重 庆 交 通 大 学

学生实验报告

课 程 名 称：数 字 图 像 处 理 .

开 课实验室：软 件 实 验 中 心 .

学 院：信息学院年级物联网工程专业 2 班

学 生 姓 名：李骏飞学 号 632109160602

指 导 教 师：蓝 章 礼 .

开 课 时 间：2023 至 2024 学年第 二 学期

成 绩	
教师签名	

实验项目名称		第二周实验作业			
姓名	李骏飞	学号	**632109160602**	实验日期	**2024.3.15**
教师评阅： 1:实验目的明确□A□B□C□D 2:内容与原理□A□B□C□D 3:实验报告规范□A□B□C□D 4:实验主要代码与效果展示□A□B□C□D 5:实验分析总结全面□A□B□C□D					
实验记录					

1. 实验目的

完成图像的**灰度化**、**二值化**、**亮度**调整实验。

基本要求：输入彩色图像，通过自己设计的算法代码编写，实现输入图像的灰度化、**固定阈值的二值化**，并能进行**亮度**调整。

拓展要求：在基本要求的基础上，实现**通道提取**，**可变化阈值的二值化**，**自适应二值化**，能进行**对比度**、**饱和度**的调整。

答案要有算法描述，核心代码，完成图片的效果。

2. 实验主要内容及原理

（1）图像灰度化：

在图像处理中，图像灰度化是将彩色图像转换为灰度图像的过程。灰度图像仅包含亮度信息，而不包含颜色信息。C#中可以通过以下原理实现图像的灰度化：

图像灰度化的实验原理基于人眼对颜色和亮度的感知差异。人眼对亮度的感知更为敏感，而对颜色的感知相对较弱。因此，将彩色图像转换为灰度图像可以简化图像处理任务，并减少计算复杂度。

在灰度化过程中，常见的方法是将彩色图像的每个像素的红、绿、蓝（RGB）分量的值进行加权平均，得到一个灰度值。常用的加权平均方法是将红、绿、蓝通道的权重分别设置为 0.2989、0.5870 和 0.1140，这些权重是根据亮度感知的心理学模型得出的。下面是 RGB 转化为灰度的公式：

$$Y = 0.299\,R + 0.587\,G + 0.114\,B$$

（2）图像二值化：

a:固定阈值的二值化：

固定阈值二值化是最简单的二值化方法，它使用一个预先设定的阈值来将灰度图像转换为二值图像。具体步骤如下：

1. 将灰度图像的每个像素的灰度值与设定的阈值进行比较。
2. 如果像素的灰度值大于阈值，则将像素设置为白色（255），否则将像素设置为黑色（0）。
3. 重复上述操作,处理图像的每个像素,直到所有像素都被处理完毕。

b:可变化阈值的二值化：

可变阈值二值化是一种根据图像局部区域的灰度特性来自适应地确定阈值的二值化方法。具体步骤如下：

1. 将灰度图像的每个像素的灰度值与其周围像素的灰度值进行比较。
2. 根据局部区域的灰度特性计算出一个适用于当前像素的阈值。
3. 将当前像素的灰度值与该自适应阈值进行比较。
4. 如果像素的灰度值大于自适应阈值，则将像素设置为白色（255），否则将像素设置为黑色（0）。
5. 重复上述操作,处理图像的每个像素,直到所有像素都被处理完毕。

c:自适应二值化

自适应二值化是一种根据图像局部区域的灰度特性来自动确定阈值的二值化方法。具体步骤如下：

1. 将灰度图像分割成多个局部区域（例如，固定大小的小块或滑动窗口）。
2. 对每个局部区域内的像素进行阈值计算。
3. 根据局部区域的灰度特性计算出一个适用于该局部区域的阈值。
4. 将局部区域内的像素根据自适应阈值进行二值化。
5. 重复上述操作，处理图像的每个局部区域，直到整个图像都被处理完毕。

（3）图像 HSI 调整

RGB 图像转换为 HSI 图像。这可以通过以下公式进行转换：

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta & \text{if } B > G \end{cases} \qquad (6.2\text{-}2)$$

with†

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

The saturation component is given by

$$S = 1 - \frac{3}{(R+G+B)}[\min(R,G,B)] \qquad (6.2\text{-}3)$$

Finally, the intensity component is given by

$$I = \frac{1}{3}(R+G+B) \qquad (6.2\text{-}4)$$

（4）图像亮度调整

对于每个像素的 RGB 分量值，可以通过增加或减小其值来调整像素的亮度。可以使用以下公式来进行亮度调节

三、实验环境

Windows11

Visual Studio2021

C#语言

四、实验主要代码与效果展示

图像灰度化

➢ 算法描述：

该段是图片灰度化按钮的点击事件，首先通过 judge_pb_img_exit 方法判断图片框中是否存在图像，当存在图像时，就创建两个 Bitmap 对象，bt 用于存储原始图像，bt1 用于存储转换后的灰度图像。还创建了一个 Color 对象 color，用于获取每个像素的颜色信息。

然后通过两次 for 循环，根据每个像素的 RGB 值计算灰度值，并利用该灰度值创建新的颜色，然后使用 SetPixel 方法将新的颜色赋值给 bt1 中对应位置的像素，从而将原始图像转换为灰度图像。

