# Vision-based Sudoku Solver

Group ID: Group 8

Changdong Cai – u5957790

Guofeng Xu – u5491523

Jianyuan Wang – u6148908

Yilin Geng – u5613613

## 1 Introduction

The aim of this project is to develop a vision-based sudoku solver that is capable of solving printed sudoku puzzles on paper using computers with camera inputs. The deliverable of the project is presented as an executable program called 'SudokuN', which can also be referred to as 'Mr. Sudoku' in Japanese. The general process involves the image capture, noise reduction, puzzle grid detection, homography projection, digit segmentation and recognition, sudoku solving and results display. During this process, information is not only obtained and understood by computers but also processed and enhanced to help with human activities, which embodies the concept of human-computer interaction. The techniques applied, thoughts manifested, and experience gained within this project could be extended and help with future developments of various augmented reality technologies.

## 2 Related work

Vision based sudoku solver is an interesting project which involves multiple independent tasks, such as reading in sudoku puzzle, extracting data from the puzzle by image processing and solving the puzzle based on a numerical algorithm. In 2001, a team from MathWorks in Japan has developed a real-time sudoku solving program [1] to illustrate this process.

Several research topics related to this project such as edge detection, camera calibration, digit recognition and sudoku solving algorithms have been extensively studied and developed over the past few decades. In 1959, Hough transform was firstly introduced for machine analysis of bubble chamber [2] and three years later patented in the U.S. with the name 'Method and Means for Recognizing Complex Patterns' [3]. Today, this algorithm has been improved and optimized to detect lines and curves in pictures. Direct Linear Transformation was initially invented in 1971 by Abdel [4] and further extended in many aspects including 2D homography of camera calibration. Digit classification using Histogram of Orientated Gradient (HOG) features [5] was introduced in 2005 and it is now used in many Optical Character Recognition (OCR) [6] applications. Statistical template matching [7] was also one of the early recognition method. But in recent studies, Convolutional Neural Network (CNN) [8, 9] is most widely used and has the best recognition performance. In terms of sudoku solving algorithms, Backtracking [10], Stochastic search [11], Constraint programming [12] and Exact cover [13] are four main methods to solve the sudoku puzzle.

## 3 Methodology

### 3.1 Image input and pre-processing

SudokuN will access the customer's computer camera and allow the customer to select a region of interest. After a double-click, SudokuN will capture the image within a region of interest and then conduct image processing.

Image pre-processing is essential because the quality of the captured image will affect the accuracy of edge detection and digit recognition. Firstly, we convert the input RGB image into grayscale and increase its contrast. We can therefore avoid the effects of background and reduce annoying noise. Then, we sharpen the image and convert it to a binary image, which is the basis of following operations.

## 3.2 Puzzle frame detection

In order to comprehend the sudoku puzzle, a significant step is to locate and calibrate the puzzle grid. The general approach used in this report was to make use of four vertices of the outer frame. Different methods were implemented and compared during this explorative process.

### 3.2.1 Method 1 – Harris corner detection

With the help of harries corner detection [14], we can detect hundreds of corners in a decent puzzle paper sheet, including the four vertexes of the puzzle (Figure 1). After selecting the region of interest and cleaning the image, we can assume the detected corners are all inside the region of the sudoku puzzle. Hence, the four vertexes constitute a quadrilateral with the largest area. Exhaustive Search could help us find these four vertexes from candidates, as shown below:

1. Randomly select four corners.
2. Calculate the area the corners constitute.
3. Check whether it is the largest area so far. If yes, record the area and the corners; If not, continue.
4. Iterate step 1-3 until all possible combinations have been checked.

After multiple experiments, this algorithm is effective and robust to most disturbances. However, it suffers from high computational complexity O(n^4) while SudokuN is a real-time application. We will explore a more efficient method.

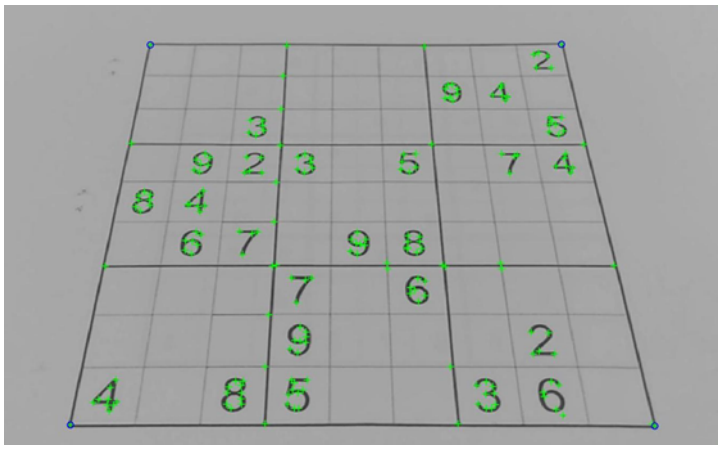

**Fig.1.** Example result of corner detection, where the blue circles represent the four vertices and green circles represent the detected corners

### 3.2.2 Method 2 – Hough transform edge detection

Besides using corner detection techniques, we also explore the possibility of edge detection utility. The theory aims to detect the frame based on the gridlines.

