Image processing for bioinformatics

Laboratory
Segmentation

1 Examples, Matlab functions

1.1 Automatic thresholding: OTSU's method

Code

```
%% Otsu's method
I_1 = imread('coins.png');
I_2 = imread('eighttif.png'); I_2 = I_2(:,:,1);
I_3 = imread('riceG.png');
% Get the threshold
otsuLevel_I_1 = graythresh(I_1);
otsuLevel_I_2 = graythresh(I_2);
otsuLevel_I_3 = graythresh(I_3);
% Compute binary image using the threshold
blackWhite_I_1 = imbinarize(I_1,otsuLevel_I_1);
blackWhite_I_2 = imbinarize(I_2,otsuLevel_I_2);
blackWhite_I_3 = imbinarize(I_3,otsuLevel_I_3);
% Histogram
[myHist_1,binLocations] = imhist(I_1);
[myHist_2,binLocations] = imhist(I_2);
[myHist_3,binLocations] = imhist(I_3);
```

Image

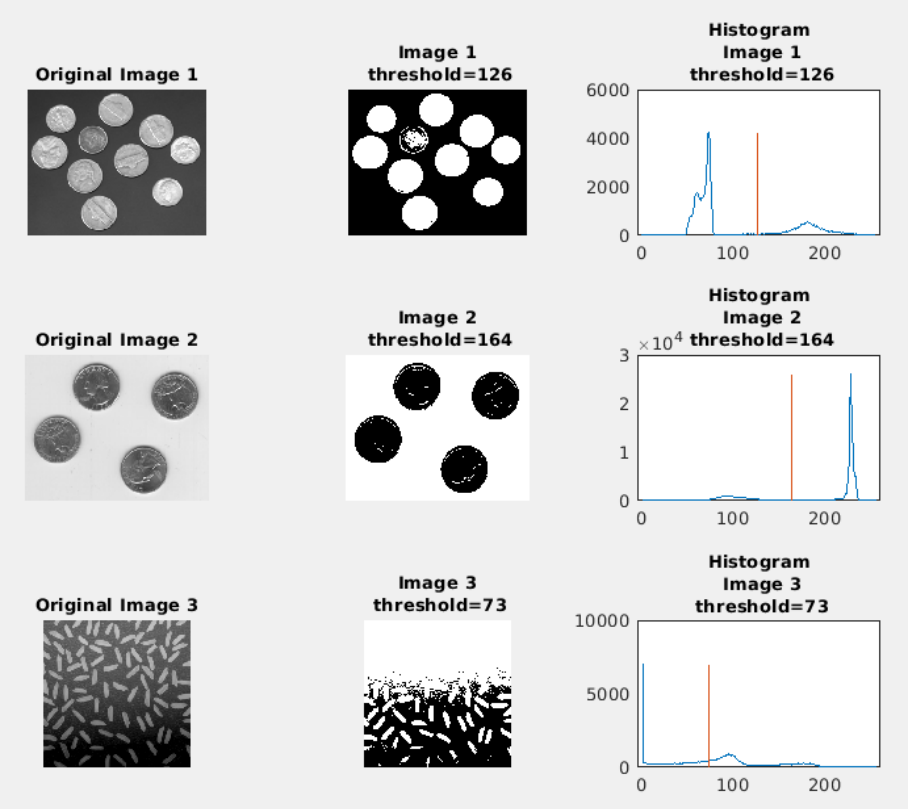


Table 1: OTSU's method

1.2 Variable thresholding: Image subdivision and OTSU's method

Code

```
%% Subdivision
I_1 = imread('riceG.png');

otsuLevel_I_1 = graythresh(I_1);
blackWhite_I_1 = imbinarize(I_1,otsuLevel_I_1);

[m,n] = size(I_1);

% Subdivision image 1x1 -> 2x2 subimages
I_2_cells = mat2cell(I_1,[m/2,m/2],[n/2,n/2]);
otsuLevel_I_2_cells = [];
blackWhite_I_2_cells = {};