最后，通过调用 pictureBox_show.Refresh()方法刷新图片框，确保更新后的图像显示出来。然后将转换后的灰度图像 bt1 赋值给 pictureBox_show.Image，从而在图片框中显示灰度图像。

```
//图片灰度化按钮? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
private void btn_gray_Click(object sender, EventArgs e)
{
    if (!judge_pb_img_exit()) return;
    Bitmap bt = new Bitmap(original_image.Image);
    Bitmap bt1 = new Bitmap(pictureBox_show.Image); //定义并初始化两个位图对象
    Color color = new Color();//定义一个颜色对象
    for (int i = 0; i < bt.Width; i++)
    {
        for (int j = 0; j < bt.Height; j++)
        {
color = bt.GetPixel(i, j); //遍历整张图像，获取每个像素的色彩信息
            //根据 GRB 的不同的权值计算每个像素点的亮度，利用该亮度作为灰度图像中每个像素的灰度值
            int n = (int)((color.G * 59 + color.R * 30 + color.B * 11) / 100);
            bt1.SetPixel(i, j, Color.FromArgb(n, n, n)); //给该像素的每种色彩分量均赋予相同的灰度值，完成灰度图像的转换
        }
    }
    pictureBox_show.Refresh();//刷新图片框
    pictureBox_show.Image = bt1;

}

//判断 picturebox 是否有图片

public bool judge_pb_img_exit()
```

```
{
    if (pictureBox_show.Image == null)//判断图片框是否有图片，如无图片，则给出错误信息
    {
        MessageBox.Show("错误，没有导入图片");
        return false;
    }
    return true;
}
```

➢ **演示效果：**

没有图片弹窗提示：

图像的二值化

图像二值化处理需要确定一个阈值，这里大于阈值的用白色显示，小于阈值的用黑色显示。我采用了滑动条的形式实现了**固定阈值**、**可变化阈值**以及**自适应二值化**操作。

首先判断是否存在图片，然后根据滑动条的值选择二值化的方式，如果滑动条为 0，则调用 **btn_binary_Click_by_default 方法**，利用类判别分析方法根据待处理图像的特点自动计算一个适当的阈值进行判断和二值化；如果滑动条有值，则调用 **btn_binary_Click_by_trackBar_brightness_value 方法**根据滑块的值进行二值化处理：

```
1.          //图像二值化
2.          private void btnBinaryadjust_Click(object sender, EventArgs e)
3.          {
4.              if (!judge_pb_img_exit()) return;
5.              //如果滑块为0，利用类判别分析方法根据待处理图像的特点自动计算一个适当的阈值
    进行判断和二值化。
6.              if (trackBar_binary.Value == 0)
7.              {
8.                  btn_binary_Click_by_default(sender, e);
9.              }
10.             else//否则根据滑块的值进行二值化
11.             {
12.                 btn_binary_Click_by_trackBar_brightness_value(sender, e);
13.             }
14.         }
```

- **算法描述：**

a:自适应阈值的二值化：

先初始化一些变量和数组，用于存储图像处理过程中的中间结果，像灰度值、灰度类均值、直方图和函数值等。

然后，通过遍历图像的像素点，获取每个像素点的灰度值，并将其存储在 array 数组中。并且统计每个灰度值在整个图像中出现的次数，并将结果存储在 tt 数组中，以便后续计算灰度值占比。