#### 3.2.2.1 Edge detection

A standard 9x9 Sudoku puzzle contains 81 separate blocks and thus 20 gridlines with equal length. As a voting procedure, Hough transform is specialized in locating legible straight lines [15]. We detect the gridlines by finding peaks in the parameter space as shown in Figure 2. These peaks indicate the potential straight lines with the most dots on them.

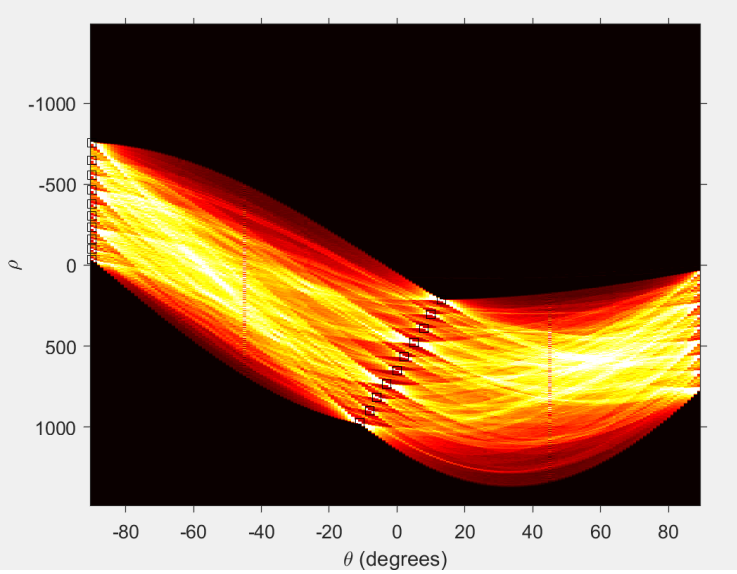


**Fig. 2.** The parameter space of 'Hough transform' and the strongest twenty peaks

By finding the strongest twenty peaks, we can pinpoint all the gridlines of the puzzle as shown by the colored lines in Figure 3.

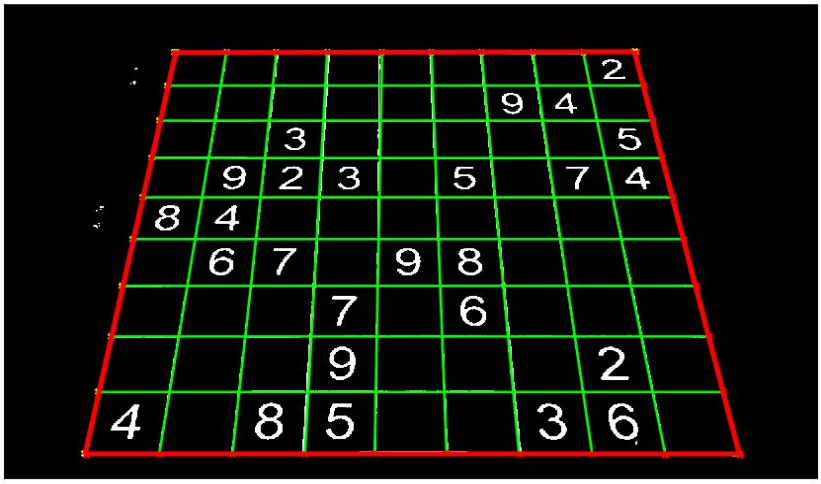

**Fig. 3.** Example result of the identified gridlines, where the borders were marked by red

We record the lines using two set of variables defined in two coordinates, where the origin of both coordinates is defined on the top-left corner of the image:

- The coordinates of the start and end points of the lines defined in Cartesian coordinates
- The radius and polar angles of the lines defined in polar coordinates

Meanwhile, since the Hough transform method counts the lines with the most votes, it is robust to most noises remained after image pre-processing.

#### 3.2.2.2 Border extraction

With twenty gridlines recorded in coordinates, we cluster them into two groups, namely horizontal and vertical lines. The grouping is conducted based on their polar angles: the polar angles of the lines in the same group are very close. We can hence eliminate the influence of unitary rotation. Then, we identify and extract the outer borders by finding two lines in each group that are the closest and the furthest to the origin as shown in Figure 3.

#### 3.2.2.3 Vertices identification

The four vertices of the outer frame are the intersections of the four identified borders. When extracting the boards, we record the coordinates of their start and end points. Using these point pairs, the equations of borders can be derived, and the intersections can be located as the solutions of the equation sets.

