# COMPUTER VISION

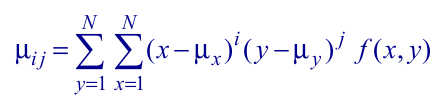
**EXERCISE 5a: REGION DESCRIPTORS**

Concepts: Hu moments

1. **Central moments:** Implement a function that computes the central moments (until order 3) of a grayscale image. The function prototype must be as follows: *Note: it really helps if you implement another function to compute the non-central, or raw moments, and you use them to retrieve the central ones. If not, you can use the expression below.*

[mu00,mu10,mu01,mu11,mu20,mu02,mu21,mu12,mu30,mu03]=momentos\_centrales(I)

Central moments expression:



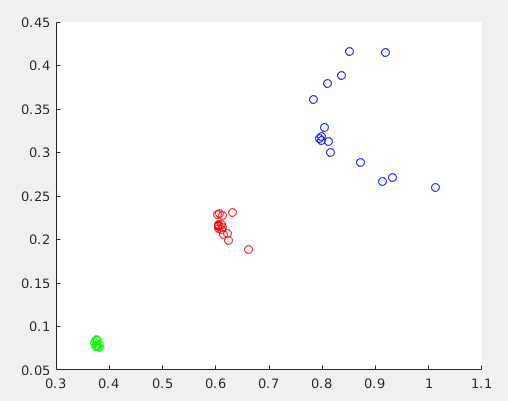
To test your code, if you run the function with the ‘’ as argument, you have to obtain the following results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **m00** | 4.5022e+03 | **m10** | 0 | **m01** | 0 |
| **m11** | -4.4394e+05 | **m20** | 1.2790e+06 | **m02** | 1.0934e+07 |
| **m21** | -4.3338e+06 | **m12** | 2.2195e+07 | **m30** | 8.9530e+06 |
| **m03** | -1.7208e+08 |  |  |  |  |

*This exercise was almost trivial and it was just fill a bunch of formulas in the script in order to calculate all the central moments. Firstly I created the function* ***momentos(I)****, which computed the non-central moments and then used the result to calculate the central moments.*

1. **Hu moments:** For the grayscale images attached, corresponding to 3 different types of bottles, do the following:
   1. Compute the Hu moments employing the “momentos\_Hu” function included below. This functions relies on the “momentos\_centrales” function implemented in the previous point.
   2. Graphically represent the values of the two first Hu moments. You should employ a different color for each bottle type. *Note: use only the first 15 images of each type.*

*Here, I used a for loop to iterate over all the 45 photos. After that, I just calculated the Hu moments of each image using the provided function and afterwards plotted the first two values in a graph. I colored each point based on its category.*

**

1. **Centroid:** Compute the centroid (center of mass) of the first two Hu moments of each bottle type and display them in the previous figure. Employ a different mark to distinguish them from the other points.

*I just calculated the centroid of each category and plotted it on top of the graph from the previous exercise. As an appointment, I made a separated function to calculate it.*

1. **Euclidean classifier:** Implement a Matlab script that:
   1. Asks for the name of an image through the keyboard and read it.
   2. Computes the two first Hu moments of that image. This will be the descriptors vector.
   3. Compares such a vector with the center of mass of each bottle type retrieved in the previous point. *Note: to compare two vectors employ the Euclidean distance.*
   4. Shows in the screen the type of the bottle in the image.

*This exercise was relatively easy, as it was only a slight modification of the previous one. I just took the euclidean distance of each category centroid to the centroid of the input image. The closest category will be the correct one. It turned out to work just great, being successful of classifying bottles in the correct category even when they were not part of the “training set”.*

