Dpto. Ingeniería de Sistemas y Automática
Universidad de Málaga

COMPUTER VISION

EXERCISE 3a: Image segmentation with region growing

Concepts: Region growing, seed

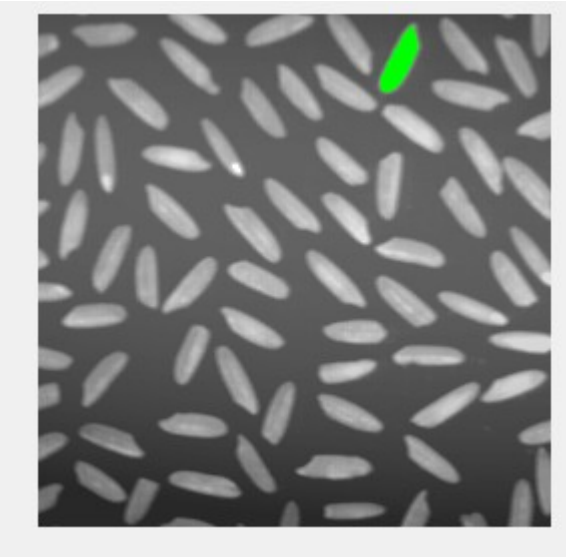
The code shown below extracts a region of the image by a growing process. This process starts with a seed in the (*x*,*y*) coordinates and adds pixels to the region recursively. Implement a code to:

- Load the *rice.tif* image and initializes the global variables **region**, **media**, **points_in_region** and **im1**.
- Permits the user to indicate a seed and initializes, using that seed, the growing process. You can use the **ginput** command for that. *Note: when testing the code, choose the seed in such a way that it falls into a rice grain.*
- Execute the region growing function (**crec_recur**), and draw the extracted region in the initial image as it is shown in the result image.

Code

```
% Esta función realiza de manera recursiva el crecimiento de una región de la imagen
% partiendo de una semilla en (x,y). El crecimiento se hace sobre sus ocho-vecinos.
%
% J. Gonzalez - Mayo 2005
function crec_recur(x,y,toler)

% Variables globales para no pasar tantos argumentos a la funcion
global region;           % Region segmentada. Es una matriz del tamaño la imagen,con
                         %      todo a cero salvo la region, a uno.
global media;            % Media (dinamica) de la region que va creciendo.
global points_in_region; % Numero de puntos actual de la region segmentada
global im1;              % Imagen de entrada en escala de grises

% Comprobacion que no estamos fuera de la imagen
if (x <= 0 | x > size(im1,2) | y <= 0 | y > size(im1,1) )
    return;
end
if (region(y,x) == 1) % si esta ya marcado (=1) no hacemos nada y salimos de la
funcion
    return;
end

if abs(double(im1(y,x)) - media) < t % Condicion de añadir pixel
    region(y,x) = 1;
    points_in_region = points_in_region + 1;
    media = (media * points_in_region + double(im1(y,x)))/(points_in_region+1);
%calculo dinamico de la media

    % Recursion sobre los 8-vecinos
    crec_recur(x-1,y+1,toler);
    crec_recur(x-1,y,toler);
    crec_recur(x-1,y-1,toler);
    crec_recur(x,y+1,toler);
    crec_recur(x,y-1,toler);
    crec_recur(x+1,y+1,toler);
    crec_recur(x+1,y,toler);
    crec_recur(x+1,y-1,toler);
end

return;
```

Results



The algorithm tries to extract a region from the image recursively, so I had some troubles of stack overflowing caused by a high threshold. I also noticed that the rows and columns of the image were rotated respect to the values that ginput returned, but after some debugging I managed to solve that issue.

Although this algorithm seems to work great with this example, it still fails with some grains of rice that have a brigther center.

EXERCISE 3b: Image segmentation using K-means

Concepts: K-means

The K-means algorithm is used for clustering, since it permits us to group data (in our case, pixels) relying on a certain criteria, for example, a similar intensity level. In the code example below it is illustrated how to use the Matlab function **kmeans** using as features the image pixel intensities.

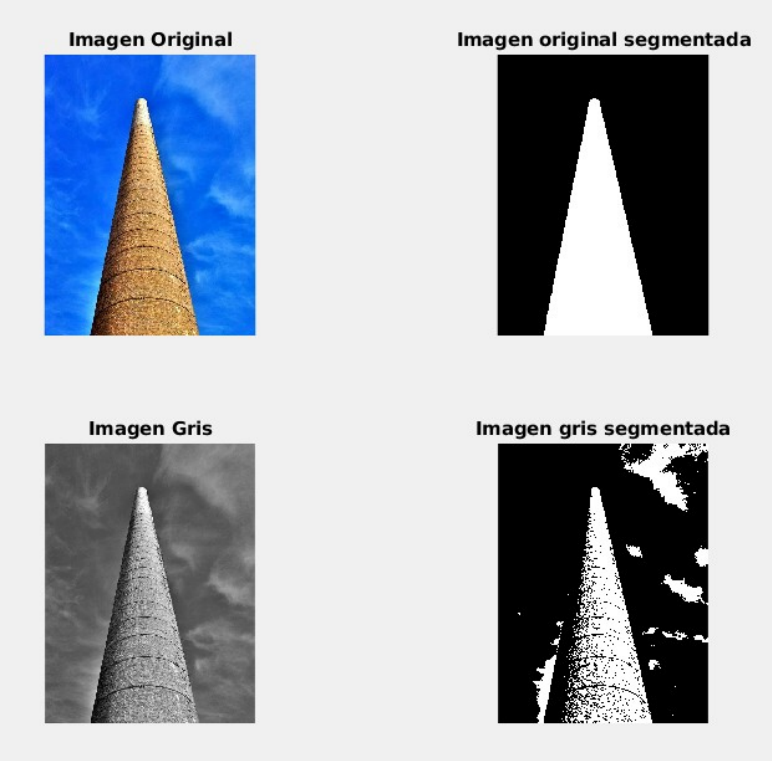
- Modify the code to segment objects in the RGB 3D space, in other words, the feature of each pixel is its RGB color.
- Segment the *torre_monica.jpg* image into two regions.
- Compare the resultant segmentation with the one obtained for the same image but in grayscale.