## 3.3 2D-homography & Digit segmentation

The raw captured image may stand in different filming angles, which will reduce our recognition rate. An intuitive idea is to conduct 2D transformation and project the raw image into a standard planar (as shown in Figure 4), with the help of direct linear transformation [4]. Since we have obtained four vertices in the raw image, we set four default locations in the standard planar. Therefore, four corresponding coplanar points are available, which ensure the transformation matrix available. We then traverse all the black pixels in the raw image and conduct projection.

The locations of corners are fixed in the standard puzzle, regardless of their original locations. Therefore, according to pre-set default locations of four corners, we generate a universal function to extract 81 sub-regions which contain a digit respectively. The segmentation results are available in Figure 5.

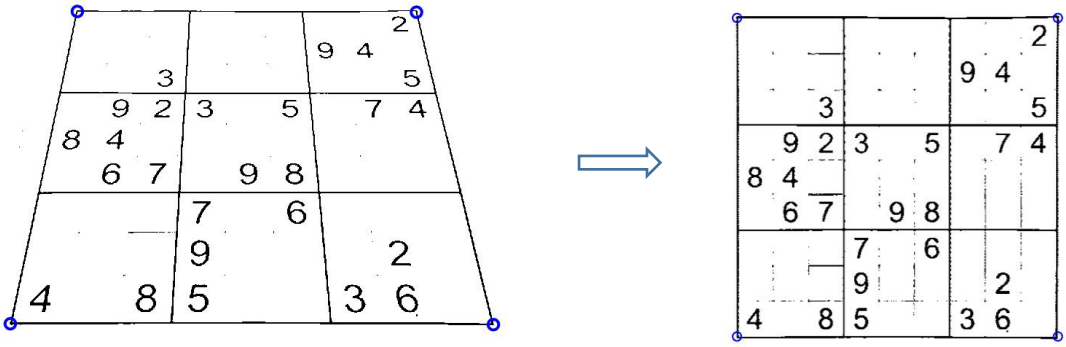

**Fig. 4.** Example result of the DLT projection, where the blue circles represent the corresponding coplanar points

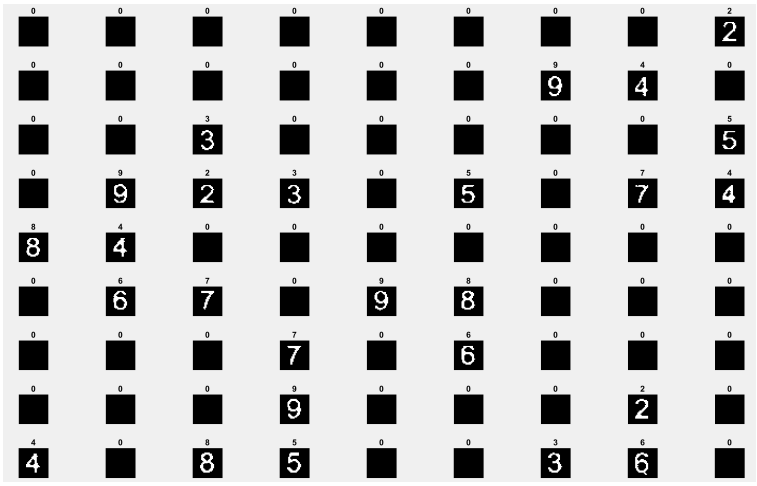

**Fig. 5.** Example segmentation result, which includes the 81 subregions in the standard planar

## 3.4 Digit recognition

In the sudoku solving process, the digit recognition procedure has the lowest error tolerance. A single mis-recognized digit can result in a different and even unsolvable sudoku puzzle. Multiple recognition techniques, such as CNN, *k*-NN, and OCR, were attempted with various confident. To further increase the accuracy, triple modular redundancy was applied. The basic idea is to simultaneously perform three

different recognition techniques and then to produce a single output through a majority-voting system [16]. Detailed illustration of these three techniques are discussed below.

### 3.4.1 CNN – convolutional neural network

To recognize the printed digits in the sudoku puzzles, we conduct transfer learning to fine-tune a convolutional neural network pre-trained on the MNIST database (Modified National Institute of Standards and Technology database) [8]. Transfer learning can transfer the learnt features to new tasks using a small new dataset [9]. The layer structure of the pre-trained network is available in Table 1.

