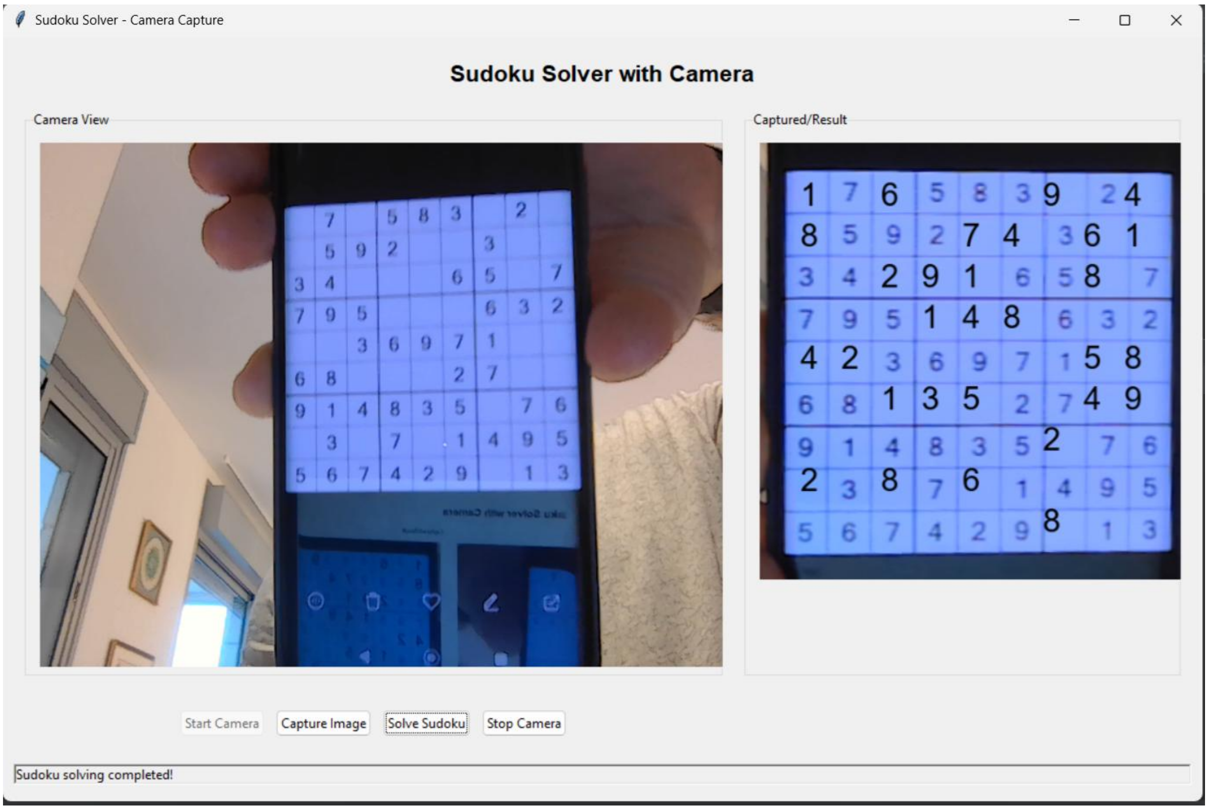
Sudoku Solver

A desktop application that captures Sudoku puzzles using your computer's camera and solves them automatically.

Features a full backtracking-based solver that can handle any valid Sudoku configuration.

