天津理工大学

计算机科学与工程学院

实验报告

2017 至 2018 学年 第 二 学期

实验八 彩色图像综合处理

课程名称	数字图像处理				
学号	20152180	学生姓名	王帆	年级	2015
专业	计算机 科学与技术	教学班号	2	实验地点	7-212
实验时间	2018 年 5 月 14 日 第 5 节 至 第 6 节				
主讲教师	杨淑莹				

实验成绩

软件运行	效果	算法分析	流程设计	报告成绩	总成绩

实验（八）	实验名称	**彩色图像综合处理**
软件环境	Windows Visual Studio 2017	
硬件环境	PC	

实验目的

设计并实现两种新的彩色图像处理方法

实验内容（应包括实验题目、实验要求、实验任务等）

一、 分析几种常用的色彩空间基本原理

说明：

1. 色彩学基本原理
   - ① 什么是颜色
   - ② 颜色的属性
   - ③ 光源能量分布图
2. CIE 色度图基本原理
   - ① CIE 色彩模型的建立
   - ② CIE 色度图的理解
3. 常用的色彩空间基本原理
   - ① RGB 颜色空间
   - ② CMY/CMYK 颜色空间
   - ③ HSV/HSB 颜色空间
   - ④ HSI/HSL 颜色空间
   - ⑤ Lab 颜色空间
   - ⑥ YUV/YCbCr 颜色空间

针对某一个你感兴趣的图像处理项目，如人脸识别、身份证号码识别、汽车牌照识别等，实现以下功能

二、 编程实现色彩空间的转换处理

说明：实现色彩空间的转换方法基本原理

- ① RGB 转换到 HSV 的方法基本原理
- ② RGB 转换到 HSI 的方法基本原理
- ③ RGB 转换到 YUV 的方法基本原理
- ④ RGB 转换到 YCbCr 的方法基本原理

编程：实现色彩空间的转换

三、 编程实现图像综合处理

实验过程与实验结果

一、分析几种常用的色彩空间基本原理

1.色彩学基本原理

1）什么是颜色

从物理成因上来说，颜色是光作用与人眼的结果。光的本质是一种电磁波，根据麦克斯韦电磁波理论，变化的磁场产生电场，变化的电场产生磁场。如果在空间某处存在变化的电场，那么变化的电场和磁场并不局限于空间的某个局域，而是由近及远向周围空间传播开去。电磁场的传播，就形成了电磁波。电磁波的波长和强度可以有很大的区别。

颜色是通过眼、脑和我们的生活经验所产生的一种对光的视觉效应，我们肉眼所见到的光线，是由波长范围很窄的电磁波产生的，不同波长的电磁波表现为不同的颜色，对色彩的辨认是肉眼受到电磁波辐射能刺激后所引起的一种视觉神经的感觉。颜色具有三个特性，即色相，明度，和饱和度。颜色的三个特性及其相互关系可以用三度空间的颜色立体来说明。

2）颜色的属性

色相：色相是色彩的首要特征，是区别各种不同色彩的最准确的标准。事实上任何黑白灰以外的颜色都有色相的属性，而色相也就是由原色、间色和复色来构成的。色相，色彩可呈现出来的质地面貌。自然界中各个不同的色相是无限丰富的，如紫红、银灰、橙黄等。色相即各类色彩的相貌称谓。

颜色测量术语，颜色的属性之一，借以用名称来区别红、黄、绿、蓝等各种颜色。即各类色彩的相貌称谓，如大红、普蓝、柠檬黄等。色相是色彩的首要特征，是区别各种不同色彩的最准确的标准。事实上任何黑白灰以外的颜色都有色相的属性，而色相也就是由原色、间色和复色来构成的。

色相的特征决定于光源的光谱组成以及有色物体表面反射的各波长辐射的比值对人眼所产生的感觉。在测量颜色时，可用色相角 H 及主波长 λd（nm）表示。在聚合物中为根据色的 XZY 系列表示的主波长和补色主波长相对应的色感觉。

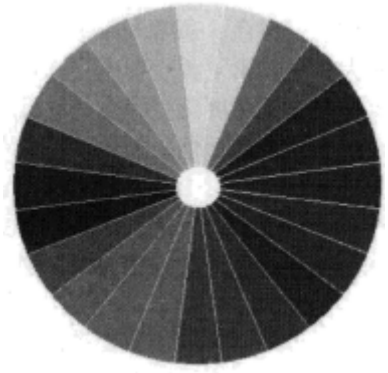

图 1-1 色相环

亮度：亮度是指发光体（反光体）表面发光（反光）强弱的物理量。人眼从一个方向观察光源，在这个方向上的光强与人眼所“见到”的光源面积之比，定义为该光源单位的亮度，即单位投影面积上的发光强度。亮度的单位是坎德拉/平方米（cd/m2）亮度是人对光的强度的感受。

纯度：纯度通常是指色彩的鲜艳度。从科学的角度看，一种颜色的鲜艳度

取决于这一色相发射光的单一程度。人眼能辨别的有单色光特征的色，都具有一定的鲜艳度。不同的色相不仅明度不同，纯度也不相同。此外化学试剂也有纯度的划分。

3）光源能量分布图

光源所发出的光谱组成以能量来表示，即光源发出不同波长的辐射功率的相关分布，称为光源能量分布图，或光谱能量分布图。光源能源分布图可用来决定光源的重要信息，颜色的主要属性都可以从光源能量分布图中获得。

对于光源能量分布图的而理解能够帮助我们更加深刻地认识颜色属性与光学本质的联系。光源能量分布图中，横轴表示电磁波的波长，而纵轴表示能量。自然光的本质多少由各种波长的光混合而成的。由于每种波长的光所发出的能量有差异，所以就呈现了不同的色彩。

