课程实践 1： 老照片的彩色化

实验目的：设计算法将黑白照片转换为彩色照片，尤其要保证人脸肤色的自然。

实验要求：

1）首先理解问题；

2）再查阅相关的资料，看看已有的工作；

3）设计自己的算法，并编程实现；

4）结果分析和进一步改进的探讨；

实验环境：MATLAB R2018a

原理和方法：

一．伪彩色化处理

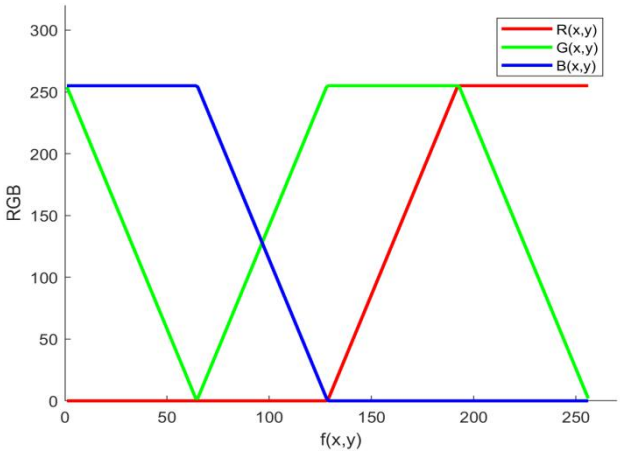
传统方法是伪彩色化处理，实现方法主要有灰度分割法、灰度级-彩色变换法、滤波法等等。其中灰度级-彩色变换法处理的图像视觉效果较好，可以将黑白灰度图像变为具有多种颜色渐变的连续彩色图像。它是将来自传感器的灰度图像送入三个不同特征的红、绿、蓝变换器实施不同变换，然后将三种变换器的不同输出分别送到彩色显示器的红、绿、蓝枪进行显示的技术。

映射关系如下，其中 R(x,y)、G(x,y)、B(x,y)分别表示 R、G、B 通道的颜色值，f(x,y)表示特定点灰度图像的灰度值。

$$R(x,y)=\begin{cases}0 & 0\le f\le 63\\ 0 & 64\le f\le 127\\ 4f(x,y)-510 & 128\le f\le 191\\ 255 & 192\le f\le 255\end{cases}$$

$$G(x,y)=\begin{cases}254-4f(x,y) & 0\le f\le 63\\ 4f(x,y)-254 & 64\le f\le 127\\ 255 & 128\le f\le 191\\ 1022-4f(x,y) & 192\le f\le 255\end{cases}$$

$$B(x,y)=\begin{cases}255 & 0\le f\le 63\\ 510-4f(x,y) & 64\le f\le 127\\ 0 & 128\le f\le 191\\ 0 & 192\le f\le 255\end{cases}$$



二．颜色迁移

目前在灰度图像彩色化的非机器学习算法实现中主流的是 *Welsh* 算法，具体过程如下：

① 将彩色参考图像和目标灰度图像由 RGB 彩色空间都变换到抗相关性的 $l\alpha\beta$ 色彩空间，公式如下：

$$\begin{bmatrix}L\\M\\S\end{bmatrix}=\begin{bmatrix}0.3897 & 0.6890 & -0.0787\\ -0.2298 & 1.1834 & 0.0464\\ 0.0000 & 0.0000 & 1.0000\end{bmatrix}\begin{bmatrix}R\\G\\B\end{bmatrix},\quad \begin{bmatrix}l\\\alpha\\\beta\end{bmatrix}=\begin{bmatrix}1/\sqrt{3} & & \\ & 1/\sqrt{6} & \\ & & 1/\sqrt{2}\end{bmatrix}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & -2\\ 1 & -1 & 0\end{bmatrix}\begin{bmatrix}L\\M\\S\end{bmatrix}$$

② 计算源彩色图像(s)与目标图像(t)的亮度均值 (L_s,L_t) 及标准偏差 (σ_s,σ_t)。

③ 对源图像亮度进行线性变换，使源图像和目标图像的亮度均值和方差一致，防止因亮度差异过大造成误匹配，计算出的结果为L_s'，即

$$L_s' = \frac{\sigma_t}{\sigma_s}(L_s - \bar{L}_s) + \bar{L}_t$$

③ 计算彩色图像和目标灰度图像每个像元点的亮度值▽L 和 5×5 邻域内的方差值▽D，加权求和，得到图像像元的距离值 $L_2 = \frac{\nabla L + \nabla D}{2}$；

