

TRABAJO DE EVALUACIÓN – IMPLEMENTACIÓN DE ALGORITMO BÁSICO DE SEGUIMIENTO DE OBJETOS POR COLOR

Generar una secuencia de video que muestre el seguimiento de un objeto de una escena captada por una WebCam. El seguimiento se basará en el color del objeto y se visualizará a través de una marca en el centroide de cada agrupación conexas de píxeles detectada.

Guía de ayuda y pasos:

1.- GENERACIÓN DE MATERIAL:

- 1.1- Secuencia de video para evaluar el funcionamiento del algoritmo de seguimiento: generar un archivo de video con el objeto de estudio moviéndose por una determinada región del espacio.
- 1.2- Imágenes de calibración: capturar varias imágenes con el objeto situado en distintas posiciones representativas de la región del espacio que queramos monitorizar, así como una imagen representativa del fondo de la escena (sin el objeto, en la situación más parecida a las imágenes que se hicieron con el objeto).

2.- GENERACIÓN DE CONJUNTO DE DATOS:

2.1.- EXTRACCIÓN DE DATOS DEL COLOR OBJETO DE SEGUIMIENTO Y OTROS COLORES DEL FONDO DE LA ESCENA

2.1.1.- Para cada imagen de calibración que contiene el objeto, seleccionar una región de píxeles con el color de seguimiento. Almacenar los valores R, G y B de todos los píxeles seleccionados. Para ello, utilizar una matriz Matlab *DatosColor*, con 4 campos: identificador de la imagen, valores R, G y B.

2.1.2.- Para la imagen de fondo, seleccionar varias regiones de píxeles que no sean del color de seguimiento (también se pueden utilizar las imágenes que tienen el objeto, siempre que se seleccionen regiones donde no esté el objeto). Almacenar los valores R, G y B de todos los

píxeles seleccionados. Para ello, utilizar una matriz Matlab *DatosFondo*, con 4 campos: identificador de la region, valores R, G y B.

Ayuda Matlab: instrucción *roipoly*

2.1.3.- Generación de un primer conjunto de datos X e Y:

- X: matriz de tantas filas como muestras de píxeles haya en *DatosColor* y *DatosFondo* y tres columnas (valores de R, G y B). Es decir, se genera concatenando verticalmente la información RGB de *DatosColor* y *DatosFondo*.
- Y: vector columna con dos posibles valores: 0 y 1. El valor 0 se asignará a aquellas filas de X que se correspondan con muestras del fondo; el 1 es el valor de codificación que se utilizará para indicar que la fila de datos de X pertenece a la clase de píxeles del color de seguimiento.

2.2.- REPRESENTACIÓN DE LOS DATOS DEL COLOR OBJETO DE SEGUIMIENTO Y OTROS COLORES DEL FONDO DE LA ESCENA

2.2.1.- Representar en el espacio RGB, con un rango de variación 0-255 en los tres ejes, todos los valores RGB de los píxeles del color de seguimiento y del fondo de la escena. En la representación, utilizar distintos colores para distinguir las dos clases consideradas: color de seguimiento, color/es de fondo.

2.3.- ELIMINACIÓN DE VALORES ATÍPICOS EN LOS DATOS DEL COLOR DE SEGUIMIENTO

2.3.1.- Eliminar valores atípicos o *outliers* en las muestras de X *correspondientes a los píxeles del color de seguimiento*. Para ello, se eliminará una instancia completa de esta clase de salida (color de seguimiento) si el valor de cualquiera de sus atributos está fuera de su rango “normal” de variación. Este rango se define para cada atributo como la media más menos tres veces la desviación estándar de sus valores.

2.3.2.- Generar el conjunto de datos final X e Y, sin *outliers* en la clase del color de seguimiento (las instancias anómalas eliminadas de X también han de eliminarse en Y).

2.3.3.- Representar en el espacio RGB todos los valores RGB de los píxeles del color de seguimiento y del fondo de la escena del conjunto de datos final, distinguiendo las muestras del color de seguimiento y las del fondo de la escena.

