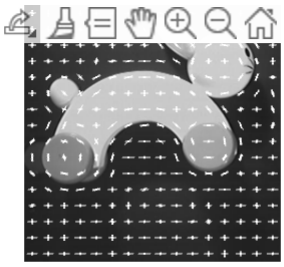
Exercici 1. Dibuixar un rectangle en la posició pos_block

Codi a classe:

```
close all
I = imread("rabbit.jpg");
I = imresize(I,[256,256]); % mida mes standard

[HoG, HoGVec] = extractHOGFeatures(I,'CellSize', [16,16], 'BlockSize',[3,3], ...
    'NumBins',9);
imshow(I);
hold on
plot(HoGVec);

% Busquem un retall per HoG
[c,f] = getpts;
```



```
rect = [c-24,f-24,48,48];
Retall = imcrop(I, rect);
figure
imshow(Retall);
```

```


HoGR = extractHOGFeatures(Retall,'CellSize',[16,16],'BlockSize',[3,3], ...
    'NumBins',9);
FIG = reshape(HoG,[81,14*14])';

semblanca = zeros([14*14,1]);
for i = 1:14*14
    semblanca(i) = sum(min(FIG(i,:), HoGR)); % similitud per interseccio de l'histogram
end

[val, pos_block] = max(semblanca);
```

Implementació:

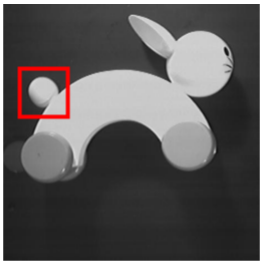
Partim de que ja tenim pos_block calculat. Per tant, per poder dibuixar un rectangle a aquella posició, simplement hem de calcular quines són les seves coordenades i dibuixar-lo.

Sabem que FIG té 14x14 blocs. Per saber la coordenada x hem de dividir l'índex de la posició (pos_block) amb el nombre de files (14). I per saber la coordenada y ens quedem la resta de la divisió de pos_block/14. Per tema indexació, no oblidar-se del -1 de la posició i el +1 en el total. Sinó, quedaria desplaçat en la imatge.

```
block_size = 3;
cell_size = 16;
x = (ceil(pos_block/14)-1)*cell_size+1;
y = (mod(pos_block,14)-1)*cell_size+1;
```

Finalment pintarem el rectangle de color vermell, amb un grosor de 2 px. Aquest rectangle ha de queda en la posició de les coordenades obtingudes.

```
figure, imshow(I);
hold on
rectangle('Position', [x, y, block_size*cell_size, block_size*cell_size], ...
```



```
    'EdgeColor', 'R', 'LineWidth', 2);
```

Diferenciar histogrames de color

En aquest exercici hem agafat 6 imatges de jugadors de futbol. La primera serà la imatge del jugador model. El que es busca en aquest exercici és trobar quina de la resta d'imatges és un jugador del mateix equip. Això serà possible gràcies a la comparació de histogrames de color de cada una de les imatges i comparant-la amb la imatge model.

En aquest cas, com podem veure, la imatge model és un jugador del Barça i de la resta d'imatges només el primer jugador és del mateix equip. Per tant, per veure que el programa funciona correctament, cada gràfic de barres final haurà de mostrar que la primera barra està, suposadament, al màxim. Això és degut a que normalitzem els valors per a que capiguen dins del rang de 0 a 100; sent 100 la imatge que més s'assimilia a la model.

```
close all
im1 = imread('team1.jpg');
im2 = imread('team2.jpg');
im3 = imread('team5.jpg');
im4 = imread('team11.jpg');
```

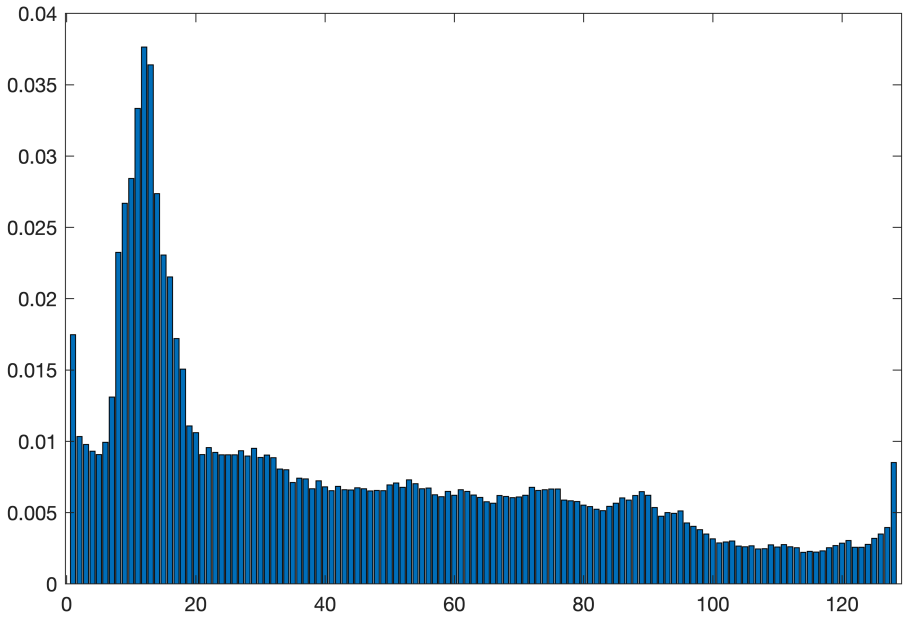
```
im5 = imread('team12.jpg');
im6 = imread('team8.jpg');

montage({im1,im2,im3,im4,im5,im6});
```