根据灰度值出现的次数，计算每个灰度值在整个图像中的占比，并将结果存储在 p 数组中。同时，根据灰度值和其对应的占比，计算图像的灰度均值 u。

根据灰度值和其对应的占比，计算灰度类均值 uu 和直方图和 w。同时，根据这两个值计算每个灰度值的函数值 b，以便寻找函数值最大的阈值。

通过遍历函数值数组 b，找到函数值最大的变量，并记录其对应的灰度值 maxb。这个灰度值将作为阈值。

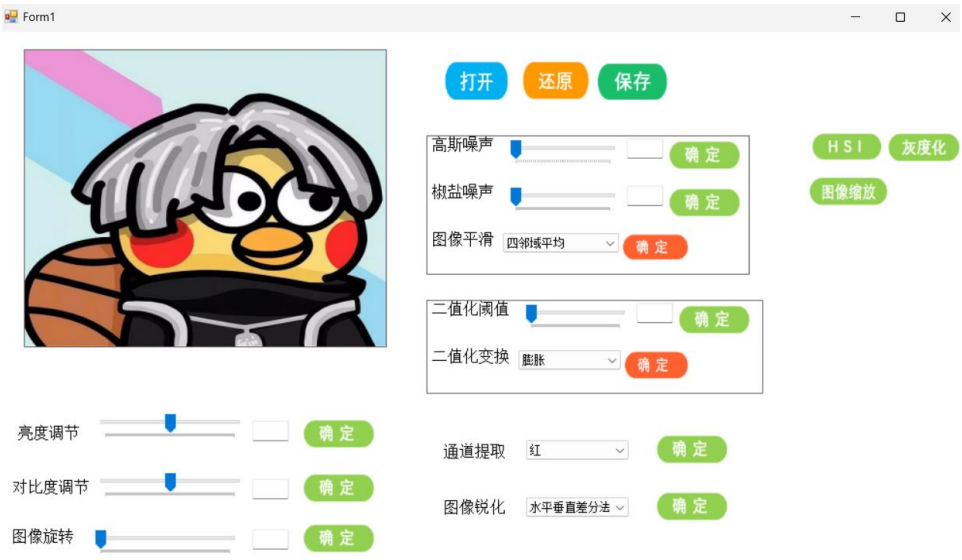
最后，根据阈值对图像进行二值化处理。遍历图像的像素点，将灰度值大于阈值的像素设为黑色，小于等于阈值的像素设为白色。再刷新显示区域并将二值化后的图像显示出来。

```
1.  //滑块为0，利用类判别分析方法根据待处理图像的特点自动计算一个适当的阈值进行判断和二值化。
2.  public void btn_binary_Click_by_default(object sender, EventArgs e)
3.  {
4.      //用于存储该像素的灰度值
5.      int[,] array = new int[1000, 1000];
6.      int[] hd = new int[350];
7.      //用于存储某灰度值在整个图像中占的百分比
8.      double[] p = new double[350];
9.      //用于存储灰度类均值
10.     double[] uu = new double[350];
11.     //用于存储某灰度值的个数
12.     double[] tt = new double[350];
13.     //用于存储直方图之和
14.     double[] w = new double[350];
15.     //函数值
16.     double[] b = new double[350];
17.     //函数值最大值
18.     double max;
19.     double u = 0;
20.     //使得函数值最大的那个变量
21.     double maxb = 0;
22.     int r;
23.     Color cc1 = Color.FromArgb(255, 255, 255);
24.     Color cc2 = Color.FromArgb(0, 0, 0);
25.     Color c = new Color();
26.     //Bitmap box1 = new Bitmap(pictureBox_show.Image);
```

```
    Bitmap box1 = new Bitmap(original_image.Image);
    //获取每个像素的灰度值
    for (int i = 1; i < pictureBox_show.Image.Width - 1; i++)
    {
        for (int j = 1; j < pictureBox_show.Image.Height - 1; j++)
        {
            c = box1.GetPixel(i, j);
            r = c.R;
            array[i, j] = r;
        }
    }
    //灰度值在 0-256 之间变换，再次扫描图像，把相同灰度值的累加起来
    for (int i = 1; i < pictureBox_show.Image.Width; i++)
    {
        for (int j = 1; j < pictureBox_show.Image.Height; j++)
        {
            for (int k = 0; k < 255; k++)
            {
                if (array[i, j] == k)
                {
                    tt[k] = tt[k] + 1;
                }
            }
        }
    }
    //求出图像里含有的各个灰度值所占的百分比
    for (int m = 0; m < 255; m++)
    {
        p[m] = tt[m] / (pictureBox_show.Image.Width * pictureBox_show.Image.Height
);
    }
    //求灰度均值
    for (int n = 1; n < 256; n++)
    {
        u = u + (n - 1) * p[n];
    }
    //求灰度类均值和直方图和
    for (int i = 1; i < 256; i++)
    {
        uu[i] = uu[i - 1] + (i - 1) * p[i];
        w[i] = w[i - 1] + p[i];
        if (w[i] * (1 - w[i - 1]) != 0)
        {
            b[i] = ((u * w[i] - uu[i]) * (u * w[i] - uu[i])) / (w[i] * (1 - w[i]))
;
        }
    }
    //初始化函数最大值
    max = b[0];
    //求出使函数达到最大值的变量
    for (int i = 0; i < 255; i++)
    {
        if (b[i] >= max)
        {
            max = b[i];
        }
```

```
81.     }
82.     for (int j = 0; j < 255; j++)
83.     {
84.         if (b[j] == max)
85.             maxb = j;
86.     }
87.     //最佳阈值就是 maxb-1，然后根据阈值对灰度图像进行二值化处理
88.     for (int i = 1; i < pictureBox_show.Image.Width; i++)
89.     {
90.         for (int j = 1; j < pictureBox_show.Image.Height; j++)
91.         {
92.             c = box1.GetPixel(i, j);
93.             r = c.R;
94.             if (r > (maxb - 1))
95.                 box1.SetPixel(i, j, cc2);
96.             else
97.                 box1.SetPixel(i, j, cc1);
98.         }
99.         pictureBox_show.Refresh();
100.          pictureBox_show.Image = box1;
101.      }
102. }
```

➢ 演示效果：



b:可变化/固定阈值的二值化：

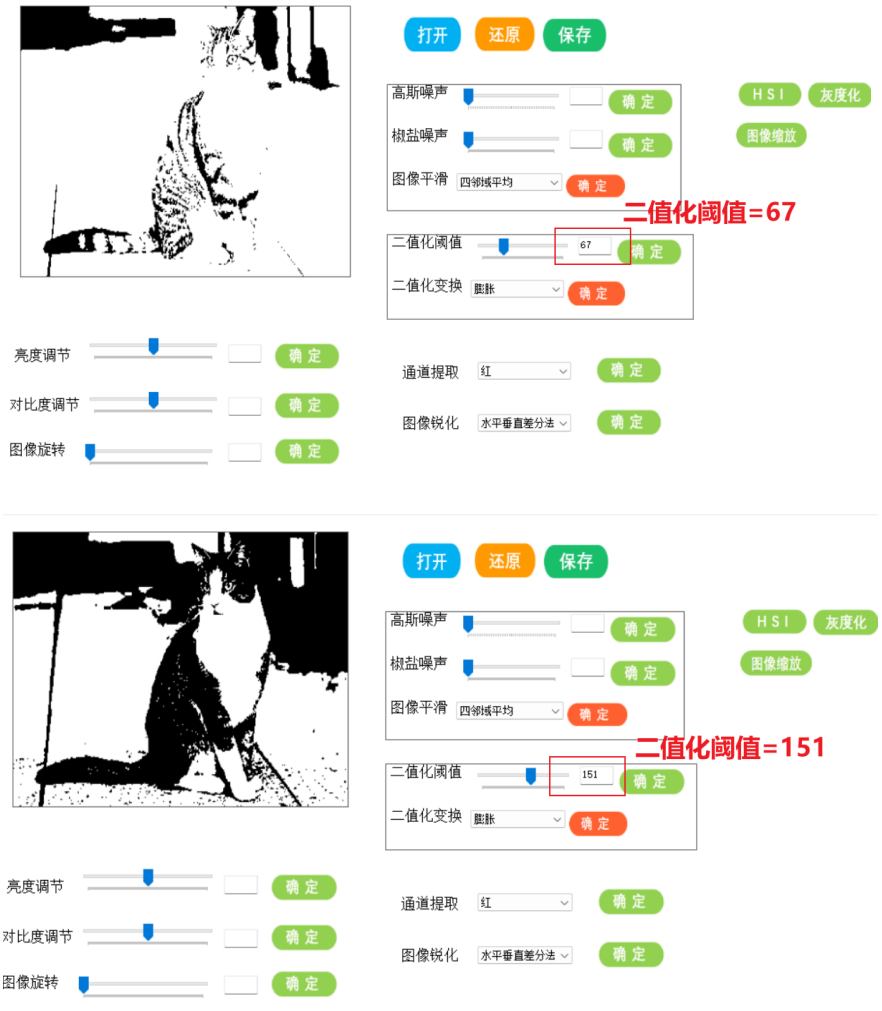
通过定义一个整型的二值化阈值，获取滑动条的值，然后使用嵌套的 for 循环遍历图像的每个像素点。外层循环迭代图像的宽度，内层循环迭代图像的高度。在循环中，获取当前像素点的颜色信息，并将红、绿、蓝分量分别存储在 Red、Green、Blue 变量中，然后根据亮度的计算公式 Y = 0.59 * Red + 0.3 * Green + 0.11 * Blue 对各个分量进行加权求和。

