JSS MAHAVIDYAPEETA

JSS SCIENCE AND TECHNOLOGY UNIVERSITY

SRI JAYACHAMARAJENDRA COLLEGE OF ENGINEERING

MYSORE-570006

COMPUTER VISION PROJECT

Project Title : SOLVING SUDOKU USING IMAGE PROCESSING TECHNIQUES

Submitted by,

Kirthan M,
01JST17IS024,
6th Sem,
Information Science and Engineering

DECLARATION

I, the undersigned solemnly declare that the project report '**SOLVING SUDOKU USING IMAGE PROCESSING TECHNIQUES'** is based on my own work carried out during the course of our study under the supervision of Dr. B S Harish, Assistant Professor, Department of Information Science and Engineering, JSS S&TU University, Mysore.

I assert the statements made and conclusions drawn are an outcome of my research work. I further certify that,

I. The work contained in the report is original and has been done by me under the general supervision of my supervisor.

II. The work has not been submitted to any other Institution for any other degree/diploma/certificate in this university or any other University of India or abroad.

III. I have followed the guidelines provided by the university in writing the report.

IV. Whenever I have used materials (data, theoretical analysis, and text) from other sources, I have given due credit to them in the text of the report and giving their details in the references.

Table of Contents

ABSTRACT

In this project, we propose a method of detecting and recognizing the elements of a Sudoku Puzzle and providing the complete solution for it. The method involves a vision-based Sudoku solver. The solver is capable of solving a sudoku puzzle directly from an image. After applying appropriate pre-processing to the acquired image, we use Canny Edge Detection Algorithm to recognize the enclosing box of the puzzle. A virtual grid is then created using Hough Transforms to identify the digit positions. Then each grid is sliced into separate independent images. Tesseract OCR (optical character recognition) engine is used as a method for digit recognition. The actual solution is computed using a backtracking algorithm. Experiments conducted on various types of Sudoku puzzles, demonstrate the efficiency and robustness of our proposed approaches in real-world scenarios. The algorithm is found to be capable of handling cases of translation, blurring, noise, outliers, scaling, and background clutter.

Keywords – Binary Thresholding; Canny Edge Detection Algorithm; Hough Transform; Slicing an Image; Digit Recognition; Vision-based Sudoku solver; Recursive Backtracking Algorithm; Tesseract OCR; Image Processing Techniques;

1 Chapter 1 : Introduction

1.1 Problem Statement

Sudoku Image Solver is a Python program that reads an image, scans for Sudoku puzzle, recognizes the digits and solves it using backtracking method.

It uses various image processing techniques such as canny edge detection algorithm to detect and remove unwanted edges, hough transform to resize, crop and extract the sudoku puzzle from the image, Tesseract OCR Engine to recognize the digits of the puzzle and backtracking to solve puzzle using the recognized digits.

1.2 General Introduction

Sudoku (originally called Number Place) is a logic-based, combinatorial number-placement puzzle. The objective is to fill a 9×9 grid with digits so that each column, each row, and each of the nine 3×3 sub-grids that compose the grid (also called "boxes", "blocks", or "regions") contain all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. The name Sudoku comes from Japan and consists of the Japanese characters Su (meaning ‘number’) and Doku (meaning ‘single’) but it was not invented in Japan. Sudoku was originated in Switzerland and then travelled to Japan through America.

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

Fig. 2.1 An Example of a Sudoku Puzzle

A complete Sudoku solution may be arrived at in more than one ways, as we can start from any of the given clues that are distributed over the mini-grids of a given Sudoku instance (to be solved). There is no known technique in the literature that we surveyed to determine how many different starting cells there are.

Traditionally, we have solved sudoku using a pen and paper and figuring out the solution by filling the blocks with numbers. The rules of the game are simple: each of the nine blocks has to contain all the numbers 1-9 within its squares. Each number can only appear once in a row, column or block. As we progressed on, if any of the numbers contradict we move one step back and overwrite the number with another number. This process continues till we satisfy the rules of

sudoku and we obtain the final result in which each number appears only once in a row, column or block.

Instead of solving it on a paper, we can make a computer to solve the sudoku puzzle using the same concept which we used i.e., backtracking. But instead of giving the input as a matrix or a grid to the computer to solve, we take it one step above. We give the input to the computer in the form of images which we can see with our eyes, and process that image to give the final output of the sudoku puzzle. We try to solve the Sudoku puzzle from just an image in our project.

1.3 General Block Diagram

This process of solving the Sudoku puzzle from an image can be implemented as given in the following block diagram :

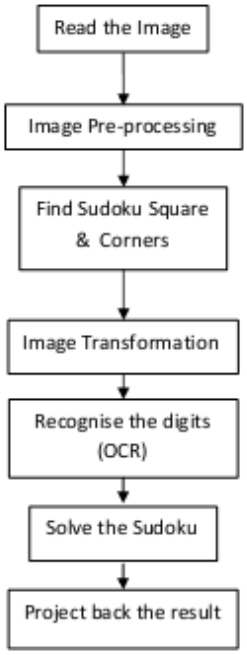


Fig. 2.2 General Block Diagram of
Solving Sudoku from an Image

We will deal with each of the above steps as follows :

1. **Reading the Image** : It is our normal image reading in python using PIL or MatplotLib.
2. **Image Pre-processing** : It includes noise removal, brightness/contrast adjustment, thresholding etc.
3. **Find Sudoku Square & Corners** : Here we find outer border of Sudoku square and its corners.
4. **Image Transformation** : Here we reshape irregular Sudoku in input image to a perfect square.

5. **Recognize the digit (OCR)** : Recognizes the digits in input image and place them in correct position.
6. **Solve the Sudoku** : Here, real solving of Sudoku take place. (using backtracking)
7. **Project back the Result** : We project the image of the solved Sudoku puzzle back to the user.

1.4 Applications

Application Domains of Solving Sudoku Instances :

1. **Steganography**

   Steganography is the art of hiding the existence of information within seemingly harmless carriers such as image, video, or audio. Using an 18×18 Sudoku puzzle and dividing it into eighteen 3×3 blocks we get as minigrids, may provide better quality in revealing the images.

2. **Visual Secret Sharing**

   Sudoku puzzle logic can be used for dividing the secrets into shadows and combining those shadows to reconstruct the original secret information.