**Table 1.** Network layer structure of the network pre-trained on MNIST database

| Index | Name | Description |
|---|---|---|
| **1** | Image Input | 28x28x1 images with 'zerocenter' normalization |
| **2** | Convolution | 8 3x3x1 convolutions with stride [1 1] and padding [1 1 1 1] |
| **3** | Batch Normalization | Batch normalization with 8 channels |
| **4** | ReLu | ReLU |
| **5** | Max Pooling | 2x2 max pooling with stride [2 2] and padding [0 0 0 0] |
| **6** | Convolution | 16 3x3x8 convolutions with stride [1 1] and padding [1 1 1 1] |
| **7** | Batch Normalization | Batch normalization with 16 channels |
| **8** | ReLu | ReLU |
| **9** | Max Pooling | 2x2 max pooling with stride [2 2] and padding [0 0 0 0] |
| **10** | Convolution | 32 3x3x16 convolutions with stride [1 1] and padding [1 1 1 1] |
| **11** | Batch Normalization | Batch normalization with 32 channels |
| **12** | ReLu | ReLU |
| **13** | Fully Connected | 10 fully connected layer |
| **14** | Softmax | softmax |
| **15** | Classification Output | crossentropyex with '0' and 9 other classes |

In Sudoku case, the digits are printed rather than handwritten as in MNIST. We tried transfer learning using a self-generated printed digit database. The digits provided in the six example puzzle pictures initially forms the basic training set. The database (including 6948 28x28 images) was augmented by adding displacement and deformation to the original sample digits. By applying different morphological operations including erosion, dilation, opening and closing, the training set would be more robust to illuminance and noise. Also, to avoid potential projection errors, the training set is further augmented by moving each pixel to four directions (up, down, left, right). The transferred network can reach an accuracy rate of 100% on the validation set. One detail worth mention is that since there is no ‘zero’ in sudoku puzzles, the zeros in transfer learning dataset represents empty blocks to be filled.

### 3.4.2 *k*NN – *k* nearest neighbours

The $k$-nearest neighbours algorithm ($k$-NN) is known as a non-parametric method used for classification [17]. A new output would be classified by a majority vote of its $k$-nearest neighbours based on the previous training. In this project, the training set of $k$-NN is the same as that of CNN method. The algorithm will assign the test image to the class most common among its $k$ nearest neighbours. Different values of $k$ has been tested and there is no apparent difference in terms of the recognition result. Accordingly, to keep the algorithm fast and simple, the default value has been chosen as 3.

### 3.4.3 OCR – optical character recognition

We also utilize MATLAB's inbuild optical character recognition (OCR) function [6]. Due to its large training set, the function ocr( ) can recognize printed digits of different fonts. Notably, this inbuilt function is greatly affected by the picture quality and is slower than CNN and $k$-NN discussed above. We ignore the sub-regions without enough information pixels (i.e., black pixels in this project) to accelerate the OCR recognition process. Meanwhile, we add white paddings to the cropped sub-regions to increase the recognition rate, as suggested by the function description. We have tuned the parameter settings of the ocr( ) function, such as focusing it on printed digits and adjusting the confidence threshold. As for the provided six testing pictures, the recognition rate of OCR is 100%.

## 3.5 Sudoku solving

### 3.5.1 Backtracking

One simple idea of solving a Sudoku puzzle is to use backtracking [10] which is mainly based on brute force search [18], namely systematically enumerating all possible candidates for the solutions and checking if they are satisfied for the constraints. This algorithm literally visits all the empty cells in order, fills all possible digits, and checks the validity of each digit in each empty cell. More clearly, starting from the first empty cell, if the digit '1' is allowed after checking all the constraints (row, column and 9 grids box), then place '1' in the current cell and move to the next empty cell. Otherwise, try digits from '2' to '9' on the current cell until one satisfying digit is found. If none of the nine digits is allowed, we leave the current cell blank and moves to the previous cell. This process ends when the last cell is filled with a proper digit or when the Sudoku puzzle is unsolvable.

### 3.5.2 Constraint programming

We turn the Sudoku problem into a constraint problem [12]. We can set and implement the constraints with the help of Matlab's YALMIP Toolbox [19]. There are three constraints in a sudoku problem: 1) each number can only appear once per row, 2) each number can only appear once per column, and 3) each number can only appear once in each 3x3 zone. We use a three-dimensional 9x9x9 binary matrix A to store Sudoku. The first two dimensions represent the rows and columns, and the last one represents the digits. '1' represents true and '0' represents false. For example, if A(3,4,5)=1, the digit in the third row and the fourth column is 5.

Meanwhile, each place can only have one digit. We can hence set another constraint that the sum of each face of the third dimension is 1, i.e.,

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

where $m$ and $n$ can be any number between 1 and 9.

With all the constraints prepared, we can achieve an appropriate solution under YALMIP Toolbox. Constraint programming is faster compared with the backtracking algorithm.

## 3.6 Re-projection

For the convenience of customers, we project the solved sudoku puzzle from the standard planar to the input one. We traverse the filled pixels (the blue ones) and conduct projection under the inverse transformation matrix.

Notably, the re-projected pixels are sparse because in most situations the standard image is smaller than the input one. Therefore, we enlarger the standard image by three times and then conduct re-projection. In the experiment, this method can improve the output image quality significantly although it will require approximately one more second of time consumption. The example result can be seen in Figure 6.

