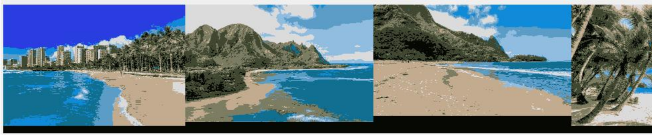
1.1 RGB colorspace

Use k-means to cluster the colors in given images into ten unique colors.

```
load('beachall.mat')
I = double(I);
A = reshape(I,194*1667,3);
k = 10;
[mc,n] = kmeans(A,k);
m=reshape(mc,size(I,1),size(I,2));
n=n/255;
clusteredImage=label2rgb(m,n);
imshow(clusteredImage)
```



Ten unique colors:

10x3 double

	1	2	3
1	0.4364	0.5675	0.6132
2	0.0576	0.4971	0.7053
3	0.2002	0.2524	0.1882
4	0.8436	0.8605	0.8814
5	0.0152	0.0201	0.0100
6	0.7724	0.7087	0.6487
7	0.1504	0.2616	0.8858
8	0.7245	0.6048	0.4170
9	0.4155	0.4280	0.3192
10	0.4412	0.6972	0.8927

Mapping using nearest neighbor model:

```
new = imread("newbeach.jpg");
B = double(new);
C = reshape(B,720*1080,3);
nnmodel = fitcknn(A,mc);
predict = nnmodel.predict(C);
y = reshape(predict,720,[],1);
y = label2rgb(y,n);
imshow(y)
```

1.2 CIE L*a*b* colorspace

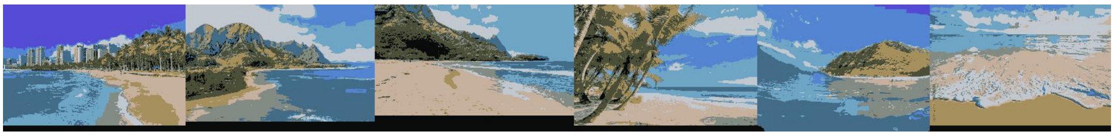
Convert the image from rgb to lab, and then implement k-means method.

```
I = double(I);
Ilab = rgb2lab(I);
A = reshape(Ilab,194*1667,3);
k = 10;
[mc,n] = kmeans(A,k);
```

10x3 double

1	2	3
81.4834	0.4226	-0.0977
44.2226	-6.4872	-18.4422
51.2239	16.6674	-53.9383
67.3901	4.3789	24.4618
36.6593	-5.7933	15.1434
66.5443	-9.7447	-19.4462
59.0194	0.8521	-37.7260
3.9904	-0.4805	0.0835
62.7608	-0.8209	4.1526
40.0388	45.1230	-70.4763

```
m=reshape(mc,size(Ilab,1),size(Ilab,2));
n=n/100;
for i=1:size(m,1)
    for j=1:size(m,2)
        num = m(i,j,1);
        m(i,j,1) = n(num,1);
        m(i,j,2) = n(num,2);
        m(i,j,3) = n(num,3);
    end
end
clusteredImage=lab2rgb(m);
imshow(clusteredImage)
```



```
new = imread("newbeach.jpg");
newd = double(new);
P = rgb2lab(new);
B = rgb2lab(newd);
C = reshape(B,720*1080,3);
nnmodel = fitcknn(A,mc);
predict = nnmodel.predict(C);
y = reshape(predict,720,[],1);
for i=1:size(y,1)
    for j=1:size(y,2)
        num = y(i,j,1);
        y(i,j,1) = n(num,1);
        y(i,j,2) = n(num,2);
        y(i,j,3) = n(num,3);
    end
end
y1 = lab2rgb(y);
imshow(y1)
```

The differences between two versions are the colors, the edges and shapes of objects.
In the RGB version, the blue color of the sky and the sea seems to be deeper and the edges and shapes of the cloud and trees can not be presented precisely.
In the L*a*b version the blue color seems to be lighter, and the edges and shapes of cloud, trees and sand are presented more real, closer to the original image.