④ 对于灰度图像的每个像素 $s(i,j)$，找到彩色参考图像中距离值与该像素的距离值最相近的一像素点 $t(i',j')$，则它就是与灰度图像的当前点匹配的颜色迁移点；

⑤ 将彩色参考图像 $t(i',j')$ 的 α,β 值传输给目标灰度图像的 $s(i,j)$ 点，同时保留 $s(i,j)$ 点 l 通道的亮度值；

⑥ 将转换后的源图像从 $l\alpha\beta$ 彩色空间转换回 RGB 彩色空间（公式略），完成颜色迁移。

实验代码和结果：

一、灰度级-彩色变换法

算法的 MATLAB 代码如下：

```
clear all
A=imread('图片1.png');
A=A(:,:,1);
[m n]=size(A);
B=zeros(m,n,3);
for x=1:m
    for y=1:n
        if A(x,y)<=63
            B(x,y,1)=0;
            B(x,y,2)=254-4*A(x,y);
            B(x,y,3)=255;
        else if A(x,y)<=127
                B(x,y,1)=0;
                B(x,y,2)=4*A(x,y)-254;
                B(x,y,3)=510-4*A(x,y);
        else if A(x,y)<=191
                B(x,y,1)=4*A(x,y)-510;
                B(x,y,2)=255;
                B(x,y,3)=0;
            else
                 B(x,y,1)=255;
                 B(x,y,2)=1022-4*A(x,y);
                 B(x,y,3)=0;
            end
            end
        end
    end
end
```

```
figure()
subplot(1,2,1),imshow(A)
title('灰度图像')
subplot(1,2,2),imshow(uint8(B));
title('伪彩色处理')
```

实现效果如下：

灰度图像

伪彩色处理

从实现效果我们可以看出，这种方法的优点是快速、简单，且不改变原始图像的信息量，但缺点是伪彩色化效果与真实颜色差距很大，完全不符，因为这只是一种为了提高图像分辨率的图像增强技术，这种技术一般大量用于医学成像、遥感测绘等领域。

