Tarea 3
Segmentación
INFORME

Integrante 1: Ana Poveda García Herrero.
Integrante 2: Cristina Taboada Mayo.

Objetivo de la parte obligatoria:

El primer objetivo de esta parte era elegir un objeto fácil de segmentar, en nuestro caso hemos elegido el balón rosa, ya que es el objeto que más resalta. Lo primero de todo, ha sido ir probando los diferentes espacios de color para la imagen, hasta llegar a uno que diese buenos resultados. Hemos elegido el espacio HSI usando una función de matlab *'convertionRgb2Hsi()'*, y tras visualizar cada componente nos hemos quedado con la componente 'H' ya que es la componente que más caracteriza el balón.*Figura 1.*

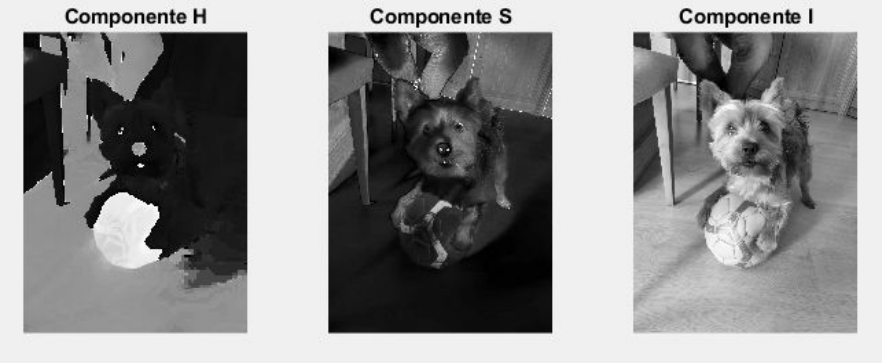

Figura 1. Componentes espacio de color HSI.

Hemos visualizado también la representación tridimensional de la componente 'H', y efectivamente los valores más altos coinciden a los pixeles del balón *(Anexo 1.1)*.*Figura 2.*

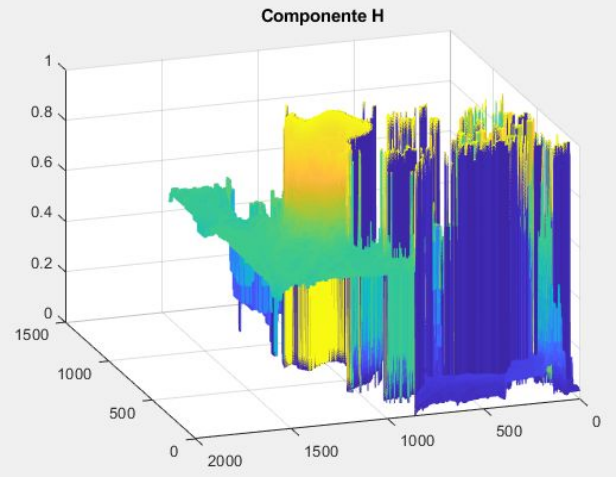

Figura 2. Representación tridimensional de la componente H

Para obtener mejores resultados en la futura segmentación, aplicamos a la componente H un filtro de media, para suavizar los distintos niveles de intensidad y crear un efecto de *'difuminado'* *(Anexo 1.2).*
De esta manera, hemos conseguido un histograma más sencillo.*Figura 3.*

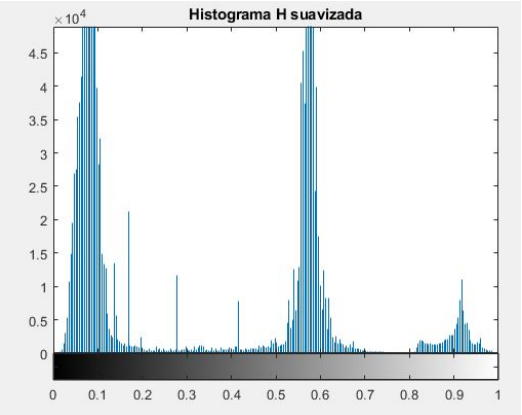

Figura 3. Histograma de la componente H suavizada

A continuación aplicamos segmentación binaria por umbralización, estableciendo un umbral de 0.8, quedando como primer plano valores por encima de ese umbral (el balón) y como fondo los valores por debajo.*Figura 4.*

