

**Práctica 5**

**Segmentación de Imagen (II) Curso 2019 – 2020**



El objetivo de esta práctica es familiarizar al alumno con las técnicas de extracción de características a fin de abordar un problema de segmentación utilizando aprendizaje no supervisado, en concreto el algoritmo *k*-medias. Para ello se trabajará con la imagen en color ‘cormoran.jpg’, que acompaña al material de esta práctica.

Utilice la ayuda de MATLAB (help+ comando) para conocer el funcionamiento de los comandos utilizados en este guión. Tenga en cuenta que a lo largo de la práctica puede utilizar las instrucciones clear all y close all para evitar posibles interferencias con otras variables o ventanas. La instrucción “imtool close all” permite cerrar todas las ventanas generadas con la función imtool.

* **Explicación general:**

El objetivo de la práctica es segmentar la imagen del cormoran.

¿Cuantos elementos consideramos que hay si queremos segmentar?

3, que serían el mar, el pájaro y el poster. Así que vamos a intentar segmentarla haciendo uso de aprendizaje estadístico, con el algoritmo k-medias. Por lo que siendo 3 objetos, k = 3. Y vamos a utilizar distintos descriptores para extraer el espacio de características y generar el espacio de características a partir del cual aplicar el algoritmo k-medias.

1. Vamos a empezar pasando la imagen a grises y viendo si podemos segmentar la imagen con umbralización, por lo que haremos el histograma y decidiremos si se puede o no
2. Vamos a ver que con umbralización no se puede, se produce sobresegmentación.
3. Luego vamos a extraer características para diferenciar los tres elementos, una de ellas es el ‘color’, es decir, vamos a utilizar descriptores de color.
4. Primero vamos a trabajar con las componentes RGB de manera independiente, entonces vamos a tener un espacio de 3 dimensiones,
5. Y en ese espacio de 3D vamos a hacer que cada pixel corresponda a un punto en ese espacio de 3D, con la componente R, G , B.
6. Y sobre ese espacio, vamos a aplicar el algoritmo k-medias

* **Desarrollo del proceso:**

1. Vamos a empezar pasando la imagen a grises y viendo si podemos segmentar la imagen con umbralización, por lo que haremos el histograma y decidiremos si se puede o no
2. Vamos a ver que con umbralización no se puede, se produce sobresegmentación, es decir, en el histograma podemos diferenciar:

* Cielo: los pixeles entre 200-250
* Cormorán: son valores oscuros
* El poste y agua: son valores que están mezclados, es difícil.

1. Por lo que vamos a extraer la componentes RGB, y representar cada pixel en este espacio, como un punto a través de un ‘scatter plot’ de 3D, entonces vamos a ver una distribución de una determinada forma.
2. Y en ese espacio, se va a aplicar el algoritmo k-medias con k = 3, entonces cuando se aplique el algoritmo, nos van a aparecer los centroides, que son los cuadraditos rojos que estarán distribuidos , por lo que van a estar centrados en la nube de puntos.
3. Vamos a determinar luego los puntos más próximos de cada centroide, y de esta manera vamos a asignar una capa de etiquetas (que empiezan con el 1,2,3), y luego lo que vamos a hacer es generar una imagen en color a partir de la capa de etiquetas con el comando ‘label2rgb’ y así vamos a ver la imagen segmentada. Que en verdad no vamos a tener un buen resultado con esto porque hay sobresegmetación, porque 1 único objeto tiene distintas etiquetas.
4. lo siguiente que vamos a hacer es seleccionar mejor las características, hemos visto que son características de color, y lo que hemos hecho es aplicar el algoritmo k-medias sobre el espacio RGB.

El problema del espacio RGB en cuanto al color, es necesario para extraerle tener información de las 3 componentes, entonces es más difícil hacer la segmentación.

1. ¿Que haríamos para concentrar mejor el color? Cambiar de espacio de color, donde la croma, no esté repartida en 3D, por lo que el adecuado seria Lab, seria las componentes ‘ab’, y en el espacio HSI, sería la ‘HS’, pero por loscolores que tenemos el más adecuado es el Lab, por lo que ‘ab’, porque tenemos azules, tenemos amarillo anaranjado del poste, y el del cormoran.Por lo que la distribución de puntos en el espacio Lab va a quedar mejor, pero aun así habrá errores porque hay ciertos pixeles que estarán muy juntos.
2. Luego lo que vamos a hacer es aplicar el algoritmo k-medias en este espacio Lab, entonces los centroides cambiarán de ubicación, habrá uno que en el centro de la nube de puntos no estará (que esa región correspondía al cormoran)
3. por qué ocurre esto???? Fijándonos en el rango dinámico del eje a y b? tienen el mismo rango dinámico? No, por lo que habrá que hacer una normalización, en concreto una estandarización (ya que estaba ponderando el algoritmo de distinta manera la distancia de a y b). Por lo que antes de k-medias habrá que realizar esa estandarización, tipificar cada variable. Y esa estandarización se hace: x’ = (x-media/desviación típica). Entonces tendremos que hacer eso con todos los puntos, y generaremos una nueva componente ‘an’ y ‘bn’, normalizamos.

Entonces ahora la dos distribuciones, están en el 0, porque la media es 0 y las desv.típica es 1, que es la dispersión de valores entorno a la media.