3. **Short Message Service (SMS)**

   Better security in hiding the SMS data can be achieved by using Sudoku puzzle and some Steganographic algorithms. The puzzle encapsulating the message is then sent in a way that it does not attract any attention of some possible intruder. After solving the Sudoku puzzle on the receiving end, one may extract the hidden data in accordance with order of numbers 1 through 9 in the sequence of certain rows and/or columns, which is equivalent to the hidden information.

4. **Digital Watermarking**

   Interestingly it has been claimed that if we use the logic of Sudoku puzzle to create a reference matrix for digital watermarking, we can attain both high volume and image quality.

5. **Image Authentication**

   Using the logic of Sudoku puzzle it has been claimed that the parts of an image can easily be identified where the tampering has been attempted.

6. **Data Security**

   Using the rectangular Sudoku matrices of size 4×5 each, a new cipher can be designed that could be used to encrypt an image data in a very efficient manner.

7. **DNA Sequencing**

Erlich and others in the Hannon laboratory came up with an idea of mixing the samples in specific patterns, and thereby creating pools of samples. Instead of tagging the individual samples contained by each pool, the scientists tagged each pool as a whole with one barcode. They used the logic and number placement rules of Sudoku puzzle for creating the pooling strategy.

8. **Air Track Maintenance**

Here, the Sudoku logic can properly be utilized to divide the zones of the tracks in proper way, so that we can overcome these limitations.

1.5 Challenges

The challenges faced during the development of this project are as follows :

- Scanning the sudoku puzzle image such that whole puzzle is scanned.
- Identifying the sudoku borders and corners in the image.
- Identifying the 9x9 grid lines in the sudoku puzzle.
- If the grid lines were very lightly visible, then it is difficult to identify them.
- Identifying the 9, 3x3 grids in the puzzle.
- Identifying further 9 boxes from each grid i.e., total of 81 boxes.
- Extracting numbers from each box of the puzzle.
- If the numbers are not parsed correctly, then it is difficult to identify them.
- If the input image is blurred, then it is difficult to extract the sudoku puzzle.
- Pre-processing the input image.

1.6 Motivation

The factors that motivated me to successfully carry out this project are as follows :

- The vast applications of Computer Vision.
- Computers can be made to gain high-level understanding from digital images or videos. We can automate tasks the human visual system does using computer vision concepts.
- It is extraordinary how computers can visualize and solve things just like humans.
- People traditionally use pen and paper to solve sudoku puzzles. But instead we can solve it very fast using a computer.
- In fact, we need not even give the inputs as numbers to a computer. We can just capture the image of the sudoku puzzle and give it as an input to the computer. The computer solves the puzzle very quickly and displays the final solution.

- In this way, the traditional method of solving the sudoku puzzle can be minimized. And new and efficient ways can be developed.

1.7 Objectives of the Project

The objective of this project is to solve Sudoku with the use of recursive back tracking algorithm while satisfying the puzzle constraints.

Constraints :

1. Sudoku puzzle can only contain the numbers 1 through 9.
2. A position constraint: Only 1 number can occupy a cell.
3. A row constraint: Only 1 instance of a number can be in the row.
4. A column constraint: Only 1 instance of a number can be in a column.
5. A region constraint: Only 1 instance of a number can be in a region.

The other objectives are :

- To train the data model such that it recognizes the sudoku borders and corners from the input image.
- To train the data model such that it recognizes the individual grid lines in the sudoku puzzle.
- To train the data model such that it recognizes the digits in the puzzle.
- To test the data model so as to check whether it is recognizing the various aspects of the sudoku puzzle or not.
- To decrease the error of parsing the image correctly so that we can efficiently extract data from the image.
- To give a sudoku puzzle image as an input, and get the complete solution of it as output.

2 Chapter 2 : Literature Survey

[1] Ray Smith (July 2007) described the Tesseract OCR Engine in a comprehensive overview. Emphasis was placed on aspects that are novel or at least unusual in an OCR engine, including in particular the line finding, features/classification methods, and the adaptive classifier.

[2] Sian K Jones et al., (Nov 2007) proposed important features of the Sudoku puzzle and offered several proofs for them. Sudoku is a deceptively simple logic puzzle which has captured great public interest. Consisting of a 9×9 grid and further subdivided into 'mini-grids' of size 3×3, published puzzles are sufficiently simple in concept for wide sections of the population to attempt their solution, which still retaining a sufficient challenge for most through the necessity of applying several methods of reasoning. Academic interest in this class of puzzles has grown in recent years, due to both their relationship with other combinatorial structures (in particular Latin Squares) and their demonstrated connection to many real-world problems. This interest has taken the form of mathematical proof of specific puzzle properties and the application of search optimization techniques for its solution.

[3] Timo Mantere et al., (July 2008) studied and proposed the problems involved in solving and analyzing Sudokus with cultural algorithms. Sudoku has been claimed to be very popular and addictive because it is very challenging but has very simple rules. The objectives of this study are – 1) to test if a cultural algorithm (CA) with a belief space, solves Sudoku puzzles more efficiently than a normal permutation genetic algorithm (GA), – 2) to see if the belief space gathers information that helps analyze the results and improve the method accordingly.

[4] Pramod J Simha et al., (March 2012) proposed a method of detecting and recognizing the elements of a Sudoku Puzzle and providing a digital copy of the solution for it using MATLAB. The method involves a vision-based sudoku solver. The solver is capable of solving a sudoku directly from an image captured from any digital camera. After applying appropriate pre-processing to the acquired image we use efficient area calculation techniques to recognize the enclosing box of the puzzle. A virtual grid is then created to identify the digit positions. Template matching is used as a method for digit recognition. The actual solution is computed using a backtracking algorithm. Experiments conducted on various types of sudoku questions demonstrate the efficiency and robustness of the proposed approaches in real-world scenarios. The algorithm is found to be capable of handling cases of translation, perspective, illumination gradient, scaling, and background clutter.

[5] Atul Patel, PhD., et al., (Oct 2012) described the comparative study of the Tesseract OCR tool with other commercial OCR tools such as Transym OCR by considering vehicle number plate as input. From vehicle number plate they tried to extract vehicle number by using Tesseract and Transym and compared these tools based on various parameters. Optical character recognition (OCR) method has been used in converting printed text into editable text. OCR is

very useful and popular method in various applications. Accuracy of OCR can be dependent on text preprocessing and segmentation algorithms. Sometimes it is difficult to retrieve text from the image because of different size, style, orientation, complex background of image etc.

