BGU

Sudoku Solver

ICBV 2023 - Final Project

Roey Lange
Barak Daniel

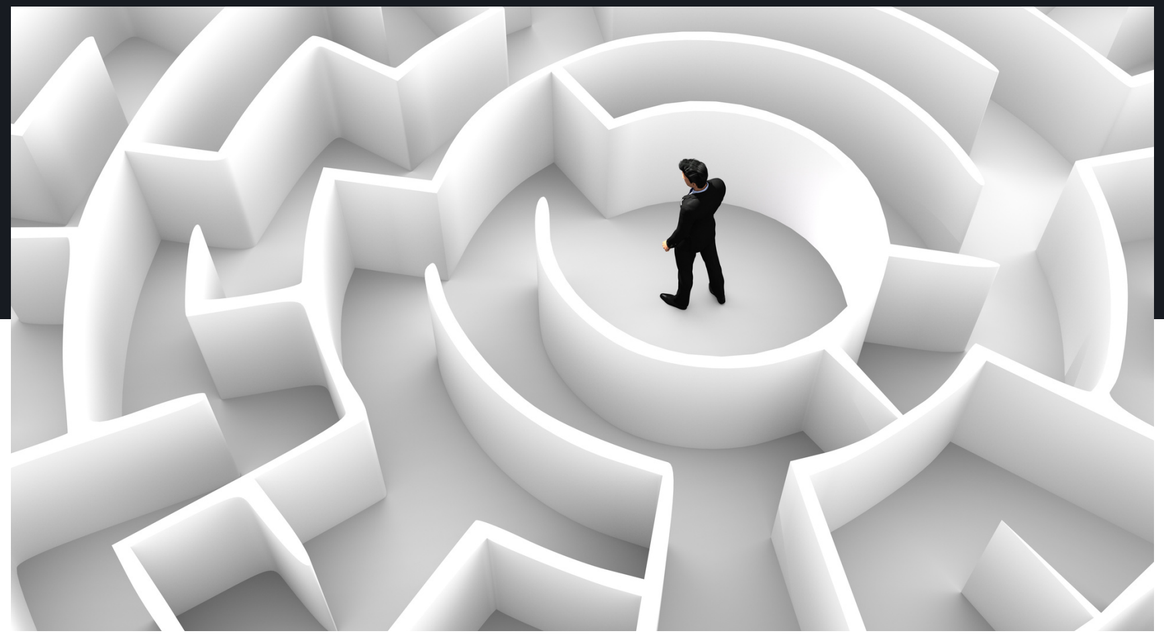
Introduction

- An image based Sudoku solver that uses CV techniques to identify and solve puzzles
- Built using Python on Google Colab
- Uses OpenCV, Pytesseract and an external OCR model and solver
- Capable of detecting Sudoku puzzles from a variety of angles and rotations

The Problem

- Sudoku is a logic based puzzle with a 9x9 grid of cells
- Many sudoku puzzles are published without a solution, making it difficult to verify the solution
- Our team set out to create an automated solver using CV techniques
- The goal of the project was to develop a program that could extract and solve a puzzle from an image

The Challenges



- Detecting the sudoku grid in various orientations and tilts
- Identifying and isolating individuals cells within the grid
- Dealing with variations in lighting, contrast and colors of input images
- Removing noise and artifacts to improve grid detections
- Difficulty in identifying the correct orientations of the numbers within the grid (0, 90, 180 or 270 degree angle)