Un cop tenim les imatges seleccionades procedirem a fer el següent flow de feina:

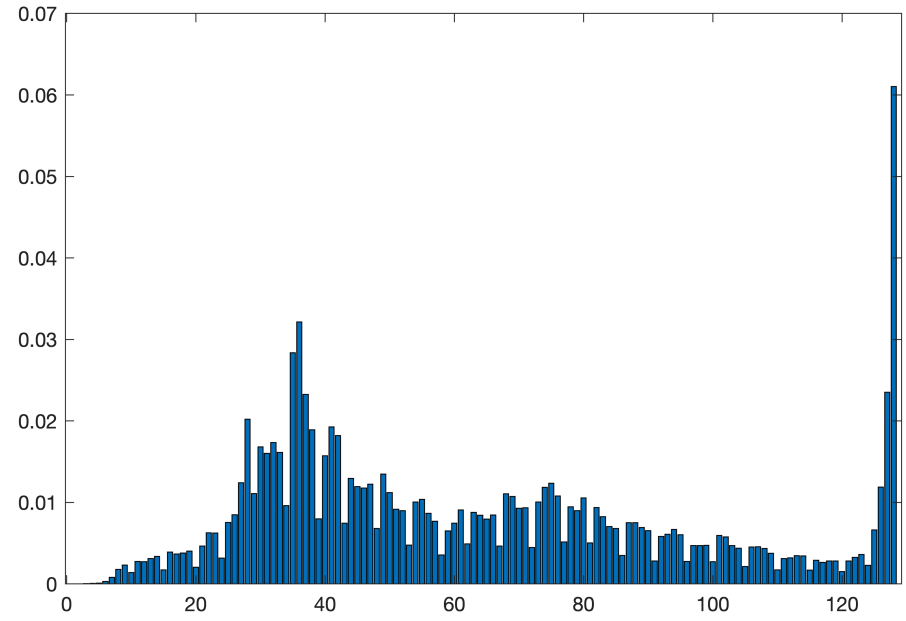
1. Conseguim l'histograma de color de cada una de les imatges (primer ho farem per canals RGB i després amb el Hue, per curiositat de funcionament).
2. Calculem la distancia euclideana o chi-square (farem les dues per comparar resultats, per a cada tipus d'histograma) comparant cada imatge amb la model.
3. Un cop tenim la distancia de cada una de les imatges amb la model en un vector Sn, normalitzarem els seus valors per a que estiguin en un rang de 0 a 100; sent 100 la imatge més semblant a la model.
4. Mostrar la gràfica de barres amb el títol corresponent de tipus d'histograma i tipus de distancia calculada per veure i comprar diferents resultats.

Comparació pels canals R,G i B

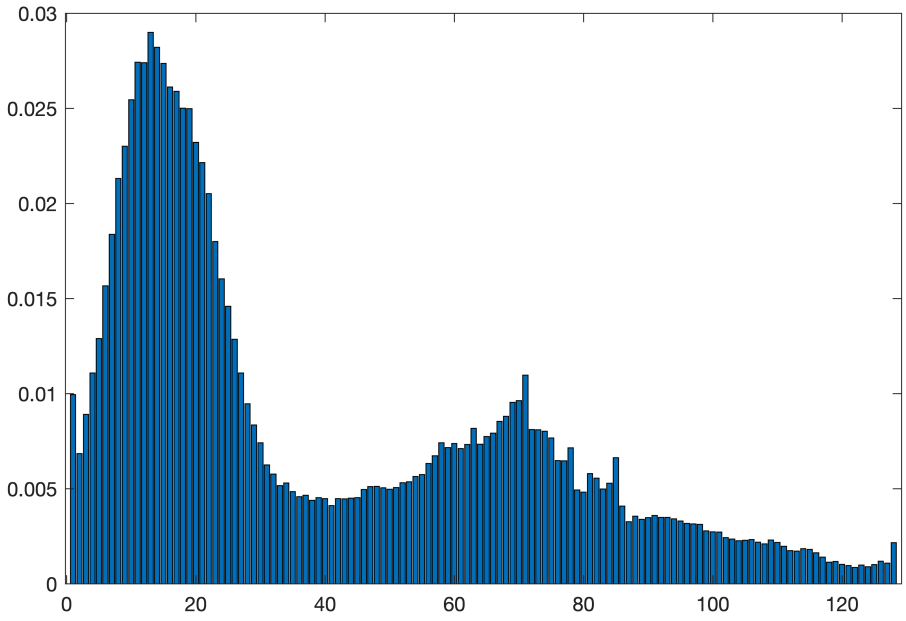
```
% calculem el histograma de cada imatge pels canals RGB
h1 = calcular_hist_rg(im1);
```



