# 实

# 验

# 报

# 告

# 学院：电子信息与通信学院

# 班级：信卓2101班

# 姓名：吴永浩

# 学号：U202114208

# 实验时间：2024年3月19日

目录

[一、 实验目的 1](#_Toc161951130)

[二、 实验原理 1](#_Toc161951131)

[三、 编程环境介绍 1](#_Toc161951132)

[四、 程序结构及流程 1](#_Toc161951133)

[五、 实验结果 1](#_Toc161951134)

# 实验目的

1. 通过编程工具，自己编写代码实现直方图均衡算法。

# 实验原理

直方图均衡算法原理：

直方图均衡化又称直方图修平，是一种很重要的非线性点运算。使用该方法可以加强图像的局部对比度，尤其是当图像的有用数据的对比度相当接近的时候。通过这种方法，亮度可以更好的在直方图上分布。

直方图均衡化的基本思想是把原始图像的直方图变换为均匀分布的形式。这样增加了灰度值的动态范围，从而达到增强图像整体对比度的效果。

算法步骤：

1）计算图像f(x,y)的各灰度级中像素出现的概率p(i)。

2) 计算p的累计概率函数c(i),c即为图像的累计归一化直方图

3）将c(i)缩放至0~255范围内。

# 编程环境介绍

本实验使用visual studio IDE的Windows窗体编程环境。

WinForm 是 Windows Form 的简称，是基于 .NET Framework 平台的客户端（PC软件）开发技术，一般使用C#编程。C# WinForm 编程需要创建「Windows窗体应用程序」项目。

.NET 提供了大量 Windows 风格的控件和事件，我们可以直接拿来使用，上手简单，开发快速

Windows 窗体应用程序是 C# 语言中的一个重要应用，也是 C# 语言最常见的应用。

本实验主要使用c#语言进行编程。

# 程序结构及流程

实验程序主要由两部分组成：主界面窗体控制部分和绘制灰度直方图窗体比分。

以下为代码：

**主窗体代码**

namespace DIP\_02

{

public partial class Form1 : Form

{

//private HiPerfTimer myTimer;

public Form1()

{

InitializeComponent();

//myTimer = new DIP\_02.HiPerfTimer();

}

private string curFileName;

//图像对象

private System.Drawing.Bitmap curBitmpap;

private void open\_Click(object sender, EventArgs e)//打开文件

{

//创建OpenFileDialog

OpenFileDialog opnDlg = new OpenFileDialog();

//为图像选择一个筛选器

opnDlg.Filter = "所有图像文件|\*.bmp;\*.pcx;\*.png;\*.jpg;\*.gif;" +

"\*.tif;\*.ico;\*.dxf;\*.cgm;\*.cdr;\*.wmf;\*.eps;\*.emf|" +

"位图(\*.bmp;\*.jpg;\*.png;...)|\*.bmp;\*.pcx;\*.png;\*.jpg;\*.gif;\*.tif;\*.ico|" +

"矢量图(\*.wmf;\*.eps;\*.emf;...)|\*.dxf;\*.cgm;\*.cdr;\*.wmf;\*.eps;\*.emf";

//设置对话框标题

opnDlg.Title = "打开图像文件";

//启用“帮助”按钮

opnDlg.ShowHelp = true;

//如果结果为“打开”，选定文件

if (opnDlg.ShowDialog() == DialogResult.OK)

{

//读取当前选中的文件名

curFileName = opnDlg.FileName;

//使用Image.FromFile创建图像对象

try

{

curBitmpap = (Bitmap)Image.FromFile(curFileName);

}

catch (Exception exp)

{

MessageBox.Show(exp.Message);

}

}

//对窗体进行重新绘制，这将强制执行paint事件处理程序

Invalidate();

}

private void Form1\_Paint(object sender, PaintEventArgs e)

{

//获取Graphics对象

Graphics g = e.Graphics;

if (curBitmpap != null)

{

//使用DrawImage方法绘制图像

//160,20：显示在主窗体内，图像左上角的坐标

//curBitmpap.Width, curBitmpap.Height图像的宽度和高度

g.DrawImage(curBitmpap, 160, 20, curBitmpap.Width, curBitmpap.Height);

}

}

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

{

//如果没有创建图像，则退出

if (curBitmpap == null)

return;

//调用SaveFileDialog

SaveFileDialog saveDlg = new SaveFileDialog();

//设置对话框标题

saveDlg.Title = "保存为";

