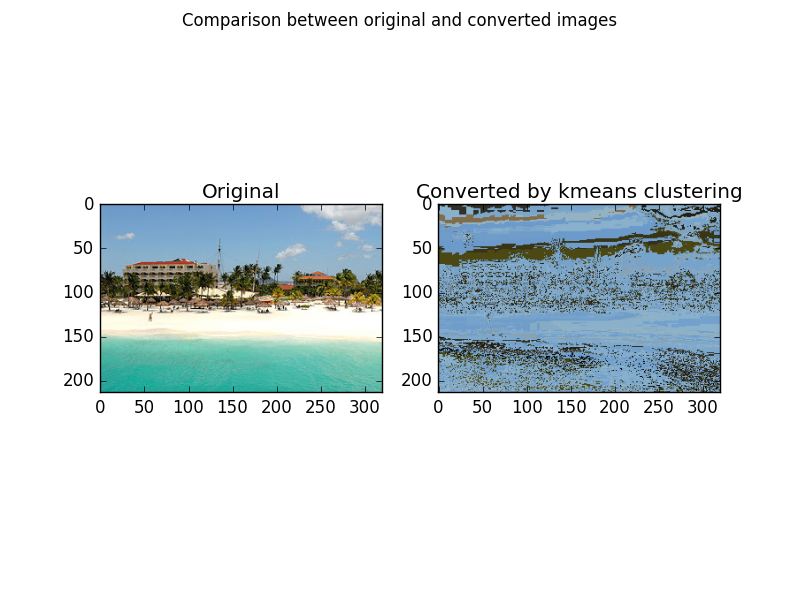
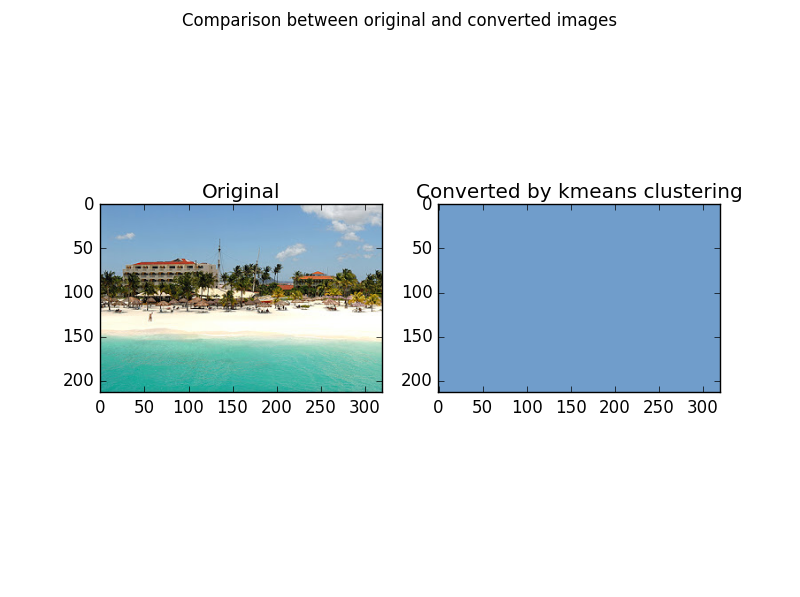
**Problem Overview**

For this homework, we used the Kmeans algorithm to compress images using K-clusters. We start off by initializing K random clusters in our dataset of pixels consisting of RGB values. We then assign each data point to the cluster with the minimum Euclidean distance to that point. Thus, we will have created the assignments vector containing the class of the index of every data point. Then, we relocate each cluster to be at the average of all the RGB values of every data point assigned to it. We would continue to do this until the centroids converge, which will happen for enough iterations since there are a finite amount of points.

**Results & Observations**

Unfortunately, while I believe that I implemented the Kmeans method perfectly, my code still continued bugs in that it resulted in a picture mostly covered in blue, when I ran it, as shown below:

However, I was still able to see variations in the graphics for varying k-values and number of iterations. As the number clusters (k-value) increases, we notice a sharper image because a wider spectrum of colors are being used to define the picture. Moreover, as we increase the number of iterations, the color-clusters used to define the pictures start to form more well-defined shapes and figures. We can describe the effects of num\_iterations and num\_clusters in the following matrix:

|  |  |  |
| --- | --- | --- |
|  | Low num\_iterations | High num\_iterations |
| Low num\_clusters | Poor Image, low quality, cannot detect image | Washed out in color, but able to tell apart shapes and figures |
| High num\_clusters | Good color scheme, but shapes are very awkwardly-shaped | While not great quality, satisfactory enough to get a wholesome view of the picture. |

Also, while it was difficult to test with my bugged program, pictures with a wide variety of colors will need a higher number of clusters in order to partition the colors accurately. On the other hand, pictures such as the one used above only need about 5 clusters for blue, white, green, red, and light blue. While there exists a small bug in my code, I was able to almost completely and successfully implement the K-means algorithm to compress a picture and I am looking forward to potentially using this learning technique in my project.