[6] Xiuqin Deng et al., (March 2013) presented a hybrid genetic algorithm (HGA) that uses a "random" technology to improve the performance of genetic algorithm. It also shows the performance of the algorithms for solving Sudoku puzzles with 46 different examples. The experimental results presented this paper indicate that HGA is able to produce very competitive results with respect to meta-heuristics algorithm at a considerably lower computational cost.

[7] Rohit Iyer et al., (May 2013) presented a novel technique for very fast Sudoku solving using recognition of various patterns like Naked Singles, Hidden Singles, Locked Candidates, etc. and reviewed it by conducting experiments and plotting the observations. Evaluation of this technique in solving random set of Sudoku puzzles collection showed that the rate of solving can be greatly improved. However, only selected patterns can be used for Sudoku solving in this review while even further improvement in solving rate may be possible if some more patterns could be detected and solved.

[8] Prof. S. T. Khandare et al., (Jan 2014) proposed a method of segmentation that can work effectively for image based automatic thresholding and color model based image segmentation. In computer vision, image segmentation is the process of partitioning a digital image into multiple sections. The goal of segmentation is to simplify and/or change the representation of an image into something that is more important and easier to examine. Image segmentation is typically used to locate objects and background in images. More exactly, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. Image segmentation is an important signal processing tool that is widely employed in many applications including object detection, object-based coding, object tracking, image retrieval, and clinical organ or tissue identification. Thresholding is the basic method of image segmentation. From a gray scale image, thresholding can be used to generate binary images. The idea of this method is to select the threshold value. A number of accepted methods are used in engineering including the maximum entropy method, Otsu's method that uses maximum variance, and k-means clustering.

[9] Arnab Kumar Maji et al., (Feb 2014) proposed a minigrid based novel technique to solve the Sudoku puzzle in guessed free manner. 'Sudoku' is a popular Japanese puzzle game that trains our logical mind. The word Sudoku means 'the digits must remain single'. The Sudoku problem is important as it finds numerous applications in a variety of research domains with some sort of resemblance. Applications of solving a Sudoku instance are found in the fields of Steganography, Secret image sharing with necessary reversibility, Encrypting SMS, Digital watermarking, Image authentication, Image Encryption, and so and so forth. All the existing

Sudoku solving techniques are primarily guess based heuristic or computation intensive soft computing methodology. They are all cell based, that is why very much time consuming.

[10] Sankhadeep Chatterjee et al., (Oct 2014) performed a comparative performance characteristics study that revealed the superiority of the Graph Referencing algorithm over the other algorithms in taking least possible time to solve Sudoku. Solving Sudoku, a NP-Complete combinatorial optimization problem has been carried out using the optimized Graph Referencing Algorithm (GRA), Genetic Algorithm (GA), Simulated Annealing (SA), Harmony Search (HS) and Brute Force algorithm. This study primarily aimed at finding out the fastest algorithm in terms of least time consumption in solving Sudoku. The performance characteristics of algorithms of interest are studied by deploying randomly selected puzzles with different difficulty levels.

[11] H. L. Zhao et al., (Dec 2014) proposed a Sudoku Designer framework for the systematic generation of a complete Sudoku by utilizing the binary integer programming (bintprog) procedure in Matlab, as well as the evaluation of its difficulty level with the Analytic Hierarchy Process (AHP). The Sudoku puzzle has attracted great attentions through its easy logic yet abstruse nature. However, most existing approaches are concentrated on its solution rather than generation and rating.

[12] Allam Shehata Hassanein et al., (Jan 2015) described the variations of the Hough Transform elaborating on the basic ones such as the line and circle Hough transforms. For more than half a century, the Hough transform is ever-expanding for new frontiers. Thousands of research papers and numerous applications have evolved over the decades. Carrying out an all-inclusive survey is hardly possible and enormously space-demanding. They emphasized on some of the most crucial milestones of the transforms. The high demand for storage and computation time is clarified with different solution approaches. Since most uses of the transform take place on binary images, they were concerned with the work done directly on gray or color images. The myriad applications of the standard transform and its variations have been classified highlighting the up-to-date and the unconventional ones. Due to its merits such as noise-immunity and expandability, the transform has an excellent history, and a bright future as well.

[13] Prof. R. N Mandavgane et al., (Feb 2015) reviewed the Canny Edge Detection algorithm. Edge detection is one of the key stages in image processing. In this paper using canny edge detection algorithm, Edges will detect the efficiently with reduction in the processing speed and reduced the memory requirement. Canny edge detection algorithm reduces the latency and increase the throughput with no loss in edge detection performance and algorithm which has the ability to compute edge of multiple blocks at the same time. Canny developed an approach to derive an optimal edge detector for clean and noisy images the canny edge detection algorithm yield better edge detection result.

[14] Snigdha Kamal et al., (Dec 2015) proposed the digital detection and decryption of a sudoku puzzle using vision based techniques and subsequent solving of the puzzle using three algorithms - Backtracking, Simulated Annealing and Genetic Algorithm. The proposed method could recognize any sudoku puzzle captured from a digital camera and after employing appropriate pre-processing algorithms which included adaptive thresholding, Hough Transform and geometric transformation, the digits were recognized using Optical Character Recognition (OCR), and based on their pixel locations in the image, they were stored in corresponding locations in the 9×9 matrix. The detected puzzles of varying complexity levels were then solved using the three algorithms and the results were compared and contrasted, indicating the relative efficiencies of the three techniques in accurately solving Sudoku puzzles. Simulated Annealing performed the best amongst the three algorithms, whereas, genetic algorithm performed the worst in the comparison.

[15] Akhil S Nair (Dec 2016) overviewed Google's open source Optical Character Recognition Software Tesseract. He described the aspects that are novel or at least unusual in Tesseract compared to other OCR engines. Optical character recognition is the machine replication of human reading and has been the subject of intensive research for more than three decades. It can be described as Mechanical or electronic conversion of scanned images where images can be hand written, type- written or printed text.

[16] Paritosh Pujari et al., (Dec 2017) discussed about various methods for digital detection and interpretation of a Sudoku puzzle using optical character recognition and vision based techniques and solving the subsequent puzzles using various computer algorithms. The popular Japanese puzzle game Sudoku is one of the most popular puzzle games of all times which is based on logical placement of numbers. The prime objective of this review is to do literature survey of methods to recognize Sudoku puzzles containing numerical digits from images taken with a mobile camera and puzzle solving techniques. An image in general portrays a visual perception of an artifact, for example a photo or a two-dimensional picture depicting any object, place or a person etc. In Computer Science, a digital image is a numeric representation of a two-dimensional picture, which often contain texts which are in a human readable format. These texts in a digital image are defined by a set of pixels.