Note: *for more information about* ***kmeans*** *you can check the Matlab help and the lectures' material.*

Example:

```
im = imread('alumgrns.tif');
figure, subplot(1,2,1), imshow(im), title('Imagen original')
nPixels = prod(size(im));
data = reshape(im, nPixels, 1); %Image as a vector
pert = kmeans(double(data), 4); %
clus=reshape(pert, size(im)); %Vector Image back to a matrix
im_clust=uint8(255*(clus-1)/(max(max(clus))-1));
subplot(1,2,2), imshow(im_clust), title('Imagen segmentada')
```

Results



We can clearly see how well this algorithm performs with the RGB image, but we can see that it's easily tricked by intensities in a gray image, and doesn't manage to get some darker parts of the tower and also have some false positives in the background.

OPTIONAL! EXERCISE 3c: Image segmentation with EM (Expectation Maximization)

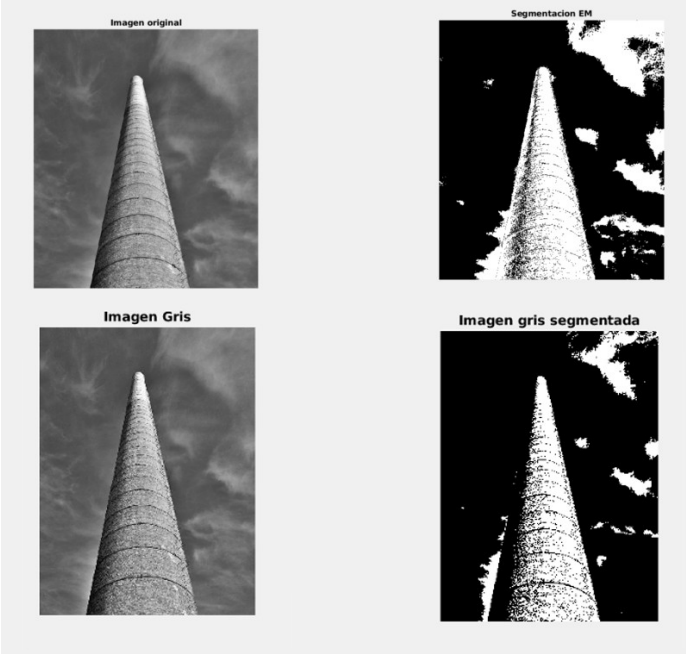
Concepts: EM

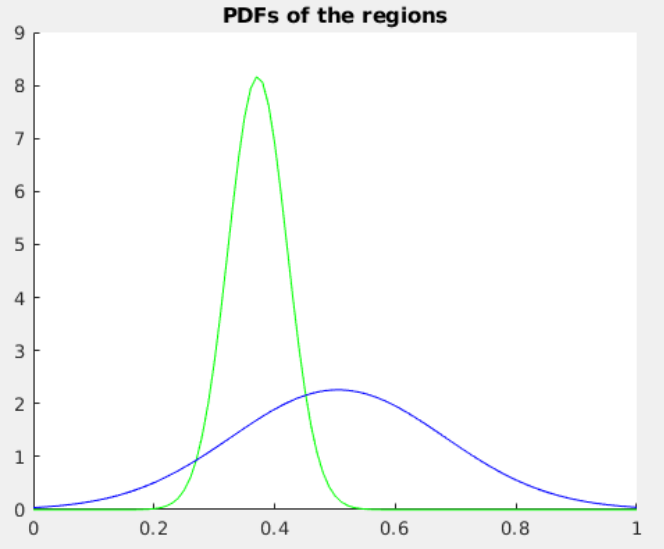
The EM algorithm assigns to each region or cluster a Gaussian probability distribution and enables us to group data (in our case, pixels) by maximizing the probability of the data (pixels) belonging to each region.

- Implement a code that segments an image in grayscale (using only the pixel intensity as data) by means of EM.
- Segment the *torre_monica.jpg* into two regions using **segmentation_em**, available as material for the course. *Note: the **sementation_em** function expects the image in double format. You can use **im2double** for that.*
- Compare the results with the output of the K-means algorithm working with the grayscale image and justify the differences.
- Interpret the probability density functions of the obtained regions. *Note: these densities are automatically displayed by the **segmentation_em** function.*

Example:

```
im = imread('torre_monica.jpg'); %Convert to grayscale & double
clus = segmentation_em(im, num_seg); %Image with segm. result
im_clust = uint8(255*(clus-1)/(max(max(clus))-1));
figure; subplot(1,3,1); imshow(im); title('Imagen original');
subplot(1,3,2); imshow(im_km); title('Segmentacion K-Means');
subplot(1,3,3); imshow(im_clust); title('Segmentacion EM');
```





We can see that both algorithms performs similarly. However, EM ensures that it will clasify all the points outputting the minimum possible error, as it takes into account not only the euclidian distance as K-Means, but also the sigma value of each gaussian function that we can see in the graph. It still performs worse than K-Means with an RGB image, but might be better for grayscale images.