

## HOMEWORK – 9

## 3. HW09\_part3.m

Using `rgb2ind`, in RGB with different values of K.

Consider the two images `BALLS_*` and `MACBETH_*`.

a. Use an appropriate noise cleaning method to remove unwanted perturbations.

Describe your noise cleaning method in your write-up.

Ans:

For `BALLS FOUR 5244 shrunk.jpg`

I have first applied the block average filter to lower down the noise due to the carpet in background of image. The output of block average filter is cleaner. Then I have applied Gaussian filter to the output of block average to smoothen the image. This removed a lot of noise.

Extra work:

**Even though for this part of homework we are asked to work in RGB color space, I just tried to play around with YCBCR.** But in this case, we have reflection of light on the top surface of balls, so we must remove that. To remove the light reflection, I have converted the RGB image to YCBCR. Then I have reduced the value of Y channel by diving the Y channel by 10. We can then merge all 3 channel back again and this merged image has lower value of Y channel then the original image. So, when we apply the `rgb2ind(k-means)` on this image, we will have cleaner output.

Code below:

```
% Applied local averaging to clean the image
dsk_filter = fspecial('disk', 7);
i_mfiltered_avg = imfilter(i_m_ycc, dsk_filter);

% Applied gaussian filter to further remove the noise
gauss_filter = fspecial('gauss', 4, 0.3);
i_mfiltered = imfilter(i_mfiltered_avg, gauss_filter);
```

For `MACBETH_HW09 shrunk.jpg`

In this case, we do not have anything like noise due to reflection as above. So, we can directly apply local averaging followed by Gaussian.

Code below:

```
% Applied local averaging to clean the image
dsk_filter = fspecial('disk', 2);
i_mfiltered_avg = imfilter(i_m_dsk_filter);

% Applied gaussian filter to further remove the noise
gauss_filter = fspecial('gauss', 4, 0.5);
i_mfiltered = imfilter(i_mfiltered_avg, gauss_filter);
```

b. Try various values of K from 1 to 256, with `rgb2ind()`.

What value of K did you like best for the `BALLS_*.jpg` image.

What value of K did you like best for the `MACBETH_*.jpg` image?

What differences do you notice between these two images?

Ans:

I tried various values of K. For `BALLS FOUR 5244 shrunk.jpg`, lower values of K work fine because it has less number of colors. If we use higher values of K in ball's image, then we will end up getting more clusters and hence the output image will show more color shades on balls and that will not be considered as good segmentation. Also, if we take  $k=3$ , we will lose objects in the image. For me  $k=5$  worked the best for `BALLS FOUR 5244 shrunk.jpg`.

