**Липецкий государственный технический университет**  
Факультет автоматизации и информатики  
Кафедра автоматизированных систем управления

Лабораторная работа №1

по предмету «Организация графических систем и систем мультимедиа»

Исследование методов обработки изображений

Максимов А.В.

Студент

Группа М-АС-21

Кургасов В.В.

Руководитель

Доцент

Липецк 2022 г.

Задание кафедры

Разработать программу, позволяющую применить не менее 3-х фильтров к загруженному изображению и экспортировать результат в иной графический формат. Необходимо наличие графического меню пользователя. Обработка должна проводиться на CPU.

Содержание

[Теоретические сведения 4](#_Toc103097177)

[Ход работы 4](#_Toc103097178)

[Вывод 8](#_Toc103097179)

[Приложение А 9](#_Toc103097180)

[Приложение Б 15](#_Toc103097181)

Теоретические сведения

В качестве языка реализации был выбран C# с использованием технологии Windows Forms. Это объектно-ориентированный язык компилируемого типа. К преимуществам можно отнести простоту создания пользовательского интерфейса, а также наличие стандартных инструментов для работы с графикой.

Ход работы

Приложение позволяет пользователю наложить на загруженное изображение один из трех фильтров: черно-белое фото, размытие по Гауссу, негатив фотографии

Исходный код программы представлен в приложении А.

Черно-белое фото

Преобразование цветного изображения в модели RGB к черно-белому заключается в замене разных значений красного, зеленого, и синего цвета для данной точки на единое значение, характеризующее яркость точки. На цвет в стандартной RGB модели отводится 1 байт, соответственно, при таком преобразовании точки могут принимать значения из диапазона [0,0,0], [1,1,1], [2,2,2]...[255,255,255]. Где [0,0,0] - черный цвет, а [255,255,255] - белый цвет. Цветное изображение таким образом представляется 256 оттенками серого. Так как цвета с разной длиной волны по-разному воспринимаются человеческим глазом, то для вычисления яркости точки разным цветам назначаются разные веса. Наиболее чувствителен человек к зеленому цвету, поэтому для него используется самый большой вес - 0.7152, наименее чувствителен к синему, для которого используется вес 0.0722, и для красного используется вес 0.2126. Эти веса прописаны в стандарте sRGB – именно так вычисляется яркость точки при переходе от sRGB в цветовое пространство CIE XYZ. На рисунке 1 представлен результат наложения фильтра.

