**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ**

**ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине **«Основы цифровой обработки изображений»**

на тему: «Частотные методы улучшения изображений»

Выполнил: студент гр. ИТ-31

Рябиков М.А.

Принял: преподаватель

Цалко И.Н.

Гомель 2019

**Цель:** научиться применять основные методы фильтрации в частотной области при решении практических задач улучшения изображений.

1. **Задание**
2. Ввести изображение размером 256х256 элементов.
3. Реализовать прямое и обратное преобразование Фурье с помощью алгоритма БПФ.
4. Вывести Фурье-образ в виде двумерного изображения, выполнив предварительную обработку методом гаммирования (подбор оптимального значения гамма выполнить самостоятельно).
5. В соответствии с вариантом реализовать низкочастотную фильтрацию изображения (см. табл. 1), вывести преобразованное изображение, сравнить с исходным.
6. Реализовать фильтрацию исходного изображения с усилением высоких частот. Выбор высокочастотного фильтра выполнить в соответствии с вариантом (см. табл. 1). Вывести преобразованное изображение, сравнить с исходным.

|  |  |  |
| --- | --- | --- |
| Вариант | Низкочастотный фильтр | Высокочастотный фильтр |
| 6(21) | Идеальный, позволяющий удалить 8% энергии изображения | Гаусса |

**Ход работы.**

Исходное изображение:

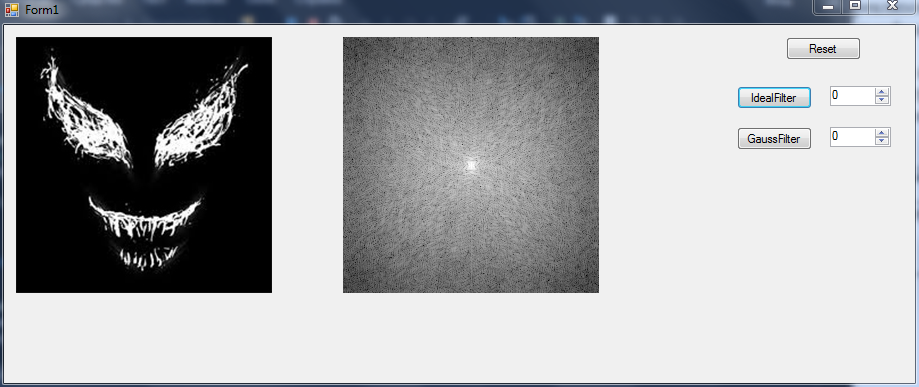


Рисунок 1 – Исходное изображение

Идеальный фильтр:

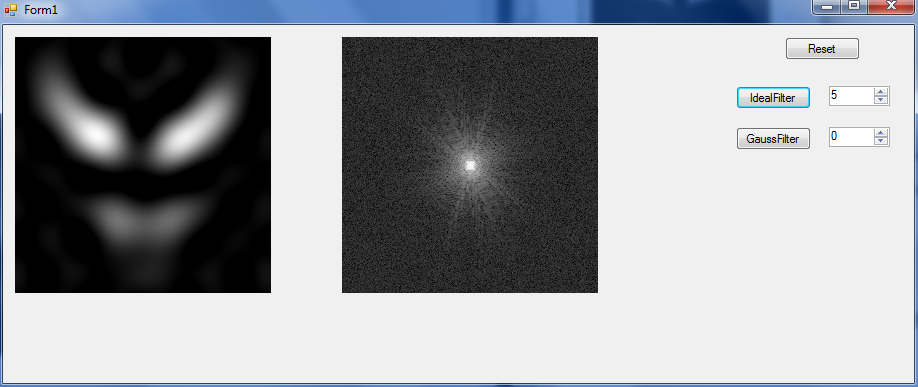


Рисунок 2 – Идеальный фильтр

Фильтр Гаусса:

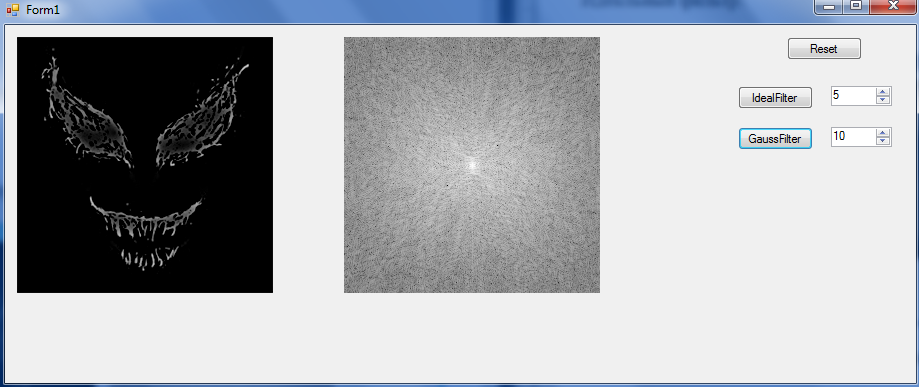


Рисунок 3 – Высококачественная фильтрация

**Листинг программы**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab4

{

public struct Complex

{

public double r;//действительная часть

public double i;//мнимая часть

public Complex(double r, double i)

{

this.r = r;

this.i = i;

}

public Complex(double real)

{

r = real;

i = 0;

}

public Complex Addition(Complex c)

{

return new Complex(r + c.r, i + c.i);

}

public Complex Subtraction(Complex c)

{

return new Complex(r - c.r, i - c.i);

}

public Complex Multiplication(Complex c)

{

return new Complex(r \* c.r - i \* c.i, r \* c.i + i \* c.r);

}

public Complex Division(int d)

{

return new Complex(r / d, i / d);

}

public Complex Conjugation()

{

return new Complex(r, -i);//комплексное сопряженное число

}

}

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

Bitmap bmp;

private void Form1\_Load(object sender, EventArgs e)

{

button3\_Click(sender, e);

}

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

{

button3\_Click(sender, e);

Complex[,] complex = ForwardFourier(bmp);

IdealFilter(complex, Convert.ToDouble(numericUpDown1.Value));

InverseFourier2(complex, bmp);

pictureBox1.Image = bmp;

bmp.Save(@"IdealFilter.bmp");

Fourier();

}

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

{

button3\_Click(sender, e);

Complex[,] complex = ForwardFourier(bmp);

Gauss(complex, Convert.ToDouble(numericUpDown2.Value));

InverseFourier2(complex, bmp);

pictureBox1.Image = bmp;

bmp.Save(@"GaussFilter.bmp");

Fourier();

}

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

{

bmp = new Bitmap(@"2.bmp");

pictureBox1.Image = bmp;

Fourier();

}

Complex w(int i, int n)

{

if (i % n == 0) return new Complex(1, 0);

double arg = -2.0 \* Math.PI \* i / n;//4.2-8

return new Complex(Math.Cos(arg), Math.Sin(arg));//4.2-7

}

Complex[] FastFourier(Complex[] x, int n) //быстрое преобразование фурье(симитрично)

{

if (n <= 1) return x;

Complex[] x1 = new Complex[n / 2];//одномерные массивы

Complex[] x2 = new Complex[n / 2];

for (int i = 0; i < n / 2; i++)

{

x1[i] = x[i \* 2];//для четных эл-ов столб или строки

x2[i] = x[i \* 2 + 1];

}

x1 = FastFourier(x1, n / 2);

x2 = FastFourier(x2, n / 2);

for (int i = 0; i < n / 2; i++)

{

Complex complex = w(i, n).Multiplication(x2[i]);

x[i] = x1[i].Addition(complex);

x[i + n / 2] = x1[i].Subtraction(complex);

}

return x;

}

Complex[] InverseFourier(Complex[] x, int n)//обратное Фурье

{

for (int i = 0; i < n; i++)

x[i] = x[i].Conjugation();

x = FastFourier(x, n);

for (int i = 0; i < n; i++)

x[i] = x[i].Conjugation().Division(n);

return x;

}

Complex[,] ForwardFourier(Bitmap bmp) //прямое преобразование Фурье

{

Complex[,] matrix = new Complex[bmp.Width, bmp.Height];

int width = bmp.Width;

int height = bmp.Height;

////формирование матр компл чисел

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

double color = (bmp.GetPixel(i, j).B + bmp.GetPixel(i, j).G + bmp.GetPixel(i, j).R) \* 1.0d / 768; //получение цвета

if ((i + j) % 2 == 1)//центровка низких частот 4.2-21

color \*= -1.0d;

matrix[i, j] = new Complex(color);