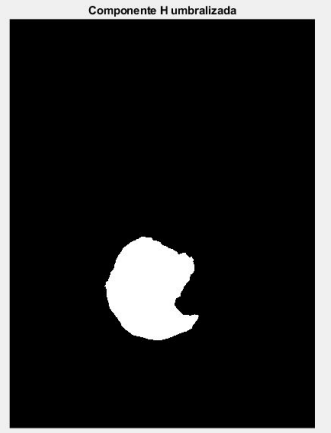

Figura 4. Componente H umbralizada.

Con esta segmentación, nos damos cuenta de que no es suficiente para extraer nuestro objeto de interés sin dejar algún pequeño píxel por la imagen, por lo tanto, hacemos una segunda segmentación y esta vez por tamaño, quedándonos con el objeto más grande.
Para esto, usamos las instrucciones de la práctica 4 identificando así nuestro objeto de mayor tamaño y creando nuestra nueva capa de segmentación solo con el objeto de interés *(Anexo 1.3).*

Tras obtener nuestra capa de segmentación deseada, la utilizamos como máscara multiplicandola pixel por pixel por la imagen original, obteniendo así solo nuestro objeto deseado *(Anexo 1.4)*. *Figura 5.*

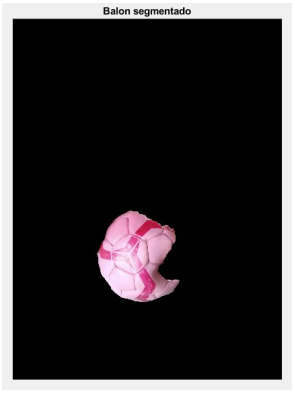

Figura 5. Objeto de interés.

Para mover el objeto segmentado a la esquina superior derecha, asignando al fondo un tono gris intermedio, se han seguido los siguientes pasos:

- Crear una imagen artificial, con el mismo tamaño de la imagen original, con un tono gris intermedio (utilizando un mapa de color). *(Anexo 1.5)*. *Figura 6.*

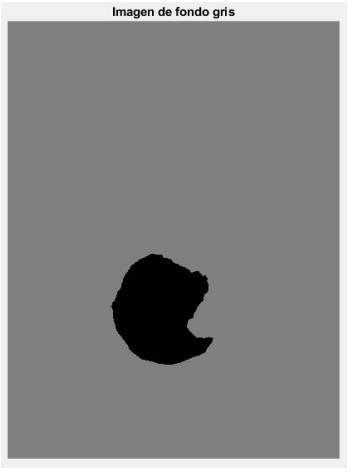

Figura 6. Imagen artificial.

- Sumar la imagen artificial, con la imagen resultado de la segmentación realizada anteriormente. *(Anexo 1.5)*. *Figura 7.*

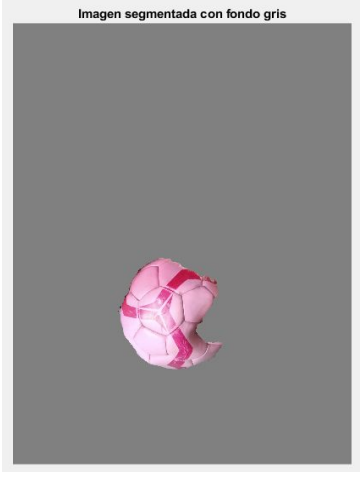

Figura 7. Imagen segmentada con fondo gris intermedio.

- Crear una bounding box que contenga el objeto a trasladar.*(Anexo 1.5)*. *Figura 8.*

Figura 8. Bounding Box.

- Trasladar la bounding box que contiene el objeto segmentado a la posición deseada (esquina superior derecha), y reasignar el color gris al fondo, obteniendo el resultado final.*(Anexo 1.5)*. *Figura 9.*

Imagen a. *Imagen b.*

Figura 9. (a) Imagen con el objeto segmentado trasladado con fondo por defecto. (b) Imagen con el objeto trasladado con fondo gris. Resultado final.