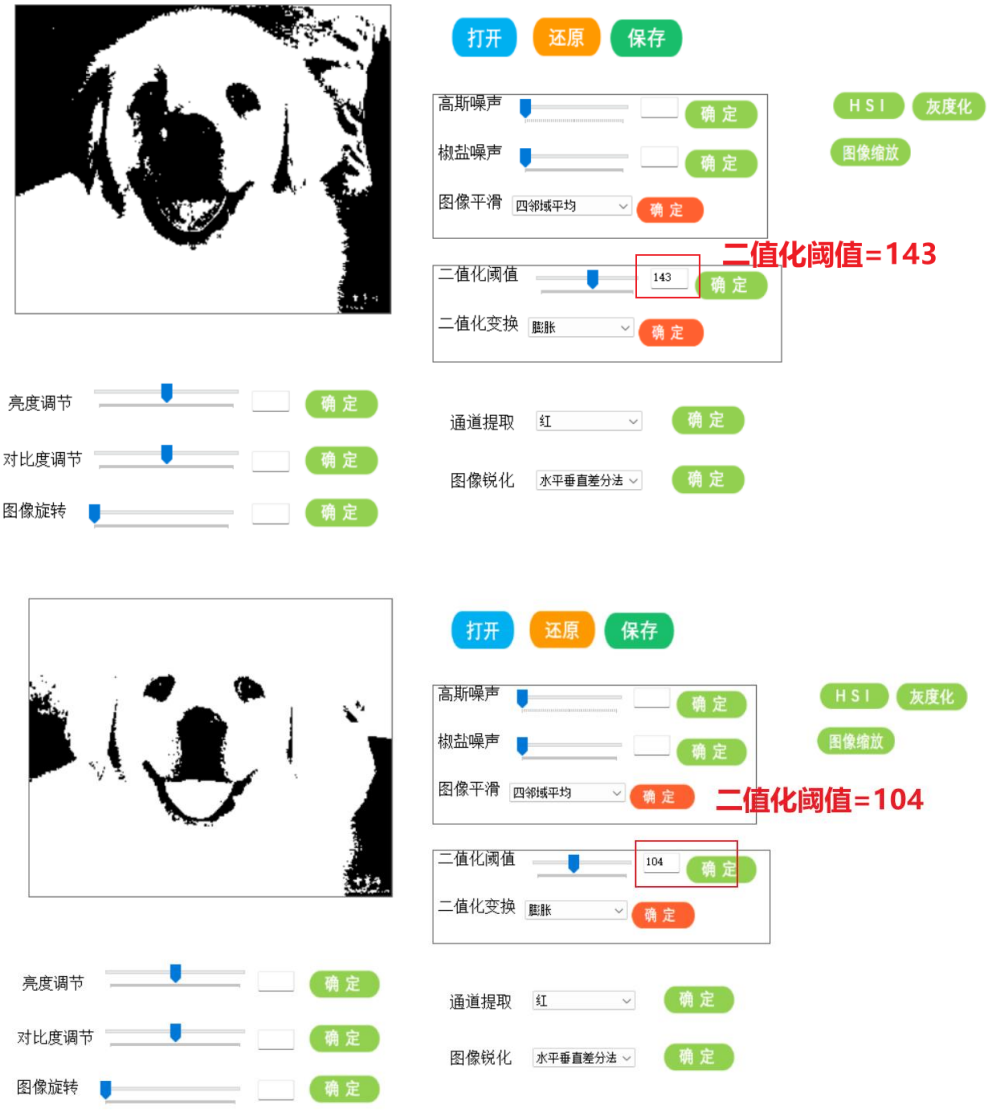
最后根据当前像素点的亮度和二值化阈值进行判断。如果当前像素点的亮度大于阈值 brightThreshole，则将该像素设为白色（RGB 值为 255）；如果当前像素点的亮度小于等于阈值 brightThreshole，则将该像素设为黑色（RGB 值为 0）。

```
1.  //根据滑块的值进行二值化处理
2.  public void btn_binary_Click_by_trackBar_brightness_value(object sender, EventArgs
     e)
3.  {
4.      Color colorOrigin = new Color();//定义一个色彩变量对象
5.      double Red, Green, Blue, Y;//定义红绿蓝三色和亮度
6.      Bitmap Bmp1 = new Bitmap(original_image.Image);//定义一个位图文件并将图片框内的图
    像赋值给它
7.      int brightThreshole = new int();//定义整型的二值化阈值
8.      brightThreshole = trackBar_binary.Value;//根据滑动条的大小 给该阈值赋值，也可以直
    接赋值
9.      for (int i = 0; i < pictureBox_show.Image.Width; i++)
10.     {
11.         for (int j = 0; j < pictureBox_show.Image.Height; j++)//循环处理图像中的每一
    个像素点
12.         {
13.             colorOrigin = Bmp1.GetPixel(i, j);//获取当前像素点的色彩信息
14.             Red = colorOrigin.R; Green = colorOrigin.G; Blue = colorOrigin.B;//获
```

```
    取当前像素点的红绿蓝分量
15.             Y = 0.59 * Red + 0.3 * Green + 0.11 * Blue;//计算当前像素点的亮度
16.             if (Y > brightThreshole)//如果当前像素点的亮度大于指定阈值
17.             {
18.                 Color ColorProcessed = Color.FromArgb(255, 255, 255); /*那么定义一
    个纯白
19.                 的色彩变量，即各分量均为 255*/
20.                 Bmp1.SetPixel(i, j, ColorProcessed); //将白色变量赋给当前像素点
21.             }
22.             if (Y <= brightThreshole) //如果当前像素点的亮度小于指定阈值
23.             {
24.                 Color ColorProcessed = Color.FromArgb(0, 0, 0); /*//那么定义一个纯
    黑的色彩
25.                 变量，即各分量均为 0*/
26.                 Bmp1.SetPixel(i, j, ColorProcessed); //将黑色变量赋给当前像素点
27.             }
28.         }
29.         pictureBox_show.Refresh();//刷新图片框
30.         pictureBox_show.Image = Bmp1; //将重新生成的图片赋值给图片框
31.     }
32. }
```

