2019

Vision-based Sudoku Solver Report

GROUP 62

NIMISH MAGRE – U6434979 | KAUSHIK GHOSH – U6425134 |
HARPRAGAAS SINGH – U6424131 | LAKSHMI MEDICHERLA – U6420703

Table of Contents

List of Figures

1. Introduction

A Sudoku puzzle is a well-known logical number-based puzzle. For a 9x9 puzzle grid, the puzzle grid is partially filled with certain integers between 1 to 9 at certain locations. To solve the puzzle, integers between 1 to 9 need to be allocated at each empty grid location such that:

- The integer is unique with respect to all the integers in the same row
- The integer is unique with respect to all the integers in the same column
- The integer is unique with respect to all the integers in the 3x3 local grid

The overall project goal is to solve a sudoku puzzle, which is printed on a page, using computer vision techniques. The process reflects the concept of human-computer interaction because the information gained is not only obtained by computers but also enhanced with human activities.

2. Related Work

Vision-based sudoku solving involves multiple processes to capture the puzzle image, detect the puzzle grid, identify the digits in the puzzle and to solve the puzzle. The task itself is not completely novel and a few successful attempts have been made to solve sudoku puzzles based on computer vision techniques. Important areas of research involve the specific tasks of puzzle grid detection, digit recognition and sudoku solving.

An important task to perform before digit recognition and sudoku solving, has been the process of image grid detection. Normalized Direct Linear Transforms [1] have popularly been used to perform 2D Homography so that the puzzle image can be transformed into a 2-d standard planar form from the original plane in which the image is captured. W. Baptiste et al. [2] in 2014 conducted research specifically for segmenting and detecting the digits present in an unsolved sudoku puzzle. The authors used mobile phone images of sudoku puzzles, Hough Transform for puzzle grid detection and a Deep Belief Network (DBN) based on their own training and testing dataset to recognize the digits. The authors were able to obtain an 87.5% success rate on their test dataset and obtain the results for any sudoku puzzle within 100 ms. Another research from 2015 by K. Snigdha et al. [3], utilized the Optical Character Recognition (OCR) and compared three methods (backtracking, simulated annealing and genetic algorithms) to solve the puzzle. The research concluded with the best performance from simulated annealing and the worst performance from genetic algorithms.

However, recent works in implementing the Modified National Institute of Standards and Technology (MNIST) [4] with the k-Nearest Neighbour algorithm [5] achieving a 96.94% successful digit recognition result and Convolutional Neural Based algorithms [6] achieving a 98.91% recognition result have provided greater confidence in these digits recognition processes.

3. Methodology

The following figure demonstrates the high-level logical flow of the Sudoku solving algorithm developed. Brief explanation of each step of the logical flow has been provided in the following sections.