2.1 parameters

1 conv 3x3 kernal, 16 output channels, stride 1 padding 1. 16 kernals
Size. (32-3+2)/1+1=32 so 32x32x16
2 RelU layer
3 4x4 max pooling stride 4
Size. (32-4)/4+1=8 so 8x8x16
4 conv 3x3x16 kernal, 32 output channels, stride 1 padding 1. 32 kernals
Size. (8-3+2)/1+1=8 so 8x8x32
5 RelU layer
6 2x2 max pooling stride 2
Size. (8-2)/2+1 = 4 so 4x4x32
7 fully connected 4 × 4 × 32 input nodes to 5 output nodes
8 a Softmax layer.

input is 32 × 32 grayscale images, so the number of channel of input of conv1 is 1.
1. Weights of conv=3*3*1*16=144 Biases of conv= 16
2. 0 There are no parameters associated with a RelU layer.
3. 0. There are no parameters associated with a Max Pool layer.
4. Weights of conv =3x3x16x32=4608 Biases of conv =32
5. 0 6. 0
7. Weights of fc=4x4x32x5=2560 Biases of fc=5
8. Weights=5x5=25 Biases=5
Parameters in total=144+16+4608+32+2560+5+25+5=7395

2.2 receptive field

2nd pool.Rf=2. equals to the size of this layer
2nd conv.Rf=(2-1)x1+3=4
1st pool.Rf=(4-1)x4+4=16
1st conv.Rf=(16-1)x1+3=18
So the receptive field of the features in the final max pooling layer is 18.

3.1 least-square estimate

split the image into three color channels. We have the transformation:

T(r) = bi*r + (1 − bi)r^2.

b=(r-r^2)\(T-r^2). For each b1,b2,b3 in 3 channels

```
r = imread("butterfly_color.jpg");
T = imread("butterfly_blobs.jpg");
```

```
r = double(r)/255;
T = double(T)/255;
for i= 1:3
    xm = (r(:,:,i)-r(:,:,i).^2);
    xm = xm(:);
    ym = (T(:,:,i)-r(:,:,i).^2);
    ym = ym(:);
    b(i) = xm\ym;
    new(:,:,i)=b(i)*r(:,:,i)+(1-b(i))*r(:,:,i).^2;
end
imshow(new)
```

the values of b1,b2,b3

	1	2	3
1	-2.0556	1.6569	0.7567

Transform using the estimated transformation:

In this image, the color of everything are different from the correct colors.

3.2 robust method RANSAC

With the RANSAC method, we can find the case with the most inliers and git rid of the influence of outliers.

```
for i=1:3
    xm = (r(:,:,i)-r(:,:,i).^2);
    xm = xm(:);
    ym = (T(:,:,i)-r(:,:,i).^2);
    ym = ym(:);
    bestins = 0;
    bnd = 0.1;
    iters = 100;
    nn = length(xm);
    for ii = 1:iters
        id = randi(nn);
        a_test = xm(id)\ym(id);
        ins = sum(abs(ym-(a_test*xm+(1-a_test)*xm.^2))<bnd);
```

```
        if ins>bestins
            bestins = ins;
            a_rs = a_test;
        end
    end
    b(i)=a_rs;
    new(:,:,i)=b(i)*r(:,:,i)+(1-b(i))*r(:,:,i).^2;
end
imshow(new);
```

1x3 double

	1	2	3
1	-3.3494	2.0241	0.7924

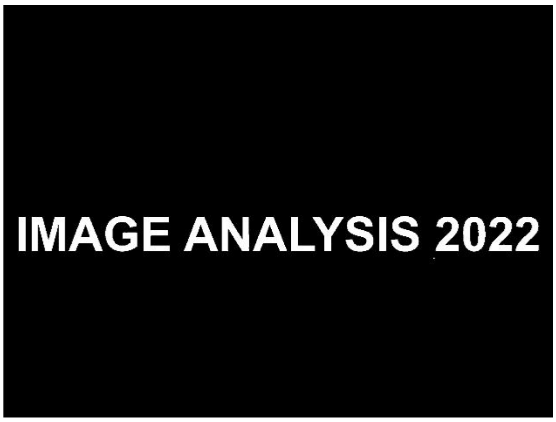
Transform using the estimated transformation:

Now in this image, all the colors are correct, without the influence of outliers.