Final result



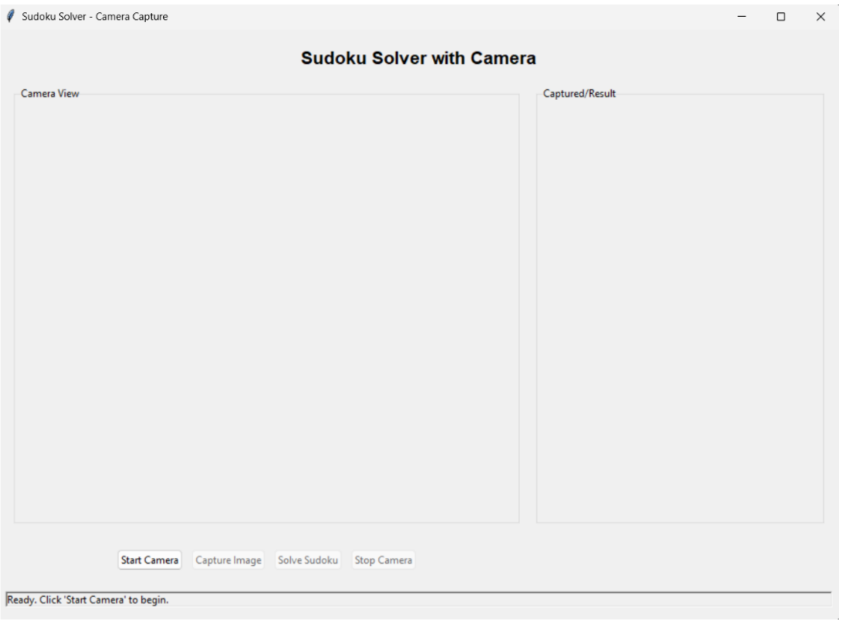
Sudoku Solver with Camera

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

Sudoku solving completed!

sudoku_camera_app

1. Start by running sudoku_camera_app.py



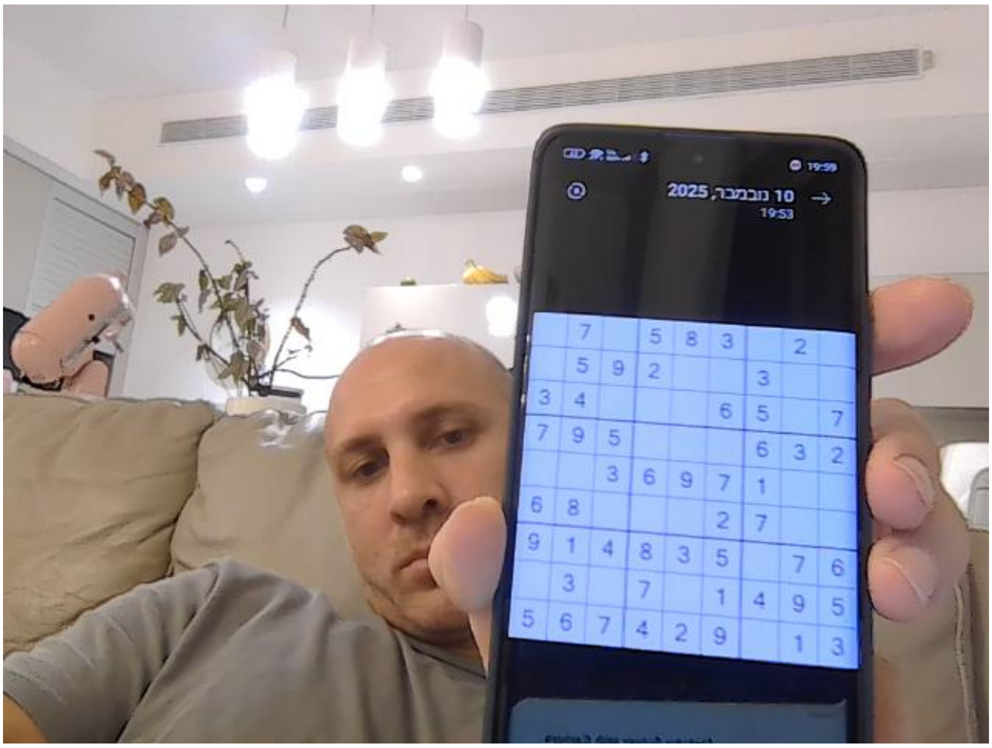
Ready. Click 'Start Camera' to begin.

2. You have to press “Start Camera”. This enables the camera and a video view will be presented at the left side.
3. Bring the sudoku board in front of the camera. When it’s fully visible press “capture Image”. Please make sure the board is clean without any obscures.
4. The captured image would be visible at the right side. If you’re please with the image, press “Solve Sudoku” and the sudoku puzzle hopefully be solved.