% For all subimages
for i=1:2
        for j=1:2
                % Get the threshold
                otsuLevel_I_2_cells(i,j) = graythresh(I_2_cells{i,j});
                % Compute binary image using the threshold
                blackWhite_I_2_cells{i,j} = imbinarize(I_2_cells{i,j},otsuLevel_I_2_cells(i,j));
        end
end
% Union subimage 2x2 -> 1x1 Image
blackWhite_I_2_2 = cell2mat(blackWhite_I_2_cells);


% Subdivision image 1x1 -> 4x4 subimages
I_2_cells = mat2cell(I_1,[m/4,m/4,m/4,m/4],[n/4,n/4,n/4,n/4]);
otsuLevel_I_2_cells = [];
blackWhite_I_2_cells = {};
% For all subimages
for i=1:4
        for j=1:4
                % Get the threshold
                otsuLevel_I_2_cells(i,j) = graythresh(I_2_cells{i,j});
                % Compute binary image using the threshold
                blackWhite_I_2_cells{i,j} = imbinarize(I_2_cells{i,j},otsuLevel_I_2_cells(i,j));
        end
end
% Union subimage 4x4 -> 1x1 Image
blackWhite_I_2_4 = cell2mat(blackWhite_I_2_cells);
```

Table 2: Image subdivision and OTSU's method (code)

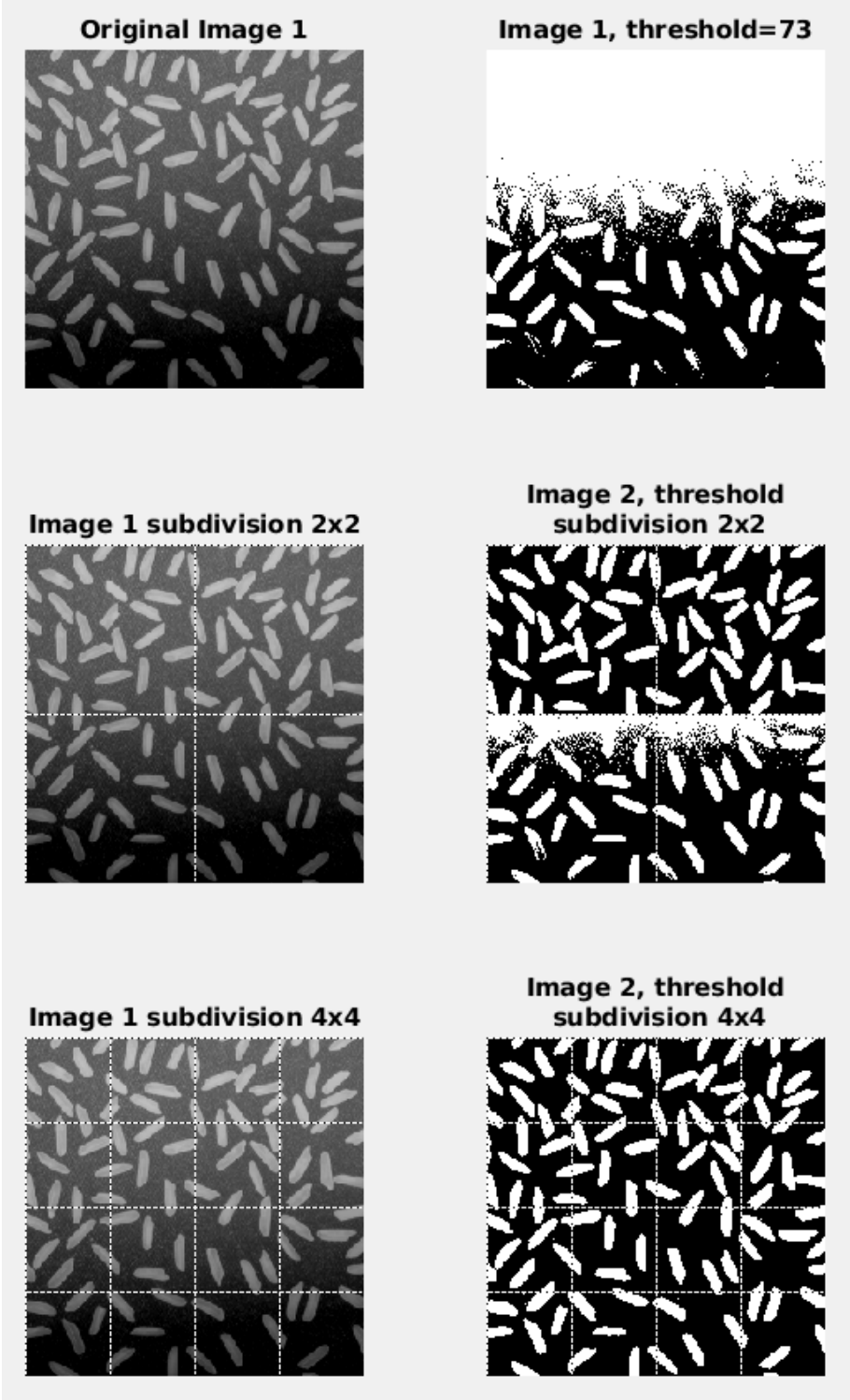
Image


Table 3: Image subdivision and OTSU's method (image)

