

# Image processing for bioinformatics

Laboratory  
Segmentation

# 1 Examples, Matlab functions

## 1.1 Automatic thresholding: OTSU's method

Code		
<pre> 1 %% Otsu's method 2 I_1 = imread('coins.png'); 3 I_2 = imread('eight.tif'); I_2 = I_2(:,:,1); 4 I_3 = imread('riceG.png'); 5 % Get the threshold 6 otsuLevel_I_1 = graythresh(I_1); 7 otsuLevel_I_2 = graythresh(I_2); 8 otsuLevel_I_3 = graythresh(I_3); 9 % Compute binary image using the threshold 10 blackWhite_I_1 = imbinarize(I_1,otsuLevel_I_1); 11 blackWhite_I_2 = imbinarize(I_2,otsuLevel_I_2); 12 blackWhite_I_3 = imbinarize(I_3,otsuLevel_I_3); 13 % Histogram 14 [myHist_1,binLocations] = imhist(I_1); 15 [myHist_2,binLocations] = imhist(I_2); 16 [myHist_3,binLocations] = imhist(I_3); </pre>		
Image		

Table 1: OTSU's method

## 1.2 Variable thresholding: Image subdivision and OTSU's method

Code
<pre>1 %% Subdivision 2 I_1 = imread('riceG.png'); 3 4 otsuLevel_I_1 = graythresh(I_1); 5 blackWhite_I_1 = imbinarize(I_1,otsuLevel_I_1); 6 7 [m,n] = size(I_1); 8 9 % Subdivision image 1x1 -&gt; 2x2 subimages 10 I_2_cells = mat2cell(I_1,[m/2,m/2],[n/2,n/2]); 11 otsuLevel_I_2_cells = []; 12 blackWhite_I_2_cells = {}; 13 % For all subimages 14 for i=1:2 15     for j=1:2 16         % Get the threshold 17         otsuLevel_I_2_cells(i,j) = graythresh(I_2_cells{i,j}); 18         % Compute binary image using the threshold 19         blackWhite_I_2_cells{i,j} = imbinarize(I_2_cells{i,j},otsuLevel_I_2_cells(i,j)); 20     end 21 end 22 % Union subimage 2x2 -&gt; 1x1 Image 23 blackWhite_I_2_2 = cell2mat(blackWhite_I_2_cells); 24 25 26 % Subdivision image 1x1 -&gt; 4x4 subimages 27 I_2_cells = mat2cell(I_1,[m/4,m/4,m/4,m/4],[n/4,n/4,n/4,n/4]); 28 otsuLevel_I_2_cells = []; 29 blackWhite_I_2_cells = {}; 30 % For all subimages 31 for i=1:4 32     for j=1:4 33         % Get the threshold 34         otsuLevel_I_2_cells(i,j) = graythresh(I_2_cells{i,j}); 35         % Compute binary image using the threshold 36         blackWhite_I_2_cells{i,j} = imbinarize(I_2_cells{i,j},otsuLevel_I_2_cells(i,j)); 37     end 38 end 39 % Union subimage 4x4 -&gt; 1x1 Image 40 blackWhite_I_2_4 = cell2mat(blackWhite_I_2_cells);</pre>

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

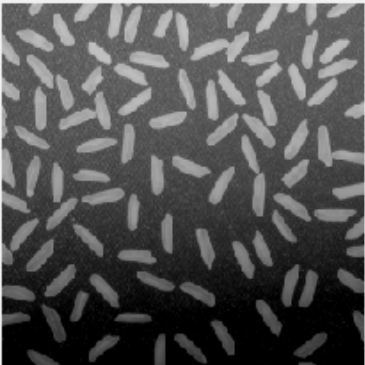

Image	
	<div> <div>Original Image 1</div>  </div>
	<div>Image 1, threshold=73</div> 

