University of Rochester. ECE 449 Machine Vision.

Tiecheng Su, Dept. of Electrical Engineering

HW3. Submission: 20 March 2016

%% Initialization

close all; clc; clear

%% Get rid of the specular highlights by threshold

I = double(imread('E:\2016Spring\MV\HW3\PeppersRGB.tif'));

I\_flat = I;

I\_flat(I>180)=180;

figure;

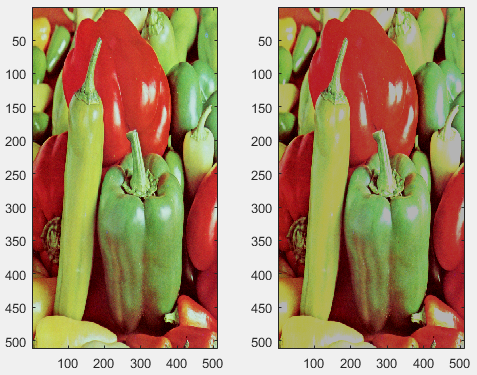
subplot(1,2,1);

imagesc(I/255);

subplot(1,2,2);

imagesc(I\_flat/255);

The pixel intensity of specular highlights should be a big value, so we get rid pf it by thresholding

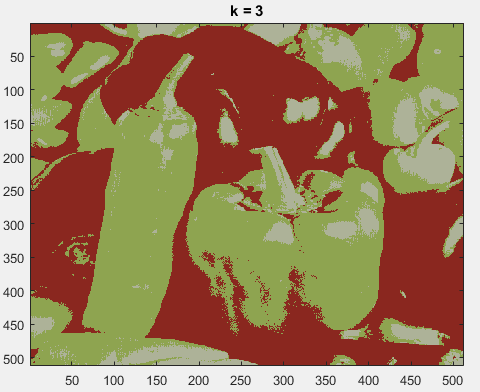
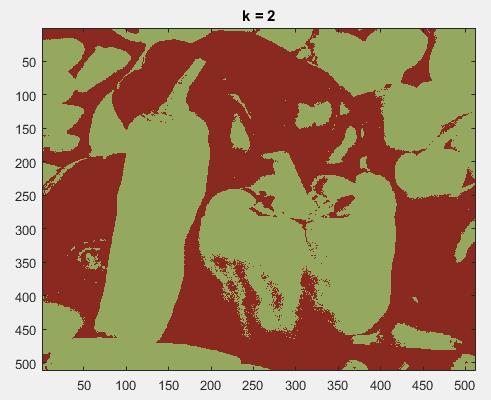


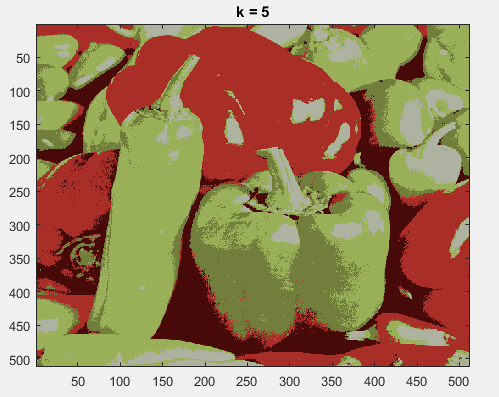
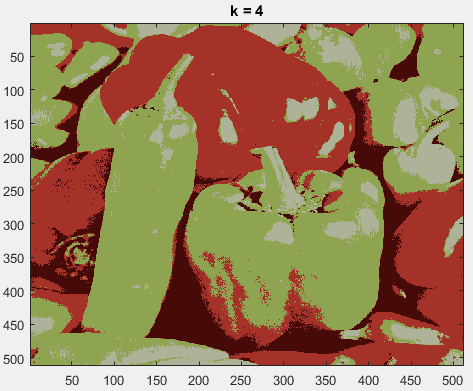
%% RGB feature vector

max\_iterations = 30;

k\_values = 2:5;

[I\_segmentation,I\_segmentation\_class] = k\_means(I\_flat,k\_values,max\_iterations);



 For k = 2:

Iteration number for converge: 9

For k = 3:

Iteration number for converge: 13

For k = 4:

Iteration number for converge: 15

According our observation, we can roughly segment red and green peppers when k=4 by using RGB feature vector. And it converges pretty fast.