[17] Onokpasa Eva et al., (Feb 2019) compared the three methods of solving the Sudoku puzzle: Pencil and paper method, backtracking and the method of alternating projections. The comparison was carried out by counting the number of iterations taken to solve 40 puzzles of various levels of difficulty, using php implementations of the solver algorithms. The 9 X 9 board game of Sudoku is intriguing and brain tasking.

3 Chapter 3 : Proposed Method

3.1 Design of the Project

We are going to solve a sudoku puzzle from just an image. We use various image processing techniques to read, process, detect, predict and solve the puzzle.

Steps involved in the design process are as follows :

Step-1: Reading an image
Step-2: Basic image preprocessing - **Thresholding**
Step-3: Edge Detection - **Canny edge detection algorithm**
Step-4: Resizing and cropping the image - **Hough transform**
Step-5: Slicing the image - using **PIL** library
Step-6: Recognizing digits from each slice – **Tesseract OCR Engine**
Step-7: Solving the sudoku puzzle - **Backtracking**
Step-8: Projecting back the result

1. Reading the image :

First, we need to read the image containing the Sudoku puzzle. We can use various Python libraries to read an image. Python supports very powerful tools when it comes to image processing. We can read an image using any one of the following libraries:

a. Using **MatplotLib**
b. Using **PIL (Python Imaging Library)**
c. Using **OpenCV (Open Source Computer Vision)**

2. Basic image processing :

We perform the basic image processing task i.e., Thresholding of the image. Thresholding is the simplest method of segmenting image. From a grayscale image, thresholding can be used to create binary images.

We use **Binary Thresholding** in our project to separate background from foreground, i.e., making the background white and foreground (digits and lines) black.

3. Edge Detection :

Edge detection is an image processing technique for finding the boundaries of objects within images. Common edge detection algorithms include Sobel, Canny, Prewitt, Roberts, and fuzzy logic methods.

In our project, we use **Canny Edge Detection Algorithm** to detect edges and remove unwanted edges.

4. Resizing and cropping the image :

We need to resize the image to obtain only the part of the image which we are interested in. Resizing means changing how large an image appears on some output medium, whether printing or on screen. Cropping is the process of removal of unwanted outer areas from an image.

We use **Hough Transform** to resize and crop the image. The **Hough Transform** is a method used in image processing to detect any shape, if that shape can be represented in mathematical form. Hough transform works for line detection using the HoughLine transform method.

5. Slicing the image :

We need to cut the image into smaller parts to get the required information from each part. Slicing is the process of cutting up the image into smaller logical images.

We use **PIL** library of python to slice the image into smaller images.

6. Recognizing the digits from each slice :

After slicing, we need to recognize the digits present in each slice of the image.

We use **Tesseract OCR Engine** to recognize the digits in each slice. Tesseract is an open source optical character recognition (OCR) engine originally developed at HewlettPackard between 1985 and 1995. Since HP had independently-developed page layout analysis technology that was used in products, (and therefore not released for open-source) Tesseract never needed its own page layout analysis. Tesseract therefore assumes that its input is a binary image.

7. Solving the sudoku puzzle :

After recognizing the digits of the puzzle, we need to solve the sudoku puzzle. We solve it using **Backtracking** approach. Backtracking is an algorithmic - technique for solving problems recursively by trying to build a solution incrementally, one piece at a time, removing those solutions that fail to satisfy the constraints of the problem at any point of time.

8. Projecting back the result :

After solving the puzzle completely, we need to print or project the result back to the user.

We project back the result in a browser window using the concept of **HTML** and **CSS**.

3.2 Algorithms and Mathematical Equations Used

1. Thresholding :

We use Binary Thresholding to separate the foreground from the background. The threshold value is kept as 40 and the maximum value of the pixel is generally 255. The pixel values above the 40 are set to the highest value i.e., 255, and the pixel values below 40 are set to 0.

```
If f (x, y) > T
   then f (x, y) = 0
else
   f (x, y) = 255

where
f (x, y) = Coordinate Pixel Value
T = Threshold Value.
```

2. Canny Edge Detection Algorithm :

Canny edge detection is a multi-step algorithm that can detect edges with noise suppressed at the same time. It is implemented as given below:

a. Smooth the image with a Gaussian filter to reduce noise and unwanted details and textures.

$$g(m,n) = G_\sigma(m,n) * f(m,n)$$

Where,

$$G_\sigma = \frac{1}{\sqrt{2\pi\sigma^2}} exp\left(-\frac{m^2+n^2}{2\sigma^2}\right)$$

b. Compute gradient of g (m, n) using any of the gradient operators (Roberts, Sobel, Prewitt, etc.,) to get:

$$M(n,n) = \sqrt{g_m^2(m,n) + g_n^2(m,n)}$$

And,

$$\theta(m,n) = tan^{-1}[g_n(m,n)/g_m(m,n)]$$

c. Threshold M:

$$M_T(m,n) = \begin{cases} M(m,n) & \text{if } M(m,n) > T \\ 0 & \text{otherwise} \end{cases}$$

where 'T' is so chosen that all edge elements are kept while most of the noise is suppressed.

d. Suppress non-maxima pixels in the edges in M_T obtained above to thin the edge ridges (as the edges might have been broadened in step 1). To do so, check to see whether each non-zero M_T (m, n) is greater than its two neighbors along the gradient direction θ (m, n). If so, keep M_T (m, n) unchanged, otherwise, set it to 0.

e. Threshold the previous result by two different thresholds τ_1 and τ_2 (where $\tau_1<\tau_2$) to obtain two binary images T_1 and T_2. Note that T_2 with greater τ_2 has less noise and fewer false edges but greater gaps between edge segments, when compared to T_1 with smaller τ_1.

3. Hough Transform:

To apply the Houghline method, first an edge detection of the specific image is desirable.

Houghline method :

A line can be represented as y = mx + c or in parametric form, as r = xcosθ + ysinθ where r is the perpendicular distance from origin to the line, and θ is the angle formed by this perpendicular line and horizontal axis measured in counter-clockwise (That direction varies on how you represent the coordinate system).

