УДК 517.444
ГРНТИ 27.23.21

РАЗРАБОТКА ДЕСКТОПНОГО ПРИЛОЖЕНИЯ ДЛЯ ЛИНЕЙНОЙ ФИЛЬТРАЦИИ РАСТРОВЫХ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ ЯЗЫКА ПРОГРАММИРОВАНИЯ C#

Близнец Игорь Васильевич
к.ф.-м.н., доцент кафедры фундаментальной и прикладной математики
Анисимов Сергей Алексеевич
студент 4-го курса специальности 1-31 03 06-01 «Экономическая кибернетика»
факультета математики и технологий программирования
УО Гомельский государственный университет им. Ф. Скорины
Республика Беларусь, г. Гомель

Аннотация: В статье рассматривается реализация алгоритма линейной фильтрации на основе дискретной свёртки с использованием языка программирования C. Описаны методы по представлению изображений, операции свёртки, шаблонам проектирования архитектуры. Представлена архитектура и код реализации приложения. Также показан результат фильтрации изображения приложением.

Ключевые слова: Фильтрация, пиксель, растр, представление, модель, C#, MVVM.

DEVELOPING DESKTOP APPLICATION FOR IMAGE LINEAR FILTRATION WITH C#

Bliznets Igor Vasilievich
Ph.D., Associate Professor of the Department of Fundamental and Applied Mathematics
Anisimau Siarhei Aliakseevich
4th year student of specialty 1-31 03 06-01 "Economic Cybernetics"
Faculty of Mathematics and Programming Technologies
Francisk Skorina Gomel State University
Republic of Belarus, Gomel

Abstract: The article discusses the implementation of a linear filtering algorithm based on discrete convolution using the C programming language. Methods for representing images, convolution operations, and architecture design patterns are described. The architecture and implementation code of the application are presented. The result of filtering the image by the application is also shown.

Keywords: Filtration, pixel, rast, view, model, C#, MVVM.

Данными для линейной фильтрации является растровое изображение. Растр – способ описания изображения через дискретизацию его на одинаковые элементы по однородной сетке и присвоению каждому элементу своего цветового атрибута. В исследовании рассмотрены прямоугольный растр. С математической точки зрения растр – это кусочно-постоянное приближение на плоскости непрерывной функции изображения.

Элементом растра является пиксель. Пиксель можно идентифицировать, как

$$f(i, j) = (A(i, j), C(i, j)),$$

где $A(i, j) \in R^2$ – область пикселя, $C(i, j)$ — атрибут пикселя (например, цвет).

Для фильтрации будем считать, что $C(i, j) = (R(i, j), G(i, j), B(i, j))$ - цветовые атрибуты в модели *RGB*.

Модель *RGB* в настоящее время является самой распространённой. Модель представляет цвет, как вектор пространства R^3, где R – красный цвет, G – зеленый, B – синий. Диапазон значений вектора является отрезок *[0, 255]* или *[0, 1]*. На рисунке 1 представлено полное трехмерное евклидово пространство цветов модели RGB

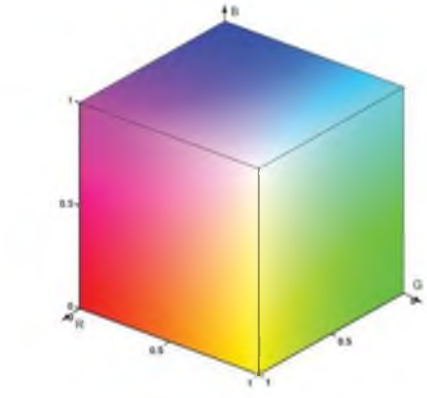

Рисунок 1 - Полное трехмерное евклидово пространство цветов модели RGB

Под фильтрацией понимается операция, которая в результате строит новое изображение того размера, полученное из исходного по некоторым правилам. Обычно цвет каждого пикселя результата обусловлен цветом пикселей, расположенных в некоторой его окрестности в исходном изображении.

Фильтрация является одной из самых основных операций компьютерного зрения. С той или иной фильтрации начинается работа подавляющего большинства методов обработки изображений.

