# COMPUTER VISION

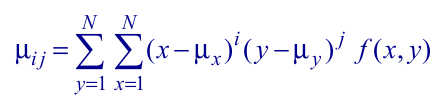
**EXERCISE 5a: REGION DESCRIPTORS**

Concepts: Hu moments

1. **Central moments:** Implement a function that computes the central moments (until order 3) of a grayscale image *I*. 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 it 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 *‘botella\_A\_1.bmp’* as argument, you have to obtain the following results:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **mu00** | 1148052 | **mu10** | 0 | **mu01** | 0 |
| **mu11** | -4.3331e+06 | **mu20** | -2.6373e+06 | **mu02** | -7.1258e+06 |
| **mu21** | 1.0521e+07 | **mu12** | 1.9549e+07 | **mu30** | 5.1157e+06 |
| **mu03** | 3.4196e+07 |  |  |  |  |

[m00,m10,m01,m11,m20,m02,m21,m12,m30,m03]=momentos\_no\_centrales(I);

xm=m10/m00;

ym=m01/m00;

u=m00;

mu00=m00;

mu01=0;

mu10=0;

mu20=m20-u\*xm^2;

mu11=m11-u\*ym\*xm;

mu02=m02-u\*ym^2;

mu30=m30-3\*m20\*xm+2\*m10\*xm^2;

mu21=m21-2\*m11\*xm-m20\*ym+2\*m01\*xm^2;

mu12=m12-2\*m11\*ym-m02\*xm+2\*m10\*ym^2;

mu03=m03-3\*m02\*ym+2\*m01\*ym^2;

function [m00,m10,m01,m11,m20,m02,m21,m12,m30,m03]=momentos\_no\_centrales(I)

m00=moment\_no\_central(0,0,I);

m01=moment\_no\_central(0,1,I);

m10=moment\_no\_central(1,0,I);

m20=moment\_no\_central(2,0,I);

m11=moment\_no\_central(1,1,I);

m02=moment\_no\_central(0,2,I);

m03=moment\_no\_central(0,3,I);

m12=moment\_no\_central(1,2,I);

m21=moment\_no\_central(2,1,I);

m30=moment\_no\_central(3,0,I);

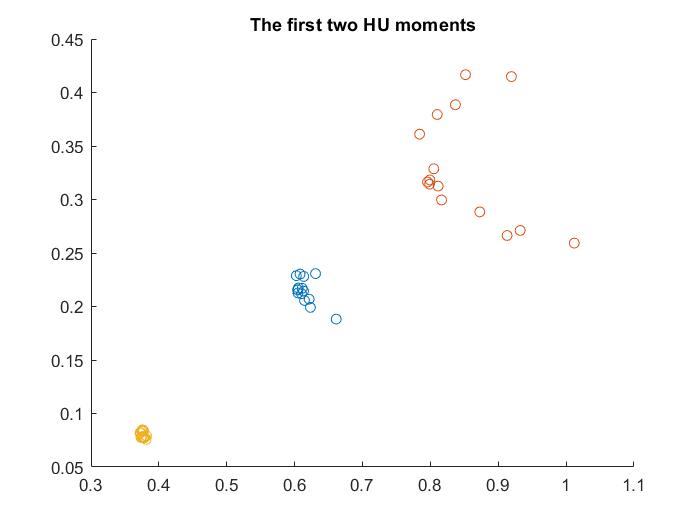
function y=moment\_no\_central(i,j,I)

[M,N]=size(I);

ax=(1:N).^i; ay=(1:M).^j;

y=ax\*I'\*ay';

1. **Hu moments:** For the grayscale images attached, corresponding to 3 different types of bottles, do the following: *Note: use only the first 15 images of each type.*
   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.

****

Plot 1

As we see in the plot, we can distinguish three different group, which are the different type of bottles we have. From this we could analyse a picture and say which type of bottle is it, but it is difficult to do it.

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.

im=['A','B','C'];

centroid=zeros(2,3);

for i=1:length(im)

l=im(i);

hu=zeros(7,15);

for j=1:15

hu(:,j)=

momentos\_Hu(imread(sprintf('botella\_%s\_%d.bmp',l,j));

end

scatter(hu(1,:),hu(2,:));

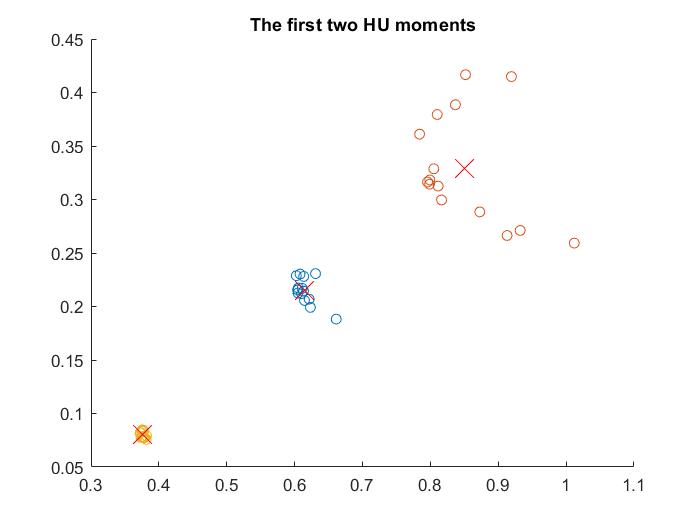
centroid(1,i)=mean(hu(1,:));

centroid(2,i)=mean(hu(2,:));

end

scatter(centroid(1,:),centroid(2,:),250,'rx');

title('The first two HU moments');

****

Plot 2

As we see in the plot, we can distinguish three different group, which are the different type of bottles we have. Furthermore, this group have each one a centre which we could use to classify an image into the three types of bottle. This time s more simple because we could use the minimum distance between them.

