ЗВІТ З ЛАБОРАТОРНОЇ РОБОТИ №3

За курсом «Обробка та розпізнавання зображень»

Студента групи ПА-19-2

Ільяшенко Єгора Віталійовича

Кафедра комп’ютерних технологій, ДНУ

**Точкові методи обробки зображень.**

Постановка задачі

Table

Description automatically generated

Варіант 7





Graphical user interface, application, PowerPoint

Description automatically generated

Рельєфне тиснення - це техніка комп'ютерної графіки, в якій кожен піксель зображення замінюється або світлим, або темним, залежно від світлих/темних меж на оригінальному зображенні. Низько контрастні ділянки замінюються сірим фоном. Відфільтроване зображення відображатиме швидкість зміни кольору в кожній точці оригінального зображення. Застосування фільтра рельєфу до зображення часто призводить до того, що зображення нагадує паперове або металеве тиснення оригінального зображення, звідси і назва.

Рельєфний фільтр, який також називають фільтром різниці напрямків, покращує краї в напрямку вибраної маски (масок) згортки. Коли застосовується фільтр рельєфу, матриця фільтра обчислюється з тією самою площею на оригінальному зображенні. Це призводить до значних обчислень, якщо розмір зображення або розмір маски фільтра рельєфного тиснення великий. Рельєфний фільтр повторює обчислення, закодовані в матриці фільтра, для кожного пікселя зображення; сама процедура порівнює сусідні пікселі на зображенні, залишаючи мітку там, де виявлено різку зміну значення пікселя. Таким чином, мітки утворюють лінію, що повторює контур об'єкта. У результаті виходить рельєфне зображення з виділеними краями.

При застосування алгоритму тиснення, я використовував динамічну ділянку порівняння пікселів (радіус). Після того, як ми вибрали бажаний радіус, для пікселя буде знайдено інший піксель у цьому радіусі, з яким і буде іти зрівняння. Оскільки ці рамки рандомізації визначаються динамічно, і ми не можемо отримати outOfBound exception, ми можемо задати дуже великі значення цього радіусу, та отримати незвичайні результати.

A picture containing text

Description automatically generated

Text

Description automatically generated with low confidence

A picture containing text

Description automatically generated

A picture containing graphical user interface

Description automatically generated

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

Graphical user interface, website

Description automatically generated

Висновки: Виконуючи лабораторну роботу №3, я розібрався у темі обробки зображень матричними фільтрами, розробив програму на мові програмування c#, яка приймає на вхід зображення, та виконує на ньому алгоритм тиснення. Радіус тиснення береться рандомізовано, враховуючи границі. Вивід програми можна завантажити у файл. Результатом використання алгоритму стає виділення границь зображення. При збільшенні радіусу, збільшується шум, але при деяких зображеннях, це виглядає стильно.

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Globalization;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab2

{

public partial class Form1 : Form

{

string[] args;

Graphics gr;

Bitmap inputBmp;

Bitmap outBmp;

public Form1()

{

InitializeComponent();

gr = pictureBox1.CreateGraphics();

gr.Clear(Color.Yellow);

pictureBox1.Refresh();

args = Environment.GetCommandLineArgs();

label4.Text = $"Pixel radius: {trackBar1.Value}";

}

void LoadImage(string path)

{

Image inputImage = Bitmap.FromFile(path);

gr.DrawImage(inputImage, pictureBox1.DisplayRectangle);

inputBmp = new Bitmap(inputImage);

}

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

{

if (openFileDialog1.ShowDialog() != DialogResult.OK)

return;

//if (openFileDialog1.FileName.Substring(openFileDialog1.FileName.Length - 4) != ".bmp")

// return;

LoadImage(openFileDialog1.FileName);

}

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

{

if (inputBmp == null)

return;

outBmp = ApplyEmbossFilter(inputBmp);

Graphics outGr = pictureBox2.CreateGraphics();

outGr.DrawImage(outBmp, pictureBox2.DisplayRectangle);

}

Bitmap ApplyEmbossFilter(Bitmap image)

{

Bitmap result = new Bitmap(image.Width, image.Height);

Random rnd = new Random();

for (int x = 0; x < image.Width; ++x)

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

{

Color pixelColor = image.GetPixel(x, y);

Color embossedColor = Color.FromArgb(pixelColor.R, pixelColor.G, pixelColor.B);

int border = trackBar1.Value;

int newX = x - rnd.Next(Math.Max(x - image.Width + 1, -border), Math.Min(x, border));

int newY = y - rnd.Next(Math.Max(y - image.Height + 1, -border), Math.Min(y, border));

Color prevPixelColor = image.GetPixel(newX, newY);

int diffR = pixelColor.R - prevPixelColor.R + 128;

int diffG = pixelColor.G - prevPixelColor.G + 128;

int diffB = pixelColor.B - prevPixelColor.B + 128;

embossedColor = Color.FromArgb(Clamp(diffR, 0, 255), Clamp(diffG, 0, 255), Clamp(diffB, 0, 255));

result.SetPixel(x, y, embossedColor);

}

return result;

}

static int Clamp(int value, int minValue, int maxValue)

{

if (value < minValue)

{

return minValue;

}

else if (value > maxValue)

{

return maxValue;

}

else

{

return value;

}

}

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

{

if (outBmp == null)

return;

outBmp.Save($"{args[0].Substring(0, args[0].LastIndexOf('\\'))}\\{textBox2.Text}.bmp");

}

private void trackBar1\_Scroll(object sender, EventArgs e) => label4.Text = $"Pixel radius: {trackBar1.Value}";

}

}