PRÁCTICA: Unidad 3 - Detección de Bordes y Segmentación

Ejercicio 1

En la figura 1 se muestra un Sudoku (archivo *sudoku.jpeg*), se pretenden detectar las 20 líneas que delimitan las celdas a partir de la **transformada de Hough** 'cv2.*HoughLines()'* y dibujarlas en la imagen.

Luego, con el resultado obtenido anteriormente, se debe calcular **automáticamente** el número total de celdas vacías para determinar el porcentaje de avance del juego (independientemente si está correcto o no).

Por último, las celdas vacías deberán ser pintadas de color gris. El programa mostrará la imagen final con las líneas detectadas resaltadas, las celdas vacías en gris y el porcentaje de avance del juego como título de la imagen.

5		9				4		
7		8	3		4	9		
6		1				7	3	
4	6	2	5					
3	8	5	7	2		6	4	9
1		7	4		8	2		
2			1					4
		3		4			8	7
	7			5	3			6

Figura 1: Imagen del archivo *sudoku.jpeg*.

Ejercicio 2

A partir del *dataset* de imágenes que se muestra en la figura 2 (archivos *tateti_<id>.png*), se requiere realizar un **análisis automático** mediante técnicas de PDI, sobre cada partida de TA-TE-TI.

Implemente los ítems que se detallan a continuación, en cada una de las imágenes del *dataset* provisto y exponga cada resultado obtenido.

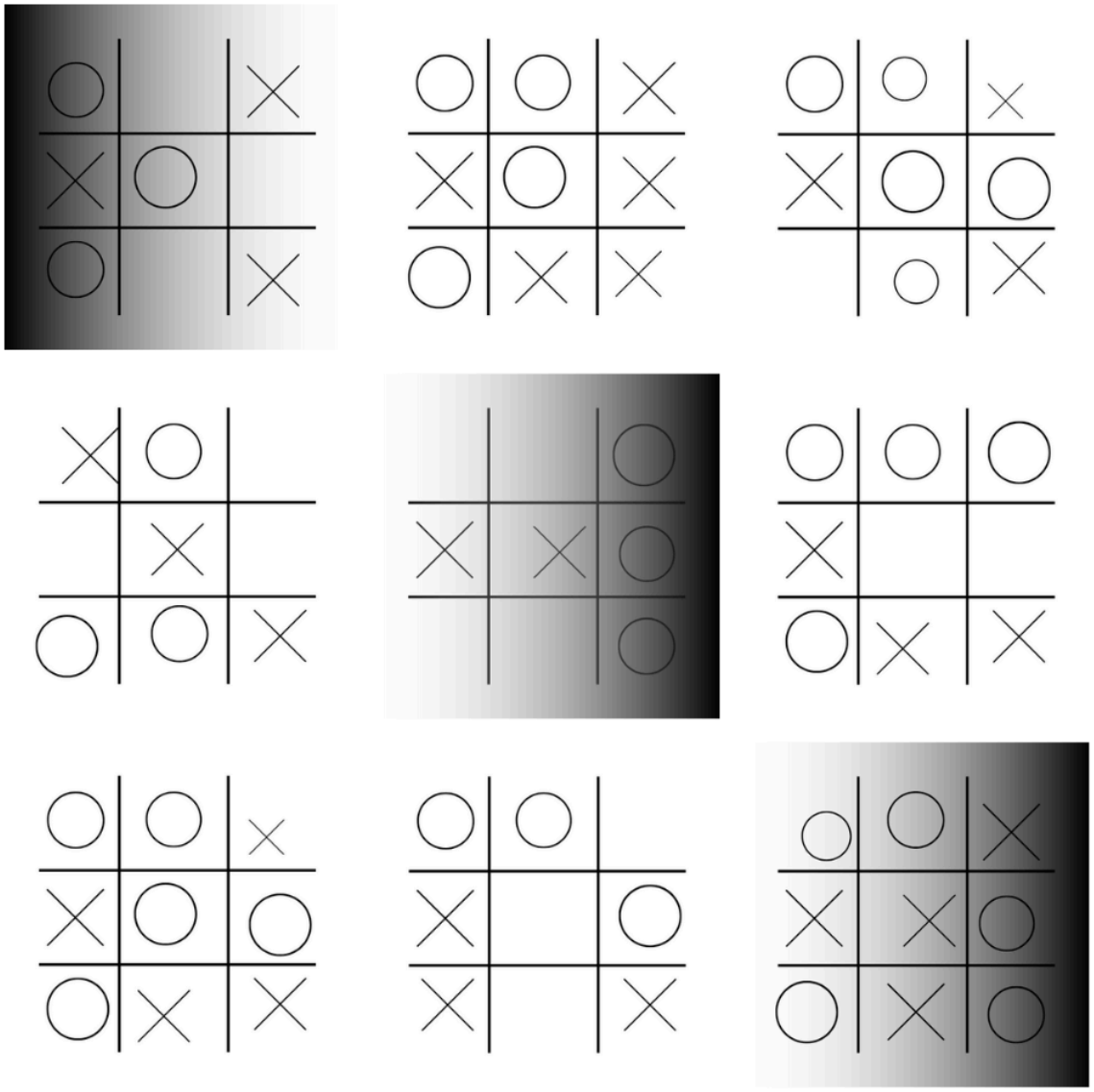

Figura 2: *Dataset* de imágenes de partidas de TA-TE-TI.

2.1 Carga de la imagen de entrada

a) Cargar la imagen desde el archivo *tateti_<id>.png*, extraer su información básica y mostrarla en una figura.

2.2 Detección de bordes y figuras geométricas

b) Elaborar un *script* que implemente las siguientes consignas de forma automática:
   i) A partir de la imagen original, convertir la misma a escala de grises y obtener su representación binaria con alguna técnica de **detección de bordes**.
   Nota: Para la detección puede hacer uso del método *cv2.Canny()*.
   ii) Con la imagen binaria y alguna técnica de **detección de líneas**, identifique las rectas horizontales y verticales que conforman las divisiones del tablero.
   Nota: Para la detección puede hacer uso del método *cv2.HoughLines()*.
   iii) Mediante el uso de *subplots*, mostrar la imagen en escala de grises, la imagen binaria y la imagen original con la representación de cada recta detectada (trazos de color azul), en una única figura.
   iv) Recortar y etiquetar cada región del tablero mediante el uso de la información provista por las rectas detectadas. Muestre el resultado en una única figura, utilizando *subplots* y respetando la disposición original de las mismas según el tablero de TA-TE-TI.
   v) Utilizar alguno de los métodos conocidos que permita determinar si la misma posee algún elemento o si se encuentra vacía.
   vi) En aquellas regiones que posean un objeto, identificar si el mismo es un círculo o una cruz.
   Nota: Para la identificación puede hacer uso del método *cv2.HoughCircles()*.
   vii) Bajo la disposición original del tablero de TA-TE-TI, utilizar *subplots* para mostrar cada recorte. Asignar la etiqueta correspondiente (Cruz, Círculo o Vacío), como título de los mismos.

c) Extra: En este juego, cada jugador debe colocar una figura (cruz o círculo) en algún espacio vacío con el objetivo de formar una línea (tres figuras) horizontal, vertical o diagonal. Si el tablero se completa y ningún jugador pudo formar alguna de ellas, el juego finaliza en un empate.
   i) Implementar en el *script*, una función que permita determinar de forma automática, si la partida concluyó y en tal caso, qué figura fue la vencedora (círculos o cruces).
   ii) Mostrar la imagen original exponiendo el resultado de la partida en su título.