Objetivo de la parte creativa:

Cambiar nuestra imagen de espacio de color y aplicar una clasificación por el algoritmo k-medias.

Lo primero de todo, hemos preprocesado nuestra imagen, la hemos recortado quedándonos con la pelota ya que con la imagen entera los resultados no eran buenos.*(Anexo 1.6)Figura 10.*

Figura 10. Imagen Recortada.

A continuación, hemos cambiado de espacio de color, al espacio Lab y nos hemos quedado con las componentes 'a' y 'b' que son las que contienen información de color. Tras esto, las hemos redimensionado a un vector columna.*(Anexo 1.6)Figura 11.*

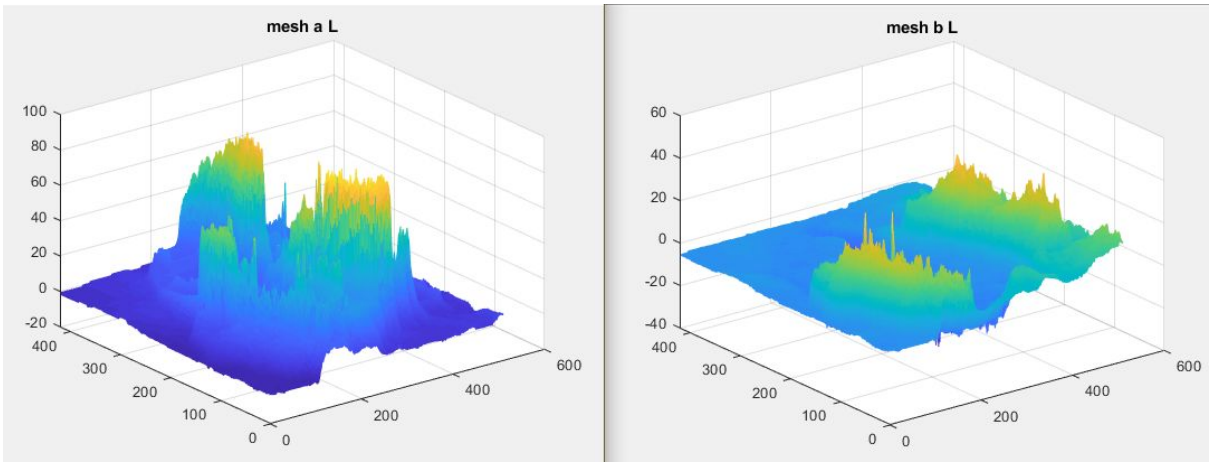

Figura 11. Mesh de las componentes a y b.

Después, hemos aplicado el algoritmo k-medias, considerando k = 4 grupos, distancia Euclídea como forma de medida y repitiendo el algoritmo con 10 inicializaciones, es decir, 10 posibles posiciones distintas de los centroides y matlab se queda con la mejor.

