pos;
    for (int y = 1; y < height - 1; y++)
        for (int x = 1; x < width - 1; x++)
        {
            int size_byte = med_size * med_size;

            for(int y_i = y - 1, j = 0; y_i < y + 2;
y_i++, j++)
                for (int x_i = x - 1, i = 0; x_i < x +
2; x_i++, i++)
                {
                    pos = y_i * width + x_i;
                    medium_blue[j * med_size + i] =
src_image[pos].blue;
                    medium_green[j * med_size + i] =
src_image[pos].green;
                    medium_red[j * med_size + i] =
src_image[pos].red;
                }
            pos = y * width + x;

            src_image[pos].blue =
sort_byte(medium_blue, size_byte);
            src_image[pos].green =
sort_byte(medium_green, size_byte);
            src_image[pos].red = sort_byte(medium_red,
size_byte);
        }

}
```

Метод sort_byte необходим для поиска «среднего» значения сетки:

```
BYTE sort_byte(BYTE* mas, int size) {
    BYTE temp;
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (mas[j] > mas[j + 1]) {
                temp = mas[j];
                mas[j] = mas[j + 1];
                mas[j + 1] = temp;
            }
        }
    }
    return mas[size/2];
}
```

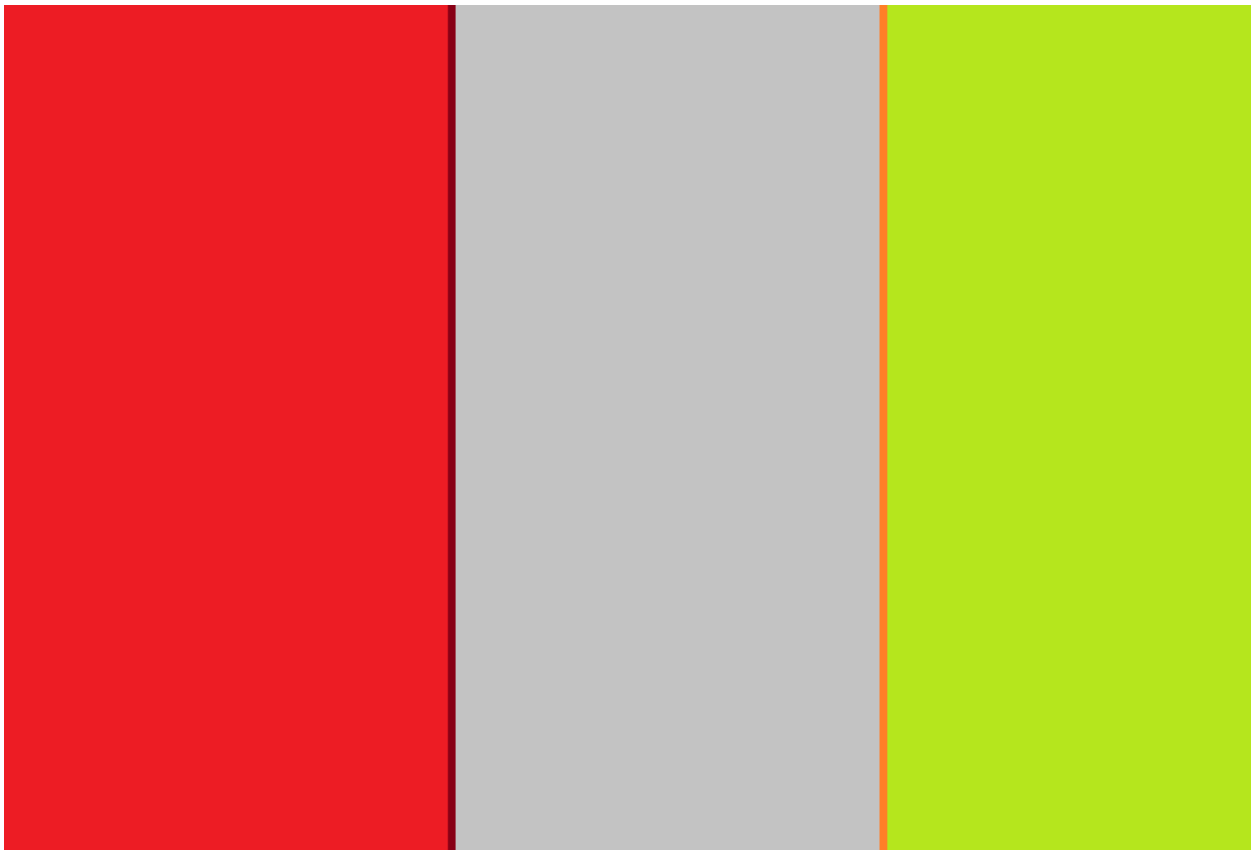
Пример работы программы:

Зашумление

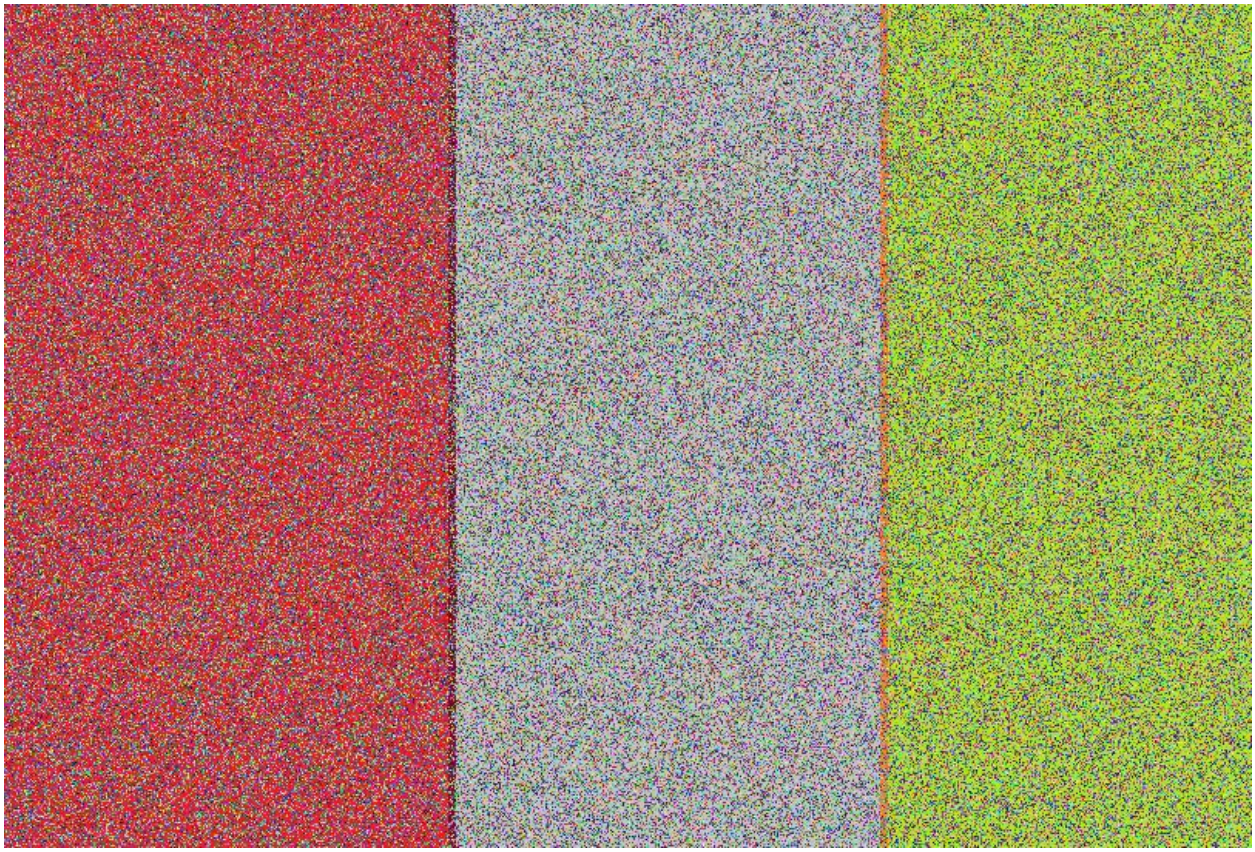
```
Меню:
        Выход                   ввести 0
        Зашумление              ввести 1
        Фильтрация              ввести 2
1
Ввести коэффициент зашумления (по умолчанию 15)
75
Enter path to image
C:\Users\Evgeny-PC\Pictures\Input\Lab1.bmp
```