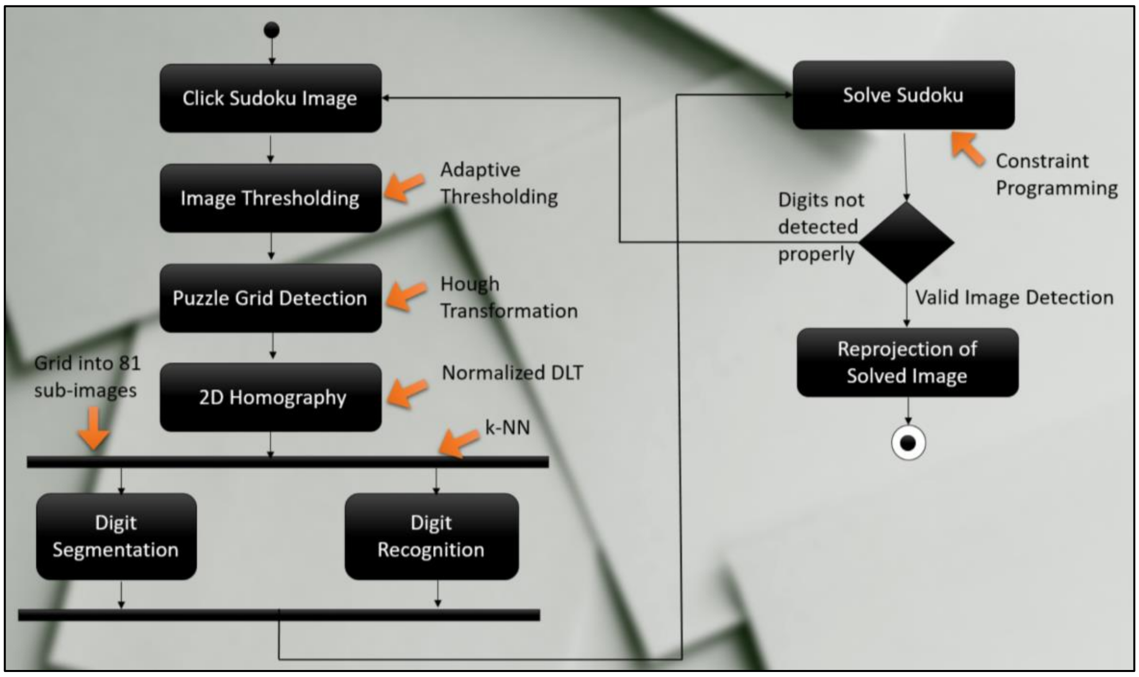


Figure 1: Logical Flow diagram of Sudoku Solver algorithm

3.1 Image Capturing

The sudoku solver main script will access the laptop's camera and will allow the user to select a rectangular region containing the unsolved sudoku puzzle. The selected region will then be scanned and captured to obtain an image of the unsolved sudoku puzzle as shown below:

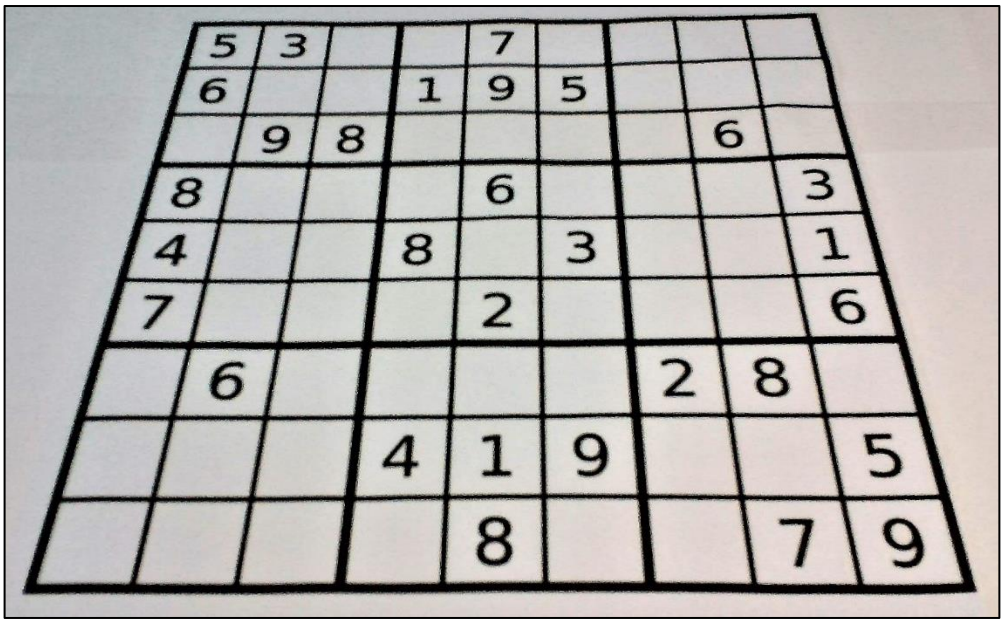


Figure 2: Original Captured Image of unsolved Sudoku Puzzle

3.2 Image Pre-processing

Image pre-processing is an important step because the quality of the image can severely affect the further steps of correct puzzle grid detection and digit recognition.
The image is initially converted from RGB to grayscale and adaptive thresholding is applied. Adaptive thresholding is used to binarize the image and highlight the puzzle from the background that will later be used to detect the puzzle grid.