}

}

Complex[] complex = new Complex[width];

for (int j = 0; j < height; j++)//постолбцовое преобразование Фурье

{

for (int i = 0; i < width; i++)

complex[i] = matrix[i, j];

complex = FastFourier(complex, width);//быстр преоб для строки

for (int i = 0; i < width; i++)

matrix[i, j] = complex[i];

}

complex = new Complex[height];

for (int i = 0; i < width; i++)//построчное преобразование Фурье

{

for (int j = 0; j < height; j++)

complex[j] = matrix[i, j];

complex = FastFourier(complex, height);

for (int j = 0; j < height; j++)

matrix[i, j] = complex[j];

}

return matrix;

}

//идеальный фильтр низких частот

void IdealFilter(Complex[,] c, double r)

{

int n = c.GetLength(0);

int m = c.GetLength(1);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

if (Math.Sqrt((n / 2 - i) \* (n / 2 - i) + (m / 2 - j) \* (m / 2 - j)) > r)//нахож рас от центра 4.3-2 4.3-3

c[i, j] = c[i, j].Multiplication(new Complex(0.0d));//4.3-2

}

}

}//фильтры высоких частот

void Gauss(Complex[,] c, double r)

{

int n = c.GetLength(0);

int m = c.GetLength(1);

for (int i = 0; i < n; i++)

{

for (int j = 0; j < m; j++)

{

double r2 = Math.Sqrt((n / 2 - i) \* (n / 2 - i) + (m / 2 - j) \* (m / 2 - j));//4.4-4

c[i, j] = c[i, j].Multiplication(new Complex(1.0 - Math.Exp(-r2 \* r2 / (2 \* r \* r))));//умн на фурье образ

}

}

}

void InverseFourier2(Complex[,] c, Bitmap bmp)

{

int width = c.GetLength(0);

int height = c.GetLength(1);

Complex[] complex = new Complex[width];

for (int j = 0; j < height; j++)//постолбцовое обратное преобразование фурье

{

for (int i = 0; i < width; i++)

complex[i] = c[i, j];

complex = InverseFourier(complex, width);

for (int i = 0; i < width; i++)

c[i, j] = complex[i];

}

complex = new Complex[height];

for (int i = 0; i < width; i++)//построковое обратное преобразование фурье

{

for (int j = 0; j < height; j++)

complex[j] = c[i, j];

complex = InverseFourier(complex, height);

for (int j = 0; j < height; j++)

c[i, j] = complex[j];

}

for (int i = 0; i < width; i++)

{

for (int j = 0; j < height; j++)

{

double color = c[i, j].r;

if ((i + j) % 2 == 1)//убираем центровку

color \*= -1.0d;

if (color < 0.0d)

color = 0;

else if (color > 1.0d)

color = 1.0d;

int rgb = (int)(color \* 255);

bmp.SetPixel(i, j, Color.FromArgb(rgb, rgb, rgb));

}

}

}

double GetEnergy(Complex c)//ф обр правл

{

double r = c.r;

double i = c.i;

return r \* r + i \* i;

}

void Fourier()

{

Complex[,] c = ForwardFourier(bmp);

Bitmap bmp2 = new Bitmap(bmp);

double max = double.MinValue;

double min = double.MaxValue;

for (int i = 0; i < bmp.Width; i++)

{

for (int j = 0; j < bmp.Height; j++)

{

double p = GetEnergy(c[i, j]);//для каждого пикселя

if (p > max)

max = p;

if (p < min)

min = p;

}

}

double d = (Math.Exp(25) - 1.0d) / (max - min);

for (int i = 0; i < bmp.Width; i++)

{

for (int j = 0; j < bmp.Height; j++)

{

double p = GetEnergy(c[i, j]);

double s = Math.Log((p - min) \* d + 1.0d, Math.E) \* 11.0d;

if (s > 255)

s = 255;

else if (s < 0)

s = 0;

bmp2.SetPixel(i, j, Color.FromArgb((int)s, (int)s, (int)s));

}

}

pictureBox2.Image = bmp2;

}

}

}

**Вывод**: В данной лабораторной работе были получены навыки применения основных методов фильтрации в частотной области при решении практических задач улучшения изображений.