➢ 演示效果：





通道提取：

➢ 算法描述：

根据用户选择的颜色通道（红、绿、蓝），将图像中对应通道的分量提取出来，并将其他通道置为 0，从而得到只包含指定颜色通道的图像。

先根据用户选择的颜色通道，确定要提取的通道的位置 pos。通过判断 comboBox_colorCollect 控件中选择的文本来确定，如果选择的是"红"则 pos 为 0，如果选择的是"绿"则 pos 为 1，否则 pos 为 2，即提取蓝色通道。然后定义 ColorOrigin（用于存储像素点的颜色信息）、红绿蓝三色分量 Red、Green、Blue、亮度 Y、Bmp1（用于存储图像）。

使用嵌套的 for 循环遍历图像的每个像素点，在循环中，创建一个长度为 3 的整型数组 arr，用于存储当前像素点的红、绿、蓝分量，获取当前像素点的色彩信息，并将红、绿、蓝分量分别存储在 arr 数组中。然后根据用户选择的颜色通道位置 pos，将 arr 数组中对应通道的分量作为灰度值，创建一个新的颜色对象 ColorProcessed。这里使用 Color.FromArgb() 方法创建一个颜色对象，将红、绿、蓝分量都设为对应通道的值，从而得到只包含指定颜色通道的灰度值，将新的颜色对象 ColorProcessed 赋给当前像素点，实现对图像的颜色通道提取。

```
//颜色通道选取确定按钮? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
private void btn_colorCollect_Click(object sender, EventArgs e)
{
    if (!judge_pb_img_exit()) return;
    int pos;
    if (comboBox_colorCollect.Text.ToString().Equals("红"))
    {
        pos = 0;
    }
    else if (comboBox_colorCollect.Text.ToString().Equals("绿"))
    {
        pos = 1;
    }
    else//蓝
    {
        pos = 2;
    }

    Color ColorOrigin = new Color();//定义一个色彩变量对象
    double Red, Green, Blue, Y; //定义红、绿、蓝三色和亮度
    Bitmap Bmp1 = new Bitmap(original_image.Image); //定义一个位图文件并将图片框内的图像赋值给它
    for (int i = 0; i <= pictureBox_show.Image.Width - 1; i++)
    {
        for (int j = 0; j <= pictureBox_show.Image.Height - 1; j++)//循环处理图像中的每一个像素点
        {
            int[] arr = new int[3];
            ColorOrigin = Bmp1.GetPixel(i, j); //获取当前像素点的色彩信息
            arr[0] = (int)ColorOrigin.R;//获取当前像素点的红色分量
            arr[1] = (int)ColorOrigin.G;//获取当前像素点的绿色分量
            arr[2] = (int)ColorOrigin.B;//获取当前像素点的蓝色分量

            //Red = ColorOrigin.R; //获取当前像素点的红色分量
            //Green = ColorOrigin.G; //获取当前像素点的绿色分量
            //Blue = ColorOrigin.B; //获取当前像素点的蓝色分量
            Color ColorProcessed = Color.FromArgb(arr[pos], arr[pos], arr[pos]); //用红色分量作为图像的灰度，也可用绿色或蓝色。
            Bmp1.SetPixel(i, j, ColorProcessed); //将新的灰度值赋给当前像素点
        }
```

```
38.         pictureBox_show.Refresh();//刷新图片框
39.         pictureBox_show.Image = Bmp1; //将重新生成的图片赋值给图片框
40.     }
41.     //MessageBox.Show(comboBox_colorCollect.Text.ToString());
42. }
```

➢ **实现效果：**

原始图像：

当提取某一通道时，最后的处理我是将某一颜色分量作为图像的灰度并赋值给像素点，如果某颜色分量越大，那么最后的新的像素点肉眼上越白。

提取红色通道：

提取绿色通道：

提取蓝色通道：

亮度调整

➢ 算法描述：

图像的亮度调整就是修改像素分量的值使得其根据调节值改变图像的亮度，判断各颜色分量的值是否超过各分量允许的范围，如果大于 255 则只能等于 255，如果小于 0 则只能等于 0，利用调整后的值生成新的颜