3.- DISEÑO Y ENTRENAMIENTO DEL CLASIFICADOR

3.1.- ELECCIÓN DE ESTRATEGIA DE CLASIFICACIÓN

Objetivos al realizar el seguimiento del objeto. Diferentes posibilidades:

- 1.- No perder el objeto de seguimiento en sus diferentes posiciones, aunque esto suponga detectar fondo de la escena no deseado (ruido).
- 2.- No detectar nada de ruido de fondo, aunque esto suponga dejar de detectar algunos píxeles del color objeto de seguimiento.
- 3.- Compromiso en la detección de píxeles del objeto y el ruido de fondo: se debe intentar detectar el mayor número de píxeles del objeto, minimizando la cantidad de ruido.

Planteamiento:

Considerar que el color de seguimiento está compuesto por todos los puntos de una determinada región del espacio RGB. Para determinar esta región pueden utilizarse diferentes estrategias. Vamos a considerar que esta región estará delimitada por superficies esféricas, cuyos centros y radios habrá que determinar para que se ajusten y contengan a todas las muestras disponibles del color de seguimiento.

Para ello, como paso previo, se debe aplicar el algoritmo de agrupamiento de datos que se facilita como documentación anexa. Este algoritmo permite dividir las muestras del color de seguimiento en un número de agrupaciones igual al número de esferas que se deseen emplear.

Este planteamiento considera que un píxel cuya componente de color RGB se encuentre contenido en una de las esferas anteriores, dada por un centro y un radio, es un píxel del color de seguimiento. Esto se puede evaluar verificando que la distancia Euclídea entre los valores RGB del píxel en cuestión y el centro de cualquier esfera sea menor que el radio de ésta.

3.1.1.- Añadir a la representación del apartado 2.2.1 las superficies esféricas a las que hace referencia el planteamiento anterior. Para ello, se deben agrupar los datos disponibles para el color de seguimiento y, para cada agrupación de datos obtenida, se determinará el centro y radios posibles de las esferas asociadas a cada agrupación. Según el objetivo perseguido, los valores de radio de las esferas que pueden ser de interés pueden calcularse de la siguiente forma:

- Valores de radio para detectar el mayor número posible de píxeles del color objeto de seguimiento (objetivo número 1): para cada esfera, calcular todos los valores de distancia entre los valores (R, G, B) de los píxeles del color de la agrupación y el color medio (R_c , G_c , B_c). El valor de distancia máxima es el valor de radio de la esfera que contiene a todos los píxeles de la agrupación.
- Valores de radio para no detectar ruido de fondo (objetivo número 2): se debe seleccionar un valor de radio de las esferas que no contengan ruido de fondo, para lo cual habrá que medir la distancia de todos los puntos de fondo al centro de la esfera y comprobar que se sitúan fuera de la esfera.
- Valores de radio de compromiso en la detección de píxeles del objeto y ruido: pueden considerarse los valores de radio promedio de los valores obtenidos en los puntos anteriores.

Ayuda Matlab:

Representación de superficie esférica centrada en (R_c , G_c , B_c) de radio *Radio*

```
[R,G,B] = sphere(100);
```

```
% Matrices de puntos de una esfera centrada en el origen de radio  
unidad
```

```
x = Radio*R(:)+Rc; y = Radio*G(:)+Gc; z = Radio*B(:)+Bc;  
plot3(x,y,z, 'b')
```

Cálculo de la distancia Euclídea entre dos vectores columna A y B :

```
Distancia = sqrt (sum((A - B).^2)); % Distancia = sqrt ((A - B)'*(A - B));
```

3.2.- ENTRENAMIENTO DEL CLASIFICADOR: CALIBRACIÓN Y AJUSTE DE PARÁMETROS

Parámetros de calibración:

Umbral de distancia Euclídea (radios de las superficies esféricas): el algoritmo de seguimiento calculará la distancia Euclídea de todos los píxeles de la imagen respecto a los centros de las superficies esféricas consideradas; considerará que los píxeles cuyas componentes de color se desvíen menos de una distancia umbral respecto al centro de cualquier esfera considerada, son de ese color. Hay que encontrar valores apropiados para estos umbrales de distancia.

Umbral de conectividad: el paso anterior decide valores adecuados de umbral de distancia y da lugar a una imagen binaria resultado de umbralizar medidas de distancia. A continuación, el algoritmo descartará aquellas componentes conexas cuyo número de píxeles sea inferior a uno dado. Hay que ajustar este parámetro.

