HW09 Niyati Shah – Kmeans Implementation

Part 3: Using rgb2ind, in RGB with different values of K.
Solution:

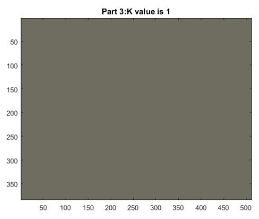
I used the Gaussian filter of size 5 and sigma 10 to remove unwanted noise from the image. It was quite useful to remove the salt and pepper kind of noise from the balls image and also made a smooth background in Macbeth and coins image.

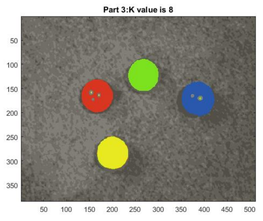
First I tried using each different value of k from 1 to 256 it took quite some time to just run for a single image but I noticed that at one point the change in the image was too small to detect for the human eye. So I decided to use only some values of k which were a good representation of the all the 256 values.

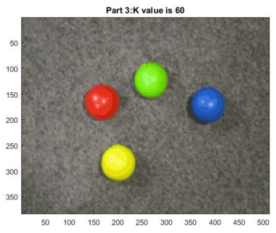
When K was 1 the whole image had just 1 color and was blank. Once the the value of K increased a few colors popped up which means that a few details were visible like in the balls image and the boxes in the Macbeth image. At lower values of k there are not enough colors in the image to show all the details in it. Like in the Macbeth image many boxes have similar colors instead of the original colors (pseudo colors) which are made up to fill that area.

Balls Image:

After k was 32 to k was 100 there was hardly any noticeable changes in the image. Only thing that could be observed is that the surface of the ball got a little bit smoother, at k = 128 it was quite smooth. After k was 8 all the balls had their original color but the image itself was too flat. At k above 70 you could see near smooth surface of the ball with its specular reflection and shadows.

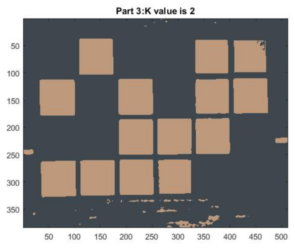


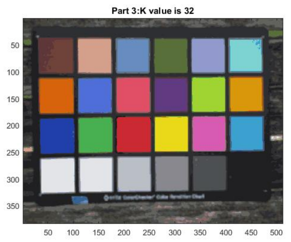




Macbeth:

After k was 32 all the colors of Macbeth chart were almost the same as the the original image. After which when the values were increased the image only got sharper with a few random noise like the shadow in the brown box appeared.



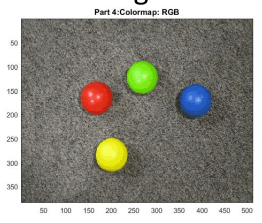


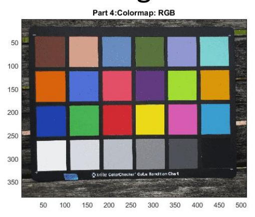
Part 4: Using a set value of K, with different color spaces, using rgb2ind.
Solution:

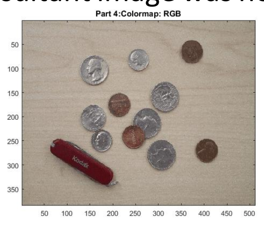
Different color maps gave different kind of outputs after quantization. As for each color map, the different channels and pixel value signify some different kind of information about the image, so when we quantize each of them we get different kind of images. After quantizing, the image needs to make up the values of the details that are lost, which means it gives out pseudo colors/ details on the image.

RGB:

As the pixels in this colormap represent the colors red green and blue, the original colors of the image was not lost though the resultant image was not as sharp and smooth as the original.

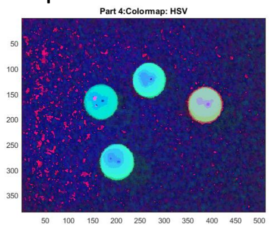


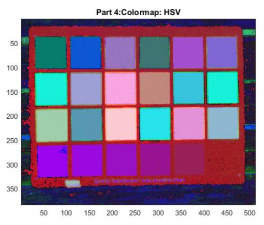


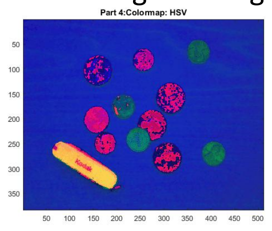


HSV:

As the pixels of this color map represent hue, saturation and value the quantized image had lost all the original color and was in the shades of blue/cyan magenta and yellow. This color map did not retain any color information from the original image.