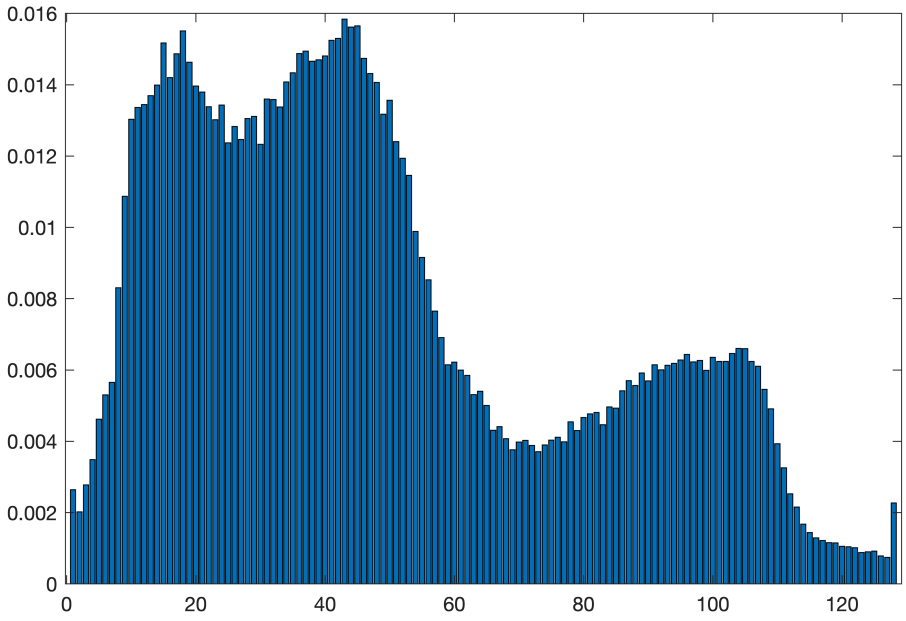

```
h2 = calcular_hist_rg(im2);
```



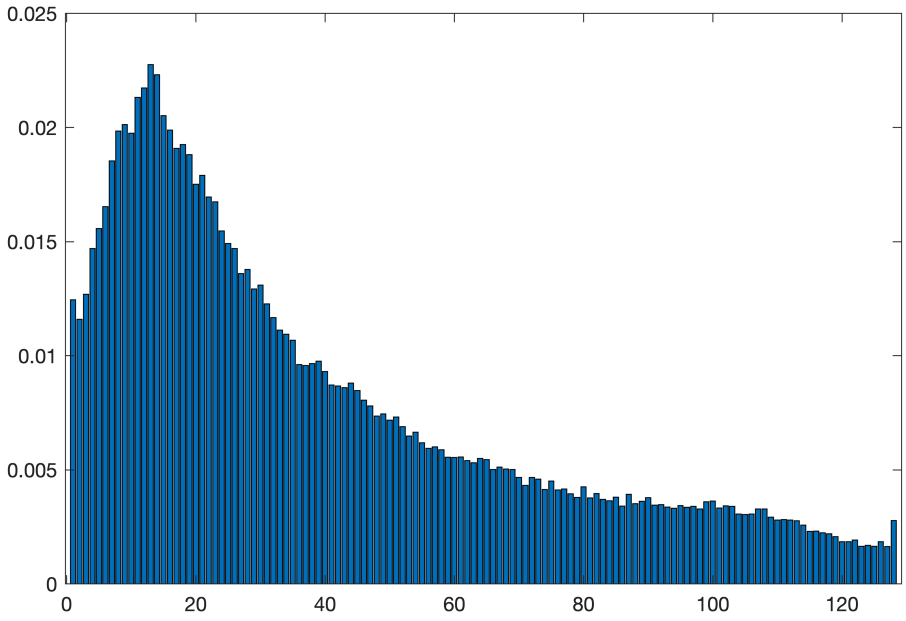

```
h3 = calcular_hist_rg(im3);
```



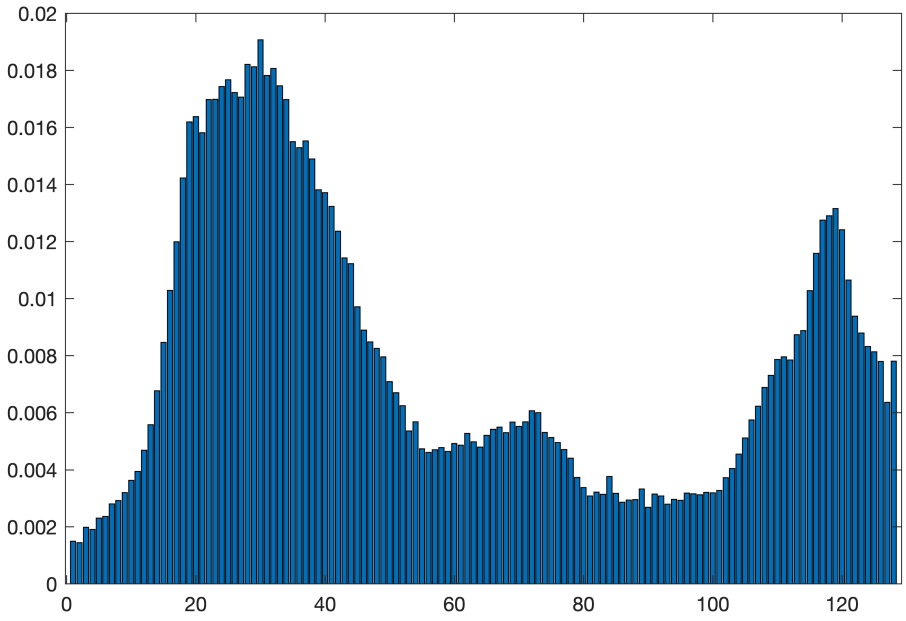

```
h4 = calcular_hist_rg(im4);
```




```
h5 = calcular_hist_rg(im5);
```

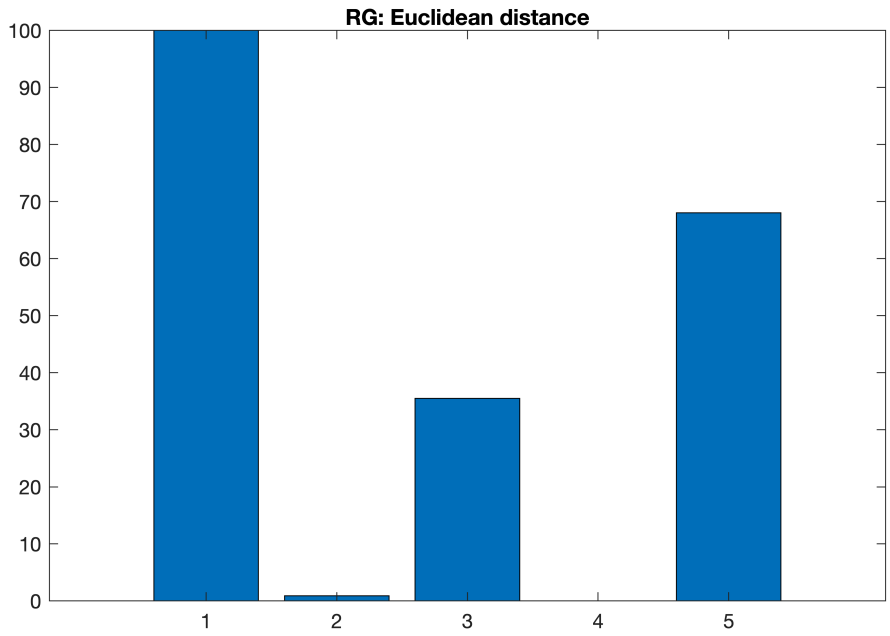



```
h6 = calcular_hist_rg(im6);
```

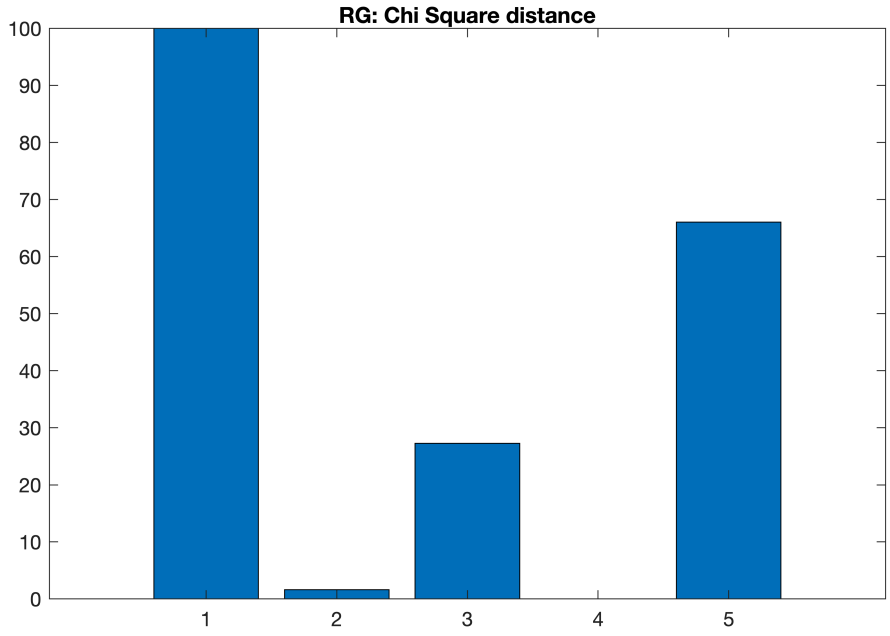



```
% calcular la distancia euclideana entre els histogrames amb la model
S1(1) = Euclidean_distance(h1,h2);
S1(2) = Euclidean_distance(h1,h3);
S1(3) = Euclidean_distance(h1,h4);
S1(4) = Euclidean_distance(h1,h5);
S1(5) = Euclidean_distance(h1,h6);
```

```
% normalitzar valors en un rang de 0 a 100
S1 = normalize_dist(S1);

bar(S1);
title('RG: Euclidean distance');
```