Tras esto, visualizamos los resultados.*(Anexo 1.6)Figura 12.*

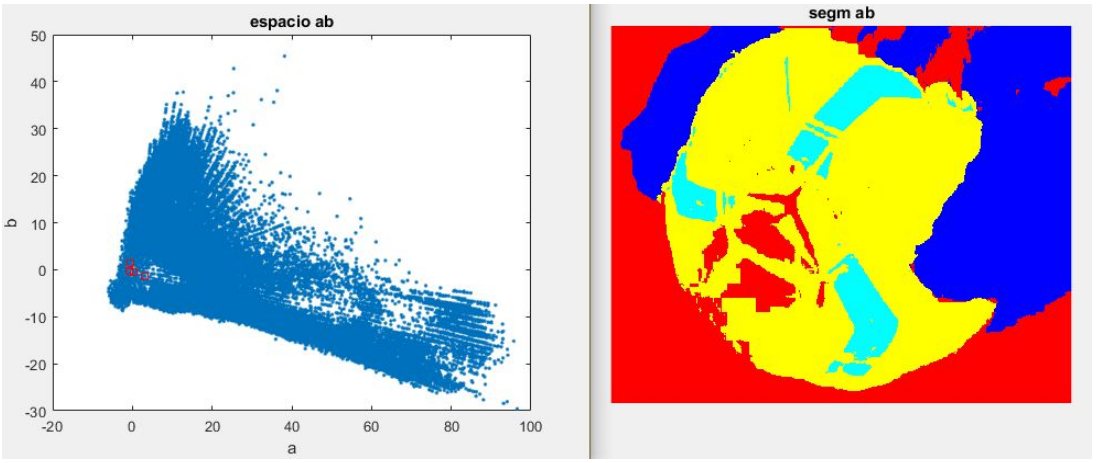

Figura 12. Scatter plot e imagen en falso color del resultado del agrupamiento del algoritmo k-medias.

Con los resultados obtenidos, nos damos cuenta de que la componente b tiene más influencia en determinar la posición de los centroides debido a que tiene mayor rango dinámico, entonces los resultados no son buenos.
Para evitar que ningún componente domine en el cálculo de distancias, normalizamos ambas componentes en una nueva estructura 'ab', es decir, las estandarizamos mediante la fórmula vista:

```
datos_norm = (datos-mean(datos))/std(datos);
```

Finalmente, volvemos a aplicar el algoritmo k-medias, obteniendo mejores resultados.*(Anexo 1.6)Figura 13.*

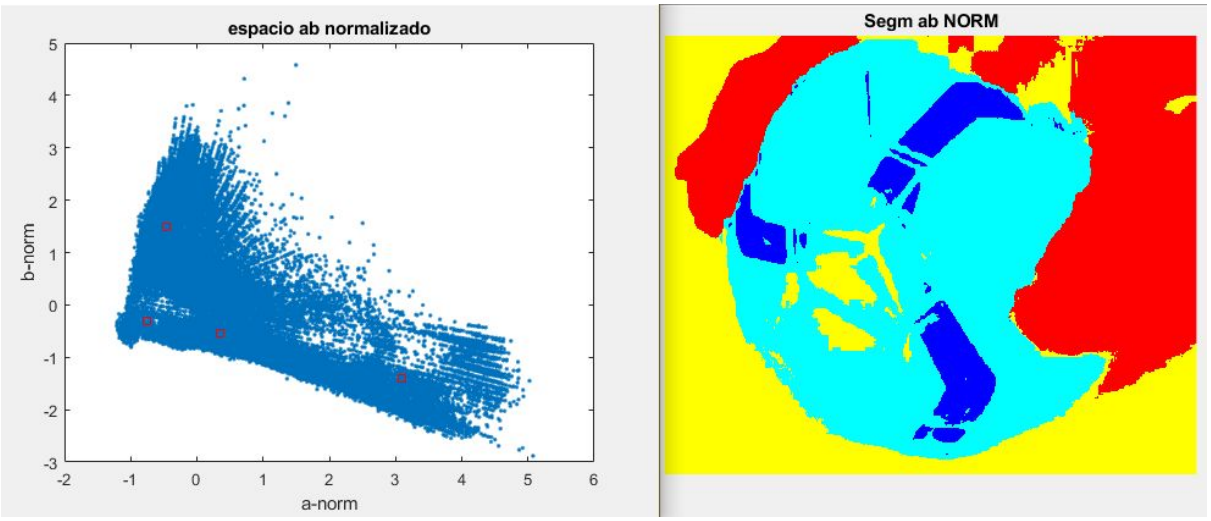

Figura 13. Scatter plot e imagen en falso color del resultado del agrupamiento del algoritmo k-medias normalizado.

Para finalizar, hemos llegado a la conclusión que este método para segmentar nuestra imagen no es correcto, ya que los grupos obtenidos por el algoritmo k-medias no son de objetos independientes, como por ejemplo el suelo y parte del balón, que quedan englobados en un mismo grupo.