1. **Euclidean classifier:** Implement a Matlab script that:
   1. Asks for the name of an image through the keyboard and read it.

str\_im = input('Introduce the image name of the bottle\n');

* 1. Computes the two first Hu moments of that image. This will be the descriptors vector.

im\_hu = momentos\_Hu(im);

* 1. 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.*

for i=1:length(im\_type)

( … )

d(i)=pdist2(centroid,im\_hu(1:2),'euclidean');

end

* 1. Shows in the screen the type of the bottle in the image.

[~,point] = min(d);

fprintf('The image corresponds to a %s type bottle\n',im\_type(point));

**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.*

function Centroid\_and\_principal\_direction(I)

global center;

[mu00,~,~,mu11,mu20,mu02,~,~,~,~]=momentos\_centrales(I);

plot(center(1),center(2),'r+');

Cov=1/mu00.\*[mu20 mu11;mu11 mu02];

[V,D]=eig(Cov);

V1=V(:,1); V2=V(:,2);

a1=D(1,1); a2=D(2,2);

x0=center(1); y0=center(2);

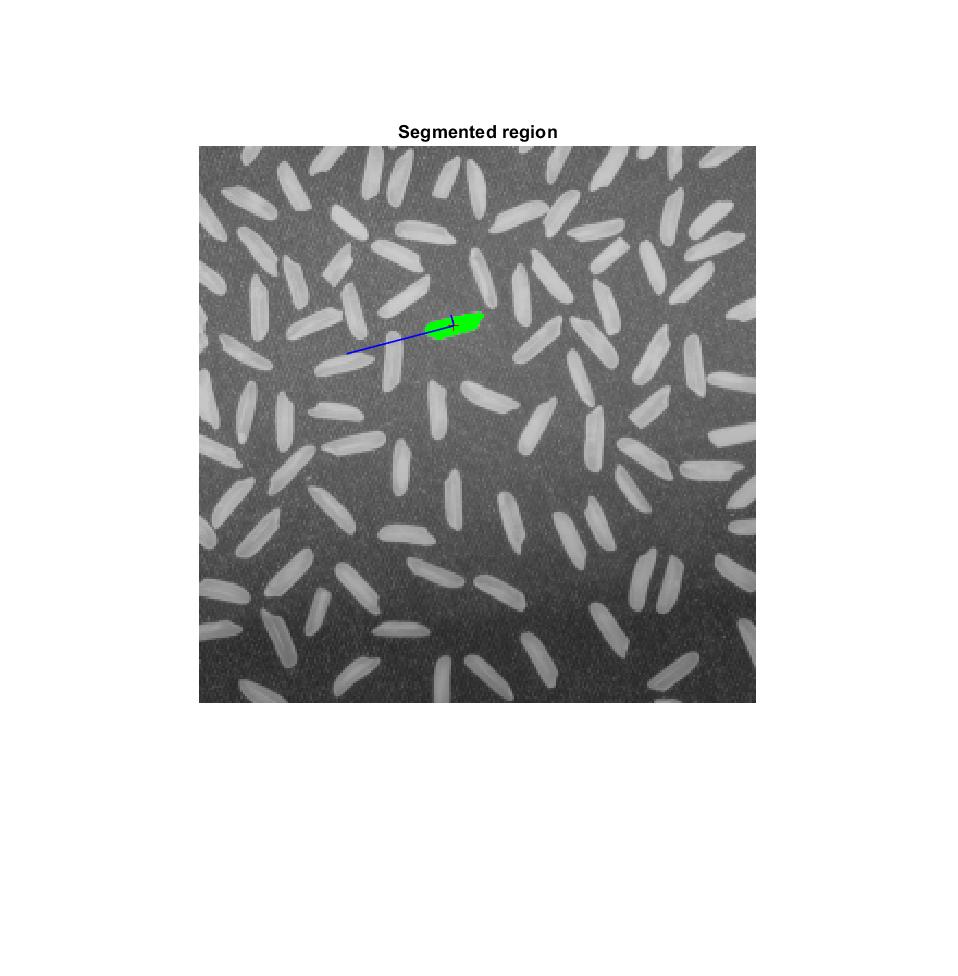
x1=V1(1)\*a1+x0; y1=V1(2)\*a1+y0;

line ([x0,x1],[y0,y1], 'LineWidth',1, 'Color',[0 0 1]);

x1=V2(1)\*a2+x0; y1=V2(2)\*a2+y0;

line ([x0,x1],[y0,y1], 'LineWidth',1, 'Color',[0 0 1]);

end



Image

As we see in the image, the vector indicates the orientation of the segmented region and the intensity of each orientation from the base x=(1,0) y=(0,1). The one who have the major intensity indicate the predominant orientation.