3.3 Puzzle Grid Detection

After the binarized form of the original image is found from the previous step, it is crucial to detect the puzzle grid for further progress. In order to do so, all the lines in the puzzle are found initially using Hough transforms. Since the puzzle size is 9x9, the total number of lines present in the image are expected to be 20 (10 vertical and 10 horizontal) with equal lengths. For the sudoku example image above, the Hough matrix representation was used to identify the peaks in the parameter space which indicates the possible straight lines present in the image:

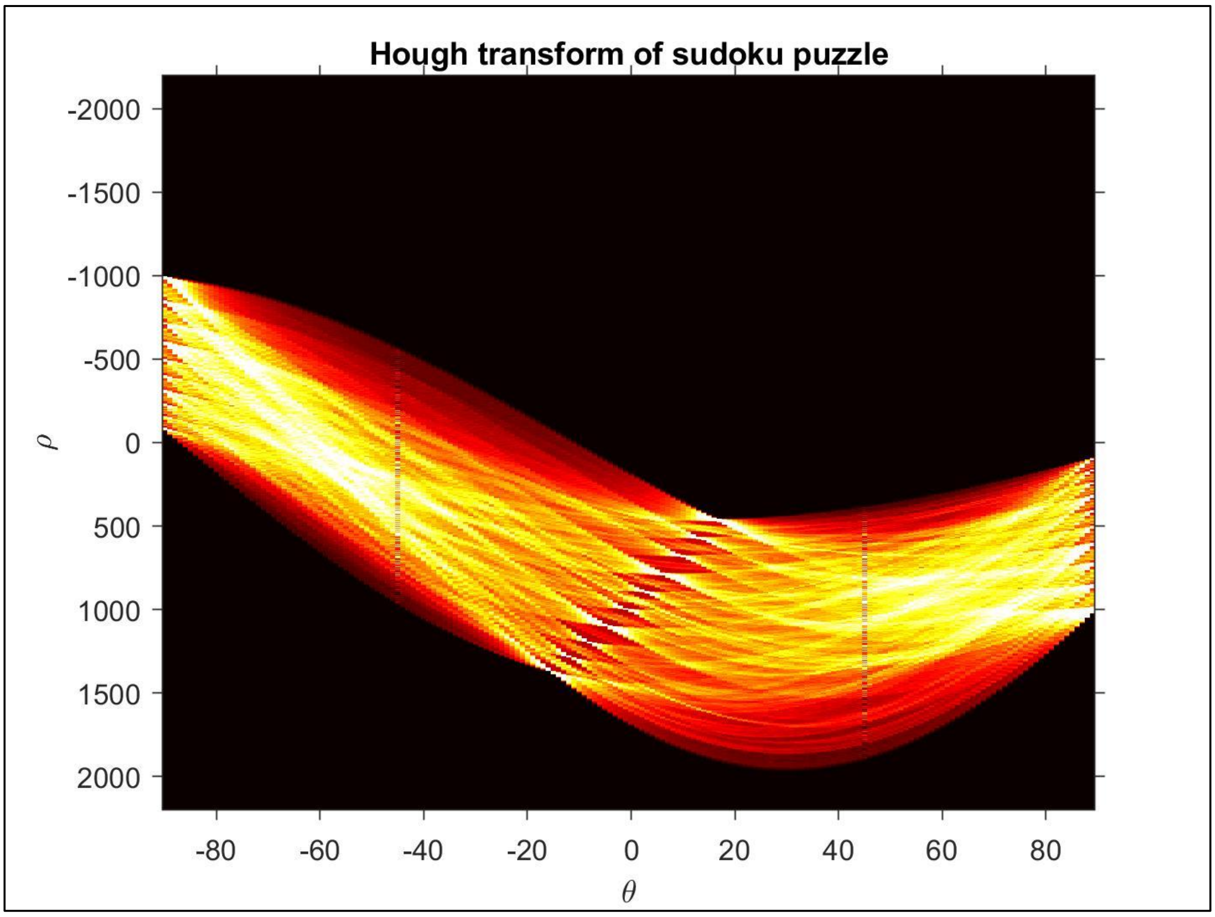


Figure 3: Hough Matrix representation for sudoku puzzle

From the 20 lines identified using the Hough transform, the vertical and horizontal lines are separated using the (θ) values or polar angles of each line. For each of the 2 polar angles (0 or $\frac{\pi}{2}$ $radians$) representing the vertical and horizontal lines, the 4 border lines for the puzzle are found by checking the minimum and the maximum ρ values where ρ is the perpendicular distance from the origin to the line.