ANEXO:

Anexo 1.1:

```
%% -------------------- PARTE OBLIGATORIA --------------------
clear all, close all, clc
% Cargamos la imagen con la que vamos a trabajar y la visualizamos.
Img = imread('G15.jpeg');
figure, imshow(Img), title('Imagen Original');

%% Pre-procesado de la imagen original para facilitar la segmentación.

% Cambiamos nuestra imagen del espacio RGB al espacio HSI
% Para poder hacer esto nos hemos descargado la función de matlab =>
% convertionRgb2Hsi.m, que adjuntamos junto con el script de la práctica.
I_HSI = convertionRgb2Hsi(Img);
figure, imshow(I_HSI), title('Imagen en el espacio HSI')

% Visualizamos cada componente.
H = I_HSI(:,:,1); % Componente del tono
S = I_HSI(:,:,2); % Componente de saturación.
I = I_HSI(:,:,3); % Componente de brillo.

figure, subplot(1,3,1), imshow(H), title('Componente H')
subplot(1,3,2), imshow(S), title('Componente S')
subplot(1,3,3), imshow(I), title('Componente I')

% Cuando visualizamos las componentes observamos que en la componente H los
% valores mas altos corresponden a los pixeles del balón.
% Visualizamos un mesh para ver cuales son esos valores.
figure, mesh(H), title('Componente H')
```

Anexo 1.2:

Anexo 1.3:

```
%% Proceso de segmentacion.

% Tenemos que hacer es umbralizar la imagen correspondiente
% a la componente H, que previamente hemos pasado por un filtro paso bajo
% para unificar los distintos niveles de la imagen.

% Primero, pasamos a la imagen de la componente H a blanco y negro usando
% segmentación binaria por umbralización.
H_U = im2bw(H_FPB,0.8);
figure, imshow(H_U), title('Componente H umbralizada')

% Creamos la capa de segmentación en blanco y negro y la visualizamos.
[Seg_H_U, Nobjetos] = bwlabel(H_U);
imtool(uint8(Seg_H_U)), title('Capa de etiquetas normal')
fprintf('El número de objetos es: %0i\n', Nobjetos);

% Visualizamos la capa de etiquetas en falso color. Blanco en matlab es la etiqueta 0 que
% veíamos en negro en los apuntes.
RGB_Segment = label2rgb(Seg_H_U);
imtool(RGB_Segment), title('Capa de etiquetas en falso color')

% Calculamos el tamaño de los objetos.
Props = regionprops(Seg_H_U, 'Area');
V_Area = [];
for ind_obj = 1:Nobjetos
    V_Area = [V_Area Props(ind_obj).Area];
end
figure,stem(V_Area)
xlabel('Número de objeto'), ylabel('Tamaño')

% Identificamos que la etiqueta asignada a la region de mayor tamaño (el
% balon) es la 4.
V_Interes = [4];

% Bucle que filtra de la imagen binaria I_U las regiones de no interés.
[n_filas, n_cols] = size(H_U); % nos dice el numero de filas y columnas que tiene la imagen.
```

Anexo 1.4:

```
% Usamos la imagen umbralizada como mascara y conseguimos extraer nuestro
% objeto de interés.
% Como la imagen que queremos segmenta es RGB tenemos que aplicar la
% mascara a cada una de sus componentes y volver a recomponer la imagen.

I_Balon = uint8(H_U).*Img; % .* multiplica px por px.
figure, imshow(I_Balon), title('Balon segmentado')
```