Для линейной фильтрации будем считать, что задано полутоновое изображение A, и обозначим его цвета $A(x, y)$. Линейный фильтр определяется отображением $h: N^2 \rightarrow R$. Функцию $h(x, y)$ называют ядром фильтра. Операция линейной фильтрации производиться путём дискретной свертки:

$$B(x, y) = h(i, j) \times A(x, y) = \Sigma_i \Sigma_j h(i, j) * A(x - i, y - j)$$

Результат является изображение B. Обычно ядро фильтра не обращается в ноль только в некоторой окрестности (x, y). За пределами окрестности $h(i, j) \rightarrow 0$. Ядро фильтра может рассматриваться как матрица $m \times n$, где $m, n \in N$ – нечетные.

Для реализации фильтрации использовался язык C# вместе с технологией WPF. Само приложение представляет десктопную программу и библиотеку по обработке растровых изображений. Архитектура приложения выполнена по шаблону MVVM. Архитектура приложения изображена на рисунке 2.

MVVM (Model – View – ViewModel) используется для разделение моделей и их представления, что необходимо для независимого изменения.

MVVM делится на три части:

1) модель — логика работы с данными и описание фундаментальных данных,

2) представление — графический интерфейс,

3) модель представления — Содержит модель, преобразованную к представлению, а также команды, которыми представление может влиять на модели.

Разработанный пользовательский интерфейс представлен на рисунке 3.

Рисунок 2 - Архитектура приложения

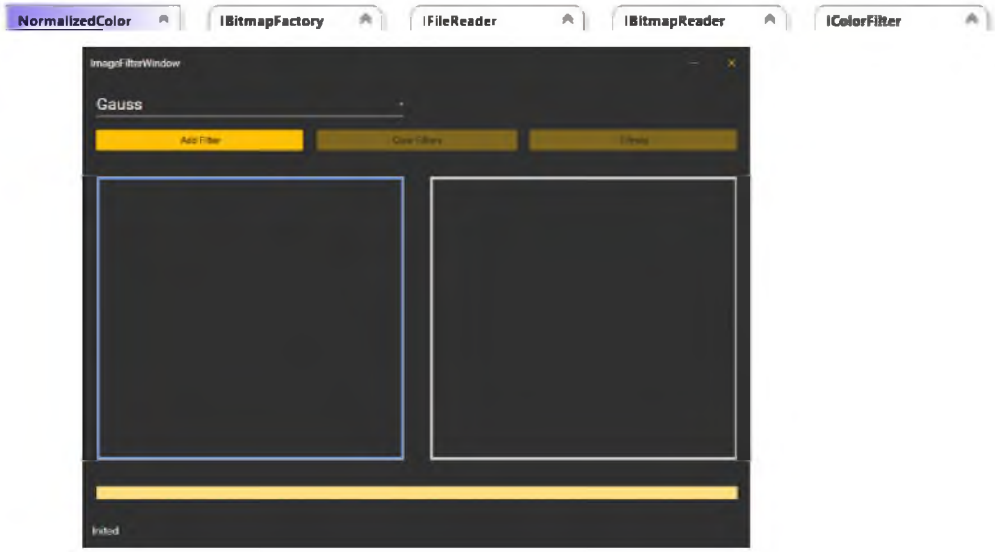

Рисунок 3 - Макет окна

Основная команда выполняется на кнопку «Filtrate». Происходит считывание данных входного изображения в память, далее идёт применение фильтров выбранных пользователем заранее из списка доступных, в конце идёт восстановление изображения (Рисунок 4).

