基于最邻近匹配的图像上色算法

实验人：李东嵘 16342080

实验平台：Windows10, MATLABR2016a

在本报告中，我们讨论基于最邻近匹配的图像上色算法。众所周知，对一副灰白图像上色是一个数学中的病态问题，即数学上不存在从灰度图到rgb图片的双射。因此，本算法是一种基于风格迁移的上色算法，即程序读入一副灰白照片和一副彩色照片，然后用彩色照片辅助灰白图片上色。

Tomihisa Welsh 在其论文 *Transferring color to greyscale images* 中介绍了一种上述类型的算法，本实验主要基于该算法的实现。与许多传统方法不同，这一算法主要在lab空间对图像进行操作。

下面先对lab空间进行一个简要的介绍。lab如同rgb一样，是一种图像的颜色模型。但与rgb不同，在lab模型中，l代表灰度分量，本身不包含任何颜色的成分，一副lab图像的颜色成分全部被包含在其a，b通道中。这一模型的优势在于，lab三个通道分量均不相关，因此对其中任何一个通道进行操作都不会显著改变其余两个通道的值。此外，借由l通道，我们可以单独地对图像的灰度进行操作。

在介绍完lab空间后，我们开始详述本实验采用的算法。Welsh算法的大致思路为，对灰度图中的每一点p，我们计算其与彩色图中每一点q的亮度和3x3邻域之差的方差（用于保证邻域的光滑度）的加权和，以此作为两点之间的距离，然后，我们选取距离p最短的点Q，将Q的颜色作为p的颜色，如此循环，最终对整幅图片完成上色。

算法的具体描述如下：

输入：灰色图片P，彩色图片**Q**

输出：彩色化的灰色图片**P**

1.对**Q**中的每一点q，生成其**3x3**邻域E_q，将这些邻域存储在矩阵**E**中

2.for p in P

对**Q**中每一点q，分别计算

$$d_l(p,q) = |luminance(p) - luminance(q)|$$

$$d_v(p,q) = var(|E_p - E_q|)$$

$$d_v(p,q) = d_v(p,q)/max(d_v(p,q))$$

（方差归一化）

$$d(p,q) = \lambda_1 d_v(p,q) + \lambda_2 d_l(p,q)$$

其中E_p是**p**的**3x3**邻域。

然后选取$q_1 = argmin_q(d(p,q))$，使**p.a=q1.a, p.b=q1.b.**

3.输出上色后的图片。

在计算距离的过程中，一个有趣的问题是如何对参数$\lambda_1\lambda_2$进行选取，事实上，两个参数的比例决定了我们对光滑度的要求，当λ_2较大时，意味着我们想要加大光滑度的权重。

在原论文中，作者提出，相等的权值具有较好的上色效果，但事实上在实际操作中，我们发现相等权值带来的上色效果并不那么尽如人意，主要是上色后的图片存在大块的红蓝色块，因此，为了使光滑性重要性进一步凸显，我们适当调大光滑性的权值，不妨令$\lambda_1 = 1, \lambda_2 = 3$. 下面我们考察算法的实际效果。

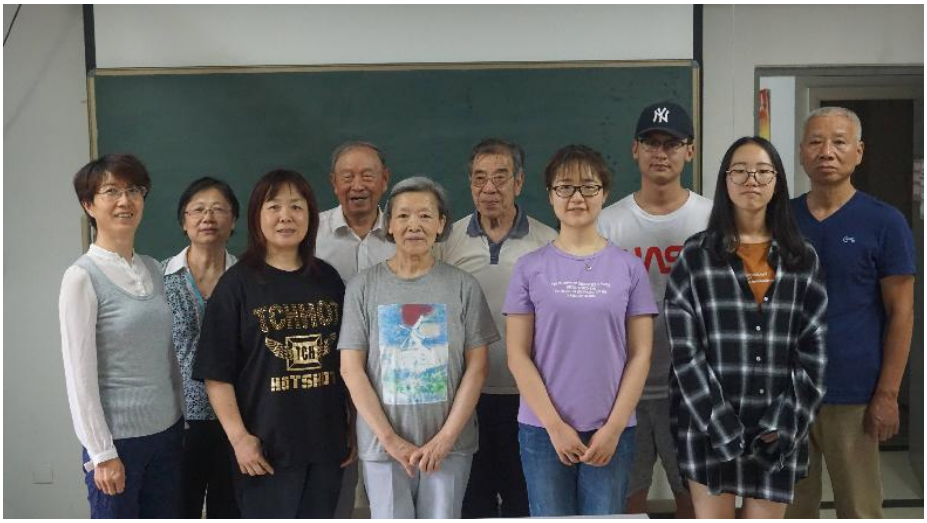
首先，我们从桌面引入两张图像：

```
grey=imread('c:/users/a/desktop/c.jpg');
colorful=imread('c:/users/a/desktop/d.jpg');
```

