6 使用HSI模型的肤色检测

6.1 实验目的

使用VS2008开发工具，c#编程语言条件下，通过学习HSI图像模型加深理论认识，了解图像处理知识在实际中的应用，根据HSI图像模型实现对彩色图像肤色检测的图像处理操作。

6.2 算法原理

肤色识别是图像特征提取的一种方法，彩色图像最常用RGB分量来表示，而在色彩的识别方面使用HSI模型会更有效。HSI色系是基于色调(H)、饱和度(S)、亮度(I)的颜色空间，下面是RGB颜色空间转换为HSI空间的公式：

$$I = \frac{(R+G+B)}{3}$$

$$S = 1 - \frac{min(R+G+B)}{I}$$

$$H = \begin{cases} \theta & G \geq B \\ 2\pi - \theta & G < B \end{cases}$$

$$\theta = arccos\left\{\frac{[(R-G)+(R-B)]}{2\cdot[(R-G)^2+(R-B)(R-G)]^{1/2}}\right\}$$

HSI色坐标与人眼视觉要素接近，在肤色检测方面方便实用，只要设定适当的阈值，就可以实现对图像中人体肤色象素的识别目标。在本实验中，对像素的的H、S、R、I分量进行肤色提取，程序中认定像素为肤色的阈值如下：

- H值范围为(0 1.6)∪(5.6 π)；
- I > 100 ；
- 0.1<S<0.88；
- R>240；

根据上述模型分析，读取图片文件的像素字节值，进行运算后，对符合肤色范围的像素保留原值，不符合的将像素值设为0，即可产生运算结果图片。

6.3 实验过程

1. 新建窗体应用程序；

2. 添加系统API引用，注意这段代码应该在Form1.cs源文件中的窗体构造函数Form1()后面加入；

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

3. 在窗体上添加两个按钮分别命名为打开图片，肤色查找，两个图片框，一个文件打开对话框控件；

4. 在打开图片点击事件添加如下代码：

```
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;
    }
}
```

5. 在肤色查找按钮点击事件添加如下代码启动工作线程：

```
private void button2_Click(object sender, EventArgs e)
{
    Thread workThread = new Thread(new ThreadStart(skin_mark));
    workThread.IsBackground = true;
    workThread.Start();
}
```

6. 窗体消息处理重载函数：

```
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;
            }
        }
```

7. 工作线程代码：

```
    static void skin_mark()
    {
        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);
            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

            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;
        }
        byte b_val,g_val,r_val;
        double H, S, I;
        double val_R, val_G, val_B,val_min, tem1, angle;
        double I_boarder=100/255.0;
        //RGB-->HSI
        //
        for (int i_height = 0; i_height < src_height; i_height++)
        {
            for (int i_width = 0; i_width < src_width; i_width++)
            {
                //
                b_val =buf_ms_src[54 + i_height * scan_line_len + i_width * 3];
                g_val =buf_ms_src[54 + i_height * scan_line_len + i_width * 3+1];
                r_val =buf_ms_src[54 + i_height * scan_line_len + i_width * 3+2];
                val_B = (double)(b_val / 255.0);
                val_G = (double)(g_val / 255.0);
                val_R = (double)( r_val / 255.0);

                I = (val_R + val_G + val_B) / 3;
                //Math.acos(double) 0 ≤ θ ≤ π 其中 -1 ≤d≤ 1
                //H=Math.acos(double) B<=G H=2 π -Math.acos(double) B>G H 0 1.6 5.6 2pi
                tem1 = ((val_R - val_G) + (val_R - val_B)) / 2*( Math.Sqrt((val_R -
val_G) * (val_R - val_G) + (val_R - val_G) * (val_G - val_B)));
                ;

                //angle = Math.Acos(tem1);
                //if (b_val <= g_val)
                //{
                // H = angle;
                //}
                //else {
                // angle=2*Math.PI-angle;
                // H
                //}
                //I = (R + G + B) / 3; I >100/255
                val_min=Math.Min(Math.Min(val_B,val_G),val_R);
                S=1-val_min*3/(val_R + val_G + val_B);
```

```
                    //S=1-((min(R,G,B))*3/(R+G+B)) 0.1<S<0.88
                    //R>240
                    if ((I > I_boarder) && (S < 0.88) && (S > 0.1) && (r_val>200) &&
(tem1 > Math.Cos(1.8)) && (tem1 <= 1.0))
                    {
                        //得出结果后设置像素值，这是皮肤像素

                    }else
                    {
                        buf_ms_result[54 + i_height * scan_line_len + i_width * 3] = (byte)0;
                        buf_ms_result[54 + i_height * scan_line_len + i_width * 3 + 1] =
(byte)0;
                        buf_ms_result[54 + i_height * scan_line_len + i_width * 3 + 2] =
(byte)0;
                    }

                }
            }//根据HSI模型运算每个像素是否是肤色
        }
        SendMessage(DataClass.frm1_wnd_handle, DataClass.GRAY_FINISHED, 100, 100);
    }
```

8. 添加必要命名空间，编译调试程序，读入给定的人像图形，利用代码试验处理结果；参考结果如图：

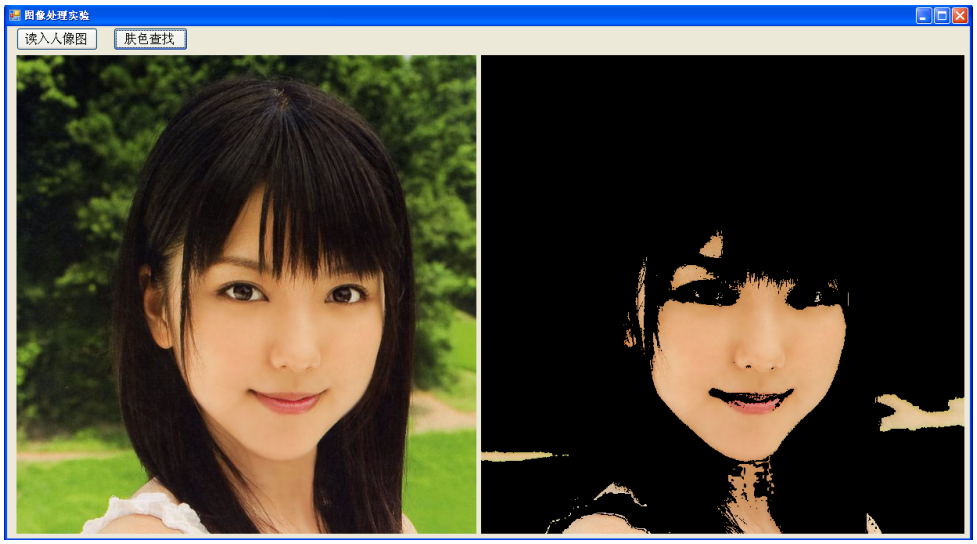

图 6.1: 程序运行结果图

9. 分别读入不同的人物图像，查看运行结果，对算法有效性进行分析。

6.4 结果说明

虽然基于像素的运算能初步进行肤色识别，也有其它颜色空间的识别模型，由于各种颜色空间表示方法能够进行等价转换，其方法的本质一样。人的肤色变化范围较大，也容易受到环境光线的影响，而阈值的选择是出于人的经验或者是图片库的训练，这种方法的有效性也受到一定局限。单独的肤色检测效用离实际应用有差距，在实际中往往会结合形态学原理等其它理论进行综合评定，比如包含人眼定位及脸型比例的人脸识别将会更有效一些。