//改写已存在文件时提示用户

saveDlg.OverwritePrompt = true;

//为图像选择一个筛选器

saveDlg.Filter = "BMP文件(\*.bmp)|\*.bmp|" + "Gif文件(\*.gif)|\*.gif|" + "JPEG文件(\*.jpg)|\*.jpg|" + "PNG文件(\*.png)|\*.png";

//启用“帮助”按钮

saveDlg.ShowHelp = true;

//如果选择了格式，则保存图像

if (saveDlg.ShowDialog() == DialogResult.OK)

{

//获取用户选择的文件名

string filename = saveDlg.FileName;

string strFilExtn = filename.Remove(0, filename.Length - 3);

//保存文件

switch (strFilExtn)

{

//以指定格式保存

case "bmp":

curBitmpap.Save(filename, System.Drawing.Imaging.ImageFormat.Bmp);

break;

case "jpg":

curBitmpap.Save(filename, System.Drawing.Imaging.ImageFormat.Jpeg);

break;

case "gif":

curBitmpap.Save(filename, System.Drawing.Imaging.ImageFormat.Gif);

break;

case "tif":

curBitmpap.Save(filename, System.Drawing.Imaging.ImageFormat.Tiff);

break;

case "png":

curBitmpap.Save(filename, System.Drawing.Imaging.ImageFormat.Png);

break;

default:

break;

}

}

}

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

{

this.Close();

}

/// <summary>

/// 直方图均衡化 直方图均衡化就是对图像进行非线性拉伸，重新分配图像像素值，使一定灰度范围内的像素数量大致相同

/// 增大对比度，从而达到图像增强的目的。是图像处理领域中利用图像直方图对对比度进行调整的方法

/// </summary>

/// <param name="srcBmp">原始图像</param>

/// <param name="dstBmp">处理后图像</param>

/// <returns>处理成功 true 失败 false</returns>

public static bool Balance(Bitmap srcBmp, out Bitmap dstBmp)

{

if (srcBmp == null)

{

dstBmp = null;

return false;

}

int[] histogramArrayR = new int[256];//各个灰度级的像素数R

int[] histogramArrayG = new int[256];//各个灰度级的像素数G

int[] histogramArrayB = new int[256];//各个灰度级的像素数B

int[] tempArrayR = new int[256];

int[] tempArrayG = new int[256];

int[] tempArrayB = new int[256];

byte[] pixelMapR = new byte[256];

byte[] pixelMapG = new byte[256];

byte[] pixelMapB = new byte[256];

dstBmp = new Bitmap(srcBmp);

Rectangle rt = new Rectangle(0, 0, srcBmp.Width, srcBmp.Height);

BitmapData bmpData = dstBmp.LockBits(rt, ImageLockMode.ReadWrite, PixelFormat.Format24bppRgb);

unsafe

{

//统计各个灰度级的像素个数

for (int i = 0; i < bmpData.Height; i++)

{

byte\* ptr = (byte\*)bmpData.Scan0 + i \* bmpData.Stride;

for (int j = 0; j < bmpData.Width; j++)

{

histogramArrayB[\*(ptr + j \* 3)]++;

histogramArrayG[\*(ptr + j \* 3 + 1)]++;

histogramArrayR[\*(ptr + j \* 3 + 2)]++;

}

}

//计算各个灰度级的累计分布函数

for (int i = 0; i < 256; i++)

{

if (i != 0)

{

tempArrayB[i] = tempArrayB[i - 1] + histogramArrayB[i];

tempArrayG[i] = tempArrayG[i - 1] + histogramArrayG[i];

tempArrayR[i] = tempArrayR[i - 1] + histogramArrayR[i];

}

else

{

tempArrayB[0] = histogramArrayB[0];

tempArrayG[0] = histogramArrayG[0];

tempArrayR[0] = histogramArrayR[0];

}

//计算累计概率函数，并将值放缩至0~255范围内

pixelMapB[i] = (byte)(255.0 \* tempArrayB[i] / (bmpData.Width \* bmpData.Height) + 0.5);//加0.5为了四舍五入取整

pixelMapG[i] = (byte)(255.0 \* tempArrayG[i] / (bmpData.Width \* bmpData.Height) + 0.5);

pixelMapR[i] = (byte)(255.0 \* tempArrayR[i] / (bmpData.Width \* bmpData.Height) + 0.5);

}

//映射转换

for (int i = 0; i < bmpData.Height; i++)

