LABORATORY EXPERIENCE #8
(Images of moles)

Introduction

In this experiment, we apply **K-means clustering** algorithm to reduce the number of colours in images of moles and to analyse the resulting image to find borders of the moles.
The archive of images contains photos of moles divided into three categories: 'low risk', 'medium risk' and 'melanoma'.

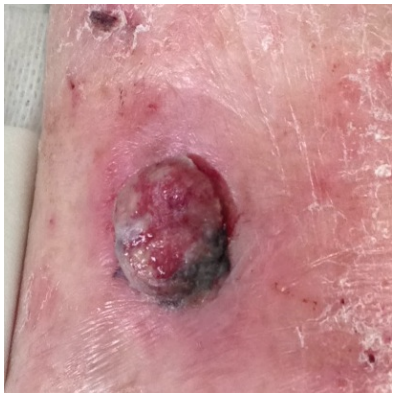
Data preparation

Matlab provides a function to read images, which output is a 3-dimensional matrix defined by the two dimensions of the image in pixels and the quantization levels for each colour: red, green and blue (**RGB**). To ease the processing, this matrix has been reshaped collapsing the two dimensions of the image in a unique dimension.

Algorithms implementation and results

A simplified version of the K-means algorithm is implemented, not taking into consideration the variances and the probabilities of each clusters, but only comparing the **squared norm distance** of each pixel of the image to the RGB triplets that identify the centroids.
The number of clusters in which divide the pixels is chosen as K=4, but depending on how the centroids are initialized by the random generator, it can happen to have all the data classified into 3 or even 2 clusters. Anyway, once the clustering is performed, each pixel is re-assigned the value of RGB triplet defined by its correspondent cluster and a new image is created.
This method should allow a better recognition of the borders of the moles, that in this way appear to be better defined.
'Active contours implementation & test platform' application is used to perform this test: through a graphical interface it is possible to define a mask of the mole, that will be used to characterize the contours. If the border of the mole is not well defined, this process might take a larger number of iterations.
The pictures below show two examples of clustering and contour definition respectively for a 'low risk' example and for a 'melanoma' case.

Figure 1: a-Original image 'low risk'; b-Image after clustering; c-Contour definition with 40 iterations



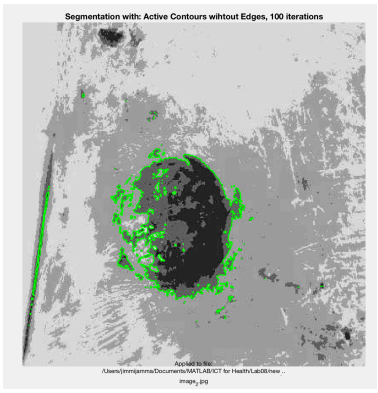


Figure 1: a-Original image 'melanoma'; b-Image after clustering; c-Contour definition with 100 iterations

Data Preparation

```
clear all
close all
clc

A=imread('melanoma_4.jpg');

figure
imshow(A);


% A is a 3D-matrix made of 3 N1 ? N2 matrices, the first one stores
% unsigned integers from 0 to 255 that encode red, the second green,
% the third blue.

[N1,N2,N3]=size(A);
N=N1*N2;% N is the total number of pixels
B=double(reshape(A,N,N3));

% B is a matrix with N rows and 3 columns: the three columns can be
 thought
% of as features, and we can apply the clustering algorithms like the
% hard and soft K-means.


F=N3;
y=B;
```

--------- Hard K-Means Algorithm -------- %%

```
% starting conditions
K=4; % number of clusters

% we can decide to start with x_k random, or taken as the mean of the
 two
% known classes. In a clustering case, the random choice is more
 realistic
rng('shuffle');
x_k = rand(K,F)*255;
init = x_k;

for iteration=1:10
    % evaluating the distance
    for n=1:N
        for k=1:K
            dist(n,k)=norm(y(n,:)-x_k(k,:)).^2;
        end
    end

    [M,decision]=min(dist,[],2); % taking the decision
```

```
    %'decision' is an array of length N with the corresponding closest
 region
    % for each element

    for k=1:K
        w_k=y(find(decision==k),:);
        N_k(k) = size(w_k,1);
        x_k(k,:) = mean(w_k,1);
    end

end

for k=1:K
    i_k=find(decision==k);
    n_k(k)=length(i_k);
    for ii=1:n_k(k)
        y(i_k(ii),:)=x_k(k,:);
    end
end

Bnew=floor(y);
Anew=reshape(uint8(Bnew),N1,N2,N3);
imwrite(Anew,'new_image_2.jpg');

figure
imshow(Anew);
```

Published with MATLAB® R2016a