1.3 Region growing

Code

```
%% Region growing
I_1 = im2double(imread('coins.png'));

% Seedpoints
x1=50; y1=50;
reg_maxdist = 0.2;
J1 = regiongrowing(I_1,x1,y1,reg_maxdist);

x2=100; y2=100;
reg_maxdist = 0.25;
J2 = regiongrowing(I_1,x2,y2,reg_maxdist);

x3=180; y3=240;
reg_maxdist = 0.3;
J3 = regiongrowing(I_1,x3,y3,reg_maxdist);

x4=10; y4=10;
reg_maxdist = 0.2;
J4 = regiongrowing(I_1,x4,y4,reg_maxdist);
```

Image

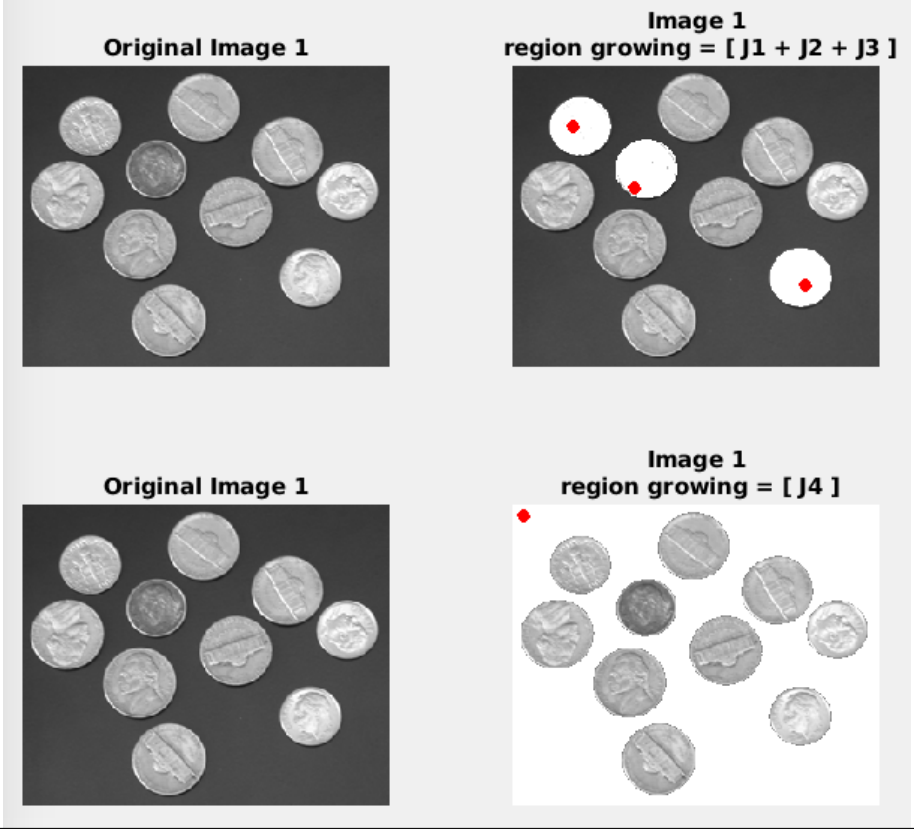


Table 4: Region growing

1.4 Clustering: K-Means

Code

```
%% Clustering K−Means
rng(4); % Try changing it!!
I_1 = rgb2gray(im2double(imread('univrLogo.jpeg')));
I_1 = imgaussfilt(I_1,1);
[m,n] = size(I_1);

[idx,C] = kmeans(I_1(:),2);
I_1_k2 = reshape(idx, m,n);

[idx,C] = kmeans(I_1(:),3);
I_1_k3 = reshape(idx, m,n);

[idx,C] = kmeans(I_1(:),4);
I_1_k4 = reshape(idx, m,n);
```