{

byte\* ptr = (byte\*)bmpData.Scan0 + i \* bmpData.Stride;

for (int j = 0; j < bmpData.Width; j++)

{

\*(ptr + j \* 3) = pixelMapB[\*(ptr + j \* 3)];

\*(ptr + j \* 3 + 1) = pixelMapG[\*(ptr + j \* 3 + 1)];

\*(ptr + j \* 3 + 2) = pixelMapR[\*(ptr + j \* 3 + 2)];

}

}

}

dstBmp.UnlockBits(bmpData);

return true;

}

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

{

Bitmap dstBmp;

Balance(curBitmpap, out dstBmp);

curBitmpap = dstBmp;

Invalidate();

}

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

{

histogram winform = new histogram(curBitmpap);

winform.ShowDialog();

}

}

}

**绘制灰度直方图代码**

namespace DIP\_02

{

public partial class histogram : Form

{

//利用构造函数实现窗体之间的数据传递

public histogram(Bitmap bmp)

{

InitializeComponent();

//把主窗体的图像数据传递给从窗体

bmpHist = bmp;

//灰度级计数

countPixel = new int[256]; //8位可表示256个灰度级

}

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

{

this.Close();

}

//图像数据

private System.Drawing.Bitmap bmpHist;

//灰度等级

private int[] countPixel;

//记录最大的灰度级个数

private int maxPixel;

/// <summary>

/// 计算各个灰度级所具有的像素个数

/// </summary>

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

{

//锁定8位灰度位图

Rectangle rect = new Rectangle(0, 0, bmpHist.Width, bmpHist.Height);

System.Drawing.Imaging.BitmapData bmpData = bmpHist.LockBits(rect,

System.Drawing.Imaging.ImageLockMode.ReadWrite, bmpHist.PixelFormat);

IntPtr ptr = bmpData.Scan0;

int bytes = bmpHist.Width \* bmpHist.Height;

byte[] grayValues = new byte[bytes];

System.Runtime.InteropServices.Marshal.Copy(ptr, grayValues, 0, bytes);//灰度值数据存入grayValues中

byte temp = 0;

maxPixel = 0;

//灰度等级数组清零

Array.Clear(countPixel, 0, 256);

//计算各个灰度级的像素个数

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

{

//灰度级

temp = grayValues[i];

//计数加1

countPixel[temp]++;

if (countPixel[temp] > maxPixel)

{

//找到灰度频率最大的像素数，用于绘制直方图

maxPixel = countPixel[temp];

}

}

//解锁

System.Runtime.InteropServices.Marshal.Copy(grayValues, 0, ptr, bytes);

bmpHist.UnlockBits(bmpData);

}

/// <summary>

/// 绘制直方图

/// </summary>

private void histForm\_Paint(object sender, PaintEventArgs e)

{

//获取Graphics对象

Graphics g = e.Graphics;

//创建一个宽度为1的黑色钢笔

Pen curPen = new Pen(Brushes.Black, 1);

//绘制坐标轴

g.DrawLine(curPen, 50, 240, 320, 240);//横坐标

g.DrawLine(curPen, 50, 240, 50, 30);//纵坐标

//绘制并标识坐标刻度

g.DrawLine(curPen, 100, 240, 100, 242);

g.DrawLine(curPen, 150, 240, 150, 242);

g.DrawLine(curPen, 200, 240, 200, 242);

g.DrawLine(curPen, 250, 240, 250, 242);

g.DrawLine(curPen, 300, 240, 300, 242);

g.DrawString("0", new Font("New Timer", 8), Brushes.Black, new PointF(46, 242));

g.DrawString("50", new Font("New Timer", 8), Brushes.Black, new PointF(92, 242));

g.DrawString("100", new Font("New Timer", 8), Brushes.Black, new PointF(139, 242));

g.DrawString("150", new Font("New Timer", 8), Brushes.Black, new PointF(189, 242));

g.DrawString("200", new Font("New Timer", 8), Brushes.Black, new PointF(239, 242));

g.DrawString("250", new Font("New Timer", 8), Brushes.Black, new PointF(289, 242));

g.DrawLine(curPen, 48, 40, 50, 40);

g.DrawString("0", new Font("New Timer", 8), Brushes.Black, new PointF(34, 234));

g.DrawString(maxPixel.ToString(), new Font("New Timer", 8), Brushes.Black, new PointF(18, 34));

//绘制直方图

double temp = 0;

for (int i = 0; i < 256; i++)

