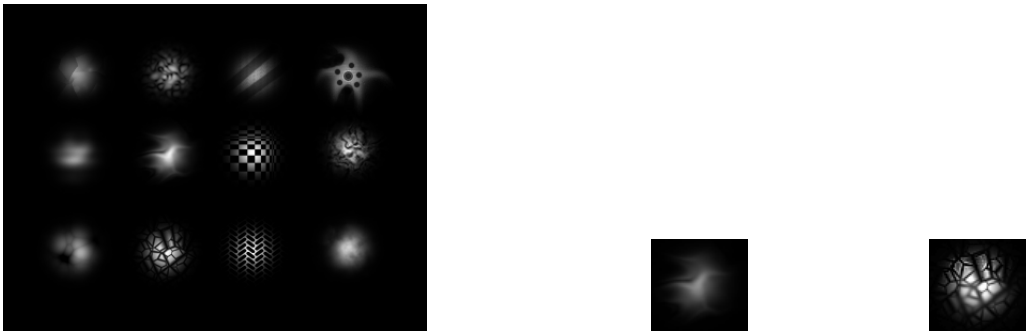
5　基于相关的图像匹配算法

5.1　实验目的

使用VS2008开发工具，c#编程语言条件下，实现基于相关的图像匹配算法的图像操作。

5.2　算法原理

本实验提供三张图片，patterns.bmp包含12种不同的图案模式对应图(a)，pat1.bmp对应图(b)，pat2.bmp对应图(c)，要在(a)中找到(b)和(c)中的子图像的最佳匹配。(b)中的图像对应(a)中第 2 行的第 2 个小图案，但整体亮度较其在(a)中更暗，(c)中图像则与(a)中第3行第 2 个小图案相似，略有细节不同。



a　　b　　c

图 5.1: 基于相关的图像匹配

在一个原始图片中查找最相似的模板图片，通过遍历原始图像素，与模板相素进行相关运算，相关值最大的位置即为匹配的坐标。 相关运算与卷积运算类似，需要对应分量相乘求和，但卷积是对模板进行了180的翻转；在计算过程中对模值进行了归一化，其实际是计算了向量的夹角。

使用的相关计算公式如下:

$$r(x,y)=\frac{\sum_{s=0}^{K}\sum_{t=0}^{J}w(s,t)f(x+s,y+t))}{\left[\sum_{s=0}^{K}\sum_{t=0}^{J}w^2(s,t)\cdot\sum_{s=0}^{K}\sum_{t=0}^{J}f^2(x+s,y+t))\right]^{1/2}}$$