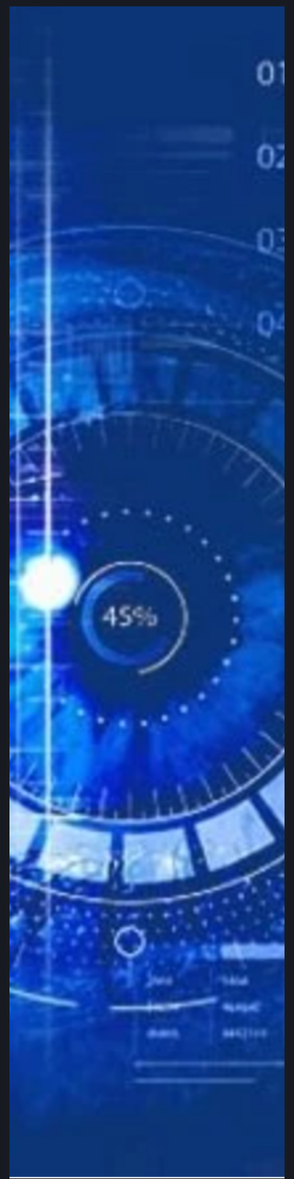
Approach

01 **Image Processing:** Conversion to Greyscale and Gaussian blurring - simplify the image and reduce the noise

02 **Edge Detection:** Find a group of points that represent a wide range of edges (Canny's algorithm)

03 **Contour Detection:** Find the boundaries of all objects in the image grid of the puzzle using its geometrical features

04 **External Frame Detection:** Find the grid of the puzzle using its geometrical features (cv2.approxPolyDP)



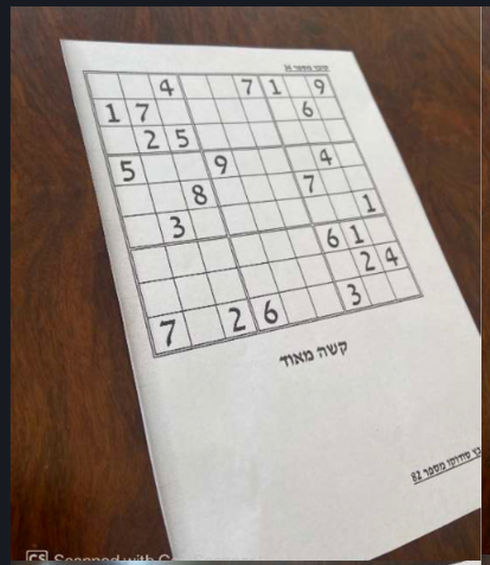
05 **Perspective Transformation:** Transform the image to a 450x450 pixel frame

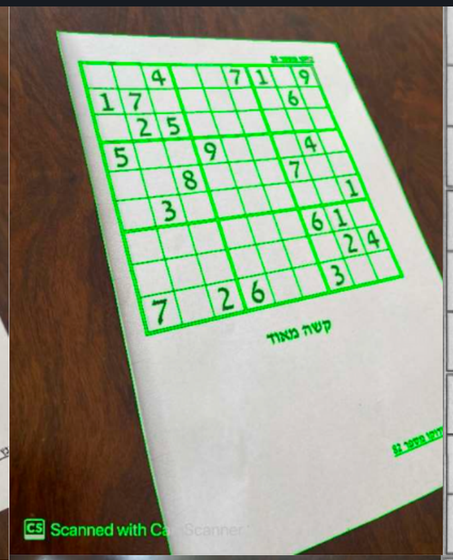
06 **Image Rotation:** Rotate the image so all numbers are aligned with the default axes

07 **Digit inferring:** Split the image into cells (50x50 each), predicting content using OCR model