%%

[I\_segmentation,I\_segmentation\_class] = k\_means(I\_flat,4,max\_iterations);

figure;

subplot(3,3,1)

imagesc(I\_segmentation/255);

subplot(3,3,2)

imagesc(I\_segmentation\_class(:,:,:,1)/255);

subplot(3,3,3)

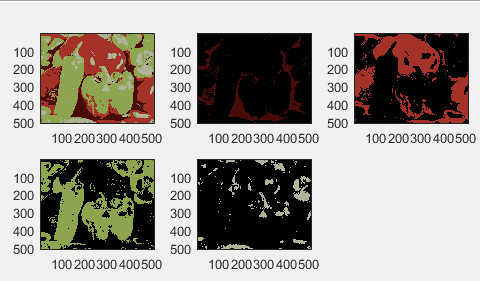
imagesc(I\_segmentation\_class(:,:,:,2)/255);

subplot(3,3,4)

imagesc(I\_segmentation\_class(:,:,:,3)/255);

subplot(3,3,5)

imagesc(I\_segmentation\_class(:,:,:,4)/255);



%% LST as the feature vector

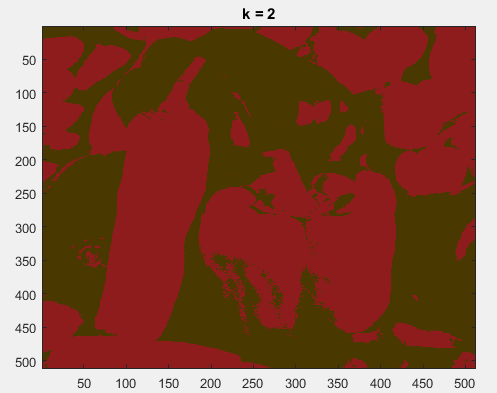
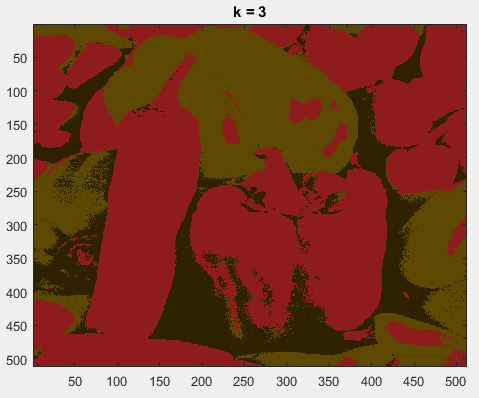
LST = zeros(size(I));

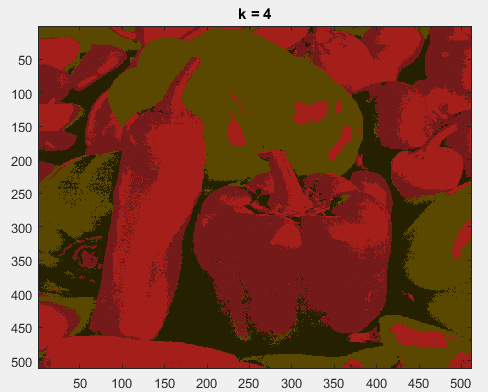
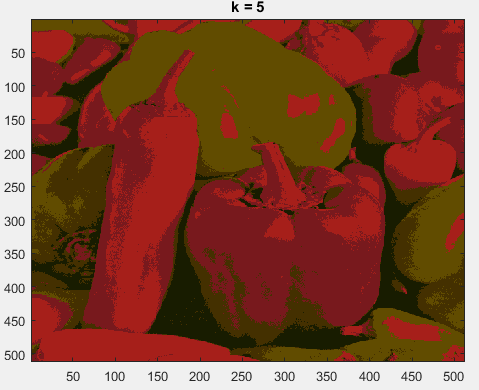
LST(:,:,1) = (I(:,:,1)+I(:,:,2)+I(:,:,3))./3;