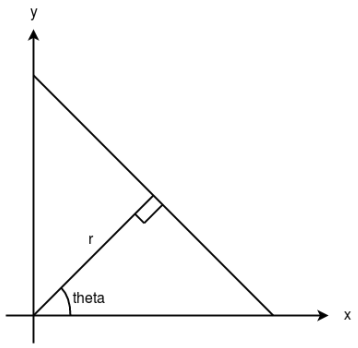


Fig. 3.1 A Straight Line represented using r and θ

So Any line can be represented in these two terms, (r, θ).

Working of Houghline method :

- First it creates a 2D array or accumulator (to hold values of two parameters) and it is set to zero initially.

- Let rows denote the r and columns denote the (θ)theta.
- Size of array depends on the accuracy you need. Suppose you want the accuracy of angles to be 1 degree, you need 180 columns (Maximum degree for a straight line is 180).
- For r, the maximum distance possible is the diagonal length of the image. So taking one pixel accuracy, number of rows can be diagonal length of the image.

4. Slicing an Image using PIL Library in Python :

The steps involved in slicing an image are as follows:

a. Load the image.
b. Calculate the width and height of the image.
c. Slice an image into parts slice_size tall.
d. If we are at the end, set the lower bound to be the bottom of the image.
e. Set the bounding box.
f. Crop the image according to the bounding box.
g. Save the slice.

5. Tesseract OCR Engine :

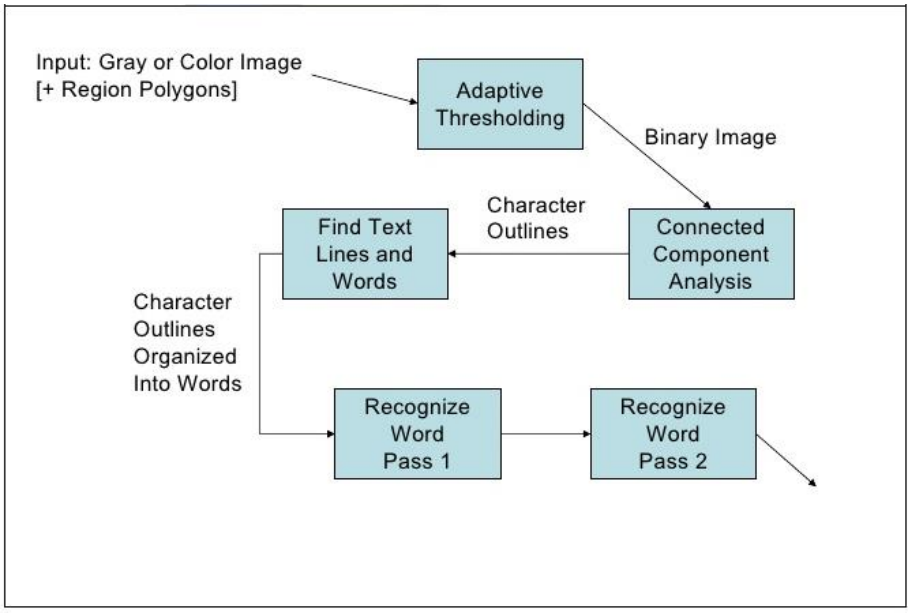


Fig. 3.2 Architecture of Tesseract OCR

The following is a brief overview of how Tesseract works:

1. Outlines are analyzed and stored.
2. Outlines are gathered together as Blobs.
3. Blobs are organized into text lines.

4. Text lines are broken into words.
5. First pass of recognition process attempts to recognize each word in turn.
6. Satisfactory words passed to adaptive trainer.
7. Lessons learned by adaptive trainer employed in a second pass, which attempts to recognize the words that were not recognized satisfactorily in the first pass.
8. Fuzzy spaces resolved and text checked for small gaps.
9. Digital texts are outputted.

During these processes, Tesseract uses:

- Algorithms for detecting text lines from a skewed page.
- Algorithms for detecting proportional and non-proportional words (a proportional word is a word where all the letters have the same width).
- Algorithms for chopping joined characters and for associating broken characters.
- Linguistic analysis to identify the most likely word formed by a cluster of characters.
- Two character classifiers: a static classifier, and an adaptive classifier which employs training data, and which is better at distinguishing between upper and lower case letters.

6. Backtracking:

All the characters read is stored in a file named 'data.txt'. Characters from 'data.txt' is then read to create a Sudoku board. We then use backtracking to solve the sudoku puzzle.

Algorithm:

We can solve Sudoku by assigning numbers one at a time to empty cells. Before assigning a number, we check whether it is safe to assign. We basically check that the same number is not present in the current row, current column and current 3X3 sub grid. After checking for safety, we assign the number, and recursively check whether this assignment leads to a solution or not. If the assignment doesn’t lead to a solution, then we try the next number for the current empty cell. And if none of the number (1 to 9) leads to a solution, we return false.

Finally, after finding the whole solution to the puzzle, we project back the result.

4 Chapter 4 : Experimental Analysis

4.1 Information on the Dataset Used

We have used the Tesseract OCR Engine to detect and recognize the digits in each slice of the image of the Sudoku puzzle. Tesseract is an open source text recognition (OCR) Engine, available under the Apache 2.0 license. It can be used directly, or (for programmers) using an API to extract printed text from images. It supports a wide variety of languages. It can be used with the existing layout analysis to recognize text within a large document, or it can be used in conjunction with an external text detector to recognize text from an image of a single text line.

We use the version 4 of tesseract. The version 4 adds LSTM based OCR engine and models for many additional languages and scripts, and contains up to 116 languages which can be recognized. Additionally, scripts for 37 languages are supported so it is possible to recognize a language by using the script it is written in.

The classifier is able to recognize damaged characters easily, therefore the classifier is not trained on damaged characters. For the English Language, it is trained on a mere 20 samples of 94 characters from 8 fonts in a single size, but with 4 attributes (normal, bold, italic, bold italic), making a total of 60160 training samples.

Tesseract library contains many parameters. Since we are detecting only the numbers, we select language as English. The parameter ‘psm’ is selected as 6 which assumes a single uniform block of text since we detect only a single number at a time from each slice. Finally we add only the digits i.e., 0-9 to the whitelist of Tesseract which is done as follows : tessedit_char_whitelist=0123456789 so that it detects and recognizes only the digits in the image. If the image does not contain any numbers, we consider it as 0.

