7　基于梯度算子的边缘检测

7.1　实验目的

使用VS2008开发工具，c#编程语言条件下，实现基于梯度算子的边缘检测的图像算法。

7.2　算法原理

图像分割算法是基于图像灰度值不连续性或相似性，边缘检测是图像分割的一种方法。边缘检测用于标识数字图像中亮度变化明显的点，可以大幅度减少不相关信息，保留图像重要的结构属性。 边缘检测的算法分为两类：基于查找的算法和基于零穿越的算法。 基于查找的方法指通过寻找图像一阶导数的最大和最小值来检测边界。通常将边界定位在梯度方向的方向。基于一阶导数的边缘检测算子包括Roberts算子、Sobel算子、Prewitt算子等，它们都是梯度算子。 基于零穿越的方法指通过寻找图像二阶导数零穿来寻找边界，通常是拉普拉斯过零点或者非线性差分表示过零点。算子主要是高斯－拉普拉斯边缘检测算子。

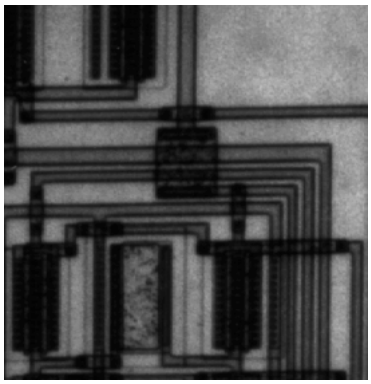

图 7.1: 边缘检测原图

梯度的计算不使用平方运算，经常使用的方法是绝对值梯度，公式如下：

$$\nabla f \approx |G_x| + |G_y|$$

要使用的sobel梯度算子有下面几个模板，分别用于不同方向的梯度检测。

-1	-1	-1
0	0	0
1	1	1

表 7.1: 水平边缘

-1	0	1
-1	0	1
-1	0	1

表 7.2: 垂直边缘

-1	-1	0
-1	0	1
0	1	1

表 7.3: 45度边缘

0	1	1
-1	0	1
-1	-1	0

表 7.4: 135度边缘

7.3 实验过程

1. 新建窗体应用程序。

2. 在项目中添加一个新类DataClass public static class DataClass

```
{
    public static MemoryStream ms_bmp_src, ms_bmp_result, ms_bmp_temp;
    public static Bitmap bp_1;//原始大图
    public static Bitmap bp_2;//模板
    public static IntPtr frm1_wnd_handle;
    public const int THREAD_FINISHED = 0x501;
}
```

3. 在Program.cs文件修改系统启动代码如下

```
static void Main()
{
    Application.EnableVisualStyles();
    Application.SetCompatibleTextRenderingDefault(false);
    Form1 frm1 = new Form1();
    DataClass.frm1_wnd_handle = frm1.Handle;
    DataClass.ms_bmp_src = new MemoryStream(5000000);
    DataClass.ms_bmp_result = new MemoryStream(5000000);
    Application.Run(frm1);
}
```

4. 设计窗体界面，在界面上添加三个图片框，图片框为并排排列，三个按扭竖向排列，可参考最后的运行结果图。

5. 加入全局变量声明

```
[DllImport("User32.dll", EntryPoint = "SendMessage")]//动态链接库引入
private static extern int SendMessage(
IntPtr hWnd, // handle to destination window
int Msg, // message
int wParam, // first message parameter
```

```
int lParam // second message parameter
);
//定义消息常数
public const int TRAN_FINISHED = 0x500;
```

6. 读入图片一代码，读入模板图片代码类似。

```
private void button1_Click(object sender, EventArgs e)
{
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {//选择文件
        DataClass.bp_1 = new Bitmap(openFileDialog1.FileName);
        pictureBox1.Image = DataClass.bp_1;
    }
}
```

