Lab 5

Manwen Li

Experiments

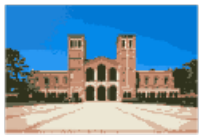
Choosing K = 2, 3, 4, 5, 10, 20, 30, 40, 50, the segmented images in which each pixel has the value of the centroid it was assigned to are:


K = 2


K = 3


K = 4


K = 5


K = 10


K = 20


K = 30


K = 40


K = 50

When K = 2, the reconstructed picture is represented by two colors: blue and beige/taupe. The cool and dark tones in the picture are represented by blue while the warm and light tones are represented by beige.

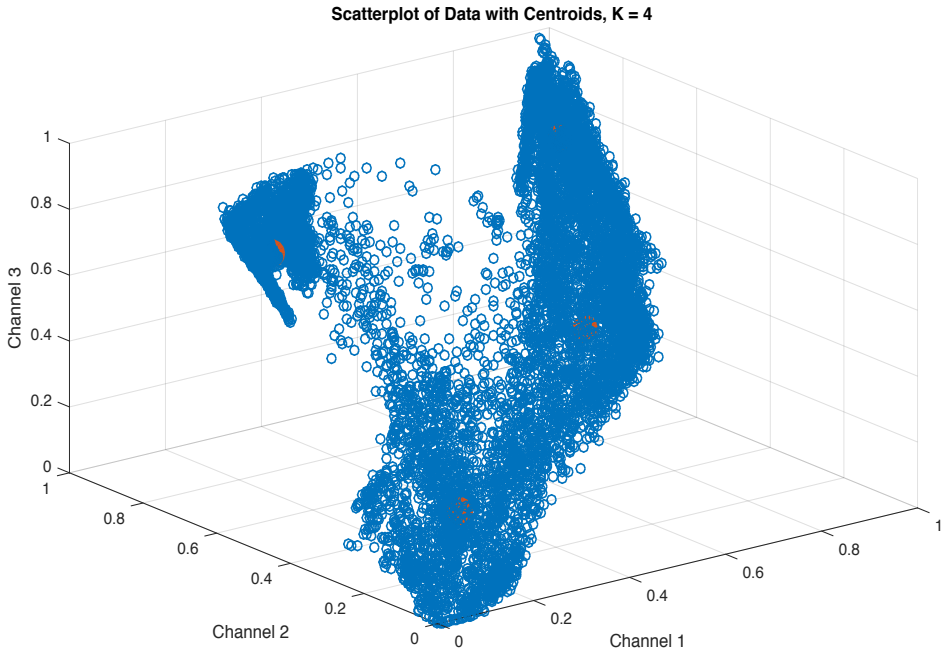
When K = 3, there are three parts in the picture: blue, dark green, and beige. Also, there exists contrast between light and shadow, where the shadow, as well as the grass and trees, is represented by the color dark green. The light is represented by beige, while the sky alone is represented by blue.

When K = 4 and 5, we start noticing more details and differences between colors. For example, we see that the architecture is in the color brown, the color differences of the textiles on the ground, and color differences between trees (darker green in the original picture) and grass (light green in the original picture).

Starting from K = 10, more details appear in the reconstructed images. For example, there are different levels of blues in the sky, more clear differences in the textiles on the ground, color differences between shadows in the building and trees (they are of the same color in the image K = 4).

In my opinion, K = 4 and 5 give the best segmentation, because a) we can notice the clear differences between different regions, such as building, sky, vegetation, and ground; b) the same region is not represented by different levels of the same color, for example, the sky is represented by only one level of blue instead of different levels of blues as in K = 10, containing some redundant color differences. K = 4 and 5 will give us an image that effectively distinguishes the different regions yet save us some time cost with a low value of K.

1. Here is the 3D scatterplot of the data using the three color channels, using K = 4. The centroids are the 4 red dots.



Centroid 1 has the lowest values of all Channel 1, Channel 2, Channel 3, which means that it corresponds to a dark object. From the image K = 4, it should represent the tree, grass, and the dark shades of the building.

Centroid 2 has the largest values of all Channel 1, Channel 2, and Channel 3, which means that it corresponds to a bright object. It should represent the bright ground in front of the building in the image K = 4.

Centroid 4 consists of high values of Channel 2 and Channel 3 and very low value of Channel 1, which corresponds to the RGB values of sky blues family; thus, Centroid 4 represents the sky.

Centroid 3 then represent the brown color of the building, since it has a moderate value of each channel of R, G, B.

2. Adding the zero-mean Gaussian noises, the segmented image using K = 20 becomes

Original, with noise Segmented, with noise Segmented, no noise

We see that adding Gaussian noise gives the segmented image using K-means unwanted "dots". In the "segmented, no noise" image, the pixels in each cluster are assigned with the value of corresponding centroid, creating different "regions" of colors, having the same color within each region. But adding noise is like changing values of the pixels in each region, which are supposed to have the values of the centroids. Thus the image becomes less clear, and there appeared different colors within the same region.