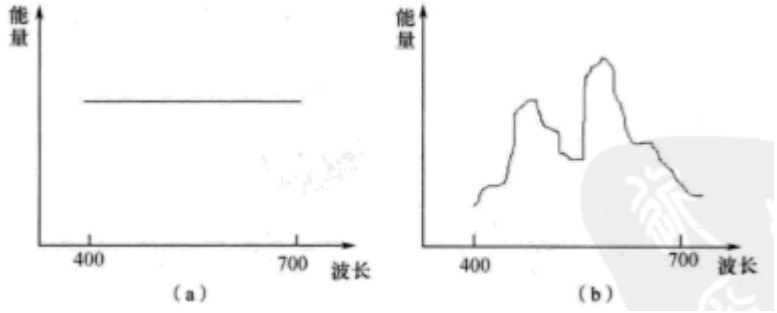

图 1-2 光源能量分布图

2.CIE 色度图基本原理

1）CIE 色彩模型的建立

CIE 根据一些数据，对不同波长的红，绿，蓝光做出椎体细胞的敏感度情况描述，分别称为 RGB 三刺激值，并由此建立“标准色度观察值”标准，该标准定了普通人眼对颜色的响应，从而奠定了现代 CIE 标准色度学的定量基础。使用 R，G，B3 种颜色匹配可见光光谱中的颜色的匹配表达式：C=rR+gG+Bb

其中 r,g,b 分别为 3 种原色的权值。但是标准 RGB 三原色匹配任意颜色的光谱三刺激值曲线中的一部分 500um 附近的 r 的是负值。矛盾在于颜色匹配过程中，权值有可能为负值，但是实际中却并不存在负的光强，所以必须找到一组原色来代替 RGB，是权值都为正。

在使用 CIE 标准三原色 X，Y,Z 去匹配颜色时，XYZ 空间包含所有可见光的部分将形成一个椎体，也就是 CIE 颜色空间。

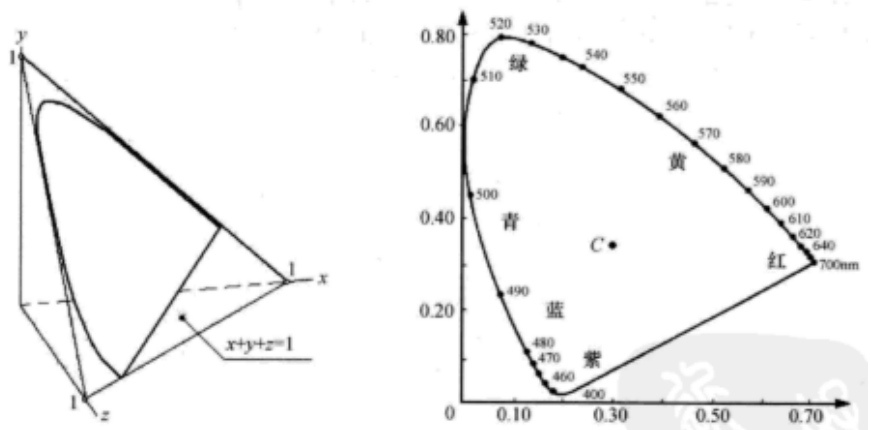

图 1-3 CIE 色度图

2）CIE 色度图的理解

1.确定互补颜色

两种彩色光源混合后能够生成白色光，则称他们为互补色。利用 CIE 色度图可以得到光谱色的互补色。从颜色点过白光点 C 点做一条直线，求其与侧光谱曲线的交点即可。从互补色的定义可知，两种补色按照一定比例相加后可以得到白色。因此，一种颜色的补色并不仅仅是明确的一种颜色，而是一组颜色。但互为补色的两颜色点连线，一定通过白光点 C。

2.确定色光主波长

如果有一点 C1，那么将其与白光点 C 相连所形成的直线与马蹄形曲线轮廓的交点所指示的波长，就是生成该种色彩的所有混合光线中能量最大的那种光的波长，或称为主波长。还存在我们实际上看到的它们的延长线与紫外线相交于一点 Cp，但是紫外线上的点并不属于可见光光谱的范围，此时主波长应是位于颜色反侧的光谱轨迹交点。

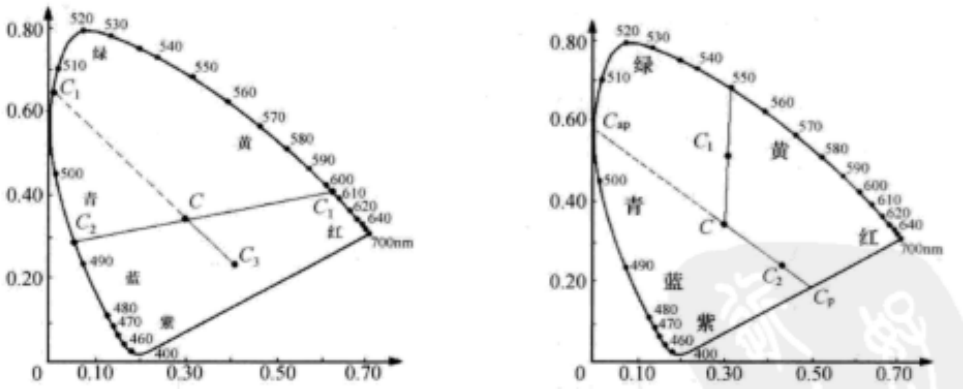

图 1-4 色光主波长示意图

3.定义颜色区域

假设 I 和 J 是两种任意的颜色，那么当二者按不同比例进行混合后，可以产生的颜色就必然是它们连线上的一种颜色。这 3 点可以合成移它们作为顶点的三角形中的任意一种颜色。从这个角度出发，也可以解释为什么 RGB3 种原色无法合成可见光光谱上的所有颜色。因此在 CIE 色度图上，以红，绿和蓝 3 种颜色混合后所能生成的全部颜色能且仅能位于以这 3 点为顶点的三角形中。