08 **Puzzle Solver:** Use external code to solve the puzzle

Examples

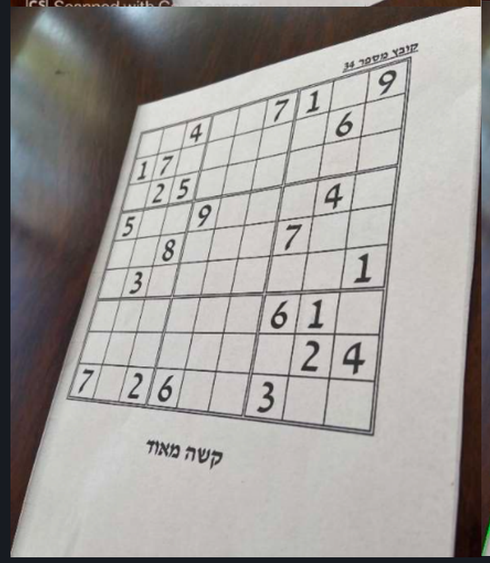


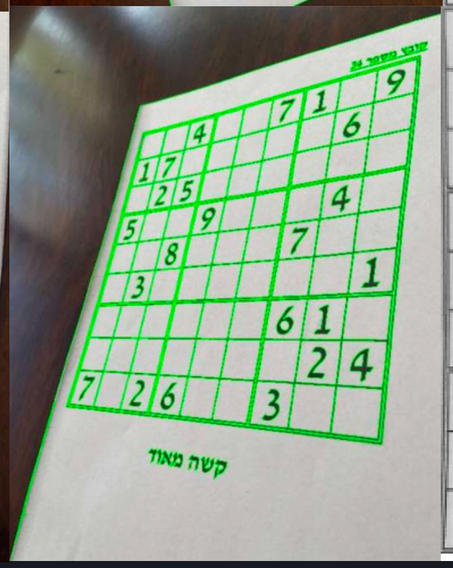


		4			7	1		9
1	7						6	
	2	5						
5			9				4	
		8				7		
	3							1
						6	1	
							2	4
7		2	6			3		

```
[[0 0 4 0 0 7 1 0 9]
 [1 7 0 0 0 0 0 6 0]
 [0 2 5 0 0 0 0 0 0]
 [5 0 0 9 0 0 0 4 0]
 [0 0 8 0 0 0 7 0 0]
 [0 3 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 6 1 0]
 [0 0 0 0 0 0 0 2 4]
 [7 0 2 6 0 0 3 0 0]]
```

```
| 3 8 4 | 2 6 7 | 1 5 9 |
| 1 7 9 | 3 5 8 | 4 6 2 |
| 6 2 5 | 1 4 9 | 8 7 3 |
+-------+-------+-------+
| 5 1 6 | 9 7 3 | 2 4 8 |
| 4 9 8 | 5 1 2 | 7 3 6 |
| 2 3 7 | 4 8 6 | 5 9 1 |
+-------+-------+-------+
| 9 5 3 | 8 2 4 | 6 1 7 |
| 8 6 1 | 7 3 5 | 9 2 4 |
| 7 4 2 | 6 9 1 | 3 8 5 |
+-------+-------+-------+
```



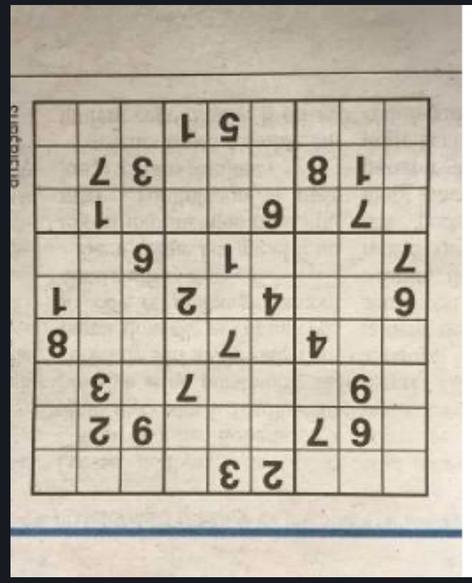


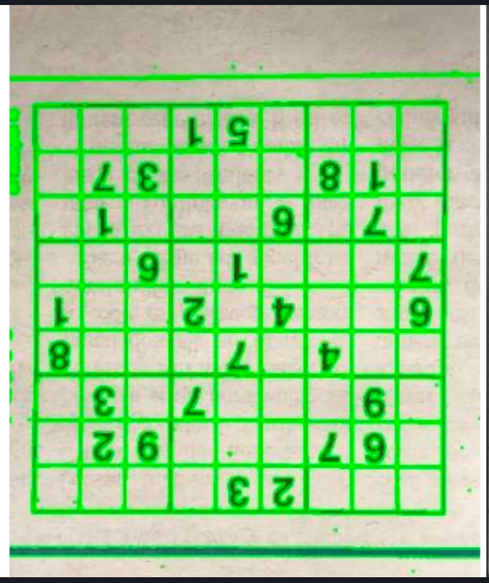
		4			7	1		9
1	7						6	
	2	5						
5			9				4	
		8				7		
	3							1
						6	1	
							2	4
7		2	6			3		