Sudoku Puzzle Capture and Extraction Process

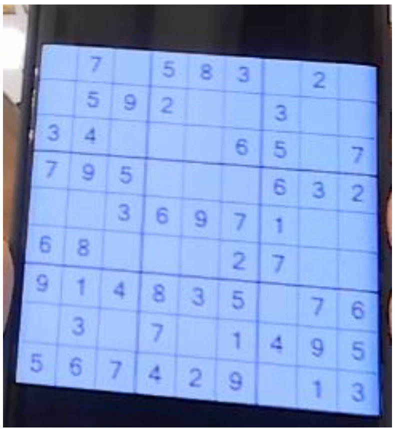
1. Image Capture

Follow the instructions and capture a clear picture of the Sudoku puzzle, ensuring the entire puzzle is visible without reflections or obstructions.



2. Preprocessing

The app identifies the puzzle boundaries and extracts the region of interest.



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

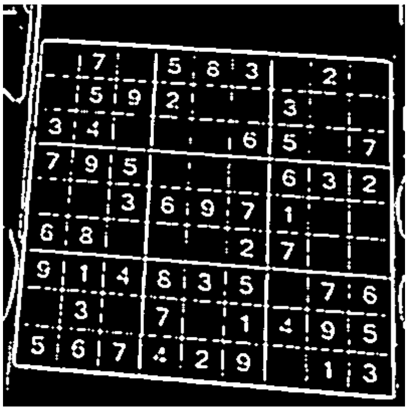
It then:

- Converts the image to grayscale

- Resizes it

- Applies thresholding

- Uses morphological operations to clean up noise



3. Grid Detection

To detect the Sudoku grid:

- Apply the Canny edge detector

- Use the Hough Transform to find long lines

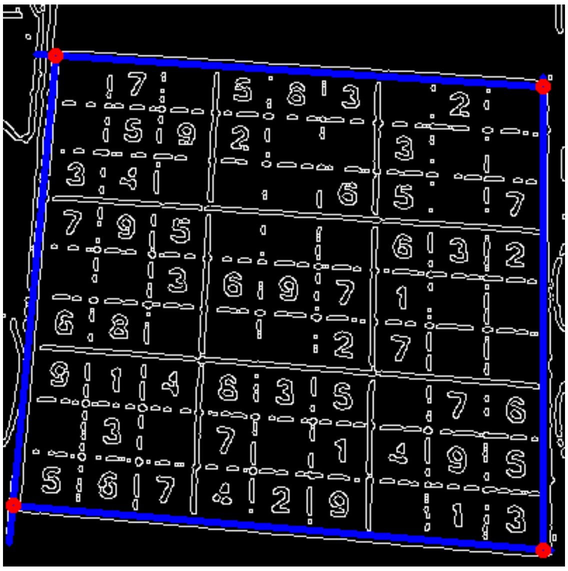
- Classify lines as either vertical or horizontal

- Filter lines by length, remove duplicates, and validate angle consistency

- Calculate spacing between lines and use a “missing lines” algorithm to reconstruct any undetected lines

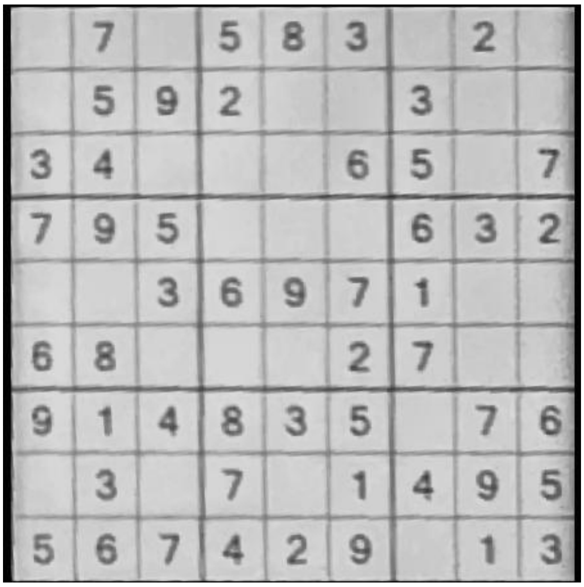
- Detect the puzzle’s outer boundaries and compute the corner points

Finally, extract the puzzle’s external boundaries and calculate their contact points (the corners).