Finally, to find the 4 vertices of the puzzle grid, the 4 border lines were first represented in Cartesian coordinate space and then their intersections were found. The corners found on the binarized image for the sudoku puzzle in {***Figure 2:*** Original Captured Image of unsolved Sudoku Puzzleis shown below with red stars:

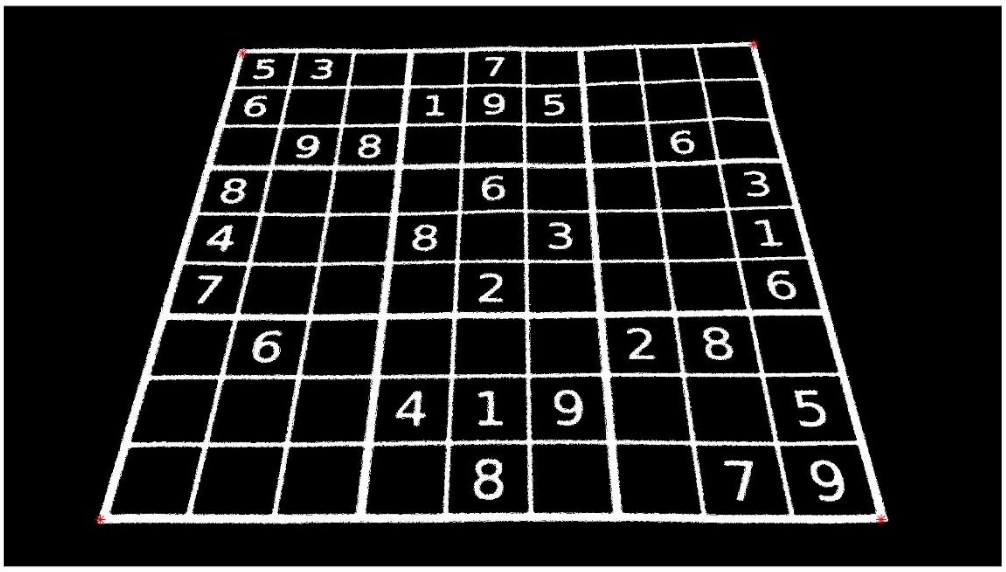


Figure 4: binarized sudoku with corners shown in red stars

3.4 Planar Image Reprojection and Digit Segmentation

In order to ensure the success of the digit recognition process, it is crucial that the puzzle image is reprojected in a standard planar form. In order to carry out the reprojection, the location for the four image vertices or corners in the standard plane is prefixed and the four corners of the original image have been found in the previous tasks. These eight points were then used along with the normalized Direct Linear Transform (DLT) algorithm [1] to obtain the Homography matrix between the original and standard planar image. Based on the calculated Homography matrix, the original image was warped onto the standard plane.

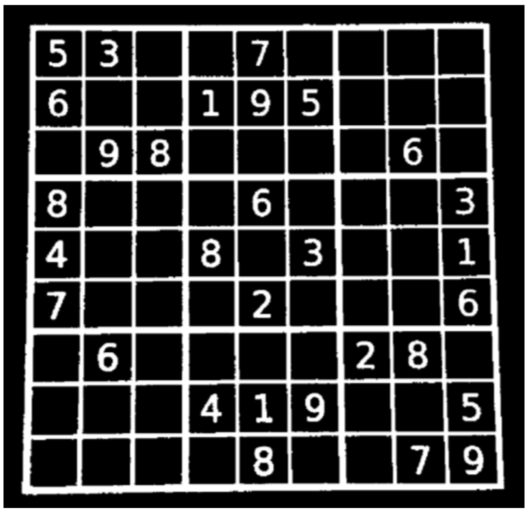


Figure 5: Sudoku Puzzle in 2D Standard Planar form

After obtaining the standard planar image of the puzzle with known vertices coordinates, the top left corner and total side size of the 9 x 9 square puzzle was used to crop the entire image into 81 sub-regions. The digit recognition techniques involved the use of the MNIST dataset which includes the image for handwritten numbers in a 28 x 28 size and therefore, each sub-region was resized to 28 x 28 and binarized.

