Lab 2 (Weeks 5,6): Colour k-means clustering

***Each task in this lab exercise is worth 1.5% of your final unit grade (total 7.5%).** A task is only considered complete if you can demonstrate a working program and show understanding of the underlying concepts. Note that later tasks should reuse code from earlier tasks.*

In this laboratory exercise, you will create a program that clusters and re-colours each pixel in a colour image to *k* number of *mean* colours using the k-means clustering algorithm. In all of the tasks below, you should not use the kmeans function in matlab.

References:
K-means clustering: http://en.wikipedia.org/wiki/K-means_clustering
K-means++: https://en.wikipedia.org/wiki/K-means%2B%2B

Resources:
mandrill.jpg

Tasks 1 and 2: Perform k-means clustering on the input image

Load the mandrill.jpg colour image on disk into the MATLAB workspace. Show the loaded image in a window. Perform k-means clustering on the image with k=4 means. Use the following steps:

1) Randomly choose 4 pixels from the image and initialize each of the means to the colour values of these pixels.
2) Go through all the pixels in the image and calculate which of the 4 means it is closest to. For each pixel, store the index of the nearest mean at the same pixel location in the labels image.
3) Re-compute each mean by going through all the pixels in the colour image that were assigned to that mean.
4) Create an output RGB image where each pixel is colour coded with the newly computed mean to which it was assigned in step 2.
5) Wait for a keypress and redo steps 2-4.

Make sure you initialize the random number generator with a new seed each time(e.g. from the clock) and run the program several times to see if you always converge on the same answer. Try changing k from 4 to other numbers (from 2 to 10) and see how this affects the output and the repeatability of the program.

Task 3: Visualize the clustering results

From tasks 1 and 2, you should be able to observe the output RGB image converging towards 4 mean RGB values. In this task, write code to visualize the clustering process at every iteration. Draw the colour values from the mandrill.jpg image as well as the mean colour values as a 3D scatter plot. The colour of each data point should reflect the mean colour value that it is closest to. As it is computationally expensive to plot all the data points, only draw the value of every 100th pixel in every row.

Tasks 4 and 5: Initializing k-means clustering

Implement a variant of the k-means algorithm called the k-means++ algorithm. You can reuse code from tasks 1 and 2, with the only difference being that step (1) is now replaced with the a few additional steps:

1) Choose the first mean colour value randomly from the data points.
2) Compute the distance *d* between every data point and the most recently selected mean colour value. Compare *d* with distance values from previous iterations and record the minimum value as *D(x)*.

HINT: Keep a record of the distance values in an $m \times k$ array, where m represents the number of data points and k is the number of mean colour values. Initialize all elements in the array as large values (> 255^2).

3) Compute the probability of selecting this data point as $D(x)^2 / \sum_x D(x)^2$.
4) Randomly draw a new mean colour value based on the computed probability distribution.

   HINT: Compute an array that represents the cumulative distribution from the probability values using the cumsum function. Generate a random number *r* between 0 and 1. The new mean colour value corresponds to the first element value in the cumulative distribution array that is greater than or equal to the value *r*.
5) Repeat steps 2-4 until k number of means have been chosen.
6) Proceed with the standard k-means algorithm using the selected mean colour values.

Compare the k-means++ algorithm with the original k-means clustering algorithm. What do you observe? Which algorithm converges in a smaller number of iterations?