Image

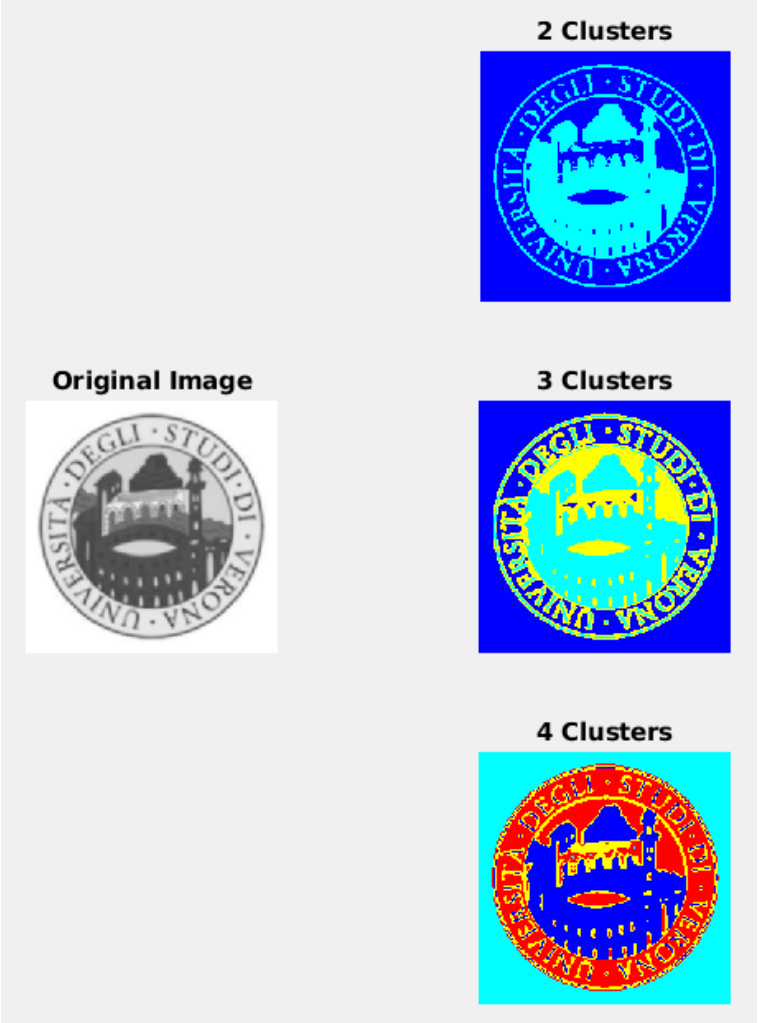


Table 5: Clustering: K-Means

Code

```
%% Clustering K−Means color
rng(1); % Try changing it!!
I_1 = im2double(imread('univrLogo.jpeg'));

R = I_1(:, :, 1);R = R(:);
G = I_1(:, :, 2);G = G(:);
B = I_1(:, :, 3);B = B(:);

vecRGB = [R G B];

[m,n,o] = size(I_1);

[idx,C] = kmeans(vecRGB,2);
I_1_k2 = reshape(idx, m,n);

[idx,C] = kmeans(vecRGB,3);
I_1_k3 = reshape(idx, m,n);

[idx,C] = kmeans(vecRGB,4);
I_1_k4 = reshape(idx, m,n);
```