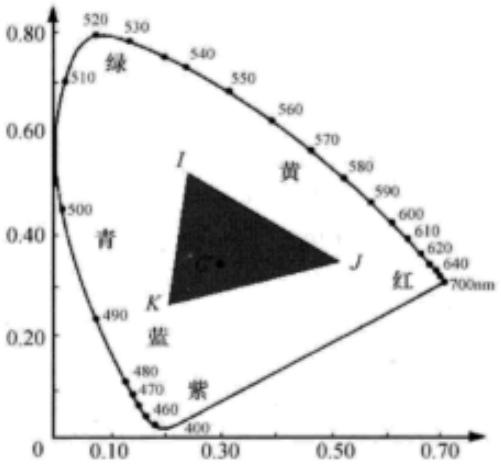

图 1-5 颜色区域图

3.常见的的色彩空间基本原理

1）RGB 颜色空间

RGB 颜色空间是常见的一种颜色模型，它被称为是与设备相关的色彩空间。在 CRT 显示系统中，彩色阴极射线管使用 R,G，B 数值来驱动电子枪发射电子，并分别激发荧光屏上的 R，G，B 这 3 中颜色的荧光粉以发出不同亮度的光线，并通过相加混合物产生各种颜色。这也就是 RGB 颜色系统的原理。RGB 色彩系统之所以能够表示用来表示彩色，归根到底还是因为人眼中的锥状细胞和棒状细胞对红色，蓝色和绿色特别敏感。

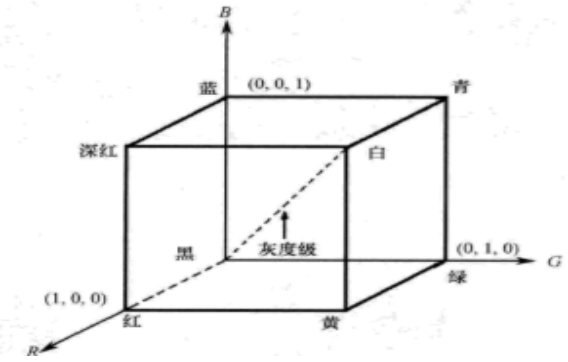

图 1-6 RGB 颜色空间示意图

2）CMY/CMYK 颜色空间

颜色的特性与光线相反，颜料吸收光线，而不增强光线。所以在使用颜色进行作画或者印刷时，RGB 将不再适用。因为颜色特性与光线相反，所以很容易让人想到只有将光的三原色进行补色就可以很好的解决问题，而红，绿，蓝 3 色的补色刚好是青，洋红和黄色。

CMY 颜色空间的设计原理。CMY 颜色空间常应用于印刷工业，印刷业通过 C,M，Y 三原色油膜的不同网点面积率的叠印来表示丰富多彩的颜色，真便是 CMY 颜色空间。通常使用的大多数纸张上沉淀颜色的设备。都要求输入 CMY 数据，即使输入的是 RGB 颜色数据，在内部也会进行 RGB 到 CMY 的转换。

在 CMY 的基础上加入第四重颜色——黑色，从而提出 CMYK 彩色模型。CMYK 颜色空间是和设备或者印刷过程相关的，因此不同的条件有可能产生不同的印刷结果，最终结果将受工艺方法，油墨的特性或者纸张特性等多种因素影响。

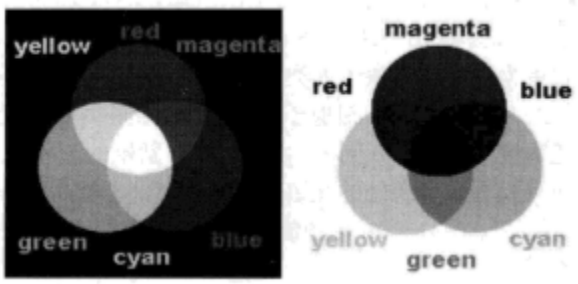

图 1-7 CMY/CMYK 颜色空间示意图

3）HSV/HSB 颜色空间

RGB 和 CMYK 都是对于机器而言，但对于用户却是不可见的。相对于

RGB 和 CMYK 颜色模型，HSV，有时也称 HSB，对用户老说是一种更加直观的颜色模型，它更为准确的反映了人类视觉系统对颜色的理解方式。HSV 模型色相，Saturation 饱和度，Value 纯度。对应于圆柱坐标系的一个圆锥形子集。V 表示色彩的明亮程度，范围由 0 到 1.圆锥的顶面对应于 V=1，代表颜色的亮度。H 参数表示彩色信息，即所处的光谱颜色的位置，也就相当于前面所提到的色相，该参数用一个角度量来表示，它由绕 V 轴的旋转角给定。

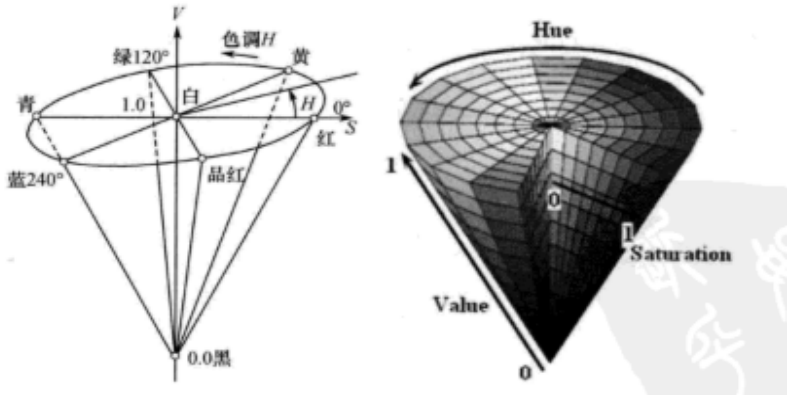

图 1-8 HSV/HSB 颜色空间示意图

4）HSI/HSL 颜色空间

HSI 彩色空间，也称 HSL 彩色空间，同样是从人类的视觉系统出发的，它与 HSV 非常相似，区别在于一种纯色的亮度等于白色的亮度，而纯色的亮度却等于中度灰的明度。