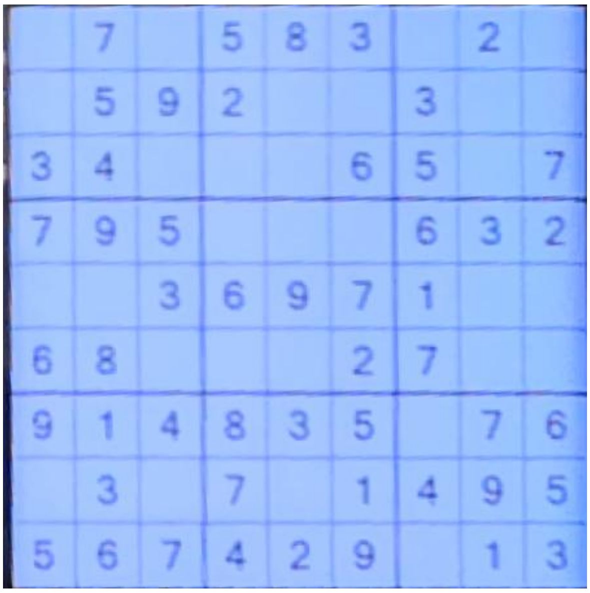


4. Image Alignment

Using the corners, apply a perspective transform to align the puzzle for consistent processing.



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



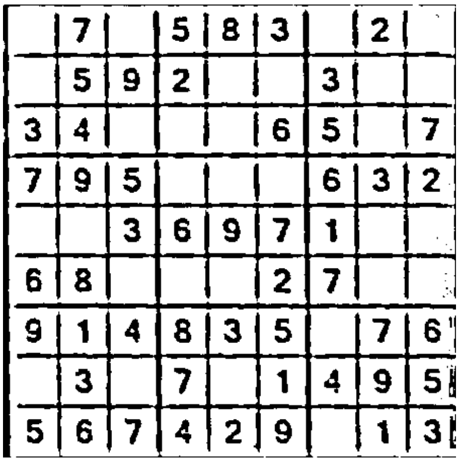
	7		5	8	3		2	
	5	9	2			3		
3	4				6	5		7
7	9	5				6	3	2
		3	6	9	7	1		
6	8				2	7		
9	1	4	8	3	5		7	6
	3		7		1	4	9	5
5	6	7	4	2	9		1	3

5. Image Cleaning

Enhance contrast and remove isolated noise. This step prevents common issues such as faded digits or broken lines, which can impact recognition.

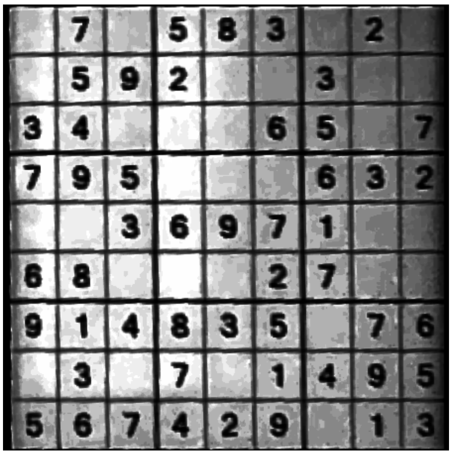
Without doing so, I may suffer from the following cases:

isolated noise:



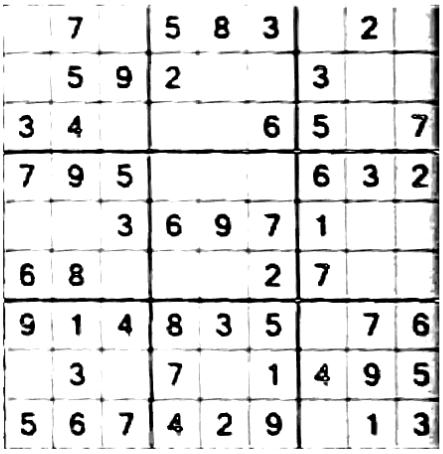
	7		5	8	3		2	
	5	9	2			3		
3	4				6	5		7
7	9	5				6	3	2
		3	6	9	7	1		
6	8				2	7		
9	1	4	8	3	5		7	6
	3		7		1	4	9	5
5	6	7	4	2	9		1	3

