EE5005 Digital Image Processing

IIT PALAKKAD

MINI PROJECT

Project Topic:

Automatic Sudoku Solver

Submitted By

Arun Kumar V - 122001049

Vandana P - 122001045

Contents

1 Aim

The aim is to develop a comprehensive image processing and computer vision framework that can robustly identify the Sudoku grid and its digits from a given image, with the ultimate goal of solving it through computational techniques.

2 Introduction

Sudoku, a popular puzzle game, has transcended its origins as a pen-and-paper pastime to find itself in the digital age. As technology advances, the integration of image processing and computer vision techniques has opened up new possibilities, including the extraction of Sudoku puzzles directly from photographs. This technological innovation links the physical and digital domains, facilitating modern interaction with Sudoku puzzles.
The comprehensive image processing framework we aim to develop aligns seamlessly with this evolution, enhancing the accessibility and engagement with Sudoku by offering a dynamic and technologically advanced solving experience. This involves accurately identifying both the Sudoku grid and its constituent digits within a given image.

3 Methodology and Results

The input image (Figure 1) for the project is a photo of unsolved sudoku from a printed paper. The image may contain words or pictures beyond the puzzle grid. The primary task is to extract the complete Sudoku grid from the picture, which essentially comprises multiple lines. These lines can be extracted using the Hough transform after proper image processing steps. Furthermore, the task involves the identification and resolution of digits within the Sudoku grids, accomplished through the utilization of KNN to correlate image digits with their respective actual values. Once the digit identification process is refined, a combination of computer vision and algorithmic techniques is employed to successfully resolve the Sudoku puzzle.[1]

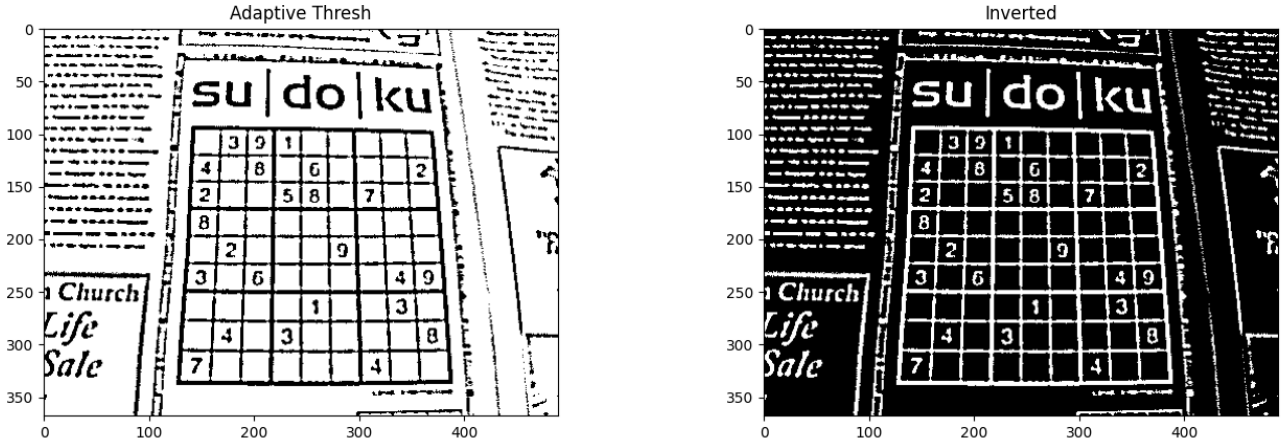
Figure 1: Input Sudoku Image

Before extracting information, the image undergoes preprocessing steps to enhance its quality. This includes operations such as resizing, noise reduction, and contrast adjustment to optimize the image for subsequent analyses. The whole process involves the following key steps:

- Gaussian Blurring: A Gaussian filter is applied to the input image to reduce high-frequency noise, smoothing it while preserving the overall structure and important features. This removes any noise or fine details if present in the image, resulting in a cleaner representation. Additionally, Gaussian blurring enhances the contrast between key features, making it easier to identify edges and boundaries. By blurring the image, we create a more refined and simplified representation, setting the stage for subsequent processing steps (Figure 2).