HSI 用色调，饱和度和强度描述色彩。色彩是描述纯色的属性，它反映了色彩的本质。饱和度的作用在于给出一种纯色被白光稀释的程度描述。强度是颜色的亮度或光亮度，取值范围从黑到最亮。强度是单色图像最有力最有效的描述方式。它的好处在于它可测而且易于解释。

HSB 和 HSL 是两种十分相近的彩色空间，它们都定义台式机图形程序中的颜色，而且它们多是利用 3 条轴来定义颜色。唯一的不同点只是 B 和 L 两个分量存在区别。需要注意的是：HSL 颜色饱和度最高时的光亮度 L 定义为 0.5，而 HSV 颜色饱和度最高时的明度值 B 则为 1.0.

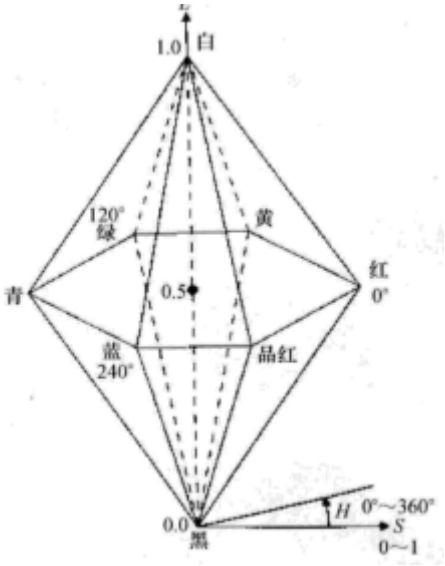

图 1-8 HSI/HSL 颜色空间示意图

5）Lab 颜色空间

Lab 颜色空间是由 CIE 制定的另外一种彩色模型，它是应用最广泛的颜色模型之一。CIE 与 1976 年开发完成了这套色彩模型。Lab 彩色模型用 3 组数值表示色彩：即亮度数值 L,其值从 0 到 100.红色和绿色两种原色之间的变化区域 a，a 取正值时表示红色。取负值表示绿色。b 表示黄色到蓝色两种原色之前的变化区域，b 去正直表黄色，取负值表示蓝色。。Lab 的色彩理论建立在人对色彩感觉的基础上。Lab 颜色理论认为，在一个物体中，红色和绿色两种原色不能同时并存。黄色和蓝色两种原色也不能同时并存，所以 a 值只能表示红色或绿色中一种颜色。

Lab 色彩模型可以说是最大范围的色彩模型，自然界中任何颜色都可以在 Lab 空间中表达出来，它的色彩空间比 RGB 空间还要大。同样 Lab 也是一种也设备无关的色彩空间，无论使用何种设备创建或输出图像，这种模型都能生成一种颜色。

6）YUV/YCbCr 颜色空间

YUV 是应用于电视机系统的而一种颜色编码方式，它主要用于优化彩色视频信号的传输，使其向后兼容老式黑白电视。因为他的亮度信号 U,V 是分离的。所以如果没有 U,V 分的图量，那么表示的图就是黑白灰度图，这样黑白电视机就也可以接受彩色信号了。除此之外，YUV 表示法的另一个优点是可以利用人眼的特性来降低狮子彩色图像所需要的存储你容量。

YUV 色彩空间与 Lab 色彩空间十分相似，它也是用亮度和色差来描述色彩分量。其中亮度信号用 Y 表示，U 表示色差信号 R-Y，V 表示色差信号 B-Y。

YCbCr 色彩空间是由 YUV 颜色空间派生出来的一种颜色空间，主要用于数字视频系统中，其中 Y 是指亮度分量，Cb 指蓝色亮度分量，Cr 指红色色度分量。我们在数字电子媒体领域也常常谈到的 YUV 格式，但事实上，这里所说的 YUV 是以 YCbCr 色彩空间模型为基础的具有多种存储格式的一类颜色模型的家族。这种彩色编码方案的原理都依赖于这样一个事实，即肉眼对视频的 Y 分量更敏感。YCbCr 模型的区别主要在于 UV 数据的采样方式和存储方式。

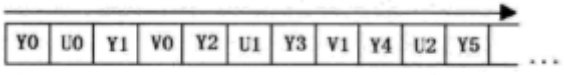

图 1-9 YUV/YCbCr 颜色空间示意图

二、编程实现色彩空间的转换处理

1. RGB 转换到 HSV 的方法基本原理

将 R，G，B 归一化，将变换后的 H 的取值范围设为 0-360，则 RGB 到 HSV 的变换公式重写为:

$$V = \max(R, G, B)$$

$$S = \max(R, G, B) - \frac{\min(R, G, B)}{\max(R, G, B)}$$

$$H = B * 60 \quad B \geq 0$$

$$H = B * 60 \quad B < 0$$

代码：

```
private void hSVToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        Color color;
        int r, g, b = 0;
        height = opeBitmap.Height;
        width = opeBitmap.Width;
        Bitmap bitmap = new Bitmap(opeBitmap);
        for (int i = 0; i < width; i++)
        {
            for (int j = 0; j < height; j++)
            {
                color = bitmap.GetPixel(i, j);
                r = color.R;
                g = color.G;
                b = color.B;
                int Hi, f, p, q, t;
                int H = 0, S = 0, V = 0;
                int max = COMUtil.getMax(r, g, b);
                int min = COMUtil.getMin(r, g, b);
                if (max != min)
                {
                    if (max == r)
                    {
                        H = (g - b) / (max - min);
                    }
                    else if (max == g)
                    {
                        H = 2 + (b - r) / (max - min);
                    }
                    else if (max == b)
                    {
                        H = 4 + (r - g) / (max - min);
                    }
                }
                H = H * 60;
                if (H < 0)
                {
                    H = H + 360;
                }
                V = max;
                if (max != 0)
```

```
{
    S = (max - min) / max;
}

Hi = Math.Abs(H / 60);
f = H / 60 - Hi;
p = V * (1 - S);
q = V * (1 - f * S);
t = V * (1 - (1 - f) * S);
if (Hi == 0)
{
    r = V;
    g = t;
    b = p;
}
else if (Hi == 1)
{
    r = q;
    g = V;
    b = p;
}
else if (Hi == 2)
{
    r = p;
    g = V;
    b = t;
}
else if (Hi == 3)
{
    r = p;
    g = q;
    b = V;
}
else if (Hi == 4)
{
    r = t;
    g = p;
    b = V;
}
else if (Hi == 5)
{
    r = V;
    g = p;
    b = q;
```

```
                }
                bitmap.SetPixel(i, j, Color.FromArgb(r, g, b));
            }
        }
        curBitmap = new Bitmap(bitmap);
        bitmap.Dispose();
        pictureBox_new.Image = curBitmap;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageB
oxIcon.Stop);
    }
}
```

示意图：

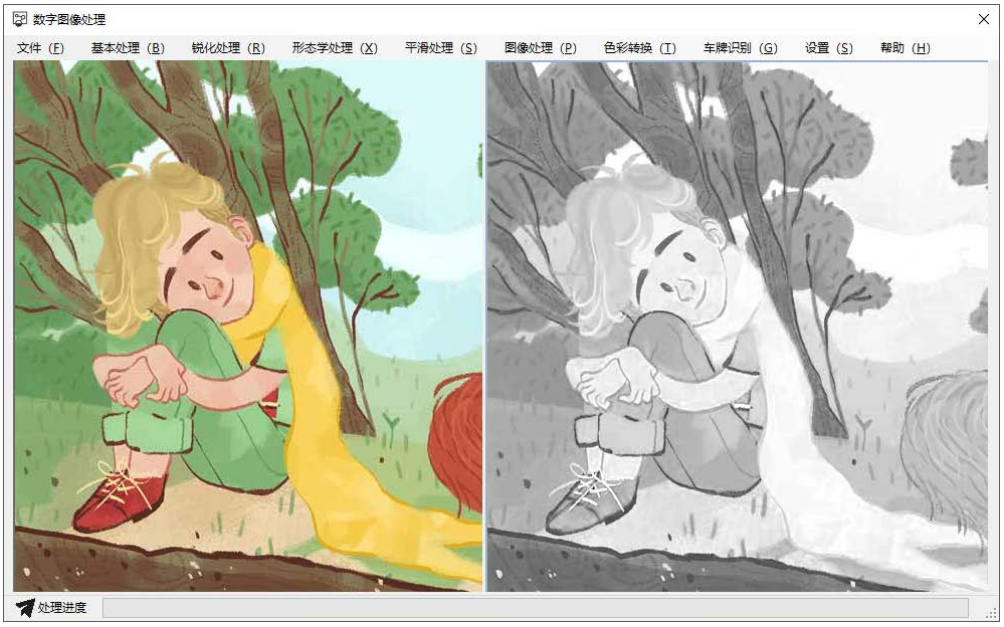

图 2-1 RGB->HSV 颜色空间转换

2. RGB 转换到 HSI 的基本原理与方法

RGB → HSI 其算法描述如下：

$$I = 1/3(R + G + B)$$

$$S = 1 - \left\{\frac{3}{R + B + G}\right\} * temp$$

$$H = \arccos\left\{0.5 * \frac{\{(R - G) + (R - B)\}}{\{(R - G)^2 + (R - B) * (G - B)\}^{0.5}}\right\}$$

$$If\ S = 0\ then\ H = 0$$

$$If (B/I) > (G/I) then H = 360 - H$$

将R，G，B归一化，RGB到HSV的变换公式重写为:

$H = O \ \ B \leq G$
$H = 360 - O \ \ \ B > G$
$\mathrm{O} = \arccos\{1/2 * (\mathrm{R} - \mathrm{G}) + (\mathrm{R} - \mathrm{B0})\}/(\mathrm{R} - \mathrm{G})\text{^}2 + (\mathrm{R} - \mathrm{G})(\mathrm{G} - \mathrm{B})\text{^}2\}\text{^}0.5$
$S = 13/(R + G + B) - min(R, G, B)$
$I = 1/3(R + G + B)$