Extra work (with YCBCR) :-

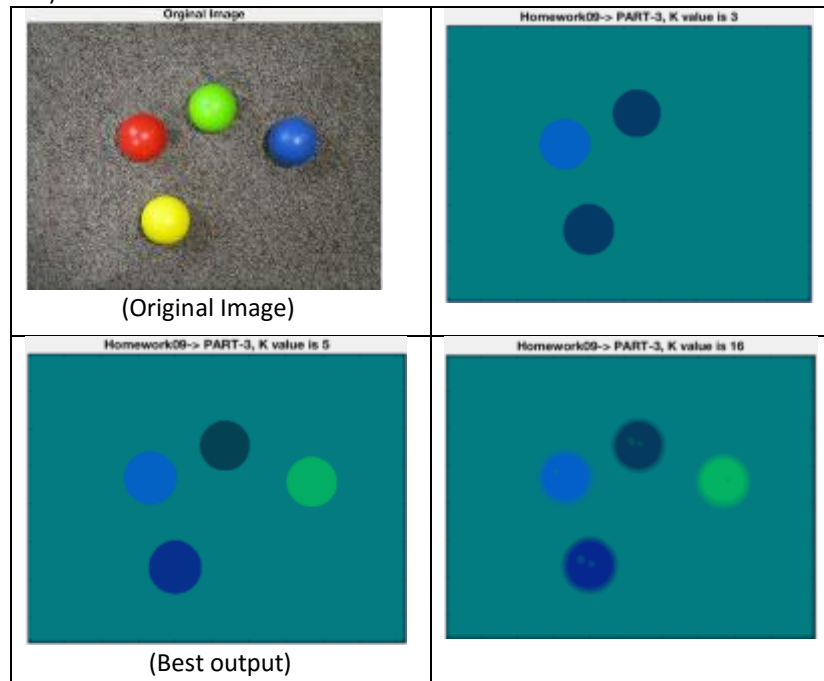
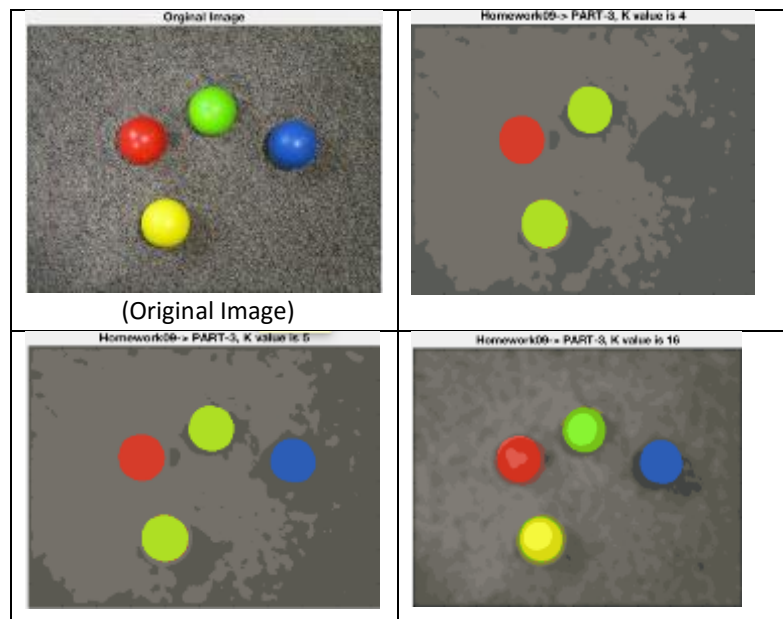


Figure a : Showing output images with various values of  $k$

With RGB:-



On the other hand, for MACBETH HW09 shrunk.jpg, we have many number of colors in the image. So, if we keep lower values of  $K$  then we will have less number of clusters and different color shaded will fall under same cluster. Hence, we will not be able to distinguish some blocks which came under same cluster, even though they have different color shaded. So, we must keep the  $K$  value on little higher end. For me  $K=40$  and above worked fine.

The main difference between two images is that number of colors in each image. From this, I have learnt that if we need to segment an image with less colors then we should take low value of K and vice versa.

Images on next page...