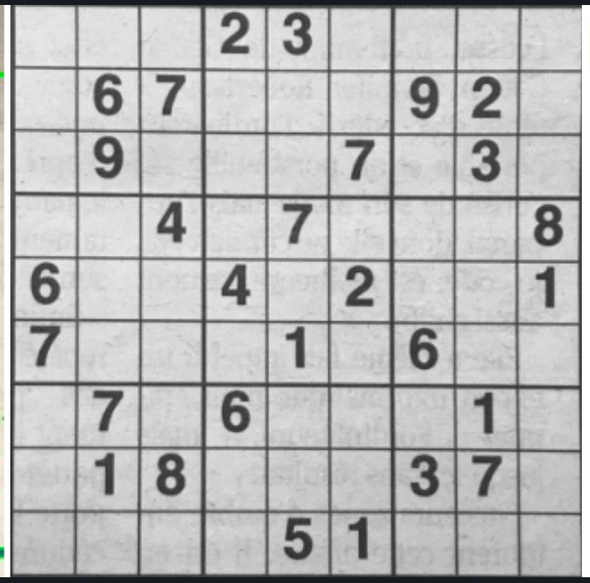
```
[[0 0 4 0 0 7 1 0 9]
 [1 7 0 0 0 0 0 6 0]
 [0 2 5 0 0 0 0 0 0]
 [5 0 0 9 0 0 0 4 0]
 [0 0 8 0 0 0 7 0 0]
 [0 3 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 6 1 0]
 [0 0 0 0 0 0 0 2 4]
 [7 0 2 6 0 0 3 0 0]]
```

```
| 3 8 4 | 2 6 7 | 1 5 9 |
| 1 7 9 | 3 5 8 | 4 6 2 |
| 6 2 5 | 1 4 9 | 8 7 3 |
+-------+-------+-------+
| 5 1 6 | 9 7 3 | 2 4 8 |
| 4 9 8 | 5 1 2 | 7 3 6 |
| 2 3 7 | 4 8 6 | 5 9 1 |
+-------+-------+-------+
| 9 5 3 | 8 2 4 | 6 1 7 |
| 8 6 1 | 7 3 5 | 9 2 4 |
| 7 4 2 | 6 9 1 | 3 8 5 |
+-------+-------+-------+
```

Examples







```
[[0 0 0 2 3 0 0 0 0]
 [0 6 7 0 0 0 9 2 0]
 [0 9 0 0 0 7 0 3 0]
 [0 0 4 0 7 0 0 0 8]
 [6 0 0 4 0 2 0 0 1]
 [7 0 0 0 1 0 6 0 0]
 [0 7 0 6 0 0 0 1 0]
 [0 1 8 0 0 0 3 7 0]
 [0 0 0 0 5 1 0 0 0]]
```

```
+-----------+-----------+-----------+
| 8 4 5 | 2 3 9 | 1 6 7 |
| 3 6 7 | 1 4 8 | 9 2 5 |
| 2 9 1 | 5 6 7 | 8 3 4 |
+-----------+-----------+-----------+
| 1 5 4 | 3 7 6 | 2 9 8 |
| 6 8 3 | 4 9 2 | 7 5 1 |
| 7 2 9 | 8 1 5 | 6 4 3 |
+-----------+-----------+-----------+
| 4 7 2 | 6 8 3 | 5 1 9 |
| 5 1 8 | 9 2 4 | 3 7 6 |
| 9 3 6 | 7 5 1 | 4 8 2 |
+-----------+-----------+-----------+
```

Questions