IMPLEMENTACIÓN DEL SISTEMA INTELIGENTE “SODOKU”

CURSO 21-22

SANTIAGO BAUZÁ HIRSCHLER, bo0476

Índice

1. Introducción

La implementación del trabajo se va a realizar con el lenguaje de Python. Consistirá en resolver a partir de una imagen el problema del sudoku de esta. Este PDF presentará como forma de guía los pasos que hemos utilizado para poder resolver este problema. Gracias a lo dado en clase e información que hemos encontrado en internet para poder realizar esta implementación. El proyecto estará dividido en 3 clases una que será la main y otras dos que serán el algoritmo para resolver el sudoku y una clase de utilidades. Se añadirá una carpeta de recursos con las imágenes capturadas de sudokus con los que se harán las pruebas y a medida que vayamos viendo la forma que van tomando esos resultados a partir de las imágenes los iremos representando aquí, al final pondremos una breve conclusión de lo aprendido en este proyecto.

2. SudokoSolver.py

Este archivo del proyecto se dedica a resolver problemas de sudokus, el código que hemos extraído de internet tiene como créditos el autor Tim Ruscica. Donde dejaremos en el anexo un link para poder recoger más información sobre su código.

3. Utils.py

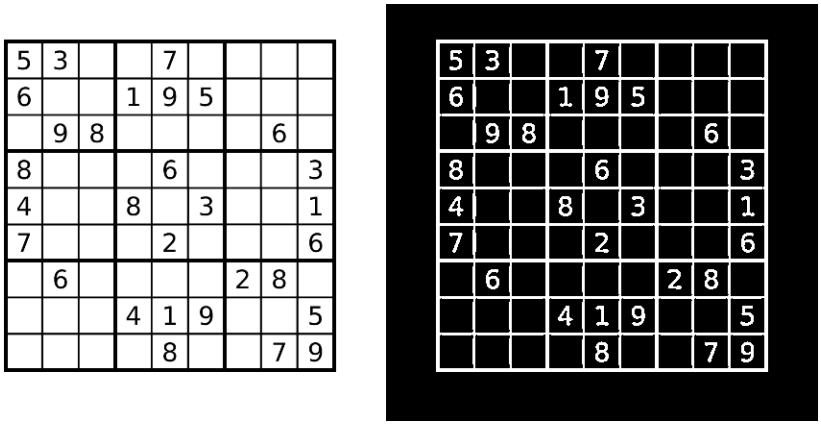
El archivo de útiles servirá para preparar las distintas imágenes de las cuales vamos a hacer diferentes capturas para identificar paso por paso los procesos que se encargará por ejemplo de convertir la imagen en escala de grises o por ejemplo encuadrar las imágenes de los sudokus y escribir sobre la misma imagen.

4. SodokuMain.py

Aquí vamos a demostrar detalladamente paso por paso como el programa analiza las diferentes imágenes y procesa los resultados.

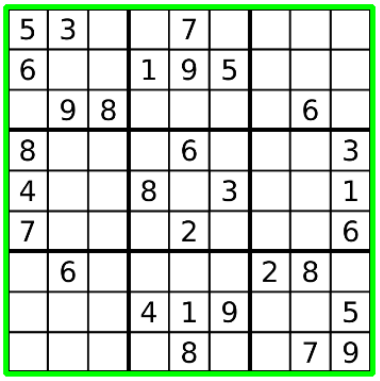
4.1 Preparación de la imagen

Para poder procesar las imágenes tienen que estar en formato JPG, leemos la imagen y escaldamos con una altura y anchura de 450. Una vez ajustada la imagen procesamos a través de una función el umbral de la imagen convertida en escala de grises.



