计算机视觉Exp4报告

姓名：吴宇祺 学号：16340242

算法描述

直方图均衡

1. 计算概率直方图

```
float* ColorProcessing::getHist(CImg<unsigned char> &img1, int flag)
{
    float* hist = new float[256];
    for (int i = 0; i < 256; ++i)
        hist[i] = 0;
    switch (flag)
    {
        case 4:
            cimg_forXY(img1, x, y)
            {
                hist[img1(x, y)]++;
            }
            break;
        default:
            cimg_forXY(img1, x, y)
            {
                hist[img1(x, y, flag)]++;
            }
    }
    int size = img1.width() * img1.height();
    for (int i = 0; i < 256; ++i)
        hist[i] /= size;
    return hist;
}
```

2. 累计分布函数

```
float* ColorProcessing::getCDF(float* hist)
{
    float* cdf = new float[256];
    for (int i = 1; i < 256; ++i)
        cdf[i] = hist[i] + cdf[i-1];
    return cdf;
}
```

3. 灰度映射

```
//CImg<unsigned char> ColorProcessing::grayHisteq(string path, string root)
    cimg_forXY(img1, x, y)
    {
        img1(x, y) = (int)255*cdf[img1(x, y)];
    }
//CImg<unsigned char> ColorProcessing::RGBHisteq(string path)
    cimg_forXY(img1, x, y)
    {
        img1(x, y, 0) = (unsigned char)255*cdfs[0][img1(x, y, 0)];
        img1(x, y, 1) = (unsigned char)255*cdfs[1][img1(x, y, 1)];
        img1(x, y, 2) = (unsigned char)255*cdfs[2][img1(x, y, 2)];
    }

```

用函数 *showHist*显示处理前后直方图：

```
void ColorProcessing::showHist(float* hist)
{
    CImg<unsigned char> fig = CImg<unsigned char>(256, 300, 1, 1, 0);
    cimg_forXY(fig, x, y)
    {
        if (y>(300 - 1500*hist[x]))
            fig(x, y) = 255;
    }
    fig.display();
}
```

颜色迁移

1. convert from RGB to lab space

   lab颜色空间是一个个颜色相关度第的正交色彩空间 ，l为亮度，ab为正交的色彩分量。

   1. convert from RGB to XYZ 三色值，（取决于显示器荧光，此处用设备无关），将白色从色度图(CIE xy)映射到RGB空间，反之亦然。在色度图中，白色被定义为 x = X/（X+Y+Z）=0.333，y = y/(X+Y+Z)=0.333。为了将X=Y=Z=1映射到R=G=B=1，设标准转换矩阵

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.5141 & 0.3239 & 0.1604 \\ 0.2651 & 0.6702 & 0.0641 \\ 0.0241 & 0.1228 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

2. 从XYZ转换到lab空间

$$\begin{bmatrix} L \\ G \\ B \end{bmatrix} = \begin{bmatrix} 0.3897 & 0.6890 & -0.0787 \\ -0.2298 & 1.1834 & 0.0464 \\ 0.0000 & 0.0000 & 1.0000 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

3. 综合以上两个变换矩阵得到一步变换

$$\begin{bmatrix} L \\ M \\ S \end{bmatrix} = \begin{bmatrix} 0.3811 & 0.5782 & 0.0402 \\ 0.1967 & 0.7244 & 0.0782 \\ 0.0241 & 0.1288 & 0.8444 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

4. 取以10为底的对数，再转为lab空间：

$$L' = log_{10} L \;\; M' = log_{10} M \;\; S' = log_{10} S$$

$$\begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{3}} & 0 & 0 \\ 0 & \frac{1}{\sqrt{6}} & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -2 \\ 1 & -1 & 0 \end{bmatrix} = \begin{bmatrix} L' \\ M' \\ S' \end{bmatrix}$$

```
CImg<float> ColorProcessing::rgb2lab(CImg<float> img);
```

2. calculate mean and standard deviation in lab space

以下两个函数分别返回某一张图的像素值均值，和两张图之间标准差的比值：

```
float* getMean(CImg<float> img);

float* getStandardDviationRatio(CImg<float> img1, float* mean1,
                                CImg<float> img2, float* mean2);
```

3. process image with following formula

假设l、a、b分别是源图像lαβ通道原有的数据，L、A、B分别是变换后得到新的源图像lαβ通道的值，m_l、m_a、m_b和$m_{l'}$、$m_{a'}$、$m_{b'}$分别是源图像和着色图像的三个颜色通道的均值，n_l、n_a、n_b和$n_{l'}$、$n_{a'}$、$n_{b'}$表示它们的标准方差。