3.2.1.- Procedimiento de ajuste umbral de distancia: para cada una de las imágenes de calibración y varios valores posibles de umbral de distancia (en el apartado 3.1.1 se han calculado 3 posibles valores con distintos criterios):

- Calcular matrices de distancias.
- Detectar aquellos píxeles cuyo color “se parezca” al color del seguimiento (binarizar la matriz distancia los umbrales de distancia).
- Visualizar, sobre la imagen original, el resultado de la detección.
- Analizar las gráficas, crear nuevas con otros valores de umbrales si fuese necesario y decidir un valor de umbral apropiado.

3.2.2.- Procedimiento de ajuste umbral de conectividad: para cada una de las imágenes de calibración y varios valores posibles de umbral de conectividad (en este caso es una referencia

útil saber el número de píxeles que tiene el objeto de seguimiento en su posición más alejada, que es cuando tiene menor tamaño):

- Calcular las matrices de distancias y binarizarlas con los umbrales seleccionados.
- Eliminar las componentes conexas más pequeñas de acuerdo al umbral de conectividad.
- Visualizar sobre la imagen original el resultado de este proceso de filtrado.
- Analizar las gráficas, crear nuevas con otros valores de conectividad si fuese necesario y decidir un valor de umbral apropiado.

3.2.3.- Guardar en una variable matlab los parámetros de calibración: color medio de seguimiento, valores elegidos de umbral de distancia y conectividad.

4.- IMPLEMENTACIÓN Y VISUALIZACIÓN DE ALGORITMO DE SEGUIMIENTO

Para comprobar cómo se comporta el algoritmo de seguimiento, este se aplicará sobre el archivo de video generado. Atendiendo a la estrategia de funcionamiento elegida, el algoritmo debe:

- Cargar los parámetros de calibración.
- Leer la secuencia de video. Para cada *frame* de la misma:
 - Calcular matrices de distancias.
 - Detectar aquellos píxeles cuyo color se considere que sea del color del seguimiento
 - Eliminar las componentes conexas más pequeñas.
 - Marcar el centroide de los objetos presentes.
- Generar la secuencia de video que muestre el seguimiento del objeto.

DOCUMENTACIÓN ANEXA

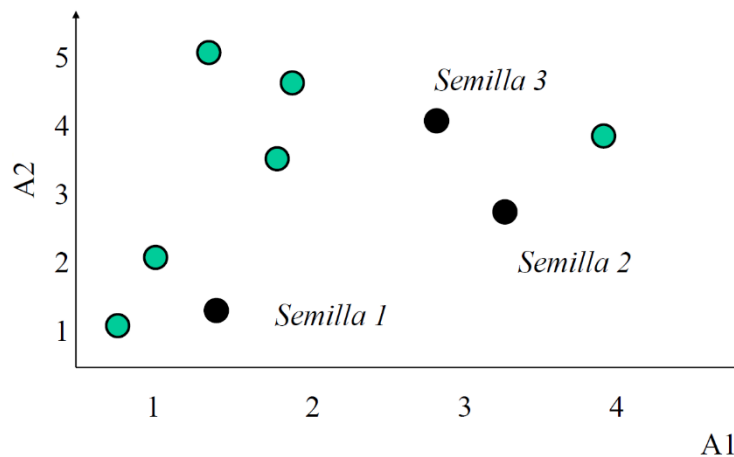
ALGORITMO DE AGRUPAMIENTO K-MEDIAS (K-MEANS)

Objetivo: dividir un conjunto de datos en k grupos de mínima distancia (minimizando las distancias entre elementos dentro de cada grupo).

Observación: no está permitido utilizar la función de Matlab `kmeans`, se debe implementar la siguiente función de acuerdo a las siguientes indicaciones.

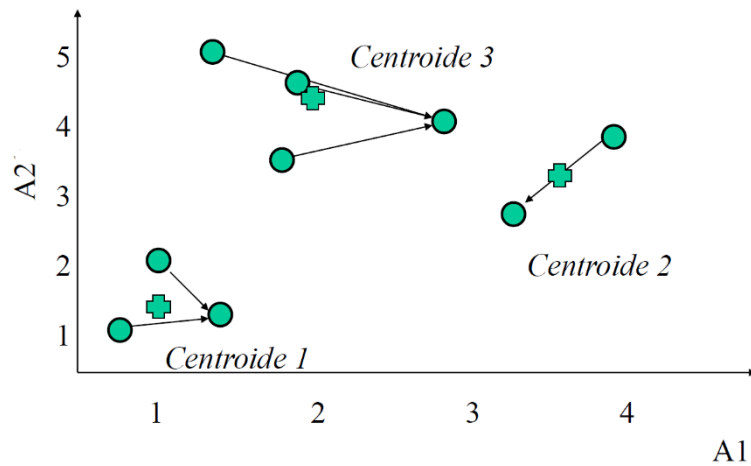
Planteamiento del algoritmo:

1. Se eligen k muestras del conjunto de datos para que actúen como semillas. Estas semillas definen las k clases en las que se pretenden agrupar los datos.

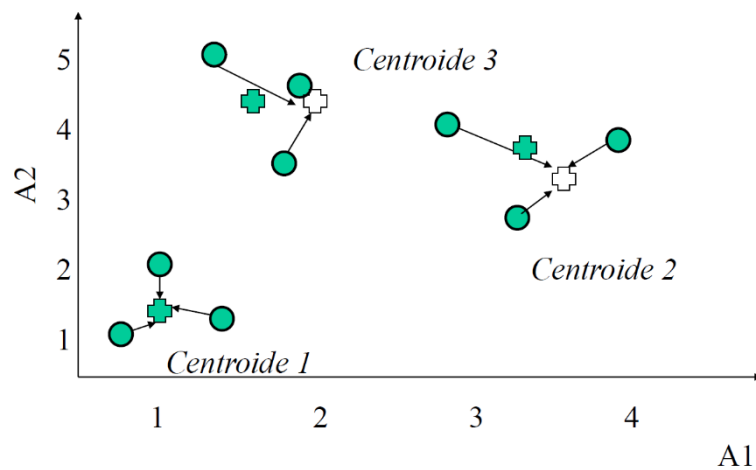


Inicio de inicio del algoritmo: se asume que cada muestra está descrita por dos atributos A1 y A2 (espacio de observaciones bidimensional definido por estos atributos)

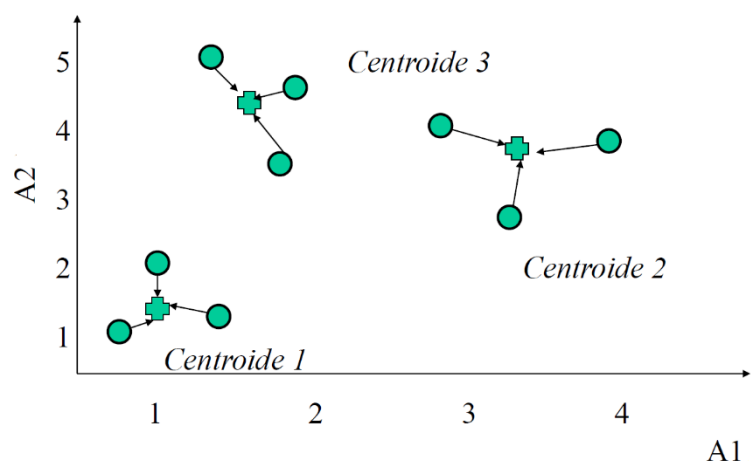
2. Cada muestra del conjunto de datos se añade a la clase más similar (por ejemplo, la muestra se asocia a la clase que se encuentre más cerca en el espacio de observaciones).
3. Se calcula el centroide de cada clase resultante del punto anterior, que pasan a ser las nuevas semillas de las clases.
4. Se repiten los puntos 2 y 3 hasta que se llega a un criterio de convergencia (por ejemplo, dos iteraciones no cambian las clasificaciones de las muestras)



Ejemplo de primera iteración (pasos 2 y 3): cada muestra se asocia a la semilla que tiene más carga y se calculan los centroides de cada agrupación resultante.