二、Welsh 算法

算法的 MATLAB 代码如下：

```
clear all
A=imread('1.jpg');%灰度图片
B=imread('2.jpg');%彩色图片
A=A(:,:,1);
A=histeq(A);
[l1,m1,s1]=match(A,B);
C=lms2rgb(l1,m1,s1);
subplot(1,3,1),imshow(A)
title('灰度目标图像')
subplot(1,3,2),imshow(C);
title('Welsh算法处理')
subplot(1,3,3),imshow(B)
title('彩色源图像')
```

```
function [l,m,s]=rgb2lms(A) %RGB转lαβ
R=A(:,:,1);
G=A(:,:,2);
B=A(:,:,3);
L=double(0.3811*R+0.5783*G+0.0402*B);
M=double(0.1967*R+0.7244*G+0.0782*B);
S=double(0.0241*R+0.1288*G+0.8444*B);
l=0.5774*L+0.5774*M+0.5774*S;
m=0.4082*L+0.4082*M-0.8165*S;
s=0.7071*L-0.7071*M;
end
```

```
function P=LD(l)  %求(∇L+∇D)/2
[M,N]=size(l);
l=double(l);
L=zeros(M,N);D=L;
for i=3:M-2
    for j=3:N-2
        sq1=l(i-2:i+2,j-2:j+2);
        L(i,j)=mean(sq1(:));
        D(i,j)=std(sq1(:),1);
    end
    sq2=l(i-2:i+2,1:3);
    L(i,1)=mean(sq2(:));
    D(i,1)=std(sq2(:),1);
    sq3=l(i-2:i+2,1:4);
    L(i,2)=mean(sq3(:));
    D(i,2)=std(sq3(:),1);
    sq4=l(i-2:i+2,end-3:end);
    L(i,end-1)=mean(sq4(:));
    D(i,end-1)=std(sq4(:),1);
    sq5=l(i-2:i+2,end-2:end);
    L(i,1)=mean(sq5(:));
    D(i,1)=std(sq5(:),1);
end
```

```
for j=3:N-2
    sq6=l(1:3,j-2:j+2);
    L(1,j)=mean(sq6(:));
    D(1,j)=std(sq6(:),1);
    sq7=l(1:4,j-2:j+2);
    L(2,j)=mean(sq7(:));
    D(2,j)=std(sq7(:),1);
    sq8=l(end-3:end,j-2:j+2);
    L(end-1,j)=mean(sq8(:));
    D(end-11,j)=std(sq8(:),1);
    sq9=l(end-2:end,j-2:j+2);
    L(end,j)=mean(sq9(:));
    D(end,j)=std(sq9(:),1);
end
%左上
a=l(1:3,1:3);L(1,1)=mean(mean(a));D(1,1)=std(a(:),1);
a=l(1:4,1:3);L(2,1)=mean(mean(a));D(4,1)=std(a(:),1);
a=l(1:3,1:4);L(1,2)=mean(mean(a));D(1,2)=std(a(:),1);
a=l(1:4,1:4);L(2,2)=mean(mean(a));D(2,2)=std(a(:),1);
%右上
a=l(M-2:M,1:3);L(M,1)=mean(mean(a));D(M,1)=std(a(:),1);
a=l(M-3:M,1:3);L(M-1,1)=mean(mean(a));D(M-1,1)=std(a(:),1);
a=l(M-2:M,1:4);L(M,2)=mean(mean(a));D(M,1)=std(a(:),1);
a=l(M-3:M,1:4);L(M-1,2)=mean(mean(a));D(M-1,2)=std(a(:),1);
%左下
a=l(1:3,N-2:N);L(1,N)=mean(mean(a));D(1,N)=std(a(:),1);
a=l(1:4,N-2:N);L(2,N)=mean(mean(a));D(2,N)=std(a(:),1);
a=l(1:3,N-3:N);L(1,N-1)=mean(mean(a));D(1,N-1)=std(a(:),1);
a=l(1:4,N-3:N);L(2,N-1)=mean(mean(a));D(2,N-1)=std(a(:),1);
%右下
a=l(M-2:M,N-2:N);L(M,N)=mean(mean(a));D(M,N)=std(a(:),1);
a=l(M-3:M,N-2:N);L(M-1,N)=mean(mean(a));D(M-1,N)=std(a(:),1);
a=l(M-2:M,N-3:N);L(M,N-1)=mean(mean(a));D(M,N-1)=std(a(:),1);
a=l(M-3:M,N-3:N);L(M-1,N-1)=mean(mean(a));D(M,N-1)=std(a(:),1);
```

```
P=(L+D)/2;
end
```

```
function [l1,m1,s1]=match(A,B) %匹配像素点，A为灰度图，B为彩色图
[M,N]=size(A);
[l,m,s]=rgb2lms(B);
A=double(A);
meanA=mean(mean(A));meanB=mean(mean(l));
stdA=std(A(:),1);stdB=std(l(:),1);
A1=stdB/stdA*(A-meanA)+meanB;
l1=A1;m1=zeros(M,N);s1=m1;
PA=LD(A1);PB=LD(l);
for i=1:M
    for j=1:N
        PAB=abs(PB-PA(i,j));
        [x,y]=find(PAB==min(min(PAB)));
        m1(i,j)=m(x(round(end/2)),y(round(end/2)));
        s1(i,j)=s(x(round(end/2)),y(round(end/2)));
    end
end
end
```

```
function C=lms2rgb(l,m,s)  %lαβ转RGB
L=0.5774*l+0.4082*m+0.7071*s;
M=0.5774*l+0.4082*m-0.7071*s;
S=0.5774*l-0.8165*m;
R=4.4679*L-3.5873*M+0.1193*S;
G=-1.2186*L+2.3809*M-0.1624*S;
B=0.0497*L-0.2439*M+1.2045*S;
[m,n]=size(l);
C=zeros(m,n,3,'uint8');
C(:,:,1)=uint8(R);
C(:,:,2)=uint8(G);
C(:,:,3)=uint8(B);
end
```

实现效果如下：

灰度目标图像

Welsh算法处理

彩色源图像

这是一种自动的、全局的彩色化方法，当两幅图像中亮度相同的像素也具有相同的色彩时，这种算法可以取得很好的效果。但当源图像和目标图像的相应区域不具有相同亮度值时，这种匹配就会发生错误，导致结果不理想。

接下来我们对该算法进行进一步探索，首先鉴于上面的彩色化图片颜色较暗淡，尝试先将彩色源图片的亮度、对比度和饱和度都增强再做颜色迁移，看是否能得到更好的效果。

灰度目标图像　　Welsh算法处理　　彩色源图像

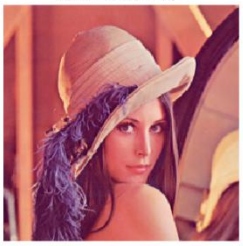


可以看到，目标图片只是整体色调鲜艳了点，但是色彩还是非常单调，而且出现了很多噪声点。我们再用不同类型的图片测试一下：

1）标准测试图片 lenna——人像图片

灰度目标图像　　Welsh算法处理　　彩色源图像





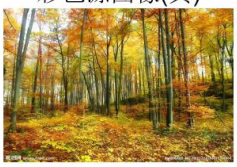
将源图片和目标图片都选为同一张，结果却也并不尽如人意，只是基本上恢复了肉色部分的颜色迁移，而头发的紫色完全没有显现出来。