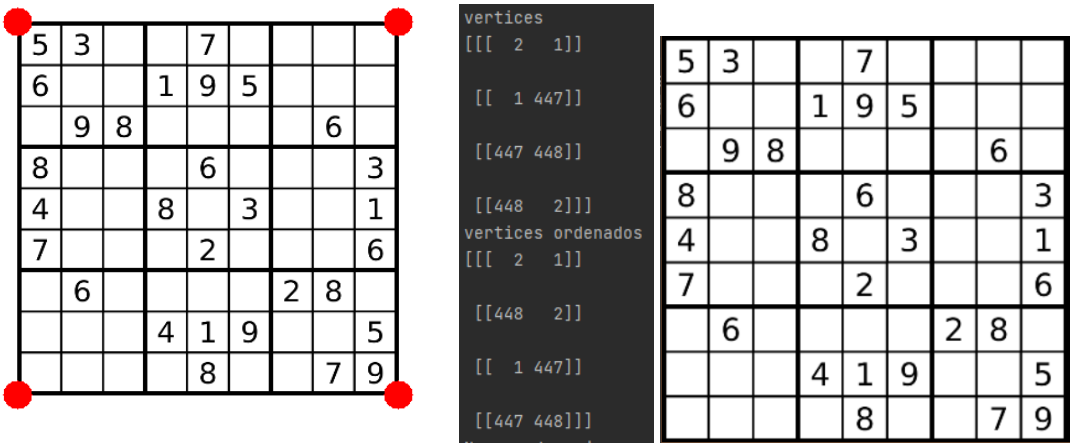
4.2 Encontrar contornos

El segundo paso es encontrar todos los contornos exteriores a lo que sería la imagen del sudoku para ello utilizamos una función utilizada en clase para buscar contornos fuera de los márgenes del sudoku.



4.3 Encontrar los márgenes del sudoku

Para encontrarlos los márgenes del sudoku utilizaremos unas funciones creadas en el archivo de utilidades y utilizaremos el contorno más grande. Lo primero, es buscar el contorno más grande y con unas probabilidades en relación con las esquinas marcadas y unas medidas del área predefinidas en la función, conseguiremos hallar la misma imagen del sudoku sin contenido fuera de los márgenes.

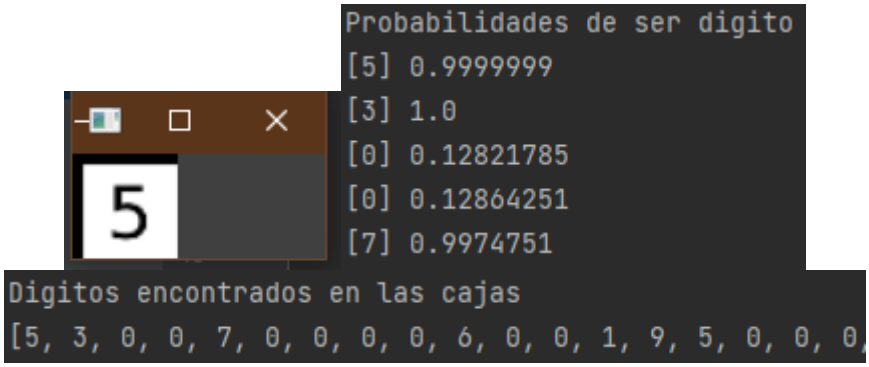


4.4 Dividir la imagen en 81 y hallar los dígitos

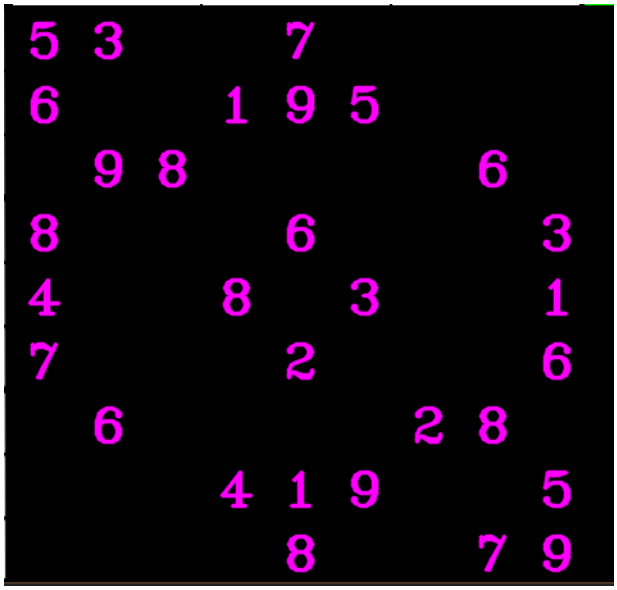
Ahora que tenemos la imagen del sudoku debemos dividir esa imagen en 81 imágenes. Para ello utilizaremos unas funciones dadas en el archivo de utilidades, lo primero es dividir la imagen en 81 cajas (utilizando numpy que trabaja por píxel de arrays), luego preparar la imagen, es decir dividir la imagen de diferentes cajas de las cuales utilizaremos un modelo que nos ayudará a predecir si se encuentra un número o si está vacío.

La probabilidad de que se encuentre un número en la imagen tiene que ser superior a 0,8% en cualquier otro caso se asignará como vacío. para poder realizar estas operaciones se deberá implementar la librería tensorflow para poder utilizar el modelo de carga que se encargará de salvar los números.