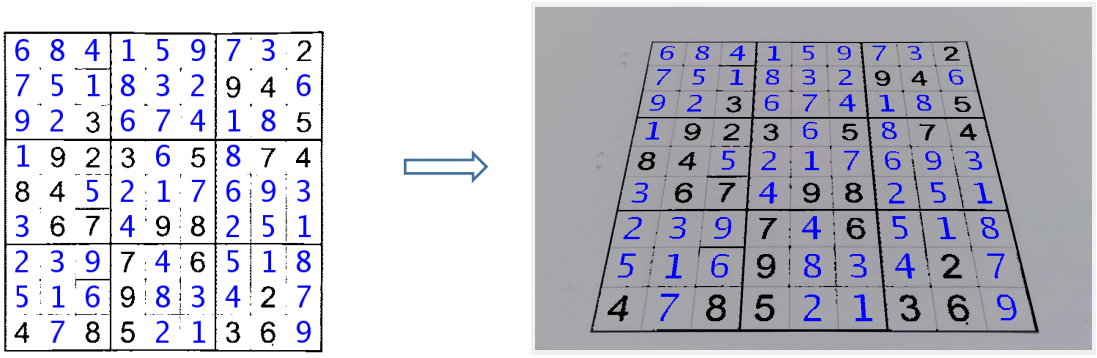

**Fig. 6.** Example result of the re-projection, from the standard planar (left) to the input planar (right)

# 4 Results and discussion

## 4.1 Results display

We have tested SudokuN on the six provided images, the example results are shown in Figure 7. Others are available in Appendix A.

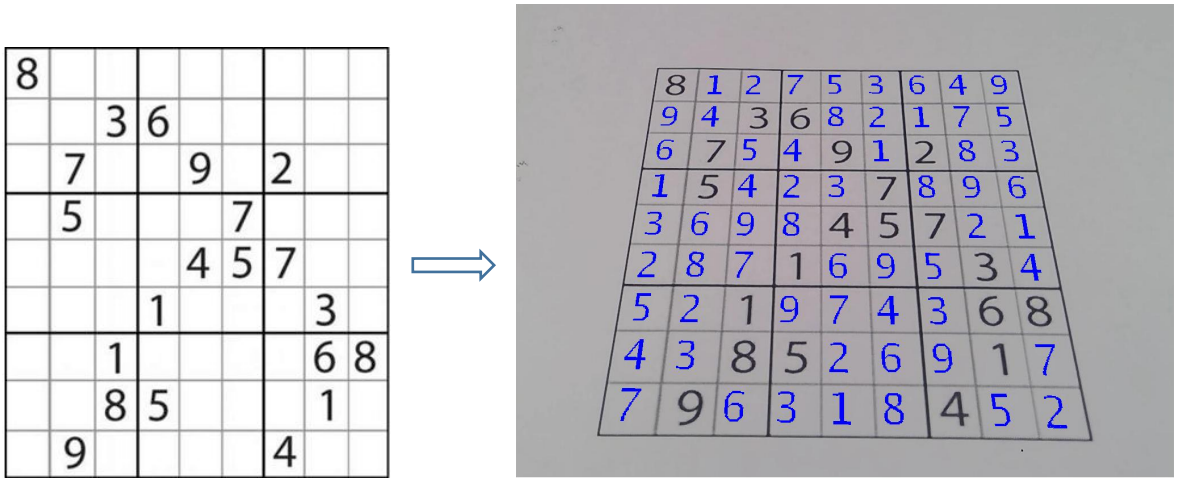

**Fig. 7.** Example result of SudokuN; the left image is the provided test image while the right one is the result

## 4.2 Difficulty evaluation

To enhance customers' interest, we designed a simple difficulty assessment function. We calculated the average time of solving 20 puzzles by SudokuN, which was around 8 seconds.

We endow SudokuN with a character of a cute but proud artificial intelligence (AI). For each input image, if SudokuN takes less than 8 seconds, it will display *'You are looking down on the most advanced AI! This sudoku is a no-brainer!'*. If it takes more than 8 seconds, it will admit that *'This sudoku is worth a fight'*.

## 4.3 Time-consuming problems

The first version of SudokuN will take over 15 seconds from identifying the interested region to showing the result. After several optimizations, the running time is reduced to less than 8 secs under our working environment.

We ignore the sub-regions without enough information pixels to avoid unnecessary recognition. Meanwhile, since the projection process requires high computation resources and is time-consuming,

we only project the sub-regions with notable information pixels. After implementing those two optimizations, the running time decreases to 8 secs. Then, we choose the constraint programming as our Sudoku solving algorithm instead of the backtracking algorithms, which saves approximately 2 seconds.

Notably, there is still space to be improved. For example, our program spends a lot of time to fill in the results by MATLAB's inbuild function insertText( ). In future, we can use pre-set templates to complete digits filling faster, although it may require additional storage space in the integrated software.