代码：

```
private void hSIToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        Bitmap bitmap = new Bitmap(opeBitmap);
        Color color;
        int r, g, b = 0;
        for (int i = 0; i < opeBitmap.Width; i++)
        {
            for (int j = 0; j < opeBitmap.Height; j++)
            {
                int H, S, I;
                color = opeBitmap.GetPixel(i, j);
                r = color.R;
                g = color.G;
                b = color.B;
                H = S = I = 0;
                if (g != b && r != 0 && g != 0 && b != 0)
                {
                    double F = (2 * r - g - b) / (g - b);
                    I = (r + g + b) / 3;
                    if (g > b)
                    {
                      H=(90-(int)Math.Tan((double)F/(int)Math.Sqrt(3.0)))/360;
                    }
                    else
                    {
                      H=((90-(int)Math.Tan((double)F/Math.Sqrt(3.0)))+180)/360;
                    }
                    S = 1 - COMUtil.getMin(r, g, b) / (b);
                }
                if (H >= 0 && H < 120)
                {
            r=(int)(1+S*Math.Cos(H*1.0)/Math.Cos(1.0*(60-H))/ Math.Sqrt(3.0));
                    b = (1 - S) / (int)Math.Sqrt(3.0);
                    g = I * (int)Math.Sqrt(3.0) - r - b;
```

```
                }
                else if (H >= 120 && H < 240)
                {
r=(int)(1+S*Math.Cos(H*1.0-120)/Math.Cos((180-H)*1.0))/(int)Math.Sqrt(3.0);
                    b = (1 - S) / (int)Math.Sqrt(3.0);
                    g = I * (int)Math.Sqrt(3.0) - r - b;
                }
                else if (H >= 240 && H < 360)
                {
r=(int)(1+S*Math.Cos(H*1.0-240)/Math.Cos((300-H)*1.0))/(int)Math.Sqrt(3.0);
                    b = (1 - S) / (int)Math.Sqrt(3.0);
                    g = I * (int)Math.Sqrt(3.0) - r - b;
                }
                g = Math.Abs(g);
                bitmap.SetPixel(i, j, Color.FromArgb(r, g, b));
            }
        }
        curBitmap = new Bitmap(bitmap);
        bitmap.Dispose();
        pictureBox_new.Image = curBitmap;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageB
oxIcon.Stop);
    }
}
```

示意图：

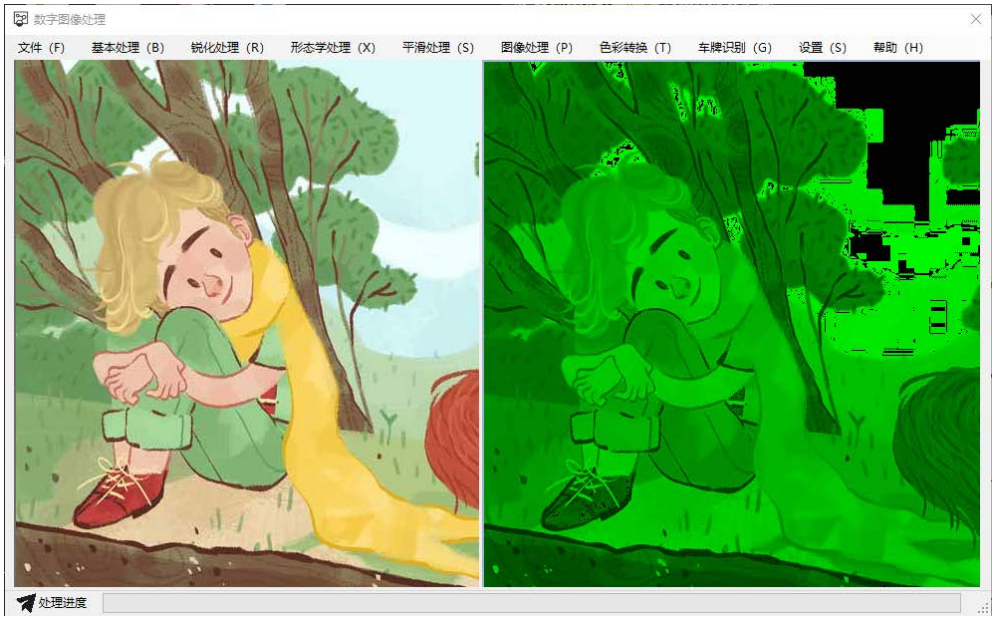

图 2-2 RGB->HSI 颜色空间转换

3. RGB 转换到 YUV 的方法基本原理

下面给出 RGB 到 YUV 的变换公式，将 R，G，B 归一化

$$Y = 0.299R + 0.587G + 0.114B$$
$$U = -0.14713R - 0.28886G + 0.436B$$
$$V = 0.615R - 0.51499G - 0.10001B$$

RGB 到 YUV 的变换公式

$$R = Y + 1.13983U$$
$$G = Y - 0.39465U - 0.58060V$$
$$B = Y + 2.03211U$$

代码：

```
private void yUVToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        Bitmap bitmap = new Bitmap(opeBitmap);
        Color color;
        int r, g, b = 0;
        double Y, U, V;
        for (int i = 0; i < opeBitmap.Width; i++)
        {
            for (int j = 0; j < opeBitmap.Height; j++)
            {
                color = opeBitmap.GetPixel(i, j);
                r = color.R;
                g = color.G;
                b = color.B;
                Y = U = V = 0;
                Y = (0.299 * r + 0.587 * g + 0.114 * b);
                U = (-0.1687 * r - 0.3313 * g + 0.5 * b);
                V = (0.5 * r - 0.4187 * g - 0.0813 * b);
                r = (int)(Y + 1.401 * V) / 2;
                g = (int)(Y - 0.34414 * U - 0.71414 * V) / 2;
                b = (int)(Y + 1.1772 * U) / 2;
                g = Math.Abs(g);
                bitmap.SetPixel(i, j, Color.FromArgb(r, g, b));
            }
        }
        curBitmap = new Bitmap(bitmap);
        bitmap.Dispose();
        pictureBox_new.Image = curBitmap;
    }
    catch (Exception ex)
```

```
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
    }
}
```