Image

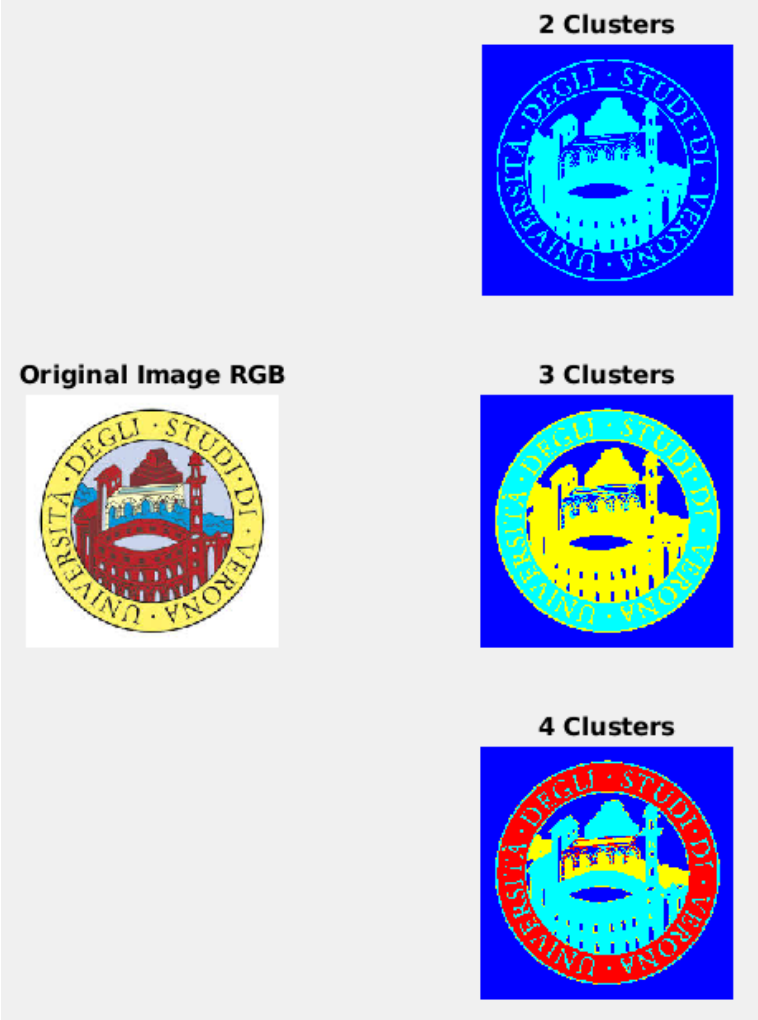
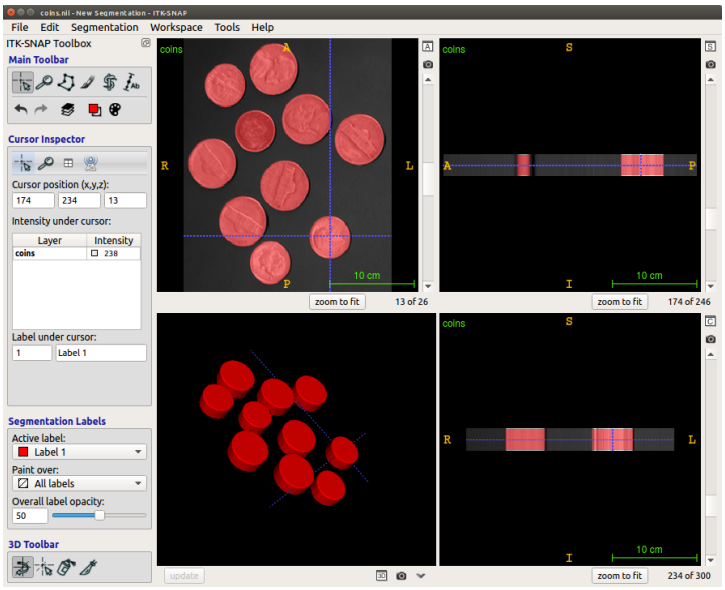


Table 6: Clustering: K-Means

1.5 Active contours: Snakes

Try the implementation of the Snakes algorithm of the ITK-SNAP program.



Use the next code to create files .nii, to try in the ITK-SNAP program.

Code

```
%% Create nii file
% if the function niftiwrite is not installed add the next libraries
% addpath(genpath('iptformats'));
% addpath(genpath('nifti'));
I_1 = imread('coins.png');
% Create 3D matrix
B = I_1;
for i=1:25
        B = cat(3, B, I_1);
end
niftiwrite(B,'coins.nii')
```

2 Assignment

1. Implement the K-means algorithm

Algorithm 1 K-means

1: Set $ic = 1$
2: Choose randomly a set of Kmeans: $m_1(1), ..., m_K(1)$
3: **for** each vector x_i **do**
4: **for** each $k = 1, ...K$ **do**
5: compute $D(x_i, m_k(ic))$
6: Assign x_i to the cluster C_j with nearest mean
7: Increment ic by 1
8: Update the means to get a new set $m_1(ic), ..., m_K(ic)$
9: Repeat steps 3 to 8 until $C_k(ic) = C_k(ic + 1)$ $\forall k$

where x_i is the value for the pixel i (grayscale or RGB array), C_j is the cluster j (set of pixels), m_j is the mean value of the pixels of the cluster C_j for $j = 1, ..., K$. The function of distance $D(x_i, m_k(ic))$ can be the Euclidean distance between x_i and $m_k(ic)$.