2）树林——风景图片（一个色调）

灰度目标图像　　Welsh算法处理　　彩色源图像(绿)



Welsh算法处理　　彩色源图像(黄)

对同一张灰度图片，用不同色调的源图片做色彩迁移，得到不同色调的目标图像，但是细节处的颜色变化如枝干的颜色则完全没有成功迁移。

3）天空草地——静物图片（两个色调）

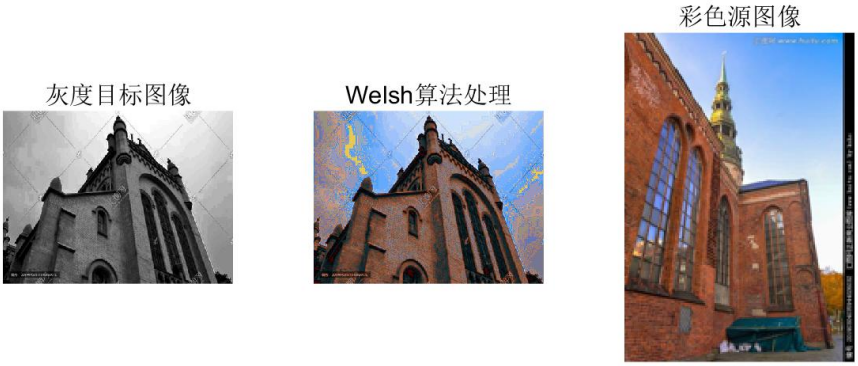
灰度目标图像　　Welsh算法处理　　彩色源图像



这张图片的恢复效果整体较好，但是仍有部分区域匹配错误，即部分天空被迁移成了蓝

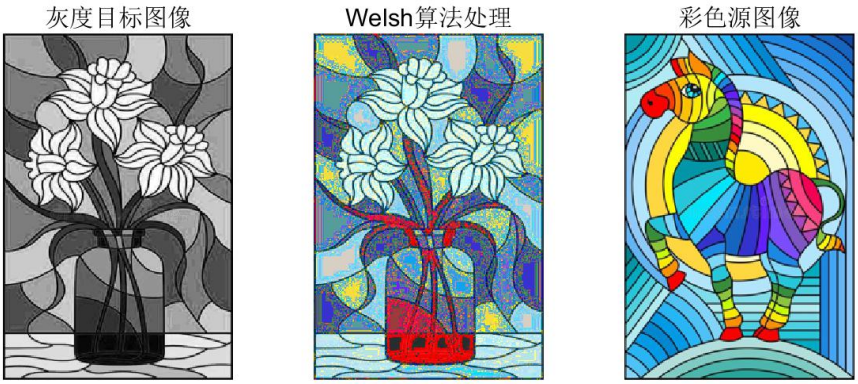
色，部分草地被恢复成了蓝色。

4）教堂——建筑图片（两个色调）



同前例，总体匹配效果尚可，天空有部分区域误配。

5）琉璃画——色块图片（多个色调）

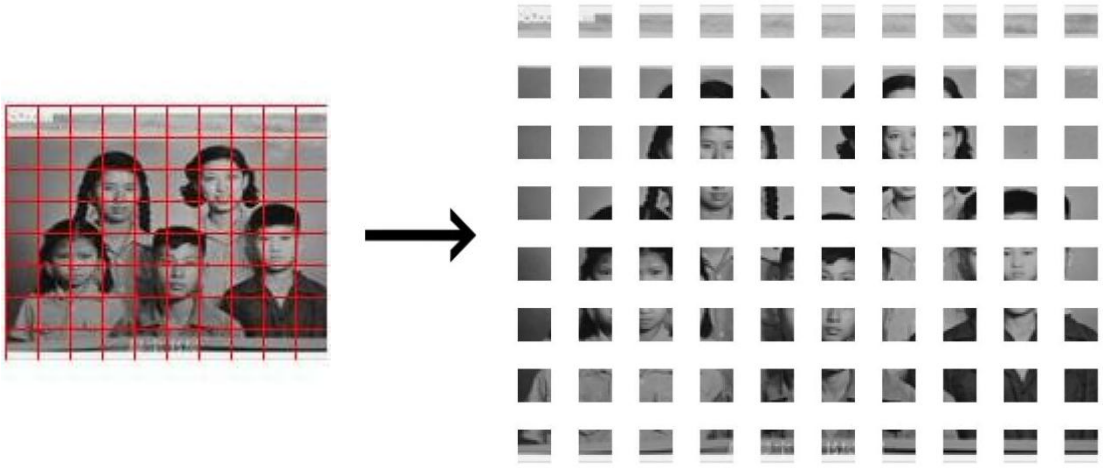


撇开由于图像分辨率造成的噪声，这种纯色块图片有较好的迁移效果，不同灰度迁移到了不同颜色，其实感觉这个跟伪彩色化有点像了？？

结果分析与实验改进：

Welsh 算法对单色调的彩色图片做色彩迁移效果较好，这点不难理解因为不同的颜色灰度化之后的灰度值可能是一样的，因此对该点做图像迁移的颜色完全取决于彩色源图像的匹配点的颜色，对于不同色块之间映射到灰度值有明显差异的彩色图片做迁移，效果也会较为完整连续，不会出现同一区域颜色混杂的效果。因此彩色源图像的选择非常关键，不仅风格要相近，不同颜色的明暗分布也要基本一致才不会导致颜色匹配错误。

针对这个问题，Welsh 算法可以有进一步的改进，即首先对图像进行分块（见下图，代码见附录），指定每个区域的色系，比如在 Lenna 图片中，指定头发区域迁移紫色系列的颜色，这样就能保证图片能够迁移到正确的颜色。但是这一步骤如果不使用机器学习的话，需要手动基于经验设置颜色，其实这与 PS 给图像上的原理非常相似。



如上图示意，选择合适的窗口大小将灰度目标图像和彩色源图像分别进行分块，将灰度图像恢复后目标颜色相同的块合并成同一区域，将彩色源图片色彩相同的块也合为一个区域，然后指定灰度图像和彩色图像之间的区域匹配，比如天空草原的例子中指定灰度图片的天空和草地分别匹配彩色图片的天空和草地，就能得到较准确的结果。但是这种操作人工成分明显、工作量大且代码不具有普适性，需要具体图片具体分析，因此这里只是简单描述一下想法，不做代码实现与验证。