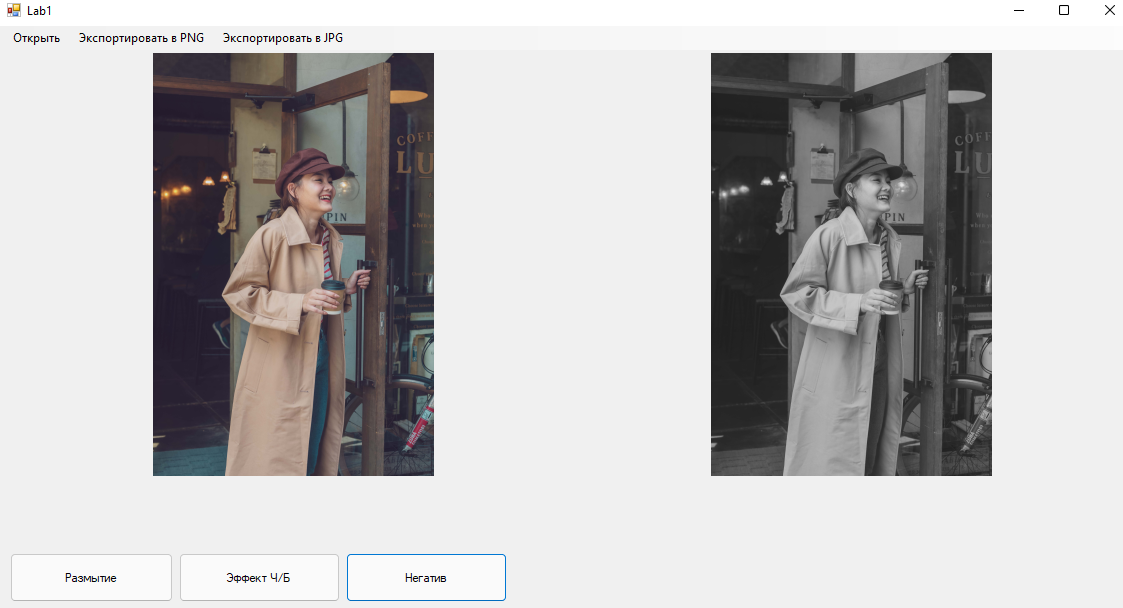


Рисунок 1 – Черно-белое фото

Размытие по Гауссу

Размытие по Гауссу в цифровой обработке изображений — способ размытия изображения с помощью функции Гаусса, названной в честь немецкого математика Карла Фридриха Гаусса.

Этот эффект широко используется в графических редакторах для уменьшения шума изображения и снижения детализации. Визуальный эффект этого способа размытия напоминает эффект просмотра изображения через полупрозрачный экран, и отчётливо отличается от эффекта боке, создаваемого расфокусированным объективом или тенью объекта при обычном освещении.

Размытие по Гауссу также используется в качестве этапа предварительной обработки в алгоритмах компьютерного зрения для улучшения структуры изображения в различных масштабах.

Результат наложения фильтра представлен на рисунке 2.

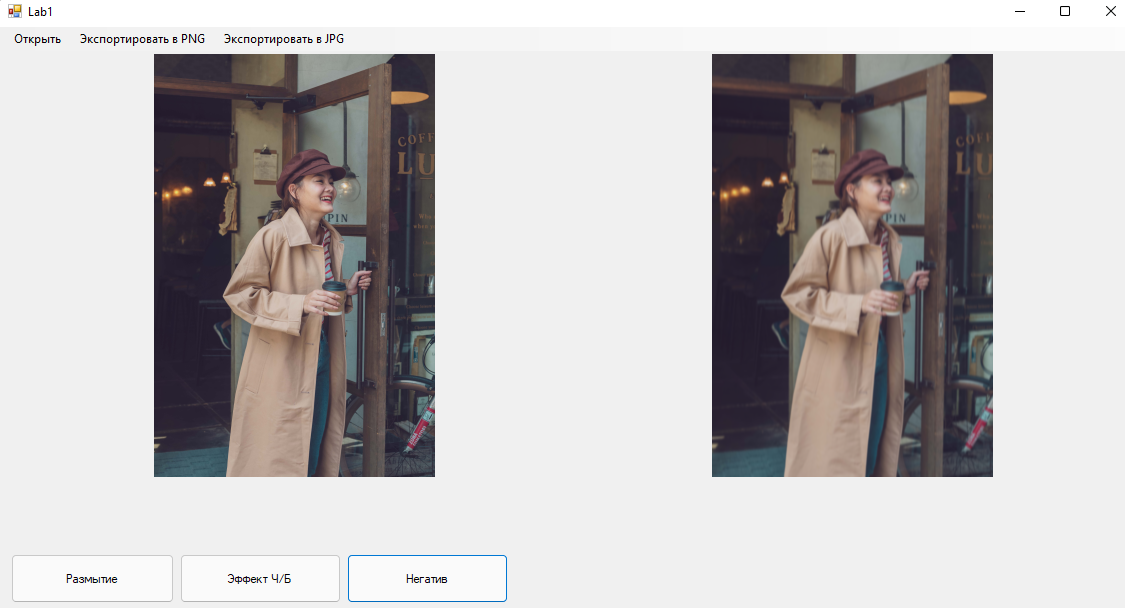


Рисунок 2 – Размытие по Гауссу

Негатив фотографии

Чтобы перевести изображение в негатив нужно пройти все пиксели по одному и вычесть из 255 текущее значение пикселя, чтобы создать пиксель отрицательного цвета. Затем сохранить значение этого пикселя на изображении в том же положении. Код, выполняющий

Результат работы фильтра представлен на рисунке 3.