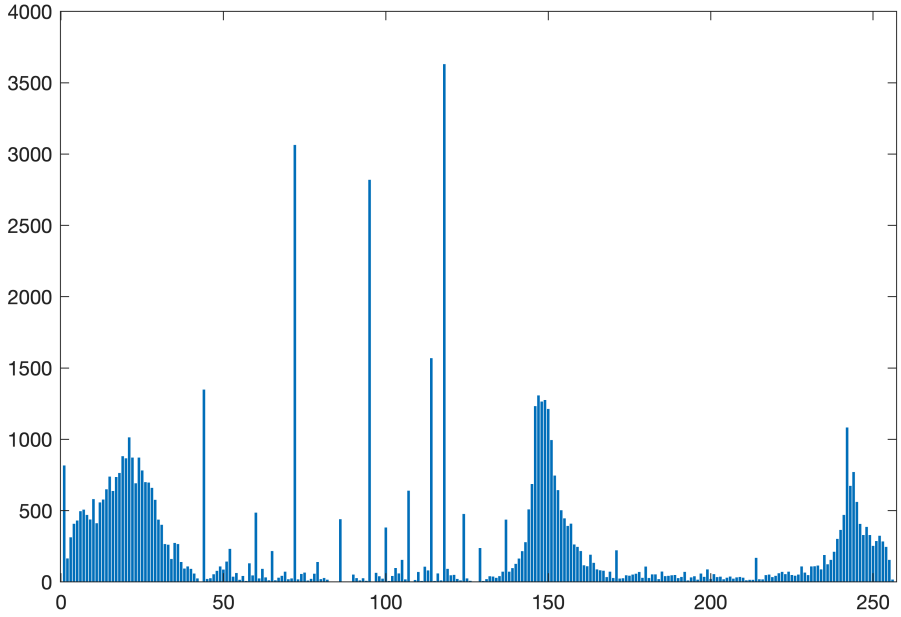



```
% calcular la distancia chi-square entre els histogrames amb la model
S2(1) = Chi_Square_distance(h1,h2);
S2(2) = Chi_Square_distance(h1,h3);
S2(3) = Chi_Square_distance(h1,h4);
S2(4) = Chi_Square_distance(h1,h5);
S2(5) = Chi_Square_distance(h1,h6);

% normalitzar valors en un rang de 0 a 100
S2 = normalize_dist(S2);

bar(S2);
title('RG: Chi Square distance');
```