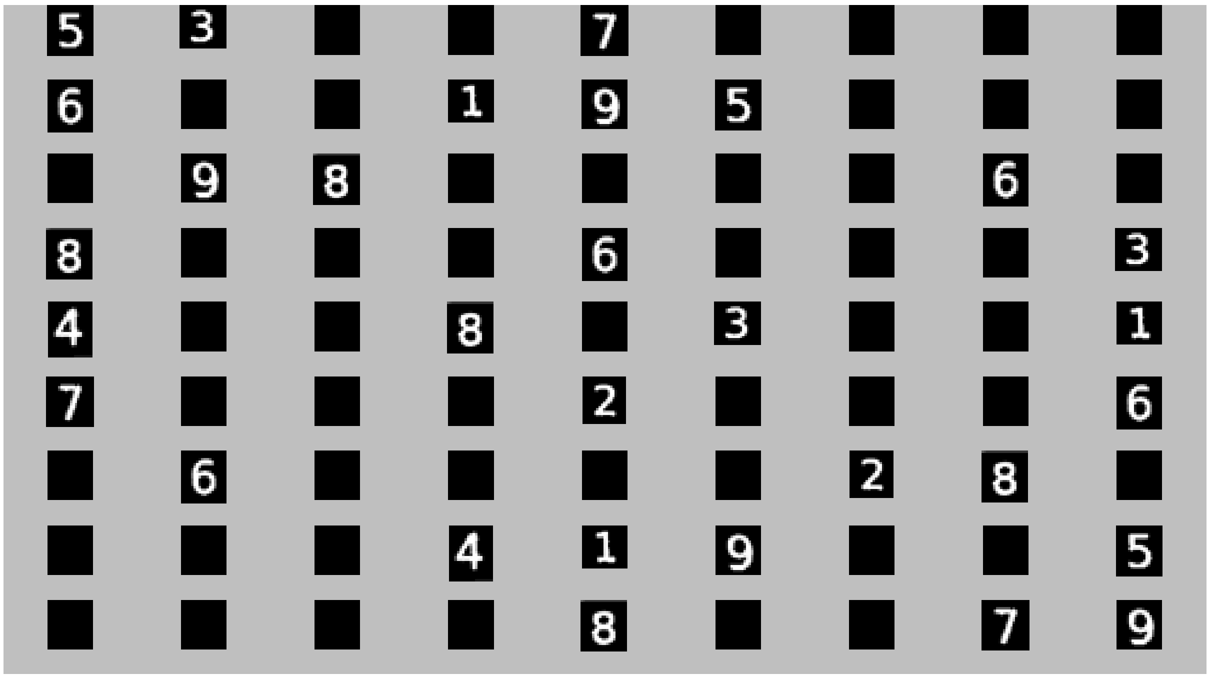


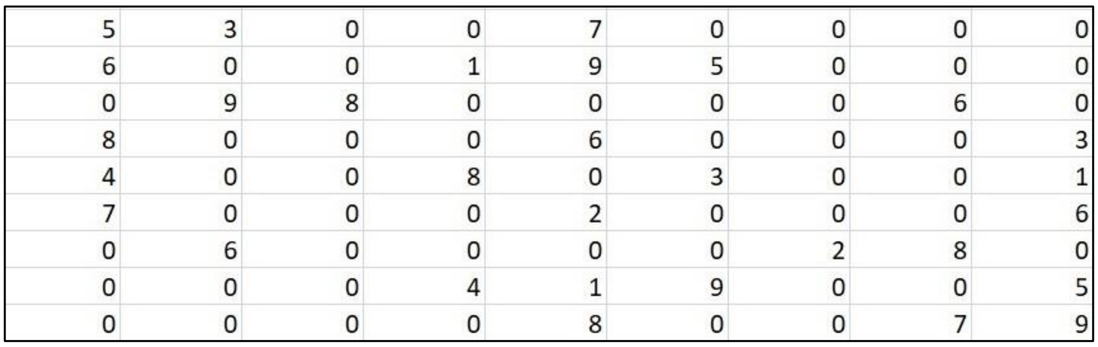
Figure 6: The 81 sub-regions obtained after digit segmentation of the original Sudoku Puzzle