Figure 2: Gaussian Blurred Image

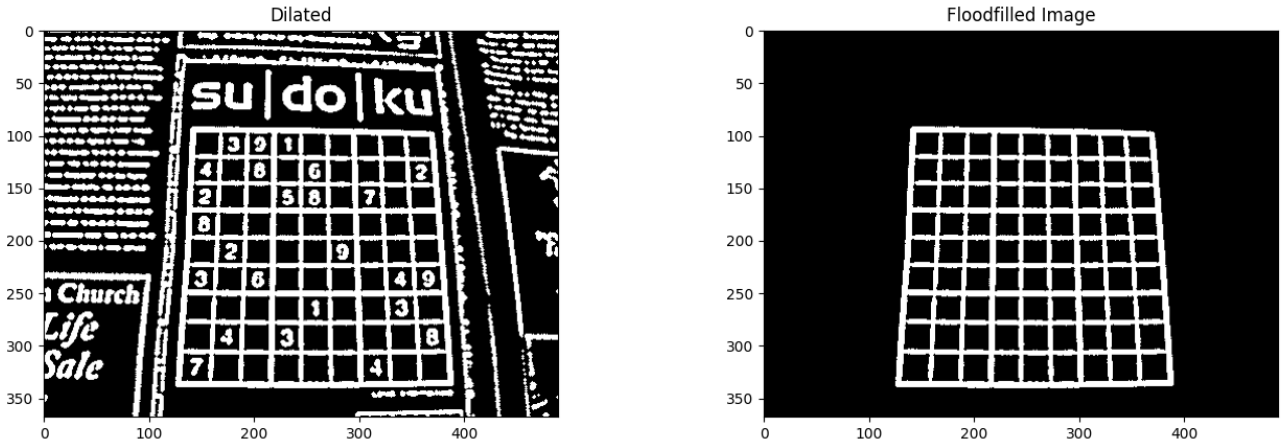
- Adaptive Gaussian Thresholding: Adaptive thresholding is applied to make sure the image is illuminated properly. It is chosen rather than normal thresholding because it can handle images with varying illumination conditions, and it leads to better results in the presence of noise or uneven lighting [2]. Adaptive thresholding considers local information in the image, and different threshold values are used for different regions, taking into account variations in illumination across the image (Figure 3a).



(a) (b)

Figure 3: (a) Image after thresholding (b) Image after inverting

- Negative: The negative of the image is then taken to make the digits and lines white while making the background black for easy processing in the further steps (Figure 3b).
- Dilation: Dilation adds pixels to the boundaries of objects in an image. The inverted image is dilated with a plus-shaped 3×3 Kernel to fill out any cracks in the board lines and thicken the board lines (Figure 4a). This is done to prepare for the extraction of the biggest blob from the grid (the borders of the sudoku grid).
- Flood Filling: Flood filling from various seed points and identifying all connected components, followed by determining the largest flooded area region, will likely reveal the board as the largest connected component with the greatest area [3]. The outer box of the sudoku grid is obtained after this process (Figure 4b).



(a) (b)

Figure 4: (a) Image after Dilation (b) Image after Floodfilling

- Erosion: Applying a slight erosion to the grid is done to counteract the impact of the earlier dilation on the outer box. It entails the gradual removal of pixels from the grid's boundaries (Figure 5). This adjustment ensures that the overall structure and integrity of the Sudoku grid are maintained, preventing potential distortions or inaccuracies resulting from the initial dilation process[4].