示意图：

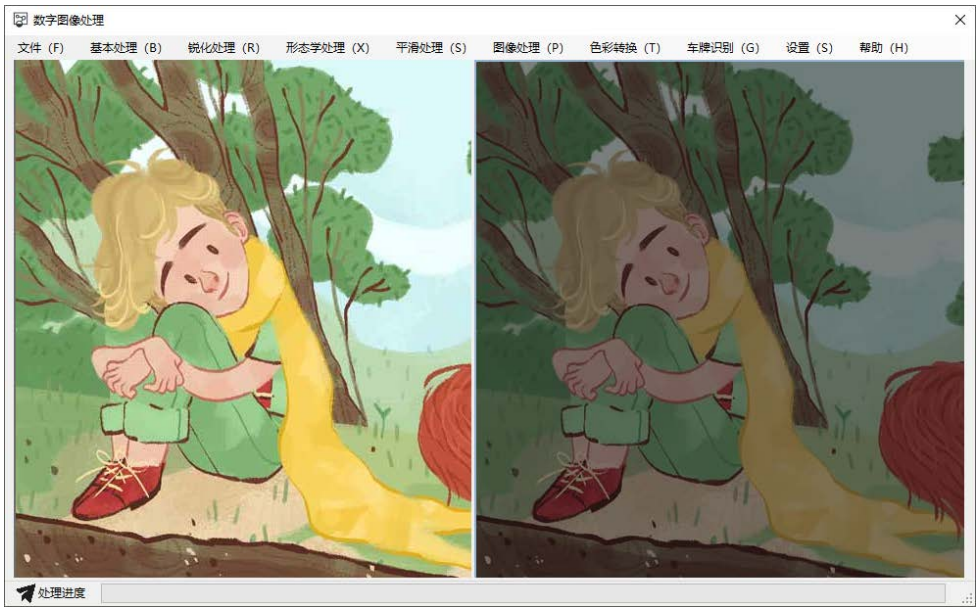

图 2-3 RGB->YUV 颜色空间转换

4. RGB 转换到 YCbCr 的方法基本原理

$$Y = 0.29900R + 0.58700G + 0.11400B$$
$$Cb = -0.16874R - 0.33126G + 0.50000B$$
$$Cr = 0.50000R - 0.41869G - 0.08131B$$

若 Cb，Cr 的值取正，以上公式变换为：

$$Y = 0.29900R + 0.58700G + 0.11400B$$
$$Cb = -0.16874R - 0.33126G + 0.50000B + 128$$
$$Cr = 0.50000R - 0.41869G - 0.08131B + 128$$

YcbCr 色彩空间转化为 RGB 色彩空间的公式：

$$R = Y = 1.402（Cr - 128）$$
$$G = Y - 0.34414（Cb - 128）- 0.71414（Cr - 128）$$
$$B = Y + 1.772（Cb - 128）$$

代码：

```
private void yCbCrToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        Bitmap bitmap = new Bitmap(opeBitmap);
        Color color;
        int r, g, b = 0;
```

```
        double Y, Cb, Cr;
        Y = Cb = Cr = 0;
        for (int i = 0; i < opeBitmap.Width; i++)
        {
            for (int j = 0; j < opeBitmap.Height; j++)
            {
                color = opeBitmap.GetPixel(i, j);
                r = color.R;
                g = color.G;
                b = color.B;
                Y = 0.299 * r + 0.587 * g + 0.114 * b;
                Cb = -0.1687 * r - 0.3313 * g + 0.5 * b + 128;
                Cr = 0.5 * r - 0.4187 * g - 0.0813 * b + 128;
                r = (int)(Y + 1.402 * (Cr - 128)) / 2;
                g = (int)(Y - 0.34414 * (Cb - 128) - 0.71414 * (Cr - 128))/ 2;
                b = (int)(Y + 1.772 * (Cb - 128)) / 2;
                bitmap.SetPixel(i, j, Color.FromArgb(r, g, b));
            }
        }
        curBitmap = new Bitmap(bitmap);
        bitmap.Dispose();
        pictureBox_new.Image = curBitmap;
    }
    catch (Exception ex)
    {MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageB
oxIcon.Stop); }
}
```