## 4.4 Limitations

It should be noted that following conditions may affect the performance of SudokuN, especially the digit recognition part:

- The webcam resolution is less than 1920x1280, which will influence the quality of the captured image.
- The paper sheet is apparently deformed, where DLT algorithm cannot work.
- The paper sheet is reversed or mirrored.
- The paper sheet is in dim light, which decreases the contrast between digits and background.

If the recognized puzzle cannot be solved, SudokuN will display 'Infeasible problem or Recognition error'. In future, we can expand the training set of SudokuN to enhance its robustness.

## 4.5 Handwritten recognition

We also developed the handwritten digits recognition capability for SudokuN, by changing the training set of the CNN model discussed above. The example result is shown below:

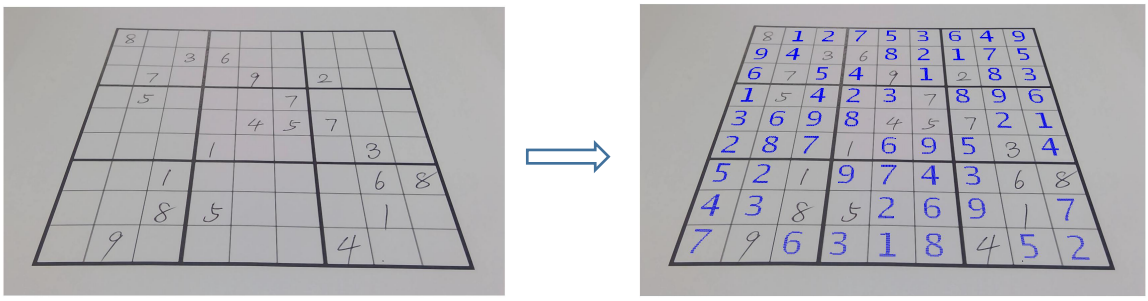

**Fig. 8.** Example result of solving handwritten sudoku

After multiple experiments, it has been verified that SudokuN can accurately recognize the normal handwritten digits. However, we also noticed that this would decrease the recognition rate of printed digits. Considering most sudokus are printed and the assignment instruction focuses on the printed digits, we removed the handwritten digits recognition capability in the final version.

## 4.6 Application Development

With the help of MATLAB's Application Compiler Toolbox, we have converted the SudokuN's codes into an independent software, as can be found in our supplementary materials. Notably, this software still requires that the customer has met the configuration requirements listed in the *readme* file.

We also tried to implement a mobile phone application. We found it is very difficult to reproduce SudokuN with C or Java, since it heavily relies on some complicated Matlab Toolboxes (including Neural Network Toolbox and Mapping Toolbox). In future we will possibly redesign this application on the basis of Java to generate the platform-free application.

# 5 Conclusion

SudokuN automates the task of solving a printed sudoku puzzle using a computer camera. A sudoku puzzle within the region of interest is firstly captured and processed to generate a binary version with almost no noise. Hough transform is then applied to detect twenty gridlines of the puzzle frame followed by extracting four outer-most borders from these gridlines. The four vertices, as the intersections of the borders, are used as reference points of a DLT process to project the captured puzzle to a frontal plane. Eighty-one equally large blocks with a single digit in each block are sliced out from the projected and normalized puzzle. Triple modular redundancy (TMR) is used as the decision process to reduce error in the digit detection. Three independent recognition methods used are convolutional neural network (CNN), $k$-nearest neighbour ($k$-NN), and optical character recognition (OCR). The recognized puzzle is solved using constraint programming and the results are filled into the puzzle on the frontal plane. Finally, it is projected back to the input form. The difficulty level of the puzzle is evaluated based on the computational time.

SudokuN performs stably from different viewing angle and under any angle of rotations within a relatively short time. It is also robust to various kinds of noise. By modifying the training set of networks and several coefficients, SudokuN can also solve for handwritten puzzles. However, this adjustment sacrifices the performance of recognizing the printed digits. One important lesson learnt is that the practical system can be very different from the theory with many kinds of unexpected noise.

SudokuN is currently capable of solving the puzzle with a single image input. For further improvement, a real-time solver can be expected, which provides feedback instantly and simultaneously back to the video input. The current 9x9 sudoku solver can also be extended to solve arbitrary dimensional puzzles. A balanced solution may be found to solve both handwritten and printed puzzles with satisfying accuracy. For convenient usage, a smartphone application is recommended for future development.

# 6 Summary of learning outcomes

This project involves multiple independent tasks which are closely related to satisfying the learning outcomes of this course. Firstly, to generate an input image, we have learnt image capturing, basic image processing techniques (resize, RGB to grey conversion, etc.) and image denoising & filtering. Then, some state of art algorithms in computer vision like Hough transform, Harris corner detector and DLT have been used to extract the image features and segment the image. What's more, the implementation of Convolution Neutral Network has brought us a better understanding on object classification and recognition which is essentially the high-level vision requirement. Overall, with all above steps, this project has shown how a working computer vision system should be. Through this project, we believe we have gained some qualified skills on critically reviewing and assessing scientific literature and applying the knowledge and techniques of modern computer vision theory to overcome encountered difficulties.