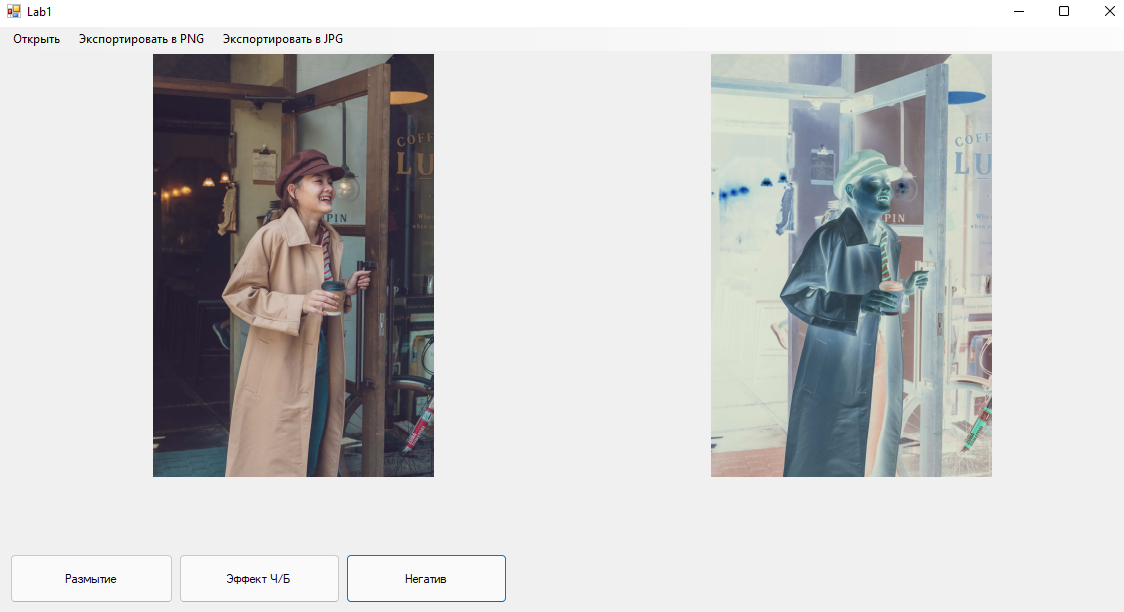


Рисунок 3 – Негатив

Для сохранения полученного изображения можно воспользоваться функцией Save класса Bitmap, указав в аргументах путь, по которому сохранять изображение и формат, с помощью класса ImageFormat.

Создадим инсталлятор для разработанного приложения.

Для этого установим плагин Microsoft Visual Studio Installer Projects. После этого добавим к решению инсталлятор. На рисунке 4 показана структура проекта после добавления инсталлятора.

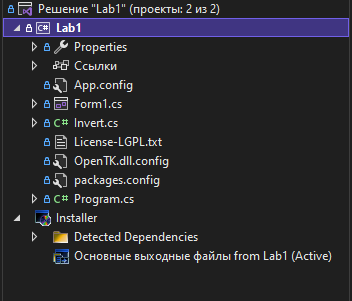


Рисунок 4 – Структура проекта

После сборки проекта получаем два файла: setup.exe и lab1.msi.

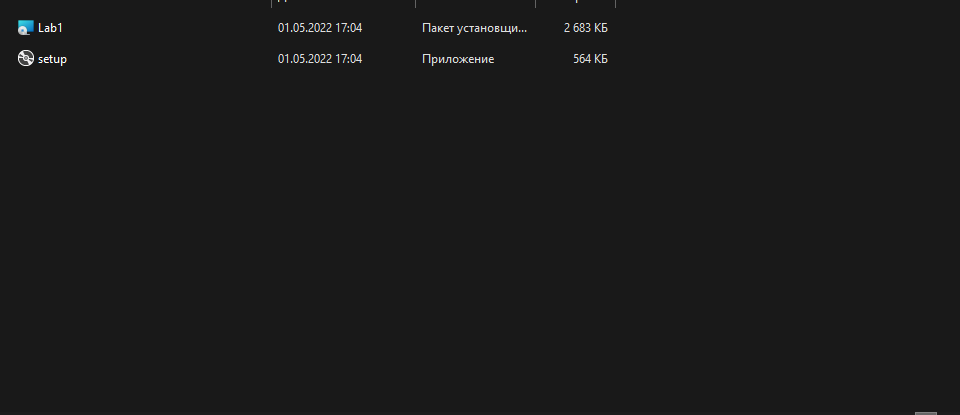


Рисунок 5 – Результат сборки проекта Installer.