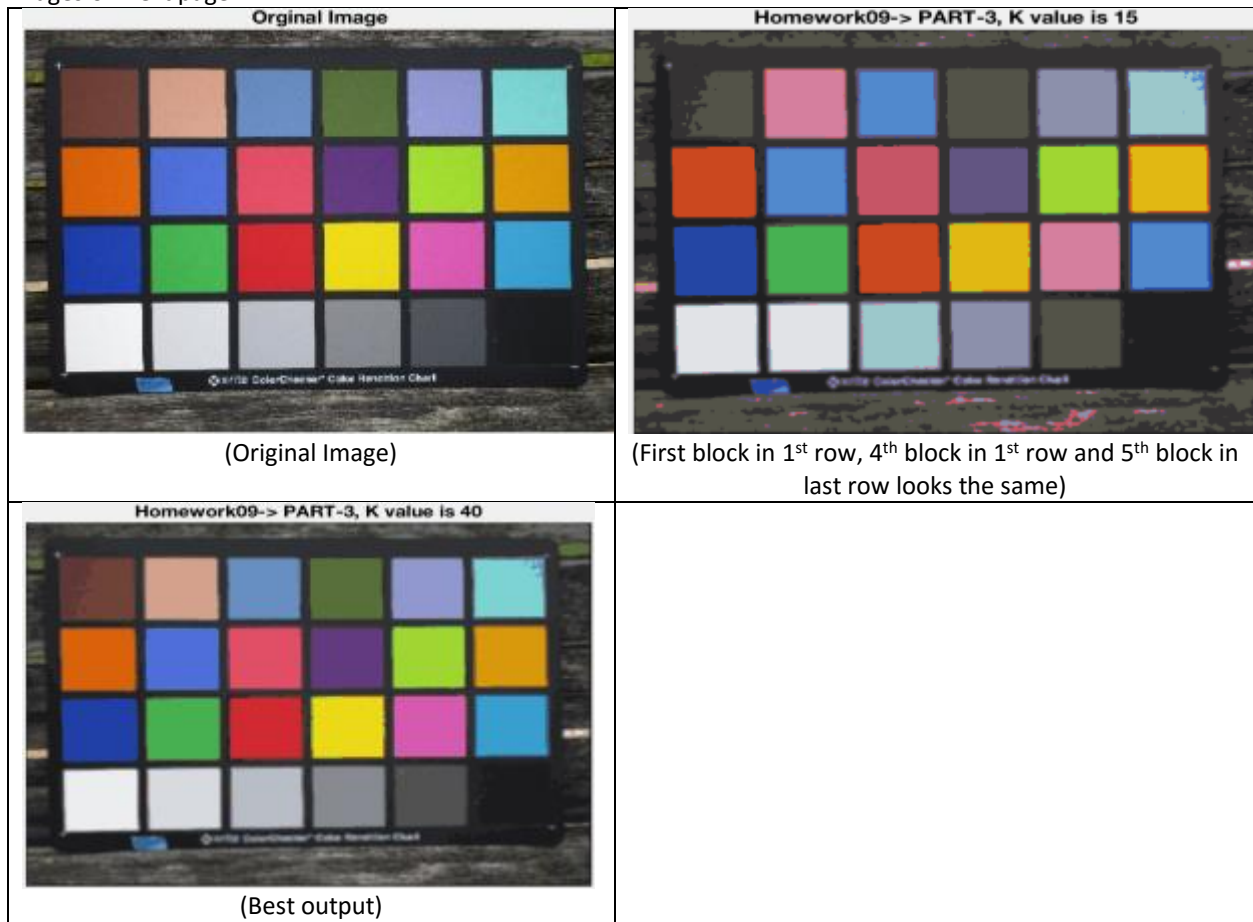


Figure b : Showing output images with various values of k

c. What general differences do you find between these two images? Are there any changes when segmenting balls versus squares?

Ans:

The main difference between two images is that number of colors in each image. From this, I have learnt that if we need to segment an image with less colors then we should take low value of K and vice versa.

Also, I have learnt that value of K totally depends on the image that we must segment. Even very large values of K can crowd the number of clusters and will give us bad output.

#### 4. HW09\_part4.m

Using a set value of K set to 64, `rgb2ind( )`, but do the pixel clustering on the images: `BALLS*`, `MACBETH*`, and `POCKET_CHANGE*`, but before doing the color quantization,

## CSCI-631: Foundations of Computer Vision

Submitted by: - Yash Jain (yj8359)

try converting to the following color spaces. As described in class, some color spaces may be problematic for some function calls.

- a. rgb,
- b. hsv,
- c. xyz,
- d. ycbcr, and
- e. lab

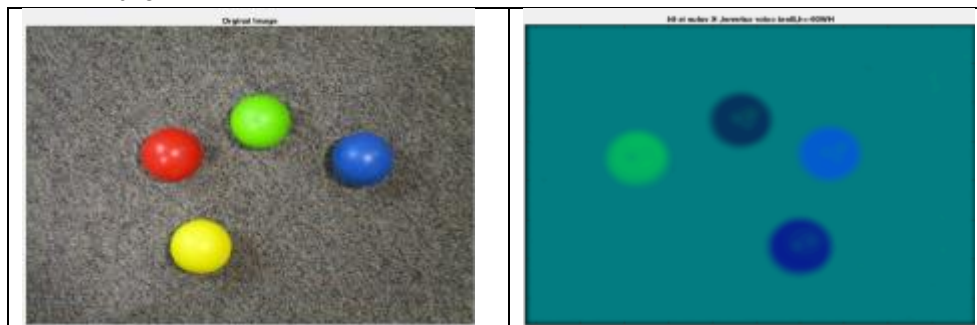
Describe any effects you notice between the colorspace.

In this part, we need play around with various color spaces. In this part homework, we are constrained to use the k value as 64. We cannot change it. But to get better results we should play around with various values of K.

I start with removing the noise from the image using local averaging and gaussian filters. In output, I displayed the images that came out after applying kmeans (rgb2ind) for all color spaces. Then I picked the best color channel by considering the best outcome.