# Reference


[1] Y. Kajikawa, W.-S. Gan, and S. M. Kuo, "Recent advances on active noise control: open issues and innovative applications," *APSIPA Transactions on Signal and Information Processing,* vol. 1, p. e3, 2012.

[2] P. V. Hough, "Machine analysis of bubble chamber pictures," in *Conf. Proc.*, 1959, pp. 554-558.

[3] P. V. Hough, "Method and means for recognizing complex patterns," ed: Google Patents, 1962.

[4] Y. Abdel-Aziz, "Direct linear transformation from comparator coordinates in close-range photogrammetry," in *Proceedings American society of photogrammetry symposium on close-range photogrammetry. Falls Church (VA). American Society of Photogrammetry Symposium on Close-Range Photogrammetry., 1971*, 1971, pp. 1-19.

[5] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, pp. 886-893.

[6] S. Mori, H. Nishida, and H. Yamada, *Optical character recognition*: John Wiley & Sons, Inc., 1999.

[7] L. Welling, H. Ney, A. Eiden, and C. Forbrig, "Connected digit recognition using statistical template matching," in *Fourth European Conference on Speech Communication and Technology*, 1995.

[8] N. Ishibashi, "Understanding Adversarial Training: Improve Image Recognition Accuracy of Convolution Neural Network," 2017.

[9] J. West, D. Ventura, and S. Warnick, "Spring research presentation: A theoretical foundation for inductive transfer," *Brigham Young University, College of Physical and Mathematical Sciences,* vol. 1, 2007.

[10] GeeksforGeeks. (2016). *Backtracking | Set 7 (Sudoku)*. Available: https://web.archive.org/web/20160828164622/http://www.geeksforgeeks.org/backtracking-set-7-suduku/

[11] M. Perez and T. Marwala, "Stochastic optimization approaches for solving Sudoku," *arXiv preprint arXiv:0805.0697,* 2008.

[12] H. Simonis, "Sudoku as a constraint problem," in *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, 2005, pp. 13-27.

[13] Wikipedia. (2017). https://en.wikipedia.org/wiki/Exact_cover#Sudoku. Available: https://en.wikipedia.org/wiki/Exact_cover#Sudoku

[14] C. Harris and M. Stephens, "A combined corner and edge detector," in *Alvey vision conference*, 1988, p. 10.5244.

[15] L. Shapiro and G. Stockman, "5, 7, 10," *Computer Vision. Upper Saddle River, New Jersey: Prentice-Hall, Inc,* pp. 157-158, 2001.

[16] D. Ratter, "FPGAs on mars," *Xcell J,* vol. 50, pp. 8-11, 2004.

[17] Y. Lee, "Handwritten digit recognition using k nearest-neighbor, radial-basis function, and backpropagation neural networks," *Neural computation,* vol. 3, pp. 440-449, 1991.

[18] Wikipedia. (2018). *Brute-force search*. Available: https://en.wikipedia.org/wiki/Brute-force_search

[19] J. Löfberg. (2018). *YALMIP*. Available: https://yalmip.github.io/

# Appendix

## A: Results of test images

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | | | | 2 |
| | | | | | | 9 | 4 | |
| | | 3 | | | | | | 5 |
| | 9 | 2 | 3 | | 5 | | 7 | 4 |
| 8 | 4 | | | | | | | |
| | 6 | 7 | | 9 | 8 | | | |
| | | | 7 | | 6 | | | |
| | | | 9 | | | | 2 | |
| 4 | | 8 | 5 | | | 3 | 6 | |