LST(:,:,2) = (I(:,:,1)-I(:,:,3))./2;

LST(:,:,3) = (2\*I(:,:,2)-I(:,:,1)-I(:,:,3))./4;

[LST\_segmentation,LST\_segmentation\_class] = k\_means(LST,k\_values,max\_iterations);

For k = 2:

Iteration number for converge: 11

For k = 3:

Iteration number for converge: 24

Consider the converging speed and the result of segementation, it is the best when k=3 for using LST feature vector

%%

[LST\_segmentation,LST\_segmentation\_class] = k\_means(LST,3,max\_iterations);

figure;

subplot(2,2,1)

imagesc(LST\_segmentation/255);

subplot(2,2,2)

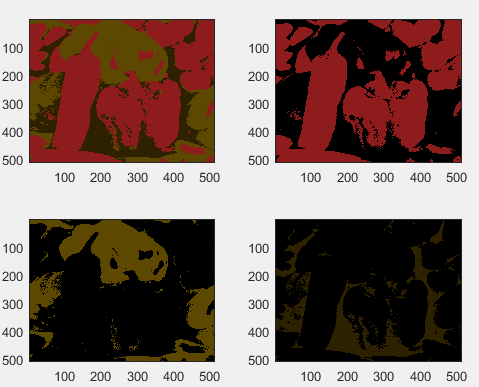
imagesc(LST\_segmentation\_class(:,:,:,1)/255);

subplot(2,2,3)

imagesc(LST\_segmentation\_class(:,:,:,2)/255);

subplot(2,2,4)

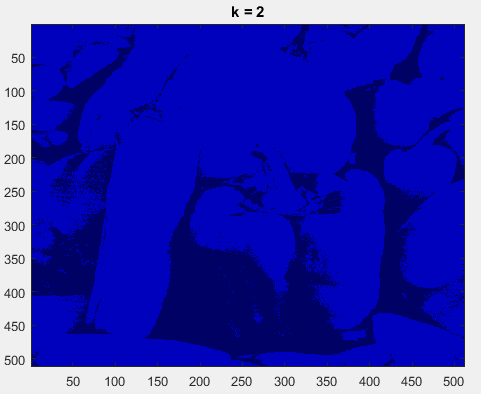
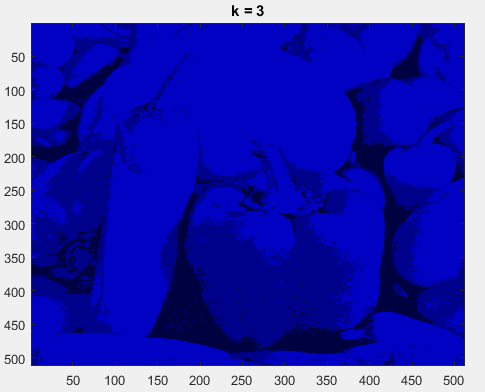
imagesc(LST\_segmentation\_class(:,:,:,3)/255);

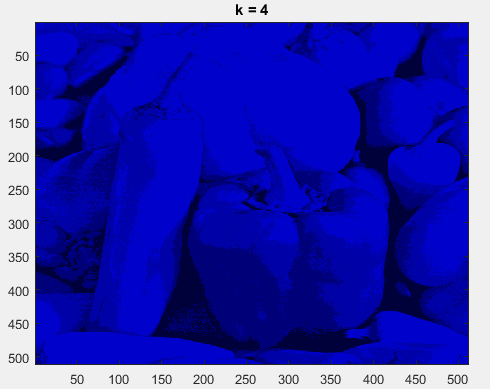
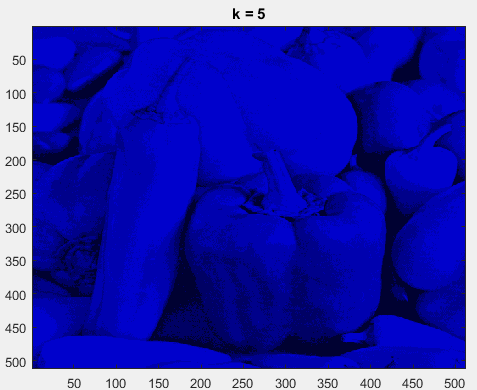


