Astor Prieto DehghanPour
niUB: 16445586

Visión Artificial – Practica 3

1. Métodos de ‘Background Substraction’

En este primer ejercicio, usamos un video (‘Barcelona.mp4’) conformado por escenas captadas por una cámara estática. En el programa, usamos los canales del histograma de cada frame como medida para saber si se ha cambiado de ‘shot’ (es decir, de escena). Si el cambio entre dos histogramas de los frames es superior a un valor threshold (tomado como ’40.000’ en nuestro caso), asumimos que tenemos un ‘shot’ aquí, y se cambia de escena. Seguidamente, aplicamos un algoritmo de ‘background substraction’ como el visto en teoría, para poder encontrar las imágenes estáticas que conforman el fondo de cada escena. Si los ‘shots’ no se extraen correctamente, a la hora de encontrar la imagen estática de fondo usando la mediana de los frames, veremos que la imagen resultante queda poco definida.

Script: ej31.m

2. Métodos de agrupación de datos numéricos

En este ejercicio se pedía que, haciendo uso de la función proporcionada *gaussRandom.m*, creáramos tres nubes de 100 puntos siguiendo una desviación estándar de 0.1. Una vez creada la distribución de datos gaussiana, se pide que usando diferentes números de centros, agrupemos los datos anteriores usando ‘k-means’. Los datos resultantes al aplicar ‘k-means’ con diferentes números de centros (*k = 2, 3, 4;*) se asemejan muchísimo a los datos originales, si usamos *k=3* podemos observar como las nubes se distribuyen (en esos 3 centros) prácticamente igual que en los datos originales.

Script: ej32.m

3. Métodos de agrupación: segmentación en el espacio RGB

En este ejercicio, se pide primero que leyendo la imagen ‘loro.png’ se segmente la imagen equivalente en *grayscale*. La imagen resultante de la segmentación en gris son las principales regiones diferentes de la imagen. Finalmente, se pide la segmentación y transformación de la imagen en RGB. Usando el método ‘k-means’, traducimos el numero original de colores a una escala de 16 colores equivalentes por regiones.

Script: ej33.m

4. Extracción de descriptores

En este apartado, se pide que instalemos el ‘VLFeat toolbox’ para poder hacer uso de los métodos de SIFT (‘*vl_sift()*’) y posteriormente, que sigamos su ejecución y entendamos su

funcionamiento. El método *vl_sift()* devuelve dos valores: ‘f’, correspondiente a los *key frames* de la imagen y ‘d’, donde se guardaran los descriptores de la imagen.
Si observamos los puntos característicos extraídos por el método SIFT, podemos ver que en los círculos dibujados en la imagen (puntos característicos) también encontramos una línea en su interior, correspondiente a la dirección de su descriptor de imagen. El método de extracción de *features* de SIFT respeta los vectores independientemente de cualquier reescalado o rotación que pueda haber sufrido la imagen.

Script: ej34.m

5. Reconocimiento por alineación de puntos característicos

En este último ejercicio, se pedía que, usando el *toolbox* instalado anteriormente, usemos SIFT para encontrar usando una imagen modelo (‘starbucks.jpg’) si diferentes imágenes de una colección tienen mucha o poca probabilidad de parecerse al modelo. Si las imágenes que le pasamos para encontrar *matching* son muy diferentes al modelo, el programa encontrara problemas a la hora de converger en un resultado y no podrá decir si la imagen se parece o no.

Script: ej35.m