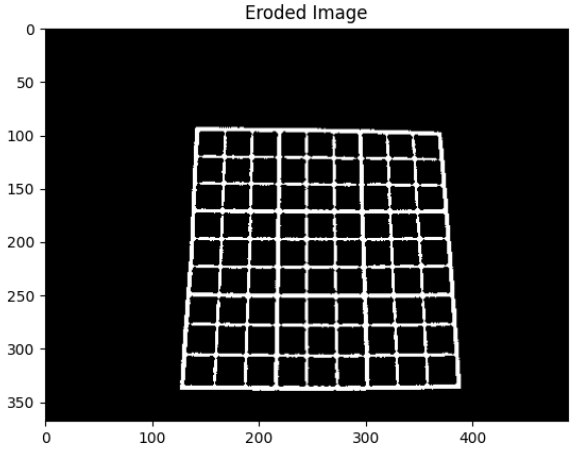
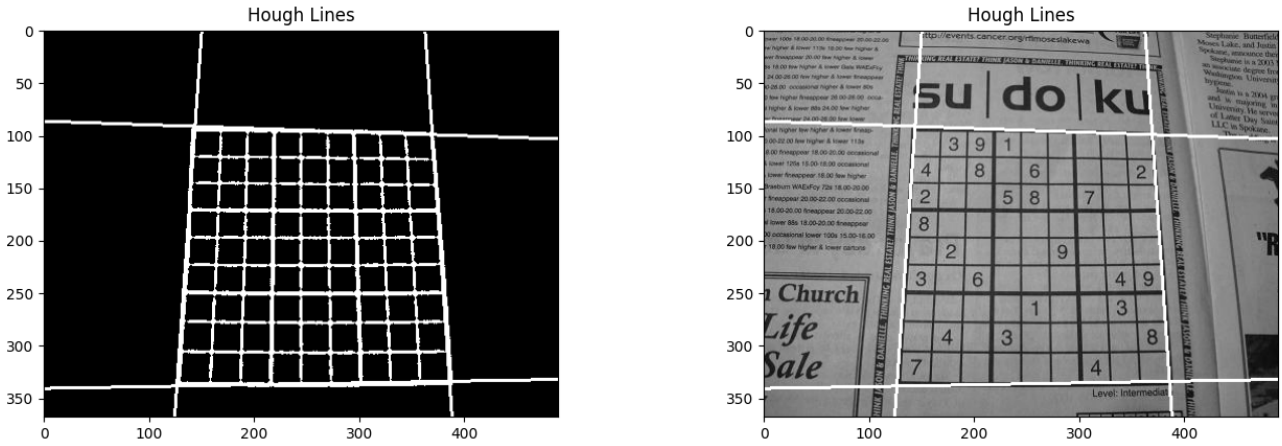


Figure 5: Image after Erosion

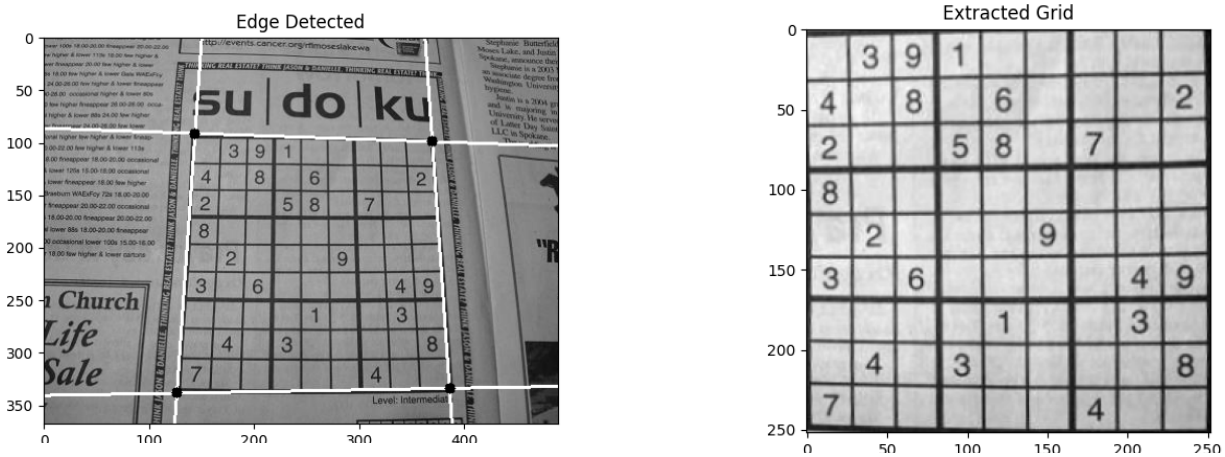
- Hough Line Transform: The Hough Transform is a mathematical technique used in image processing to detect shapes, particularly lines [5]. This is applied to the outer box to find all the lines in the detected grid. By applying this technique to the outer box, we effectively capture and delineate all the lines comprising the Sudoku grid.
- Merging: Lines identified through the Hough Transform, which are close, are merged together.
- Extreme border extraction: Border lines are determined by selecting the nearest line from the top with an almost 0 slope as the upper edge, the nearest line from the bottom with an almost 0 slope as the lower edge, the nearest line from the left with an almost infinite slope as the left edge, and the nearest line from the right with an almost infinite slope as the right edge (Figure 6a).