4.2 Experimental Settings

We have implemented our project using the following system configurations :

1. **CPU** **:** Intel Core i5-6200U CPU @ 2.30 GHz
2. **Memory** **:** 8 GB DDR4 RAM
3. **Hard Disk** **:** 500 GB
4. **OS** **:** Ubuntu 18.04
5. **Programming Language** **:** Python
6. **IDE** **:** PyCharm

We also have other requirements which have to be met such as:

1. The input image should not be captured by a digital camera.

2. The input image should be a screenshot of the Sudoku puzzle which is available on the internet.
3. The perspective of the input image should be 2-dimensional and should not be at an angle.
4. The illumination of the image should not change.
5. The input image should be in a proper orientation and should not be rotated at any angle.

4.3 Result Table

Sl. No.	Image File	Minimum Accuracy (%)	Maximum Accuracy (%)	Average Accuracy (%)	Sudoku Solved
1.	'1.png'	91	98	96.8148	Yes
2.	'2.png'	91	98	97.0617	Yes
3.	'3.png'	91	98	97.0617	Yes
4.	'4.png'	91	98	97.0864	Yes
5.	'5.png'	95	98	97.4074	Yes
6.	'6.png'	92	98	97.0617	Yes
7.	'7.png'	92	98	97.1111	Yes
8.	'8.png'	92	98	97	Yes

4.4 Discussion on Results

As we can see from the result table, the accuracy of the recognized digits is always above 90%. This means that almost all the digits which are recognized using Tesseract OCR are correct. In all the cases, the Sudoku puzzle is solved successfully.

The high accuracy of the dataset is because we have converted all the colour images to grayscale. The Tesseract OCR performs very well on grayscale images.

Tesseract's output will have very poor quality if the input images are not pre-processed to suit it : Images (especially screenshots) must be scaled up such that the text x-height is at least 20 pixels, any rotation or skew must be corrected or no text will be recognized, low-frequency changes in brightness must be high-pass filtered, or Tesseract's binarization stage will destroy much of the page, and dark borders must be manually removed, or they will be misinterpreted as characters.

Each image is recognized perfectly by the Tesseract OCR which directly reflects on the solution of the Sudoku puzzle which is always correct. Hence, we can say that the Tesseract OCR will give the right output almost 90% of the time in our case.

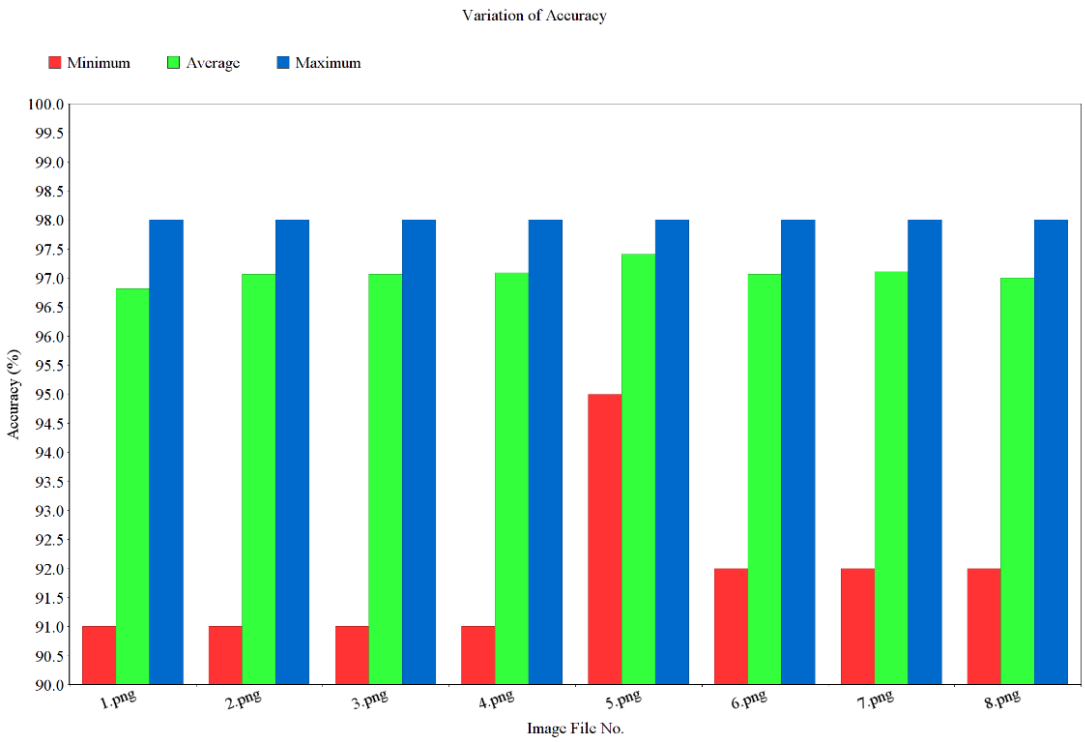


Fig. Graph showing Variation of Accuracy for different Images

We have also illustrated a graph containing the variation of accuracy values which shows the accuracy with which the digits are recognised by the Tesseract OCR Engine.

4.5 Complexity of the algorithms

The algorithms we have used are as follows:

1. Canny Edge Detection Algorithm
2. Hough Transform
3. Tesseract OCR
4. Backtracking

1. Canny Edge Detection Algorithm :

Canny edge detection consists of

1. A convolution of the image with a blur kernel,
2. Four convolutions of the image with edge detector kernels,
3. Computation of the gradient direction,
4. Non-maximum suppression, and
5. Thresholding with hysteresis.

Steps (1), (2), (3), and (4) are all implemented in terms of convolutions of the image with kernels of a fixed size. Using the Fast Fourier Transform (FFT), it is possible to implement convolutions in time O (n log n), where n is the number of elements. If the image has dimensions m × n, then the time complexity will be O (mn log mn) for these steps.

The final step works by post-processing the image to remove all the high and low values, then dropping all other pixels that aren't near other pixels. This can be done in time O (mn).

Therefore, the overall time complexity is **O (mn log mn)**, where m and n are the dimensions (width and height) of the image.

2. Hough Transform :

Using concepts from number theory we can show that this algorithm finds every line component in $\mathbf{O(N^4)}$ time, where the size of the image is $\mathbf{O(N^2)}$ where 'N' is the dimensions of the image.