También se podrá utilizar Tesseract OCR, es un motor de reconocimiento óptico.



Para poder comprobar que se ha hecho correctamente y que ha cogido todos los números disponibles, en el mismo paso cuatro imprimiremos sobre una copia de la misma imagen sobre un fondo negro los números del sudoku para comprobar qué son los dígitos encontrados en las cajas los numeros que aparecen deben coincidir con la imagen origianl en otro caso se debera repetir la captura de la imagen.

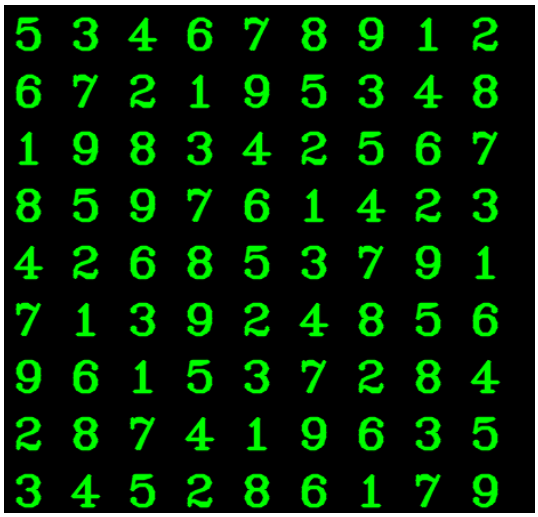


4.5 Busqueda de una solucion

En este paso se va a realizar la búsqueda de una solución para nuestro sudoku, para ello utilizaremos el archivo sudokuSolver.py mencionado en el punto dos. A continuación, debemos colocar los dígitos encontrados en una matriz de la forma qué se nos piden en el archivo sudokuSolver. Puede ocurrir a veces un tipo de fallo y lo que debemos hacer es comprobar que se correspondan los números de la imagen y en caso de que sea erróneo o falte al menos uno, debemos volver a capturar la imagen.

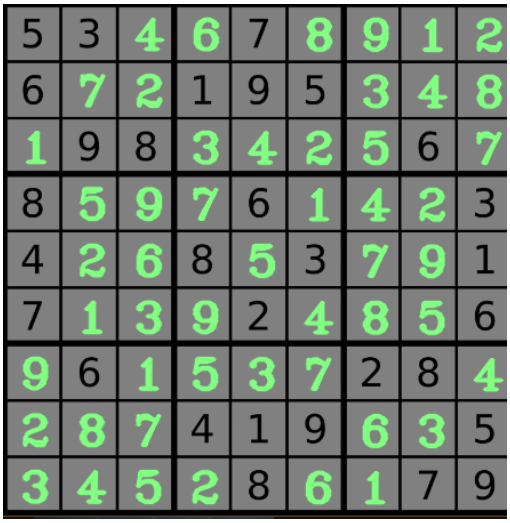
```
sudoku sin resolver
[array([5, 3, 0, 0, 7, 0, 0, 0, 0], dtype=int64), array([6, 0, 0, 1, 9, 5, 0, 0, 0], dtype=int64),
Sudoku resuleto
[array([5, 3, 4, 6, 7, 8, 9, 1, 2], dtype=int64), array([6, 7, 2, 1, 9, 5, 3, 4, 8], dtype=int64),
```

En el paso anterior habíamos creado una matriz de posiciones donde nos mostraba la imagen de los dígitos encontrados sobre esa misma matriz podemos colocar ahora los números con la solución.



4.6 Sobreponer la solución en la imagen

Último paso se encargará de sobreponer la solución del sudoku sobre los espacios vacíos de la imagen original. Para ello en el archivo de utilidades, tenemos una función que nos ayudará a escalar dentro de la imagen del sudoku la solución correspondiente a esas posiciones. A su vez y también es opcional se podrá mostrar cuál fue la respectiva eficiencia en dibujar dentro de esa misma cuadrícula.



5. Conclusión

La práctica de “sodoku” nos ha servido de mucha utilidad ya que hemos aprendido resolver un problema como es el sudoku a partir de la misma imagen.

Concluimos que con ello podemos imaginarnos un sinfín de problemas cotidianos que podríamos solucionar de una manera similar, gracias a lo que hemos dado en clase e información obtenida de distintos foros, hemos podido aprender de este problema de la forma más didáctica posible con el fin de poder utilizar estas herramientas en un futuro.

6. Anexo

Tim Ruscica

Tema 2. Percepción Computacional

SUDOKU

Murtaza Hassan