1. tras esto, aplicamos el algoritmo k.medias y ahora sí que hemos rectificado el centroide y está centrado. Ahora hacemos la segmentación, y obtendremos algo mejor de lo que teníamos antes, aunque no está del todo bien, aunque hemos mejorado.
2. ¿Como lo podemos mejorar? Pues os vamos otra vez a la imagen original y lo que hacemos es mezclar nuevos descriptores, hasta ahora solo habíamos usado 2 descriptores de color, pero si nos damos cuenta la textura de cada uno de los objetos que queremos segmentar es distinta, entonces vamos a aplicar descriptores de textura.
3. Para ello, vamos a utilizar filtros locales, de tamaño 7x7, y serán 3 tipos de filtros, el de entropía, el de rango dinámico y de desviación típica. esto lo que va a hacer es generar, es decir, si tenemos la imagen de grises ( la textura no se mide en el color) pues vamos a coger ventanas de tamaño 7x7, y vamos a ir desplazando y por cada puntos donde ubiquemos la ventana vamos a calcular o la entropía de todos los niveles de intensidad que aparecen debajo, o el rango dinámico o la desviación típica, que son cada uno de los descriptores. y vamos a generar 3 imágenes, una asociada a la entropía, otra al rango dinámico y otra a la desviación típica. Y vamos a utilizarlas como descriptores para caracterizar la textura ¿ Si tuviéramos que elegir 2 de ellos? Lo que me interesa es que haya variedad: entropía y desviación típica.
4. Elegimos esas dos (las estandarizamos) y aplicamos el algoritmo k-medias, y vemos donde caen los centroides en cada una.
5. Hacemos la segmentación por k-medias y obtenemos algo feo, es decir, sobresegmetación, y esto es debido a que los descriptores de textura que hemos elegido no son adecuados para esta imagen, esto es la ‘selección de características o de descriptores’.
6. Entonces, parece que los dos que hemos elegido el ‘ab color y textura, nos quedamos con color. pero para mejorar la de color, lo que podemos hacer es quedarnos con una de estas últimas, como por ejemplo, cuál es la más homogénea del objeto que quiero segmentar, la de entropía. y vamos a representar un nuevo espacio de 3D. Vamos a tener la componente ‘a’ normalizada, la componente ‘b’ normalizada y la componente ‘ de entropía’.
7. Sobre este espacio aplicamos el algoritmo k-medias, y obtenemos los centroides.
8. Tras esto se hace la segmentación, y obtenemos un resultado parecido a antes pero un poco mejor, pero aun así tenemos puntos del poste dentro del cormoran, y eso es porque algunas zonas del poste tienen colores parecidos, por lo que se están confundiendo los pixeles, pero son pocos.
9. ¿Qué podemos hacer? Filtrar, hacemos un FPB sobre las componentes de color, sobre ‘a’ y ‘b que son las que más información tienen.No tendria sentido filtrar sobre la imagen de grises para sacar información de textura, porque estaría suavizandola. Entonces solo sobre color.
10. Aplicamos el algoritmo k.medias, la distribución nos va a salir más o menos parecida, pero cuando hacemos la segmentación, obtenemos el resultado bueno.

# *Análisis visual de la imagen*

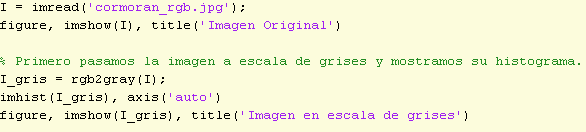
Lea en MATLAB la imagen ‘cormoran.jpg’ proporcionada como material adjunto en la práctica y almacénela en la variable I. Examine visualmente la imagen (representada en la Figura 1) y responda a las siguientes preguntas:

* + ¿qué objetos considera claramente discriminativos en la imagen?
* el cormorán
* el cielo
* y el palo
  + a simple vista, ¿qué características considera discriminativas?
* el color

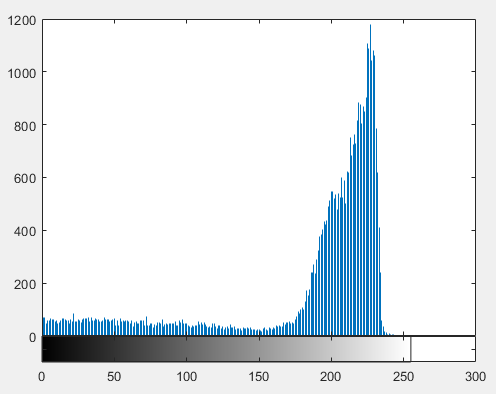
Figura 1. Imagen a segmentar en esta práctica.

Convierta la imagen a escala de grises (variable I\_gris) y analice visualmente su histograma. ¿Cree que podría realizar la segmentación mediante umbralización múltiple? Justifique su respuesta.

Usando este código, como hemos hecho anteriormente en otras prácticas, pasamos la imagen a escala de grises y obtenemos su histograma



Obteniendo un resultado tal que así:



En el histograma podemos diferenciar:

* Cielo: los pixeles entre 200-250
* Cormorán: son valores oscuros
* El poste y agua: son valores que están mezclados, es difícil.

Tras esto realizamos la segmentación por umbralización, con el comando ‘im2bw':



Obteniendo un resultado así:



# *Características RGB y algoritmo k-medias*

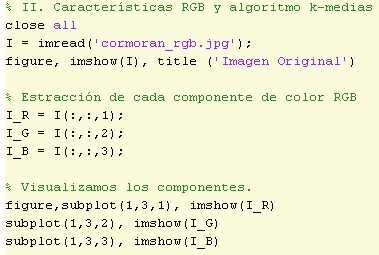
Inicialmente abordaremos el problema de segmentación considerando como características los niveles de intensidad de las componentes R, G y B de la imagen original I. Para ello:

# Y aquí nos damos cuenta de que no podemos umbralizar por grises, porque la información está en el color. Entonces tendremos que ir a la caracterización a través de descriptores de color.

Primero obtenemos cada componente del espacio de color, visualizandolas, que son imágenes en grises.

* + Extraiga cada componente de color de la imagen I.

Aquí extraemos las componentes de color, para ello accedo a todas las filas, a todas las columnas y de la tercera dimensión de RGB, (1: R, 2: G, 3: B):



Obteniendo estos resultados:



# 

Luego hacemos la representación en 3D

* + Convierta cada componente en un vector columna utilizando el comando reshape. Así, por ejemplo, para almacenar como vector columna los niveles de intensidad de los píxeles de la componente R de I, utilizaremos la siguiente secuencia de instrucciones

Para hacer representación *‘plot’* en 3D, hay que hacer otra cosa:

I\_R = I(:,:,1);

[nrows, ncols] = size(I\_R);

I\_R\_res = reshape(I\_R,nrows\*ncols,1);

Cuando nosotros extraemos cada componente RGB, lo que observamos en cada imagen de cada componente son imágenes Bidimensionales, es decir, 2D. Pero yo quiero llegar a algo que sea 3D, cogiendo cada píxel de cada imagen de cada componente (haciendo referencia a la misma posición espacial), y esto corresponderá a un punto en el nuevo espacio 3D.

Para hacer esto de manera automática, en MATLAB es muy sencillo.