Below is the best output for each case:

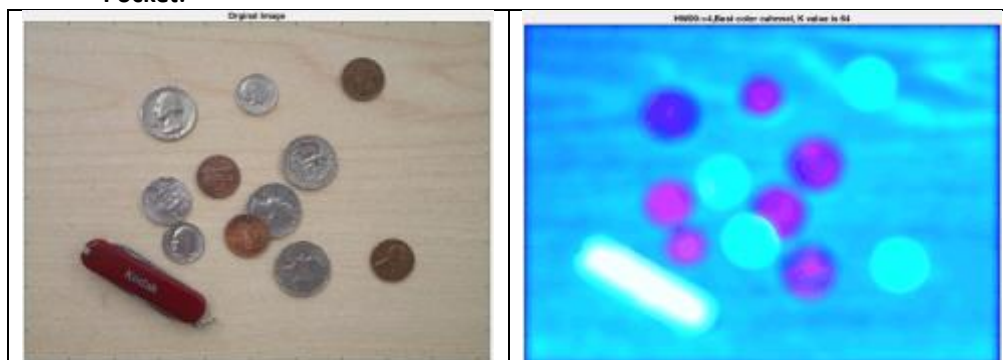
**Balls:**



**Macbeth:**



**Pocket:**



Also, I observed that colorspace is not fixed, rather its different for different images. So must look into the image we want o color segment and should find out what colorspace works the best.

I also played by increasing or decreasing the values of each color channel to get the best output. Below is the code which show the best parameters chosen.

```
% Looking at all the colorspace I found that ycbcr is good for balls,
% RGB is good for MACBETH and XYZ for pocket
% Setting values of noise reduction filters for particular image then applied them down below
if (~isempty(strfind(filename, 'BALLS')))
    dsk_size = 7;
    gauss_size = 4;
    gauss_sd = 0.25;

    i_m_ycbcr = rgb2ycbcr(i_m);

    i_m_y = i_m_ycbcr(:, :, 1)/30;
    i_m_cb = i_m_ycbcr(:, :, 2);
    i_m_cr = i_m_ycbcr(:, :, 3);
    i_m_to_process = cat(3, i_m_y, i_m_cb, i_m_cr);

    %% Noise removal
    % Applying disk filter to the given image (local average)
    dsk_filter = fspecial('disk', dsk_size);
    i_m_filtered_avg = imfilter(i_m_to_process, dsk_filter);

    % Applying gaussian filter to the image to get more smoothened output
    gauss_filter = fspecial('gauss', gauss_size, gauss_sd);
    i_m_filtered = imfilter(i_m_filtered_avg, gauss_filter);
end
if (~isempty(strfind(filename, 'MACBETH')))
    dsk_size = 7;
    gauss_size = 4;
    gauss_sd = 0.25;

    i_m_rgb = i_m;

    i_m_r = i_m_rgb(:, :, 1);
    i_m_g = i_m_rgb(:, :, 2);
    i_m_b = i_m_rgb(:, :, 3);
    i_m_to_process = cat(3, i_m_r, i_m_g, i_m_b);

    %% Noise removal
    % Applying disk filter to the given image (local average)
    dsk_filter = fspecial('disk', dsk_size);
    i_m_filtered_avg = imfilter(i_m_to_process, dsk_filter);

    % Applying gaussian filter to the image to get more smoothened output
    gauss_filter = fspecial('gauss', gauss_size, gauss_sd);
    i_m_filtered = imfilter(i_m_filtered_avg, gauss_filter);
end
if (~isempty(strfind(filename, 'POCKET')))
    dsk_size = 10;
    gauss_size = 8;
    gauss_sd = 3;

    i_m_hsv = rgb2hsv(i_m);

    i_m_x = i_m_hsv(:, :, 1)*1;
    i_m_y = i_m_hsv(:, :, 2)*5;
    i_m_z = i_m_hsv(:, :, 3)*5;
    i_m_to_process = cat(3, i_m_x, i_m_y, i_m_z);

    %% Noise removal
    % Applying disk filter to the given image (local average)
    dsk_filter = fspecial('disk', dsk_size);
    i_m_filtered_avg = imfilter(i_m_to_process, dsk_filter);

    % Applying gaussian filter to the image to get more smoothened output
```

```

gauss_filter = fspecial('gauss', gauss_size, gauss_sd);
i_mfiltered = imfilter(i_mfiltered_avg, gauss_filter);
end

%% Applying kmeans
% k value is 64
k_values = 64;

% passing in the filtered image with proper k value to rgb2nd function
[i_m_out, x] = rgb2nd(i_mfiltered, k_values, 'nodither');

% Displaying the output of rgb2nd
figure;
imagesc(i_m_out);
colorbar(x);
message = ['HW09- >4, Best color channel, K value is', num2str(k_values)];
title(message, 'FontSize', 8);

```

### 5a. HW09\_part5a.m:

Using k of 64, consider the image MACBETH\*

Look at the documentation for the options for kmeans().