示意图：

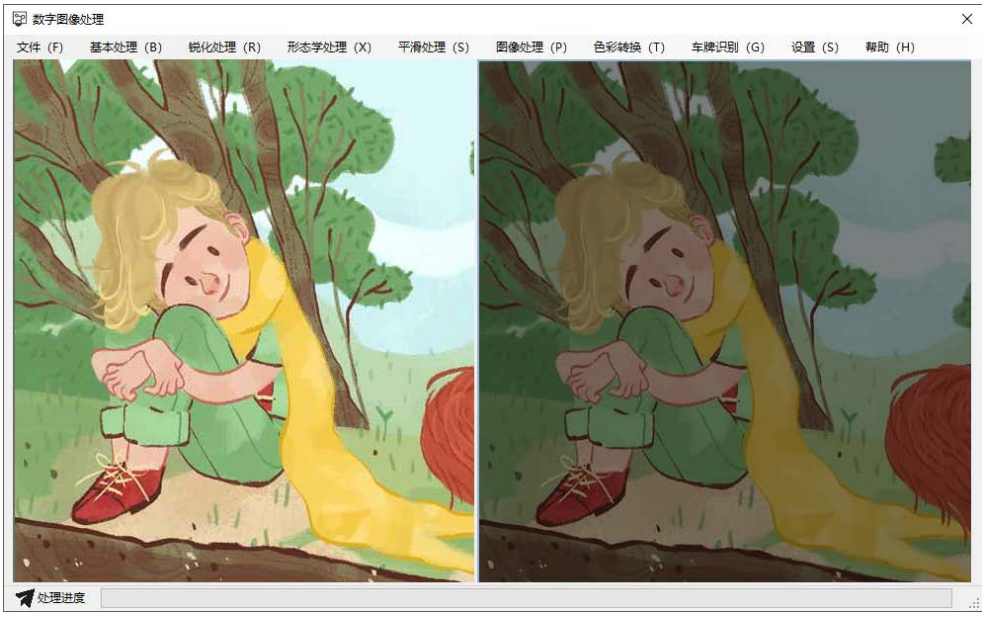

图 2-4 RGB->YCbCr 颜色空间转换

三、编程实现图像综合处理

通过前八次实验，可以实现一个综合的图像处理软件，界面如下图所示：

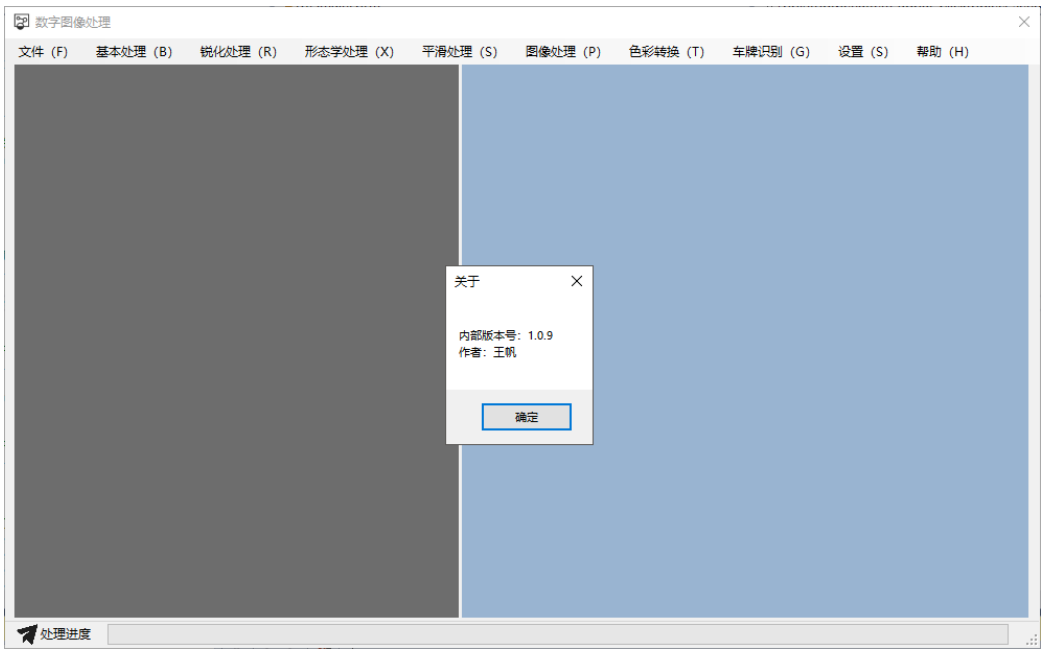

图 3-1 图像处理软件示意图

此外，由于对图像的处理存在不同的需求，某些时候需要叠加效果操作，而也存在仅需单步处理的情况，因此增加一个事件来处理这种情况。

代码：

```
//选项：设置-效果叠加-开启
private void ToolStripMenuItem_composition_open_Click(object sender,
EventArgs e)
{
    try
    {
        ToolStripMenuItem_composition_close.Checked = false;
        ToolStripMenuItem_composition_open.Checked = true;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageB
oxIcon.Stop);
    }
}

//选项：设置-效果叠加-关闭
private void ToolStripMenuItem_composition_close_Click(object sender, EventAr
gs e)
{
    try
    {
```

```
        ToolStripMenuItem_composition_close.Checked = true;
        ToolStripMenuItem_composition_open.Checked = false;
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageB
oxIcon.Stop);
    }
}
private void pictureBox_new_Paint(object sender, PaintEventArgs e)
{
    try
    {
        if (objBitmap != null)
        {
            if (ToolStripMenuItem_composition_close.Checked)
            {
                opeBitmap = new Bitmap(objBitmap);
            }
            else if (ToolStripMenuItem_composition_open.Checked)
            {
                opeBitmap = new Bitmap(curBitmap);
            }
            else
            {
                MessageBox.Show("绘图错误", "错误提示", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
            }
        }
    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK, MessageB
oxIcon.Stop);
    }}
```

示意图：

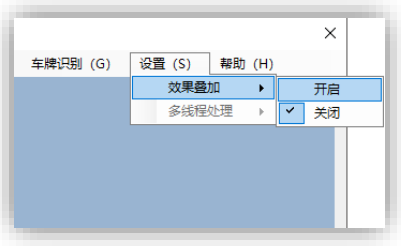

图 3-2 效果叠加选项设置

将项目（车牌识别）融合实现的代码为：

```
//选项：车牌识别
private void 车牌识别ToolStripMenuItem_Click(object sender, EventArgs e)
{
    try
    {
        //加载窗体CCSForm
        CCSForm ccsfrm = new CCSForm();

        //定义窗体所有者
        ccsfrm.Owner = this;

        ccsfrm.ShowDialog();
    }
    catch (Exception ex)
    {
        //错误提示
        MessageBox.Show(ex.Message, "错误提示", MessageBoxButtons.OK,
MessageBoxIcon.Stop);
    }
}
```

示意图：

图 3-3 车牌识别选项

附录（可包括实验所用方法的基本原理，源程序清单）

参考资料：

1.【图像处理】RGB 转 YUV 优化 - CSDN 博客
https://blog.csdn.net/jaych/article/details/50631298
2. RGB 与 YCbCr 颜色空间的转换 - CSDN 博客
https://blog.csdn.net/alfredtofu/article/details/6303305
3. 【C# / Algorithm】RGB、HSV、HSL 颜色模型的互相转换 - CSDN 博客
https://blog.csdn.net/ls9512/article/details/50001753