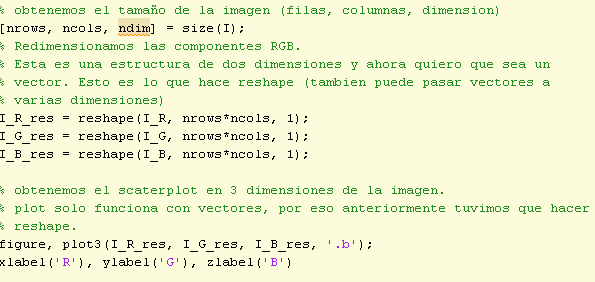
Primero o que se hace es pasar de las estructuras de 2D a vectores, un vector que tenga solo una columna o una fila, aquí lo hacemos con vector columna.

Entonces lo que hacemos es un ***‘reshape()’***, quiere decir que le doy otra forma a la estructura de datos, en lugar de ser 2D, me creo un vector columna, que va a tener ordenados (3 vectores en vertical, cada uno de RGB) los niveles de intensidad de cada componente, lo primero que hago es obtener el número de píxeles que tengo, que es el número de filas x columnas, y le digo que quiero redimensionar esas estructura de 2D cada una de ellas a un vector de nfilasxncolumnas, es decir, de nfilasx1columna (vector columna. Y eso lo hago para las 3 componentes, 3 vectores con las mismas dimensiones.

En cada fila estoy representado un pixel de los que hay en la imagen, por lo que las tres componentes de un pixel van a estar en el mismo sitio en el nuevo espacio.

Lo que puedo hacer para representar esto es el comando ***‘plot3()’***, que quiere decir, es que hace un plot pero de 3D, y le pasamos el modo de una una columna, la componente R, G, B, pero con estructuras que son columnas.

Hacemos esas representaciones gráficas y obtenemos el *‘scatter plot*’ en 3D de las componentes RGB de la imagen.



***ndim*** es la tercera variable, es la imagen rgb por lo que tiene que ser 3, porque en la 3 dimensión tengo 3 capas, es decir filas, columnas y rgb.

Número de filas: 1ºDimension

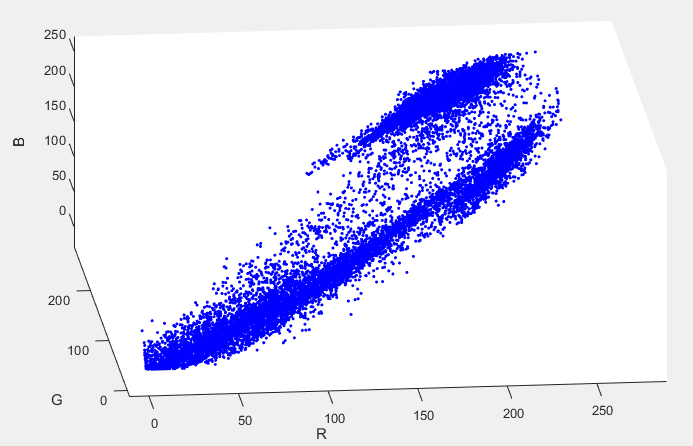
Número de columnas: 2ºDimension

Número de componentes: 3ºDimension, por lo que es 3 (rgb).

El ***‘reshape()’*** te transforma cosas, en este caso nos interesa que lo pase a vector, ya que el plot3 me exige que sean vectores. En este caso le estoy diciendo que es una estructura de 2 dimensiones, y quiero que sea un vector de todos esos elementos como filas(nrows\*ncols) y 1 columna. Todo en función del redimensionado que yo quiera hacer.

* + Represente el *scatter plot* de los datos utilizando la función plot3(haga uso del comando help para conocer su sintaxis). ¿Observa grupos de puntos claramente diferenciados?

Obteniendo este resultado:



Son muy parecidas las 3, porque el color está repartido en las 3 componentes, y a no ser que sean muy diferentes los colores, no hay una longitud de onda muy dominante, entonces son parecidas.

Y si rotamos el plot, nos damos cuenta,de que los puntos se encuentran distribuidos en el cubo en una diagonal, es decir que tengo más o menos los mismos valores de la R la G y la B.

Sobre el espacio de tres dimensiones representado en la figura anterior aplicaremos el algoritmo de agrupamiento *k*-medias con *k*=3. Utilizaremos para ello la función kmeans del siguiente modo

ngrupos = 3;

rgb\_res = double([I\_R\_res I\_G\_res I\_B\_res]); [cluster\_idx cluster\_center] =

kmeans(rgb\_res,ngrupos,'distance','sqEuclidean','Replicates',10);

Las **entradas** a la función kmeans son: (1) conjunto de ejemplos/observaciones (matriz rgb\_res, donde cada ejemplo es una fila y cada columna representa una característica), (2) medida de similitud (cuadrado de la distancia Euclídea en nuestro caso, indicado a través del parámetro

'distance'), y (3) número de inicializaciones (utilice el valor 10 en esta práctica, indicado con el parámetro 'Replicates'). El algoritmo kmeans devuelve la posición de los centroides (variable cluster\_center) y una etiqueta identificativa del *cluster* al que pertenece cada punto de entrada (variable cluster\_idx). Puesto que el algoritmo se realiza 10 veces con distintas inicializaciones, la salida proporcionada corresponde a la realización para la que se obtiene la mínima suma de distancias intra- cluster.

**Nota importante**: la matriz de ejemplos/observaciones debe ser de tipo double; cada fila de la matriz es una observación (píxel de la imagen, en este caso) y cada columna es una característica.

Las **salidas** de la función son: (1) identificador del *cluster* al que pertenece cada punto (variable cluster\_idx), y (2) centroide de cada *cluster* (variable cluster\_center).

Para observar el resultado del algoritmo *k*-medias, puede representar sobre el *scatter plot*

anterior los centroides resultantes del siguiente modo:

plot3(cluster\_center(:,1), cluster\_center(:,2), cluster\_center(:,3),'sr');

El resultado de la segmentación se puede observar en el espacio imagen generando una imagen en falso color a partir del identificador de *cluster* asociado a cada observación (píxel). Para ello, debe transformar el vector con la identificación del cluster al que pertenece cada píxel en una matriz de las mismas dimensiones que la imagen original. Puede hacer uso del siguiente código.

pixel\_labels\_rgb = reshape(cluster\_idx,nrows,ncols); I\_segm = label2rgb(pixel\_labels\_rgb);

figure, imshow(I\_segm)

Ahora lo siguiente que vamos a hacer es aplicar el algoritmo k-medias, con k = 3.