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

### 1.3 Region growing


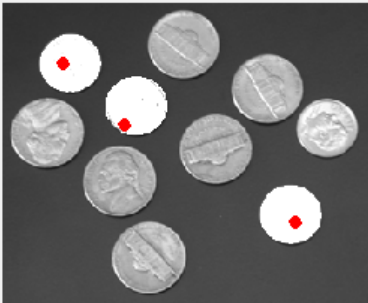

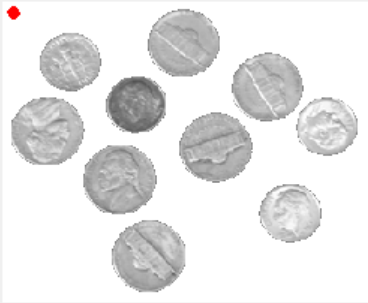
Code
<pre>1 %% Region growing 2 I_1 = im2double(imread('coins.png')); 3 4 % Seedpoints 5 x1=50; y1=50; 6 reg_maxdist = 0.2; 7 J1 = regiongrowing(I_1,x1,y1,reg_maxdist); 8 9 x2=100; y2=100; 10 reg_maxdist = 0.25; 11 J2 = regiongrowing(I_1,x2,y2,reg_maxdist); 12 13 x3=180; y3=240; 14 reg_maxdist = 0.3; 15 J3 = regiongrowing(I_1,x3,y3,reg_maxdist); 16 17 x4=10; y4=10; 18 reg_maxdist = 0.2; 19 J4 = regiongrowing(I_1,x4,y4,reg_maxdist);</pre>
Image
<div><div><p><b>Original Image 1</b></p></div><div><p><b>Image 1</b> <b>region growing = [ J1 + J2 + J3 ]</b></p></div><div><p><b>Original Image 1</b></p></div><div><p><b>Image 1</b> <b>region growing = [ J4 ]</b></p></div></div>

Table 4: Region growing

# 1.4 Clustering: K-Means





Code
<pre> 1 %% Clustering K-Means 2 rng(4); % Try changing it!! 3 I_1 = rgb2gray(im2double(imread('univrLogo.jpeg'))); 4 I_1 = imgaussfilt(I_1,1); 5 [m,n] = size(I_1); 6 7 [idx,C] = kmeans(I_1(:,2)); 8 I_1_k2 = reshape(idx, m,n); 9 10 [idx,C] = kmeans(I_1(:,3)); 11 I_1_k3 = reshape(idx, m,n); 12 13 [idx,C] = kmeans(I_1(:,4)); 14 I_1_k4 = reshape(idx, m,n); </pre>
Image
<div> <div>  <p>Original Image</p> </div> <div> <p>2 Clusters</p>  </div> <div> <p>3 Clusters</p>  </div> <div> <p>4 Clusters</p>  </div> </div>