1. 将源图像原有的数据减掉源图像的均值
2. 得到的新数据按比例放缩，其放缩系数是两幅图像标准方差的比值
3. 得到的l'、a'、b'分别加上目标图像三个通道的均值，得到最终数据

$$L = \frac{n_{l'}}{n_l}(l - m_l) + m_{l'}$$

$$A = \frac{n_{a'}}{n_a}(a - m_a) + m_{a'}$$

$$B = \frac{n_{b'}}{n_b}(b - m_b) + m_{b'}$$

```
//ColorProcessing.cpp
//CImg<float> ColorProcessing::colorTransfer(string path1, string path2)
cimg_forXY(img1, x, y)
{
    img1(x, y, 0) = sd_ratio[0]*(img1(x, y, 0) - mean1[0]) + mean2[0];
    img1(x, y, 1) = sd_ratio[1]*(img1(x, y, 1) - mean1[1]) + mean2[1];
    img1(x, y, 2) = sd_ratio[2]*(img1(x, y, 2) - mean1[2]) + mean2[2];
}
```

4. convert from lab to rgb space

$$\begin{bmatrix} L' \\ M' \\ S' \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & -1 \\ 1 & -2 & 0 \end{bmatrix} \begin{bmatrix} \frac{\sqrt{3}}{3} & 0 & 0 \\ 0 & \frac{\sqrt{6}}{6} & 0 \\ 0 & 0 & \frac{\sqrt{2}}{2} \end{bmatrix} = \begin{bmatrix} l \\ \alpha \\ \beta \end{bmatrix}$$

$$L = L'^{10} \quad M = M'^{10} \quad S = S'^{10}$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 4.4678 & -3.5873 & 0.1193 \\ -1.2186 & 2.3809 & -0.1624 \\ 0.0497 & -0.2439 & 1.2045 \end{bmatrix} \begin{bmatrix} L \\ M \\ S \end{bmatrix}$$

```
CImg<float> lab2rgb(CImg<float>);
```