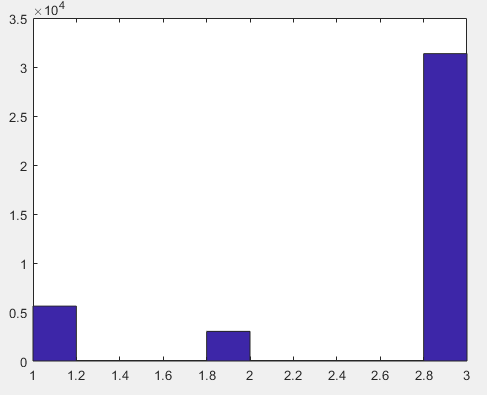
* Yo me defino una variable ‘ngrupos’ que es el valor de ‘k’
* Y ahora vamos a componer mi estructura de datos en 3D.
* Creo una nueva estructura de datos ***‘rgb\_res’,*** es decir, la componente R, G y B del reshape anterior(son uint8), las voy a agrupar en una matriz (3D), y paso todos los datos a tipo ‘double’ para poder luego encontrar valores intermedios.
* Y sobre esa matriz, aplicamos el algoritmo k-medias con el comando ***‘kmeans()’,*** que nos requiere como parámetros de entrada:

1. Variable de datos/matriz sobre la que voy a aplicar el algoritmo (rgb\_res), en esta función las dimensiones tienen que ser siempre las columnas. Y las instancias son las fila, que son cada uno de los pixeles
2. Valor de k (ngrupos)
3. El tipo de medida que voy a utilizar, que es distancia
4. El tipo de distancia: El cuadrado de la distancia Euclídea
5. ‘Replicates’: Número de veces que se repite el algoritmo, es decir, cuantas inicializaciones distintas vamos a tener
6. Número de inicializaciones: 10, por lo que voy a tener 10 posicionalidades distintas de los centroides, y matlab se queda con la mejor.

* Los parámetros de salida que os va a devolver el algoritmo son:

1. ***‘cluster\_center’,*** son los centroides de la mejor ejecución. En este caso, va a ser una matriz que va a tener los centroides (cuadraditos rojos). ¿Qué dimensión tendrá ‘cluster\_center’? Mi espacio de representación es de 3 dimensiones, por lo tanto CADA centro tiene que tener 3 dimensiones, y teniendo 3 centros, pues ‘cluster\_center’ será una matriz de 3x3. Cada fila, un centro; cada columna la componente R, G, B de ese centro.
2. **‘cluster\_idx’**, es un identificador que me dice cada instancia de ese vector (rgb\_rest) a que cluster pertenece. Los cluster eran 3, entonces voy a tener 3 valores (1,2,3). Por lo que ‘cluster\_idx’ va a ser un vector de tantas filas como pixeles tenga y 1 única columna (1: grupo 1, 2:grupo 2, 3: grupo3) que es la etiqueta que le estoy asociando a cada pixel.

* Vamos a tener como resultado los centroides en ‘cluster\_center’.
* Y en ‘cluster\_idx’ tenemos una matriz de 40000x1, que 40000 son todos los píxeles que tengo en 1 sola columna:



Da lo mismo como te salga el histograma las asociaciones de las etiquetas, k-medias asigna las etiquetas como quiere. La etiqueta 3 en nuestro caso, es la que tiene mayor número de píxeles, por lo que corresponderia al mar.

* Lo siguiente que hacemos es pasar la capa de etiquetas, bueno ‘capa’, es un vector, que lo tengo que pasar a la capa, es decir quiero ver ese vector como una imagen.
* Entonces vuelvo a hacer el ***‘reshape()’*** del vector de etiquetas (3D), el *‘cluster\_idx’* para estructurarlo en una matriz bidimensional (nfilasxncolumnas), entonces ya tendríamos nuestra capa de etiquetas, guardándolo así en la variable ***‘pixel\_labels\_rgb’***
* Después, lo que hacemos es visualizar el espacio RGB con la función ‘plot3()’ pasando como parámetros las componentes RGB pasadas por el ‘reshape’ y poner en cada eje una componente con el comando ***‘xlabel()’, ‘ylabel()’, ‘zlabel()’***, pintando así los pixeles con ***‘.b’*** que son puntos azules,y tras esto, pintar los centroides.
* Al pintar los centros lo que estamos haciendo es:

1. Pintar todas las filas componente 1(como si fuera ya x del vector), todas las filas componente 2(como si fuera la y del vector) y todas las filas componente 3(como si fuera la z del vector).
2. Como los quiero representar: ‘sr’ eso quiere decir cuadrados rojos (red square).
3. ***‘MarkerSize’:*** cómo de grande quiero representar el símbolo que está asociado a los centroides.
4. ***‘20’***, al haber tantos puntos, elegimos un cuadrado grande.
5. ***‘MarkerEdge’***: color del borde de ese marcador
6. ***‘r’*** : borde rojo.

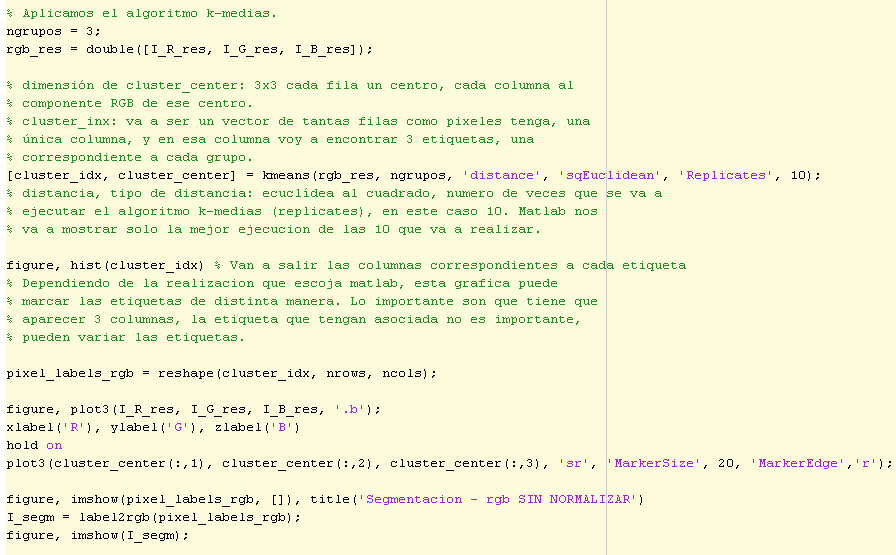
* Tras esto vamos a ver como es la capa de segmentación, porque lo que hemos visto es en el espacio de los descriptores. Entonces la capa de segmentación es volver a poner el vector ***‘pixel\_labels\_rgb’,*** en una estructura de 2D y visualizarlo con colores.
* Para ello, primero visualizamos solo la capa de etiquetas con un 1,2, 3, como si fueran grises, es decir visualizamos  ***‘pixel\_labels\_rgb’:***