7. 编写线程代码Sobel。

```
//Sobel边缘检测
static void Sobel()
{
    //线程流程--图像相关
    if (DataClass.bp_1 != null)
    {
        //准备位图 1 的字节数组
        DataClass.ms_bmp_src.Seek(0, SeekOrigin.Begin);
        DataClass.bp_1.Save(DataClass.ms_bmp_src, System.Drawing.Imaging.ImageFormat.Bmp);
        byte[] buf_ms_src = DataClass.ms_bmp_src.GetBuffer();
        //输出结果的内存区
        DataClass.ms_bmp_result.Seek(0, SeekOrigin.Begin);
        DataClass.bp_1.Save(DataClass.ms_bmp_result, System.Drawing.Imaging.ImageFormat.Bmp);
        byte[] buf_ms_result = DataClass.ms_bmp_result.GetBuffer();
        int src_width=DataClass.bp_1.Width;//位图宽
        int src_height=DataClass.bp_1.Height;//位图高
        byte[] buf_src = new byte[src_width * src_height];
        int[] buf_data = new int[src_width * src_height];
        int line_byte_count, scan_line_len;
        line_byte_count = src_width * 3;//24位需要处理成字节以4整倍数的填充行
        if ((line_byte_count % 4) == 0)
        {
```

```
            scan_line_len = line_byte_count;
        }
        else
        {
            scan_line_len = (line_byte_count / 4) * 4 + 4;
        }
        int [] index_x=new int[]{-1,0,1,
                                 -1,0,1,
                                 -1,0,1};
        int [] index_y=new int[]{-1,-1,-1,
                                 0,0,0,
                                 1,1,1};
        //位图字节顺序转换坐标，位图中是由下而上行扫描
        for (int i_height = 0; i_height < src_height; i_height++)
        {
            for (int i_width = 0; i_width < src_width; i_width++)
            {
        buf_src[src_width * (src_height - 1 - i_height) + i_width] = buf_ms_src[54 + i_height
* scan_line_len + i_width * 3];//获取颜色值
            }
        }//准备位图 1 的字节数组
        int []z=new int[9];
        int d_min=10000, d_max=0;
        for (int i_height = 1; i_height < src_height-1; i_height++)
        {
            for (int i_width = 1; i_width < src_width-1; i_width++)
            {
                int dResult = 0;
                //邻域处理 计算梯度
                for (int i = 0; i < 9; i++)
                {
                    z[i] = (int)buf_src[(i_height + index_y[i]) * src_width + (i_width
+ index_x[i])];
                }
                dResult += Math.Abs(z[6] + z[7] + z[8] - z[0] - z[1] - z[2]);
                buf_data[i_height * src_width + i_width] = dResult;
                if (d_max < dResult)
                {
                    d_max = dResult;
```

```
                    }
                    if (d_min> dResult)
                    {
                        d_min = dResult;
                    }
                }
            }//扫描图像进行滤波操作
            int d_gap;
            d_gap=d_max-d_min;
            //设置输出位图字节值
            for (int i_height = 0; i_height < src_height; i_height++)
            {
                for (int i_width = 0; i_width < src_width; i_width++)
                {
buf_ms_result[54 + (src_height - 1 - i_height) * scan_line_len + i_width * 3] =
(byte)((buf_data[i_height * src_width +i_width] - d_min) * 255 / d_gap);//设置颜色值
buf_ms_result[54 + (src_height - 1 - i_height) * scan_line_len + i_width * 3 + 1] =
(byte)((buf_data[i_height * src_width + i_width] - d_min) * 255 / d_gap);//设置颜色值
buf_ms_result[54 + (src_height - 1 - i_height) * scan_line_len + i_width * 3 + 2] =
(byte)((buf_data[i_height * src_width + i_width] - d_min) * 255 / d_gap);//设置颜色值
                }
            }//


        }
        SendMessage(DataClass.frm1_wnd_handle, DataClass.THREAD_FINISHED, 100, 100);
    }
```

8. 线程启动代码

```
            private void button4_Click(object sender, EventArgs e)
            {
                Thread workThread = new Thread(new ThreadStart(Sobel));
                workThread.IsBackground = true;
                workThread.Start();
            }
```

9. 添加消息重载函数，接收自定义消息

```
protected override void DefWndProc(ref Message m)
{//窗体消息处理重载
```

```
        switch (m.Msg)
        {
            case DataClass.THREAD_FINISHED:
            pictureBox3.Image = (Bitmap)Bitmap.FromStream(DataClass.ms_bmp_result);
                this.Invalidate();
                break;
            default:
                base.DefWndProc(ref m);
                break;
        }
    }
```

10. 添加必要的命名空间，编译项目调试运行程序，读入给定的灰度图像circuit.bmp文件，查看初次运行结果。原程序示例代码中只提供了x方向的梯度分量，请同学根据其它三个梯度分量的领域系数添加另外三个方向梯度值，查看程序执行结果。

    参考结果画面如下：

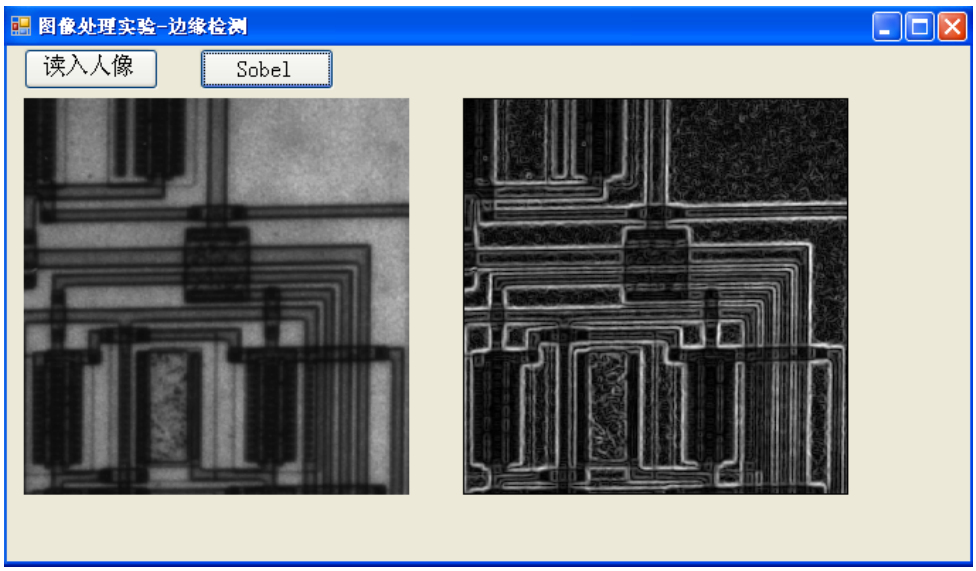

图 7.2: 程序运行结果图

7.4 算法说明

点检测与边缘检测是图像分割的基础，在进行Sobel梯度算子检测边缘时，边缘的像素会比较宽，结果需要通过细化，梯度算子方法还会受到噪声的影响产生间断的现象。

7.5 作业

1. 将原程序完善，将程序源代码以"08班级名_姓名_学号.txt"命名后，把文本形式代码上传到服务器 ftp://172.16.13.252。
2. 补充另外三个梯度分量，完善sobel算法计算。