3.5 Digit recognition

The k-NN (k nearest neighbours) algorithm has been used to recognize the digits prefilled in the puzzle. In a k-NN classification method, the output is classified on the basis of the votes from the k most common classes to the input object where k is a positive integer.

For the k-NN algorithm, the Modified National Institute of Standards and Technology (MNIST) dataset containing 60,000 training examples of handwritten digits [7] has been used. Once the digits are segmented into 81 sub-regions, the k-NN algorithm is applied for each of these images and the location in the 9x9 puzzle representing a certain sub-image is assigned a digit based on the k most common classes from the training dataset. The votes for each class are calculated based on the least to maximum Euclidian distance between each test and training image.

Various values of ‘k’ were tested for this algorithm to find no relevant effect in the results. Therefore, to save on computation time, a ‘k’ value of 3 was utilised. For the blank locations in the puzzle, a digit value of 0 was assigned.



5	3	0	0	7	0	0	0	0
6	0	0	1	9	5	0	0	0
0	9	8	0	0	0	0	6	0
8	0	0	0	6	0	0	0	3
4	0	0	8	0	3	0	0	1
7	0	0	0	2	0	0	0	6
0	6	0	0	0	0	2	8	0
0	0	0	4	1	9	0	0	5
0	0	0	0	8	0	0	7	9

Figure 7: Digit Recognised matrix obtained using k-NN algorithm

3.6 Sudoku Solving

From the previous knowledge of Artificial intelligence [8], we implied a constraint problem solving method to solve the sudoku puzzle using the ‘YALMIP’ [9] toolbox in matlab. To solve this problem, we have formulated three constraints as follows:

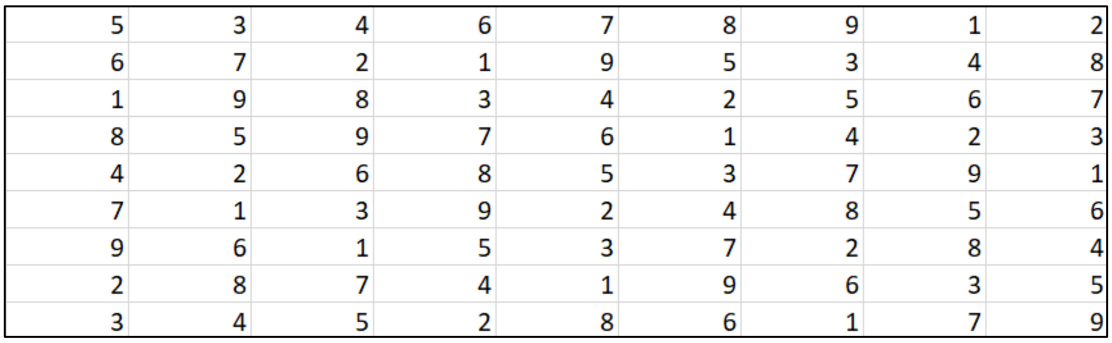
1) All the row elements of the sudoku should be unique and have values between 1 to 9.
2) All the column elements of the sudoku should be unique and have values between 1 to 9.
3) All the elements in the local 3x3 grids should be unique and have values between 1 to 9.

The solved sudoku puzzle was obtained as a 3-dimensional matrix ‘A’ with the first 2 dimensions representing the row and column number of the puzzle grid and the third dimension representing the digit value corresponding to location (row, column).