Вывод

В ходе выполнения лабораторной работы было разработано приложение, позволяющая пользователю наложить на изображение один из 3х фильтров (черно-белое фото, размытие по Гауссу, негатив), а получившийся результат перевести в другой формат. Для разработки использовался язык программирования C#, технологии Windows Form, а также библиотека для работы с изображениями EmguCV.

Приложение А

Исходный код программы

using System;

using System.Drawing;

using System.Drawing.Imaging;

using System.Threading;

using System.Windows.Forms;

using Emgu.CV;

using Emgu.CV.Structure;

namespace Lab1

{

public partial class Form1 : Form

{

private Image<Bgr, byte> inputImage = null;

private Mat outputImage = new Mat();

public Form1()

{

InitializeComponent();

}

private void cvColor()

{

CvInvoke.CvtColor(inputImage, outputImage, Emgu.CV.CvEnum.ColorConversion.Bgr2Gray);

pictureBox2.Image = outputImage.Bitmap;

}

private void cvGaussianBlur()

{

CvInvoke.GaussianBlur(inputImage, outputImage, new Size(), 8.5, 20);

pictureBox2.Image = outputImage.Bitmap;

}

private void negative()

{

Bitmap value = (Bitmap)Invert.invert(inputImage.Bitmap);

Image<Bgr, byte> img = new Image<Bgr, byte>(value);

outputImage = img.Mat;

pictureBox2.Image = outputImage.Bitmap;

}

private void открытьToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

DialogResult res = openFileDialog1.ShowDialog();

if(res == DialogResult.OK)

{

inputImage = new Image<Bgr, byte>(openFileDialog1.FileName);

pictureBox1.Image = inputImage.Bitmap;

}

else

{

MessageBox.Show("Файл не выбран", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch(Exception ex)

{

MessageBox.Show(ex.Message,"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button1\_Click(object sender, EventArgs e)

{

try

{

button1.Enabled = false;

button2.Enabled = false;

button3.Enabled = false;

Thread th = new Thread(new ThreadStart(cvGaussianBlur));

th.Start();

th.Join();

button1.Enabled = true;

button2.Enabled = true;

button3.Enabled = true;

}

catch(Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button2\_Click(object sender, EventArgs e)

{

try

{

button1.Enabled = false;

button2.Enabled = false;

button3.Enabled = false;

Thread th = new Thread(new ThreadStart(cvColor));

th.Start();

th.Join();

button1.Enabled = true;

button2.Enabled = true;

button3.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void button3\_Click(object sender, EventArgs e)

{

try

{

button1.Enabled = false;

button2.Enabled = false;

button3.Enabled = false;

Thread th = new Thread(new ThreadStart(negative));

th.Start();

th.Join();

button1.Enabled = true;

button2.Enabled = true;

button3.Enabled = true;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void exportToolStripMenuItem\_Click(object sender, EventArgs e)

{

try

{

DialogResult res = saveFileDialog1.ShowDialog();

if (res == DialogResult.OK)

{

Bitmap bmp1 = outputImage.Bitmap;

bmp1.Save(saveFileDialog1.FileName, ImageFormat.Png);

}

else

{

MessageBox.Show("Файл не выбран", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

Приложение Б

Исходный код алгоритма перевода изображения в негатив

using System;

using System.Drawing;

using System.Drawing.Imaging;

namespace Lab1

{

class Invert

{

internal static Image invert(Image original)

{

Bitmap myImage = new Bitmap(original);

BitmapData imageData = myImage.LockBits(new Rectangle(0, 0, myImage.Width, myImage.Height), ImageLockMode.ReadWrite, PixelFormat.Format32bppArgb);

int stride = imageData.Stride;

IntPtr Scan0 = imageData.Scan0;

unsafe

{

byte\* p = (byte\*)(void\*)Scan0;

int nOffset = stride - myImage.Width \* 4;

int nWidth = myImage.Width;

for (int y = 0; y < myImage.Height; y++)

{

for (int x = 0; x < nWidth; x++)

{

p[0] = (byte)(255 - p[0]);

p[1] = (byte)(255 - p[1]);

p[2] = (byte)(255 - p[2]);

p += 4;

}

p += nOffset;

}

}

myImage.UnlockBits(imageData);

return (Image)myImage;

}

}

}