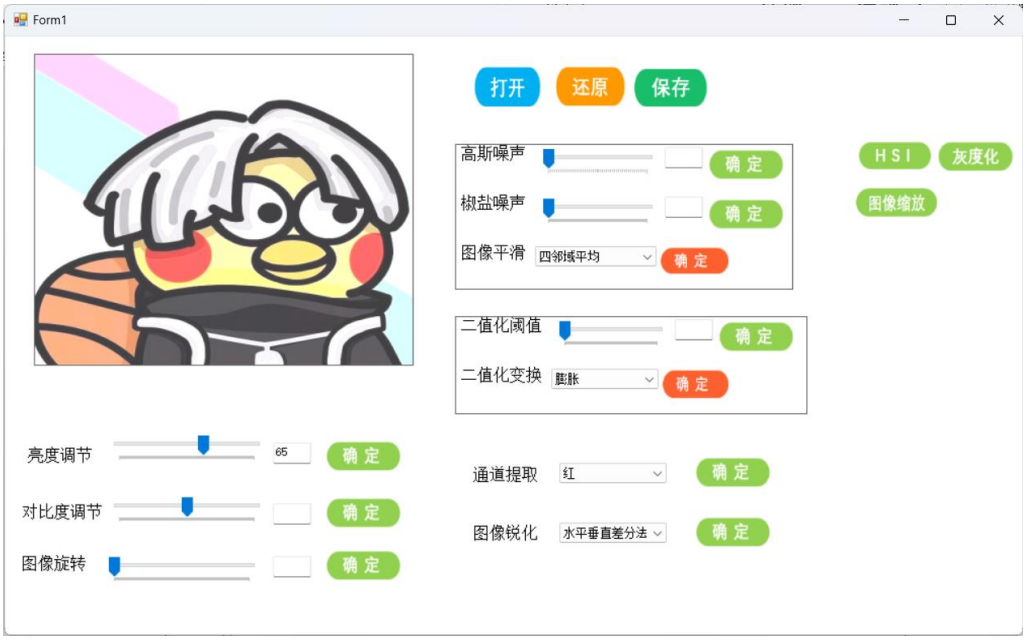
色对象。

```
//亮度调节的确认按钮的点击事件? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
private void btnBrightnessadjust_Click(object sender, EventArgs e)
{//亮度调节就是修改像素分量的值使得其根据调节值改变图像的亮度
    if (!judge_pb_img_exit()) return;

    int value;
    //为 0 表示没有进行操作
    if (textBox_showBrightness.Text.Equals(""))
    {
        return;
    }
    else
    {
        value = int.Parse(textBox_showBrightness.Text); /*将文本框中的数字读出用于作为亮度改变的依据，该值
    来源于滑杆的值，在文本框中其属性为文本类型，因此需要转换为整数类型*/
    }

    Bitmap bt = new Bitmap(original_image.Image); //声明两个位图对象并用图片框中的图片初始化它
    Bitmap bt1 = new Bitmap(original_image.Image);
    int r, g, b; //定义三个整形对象用于存储红、绿、蓝三色的信息
    //逐个扫描原始图片的像素
    for (int i = 0; i < bt1.Width; i++)
    {
        for (int j = 0; j < bt1.Height; j++)
        {
            //获取位于（i,j）坐标的像素然后提取 RGB 分量
            Color color = bt.GetPixel(i, j);
            r = color.R;
            g = color.G;
            b = color.B;
            //对 RGB 分量进行增加或删除，增加量为滑杆滑动的量
            r += value;
            g += value;
            b += value;
            //判断各颜色分量的值是否超过各分量允许的范围，如果大于 255 则只能等于 255
            if (r > 255) r = 255; if (r < 0) r = 0;//如果小于 0 则只能等于 0
            if (g > 255) g = 255; if (g < 0) g = 0;
            if (b > 255) b = 255; if (b < 0) b = 0;
            Color c1 = Color.FromArgb(r, g, b);//利用调整后的值生成新的颜色对象
            bt1.SetPixel(i, j, c1);//把新的颜色 c1 赋值给 bt1 位图对象在坐标为(i,j)的像素
        }
        //每处理一列就进行刷新这样可以动态的显示效果
        pictureBox_show.Refresh();
        //把 bt1 位图对象赋值给图像框
        pictureBox_show.Image = bt1;
    }
}
```

➢ 实现效果：



HSI 调整

➢ 算法描述：

创建一个 Form_HSI 窗体对象，并使用 ShowDialog 方法显示调节窗体。在调节窗体上生成三个滑动条，用于调整色调（H）、饱和度（S）和强度（I）。通过 form_HSI.valueH、form_HSI.valueS 和 form_HSI.valueI 获取滑动条的值。根据颜色空间转换公式，根据 H、S、I 的调节值，对图像的像素进行处理。

根据 H 的值进行分段处理：

如果 H 在 0 到 120 之间，对每个像素进行如下处理：获取像素的原始 RGB 分量值。根据转换公式，计算调整后的 RGB 分量值，其中色调 H 对应的是红色通道。对调整后的 RGB 分量值进行范围判断，确保它们在 0 到 255 之间。将调整后的 RGB 分量值设置回 bt2 中对应的像素。如果 H 在 120 到 240 之间，对每个像素进行类似的处理，但是色调 H 对应的是绿色通道。如果 H 在 240 到 360 之间，对每个像素进行类似的处理，但是色调 H 对应的是蓝色通道。

```
//点击按钮打开 HSI 调节? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ? ?
private void btn_hsi_Click(object sender, EventArgs e)
{
    if (!judge_pb_img_exit()) return;

    //声明两个位图对象并用图片框中的图片初始化它
    Bitmap bt1 = new Bitmap(original_image.Image);
    Bitmap bt2 = new Bitmap(pictureBox_show.Image);
    Color color = new Color();
    //新建一个小窗体用于调节各分量
    Form_HSI form_HSI = new Form_HSI();
    form_HSI.ShowDialog();
    int H = form_HSI.valueH;
    int S = form_HSI.valueS;
    int I = form_HSI.valueI; //在调节窗体上生成三个滑动条，分别用于调整色调、饱和度和强度
    if (H == 0 && S == 0 && I == 0) return;
    int Red, Green, Blue;
    double Hudu = Math.PI / 180;
    if (H >= 0 && H <= 120)
    {
        for (int i = 0; i < bt1.Width; i++)
        {
            for (int j = 0; j < bt1.Height; j++)
            {
                int R, G, B;
                color = bt1.GetPixel(i, j);
                R = color.R;
                G = color.G;
                B = color.B;
                Blue = B + I * (I - S); //根据转换公式，在原各 RGB 分量基础上加上调整的值
                Red = (int)(R + I * (1 + (S * Math.Cos(H * Hudu) / (Math.Cos((60 - H) *
               Hudu)))));
                Green = G + 3 * I - (Blue + Red);
                if (Red > 255) Red = 255; //判断是否超出各分量允许的范围，如果大于 255 则只能等于 255
                if (Red < 0) Red = 0; //如果小于 0 则只能等于 0
                if (Green > 255) Green = 255;
```

