Color-Based Segmentation Using K-Means Clustering

[View all machine learning examples](https://www.mathworks.com/solutions/machine-learning/examples.html)

This example shows how to segment colors in an automated fashion using the L\*a\*b\* color space and K-means clustering.

**Step 1: Read Image**

Read in hestain.png, which is an image of tissue stained with hemotoxylin and eosin (H&E). This staining method helps pathologists distinguish different tissue types.

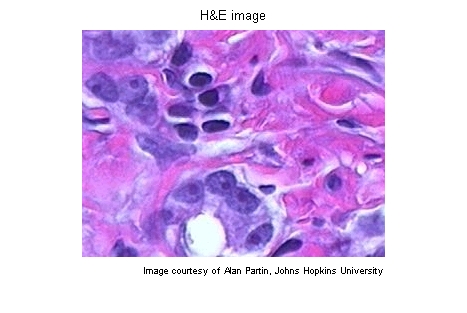
he = imread('hestain.png');

imshow(he), title('H&E image');

text(size(he,2),size(he,1)+15,...

'Image courtesy of Alan Partin, Johns Hopkins University', ...

'FontSize',7,'HorizontalAlignment','right');



**Step 2: Convert Image from RGB Color Space to L\*a\*b\* Color Space**

How many colors do you see in the image if you ignore variations in brightness? There are three colors: white, blue, and pink. Notice how easily you can visually distinguish these colors from one another. The L\*a\*b\* color space (also known as CIELAB or CIE L\*a\*b\*) enables you to quantify these visual differences.

The L\*a\*b\* color space is derived from the CIE XYZ tristimulus values. 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. You can measure the difference between two colors using the Euclidean distance metric.

Convert the image to L\*a\*b\* color space using makecform and applycform.

cform = makecform('srgb2lab');

lab\_he = applycform(he,cform);

**Step 3: Classify the Colors in 'a\*b\*' Space Using K-Means Clustering**

Clustering is a way to separate groups of objects. K-means clustering treats each object as having a location in space. It finds partitions such that objects within each cluster are as close to each other as possible, and as far from objects in other clusters as possible. K-means clustering requires that you specify the number of clusters to be partitioned and a distance metric to quantify how close two objects are to each other.

Since the color information exists in the 'a\*b\*' space, your objects are pixels with 'a\*' and 'b\*' values. Use kmeans to cluster the objects into three clusters using the Euclidean distance metric.

ab = double(lab\_he(:,:,2:3));

nrows = size(ab,1);

ncols = size(ab,2);

ab = reshape(ab,nrows\*ncols,2);

nColors = 3;

% repeat the clustering 3 times to avoid local minima

[cluster\_idx cluster\_center] = kmeans(ab,nColors,'distance','sqEuclidean', ...

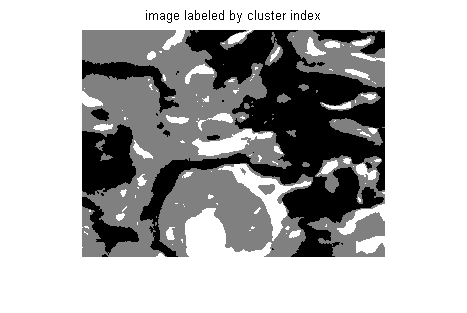
'Replicates',3);

**Step 4: Label Every Pixel in the Image Using the Results from KMEANS**

For every object in your input, kmeans returns an index corresponding to a cluster. The cluster\_center output from kmeans will be used later in the example. Label every pixel in the image with its cluster\_index.

pixel\_labels = reshape(cluster\_idx,nrows,ncols);

imshow(pixel\_labels,[]), title('image labeled by cluster index');



**Step 5: Create Images that Segment the H&E Image by Color.**

Using pixel\_labels, you can separate objects in hestain.png by color, which will result in three images.

segmented\_images = cell(1,3);

rgb\_label = repmat(pixel\_labels,[1 1 3]);

for k = 1:nColors

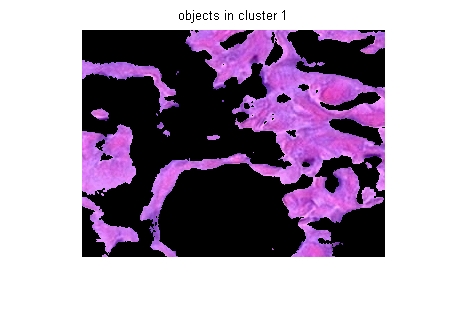
color = he;

color(rgb\_label ~= k) = 0;

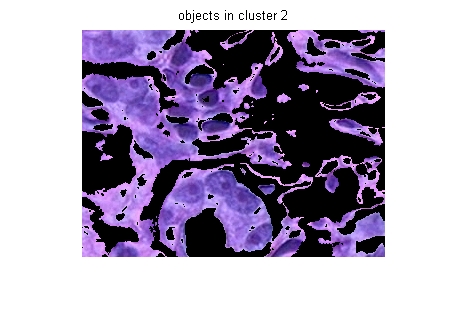
segmented\_images{k} = color;

end

imshow(segmented\_images{1}), title('objects in cluster 1');



imshow(segmented\_images{2}), title('objects in cluster 2');



imshow(segmented\_images{3}), title('objects in cluster 3');



**Step 6: Segment the Nuclei into a Separate Image**

Notice that there are dark and light blue objects in one of the clusters. You can separate dark blue from light blue using the 'L\*' layer in the L\*a\*b\* color space. The cell nuclei are dark blue.

Recall that the 'L\*' layer contains the brightness values of each color. Find the cluster that contains the blue objects. Extract the brightness values of the pixels in this cluster and threshold them using im2bw.

You must programmatically determine the index of the cluster containing the blue objects because kmeans will not return the same cluster\_idx value every time. You can do this using the cluster\_center value, which contains the mean 'a\*' and 'b\*' value for each cluster. The blue cluster has the smallest cluster\_center value (determined experimentally).

mean\_cluster\_value = mean(cluster\_center,2);

[tmp, idx] = sort(mean\_cluster\_value);

blue\_cluster\_num = idx(1);

L = lab\_he(:,:,1);

blue\_idx = find(pixel\_labels == blue\_cluster\_num);

L\_blue = L(blue\_idx);

is\_light\_blue = im2bw(L\_blue,graythresh(L\_blue));

Use the mask is\_light\_blue to label which pixels belong to the blue nuclei. Then display the blue nuclei in a separate image.

nuclei\_labels = repmat(uint8(0),[nrows ncols]);

nuclei\_labels(blue\_idx(is\_light\_blue==false)) = 1;

nuclei\_labels = repmat(nuclei\_labels,[1 1 3]);

blue\_nuclei = he;

blue\_nuclei(nuclei\_labels ~= 1) = 0;

imshow(blue\_nuclei), title('blue nuclei');