Anexo 1.5:

```
% Cambiamos el nivel del fondo a gris intermedio.
MC = [128 128 128; 1 1 1]; % Mapa de color.
I_Fondo = uint8(ind2rgb(H_U, MC));
figure, imshow(I_Fondo), title('Imagen de fondo gris')

% Sumamos la imagen del fondo que hemos creado con la imagen segmentada
Seg_Balon = I_Fondo + I_Balon;
figure, imshow(Seg_Balon), title('Imagen segmentada con fondo gris')

% Obtenemos la bounding box que contiene nuestro objeto de interes => el
% balon.
hold on
BB = rectangle('Position',[362, 840, 400, 450])
Balon = imcrop(Seg_Balon,[362, 840, 400, 450]);
figure, imshow(Balon), title('Balón contenido en la Bounding Box')
```

```
% Colocamos la bounding box con el objeto en la esquina superior derecha de
% nuestra imagen.
Imagen = uint8(ones(1600,1200));
Ibw = im2bw(Imagen);
Imapa = [1 1 1; 1 1 1];
Irgb = uint8(ind2rgb(Ibw,Imapa));
[Nrows, Ncols] = size(Balon(:,:,1));
for i=1:Nrows
    for j=1:Ncols
        Irgb(i,800+j,1) = Balon(i,j,1);
        Irgb(i,800+j,2) = Balon(i,j,2);
        Irgb(i,800+j,3) = Balon(i,j,3);
    end
end
```

```
% Volvemos a asignar al fondo el color gris intermedio.
I_gris = rgb2gray(Irgb);

[Nrows, Ncols] = size(Irgb(:,:,1));
for i=1:Nrows
    for j=1:Ncols
        if I_gris(i,j) == 1
            Irgb(i,j,:) = 128;
        end
    end
end

% Obtenemos el resultado final.
figure, imshow(Irgb), title('Nueva imagen RGB')
```

Anexo 1.6:

```
%% -------------------- PARTE CREATIVA --------------------
clear all, close all, clc

% Vamos a intentar hacer una segmentación del balon, como en la parte
% obligatoria, pero esta vez utilizando el algoritmo k-medias(técnica no
% supervisada), sobre una imagen en el espacio Lab y haremos la valoración
% de si es una buena o mala segmentación para nuestra imagen.

% Cargamos la imagen con la que vamos a trabajar y la visualizamos.
Img = imread('G15.jpeg');
figure, imshow(Img), title('Imagen Original');

% Para facilitar el proceso, recortamos la imagen, reduciendo el numero de
% niveles de intensidad en el histograma respecto al de la imagen original.
I_rec = Img(850:1270, 320:830, :);
figure, imshow(I_rec), title('Imagen recortada')

% Calculamos el tamaño de la imagen recortada.
[nrows, ncols, ndim] = size(I_rec);
```

```
% Cambiamos nuestra imagen del espacio RGB al espacio Lab.
[lab_imL, l_L, a_L, b_L] = rgb2lab(I_rec);
a_res = reshape(a_L, nrows*ncols, 1);
b_res = reshape(b_L, nrows*ncols, 1);

figure, mesh(a_L), title('mesh a_L')
figure, mesh(b_L), title('mesh b_L')

% Aplicamos el algoritmo k_medias considerando k = 4.
ngrupos = 4;
ab_res = [a_res, b_res];
[cluster_idx, cluster_center] = kmeans(ab_res, ngrupos, 'distance', 'sqEuclidean', 'Replicates', 10);

figure, plot(a_res, b_res, '.'), title('espacio ab')
xlabel('a'), ylabel('b')
hold on % Sirve para mantener la figura y sobreescribir encima.
plot(cluster_center(:,1), cluster_center(:,2), 'sr');

pixel_labels_ab = reshape(cluster_idx, nrows, ncols);
I_segm = label2rgb(pixel_labels_ab);
figure, imshow(I_segm), title('segm ab')
```

```
% normalización
ab_res = [a_res, b_res];
ndim = size(ab_res, 2);
ab_norm = ab_res;
for ind_dim = 1:ndim
    datos = ab_res(:,ind_dim);
    datos_norm = (datos-mean(datos))/std(datos);
    ab_norm(:,ind_dim) = datos_norm;
end

[cluster_idx_norm, cluster_center] = kmeans(ab_norm, ngrupos, 'distance', 'sqEuclidean', 'Replicates', 10);
pixel_labels_ab_norm = reshape(cluster_idx_norm, nrows, ncols);

I_segm = label2rgb(pixel_labels_ab_norm);
figure, imshow(I_segm), title('Segm ab NORM')

figure, plot(ab_norm(:,1), ab_norm(:,2), '.'), , title('espacio ab normalizado')
xlabel('a-norm'), ylabel('b-norm')
hold on
plot(cluster_center(:,1), cluster_center(:,2), 'sr');
```