同时，朴素的 Welsh 算法的效率非常低，处理一次 256×256 的灰度和彩色图片都需要几分钟，这是因为对于灰度图像的每个像素点，都要在彩色图像中作全局搜索。有一种提高速度的方法是减少采样点，对彩色图片进行均匀的等距选取采样点阵，如每 3 行取一行等距取 200 个点，这样就能大大减少搜索匹配点的时间。或者如前述改进的首先进行图像分块，那么相当于只需要做局部搜索就可以了，也能够在一定程度上提高搜索效率。

Welsh 算法是非机器学习算法，关于尝试神经网络，MATLAB 的效果可能不是很好，运行效率会很慢，用 Python 实现会更合适。

参考文献：

[1]和晓军,乔寅,基于 Welsh 算法的灰度图像彩色化的研究,计算机应用与软件,2014,12
[2]刘勇,关于彩色化灰度图像的算法研究及实现,山东大学硕士学位论文,2007,4,12
[3]灰度图像彩色化算法研究 - Trent1985 的专栏 - CSDN 博客
https://blog.csdn.net/trent1985/article/details/9499385

附录：

图像划线分割

```
clc; clear all; close all;
A=imread('图片1.png');
A=A(:,:,1);
A=imresize(A,[128,160]);
[m n]=size(A);
w=16; %控制分块窗口的大小，16×16
M=m/w;N=n/w;  %行列间隔块个数
% 区域块分割
t1=(0:M-1)*w+1;t2=(1:M)*w;
t3=(0:N-1)*w+1;t4=(1:N)*w;
figure()
imshow(A);
hold on
for i=1:M
    for j=1:N
        x=t1(i):t2(i);
        y=t3(j):t4(j);
        rectangle ('Position', [t3(j) t1(i) length(x) length(y)], ...,
            'EdgeColor','r','LineWidth',0.05); %绘制矩形分割格子，红色
    end
end
```

图像分块

```
A=imread('图片1.png');
A=A(:,:,1);
A=imresize(A,[128,160]);
[m n]=size(A);
w=16; %窗口大小
M=m/w;N=n/w;
t1=(0:M-1)*w+1;t2=(1:M)*w;
t3=(0:N-1)*w+1;t4=(1:N)*w;
figure()
k=0;
for i=1:M
    for j=1:N
        temp=A(t1(i):t2(i),t3(j):t4(j),:);
        k=k+1;
        subplot(M,N,k);
        imshow(temp);
    end
end
```