数字图像处理第一次作业

@Fanchy_Lee

2012-4-8

1 白平衡调整

白平衡调整是指矫正图像的偏色的过程。以图像中的在现实中人眼观察下为白色的部分为基准，通过整体调整整幅图像的 r、g、b 的值，使这部分图像回归白色（灰色），亦即 r、g、b 各个分量的值均相等的状态，从而达到矫正色调的偏差。

在白平衡调整的过程中，一般还要保持图像的亮度不会有太大变化。我采取了将基准部分的 r、g、b 的值都调整到 $\frac{R+G+B}{3}$ 这一方法实现亮度的基本不变。这一方法亦保持了该图像在 HSI 空间中的 I 值不变。

具体的代码如下：

```
function color_balance(inPic,outPic,Rw,Gw,Bw);
in=imread(inPic);
meanIntensity=(Rw+Gw+Bw)/3;
in(:,:,1)=double(in(:,:,1)).*meanIntensity./Rw;
in(:,:,2)=double(in(:,:,2)).*meanIntensity./Gw;
in(:,:,3)=double(in(:,:,3)).*meanIntensity./Bw;
imwrite(in,outPic);
```

代码 1. color_balance.m

其中的后三个参量分别代表基准部分的红、绿、蓝三个通道的值。`inPic` 和 `outPic` 分别是输入图片的文件名的字符串（单引号）和处理后要输出的图片的文件名字符串（单引号）。下同。

白平衡调整的图像实例如图 1 及图 2 所示。

2 饱和度增强

饱和度增强需要在 HSV 空间中处理，通过线性点运算调整 S 和 V 的值从而实现色彩饱和度的增强，然后需要将 HSV 空间的数据变换回 RGB 空间。

我将 HSV 空间中的 S 和 V 的值分别乘以 1.5 和 1.1，从而达到增强饱和度的效果。这两个值视具体的图像而定。

相关代码：

```
function S_enhance(inPic,outPic);
in=imread(inPic);
hsv_in=rgb2hsv(double(in)./255);
hsv_in(:,:,2)=hsv_in(:,:,2).*1.5;
hsv_in(:,:,3)=hsv_in(:,:,3).*1.1;
rgb_in=hsv2rgb(hsv_in);
out=uint8(rgb_in.*255);
imwrite(out,outPic);
```

代码 2. S_enhance.m

饱和度增强的实例如图 3 及图 4 所示。

3 直方图均衡化处理

直方图的均衡化就是指对一幅图像进行处理，从而实现其直方图的均衡化（*equalization*），即各个亮度值的点数相近。

我采用了《数字图像处理（第三版）》中式 3.3-8 所示的方法求得转换映射。即：

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) \tag{1}$$

如果，MN 代表整幅图像的像素数，n_k 表示亮度为 k 的点阵数，则

$$p_r(r_k) = \frac{n_k}{MN} \tag{2}$$

这是一个很好的处理直方图均衡化的方法，具有通用性，能够很好地将大多数图像的直方图变换到接近直线。我对这一算法的 MATLAB 实现如下：

```
function myequalize(inPic,outPic);
in = imread(inPic);
[i j k]=size(in);
if k==1;
    [counts ,x]=imhist(in);
    counts_normalized=counts./(i*j);
    [i_counts j_counts] = size(counts_normalized);
    Tmat = tril(ones([i_counts i_counts]));
    s_float = Tmat*counts_normalized*255;
    s_int = uint8(s_float);
    out = s_int(in + 1);
else
    if k==3;
        hsv_in=rgb2hsv(double(in)./255);
        hsv_in(:,:,3) = double(myequalize(uint8(255.*hsv_in(:,:,3)))./255;
        out = uint8(255.*hsv2rgb(hsv_in));
    end
end
imwrite(out,outPic);
```

代码 3. myequalize.m

其中的 `if` 分支结构用来判断图像是否为彩色图像，如果是彩色图像，则先转换到 HSV 空间，对 V 进行均衡化，再转到 RGB 空间。

对于一幅灰度图像的处理结果实例如图 5 及图 6 所示。 其相应的直方图如图 7 及图 8 所示。[1]

4 附图

图 1. 未调整的图像

图 2. 调整后的图像

[1]均衡后的直方图从生成的 `*.jpg` 类型图像文件计算得到，由于 `*.jpg` 采用了一些有损压缩算法，同时也在不改变人眼感觉的前提下，对图像的直方图产生了影响，经实际测试，均衡后的 `*.jpg` 图像比 `*.bmp` 图像更加均衡，这可能是由于其压缩算法对其进行了进一步的均衡。

图 3. 未饱和化的图像

图 4. 饱和化后的图像

图 5. 未均衡的图像

图 6. 均衡后的图像

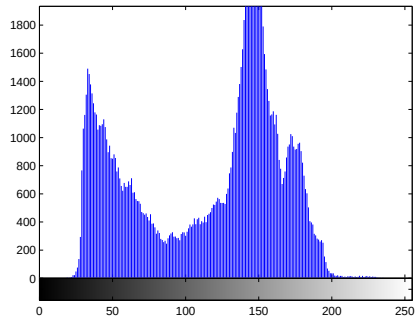

图 7. 未均衡的图像的直方图

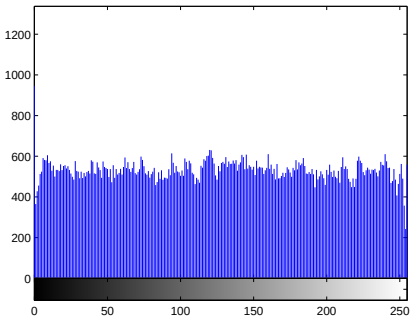

图 8. 均衡后的图像的直方图