Try using the cityblock distance versus the Euclidean distance.

Try different parameters and different attributes to get the best clustering – segmentation that does not cross the edges.

What colorspace, attributes, and parameter choices did you use?

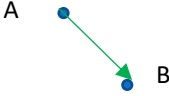

#### Working of kmeans():

First I read the documentation of kmeans to understand what parameters are required to apply kmeans. The first parameter it takes is the 2D matrix on which we are required to apply kmeans. Second parameter is the k value. Then we can name any parameter for example 'distance' and then value that is the type of distance to consider while computing kmeans clustering. In this part of homework, we have to use Euclidean and Cityblock distance. If we do not give any distance value, kmeans takes "Euclidean" by default. Another parameter is use is "MaxIter" which limits the number of iterations that kmeans should work on to perform clustering.

The output of the kmeans is cluster ids and the centroids. We use cluster ids to form the image using reshape function in MATLAB. Also, we can use the centroids to form the colormap that we can use while showing the image. While preparing colormap from centroids we need to choose all the rows but the columns which correspond to colors in attribute that have passed in kmeans.

#### Cityblock vs Euclidean:

I read the difference between cityblock and euclidean. To better understand I can show it as:

	
Euclidean Distance	Cityblock Distance

For MACBETH:

For MACBETH image, first I have converted the image to **LAB colorspace**. Also, using the meshgrid function that we have used in earlier homeworks, I found the x and y coordinates. As asked in question, we merged the LAB color channels and x and y to form the attribute that we will feed to kmeans algorithm. Before preparing attributes, I played around with various values of colorspace, below is the best set of parameters I have found.

```

% Adjusting attributes
xs1 = xs / 10;

```

```

ys1    = ys / 10;
first1 = first;
second1 = second;
third1 = third %

% Creating single attribute by combining l, a, b, x and y
attributes = [ double(first1(:)), double(second1(:)), double(third1(:)), xs1(:), ys1(:) ];

```

For MACBETH cityblock distance is better as for same value of k(64).



Second last gray block is more clear with cityblock.

#### Parameter Values Used:

<b>k</b>	<b>64</b>
<b>Disk filter size</b>	<b>3</b>
<b>Gaussian filter size, Standard Deviation</b>	<b>4 , 0.2</b>
<b>Color space</b>	<b>LAB</b>
<b>X_weight</b>	<b>0.1</b>
<b>Y_Weight</b>	<b>0.1</b>

#### 5b. HW09\_part5b.m:

Using k of 128, consider the image POCKET\_CHANGE\*

Explicitly use kmeans( ) with different attributes.

Try different parameters and different attributes to get the best clustering – segmentation that does not cross the edges.

What colorspace, attributes, and parameter choices did you use?

#### Working of kmeans():

First I read the documentation of kmeans to understand what parameters are required to apply kmeans. The first parameter it takes is the 2D matrix on which we are required to apply kmeans. Second parameter is the k value. Then we can name any parameter for example 'distance' and then value that is the type of distance to consider while computing kmeans clustering. In this part of homework, we have to use Euclidean and Cityblock distance. If we do not give any distance value, kmeans takes "Euclidean" by default. Another parameter is use is "MaxIter" which limits the number of iterations that kmeans should work on to perform clustering.



The output of the kmeans is cluster ids and the centroids. We use cluster ids to form the image using reshape function in MATLAB. Also, we can use the centroids to form the colormap that we can use while showing the image.

While preparing colormap from centroids we need to choose all the rows but the columns which correspond to colors in attribute that have passed in kmeans.