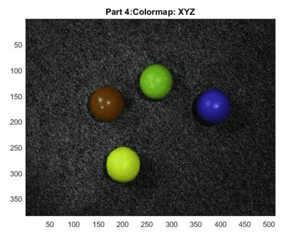


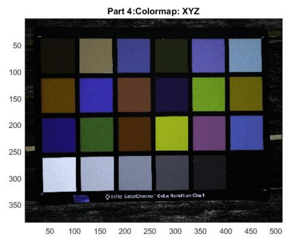


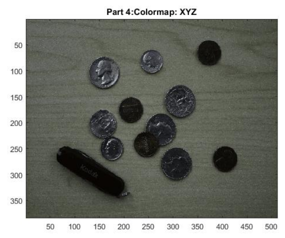


XYZ:

The pixels in the xyz color map darken when quantized using the XYZ colormap. The original colors from the image are not lost but the contrast in the image has decreased which makes the image dark.

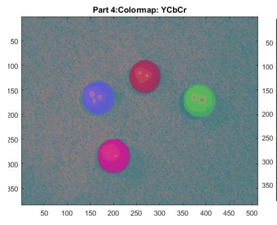


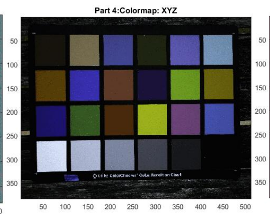


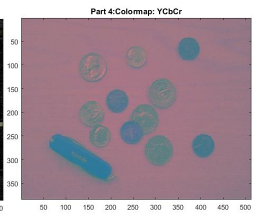


YCbCr:

The pixels in the YCbCr color map represent the luminance and chrominance value of blue and red. The quantized image of the balls has different colors then the original image and it feels like there is a greenish tinge in the image. The Macbeth image has all its colors but the image has darkened while the coins image has lost color with a pink tinge added to the whole image.

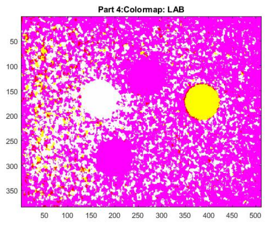


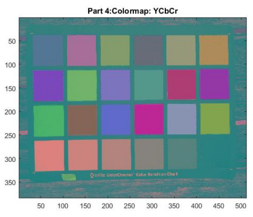


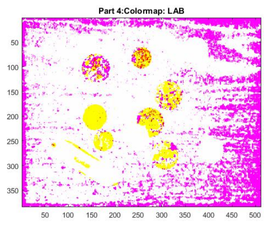


LAB:

The quantized image of the lab colorspace has lost all of its colors with the blue ball becoming yellow and 2 balls completely disappearing. The background has turned pink colored even in the coins image. The Macbeth image has a green background instead of the blackish color and the colored boxes have lost their true color.







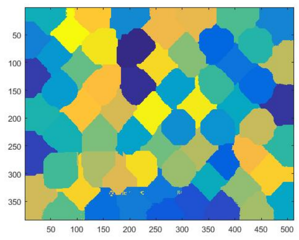
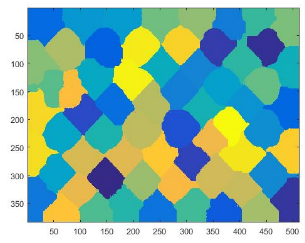
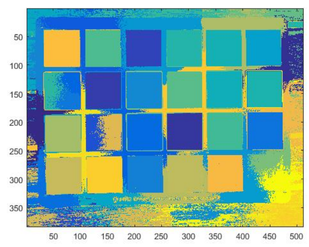
Part 5: Explicitly using kmeans( ) with different attributes.

Part 5a: The cityblock distance versus the Euclidean distance.
Solution:

Distance:

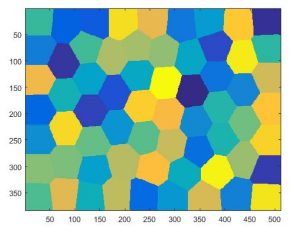
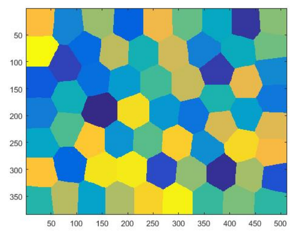
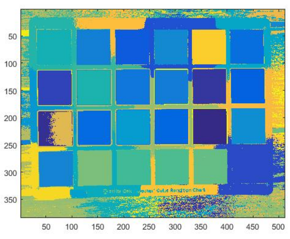
Cityblock: By using the Cityblock distance we can see some kind of box formation in the image but we can see a lot of extra edges in the image as compared to the Euclidean(seen in LAB image clearly) .