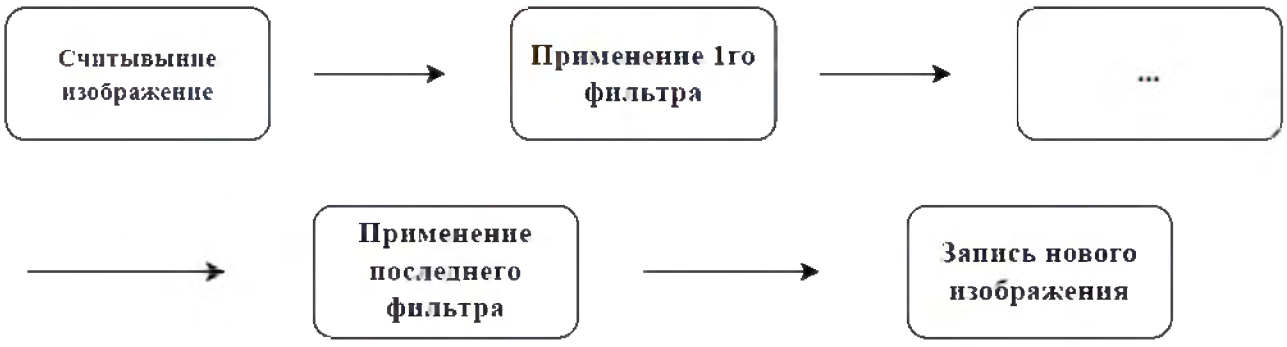

Рисунок 4 - Конвеер обработки изображения

Код реализации алгоритма линейной фильтрации представлены на рисунках на 5, 6, 7.

```
private NormalizedColor[][] SetPixels(BitmapData bitmapData, Bitmap bitmap)
{
    NormalizedColor[][] colors = new NormalizedColor[bitmapData.Height][];

    IntPtr firstPixel = bitmapData.Scan0;

    int bytes = Math.Abs(bitmapData.Stride) * bitmap.Height;
    byte[] rgbValues = new byte[bytes];

    Marshal.Copy(firstPixel, rgbValues, 0, bytes);

    for (int y = 0, t = 0; y < bitmap.Height; y++)
    {
        colors[y] = new NormalizedColor[bitmap.Width];
        for (int x = 0; x < bitmap.Width; x++, t += 3)
            colors[y][x] = CreateColor(rgbValues[t],
                                       rgbValues[t + 1],
                                       rgbValues[t + 2]);

        double percentage = y / (bitmap.Height * 1.0);

        _progress?.Report(CreateInfo(percentage * 100));
    }

    return colors;
}
```

Рисунок 5 - Реализация считывание пикселей изображения

```
public NormalizedColor[][] Filter(NormalizedColor[][] colors, Func<int, int, double> filter)
{
    _progress?.Report(CreateInfo(0));

    NormalizedColor[][] fColors = new NormalizedColor[colors.Length][];

    for (int y = 0; y < colors.Length; y++)
    {
        fColors[y] = new NormalizedColor[colors[y].Length];

        for (int x = 0; x < colors[y].Length; x++)
            fColors[y][x] = Convolution(x, y, colors, filter);

        double percentage = y / (colors.Length * 1.0);

        _progress?.Report(CreateInfo(percentage * 100));
    }

    return fColors;
}
```

Рисунок 6 - Реализация фильтрации

```
public NormalizedColor Convolution(int x, int y, NormalizedColor[][] colors, Func<int, int, double> filter)
{
    int lowX = x == 0 ? x : x - 1;
    int highX = x == colors[y].Length - 1 ? x : x + 1;

    int lowY = y == 0 ? y : y - 1;
    int highY = y == colors.Length - 1 ? y : y + 1;

    var sum = NormalizedColor.Zero;

    for (int i = lowY, t = 0; i <= highY; i++, t++)
    {
        for (int j = lowX, l = 0; j <= highX; j++, l++)
        {
            sum += filter(t, l) * colors[i][j];
        }
    }

    return sum;
}
```

Рисунок 7 - Реализация свёртки

На рисунке 8 изображен пример использования фильтрации для нахождения конура объекта на изображении.

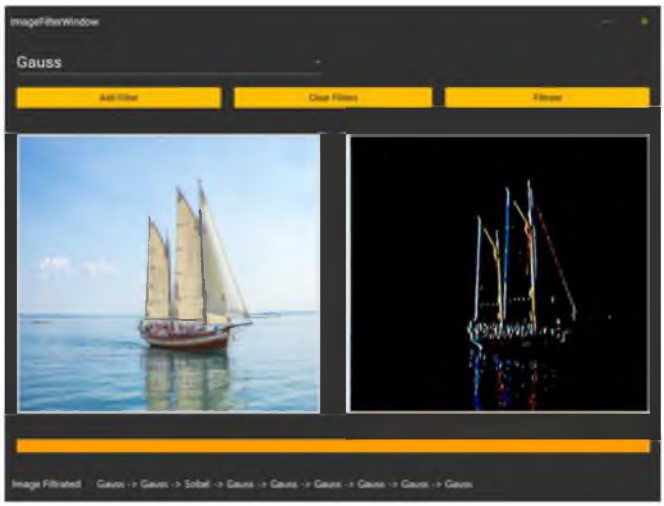

Рисунок 8 - Пример обработки

Разработанное приложение позволяет легко и просто провести базовый анализ изображения для дальнейшей его обработки. Оно может быть использовано в разный сферах касающихся компьютерного зрения.

Список литературы:

1. Лукин А. Введение в цифровую обработку сигналов (Математические основы).- Москва: МГУ, Лаборатория компьютерной графики и мультимедиа, 2007. – 54 с.