### Citiblock vs Euclidean:

I read the difference between cityblock and euclidean. To better understand I can show it as:

	
Euclidean Distance	Cityblock Distance

For POCKET:

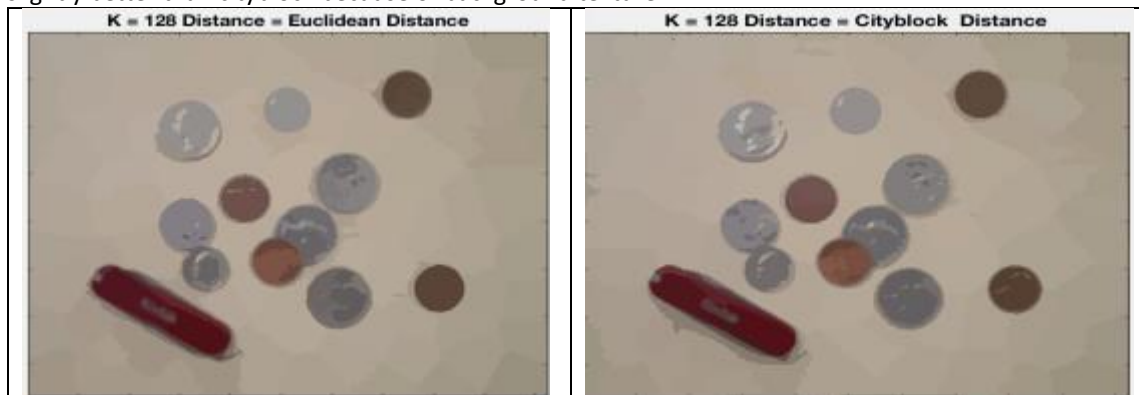
For POCKET image, first I have converted the image to **LAB colorspace**. Also, using the meshgrid function that we have used in earlier homeworks, I found the x and y coordinates. As asked in question, we merged the LAB color channels and x and y to form the attribute that we will feed to kmeans algorithm. Before preparing attributes, I played around with various values of colorspace, below is the best set of parameters I have found.

```

Adjusting attributes
xs1 = xs / 5;
ys1 = ys / 5;
first1 = first;
second1 = second;
third1 = third;

% Creating single attribute by combining l, a, b, x and y
attributes = [ double(first1(:)), double(second1(:)), double(third1(:)), xs1(:), ys1(:) ];
    
```

For POCKET image, I could barely see any difference between Euclidean and cityblock distance, but euclidean is slightly better than cityblock because of background texture.



### Parameter Values Used:

k	128
Disk filter size	2
Gaussian filter size, Standard Deviation	5 , 1
Color space	LAB
X_weight	0.2 (1/5)
Y_Weight	0.2 (1/5)



**6. HW09\_part6.m:**

Using what you learned, cartoonize your own portrait. You will use different values for this part. You may wish to use the edge strength at each pixel as an attribute. Your output cartoon may include adding the edges back in (in black). Pick an appropriate color space. Describe your solution. Did you do noise removal? Did you pre-scale your image using histogram equalization or a non-linear operation?

**Working of kmeans():**

First I read the documentation of kmeans to understand what parameters are required to apply kmeans. The first parameter it takes is the 2D matrix on which we are required to apply kmeans. Second parameter is the k value. Then we can name any parameter for example 'distance' and then value that is the type of distance to consider while computing kmeans clustering. In this part of homework, we have to use Euclidean and Cityblock distance. If we do not give any distance value, kmeans takes "Euclidean" by default. Another parameter is use is "MaxIter" which limits the number of iterations that kmeans should work on to perform clustering.

The output of the kmeans is cluster ids and the centroids. We use cluster ids to form the image using reshape function in MATLAB. Also, we can use the centroids to form the colormap that we can use while showing the image.

While preparing colormap from centroids we need to choose all the rows but the columns which correspond to colors in attribute that have passed in kmeans.

```
% Applying k-means
[duster_id, centroid ds] = kmeans( attributes, k, 'MaxIter', 1);
```

```
% Creating image from the dustered output
i_m_new = reshape( duster_id, d ms(1), d ms(2) );
```

```
% Creating colormap from the centroid ds output
lab_colormap = centroid ds( :, 1:3 );
rgb_colormap = lab2rgb( lab_colormap );
```

**For cartoonizing my image:-**

First I have performed the noise cleaning by applying local average and Gaussian filters.