4 sunset

First, extract the text in the image with threshold in a channel of L*a*b.

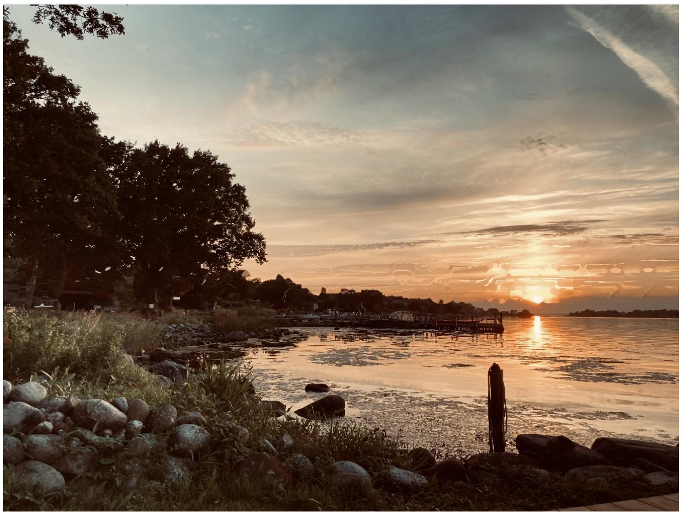
```
I = imread("sunset.jpg");
%I = double(I);
Ilab = rgb2lab(I);
aChannel=Ilab(:, :, 2);
for i=1:size(I,1)
    for j=1:size(I,2)
        if  aChannel(i,j)>40
            In(i,j)=1;
        else
            In(i,j)=0;
        end
    end
end
```



```
imshow(In)
```

then use image inpaintCoherent to restore the text region.

```
mask = logical(In);
mask= imgaussfilt(double(mask),1);
J = inpaintCoherent(I,logical(mask));
imshow(J)
```



5 Color the image

I tried to extract the colors by taking the means of RGB values of the regions with every color.

then divide the regions by bwlabel which can find the connected areas. At last, paint the different regions of the picture with different colors. But it's hard to identify the numbers written in the pictures. So it is half-done.

```
I1 = I(1:1800,:,:);
I2 = I(1801:2400,:,:);
I1g = rgb2gray(I1);
I1bi = imbinarize(I1g);
I2g = rgb2gray(I2);
for k=1:6
    color(:,:,:,k)=I2((501-25*(k-1)):(510-25*(k-1)),121:130,:);
    meanx(:,:,:,k)=mean(color(:,:,:,k));
    meany(:,:,k)=mean(meanx(:,:,:,k));
end
L = bwlabel(I1bi,4);
for i=1:size(I1,1)
    for j=1:size(I1,2)
        for k=1:6
        if L(i,j)==k+1
            I(i,j,1)=meany(1,:,k);
            I(i,j,2)=meany(2,:,k);
            I(i,j,3)=meany(3,:,k);
```

```
            end
        end
    end
end
```

6 finds the mites

First, I try to identify the mites by color, use binarization to keep those with dark color and remove those with light color. And then try to identify them by their shapes. The results are marked in the picture.

```
I = [I7,I8,I9;
    I10,I11,I12];
figure(1)
imshow(I)
g1 = rgb2gray(I);
BW = imbinarize(g1,0.4);
bw = (~BW);
bw = imfill(bw,'holes');
figure(2)
imshow(bw)
 a1 = imfindcircles(bw,[15 50],"Sensitivity",0.85,"EdgeThreshold",0.4);
 b1 = insertMarker(I,a1);
figure(3)
imshow(b1)
```