Ejemplo de segunda iteración



Ejemplo de agrupamiento final

IMPLEMENTACIÓN DE ALGORITMO DE AGRUPAMIENTO

```
[idx, centroides] = funcion_kmeans(X, k)
```

Sea un conjunto de datos compuesto por n instancias (observaciones, muestras, ejemplos) descritas por un vector de atributos p -dimensional:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{bmatrix} ; \text{ idx} = \begin{bmatrix} \text{idx}_1 \\ \vdots \\ \text{idx}_n \end{bmatrix} \text{ idx} = 1, 2, \dots, \text{ ó } k$$

Entradas y salidas:

- X : matriz de n filas y p columnas; cada fila almacena los valores de los atributos de una determinada muestra (cada atributo se corresponde con una columna)
- k : número de agrupaciones en las que queremos dividir los datos.
- idx : vector columna con tantas filas como muestras haya en X (matriz $n \times 1$). Cada fila contiene la etiqueta asignada a la clase o agrupación a la que se asocia la muestra de la fila correspondiente de X . Los valores de estas etiquetas para identificar la clase son de 1 a k (números enteros).
- centroides : matriz de k filas (número de agrupaciones) y p columnas (número de atributos que describen las muestras). Contiene las coordenadas del centroide (en el espacio de observaciones definido por los atributos) de cada agrupación final que generada por el algoritmo.

PASOS (las funciones a las que se hace referencia están definidas a continuación en el apartado *funciones auxiliares*):

- 1. Inicialización:** generar una primera agrupación de los datos, idx_semilla , mediante la función `funcion_agrupa_por_desviacion`. Fijarse que el algoritmo no empieza con unos centroides elegidos como semillas, tal como se ha descrito anteriormente en la introducción del algoritmo, sino con una agrupación inicial.

2. Repetir iterativamente:

2.1.- Calcular la matriz `centroides` para la `idx` resultante de la iteración anterior (en la primera iteración, se toma la agrupación `idx_semilla`). Para ello utiliza la función `funcion_calcula_centroides`.

2.2.- Calcula una nueva agrupación a partir de los centroides generados en el paso anterior. Para ello, utiliza la función `funcion_calcula_agrupacion`.

2.3. Compara la agrupación obtenida con la anterior generada. Para ello utiliza la una función `funcion_compara_matrices`.

2.4.- Salir si no hay cambios en las agrupaciones.

3. Devolver la última agrupación y centroides obtenidos.

FUNCIONES AUXILIARES:

1. Función agrupa por el atributo de máxima desviación:

```
idx = funcion_agrupa_por_desviacion(X, k)
```

Objetivo y criterio de agrupación de la función: generar una agrupación inicial de los datos para que la convergencia del algoritmo sea más rápida. Esta agrupación se realiza atendiendo a los valores del atributo que presenta mayor desviación en sus datos (evaluada por la desviación estándar). El rango de valores de variación de este atributo de máxima dispersión, se divide en `k` intervalos. Las muestras se agrupan atendiendo a su valor del atributo de máxima dispersión: se asigna al intervalo `k` que contiene el valor.

1.1.- Identificar el atributo de máxima dispersión (función `std`) y almacenar sus valores en una nueva variable.

1.2.- Ordenar de menor a mayor los valores de este atributo (función `sort`)

1.3.- Calcular valores representativos de estos valores: valor mínimo y máximo (funciones `min`, `max`). Si dividiéramos esos valores en `k` intervalos iguales, desde el valor mínimo hasta el máximo calculado, calcular la amplitud de ese intervalo.

1.4.- Recorrer cada uno de esos k intervalos e ir generando el vector de salida idx : todas las muestras de X que presenten valores del atributo de máxima dispersión dentro de los valores mínimo y máximo del intervalo bajo consideración, se etiquetan en idx con el número de intervalo.

Observación importante: se debe garantizar que haya muestras de X en todos los intervalos; en caso de que un intervalo no contenga ninguna muestra de X , ampliar el valor máximo que define ese intervalo para que incluya al menos una muestra de X .

2. Función calcula centroides de una determinada agrupación:

```
centroides = funcion_calcula_centroides(X,idx)
```

Objetivo: calcular la matriz de centroides (ver definición en la descripción de los parámetros de entrada de la función principal) de los datos X agrupados según idx .

3. Función calcula centroides de una determinada agrupación:

```
idx = funcion_calcula_agrupacion (X, centroides)
```

Objetivo: genera la agrupación de los datos X de acuerdo a $centroides$. Para ello, se debe asignar cada muestra de X al centroide más próximo según distancia Euclídea.

4. Función compara matrices:

```
varLogica = funcion_compara_matrices (matriz1, matriz2)
```

donde $varLogica$ toma el valor `True` si las matrices son iguales y `False` en caso contrario.