# Image Segmentation with K-means clustering

### Step 1: Read Image

Read in face_2d.jpg.

```
img=imread('face_2d.jpg')
```

### Step 2: Convert Image from RGB color space to L*a*b* color space

```
form = makecform('srgb2lab'); #Create color transformation structure L*a*b*
lab_he = applycform(img,form); # Converts the color values in img to the color space specified in the color transformation structure "form"
```

### Step 3: Classify the color using K-means clustering

The L*a*b* space consists of a luminosity layer 'L*', chromaticity-layer 'a*' indicating where color falls along the red-green axis, and chromaticity-layer 'b*' indicating where the color falls along the blue-yellow axis. All of the color information is in the 'a*' and 'b*' layers.

We can measure the difference between two colors using the Euclidean distance metric. First, we need to reshape the image data for convenience.

```
ab = double(lab_he(:,:,2:3));  # Convert symbolic values to MATLAB double precision
nrows = size(ab,1);  # obtain the number of rows  of the image data
ncols = size(ab,2);  # obtain the number of columns  of the image data
ab = reshape(ab,nrows*ncols,2); #reshape the data to a nrows*ncols-by-2 matrix
```

It can be easily found that the image contains 3 distinct color clusters, so we set nColors=3 to denote the number of clusters.

Conduct k-means to cluster the objects into three clusters using the Euclidean distance metric, we also repeat the clustering 3 times with different initial values to avoid local minima.

```
nColors = 3;
 [cluster_idx, cluster_center] = kmeans(ab,nColors,'distance','sqEuclidean', ...
                                        'Replicates',3);
#cluster_idx is a vector containing cluster indices of each pixel.
#cluster_center is a numeric matrix, where row j is the centroid location of cluster j.
'distance','sqEuclidean' #Distance measure used for minimization is Squared Euclidean distance
  'Replicates'  ,3 #repeat the clustering 3 times to avoid local minima
```

### Step 4: Label evey pixel in the image using the results from k-means

```
pixel_labels = reshape(cluster_idx,nrows,ncols); #Reshape the indices vector into a nrows-by-ncols matrix

#Present the result image
figure(1),subplot(1,4,1),imshow(pixel_labels,[]), title('image labeled by cluster index');
```


We can find that the three clusters represent the face, the tie and the rest of the image respectively. In other words, face is separated well.

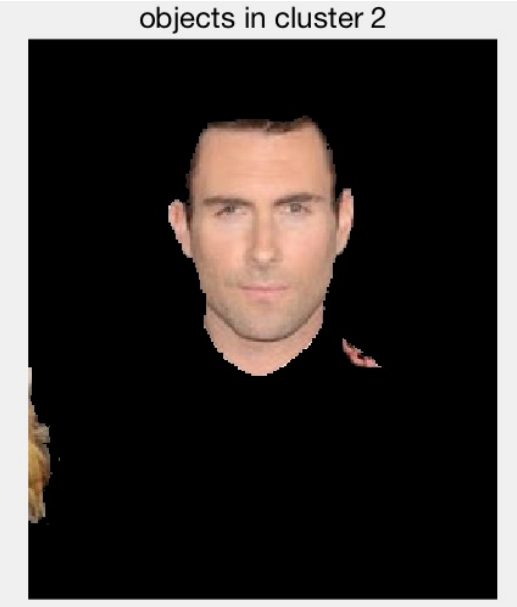
### Step 5: Create images that segment the image by color

```
segmented_images = cell(1,3); #Set 3 cells to store the image data
rgb_label = repmat(pixel_labels,[1 1 3]); #Repeat copies of  'pixel_labels'  3 times for 3 clusters
```

For each cluster index, if the pixel label does not equal to the index, change the color value to be zero, which means the color corresponding to the pixel is black. Otherwise, the color value is the same as the color value of the original image for that pixel.

```
for k = 1:nColors
    color = img;
    color(rgb_label ~= k) = 0;
    segmented_images{k} = color;
end
#present the three segmented images
figure(1),subplot(1,4,2),imshow(segmented_images{1}), title('objects in cluster 1');
figure(1),subplot(1,4,3),imshow(segmented_images{2}), title('objects in cluster 2');
figure(1),subplot(1,4,4),imshow(segmented_images{3}), title('objects in cluster 3');
```

From the below result, we can see the face is recognized well in the second cluster by k-means.



**Limitations:**
Some hair and fingers are also separated in the cluster 2.