* Luego, lo que hacemos es pasar la capa de etiquetas ***‘pixel\_labels\_rgb***’ a colores aleatorios con el comando ***‘label2rgb()’***, y la visualizamos:



* El código empleado es:



¿Considera que el resultado de la segmentación es el deseado?, ¿es posible afirmar que se produce sobresegmentación? Justifique razonadamente sus respuestas.

No es un buen resultado con esto porque hay sobresegmetación, porque 1 único objeto tiene distintas etiquetas.

El problema del espacio RGB en cuanto al color, es necesario para extraerle tener información de las 3 componentes, entonces es más difícil hacer la segmentación.

# *Características cromáticas ab*

Como sabe, en el espacio RGB la información cromática está distribuida en las tres componentes (R, G y B). En esta sección realizaremos una transformación del espacio de representación para separar las componentes cromáticas y acromáticas de la imagen, de modo que aplicaremos el algoritmo *k-*medias únicamente sobre el espacio de componentes cromáticas. La transformación considerada es la transformación Lab, transformación que implementa la función rgb2lab proporcionada como material adjunto a la práctica.

Realice la transformación de la imagen original (espacio RGB) al espacio Lab y extraiga únicamente las componentes cromáticas (componentes ab). Represente el *scatter plot* correspondiente e intente identificar visualmente a qué color corresponden las nubes de puntos.

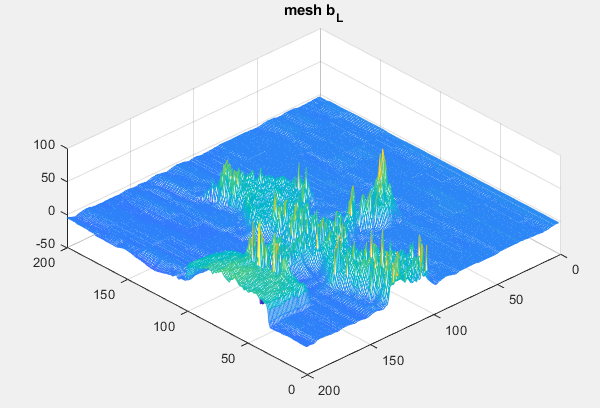
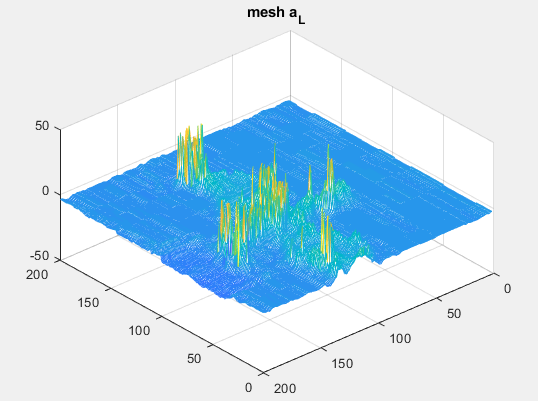
[lab\_imL, l\_L, a\_L, b\_L] = rgb2lab(I); a\_res = reshape(a\_L,nrows\*ncols,1); b\_res = reshape(b\_L,nrows\*ncols,1);

figure, plot(a\_res, b\_res,'.') xlabel('a'), ylabel('b')

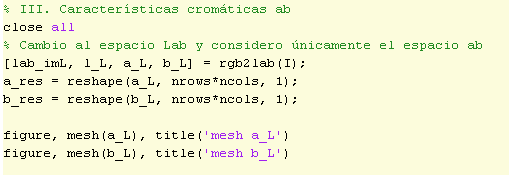
* Ahora nos vamos del espacio RGB al espacio ‘ab’, entonces lo siguiente que tenemos que hacer es un cambio en el espacio de representación, y la instrucción que me permite esto es ***‘rgb2lab()’*** y le tenemos que pasar la imagen que es RGB para que nos la cambie a ‘ab’.
* Ese comando lo que nos devuelve son 4 variables:

1. Es una estructura de datos que tiene TODO, la componente ‘L’, la componente ‘a’ y la componente ‘b’, como si fuera una matriz.
2. Componente independiente L.
3. Componente independiente a.
4. Componente independiente b.

* Y luego, como nos queremos quedar solo con las componentes ‘ab’, pues lo que hacemos es redimensionar cada una, para ponerlas cada una como un vector columnas. Esto es debido a que el comando ***‘rgb2lab()’*** me sigue dando una matriz del mismo tamaño que ‘I’, es decir, una señal bidimensional.
* Visualizamos las dos componentes a través de un ***‘mesh()’*** a ver qué pinta tienen:



* En los ejes estoy representando, las coordenadas en sí, y en el eje z el valor de esa componente en esa posición del espacio. Y observamos que nos aparece el pájaro. los valores de a y b pueden ser negativos también, por eso al hacer el *‘mesh()’* no hizo falta hacer la conversión a tipo doubles por que ya lo es.
* El código usado es así:

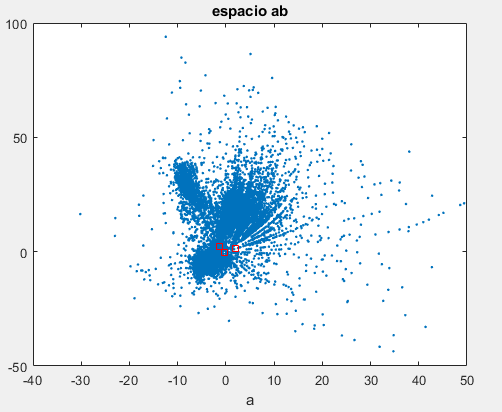


Aplique ahora el algoritmo *k*-medias sobre el nuevo espacio de características y represente la capa de etiquetas de la imagen segmentada. Comente las diferencias con la segmentación obtenida en el Apartado II.