RGB HSV LAB

Euclidean: The Euclidean distance as such gave an image with almost true edges and very few edges which were not part of the actual image (seen in LAB image clearly) .

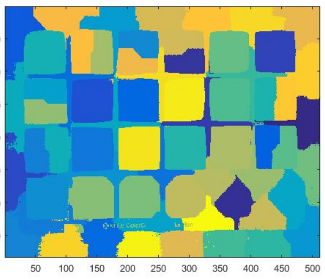
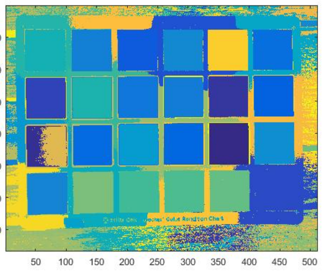
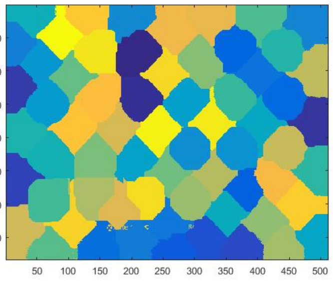
RGB HSV LAB

Parameters:

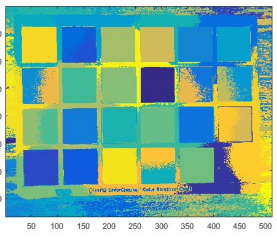
Colorspaces: A good clustered image was formed using LAB color space at weight value of 1/9 but at the same time images in RGB and HSV only contained blobs of colors. When the value was decreased to about 1/40 a good shape of boxes appeared for the RGB and HSV color space.

HSV weight 1/40 LAB weight 1/9 RGB weight 1/9

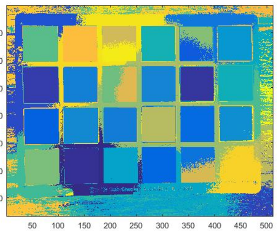




Maximum Iterations: When the number of iterations were decreased to 5 we could see an image that the clusters formed were not very intact. Default iterations of 100 also gave a scattered image with a few breaking/lines on the image.

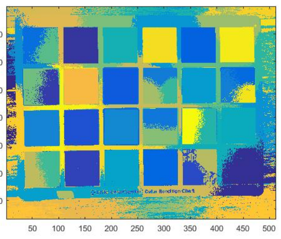
Iterations 5



Default 100

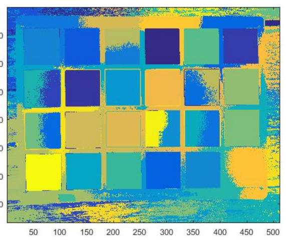


Iterations 500

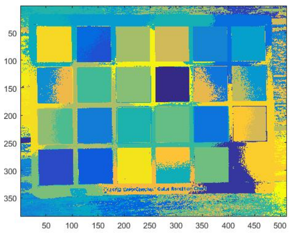


Replicate: When replicate was set to 1 we can see a lot of clusters scattered around the image while with 3 they are more gathered and at 10 the box edges and colors in them are intact and more or less single colored.

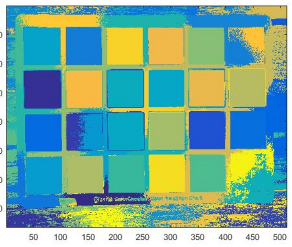
Replicate 1



Replicate 3

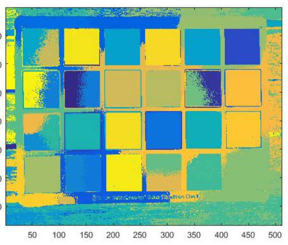


Replicate 10

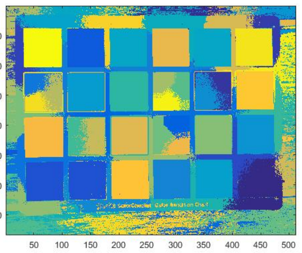


Initial Cluster/Start: Using the cluster option, the algorithm performs a clustering on random 10% subsample. While the uniform option selects k points uniformly from the range. We can see that from the cluster option the edges of the image are more or less intact than by using the default or uniform option.