Table 5: Clustering: K-Means

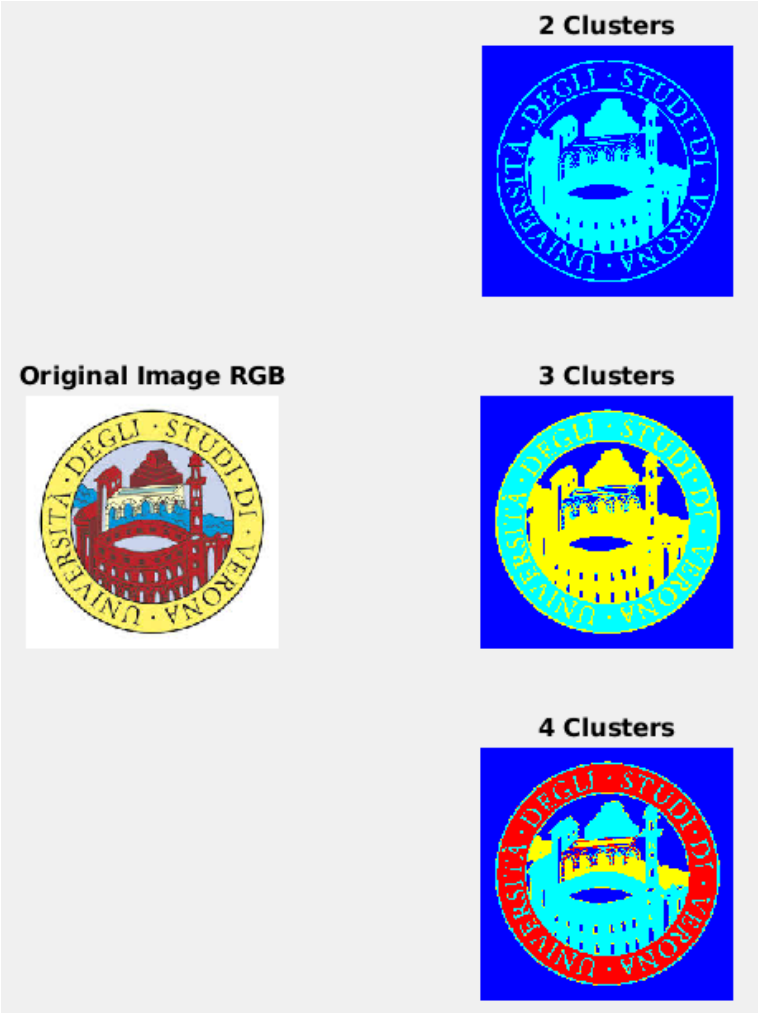
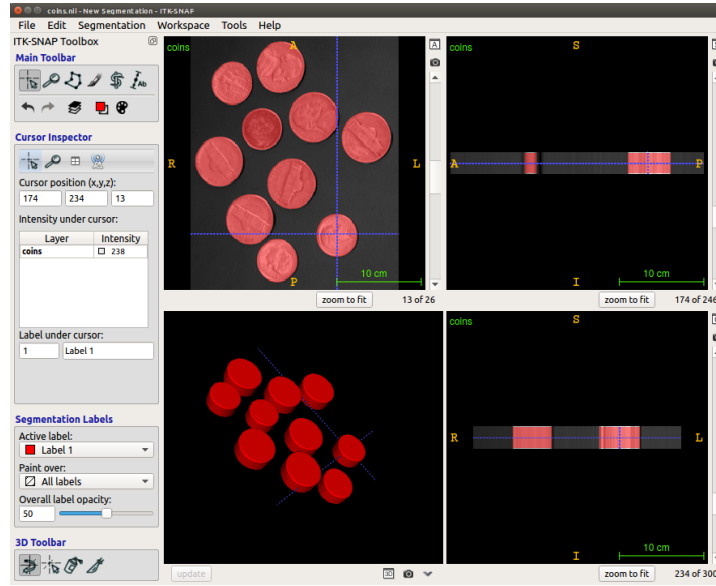
Code
<pre> 1 %% Clustering K-Means color 2 rng(1); % Try changing it!! 3 I_1 = im2double(imread('univrLogo.jpeg')); 4 5 R = I_1(:, :, 1);R = R(:); 6 G = I_1(:, :, 2);G = G(:); 7 B = I_1(:, :, 3);B = B(:); 8 9 vecRGB = [R G B]; 10 11 [m,n,o] = size(I_1); 12 13 [idx,C] = kmeans(vecRGB,2); 14 I_1_k2 = reshape(idx, m,n); 15 16 [idx,C] = kmeans(vecRGB,3); 17 I_1_k3 = reshape(idx, m,n); 18 19 [idx,C] = kmeans(vecRGB,4); 20 I_1_k4 = reshape(idx, m,n); </pre>
Image
 <p>The image displays the results of K-Means clustering on the University of Verona logo. It consists of four sub-images arranged in a 2x2 grid. The top-left image is the 'Original Image RGB', showing the logo in its natural colors. The top-right image, labeled '2 Clusters', shows the logo with two distinct colors: blue and yellow. The bottom-left image, labeled '3 Clusters', shows the logo with three distinct colors: blue, yellow, and red. The bottom-right image, labeled '4 Clusters', shows the logo with four distinct colors: blue, yellow, red, and green.</p>

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

```
1 %% Create nii file
2 % if the function niftiwrite is not installed add the next libraries
3 % addpath(genpath('iptformats'));
4 % addpath(genpath('nifti'));
5 I_1 = imread('coins.png');
6 % Create 3D matrix
7 B = I_1;
8 for i=1:25
9     B = cat(3, B, I_1);
10 end
11 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  $K$  means:  $m_1(1), \dots, m_K(1)$
  - 3: **for** each vector  $x_i$  **do**
  - 4:     **for** each  $k = 1, \dots, K$  **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  $m_1(ic), \dots, m_K(ic)$
  - 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  $j = 1, \dots, K$ . 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