Finally, a 4th constraint was implemented to verify if each location of the puzzle contained only a single unique digit as follows:

$$\sum_{i=1}^{9} A(m,n,i) = 1 \tag{1}$$

Where m and n can be any numbers between 1 and 9. Therefore if a particular location consists more than 1 digit, the answer to the constraint above will be greater than one and this constraint will not be met. This constraint is used on all the rows, columns and all the 3x3 zone to get the final sudoku solved matrix as shown below:



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

Figure 8: Matrix showing Sudoku Puzzle solved using constraint programming

3.7 Reprojection

In this step, the solved sudoku is projected back onto the original plane that the image was captured in from the standard plane. Here we transverse the filled pixels and apply inverse transformation matrix to perform reprojection. Since the solved sudoku is smaller in dimensions compared to the input image, it is enlarged by 3 times for proper reprojection.

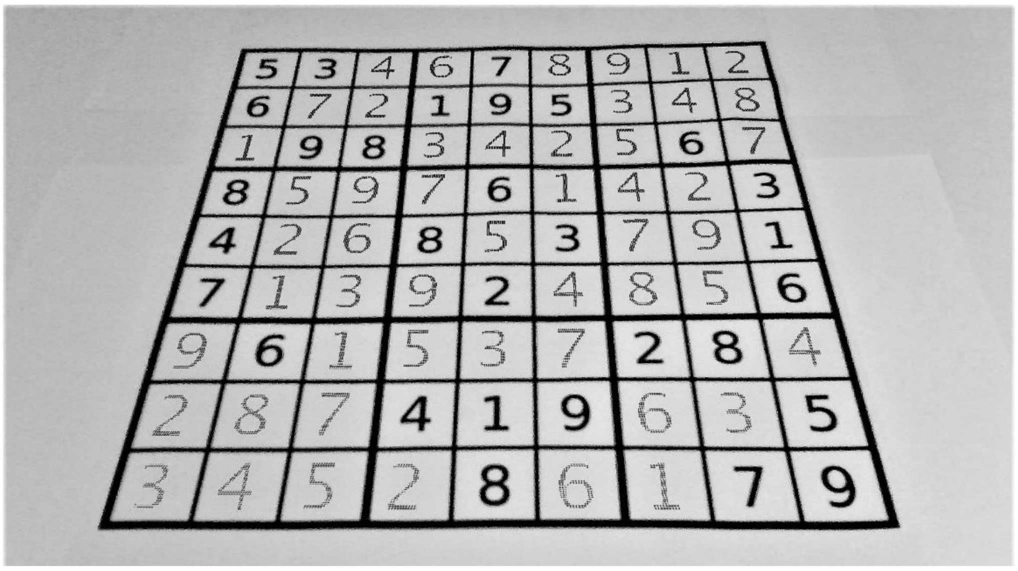
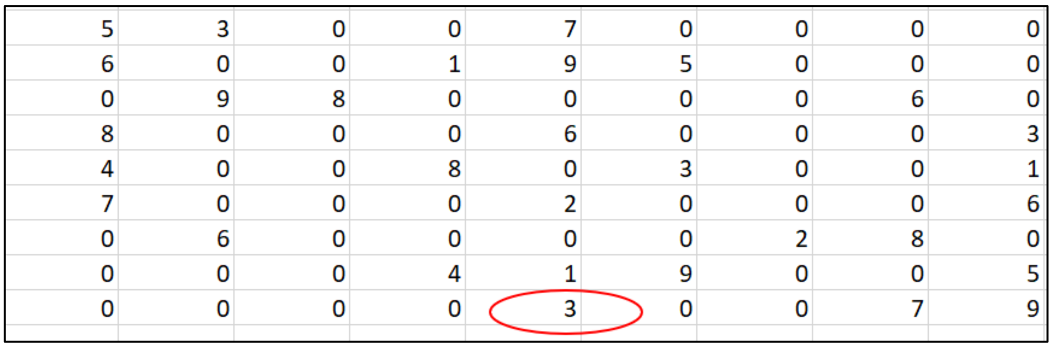


Figure 9: Reprojected image of solved Sudoku Puzzle

4. Experiments

4.1 Optical Character Recognition (OCR)

We used the MATLAB inbuilt OCR function to recognize the digits of sudoku based printed digits of different fonts, but we found that some of the digits were wrongly recognized. It was found that the algorithm failed to correctly distinguish between 3 and 8 and between 1 and 7. The figure below depicts an example of a case in which the algorithm failed to identify the digit ‘8’correctly. It was also observed that the OCR algorithm performed relatively slower in comparison to the kNN algorithm.