{

//纵坐标长度

temp = 200.0 \* countPixel[i] / maxPixel;

g.DrawLine(curPen, 50 + i, 240, 50 + i, 240 - (int)temp);

}

//释放对象

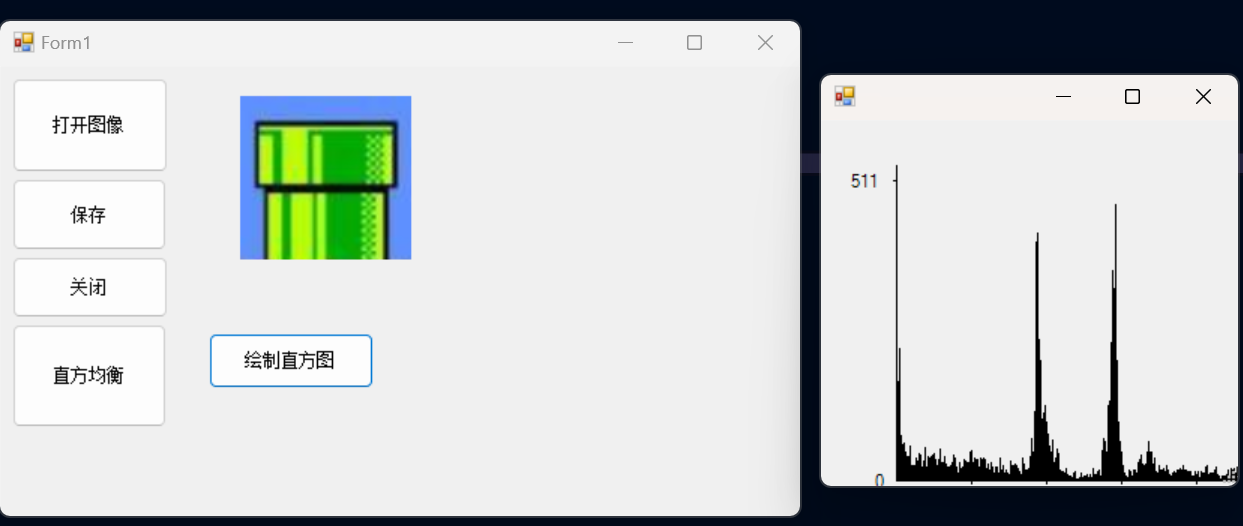
curPen.Dispose();