程序是窗体应用程序，使用线程进行运算，线程在运算完毕后，将结果通知给主窗

体。

5.3 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 bool bm_ready;
    public static IntPtr frm1_wnd_handle;
    public const int GRAY_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);
    DataClass.ms_bmp_temp = new MemoryStream(1000000);
    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corr_match。

```
//8位灰度图基于模板的相关运算的线程体
static void corr_match()
{
    DataClass.bm_ready = false;
    //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);
        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];
        byte[] buf_ms_src = DataClass.ms_bmp_src.GetBuffer();
        int line_byte_count, scan_line_len;
        line_byte_count = src_width * 3;//24位需要处理成字节以4为整倍数的填充行
        if ((line_byte_count % 4) == 0)
        {
            scan_line_len = line_byte_count;
        }
        else
        {
```

```
            scan_line_len = (line_byte_count / 4) * 4 + 4;
        }
        //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 temp_width=DataClass.bp_2.Width;
        int temp_height = DataClass.bp_2.Height;
        byte[] buf_ms_mode = new Byte[temp_width * temp_height];
        DataClass.ms_bmp_temp.Seek(0, SeekOrigin.Begin);
        DataClass.bp_2.Save(DataClass.ms_bmp_temp, System.Drawing.Imaging.ImageFormat.Bmp);
        byte[] buf_ms_temp = DataClass.ms_bmp_temp.GetBuffer(); ;
        line_byte_count = temp_width * 3;
        int scan_line_len_temp = 0;
        if ((line_byte_count % 4) == 0)
        {
            scan_line_len_temp = line_byte_count;
        }
        else
        {
            scan_line_len_temp = (line_byte_count / 4) * 4 + 4;
        }
        double dSumT=0.0;
        for(int i=0;i<temp_height;i++)
        {
            for (int j = 0; j < temp_width; j++)
            {
                buf_ms_mode[i * temp_width + j] = buf_ms_temp[54 + (temp_width-
i) * scan_line_len_temp + j * 3];
                dSumT+=(double)(((int)buf_ms_mode[i*temp_width+j])*((int)buf_ms_mode[i*tem
+j]));
            }
        }
        //获取模板的灰度字节数组
```

```
int nMaxX=0,nMaxY=0;
double MaxR=0,R;
double dSumST,dSumS;
//运算求得最大相关值
for(int i_height=0;i_height<src_height-temp_height;i_height++)
{
    for (int i_width = 0; i_width < src_width-temp_width; i_width++)
    {
        dSumST=0;
        dSumS=0;
        for(int m=0;m<temp_height;m++)
        {
            for(int n=0;n<temp_width;n++)
            {
                int nGraySrc=(int)buf_src[(i_height+m)*src_width+i_width
+n];

                int nGrayTmp=(int)buf_ms_mode[(m)*temp_width+n];
                dSumS+=(double)nGraySrc*nGraySrc;
                dSumST+=(double)nGraySrc*nGrayTmp;
            }
        }
        R=dSumST/(Math.Sqrt(dSumS)*Math.Sqrt(dSumT));
        if(R>MaxR)
        {
            MaxR=R;
            nMaxX=i_width;
            nMaxY=i_height;
        }
    }
}//运算求得最大相关值

//输出结果的设置
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();
for (int i = nMaxX; i < nMaxX + temp_width; i++)
{
    buf_ms_result[54+(src_height - nMaxY) * scan_line_len + i*3] = 255;
```

```
                buf_ms_result[54 + (src_height - nMaxY) * scan_line_len + i * 3+1] = 255;
                buf_ms_result[54 + (src_height - nMaxY) * scan_line_len + i * 3+2] = 255;
            }

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

8. 线程启动代码

```
        private void button4_Click(object sender, EventArgs e)
        {
            Thread workThread = new Thread(new ThreadStart(corr_match));
            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.GRAY_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patterns.bmp文件，模板文件pat1.bmp，pat2.bmp，查看程序执行结果。参考结果画面如下：

11. 请同学们自行设计代码在结果图中将原来的直线标识变为方框。

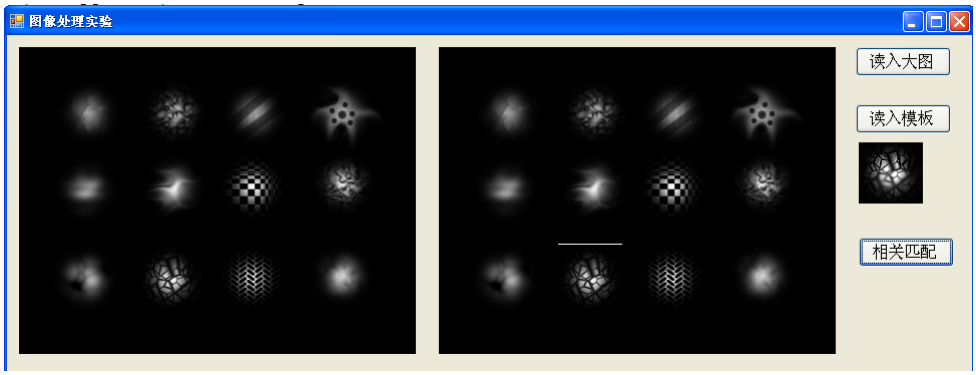


图 5.2: 程序运行结果图

5.4 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. 理解位图文件像素字节的排列规律，字节零填充的意义。