bad color equalization



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

no boundaries fix



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

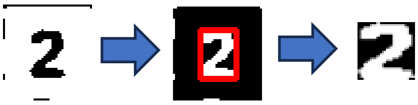
6. Cell Segmentation & Digit Detection

- Split the cleaned puzzle into 9×9 cells

- Remove cell borders

- Identify which cells contain digits

- Extract and resize each digit to 28×28 pixels



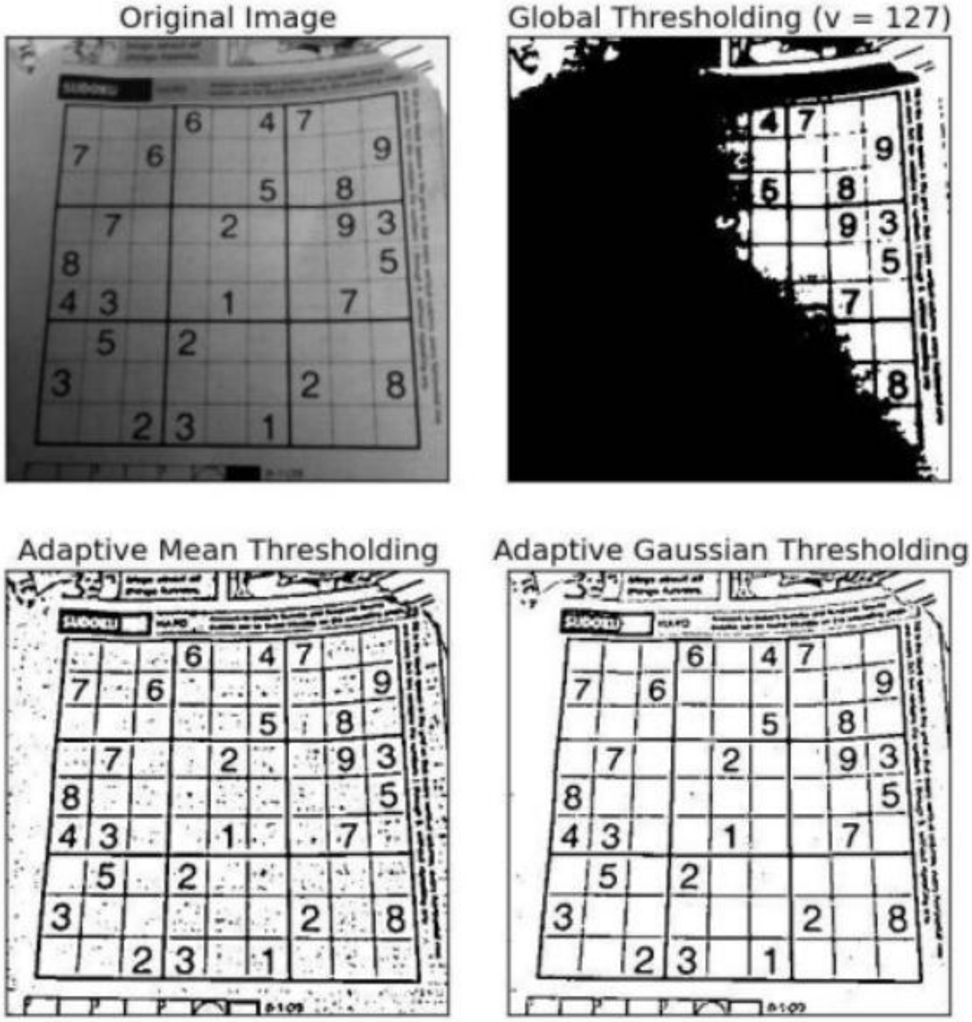
A validation step checks the digit height to correct cases where only part of a digit was initially extracted.

8. Digit Recognition

A trained neural network classifier is applied to each processed digit image to recognize the number. The final result is a complete digital reconstruction of the Sudoku board.

Anecdote:

In OpenCV documentation regarding thresholding OpenCV: Image Thresholding they use the sudoku board for an example to explain the differences between the different thresholding.



Original Image

Global Thresholding (v = 127)

Adaptive Mean Thresholding

Adaptive Gaussian Thresholding

image