E2-Segmentación y extracción de características

Instrucciones

En respuesta a las preguntas que se plantean a continuación, se debe entregar en la plataforma de enseñanza virtual (por separado, **no** en un archivo comprimido)

- un archivo con extensión .ipynb creado con notebook Jupyter de Anaconda que contenga, tanto las respuestas teóricas, como el código programado;
- el archivo anterior en .pdf (File->Print preview->guardar como pdf);
- una declaración de autoría firmada por el alumno/a según la plantilla proporcionada;
- las imágenes/vídeos usados.

La resolución de todos los ejercicios (tanto teóricos como prácticos) debe ser **razonada** y el código desarrollado debe ser **explicado en detalle, justificando todos los parámetros** escogidos y **referenciando las fuentes.**

Objetivo

El objetivo de este entregable es realizar algunos ejercicios de segmentación y extracción de características.

Ejercicio 1. Segmentación – Thresholding [0,4 puntos]

Para la realización de este ejercicio se seleccionará una imagen de la base de datos disponible en el siguiente enlace:

https://www.wisdom.weizmann.ac.il/~vision/Seg_Evaluation_DB/download/Weizmann_Seg_DB_1obj.ZIP

Todas estas imágenes contienen un único objeto principal (objeto a segmentar). Además, para cada una de ellas, podemos encontrar la imagen segmentada "ground truth" con la que compararemos con los distintos métodos utilizados.

Usaremos para ello el Coeficiente Dice, una de las medidas más extendidas para medir el rendimiento en segmentación de imagen. Se calcula como:

$$2 * |A \cap B| / (|A| + |B|)$$

Donde |A| representa el número de píxeles etiquetados como objeto principal en la imagen segmentada, y |B| representa el número de píxeles etiquetados como objeto en la imagen "ground truth". |A ∩ B| representa el número de píxeles que han sido etiquetados como objeto en la imagen original y que además lo son en "ground truth" (verdaderos positivos).

La función *threshold* de OpenCV permite aplicar distintos métodos de umbralización para la segmentación de imágenes.

https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html

1. Selecciona una de las imágenes de la base de datos indicada, cárgala como imagen en escala de grises y representa su histograma.
2. Determina razonadamente mediante la visualización del histograma el umbral que consideras podrían proporcionar una primera aproximación a una segmentación adecuada del objeto.
3. Aplica el umbral seleccionado en el apartado 2 usando la función *threshold* de OpenCV y comenta el resultado obtenido.
4. Utiliza la función *threshold* de OpenCV para aplicar el método de Otsu, y comenta el resultado obtenido con respecto a la segmentación obtenida previamente.
5. Implementa una función que calcule el coeficiente Dice comparando las segmentaciones obtenidas con la imagen "ground truth" proporcionada en el set de datos. Compara los dos resultados anteriores en términos de este coeficiente.

Ejercicio 2. Segmentación basada en clustering - MeanShift [0,7 ptos.]

La función *meanShift* de OpenCV se utiliza en el siguiente fragmento de código (Fragmento de código 1) para hacer seguimiento de uno de los objetos en movimiento del video "video.avi".

Fragmento de código 1:

```
```
import cv2
import numpy as np
video = cv2.VideoCapture("video.avi")
_, first_frame = video.read()
x = 200

```
y = 60
width = 40
height = 40
roi = first_frame[y: y + height, x: x + width]
hsv_roi = cv2.cvtColor(roi, cv2.COLOR_BGR2HSV)
roi_hist = cv2.calcHist([hsv_roi], [0], None, [180], [0, 180])
roi_hist = cv2.normalize(roi_hist, roi_hist, 0, 255, cv2.NORM_MINMAX)
term_criteria = (cv2.TERM_CRITERIA_EPS | cv2.TERM_CRITERIA_COUNT, 10, 1)

while (True):
    success, frame = video.read()
    if success==True:
        hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)
        mask = cv2.calcBackProject([hsv], [0], roi_hist, [0, 180], 1)
        _, track_window = cv2.meanShift(mask, (x, y, width, height), term_criteria)
        x, y, w, h = track_window
        cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
        cv2.imshow("Mask", mask)
        cv2.imshow("Frame", frame)
        k = cv2.waitKey(1) & 0xff
        if k == ord('q'):
            break
    else:
        break
video.release()
cv2.destroyAllWindows()
```

1. Utiliza el fragmento de código que aparece al final del enunciado (Fragmento de código 2) para capturar un video desde la cámara web del equipo y almacenarlo en el fichero "video.avi". En el video, que debe ser breve, debe aparecer un objeto en movimiento que se rote en algún momento.
2. Adapta el código anterior (Fragmento de código 1) para hacer seguimiento del objeto en movimiento que aparece en el video capturado.
3. Explica detalladamente los cambios que has tenido que realizar para que el seguimiento funcione correctamente.
4. Explica detalladamente qué objetivo tiene la llamada a la función "*calcBackProject*" dentro del método de seguimiento.
5. Explica detalladamente qué espacio de características se está usando en este caso.
6. Describe (relacionándolo con lo visto en la asignatura) qué función concreta tiene el método MeanShift dentro del seguimiento del objeto.

Se valorará la claridad de la explicación y que sea lo más didáctica posible (que no sea simplemente traducir o copiar explicaciones del tutorial).

Fragmento de código 2:

```
video_capture = cv2.VideoCapture(0) # Se abre la primera cámara (y la única) disponible en el equipo

fourcc = cv2.VideoWriter_fourcc(*'XVID') # Se indica que se va a utilizar el códec "xvid"

output = cv2.VideoWriter('video.avi', fourcc, 30.0, (640,  480)) # Archivo a guardar, con 30 FPS y con tamaño 640x480

while video_capture.isOpened(): # Mientras no se cierre (con la tecla "q") la grabación
   ret, frame = video_capture.read() # ret es True si es correcto y False cuando ha terminado el fichero o la grabación

   # Si ret es False, es que el frame no se obtiene bien o que el vídeo ha llegado a su fin
   if not ret:
      print("ret es False, ¿ha finalizado el vídeo?")
      break

   output.write(frame) # Se escribe en el archivo de output el frame actual

   cv2.imshow('frame', frame) # Muestra cada frame que se está capturando en una ventana

   if cv2.waitKey(1) == ord('q'): # Pulsar la tecla "q" para terminar la grabación
      break

video_capture.release() # Se libera el "VideoCapture"
output.release() # Se libera el archivqo de output
```

Ejercicio 3 - Extracción de características - Harris, SIFT, ORB [0,4 ptos.]

En la documentación de OpenCV se puede encontrar información sobre cómo calcular Harris Corners, SIFT y ORB:

https://docs.opencv.org/4.x/dc/d0d/tutorial_py_features_harris.html
https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html
https://docs.opencv.org/4.x/d1/d89/tutorial_py_orb.html

Selecciona dos frames del video realizado en el ejercicio anterior, en los que el objeto aparezca en distintas posiciones de rotación. Calcula los tres tipos de puntos característicos a ambas imágenes y explica el resultado obtenido. Razona cuál o cuáles de ellos (Harris, SIFT, ORB) podría utilizarse para hacer un seguimiento del objeto en vídeo.