```
Enter path to image
C:\Users\Evgeny-PC\Pictures\Res\Shum75.bmp
```

Lab1.bmp



Shum75.bmp

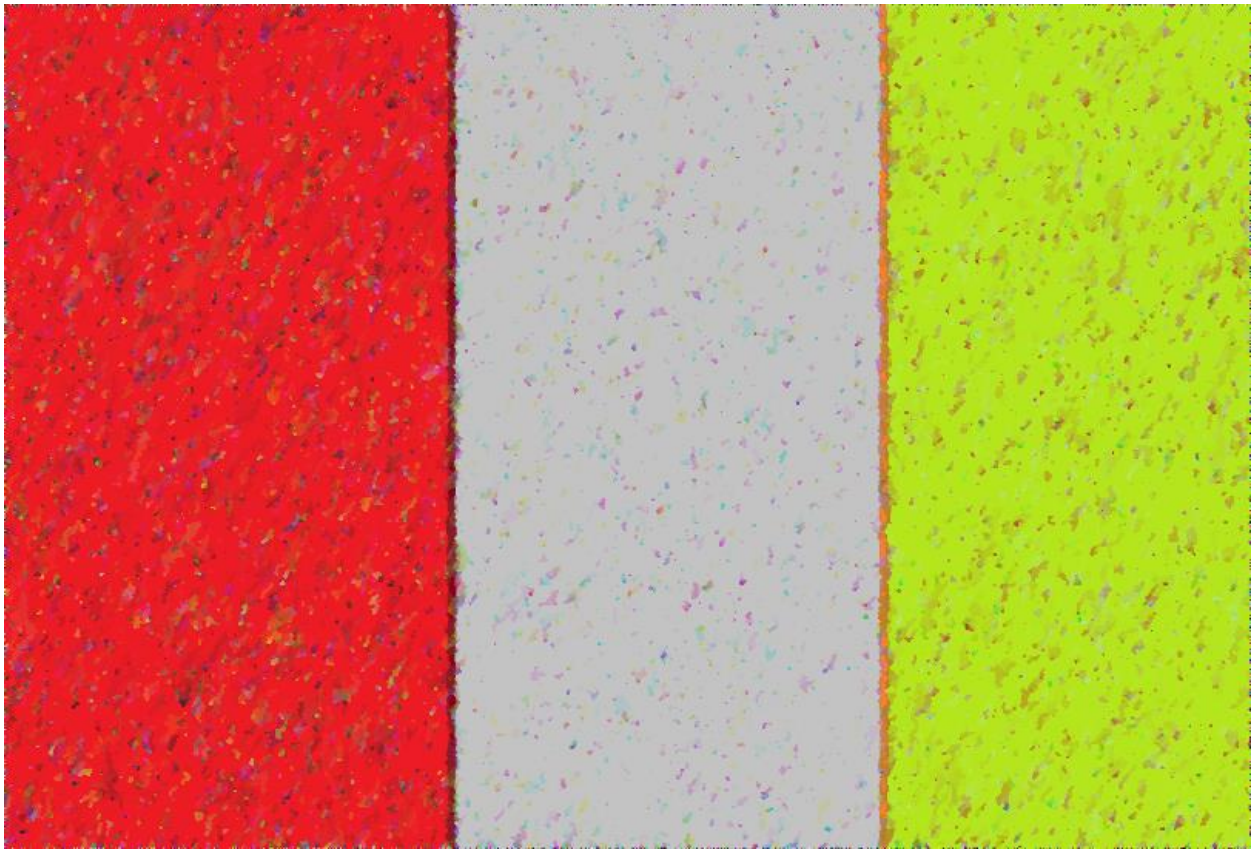


Фильтрация:

```
Меню:
        Выход                   ввести 0
        Зашумление              ввести 1
        Фильтрация              ввести 2
2
Enter path to image
C:\Users\Evgeny-PC\Pictures\Res\Shum75.bmp
```

```
Enter path to image
C:\Users\Evgeny-PC\Pictures\Res\FilterShum75.bmp
```

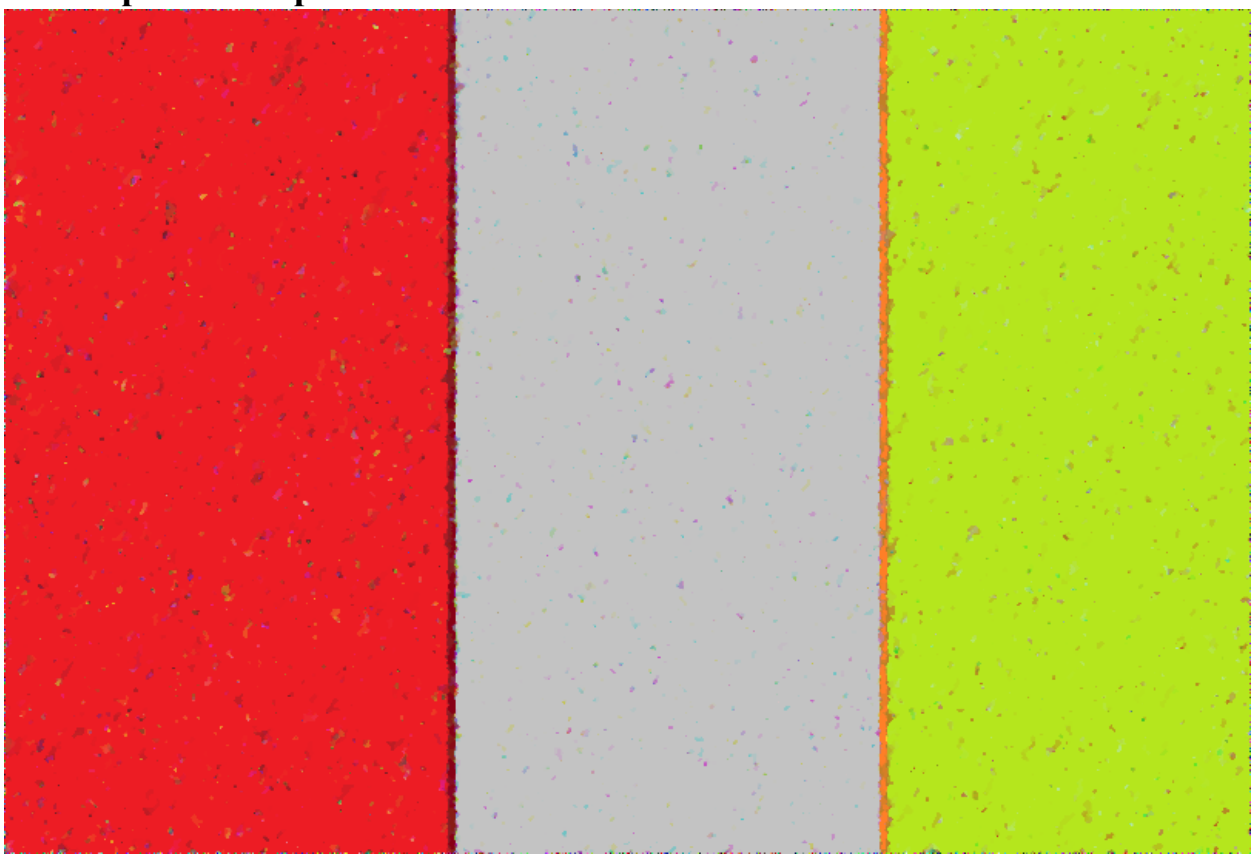
FilterShum75.bmp



Примеры фильтрации:

- **при 15% вероятности**

- **при 56% вероятности**



Вывод: в ходе работы изучил методы фильтрации изображения, в особенности двумерный медианный фильтр с квадратным окном. Узнал формат хранения изображения в байтовом представлении. А также разобрался в устройстве RGB модели.