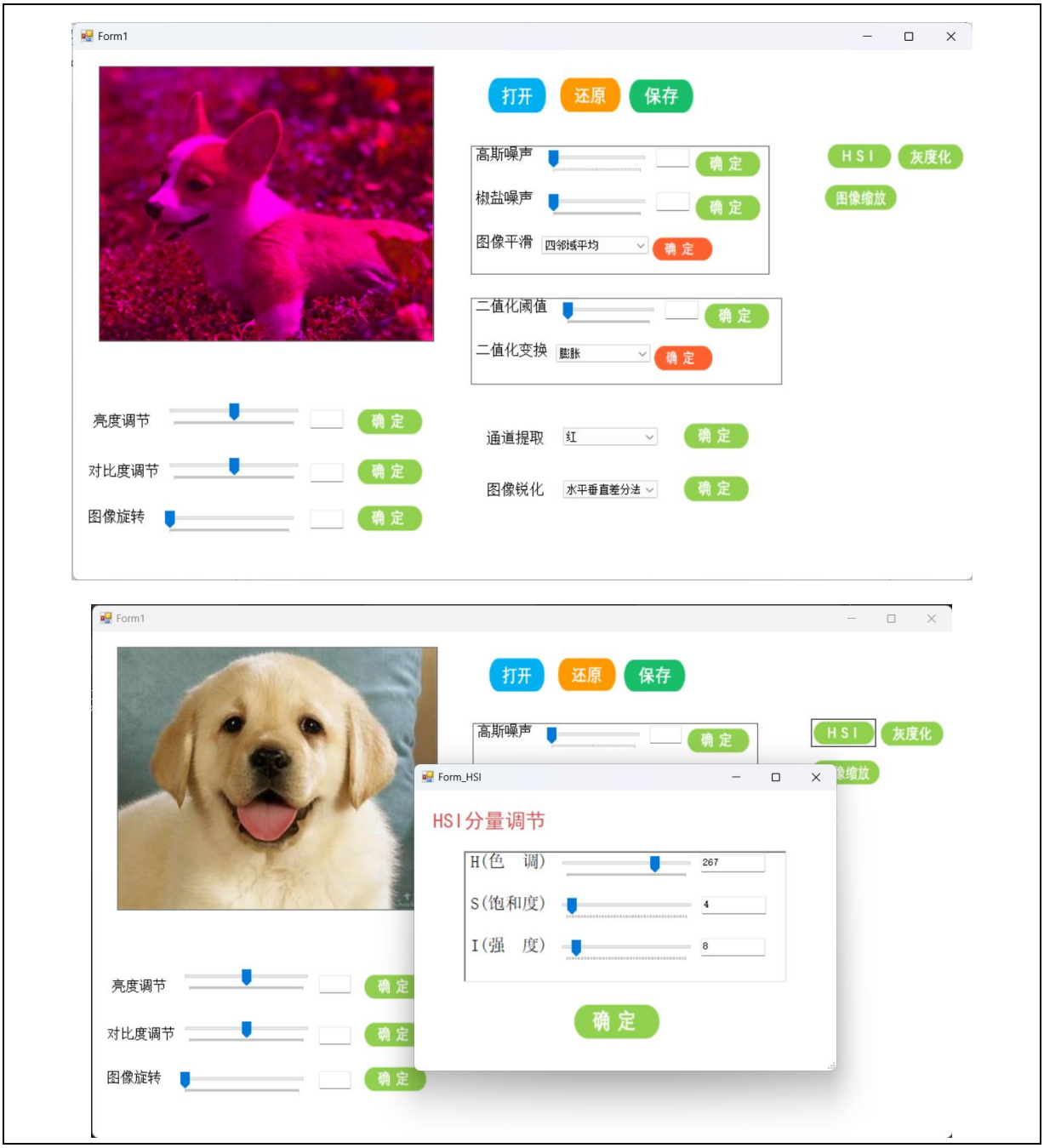
```
                if (Green < 0) Green = 0;
                if (Blue > 255) Blue = 255;
                if (Blue < 0) Blue = 0;
                bt2.SetPixel(i, j, Color.FromArgb(Red, Green, Blue));
            }
            pictureBox_show.Refresh();
            pictureBox_show.Image = bt2;
        }
    }
    if (H > 120 && H <= 240)
    {
        for (int i = 0; i < bt1.Width; i++)
        {
            for (int j = 0; j < bt1.Height; j++)
            {
                int R, G, B;
                color = bt1.GetPixel(i, j);
                R = color.R;
                G = color.G;
                B = color.B;
                Red = R + I * (I - S); //根据转换公式，在原各 RGB 分量基础上加上调整的值
                Green = (int)(G + I * (1 + (S * Math.Cos((H - 120) * Hudu) / (Math.Cos((180 -
               H) * Hudu)))));
                Blue = B + 3 * I - (Green + Red);
                if (Red > 255) Red = 255; //判断是否超出各分量允许的范围，如果大于 255 则只能等于 255
                if (Red < 0) Red = 0; //如果小于 0 则只能等于 0
                if (Green > 255) Green = 255;
                if (Green < 0) Green = 0;
                if (Blue > 255) Blue = 255;
                if (Blue < 0) Blue = 0;
                bt2.SetPixel(i, j, Color.FromArgb(Red, Green, Blue));
            }
            pictureBox_show.Refresh();
            pictureBox_show.Image = bt2;
        }
    }
    if (H > 240 && H <= 360)
    {
        for (int i = 0; i < bt1.Width; i++)
        {
            for (int j = 0; j < bt1.Height; j++)
            {
                int R, G, B;
                color = bt1.GetPixel(i, j);
                R = color.R;
                G = color.G;
                B = color.B;
                Green = G + I * (I - S); //根据转换公式，在原各 RGB 分量基础上加上调整的值
                Blue = (int)(B + I * (1 + (S * Math.Cos((H - 120) * Hudu) / (Math.Cos((300 - H)
               * Hudu)))));
                Red = R + 3 * I - (Blue + Green);
```