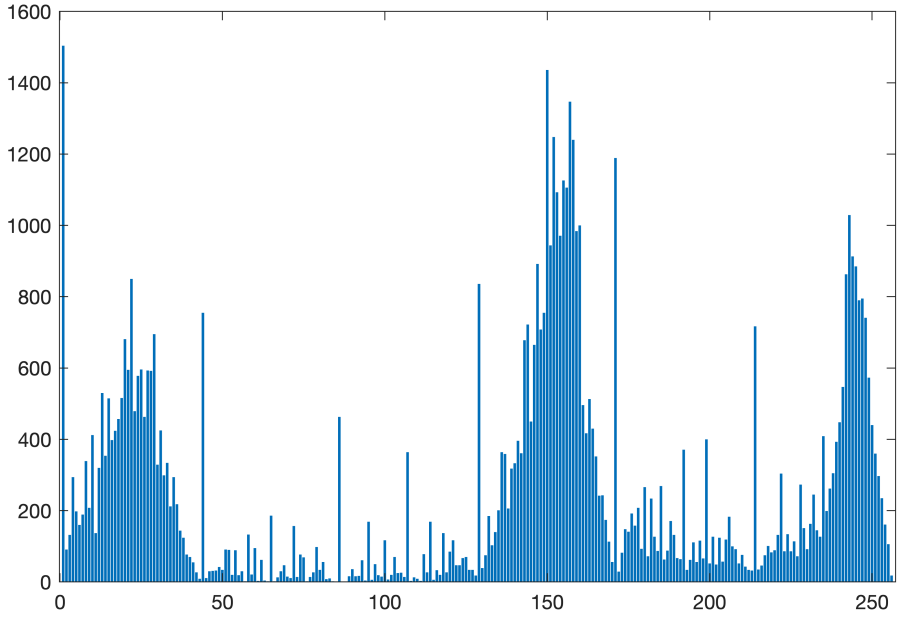


Comparació amb el hue

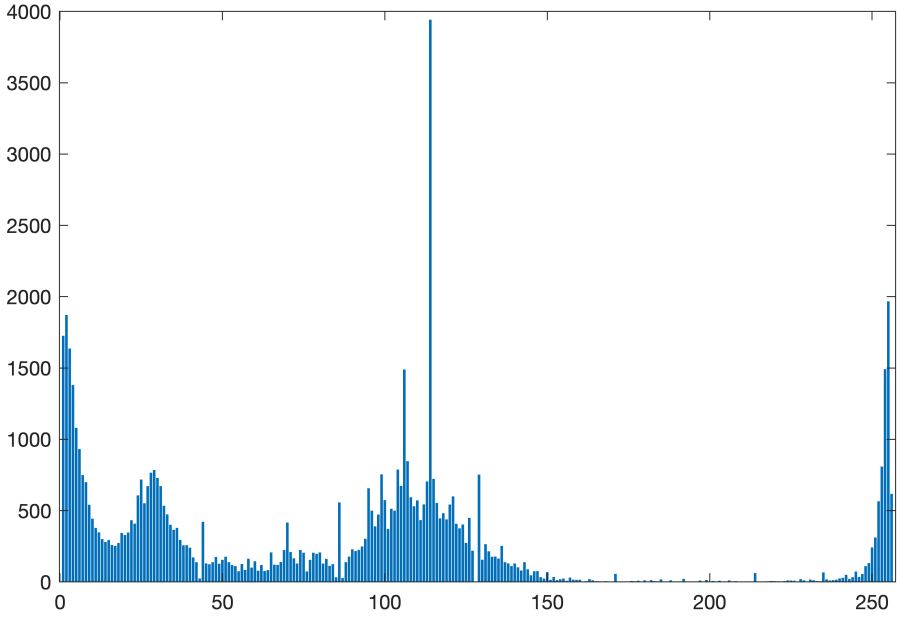
```
% calculem el histograma de cada imatge tenint en compte en hue
h1 = calcular_hist_hue(im1);
```