Then I have tried all possible color spaces. But I found LAB to be the best. Also, using the meshgrid function that we have used in earlier homeworks, I found the x and y coordinates. As asked in question, we merged the LAB color channels and x and y to form the attribute that we will feed to kmeans algorithm. Before preparing attributes, I played around with various values of colorspace, below is the best set of parameters I have found.

```
% Gathering the x and y coordinates to pass
[ xs, ys ] = meshgrid( 1:d ms(2), 1:d ms(1) );
```

```
first = i m( :, :, 1 );
second = i m( :, :, 2 );
third = i m( :, :, 3 );
```

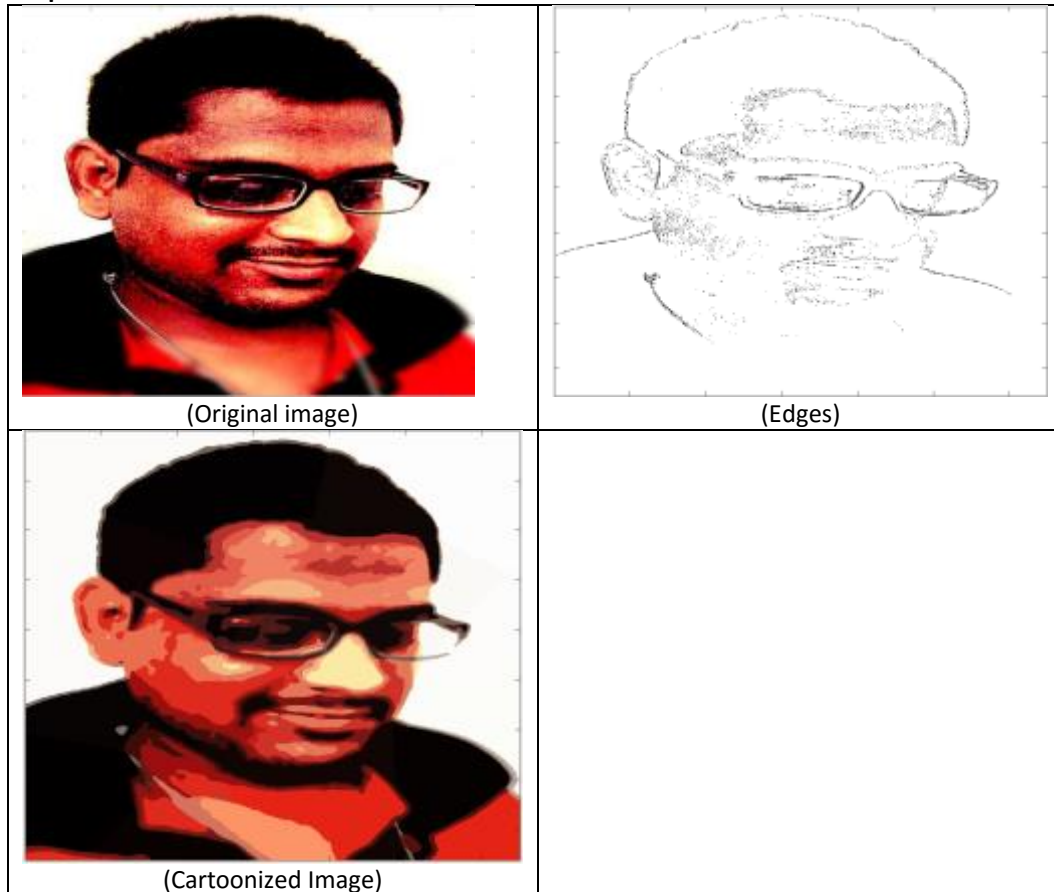
```
% Adjusting attributes
xs1 = xs / 10;
ys1 = ys / 10;
first1 = first;
second1 = second;
third1 = third;
```

```
% Creating single attribute by combining l, a, b, x and y
attributes = [ double( first1( : ) ), double( second1( : ) ) ...
             , double( third1( : ) ), xs1( : ), ys1( : )];
```

After cleaning and selecting the best color channel, I tried playing with various k values. I tried very large as well as very small values of k. As we increase the value of k, it will cartoonize very less as the number of clusters formed will be very large and we will not quantize our image much. After learning this, I tried decreasing the value of K and once I reach to k = 40, I started getting the cartoonizing effect as the number of clusters formed will be less so, closer points also mapped into same cluster and we will start losing details of image, which is all we want.

Before performing any of these operations, I calculated edges of image so that I can add it later. I have used the unsharp masking technique to add the black edges to the output image. I can use any edge detector as well e.g. sobel. But in my case I have used unsharp masking.

**Output:-**



### 7. Conclusions:

Overall this homework was great and it helped learning so many things like how to apply kmeans. How to handle various parameters for every particular image as parameter values changes image to image. It gave me good idea about how to cluster colored images.