The "Original, with noise" image is the original Royce hall image with Gaussian noise. Comparing the left two images we find that segmentation "reduces" the appearance of noise from the original image with noise. The appearance of noise pixels in the sky and the ground has been reduced a little.

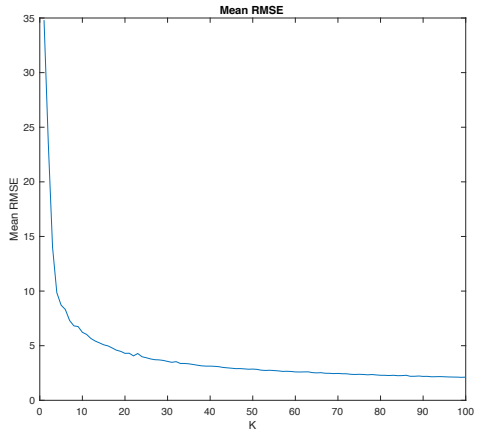
6. We reconstruct the RMSE of different values of K as images:

K = 2 K = 5 K = 20 K = 50 K = 100

The bright part in the RMSE denotes the higher error, or the discrepancy between the original image and the segmented image using the K-means algorithm, while the dark part in the RMSE denotes the smaller error. The brighter the pixel is in the RMSE images, the larger the error it has, or the more different the pixel in the segmented image is from the original image.

For small values of K (K<5), the RMSE images are bright, which means that the difference between the segmented image and the original image is big. K = 5 gives a RMSE image with a moderate brightness, and is noticeably darker in the shadow parts of the building. The RMSE images of K = 20, 50, and 100 are of the similar darkness, which means that the RMSE is converging to a certain level.

7. The values of RMSE as a function of K:

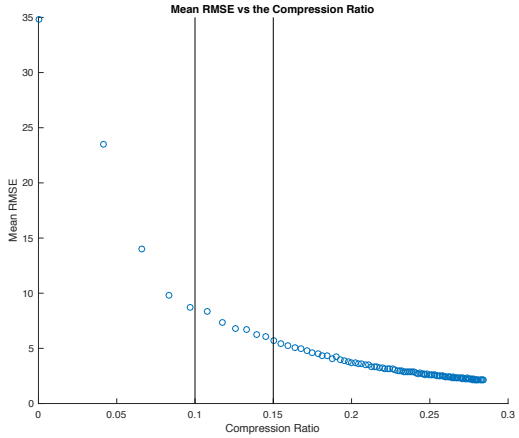


The mean RMSE decreases sharply as K increase from 0 to around 4, and then slowly decreases as K increase from 4 to 100. When K gets closer and closer to number of pixels, the RMSE will probably converge to 0.
Also, from K = 20, the absolute value of the slope of RMSE become very small, which explains in #6 why the RMSE image of K = 20, 50, 100 are not very different from each other.

K-means can be used for image compression because according to the segmented image in #2 and RMSE image in #6, the original image can be well represented by 20 clusters and centroids. The original 96 * 144 pixels' image is coded by 3*8*96*144 = 331776 bits, but with 20 clusters we only need 96*144*log2(20) + 3*8*20 = 60226 bits to reconstruct the segmented image. If we want K clusters, we need 96*144*log2(K) + 3*8*K bits.

The role of parameter K: K determines how clear we want our segmented image to be. The larger the K is, the more clusters and more diversified centroid values we get, and the clearer our segmented image is. However, larger K also means higher time cost, especially when we have a large dataset. Thus, it is important for us to choose the K that well represent the original image and perform the task within reasonable length of time.

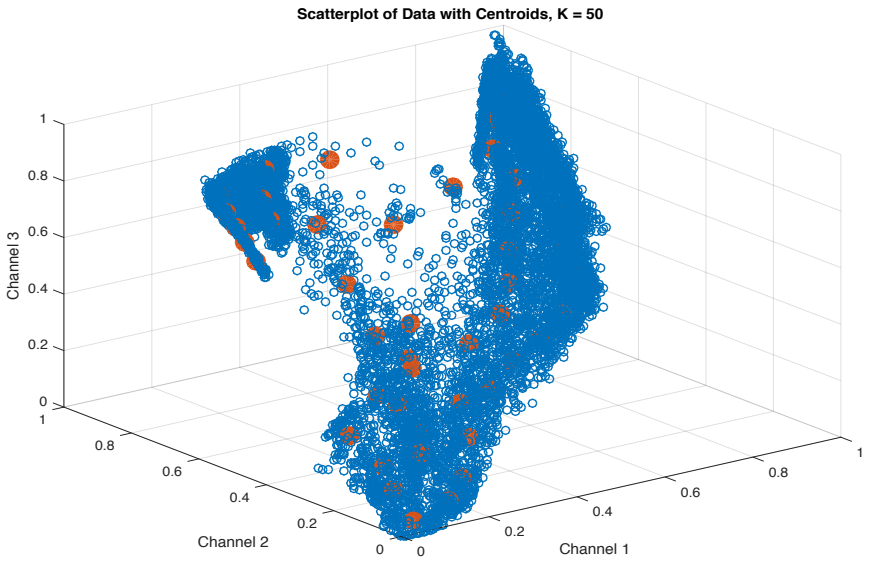
8.