```
1 function [idx] = myKmeans(vec, nk)
2   ni = length(vec);
3   m = vec(randi(ni,nk)); % random K means
4   idx = zeros(1,ni);
5   idxt = zeros(1,ni);
6
7   for ite = 1:1000
8     for i=1:ni
9       minDist = abs(vec(i)-m(1));
10      k = 1;
11      for j=2:nk
12        dist = abs(vec(i)-m(j)); % Distance
13        if dist < minDist
14          minDist = dist;
15          k = j;
16        end
17      end
18      idx(i) = k;
19    end
20    for j=1:nk
21      xi = vec(idx == j);
22      nxi = length(xi);
23      m(j) = sum(xi)/nxi;
24    end
25    if sum(idx-idxt) == 0
26      break;
27    end
28    idxt = idx;
29  end
30 end
```

Example (myKmeans)

```
1 %% Clustering my K-Means
2 rng(2); % Try changing it!!
3 I_1 = im2double(imread('coins.png'));
4 I_1 = imgaussfilt(I_1,1);
5 [m,n] = size(I_1);
6
7 [idx] = myKmeans(I_1(:,2));
8 I_1_k2 = reshape(idx, m,n);
9
10 [idx] = myKmeans(I_1(:,3));
11 I_1_k3 = reshape(idx, m,n);
12 rng(6);
13 [idx] = myKmeans(I_1(:,4));
14 I_1_k4 = reshape(idx, m,n);
15 nr = 3;
16 nc = 2;
17 subplot(nr, nc, 3); imshow(I_1); title('Original Image');
18 subplot(nr, nc, 2); imshow(label2rgb(I_1_k2), [min(I_1_k2(:)) max(I_1_k2(:))]); title('2 Clusters');
19 subplot(nr, nc, 4); imshow(label2rgb(I_1_k3), [min(I_1_k3(:)) max(I_1_k3(:))]); title('3 Clusters');
20 subplot(nr, nc, 6); imshow(label2rgb(I_1_k4), [min(I_1_k4(:)) max(I_1_k4(:))]); title('4 Clusters');
```

### Function myKmeansColor

```

1 function [idx] = myKmeansColor(vec, nk)
2 [mi,ni] = size(vec);
3 m = vec(randi(mi,nk,ni)); % random K means
4 idx = zeros(mi,1);
5 idxt = zeros(mi,1);
6
7 for ite = 1:1000
8     for i=1:mi
9         minDist = norm(vec(i,:)-m(1,:));
10        k = 1;
11        for j=2:nk
12            dist = norm(vec(i,:)-m(j,:)); % Distance
13            if dist < minDist
14                minDist = dist;
15                k = j;
16            end
17        end
18        idx(i) = k;
19    end
20    for j=1:nk
21        xi = vec(idx == j,:);
22        nxi = length(xi);
23        m(j,:) = sum(xi,1)/nxi;
24    end
25    if sum(idx-idxt) == 0
26        break;
27    end
28    idxt = idx;
29 end
30 end

```

### Example (myKmeansColor)

```

1 %% Clustering my K-Means color
2 rng(1); % Try changing it!!
3 I_1 = im2double(imread('univrLogo.jpeg'));
4 R = I_1(:, :, 1); R = R(:);
5 G = I_1(:, :, 2); G = G(:);
6 B = I_1(:, :, 3); B = B(:);
7
8 vecRGB = [R G B];
9 [m,n,o] = size(I_1);
10
11 [idx] = myKmeansColor(vecRGB,2);
12 I_1_k2 = reshape(idx, m,n);
13
14 [idx] = myKmeansColor(vecRGB,3);
15 I_1_k3 = reshape(idx, m,n);
16
17 [idx] = myKmeansColor(vecRGB,4);
18 I_1_k4 = reshape(idx, m,n);
19 nr = 3;
20 nc = 2;
21 subplot(nr, nc, 3); imshow(I_1); title('Original Image');
22 subplot(nr, nc, 2); imshow(label2rgb(I_1_k2)); title('2 Clusters');
23 subplot(nr, nc, 4); imshow(label2rgb(I_1_k3)); title('3 Clusters');
24 subplot(nr, nc, 6); imshow(label2rgb(I_1_k4)); title('4 Clusters');

```