3 Solutions

1. Implement the K-means algorithm

Function myKmeans

```
function [idx] = myKmeans(vec, nk)
  ni = length(vec);
  m = vec(randi(ni,nk)); % random K means
  idx = zeros(1,ni);
  idxt = zeros(1,ni);

  for ite = 1:1000
    for i=1:ni
      minDist = abs(vec(i)-m(1));
      k = 1;
      for j=2:nk
        dist = abs(vec(i)-m(j)); % Distance
        if dist < minDist
          minDist = dist;
          k = j;
        end
      end
      idx(i) = k;
    end
    for j=1:nk
      xi = vec(idx == j);
      nxi = length(xi);
      m(j) = sum(xi)/nxi;
    end
    if sum(idx-idxt) == 0
      break;
    end
    idxt = idx;
  end
end
```

Example (myKmeans)

```
%% Clustering my K-Means
rng(2); % Try changing it!!
I_1 = im2double(imread('coins.png'));
I_1 = imgaussfilt(I_1,1);
[m,n] = size(I_1);

[idx] = myKmeans(I_1(:),2);
I_1_k2 = reshape(idx, m,n);

[idx] = myKmeans(I_1(:),3);
I_1_k3 = reshape(idx, m,n);
rng(6);
[idx] = myKmeans(I_1(:),4);
I_1_k4 = reshape(idx, m,n);
nr = 3;
nc = 2;
subplot(nr, nc, 3); imshow(I_1); title('Original Image');
subplot(nr, nc, 2); imshow(label2rgb(I_1_k2), [min(I_1_k2(:)) max(I_1_k2(:))]); title('2 Clusters');
subplot(nr, nc, 4); imshow(label2rgb(I_1_k3), [min(I_1_k3(:)) max(I_1_k3(:))]); title('3 Clusters');
subplot(nr, nc, 6); imshow(label2rgb(I_1_k4), [min(I_1_k4(:)) max(I_1_k4(:))]); title('4 Clusters');
```

Function myKmeansColor

```
function [idx] = myKmeansColor(vec, nk)
  [mi,ni] = size(vec);
  m = vec(randi(mi,nk,ni)); % random K means
  idx = zeros(mi,1);
  idxt = zeros(mi,1);

  for ite = 1:1000
    for i=1:mi
      minDist = norm(vec(i,:)-m(1,:));
      k = 1;
      for j=2:nk
        dist = norm(vec(i,:)-m(j,:)); % Distance
        if dist < minDist
          minDist = dist;
          k = j;
        end
      end
    idx(i) = k;
    end
    for j=1:nk
      xi = vec(idx == j,:);
      nxi = length(xi);
      m(j,:) = sum(xi,1)/nxi;
    end
    if sum(idx-idxt) == 0
      break;
    end
    idxt = idx;
  end
end
```

Example (myKmeansColor)

```
%% Clustering my K-Means color
rng(1); % Try changing it!!
I_1 = im2double(imread('univrLogo.jpeg'));
R = I_1(:, :, 1);R = R(:);
G = I_1(:, :, 2);G = G(:);
B = I_1(:, :, 3);B = B(:);

vecRGB = [R G B];
[m,n,o] = size(I_1);

[idx] = myKmeansColor(vecRGB,2);
I_1_k2 = reshape(idx, m,n);

[idx] = myKmeansColor(vecRGB,3);
I_1_k3 = reshape(idx, m,n);

[idx] = myKmeansColor(vecRGB,4);
I_1_k4 = reshape(idx, m,n);
nr = 3;
nc = 2;
subplot(nr, nc, 3); imshow(I_1); title('Original Image');
subplot(nr, nc, 2); imshow(label2rgb(I_1_k2)); title('2 Clusters');
subplot(nr, nc, 4); imshow(label2rgb(I_1_k3)); title('3 Clusters');
subplot(nr, nc, 6); imshow(label2rgb(I_1_k4)); title('4 Clusters');
```