The compression ratio between 10% ~ 15% corresponding to Mean RMSE 5 ~ 10, and K values 5 ~ 10 (according to #7) .

This selection makes sense because we see from the segmented images in #2 that starting from K = 4 or 5, the segmented images can well represent the features of the original image. When K = 10, the segmented image looks very like the original image.

9. With K = 50, the 3D scatterplot is



In quantization, the real valued function (signal) is like the pixels in the original images in clustering, while the discrete valued digital representation is like the centroids in clusters. Both processes try to represent the original values using a few possible discrete and achievable values. Like the quantization, in which the real valued functions are approximated by the discrete values, clustering approximates the original image using centroids; the data points around each centroid is like the grey function around the blue discrete segments. Data points around each centroid are then assigned with the value of the centroid, while the grey curve around each blue discrete line segment is assigned with the value of the blue line segment.

Matlab codes:

```
%%
%2.Test the algorithm for different values of K.
for k1 = 2:5
    [index,centers] = kmean(royce,k1);
    segIm = segmentation(royce,centers,index,k1);
    figure
    imshow(segIm)
end


for i = 1:5
    k2 = i*10;
    [index,centers] = kmean(royce,k2);
    segIm = segmentation(royce,centers,index,k2);
    figure
    imshow(segIm)
end

%%
%3. 3D Scatterplots of data
[index3,centers3] = kmean(royce,4);

figure
scatter3(royce(:,1),royce(:,2),royce(:,3))
hold on
scatter3(centers3(:,1),centers3(:,2),centers3(:,3),300,'filled')
title('Scatterplot of Data with Centroids, K = 4')
hold off
```

```
%%
%4.add zero mean gaussian noise
noisex = sqrt((var(royce(:,1)))/10)*randn(13824,1);
noisey = sqrt((var(royce(:,2)))/10)*randn(13824,1);
noisez = sqrt((var(royce(:,3)))/10)*randn(13824,1);
noise_mat = [noisex,noisey,noisez];
noise_image = royce+noise_mat;

[labels4,centr4] = kmean(noise_image,20);

  segIm4 = segmentation(noise_image,centr4,labels4,20);
  figure
  %imshow(segIm4)
  imshow(reshape(noise_image,96,144,3))

%%
%6.RMSE

for K6 = 1:100
   [idx6,c6]=kmeans(royce,K6);
   segIm6 = segmentation(royce,c6,idx6,K6);
   segVec6 = reshape(segIm6,96*144,3);

   container(:,:,K6) = segVec6;
end

%%
for i = 1:13824
    X = royce(i,:);
    X_h = container(i,:,2);
    RMSE(i) = norm(X-X_h,2)/sqrt(3);
end
RMSE = RMSE.';
imshow(mat2gray(reshape(RMSE,96,144)),[])

%%
%7.
royce_vec = reshape(royce,13824*3,1);
for K7=1:100
    seg_vec = reshape(container(:,:,K7),13824*3,1);
    RMSE_mean(K7) = norm(royce_vec - seg_vec,2)/sqrt(3);
end
figure
plot(RMSE_mean)
title('Mean RMSE')

%%
%8. mean RMSE vs. Compression ratio
for K8 = 1:100
    CR(K8) = (24*K8 + log2(K8)*13824)/(24*13824);
end
line1 = [[ones(36,1)-0.85],[0:35].'];
line2 = [[ones(36,1)-0.9],[0:35].'];
figure
hold on
scatter(CR,RMSE_mean)
plot(line1(:,1),line1(:,2),'black')
plot(line2(:,1),line2(:,2),'black')
title('Mean RMSE vs the Compression Ratio')
hold off

%%
```

```
%9.
[index9,centers9] = kmeans(royce,50);

figure
scatter3(royce(:,1),royce(:,2),royce(:,3))
hold on
scatter3(centers9(:,1),centers9(:,2),centers9(:,3),200,'filled')
title('Data with Centroids, K = 50')
hold off
```