测试文档

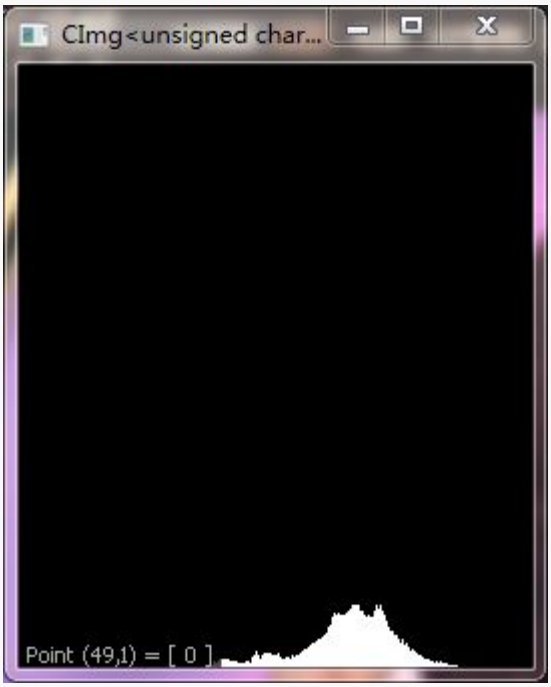
环境：Window7 mingGW

直方图均衡

测试数据 ：彩色*5，灰色*5

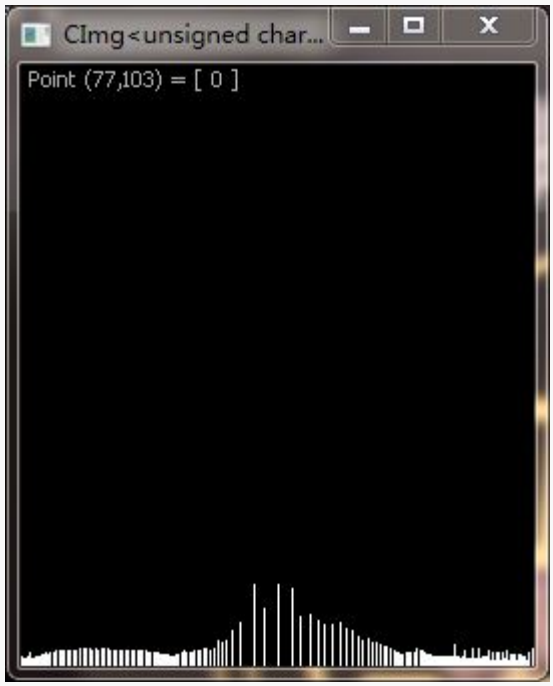
测试结果

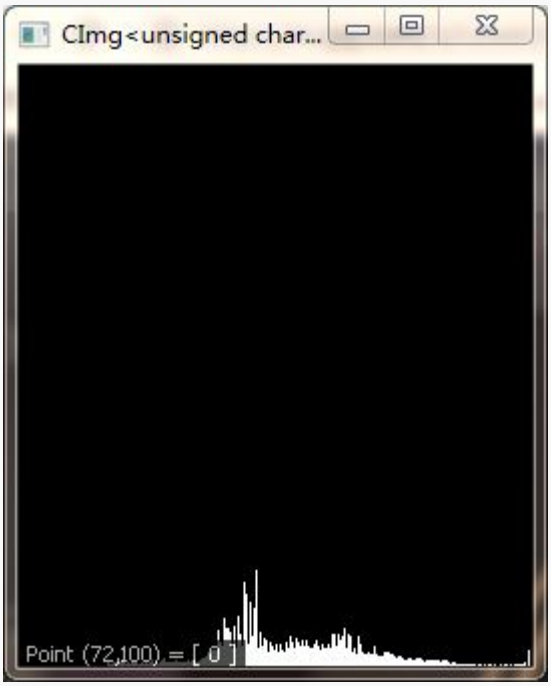
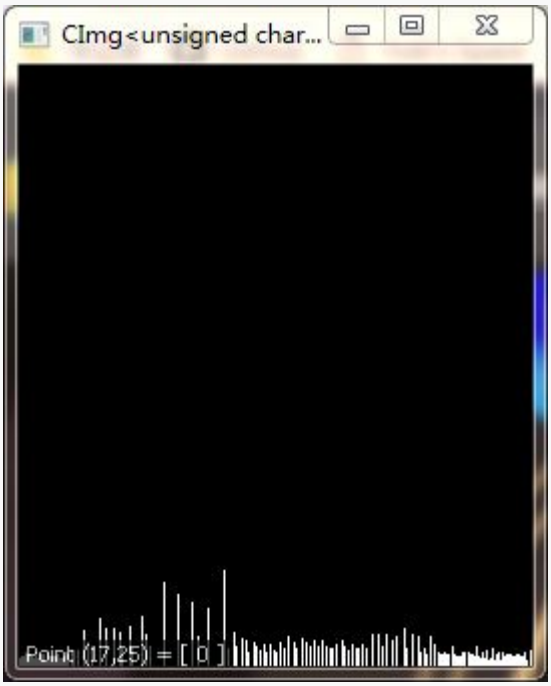
灰度图

原图	处理后
	
 CImg<unsigned char... Point (49,1) = [0]	 CImg<unsigned char... Point (13,224) = [0]
	

原图	处理后
 CImg<unsigned char... Point (128,127) = [0]	 CImg<unsigned char... Point (35,98) = [0]
	

原图	处理后
 CImg<unsigned char... Point (255,296) = [0]	 CImg<unsigned char... Point (176,287) = [0]
 (a)	 (a)

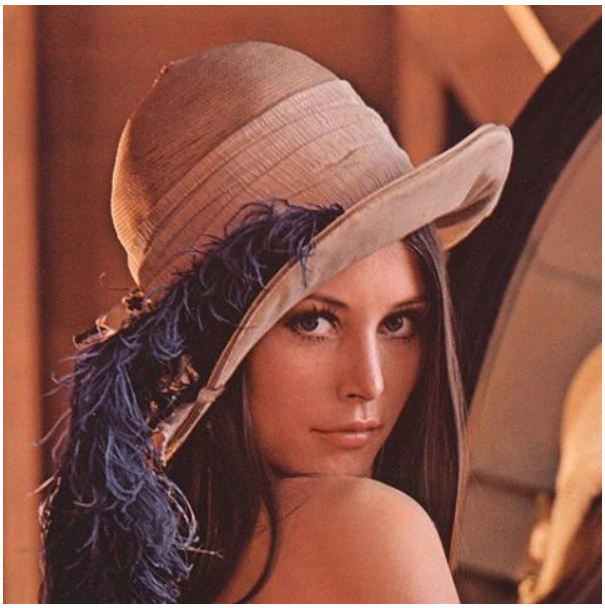
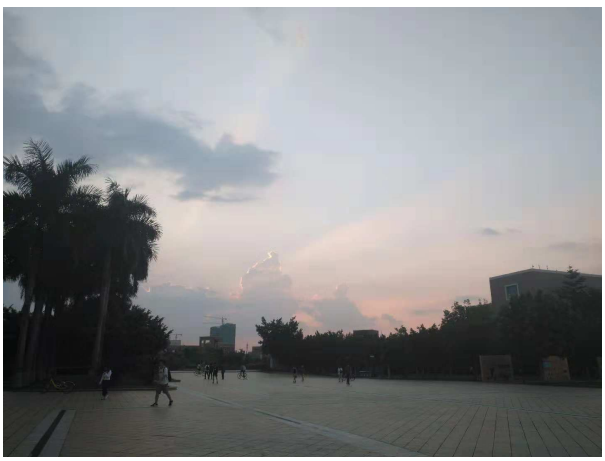
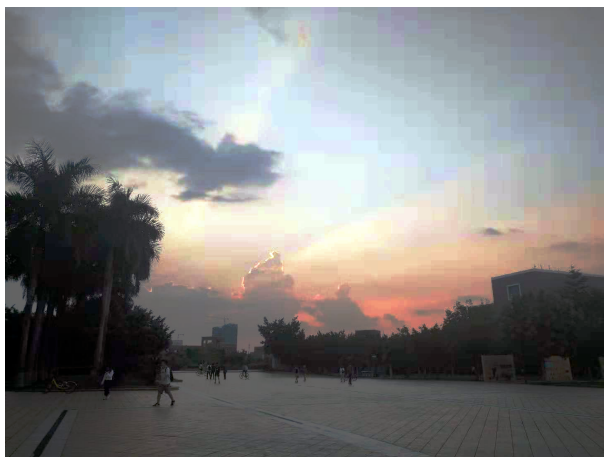
原图	处理后
 CImg<unsigned char... Point (100,276) = [0]	 CImg<unsigned char... Point (77,103) = [0]
	