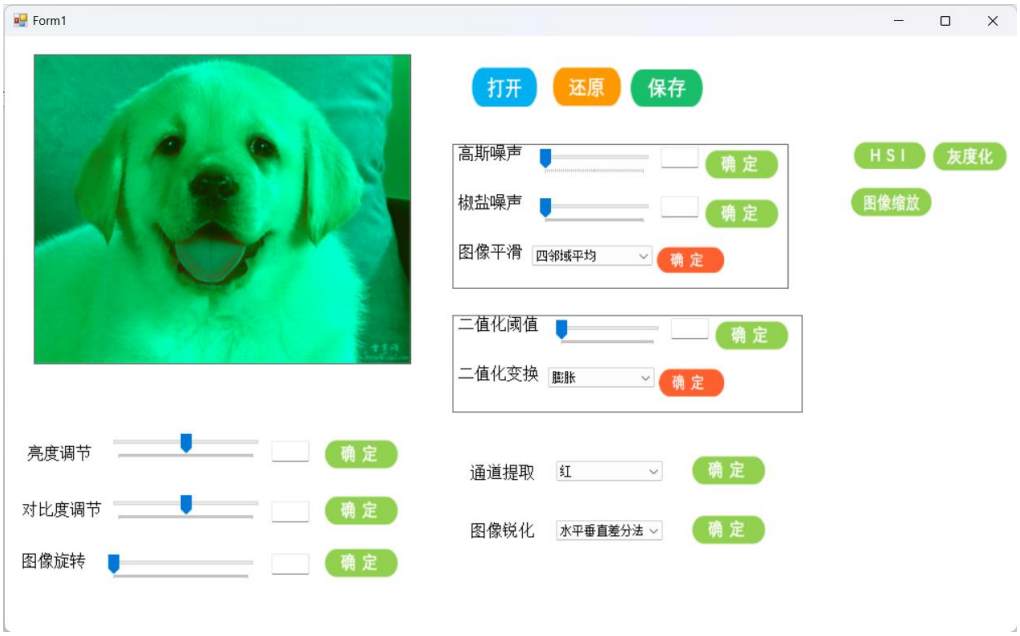
```
                if (Red > 255) Red = 255; //判断是否超出各分量允许的范围，如果大于 255
    则只能等于 255
                if (Red < 0) Red = 0; //如果小于 0 则只能等于 0
                if (Green > 255) Green = 255;
                if (Green < 0) Green = 0;
                if (Blue > 255) Blue = 255;
                if (Blue < 0) Blue = 0;
                bt2.SetPixel(i, j, Color.FromArgb(Red, Green, Blue));
            }
            pictureBox_show.Refresh();//另外建立了一个图片框用于显示处理后的图像
            pictureBox_show.Image = bt2;
        }
    }
}
```

- **实现效果：**

点击 HIS 按钮弹出调节页面，根据需要调节

点击确定按钮即可：


Form1
打开
还原
保存
高斯噪声
确 定
椒盐噪声
确 定
图像平滑
四邻域平均
确 定
HSI
灰度化
图像缩放
二值化阈值
确 定
二值化变换
膨胀
确 定
亮度调节
确 定
对比度调节
确 定
图像旋转
确 定
通道提取
红
确 定
图像锐化
水平垂直差分法
确 定
Form1
打开
还原
保存
高斯噪声
确 定
HSI
灰度化
Form_HSI
HSI分量调节
H(色 调)
267
S(饱和度)
4
I(强 度)
8
确 定
亮度调节
确 定
对比度调节
确 定
图像旋转
确 定



五、实验结果及分析(包括心得体会，本部分为重点，不能抄袭复制)

➢ 完成情况:

完成了实验全部的基本要求和全部的扩展要求，最终的结果基本达到了我的预期

➢ 踩坑记录:

imshow 函数的坑: 必须有两个形参，即第一个显示窗口名称的参数不能省略；一般需要在后面添加一个 `cv2.waitKey(0)`，代表由手动确定下一步操作，否则会出现显示图像一闪而过的情况，或是出现图像无响应的情况。还有一个坑在于，如果用 cv2.imread()读取 RGB 图像，再用其他库方法显示，就很有可能出现显示的图像和原本的图像颜色完全不一致！这是因为 cv2.imread()函数读取 RGB 图像时，返回的图像格式的通道并不是按 R、G、B 排列的，而是按 B、G、R 顺序排列的！其示例可以参考 matplotlib & visdom 的图片显示问题中给出的例图，这里也给出一段将 B、G、R 顺序转换为 R、G、B 顺序的代码:

读取与显示图像常见的坑: 路径中有中文字符；路径中使用了\被误认为转义字符;图像一闪而过,或出现图像未响应的情况:在 cv2.imshow()后添加 cv2.waitKey(0)。

使用一个 pictureBox 展示图像注意的点: 由于我使用的是一个展示

框，而在图片处理的过程难免需要原始图像的一些信息，因此我在主类中额外定义了一个静态属性(public static PictureBox original_image = new PictureBox();)用于存储原始图像，之后不管是复原操作还是其他需要用到原始图像的操作均可以从这个属性中获取。

- **实验心得**

通过完成这个实验，我对图像处理的基本技术有了更深入的理解。我学会了灰度化和二值化的算法，并且了解了如何调整图像的亮度、对比度和饱和度等。我还发现不同的算法和参数选择对图像处理结果的影响很大，需要经过不断尝试和调整才能得到满意的效果。

此外，我还学会了如何使用编程语言实现图像处理算法。我熟悉了相关的函数和库，并且能够将算法设计转化为实际的代码。通过不断的实验和调试，我提高了自己的编程能力和问题解决能力。

总的来说，这个实验对我来说是一次有意义的学习和实践。我不仅学到了图像处理的基本技术，还提高了自己的编程能力。我相信这些知识和经验对我的学习和未来的工作都将有所帮助。