5	3	0	0	7	0	0	0	0
6	0	0	1	9	5	0	0	0
0	9	8	0	0	0	0	6	0
8	0	0	0	6	0	0	0	3
4	0	0	8	0	3	0	0	1
7	0	0	0	2	0	0	0	6
0	6	0	0	0	0	2	8	0
0	0	0	4	1	9	0	0	5
0	0	0	0	3	0	0	7	9

Figure 10: Digit recognition with MATLAB inbuilt OCR function: The red circle highlights that the digit 8 was recognised incorrectly as digit 3 using OCR recognition

4.2 Harris corner detection

Harris corner method was also used for corner detection. However, this method resulted in many of the pre-filled digits in the sudoku being detected as corners and therefore we utilised the more robust method of obtaining lines and corners using the Hough Transforms.

5. Limitations

Throughout the various experiments that we carried out, the following limitations were observed:

- The light condition under which the image has been captured affected the output since it decreased the contrast between the sudoku puzzle and the background.

- The condition of the paper when the image was taken affected the homography matrix calculated to project the original image in the 2d-standard plane. Clicking an image with a deformed or non-flat paper led to incorrect grid and digit detection.
- The solver also failed for tilted and reversed sudoku paper as the kNN algorithm incorrectly recognised tilted images

6. Future Work

- For the current algorithms that we have written, we plan to wrap the entire sudoku solver into an application for ease of access using the MATLAB's Application Compiler toolbox.
- We also plan to generate our own dataset for digit recognition since the MNIST dataset contains handwritten digits and the sudoku printed page does not have handwritten digits
- We plan to use this dataset and the MNIST dataset to also train a Convolution Neural Network in order to combine the results with the kNN algorithm to make the solver more robust

7. Peer Review:

Table 1: Peer-review

Group Member	Work-load distribution	Contribution Ratio
Nimish Magre (u6434979)	Printed Sudoku puzzle image capture, Image Pre-processing	25%
Kaushik Ghosh (u6425134)	Puzzle Grid detection, 2D homography	25%
Harpragaas Singh (u6424131)	Digit segmentation, Digit recognition	25%
Lakshmi Medicherla (u6420703)	Sudoku Solving, Reprojection of solved image	25%

References

[1] Anubhav Agarwal, C. V. Jawahar, and P. J. Narayanan, “A Survey of Planar Homography Estimation Techniques,” International Institute of Information Technology, Hyderabad, 2005.

[2] Wicht. B., & Hennebert, J., “Camera-based sudoku recognition with deep belief network,” *6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)* , pp. 83-88, 2014.

[3] Snigdha Kamal ; Simarpreet Singh Chawla ; Nidhi Goel, “Detection of Sudoku puzzle using image processing and solving by Backtracking, Simulated Annealing and Genetic Algorithms: A comparative analysis,” in *Third International Conference on Image Information Processing (ICIIP),* Waknaghat, India, 2015.

[4] L. Deng, “The MNIST Database of Handwritten Digit Images for Machine Learning Research,” *IEEE SIGNAL PROCESSING MAGAZINE*, 2012.

[5] U. Ravi Babu ; Y. Venkateswarlu ; Aneel Kumar Chintha, “Handwritten Digit Recognition Using K-Nearest Neighbour Classifier,” in *World Congress on Computing and Communication Technologies*, Trichippalli, India, 2014.

[6] A. Dwivedi, “Handwritten Digit Recognition with CNN,” 2 February 2019. [Online]. Available: https://datascienceplus.com/handwritten-digit-recognition-with-cnn/.

[7] Yann LeCun, Corinna Cortes, Christopher J.C. Burges, “THE MNIST DATABASE of handwritten digits,” [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[8] H. Simonis, “Sudoku as a constraint problem,” *Workshop on modelling and reformulating constraint satisfaction problems,* vol. 12, pp. 13-27, 2005.

[9] J. Lofberg, “YALMIP github,” 2019. [Online]. Available: https://yalmip.github.io/.