The Hough transform is only efficient if a high number of votes fall in the right bin, so that the bin can be easily detected amid the background noise. This means that the bin must not be too small, or else some votes will fall in the neighboring bins, thus reducing the visibility of the main bin.

Also, when the number of parameters is large (that is, when we are using the Hough transform with typically more than three parameters), the average number of votes cast in a single bin is very low, and those bins corresponding to a real figure in the image do not necessarily appear to have a much higher number of votes than their neighbors. The complexity increases at a rate of $\mathbf{O(A^{m-2})}$ with each additional parameter, where 'A' is the size of the image space and 'm' is the number of parameters. Thus, the Hough transform must be used with great care to detect anything other than lines or circles.

Finally, much of the efficiency of the Hough transform is dependent on the quality of the input data : the edges must be detected well for the Hough transform to be efficient. Use of the Hough transform on noisy images is a very delicate matter and generally, a denoising stage must be used before.

3. Tesseract OCR :

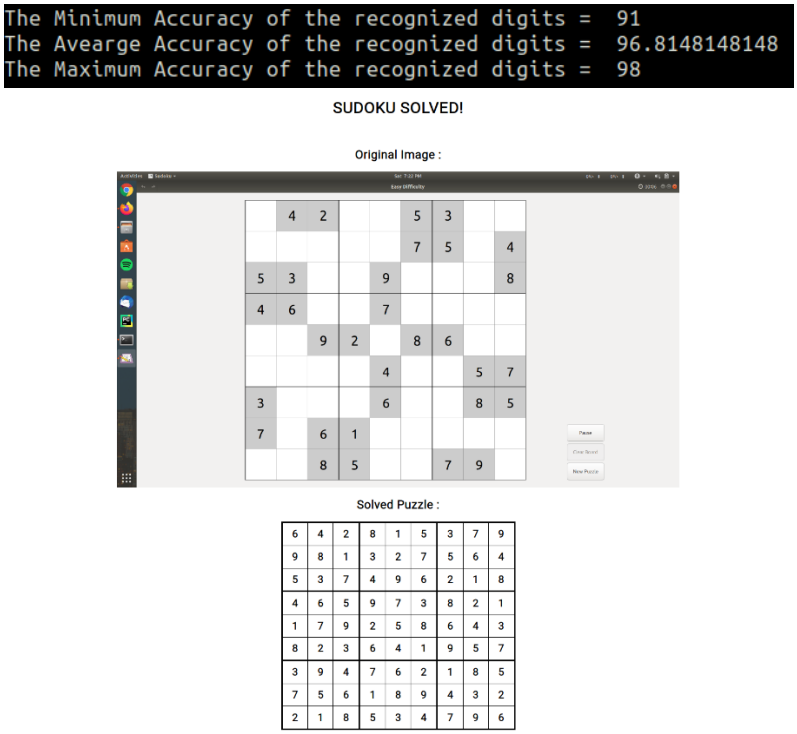
The Tesseract OCR uses a neural network approach. The time complexity of the Tesseract OCR approximately varies between $\mathbf{O(N^4)}$ and $\mathbf{O(N^5)}$, where 'N' is number of operations involved in the classification and detection. The complexity also depends on how we have implemented the algorithm. Hence the runtime will scale linearly in the number of output nodes, quadratically in the number of hidden states, and linearly in the number of output nodes.

4. Backtracking :

The time complexity for solving the sudoku puzzle using backtracking approach is **O(*n* ^ *m*)** where *n* is the number of possibilities for each square (i.e., 9 in classic Sudoku) and *m* is the number of spaces that are blank.

4.6 Snapshots of the Results

1. Image-1 :



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

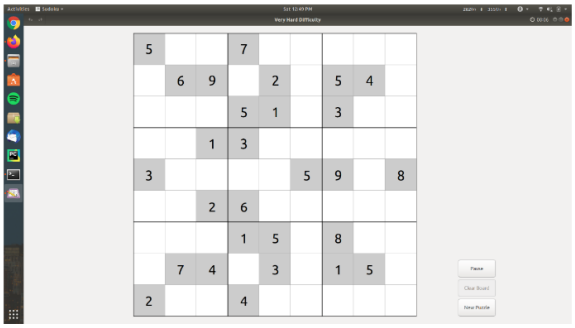
Fig. 4.1 Output for Image-1

2. Image-2 :

```
The Minimum Accuracy of the recognized digits =  91
The Avearge Accuracy of the recognized digits =  97.0617283951
The Maximum Accuracy of the recognized digits =  98
```

SUDOKU SOLVED!

Original Image :



Solved Puzzle :

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

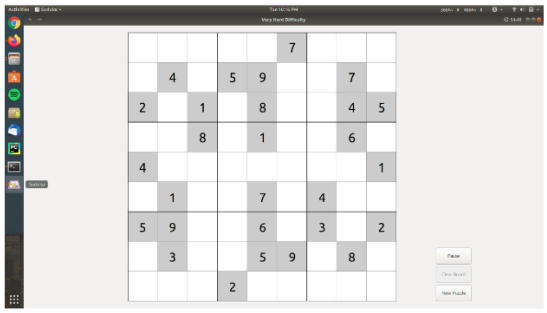
Fig. 4.2 Output for Image-2

3. Image-3 :

```
The Minimum Accuracy of the recognized digits =  91
The Avearge Accuracy of the recognized digits =  97.0617283951
The Maximum Accuracy of the recognized digits =  98
```

SUDOKU SOLVED!

Original Image :



Solved Puzzle :

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

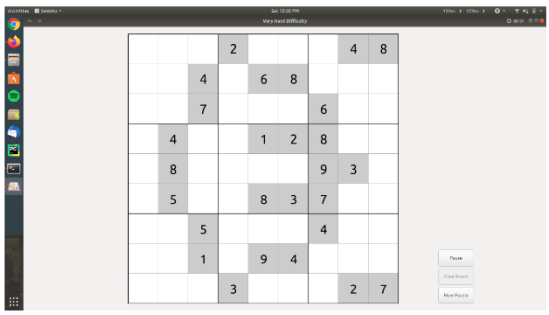
Fig. 4.3 Output for Image-3

4. Image-4 :

```
The Minimum Accuracy of the recognized digits =  91
The Avearge Accuracy of the recognized digits =  97.0864197531
The Maximum Accuracy of the recognized digits =  98
```

SUDOKU SOLVED!