然后，我们编写程序实现 Welsh 算法，并将其封装成函数，然后再灰白照片上测试其效果。（程序见附录）

我们输入的是一张黑白图片和一张彩色图片，对这一算法进行测试。灰白图片如下：

用于学习的彩色图片如下：



我们用之前已经实现的算法，用彩色图像给灰白图像上色，得到如下结果：

效果并没有做到百分百完美，但至少有了颜色的轮廓，上色的任务算是基本完成。虽然上色后的图片会显得略微老旧模糊，但还算是可以接受的彩色照片。这一定程度上说明了，这个算法有着实际应用价值，但仍有改进的空间。

附录：

1.上色算法的实现（我们将它封装为MATLAB函数）：

```
%function output=Colorize(grey,colorful)
colorful=im2double(colorful);
grey=im2double(grey);
```

```
%grey=rgb2gray(grey);
colorful=rgb2lab(colorful);

tmp=colorful(:,:,1);
tmp=tmp/max(max(tmp));

c_size=size(colorful);
c_height=c_size(1);
c_width=c_size(2);

g_size=size(grey);
g_height=g_size(1);
g_width=g_size(2);

output=zeros(g_height,g_width,3);
output(:,:,1)=grey;
%output=rgb2lab(output);

c_bg=zeros(c_height+2,c_width+2);
g_bg=zeros(g_height+2,g_width+2);

c_bg(2:c_height+1,2:c_width+1)=tmp;
g_bg(2:g_height+1,2:g_width+1)=grey;

Q=zeros(9,c_height*c_width);

for i=1:c_height
    for j=1:c_width

        Q(1,(i-1)*c_height+j)=c_bg(i,j);
        Q(2,(i-1)*c_height+j)=c_bg(i,j+1);
        Q(3,(i-1)*c_height+j)=c_bg(i,j+2);
        Q(4,(i-1)*c_height+j)=c_bg(i+1,j);
        Q(5,(i-1)*c_height+j)=c_bg(i+1,j+1);
        Q(6,(i-1)*c_height+j)=c_bg(i+1,j+2);
        Q(7,(i-1)*c_height+j)=c_bg(i+2,j);
        Q(8,(i-1)*c_height+j)=c_bg(i+2,j+1);
        Q(9,(i-1)*c_height+j)=c_bg(i+2,j+2);

        %Q(3*(i-1)+1:3*(i-1)+3,3*(j-1)+1:3*(j-1)+3)=c_bg(i:i+2,j:j+2);

    end
end


for i=1:length(grey(:))
    x=rem(i,g_height);

    if x==0
        x=g_height;
    end

    y=(i-x)/g_height+1;
    P=g_bg(x:x+2,y:y+2);
    P=P';
    P=reshape(P,[9,1]);
    B=repmat(P,1,c_height*c_width);

    deviation=abs(B-Q);
    std_error=std(deviation);
    variance=std_error.^2;
```

```
    variance=reshape(variance,[c_height,c_width]);
    variance=variance/max(max(variance));
    distances=ones(c_height,c_width)*grey(x,y);
    distances=abs(distances-tmp)+3*variance;
    [a,b]=find(distances==min(min(distances)));
    output(x,y,2)=colorful(a(1),b(1),2);
    output(x,y,3)=colorful(a(1),b(1),3);
    display(i);
    %display(y);
end

% for i=1:g_height
%     for j=1:g_width
%
%         P=g_bg(i:i+2,j:j+2);
%         P=P';
%         P=reshape(P,[9,1]);
%         B=repmat(P,1,c_height*c_width);
%         deviation=abs(B-Q);
%         std_error=std(deviation);
%         variance=std_error.^2;
%         variance=reshape(variance,[c_height,c_width]);
%         variance=variance/max(max(variance));
%         distances=ones(c_height,c_width)*grey(i,j);
%         distances=abs(distances-tmp)+3*variance;
%         [x,y]=find(distances==min(min(distances)));
%         output(i,j,2)=colorful(x(1),y(1),2);
%         output(i,j,3)=colorful(x(1),y(1),3);
%         display(i)
%         display(j)
%
%         distances=0;
%     end
% end

output(:,:,1)=output(:,:,1)*max(max(colorful(:,:,1)));

output=lab2rgb(output);

colorful=lab2rgb(colorful);

%adjust the saturation of the image
output=rgb2hsv(output);
output(:,:,2)=output(:,:,2).^0.65;
output=hsv2rgb(output);
```

Reference:

Tomihisa Welsh, Michael Ashikhmin, Klaus Mueller, *Transferring Color to Greyscale Images,* https://www.researchgate.net/publication/220183710