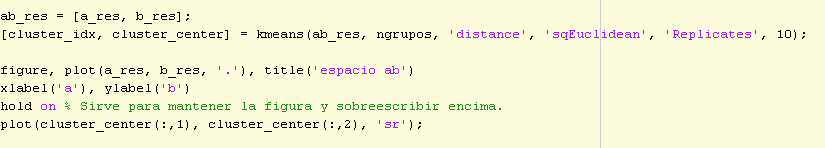
* Ahora siguiente que vamos a hacer es generar la matriz de datos para hacer el algoritmo k-medias.
* Osea, hemos dicho que no me quiero quedar con la componente ‘L’, porque no tiene croma, por eso solo me quedo con la componente ‘a’ y la componente ‘b’.
* Por lo que creo una estructura de datos, que sea solo del reshape de la componente a y b, y esa va a ser mi matriz llamada ***‘ab\_res’.***
* Antes tenia 3 componentes, mi espacio de características tenía 3 dimensiones. Ahora tengo 2 componentes (ab) por lo que mi espacio de características va a tener 2 dimensiones.
* Así que ahora puedo representar en un plano ya no necesito un volumen para hacer esa representación, por lo que ya no necesito ***‘plot3()’***, porque estoes para 3D. Entonces ahora uso ‘plot()’.
* Aplicamos el algoritmo k-medias como anteriormente, metiendo como estructura de datos la matriz  ***‘ab\_res’*** y las demás variables las que hemos visto antes.
* Las variables de salida como antes el vector de etiquetas ***‘cluster\_idx’*** (1,2,3)y la matriz de centroides ‘***cluster\_center’.*** ¿Qué dimensión va a tener ahora la matriz ***‘cluster\_center’***? filas: 3 centroides y en columnas: dimensiones (componentes) 2, ab.

(3x2).

* Luego lo pintamos todo, pintamos los puntos, tras esto usamos el comando ‘hold on’ que sirve para mantener la figura y sobreescribir encima, y luego pintamos los centroides:



* El código usado es:



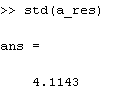
Represente la posición de los centroides en el *scatter plot* correspondiente y responda a las siguientes preguntas:

* ¿cuál es la desviación típica (comando std de MATLAB) asociada a la componente a

obtenida a partir de I?



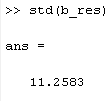
El resultado es este:



* ¿cuál es la desviación típica asociada a la componente b obtenida a partir de I?



El resultado es:

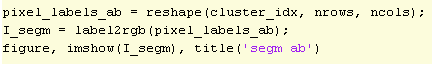


* ¿cree que alguna de las dos componentes tiene más influencia al determinar la posición de los centroides con el algoritmo *k*-medias?, ¿por qué?

Si, la que tengo mayor rango dinámico, haciendo así que la evaluación no sea muy correcta.

Es decir, ‘b’ tiene un rango dinámico de [-50,100 ], mientras que ‘a’ tiene de [-40,50].

Ahora para ver la capa de segmentación, usamos los mismos comandos que antes:



Obteniendo un resultado no muy bueno:



Para que no haya una componente que domine en el cálculo de distancias del algoritmo *k*- medias únicamente porque su rango dinámico es mayor, se propone normalizar cada componente del espacio ab para que tenga media nula y desviación típica uno. La matriz de ejemplos con características normalizadas se asignará a la variable ab\_norm (cada fila es un ejemplo y cada columna es una característica). Explique cómo se realiza la normalización si la secuencia de instrucciones es la siguiente:

ab\_res = [a\_res b\_res]; ndim = size(ab\_res,2); ab\_norm = ab\_res;

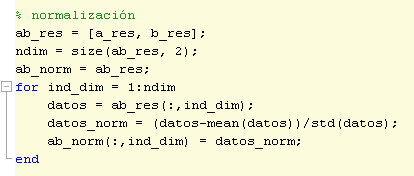
for ind\_dim=1:ndim

datos = ab\_res(:,ind\_dim);

datos\_norm = (datos-mean(datos))/std(datos); ab\_norm(:,ind\_dim)=datos\_norm;

end

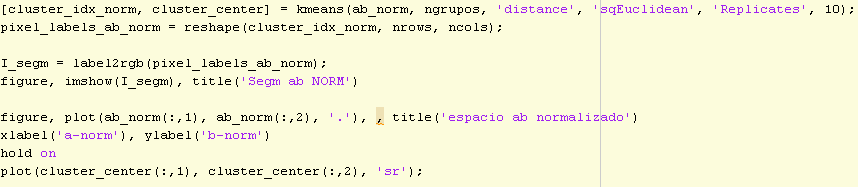
* Ahora aplicamos la normalización sobre ambas componentes (ab).
* Cogemos cada componente, y creamos la estructura ***‘ab\_res’***
* Y de cada componente le quitamos primero la media y luego dividimos por la desviación típica.
* Y yo tengo creada una nueva variable que se llama ‘ab\_norm’, que son las componentes normalizadas. Vamos guardando esas componentes normalizadas ahí.
* El codigo es asi:



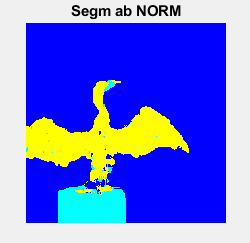
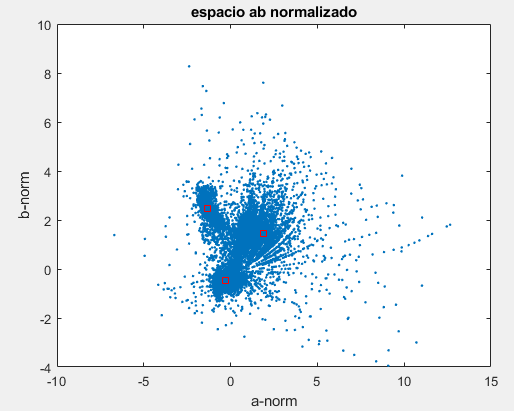
Aplique el algoritmo *k*-medias sobre el espacio de características normalizado y visualice en el espacio imagen el resultado de la segmentación.