(a) (b)

Figure 6: (a) Extreme lines in grid found using Hough Transform (b) The input image with Extreme Lines

- Four intersection points/corner points of sudoku grid: From the four extreme edges found, the intersection points were recovered. These intersection points represent the vertices of the Sudoku grid (Figure 7a). After finding these points the grid is extracted from the original image for processing (Figure 7b).



(a) (b)

Figure 7: (a) Image showing 4 corners detected (b) Cropped Sudoku grid image

- Thresholding and inverting: The cropped image from the previous step is adaptive thresholded and inverted.
- Slicing: The whole grid is divided into 81 slices to get images of each cell of the Sudoku board.
- Backfilling and centering of the number: Any white regions apart from the number are eliminated by floodfilling from the outermost points as seed points. Subsequently, the estimated bounding box of the number is determined and centered within the image (Figure 8).

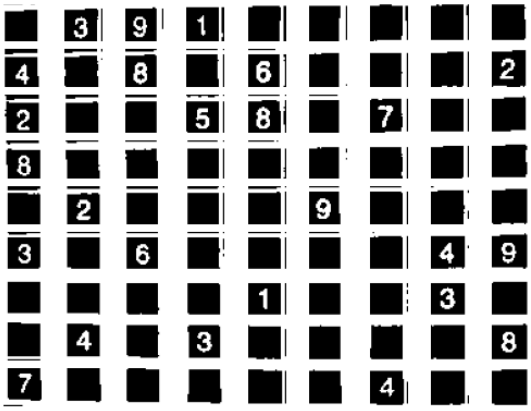


Figure 8: Numbers recovered after slicing and backfilling

- Digit Recognition: The isolated grid was recovered and the next step involves recognizing the individual digits within each cell. Some of the commonly used methods are the Optical Character Recognition (OCR) algorithm or machine learning models, Convolutional Neural Networks (CNN), K-Nearest Neighbours (KNN) etc. Here in this project, we make use of the KNN algorithm with a training dataset of 60000 handwritten digits with 50000 images for training and 10000 images for testing. The testing accuracy was found to be 0.98 (Figure 9).

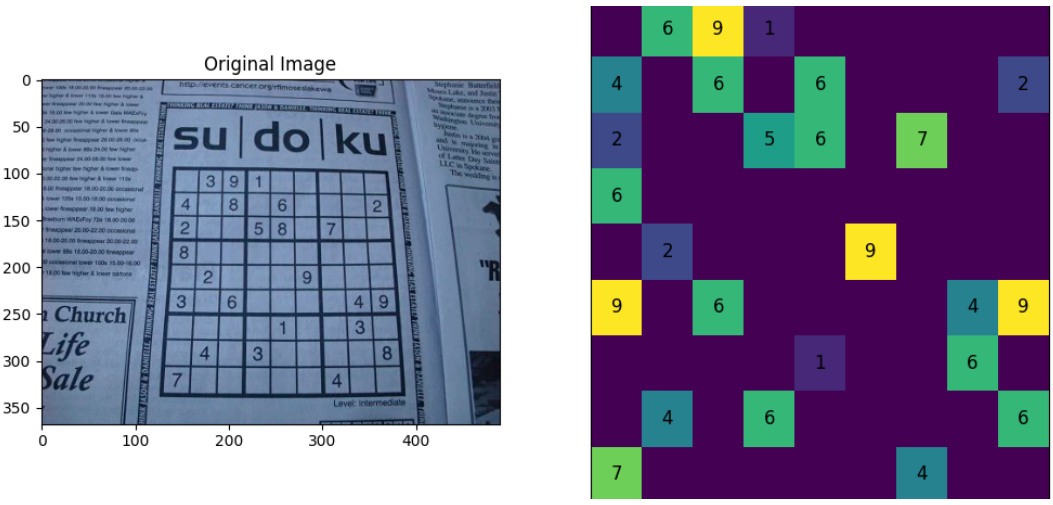


Figure 9: Digits recovered using KNN algorithm

As we can see, there is some error when the digits are 3,6,8. Which user is given a choice to correct the numbers recovered if not correct.