%% HSV as the feature vector

HSV = rgb2hsv(I);

[HSV\_segmentation,HSV\_segmentation\_class] = k\_means(HSV,k\_values,max\_iterations);

For k = 2:

Iteration number for converge: 10

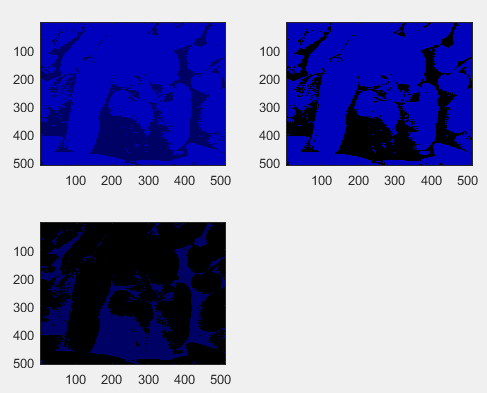
For k = 3:

Iteration number for converge: 14

For k = 4:

Iteration number for converge: 21

Seems like k=2 is enough for the segmentation



The chose of k should dependent on the iteration time and the accuracy. There is a trade-off between them. Obviously, we can get higher accuracy by using greater k. but converge speed will be slower. According to the result, the RGB feature vector is better, cause it can separate the green and red pepper with smallest iteration times.

function [I\_segmentation,I\_segmentation\_class]= k\_means(I,k\_values,max\_iterations)

[r,c,d] = size(I);

for k = k\_values

%Initialize k means

means = zeros(k,3);

for i = 1:k

means(i,:) = 256/(k+1)\*i;

end

%Maximum number of iterations

for iter = 1:max\_iterations

new\_means = zeros(size(means));

num\_assigned = zeros(k,1);

for i = 1:r

for j = 1:c

f1 = I(i,j,1);

f2 = I(i,j,2);

f3 = I(i,j,3);

diff = ones(k,1)\*[f1,f2,f3]-means;

distance = sum(diff.^2,2);

[val,index]=min(distance);

%assign fi to the cluster means

new\_means(index,1) = new\_means(index,1)+f1;

new\_means(index,2) = new\_means(index,2)+f2;

new\_means(index,3) = new\_means(index,3)+f3;

num\_assigned(index) = num\_assigned(index)+1;

end

end

for i = 1:k

if(num\_assigned(i)>0)

new\_means(i,:) = new\_means(i,:)./num\_assigned(i);

end

end

T = sum(sqrt(sum((new\_means - means).^2,2)));

%The sum of changes in the centroids<T(0.01)

if T <0.01

converge = sprintf('\t For k = %d: \n \t\t Iteration number for converge: %d \n',k,iter);

disp(converge);

break

end

means = new\_means;

end

% replace the individual pixel values with k-means

I\_segmentation = I;

I\_segmentation\_class = zeros(r,c,3,k);

for i = 1:r

for j = 1:c

f1 = I(i,j,1);

f2 = I(i,j,2);

f3 = I(i,j,3);

diff = ones(k,1)\*[f1,f2,f3]-means;

distance = sum(diff.^2,2);

[val,index]=min(distance);

I\_segmentation(i,j,:) = means(index,:);

I\_segmentation\_class(i,j,:,index)=means(index,:);

end

end

figure;

imagesc(I\_segmentation/256);title(['k = ',num2str(k)]);

end