**Function “momentos\_Hu”**

|  |
| --- |
| function HM=momentos\_Hu(I)  % Calcula los momentos de Hu invariantes de una imagen (I) en niveles de gris  % Si se desea obtener la descripción de momentos de un único objeto de la  % imagen, el resto hay que ponerlos a cero.  %  % Entrada: Imagen I en niveles de gris  % Salida: Vector HM (7x1) de momentos de Hu (invariantes)  %  % Fecha: 2009-2012 Javier Gonzalez    I=double(I)/255;    % Momentos centrales  [mu00,mu10,mu01,mu11,mu20,mu02,mu21,mu12,mu30,mu03] = momentos\_centrales(I);    %Momentos normalizados  u002 = mu00\*mu00;  u0025 = mu00^2.5;  %u0015 = mu00^1.5  n02 = mu02/u002;  n20 = mu20/u002;  n11 = mu11/u002;  n12 = mu12/u0025;  n21 = mu21/u0025;  n03 = mu03/u0025;  n30 = mu30/u0025;    %Momentos invariantes de Hu  f1 = n20+n02;  f2 = (n20-n02)^2 + 4\*n11^2;  f3 = (n30-3\*n12)^2+(3\*n21-n03)^2;  f4 = (n30+n12)^2+(n21+n03)^2;  f5 = (n30-3\*n12)\*(n30+n12)\*((n30+n12)^2 - 3\*(n21+n03)^2) + (3\*n21-n03)\*(n21+n03)\*(3\*(n30+n12)^2 - (n21+n03)^2);  f6 = (n20-n02)\*((n30+n12)^2 - (n21+n03)^2) + 4\*n11\*(n30+n12)\*(n21+n03);  f7 = (3\*n21-n03)\*(n30+n12)\*((n30+n12)^2 - 3\*(n21+n03)^2) - (n30-3\*n12)\*(n21+n03)\*(3\*(n30+n12)^2 - (n21+n03)^2);  HM = [f1 f2 f3 f4 f5 f6 f7];  return; |

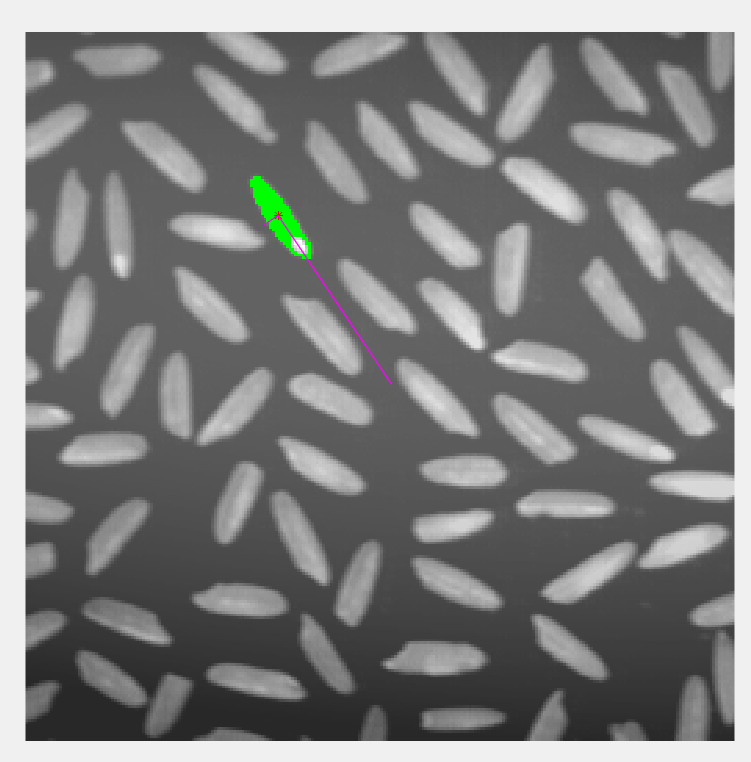
**OPTIONAL! EXERCISE 5b: Centroid and principal directions**

Concepts: Centroid and principal direction. Eigenvalue and eigenvector of a dispersion matrix.

Write a function that computes and draws the centroid and principal direction of the segmented region, given by a binary image (input argument to the function) where the background has a value of 0 and the segmented region a value of 1. *Note: for the segmentation step you can use the region growing algorithm.*

**Matlab functions to use:**

|  |  |
| --- | --- |
| **[V,D] = eig(A)** | Computes the eigenvalues and eigenvectors of the matrix A. |
| **line ([x0,x1],[y0,y1], 'LineWidth',1, 'Color',[1 0 0])** | Draws a line from (xo,y0) to (x1,y1) with the specified parameters. |



*For this last exercise, I used some code that I’ve already implemented in the last practise (crec\_recur) to get the region of the grain of rice. After that, I computed the centroid and central moments of the region in order to get the dispersion matrix. The I’ve used the eigenvector and eigenvalues to get the two principal directions of the objects and plotted them as lines in the image. I’ve also coloured the region in green so it was easier to see.*