}

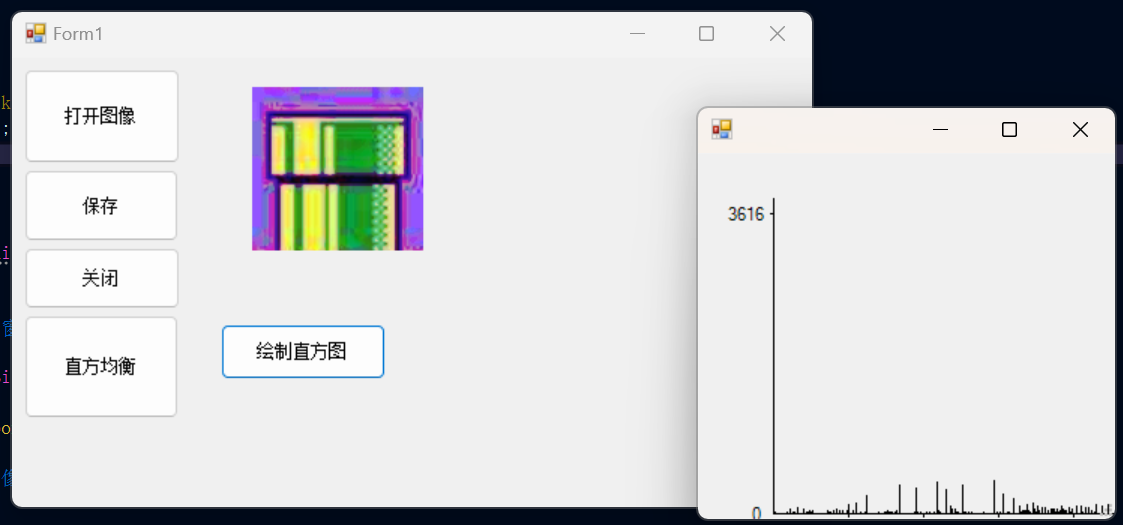
}

}

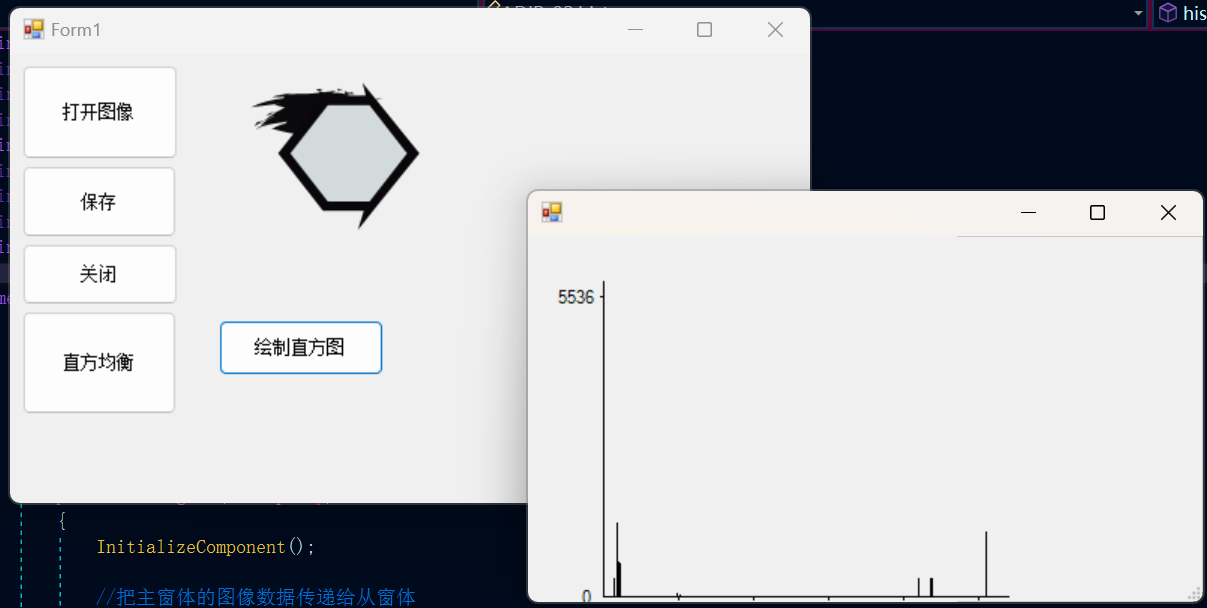
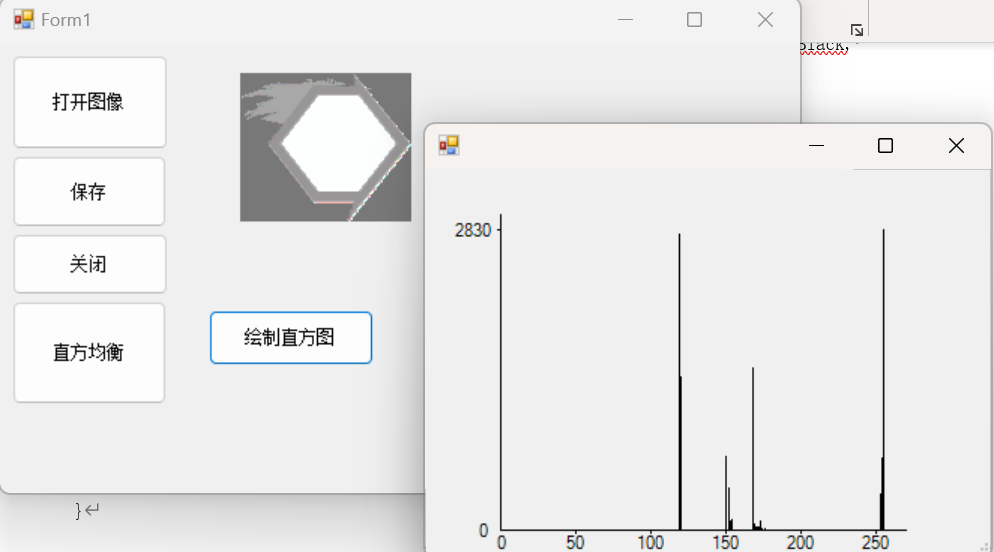
# 实验结果

在最后的窗体程序中，可以选择电脑中的文件打开，进行直方图均衡并绘制灰度直方图。

*直方均衡前的图像及其灰度直方图*

**

*直方均衡后的图像及其灰度直方图*

**