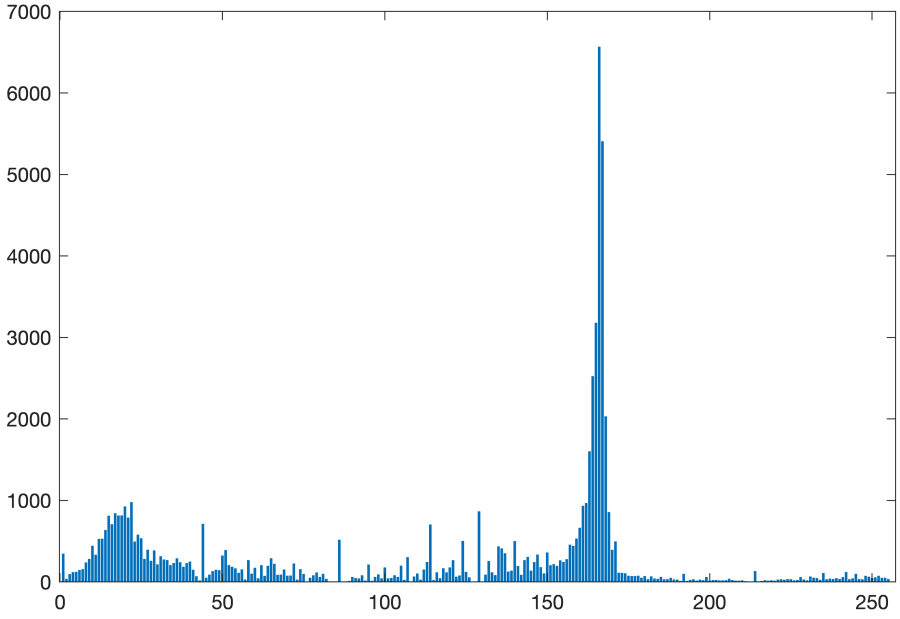

```
h2 = calcular_hist_hue(im2);
```



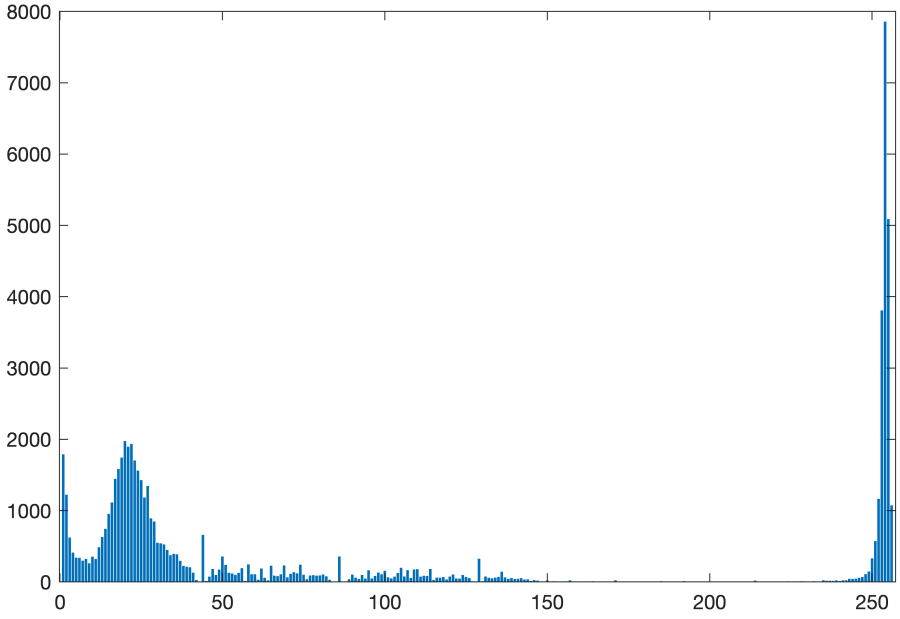

```
h3 = calcular_hist_hue(im3);
```



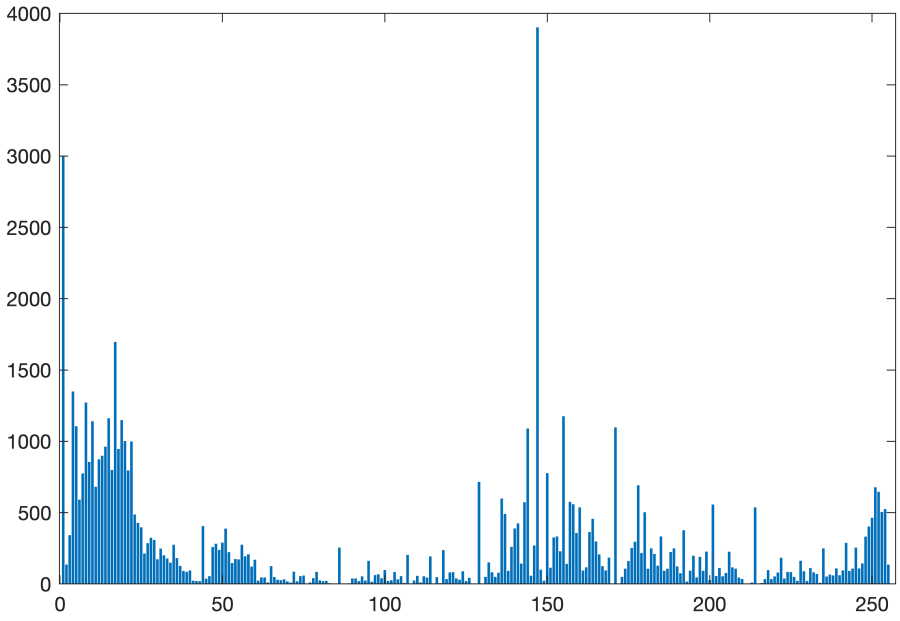

```
h4 = calcular_hist_hue(im4);
```



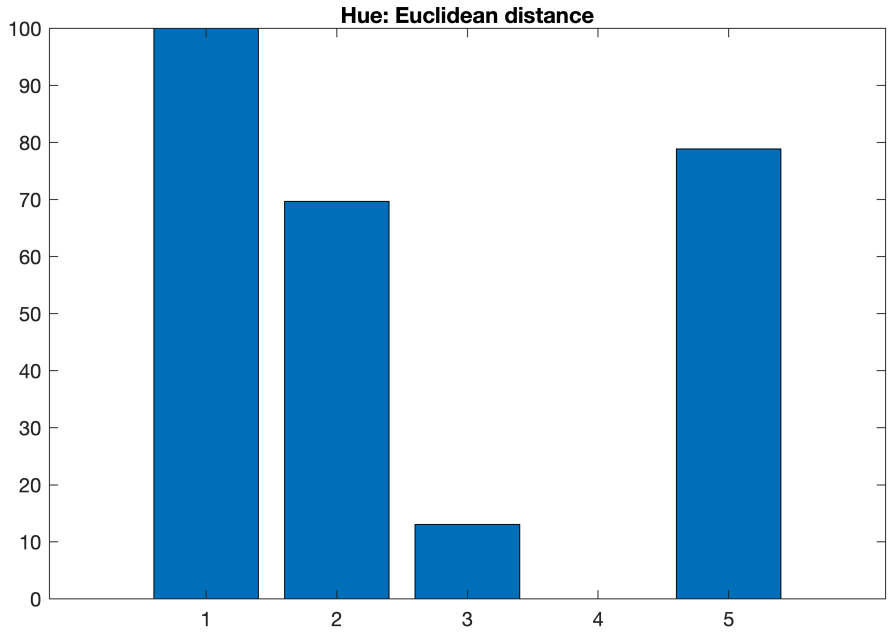

```
h5 = calcular_hist_hue(im5);
```



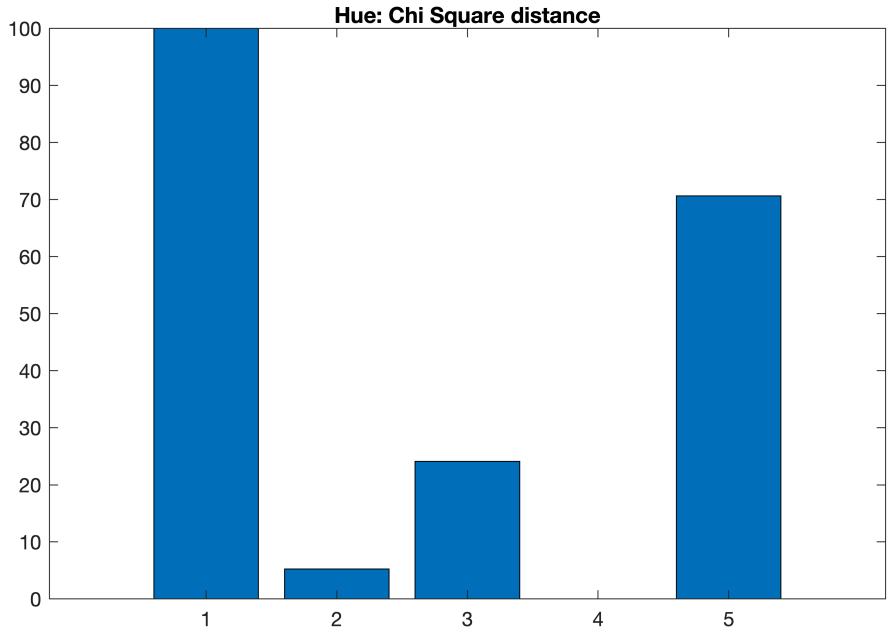

```
h6 = calcular_hist_hue(im6);
```




```
% calcular la distancia euclideana entre els histogrames amb la model
S3(1) = Euclidean_distance(h1,h2);
S3(2) = Euclidean_distance(h1,h3);
S3(3) = Euclidean_distance(h1,h4);
S3(4) = Euclidean_distance(h1,h5);
S3(5) = Euclidean_distance(h1,h6);

% normalitzem la distancia entre 0 i 100 per una millor representació i
% l'invertim degut al funcionament del hist de hue
S3 = normalize_dist(S3);
S3 = 100-S3;

bar(S3);
title('Hue: Euclidean distance');
```




```
% calcular la distancia chi-square entre els histogrames amb la model
S4(1) = Chi_Square_distance(h1,h2);
S4(2) = Chi_Square_distance(h1,h3);
S4(3) = Chi_Square_distance(h1,h4);
S4(4) = Chi_Square_distance(h1,h5);
S4(5) = Chi_Square_distance(h1,h6);

% normalitzem la distancia entre 0 i 100 per una millor representació i
% l'invertim degut al funcionament del hist de hue
S4 = normalize_dist(S4);
S4 = 100-S4;

bar(S4);
title('Hue: Chi Square distance');