Ahora hacemos lo mismo que antes, volvemos a aplicar el algoritmo k-medias pero esta vez sobre nuestro nuevo espacio que es ***‘ab\_norm’***

* Usando el mismo código que antes:



* Obteniendo como resultado:

¿Observa alguna diferencia respecto a la segmentación obtenida sin normalizar las características?, ¿qué conclusión puede extraer sobre la normalización y la aplicación del algoritmo *k*-medias considerando distancia Euclídea?

Si, antes en el scatter plot, los centroides estaban mucho más juntos, por lo que su ubicación no era buena, ya que no tenía bien en cuenta las distancias debido a la diferencia del rango dinámico de ambas componentes.

La normalización se debe hacer siempre que las componentes tengas distinto rango dinámico, antes de aplicar el algoritmo k-medias.

# *Características de textura*

MATLAB proporciona una serie de filtros para extraer características relacionadas con la textura. La salida de cada filtro es una imagen del mismo tamaño que la imagen original donde cada píxel contiene la característica de textura extraída en un entorno local del mismo. Consulte la ayuda de las funciones stdfilt, entropyfilt y rangefilt de MATLAB e indique qué descriptor estadístico se obtiene con cada una de las tres funciones.

Considere un entorno de 7x7 píxeles y represente en el espacio imagen el resultado obtenido al aplicar cada uno de los descriptores de textura anteriores sobre la imagen de grises original (almacenada en la variable I\_gris). Las instrucciones para representar el descriptor asociado al filtro stdfilt son las siguientes:

S = stdfilt(I\_gris,ones(7,7)); imtool(S,[]), title('S')

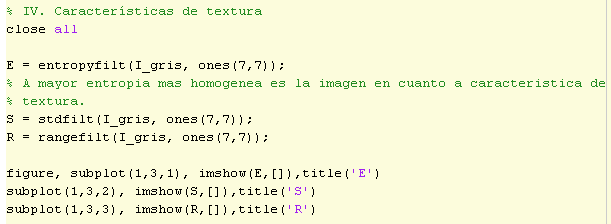
Interprete cada una de las imágenes obtenidas conforme al descriptor utilizado. Elija dos de estos descriptores como características y realice la segmentación considerando el espacio de características correspondiente.

* Hemos estado trabajando con descriptores de color, ahora vamos a trabajar con descriptores de textura:

1. El filtro de entropía: ***‘entropyfilt()’***
2. El filtro de rango dinámico: ***’rangefilt()’***
3. El filtro de desviación típica: ***‘stdfilt()’***

De tamaño 7x7.

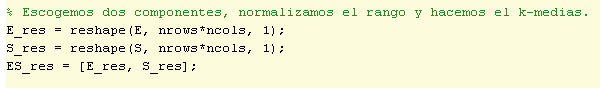
* Para los filtros de textura, hay que trabajar sobre la imagen en grises, por lo que trabajaremos sobre *‘I\_gris’.*
* Con este código:



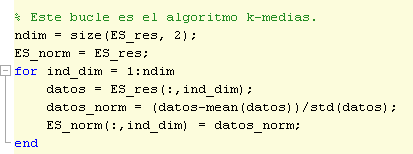
* Obteniendo este resultado:



* Elegimos dos descriptores, que van a ser E y S, ya que son los que a simple vista me van a proporcionar mayor información.
* Con esas dos componentes, las redimensionamos (porque vuelvo a tener imágenes 2D y quiero un vector columna) con el comando ***‘reshape()’*** y creo una estructura con ambas.



* Tras esto, normalizamos esas dos componentes para que tengan igual rango dinámico:



* Y aplicamos el algoritmo de k-medias sobre esta nueva estructura, ***‘ES\_norm’:***

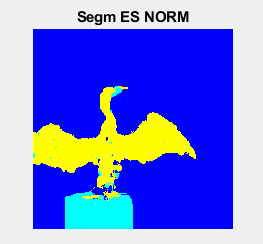


Ahora ‘cluster\_center’ tendrá una dimensión de 3x2 (3 centroides, 2 componentes (E y S).

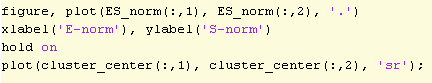
* Y tras obtener la nueva capa de etiquetas ***‘pixel\_labels\_ES\_norm’,*** la transformamos a una capa de segmentación con falso color:

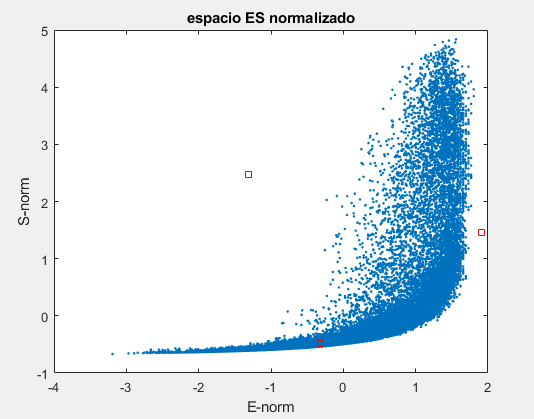


* Obteniendo un resultado así:



* Y una distribución de puntos y centroides, así:





A la vista de los resultados, y si tuviera que elegir únicamente dos características, ¿qué características elegiría? ¿las obtenidas en el Apartado III o las extraídas en este apartado?

Las del apartado anterior, ya que el color nos da mejor resultado, Y esto es debido a que en esta imagen, la textura no es tan útil como el color.

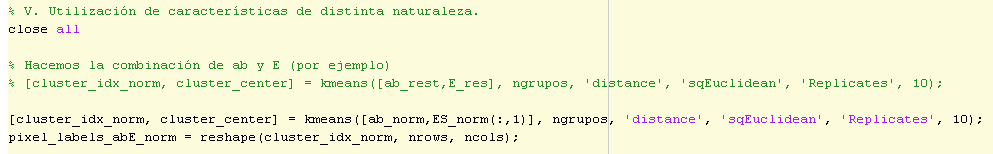
# *Utilización de características de distinta naturaleza*

Para mejorar el resultado de la segmentación obtenida en los apartados anteriores, justifique razonadamente la elección de tres características si se desea utilizar descriptores de distinta naturaleza (color y textura).