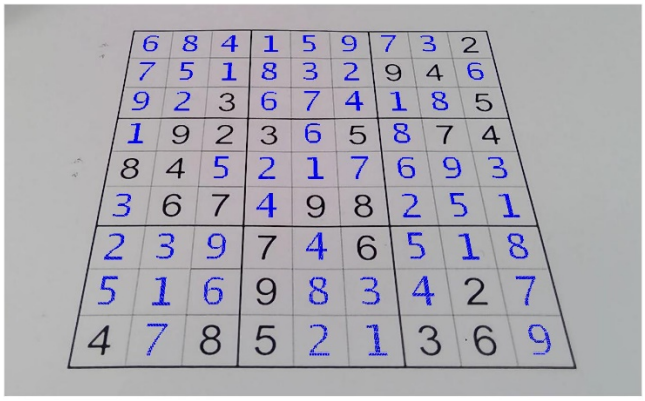

**Fig. A.** SudokuN result of test image sodoku_1

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

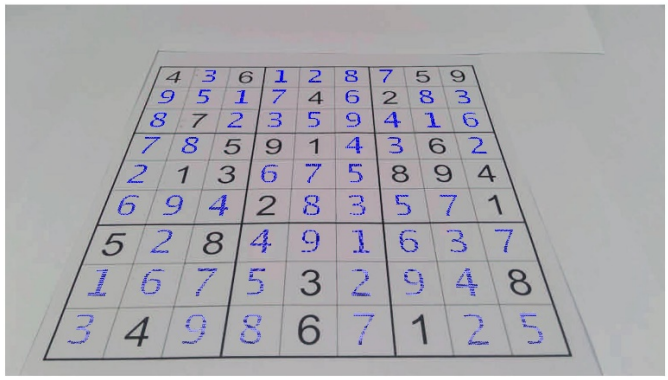

**Fig. B.** SudokuN result of test image sodoku_2

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

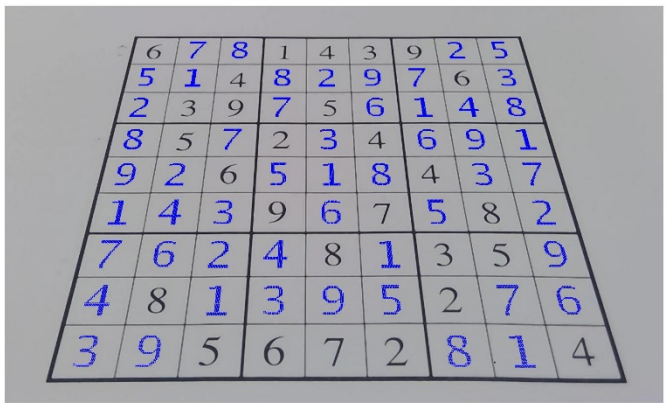

**Fig. C.** SudokuN result of test image sodoku_3

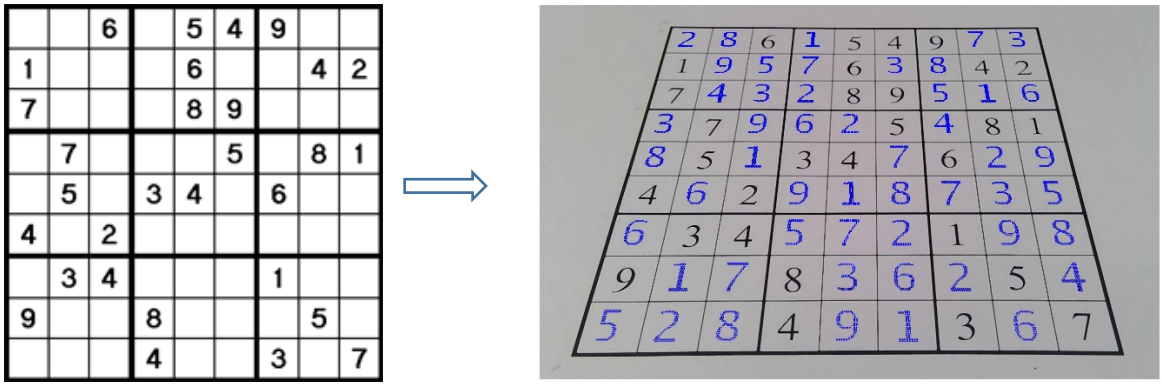

**Fig. D.** SudokuN result of test image sodoku_5

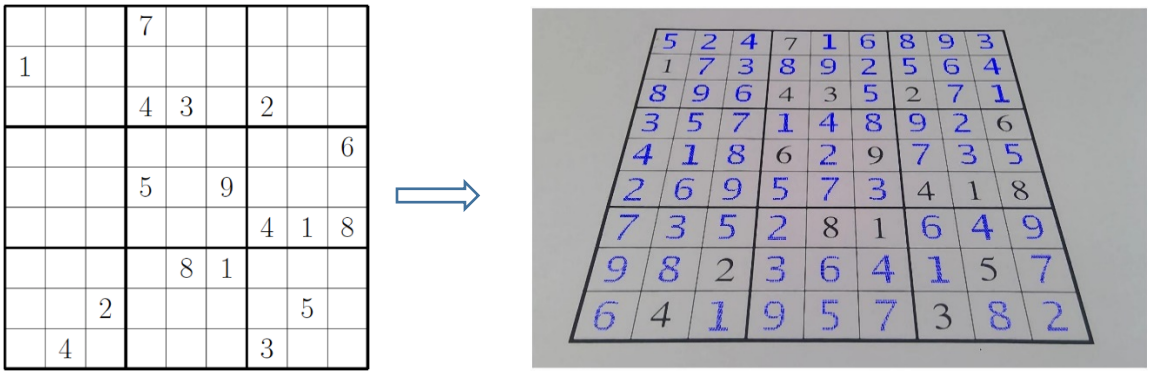

**Fig. E.** SudokuN result of test image sodoku_6