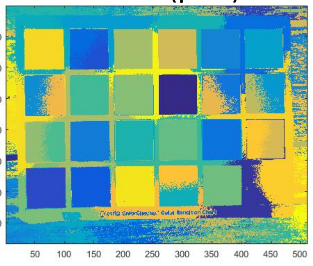
Uniform



Cluster

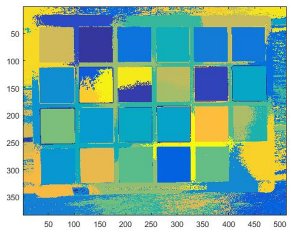


Default (plus)

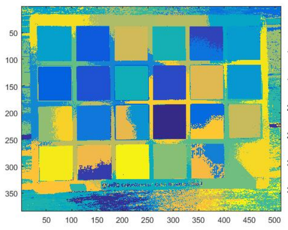


EmptyAction: There are options to what to do if clusters become empty, By using the drop option, we remove them completely and by using the error option we treat them as an error. When using the drop or error option instead of singleton(default) option we can see an improved image with boxes formed sharply. The error image also had a border on few boxes and the background and the paper border were also seen separately.

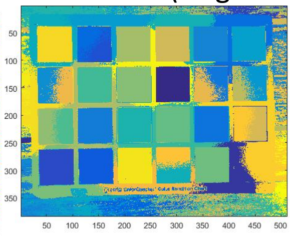
Drop



Error



Default(singleton)



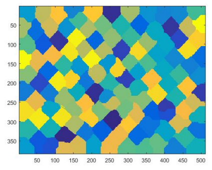
Part 5b: Kmeans learning

Solution:

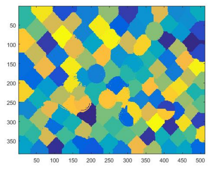
To understand the different distance parameters, I have kept all other parameters constant and just changed the weights and distance parameters to see the effect of each on an image. Here we can see for all the distance parameters at lower weight value we could hardle see the coins in the clusters formed. As the value increases we can see that the coins start to emerge.

Cityblock:

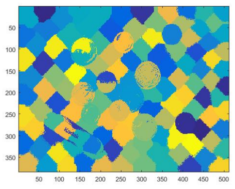
Weight 1/9



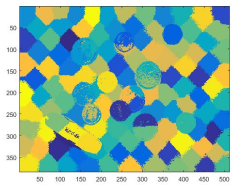
Weight 1/25



Weight 1/65

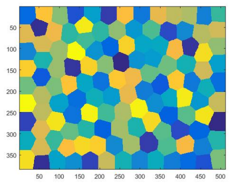


Weight 1/100

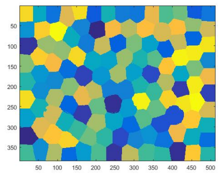


Euclidean:

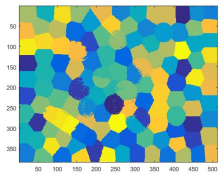
Weight 1/9



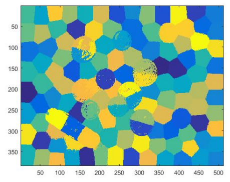
Weight 1/25



Weight 1/65

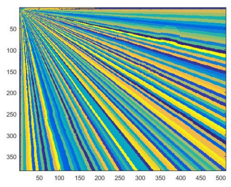


Weight 1/100

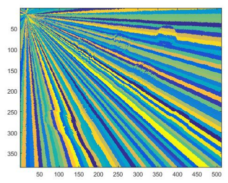


Correlation:

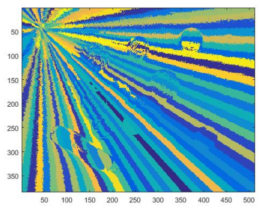
Weight 1/9



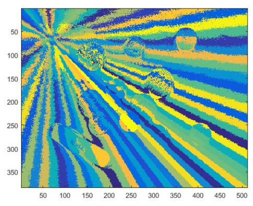
Weight 1/25



Weight 1/65

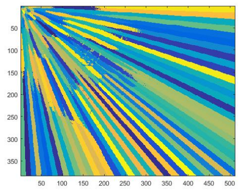


Weight 1/100

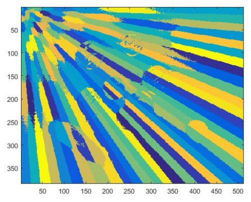


Cosine:

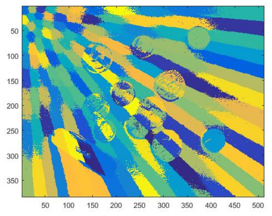
Weight 1/9



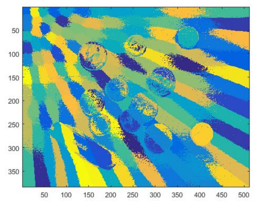
Weight 1/25



Weight 1/65



Weight 1/100



Hamming can only be used in a binary image.

Part 6: Cartoonizing your own portrait.
Solution: For cartoonizing the own image, I have used the kmeans algorithm to create clusters in the image.
To even make it more interesting I have used different colormaps and added an edge image to the image using the edge detection function inbuilt in matlab.