```




```
function [h] = calcular_hist_rg(im)
    im = imresize(im, [256, 256]);
    % compararem les imatges pels canal R,G i B
    canal_rojo = im(:, :, 1);
    canal_verde = im(:, :, 2);
    canal_azul = im(:, :, 3);
    canales_rgb = cat(3, canal_rojo, canal_verde, canal_azul);
    numBins = 128;
    histogram = histcounts(canales_rgb, numBins);
    histograma_normalizado = histogram / sum(histogram);
    h = bar(histograma_normalizado).YData;
end

function [h] = calcular_hist_hue(im)
    im = imresize(im, [256, 256]);
    % compararem les imatges pel hue
    hsvImage = rgb2hsv(im);
    hueChannel = hsvImage(:,:,1);
    numBins = 256;
    histogram = imhist(hueChannel, numBins);
    h = bar(histogram).YData;
end

function [Sim] = Euclidean_distance(h1,h2)
    % distancia euclideana entre els dos histogrames
    Sim = sqrt(sum((h1 - h2).^2));
end

function [Sim] = Chi_Square_distance(h1,h2)
    % distancia chi-square entre els dos histogrames
    Sim = sum(((h1 - h2).^2) ./ (h1 + h2 + eps));
```

```
end

function [norm] = normalize_dist(S)
    % normalitzem els valors del vector per uns resultats representatius
    % (0-100)
    valor_min = min(S);
    valor_max = max(S);
    dif = S - valor_min;
    escalado = 100 / (valor_max - valor_min);
    norm = dif * escalado;
end
```