- Solving Algorithm: Once the numbers recovered were corrected and finalized a solving algorithm was used which implements a recursive backtracking algorithm to solve the 9x9 Sudoku puzzle. It iterates through each cell of the puzzle, attempting to place digits from 1 to 9, while ensuring the placement adheres to Sudoku rules. If a valid digit is found, the algorithm continues to the next empty cell. If the digit placement leads to an unsolvable puzzle, it backtracks to the previous cell and explores alternative digit choices. This process is repeated until a solution for the entire puzzle is found or all possibilities are exhausted.
- Output Display: Finally, the solved Sudoku grid is then displayed in the screen (Figure 10).

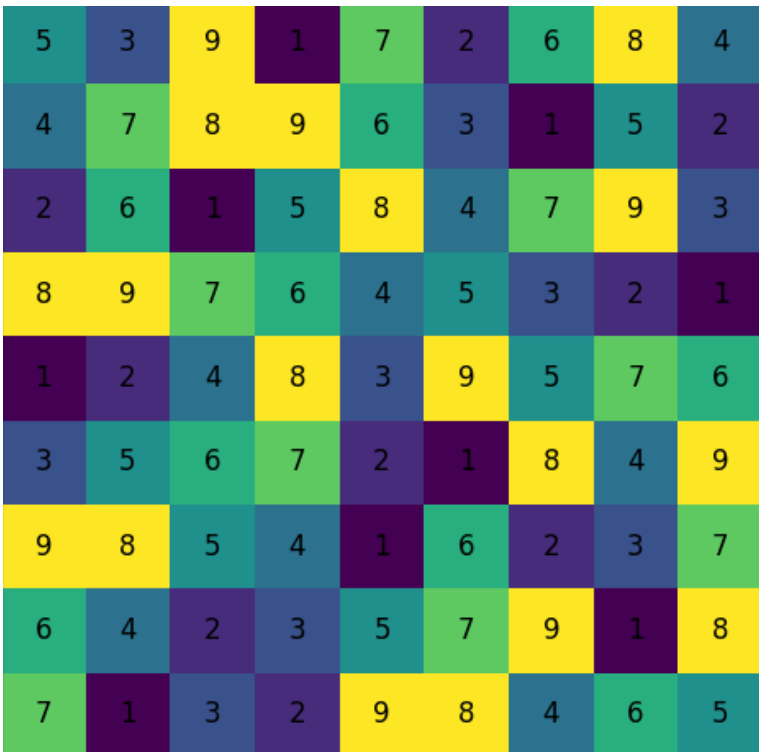


Figure 10: Final solved sudoku grid

4 Conclusion & Future Scope

In conclusion, we were able to successfully solve a Sudoku puzzle extracted from a normal newspaper cutting using a basic image processing algorithm and a Sudoku solving algorithm.

The future scope of this work includes,

- Tuning the proper parameters for image processing techniques for better perfomance.
- Train the Neural Network model more precisely for proper detection of digits.
- Develop a Graphic User Interface (GUI) that can enable user to interact with the algorithm developed more easily.

5 Source Code & Video Link

AUTOMATIC SUDOKU SOLVER

References

[1] Neeramitra Reddy, "GUI based Smart Sudoku Solver that tries to extract a sudoku puzzle from a photo and solve it." https://github.com/neeru1207/AI_Sudoku. Last accessed 7 December 2023.

[2] OpenCV-Image Thresholding, "OpenCV - Adaptive Threshold." https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html. Last accessed 7 December 2023.

[3] Flood Fill — skimage 0.22.0 documentationScikit-image, "Flood Fill." https://scikit-image.org/docs/stable/auto_examples/segmentation/plot_floodfill.html. Last accessed 7 December 2023.

[4] Wikipedia, "Erosion." https://en.wikipedia.org/wiki/Erosion_(morphology)#:~:text=Erosion%20(usually%20represented%20by%20%E2%8A%96,and%20subsequently%20to%20complete%20lattices. Last accessed 7 December 2023.

[5] OpenCV-Python Tutorial, "OpenCV-Python Tutorials on Hough Line Transform." https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html. Last accessed 7 December 2023.