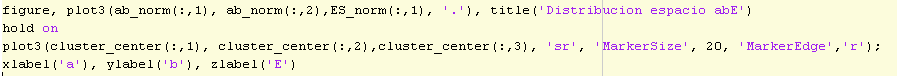
Ahora vamos a coger la componente ‘a’, ‘b’(colores) y la componente de entropía, ‘E’ (textura)

Realice la segmentación de I considerando las tres características seleccionadas (recuerde utilizar características normalizadas). A la vista de los resultados, indique qué tipo de características (color, textura), aporta más información para realizar la segmentación de esta imagen.

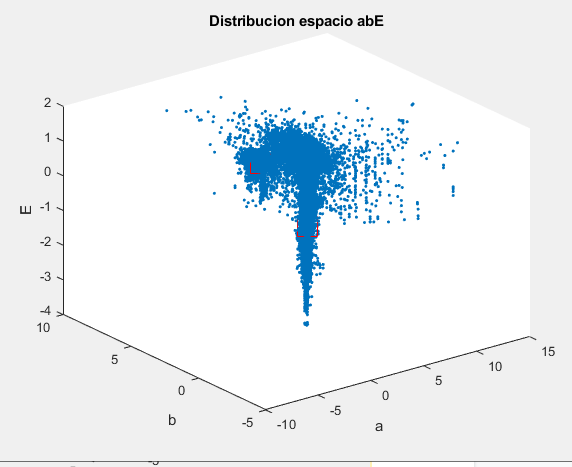
* Por lo que ahora voy a tener una nueva matriz, una nueva estructura de datos, compuesta por la componente ‘ab’ ya creada antes como ***‘ab\_norm’*** y la componente ‘E’, entropía de la creada ya antes ***‘ES\_norm(:,1)’*** , y va a tener de nuevo 3 dimensiones ahora.
* Aplicamos k-medias como anteriormente y sacamos el vector de etiquetas ***‘pixel\_labels\_abE\_norm’:***



* Representamos la distribución con ‘plot3()’ ya que ahora tenemos 3 dimensiones



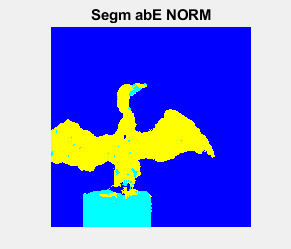
* Obteniendo un resultado así:



* Y realizamos la capa de segmentación a falso color



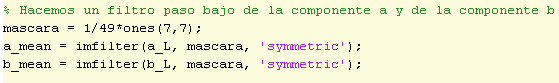
* Obteniendo un resultado así:



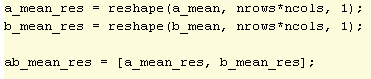
Como observará, el resultado adolece de sobresegmentación, aunque ésta corresponde a regiones de tamaño muy reducido. Para resolver este problema, una alternativa es recurrir al análisis de cada región y realizar la fusión de regiones contiguas. No obstante, en esta práctica intentaremos resolver el problema de sobresegmentación preprocesando las componentes de color, que son las que más información parece que aportan para segmentar la imagen de esta práctica. Para ello, se propone aplicar un filtro de suavizado de tamaño 7x7 píxeles sobre cada componente cromática del espacio Lab.

Aplique un filtro de suavizado sobre cada componente de color y construya el nuevo espacio de características normalizado. Aplique el algoritmo de *k*-medias y comente el resultado.

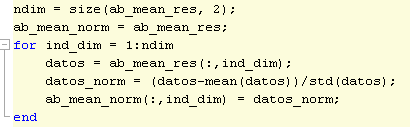
* Nos damos cuenta que para evitar la pequeña sobresegmetación que nos encontramos, tenemos que preprocesar las componente de color (ab) que son las que más información nos dan.
* Por lo que el preprocesado será aplicar un filtro paso bajo, para suavizar, de tamaño 7x7



* Redimensionamos las componentes con ‘reshape()’ y formamos una nueva estructura:



* Normalizamos las componentes para que ambas tengan el mismo rango dinámico:



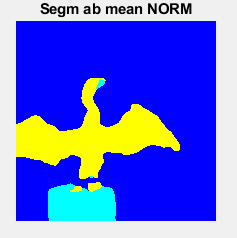
* Aplicamos el algoritmo k-medias y obtenemos el vector de etiquetas:



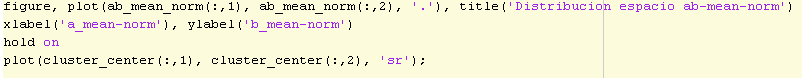
* Calculamos la capa de segmentación a partir del vector de etiquetas, y lo ponemos a falso color.



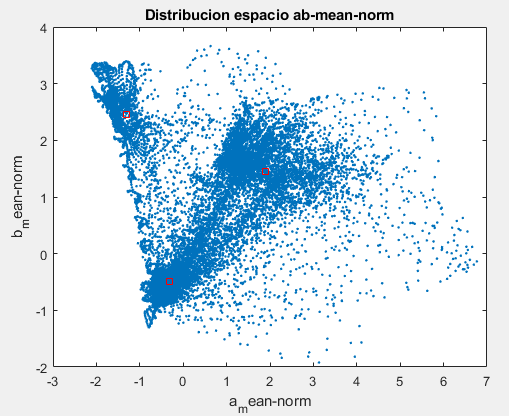
* Obteniendo esta capa de segmentación:



* Y realizamos la distribución de puntos en el nuevo espacio.



* Obteniendo este resultado:



Como comprobará, parte del pico del cormorán se asocia a la categoría “madera”, por tener un color similar. ¿Cree que la aplicación de una técnica de aprendizaje supervisado podría mejorar el resultado obtenido en esta práctica? Justifique razonadamente su respuesta.

Explique detalladamente (apoyándose en un diagrama de bloques) qué procedimiento seguiría si hubiera abordado la segmentación de esta imagen con un esquema de aprendizaje supervisado. Explique al menos una técnica de las que podría aplicar.