原图	处理后
	

CImg<unsigned char...
Point (72,100) = [0]

CImg<unsigned char...
Point (17,25) = [0]

彩色图

原图	处理后
	
	
	

原图	处理后
	
	

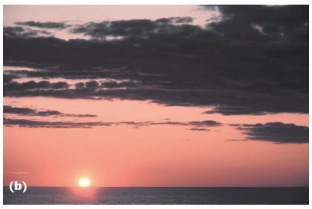
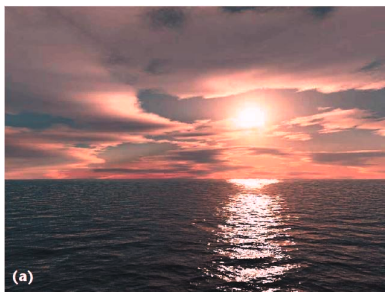
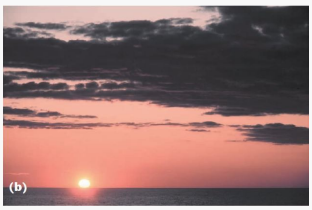
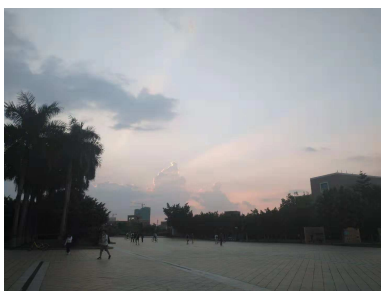
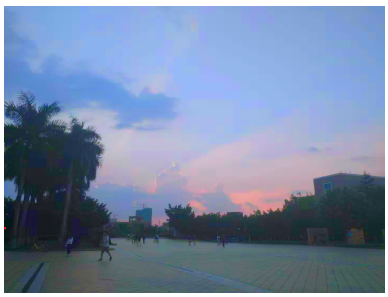
分析

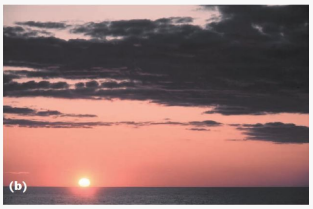
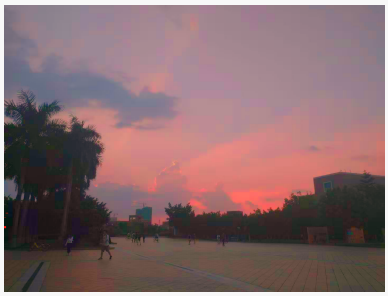
从测试2,3,5可以看出，如果彩色图本身的红黄蓝比较均衡而仅因为灰度值显得图像模糊，直接进行均衡是能够得到较好的去雾效果。但是测试4的夕阳图就完全已经换了个色调。为了保持原有的色调，可以考虑转换为lab空间直接进行分量（亮度）的均衡化化操作。

颜色迁移

测试数据：五对图像

测试结果

序号	图一	图二	结果
1			
2			
3			
4			
5			

序号	图一	图二	结果
6			

分析

如测试3，由于取色图像饱和度太高，结果图像红色区域超过255的部分太多以至于过饱和失真了。以及有集中色块的时候需要局部迁移。