Original Image :



Solved Puzzle :

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

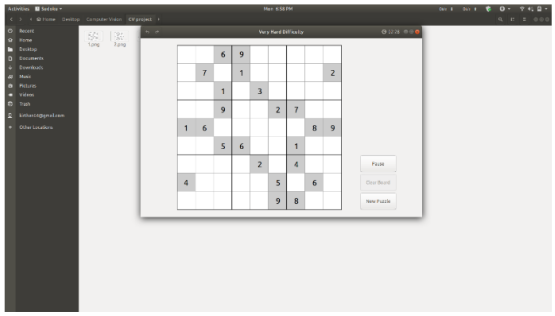
Fig. 4.4 Output for Image-4

5. Image-5 :

```
The Minimum Accuracy of the recognized digits =  95
The Avearge Accuracy of the recognized digits =  97.4074074074
The Maximum Accuracy of the recognized digits =  98
```

SUDOKU SOLVED!

Original Image :



Solved Puzzle :

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

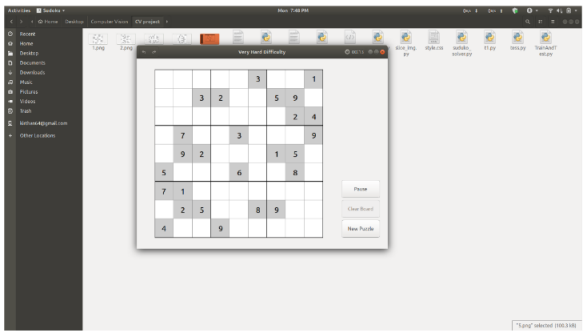
Fig. 4.5 Output for Image-5

6. Image-6 :

```
The Minimum Accuracy of the recognized digits =  92
The Avearge Accuracy of the recognized digits =  97.1604938272
The Maximum Accuracy of the recognized digits =  98
```

SUDOKU SOLVED!

Original Image :



Solved Puzzle :

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

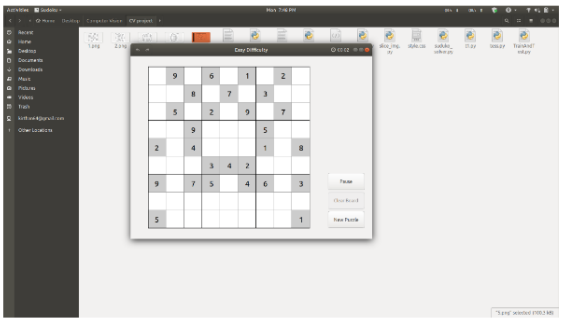
Fig. 4.6 Output for Image-6

7. Image-7 :

```
The Minimum Accuracy of the recognized digits =  92
The Avearge Accuracy of the recognized digits =  97.1111111111
The Maximum Accuracy of the recognized digits =  98
```

SUDOKU SOLVED!

Original Image :



Solved Puzzle :

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

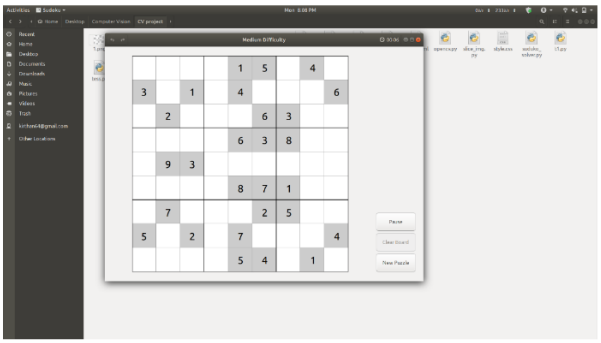
Fig. 4.7 Output for Image-7

8. Image-8 :

```
The Minimum Accuracy of the recognized digits =  92
The Avearge Accuracy of the recognized digits =  97.0
The Maximum Accuracy of the recognized digits =  98
```

SUDOKU SOLVED!

Original Image :



Solved Puzzle :

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

Fig. 4.8 Output for Image-8

5 Chapter 5 : Conclusion and Future Scope

Sudoku, as we discussed, is a popular Japanese puzzle game. This project aims to solve the Sudoku puzzle from just an image. We can input an image containing of a Sudoku puzzle to the project and it will solve and display the complete solution to the user. This is significant because majority of the current Sudoku-solving solutions require the user to manually input numbers. Scanning the puzzle makes this process faster to get a solution to it. Since the scale of the image also varies from image to image, our algorithm efficiently manages these problems. Also the Tesseract OCR engine has proved capable of correctly recognizing the digits as shown from the results table in which the accuracy is always above 90%. Since the time and memory requirement differs for various algorithms, we tested these algorithms for various difficulty levels, ranging from easy level to hard level.

We reviewed different techniques implemented by some researchers along with their advantages and disadvantages. Because of constant evolution in computation and algorithms, some methods can get outdated fairly fast such as static template matching while recognizing digits. So it is necessary to update these techniques to keep up with modern computational practices. Methods like detection of characters using machine learning can provide a faster and efficient way.

Future works can be carried out in improving the efficiency and time complexity of the algorithms used in these projects. In this project, we have only dealt with images which are already perfectly in position to carry out the image processing tasks without needing to rotate or illuminate it. Future works may also be concerned in implementing our algorithms for solving images captured by a digital camera which may be even more challenging considering the perspective of the image, illumination changes across the image and images which may also be rotated.

6 References

[1] Ray Smith. July 2007. *"An Overview of the Tesseract OCR Engine"*. In the proceedings of Conference on Document Analysis and Recognition (ICDAR), IEEE Computer Society (2007), pp. 629-633.

[2] Sian K Jones, Stephanie Perkins, and Paul A Roach. Nov 2007. *"Properties of Sudoku Puzzles"*. In the proceedings of the 2nd Research Student Workshop, University of Glamorgan.

[3] Timo Mantere, Janne Koljonen. July 2008. *"Solving and Analyzing Sudokus with Cultural Algorithms"*. In the proceedings of Evolutionary Computation. CEC 2008.

[4] P.J. Simha, K.V. Suraj, T. Ahobala. Jan 2012. *"Recognition of numbers and position using image processing techniques for solving Sudoku Puzzles"*. In the proceedings of Advances in Engineering, Science and Management (ICAESM).

[5] Atul Patel, PhD, Chirag Patel, Dharmendra Patel. Oct 2012. *"Optical Character Recognition by Open Source OCR Tool Tesseract: A Case Study"*. International Journal of Computer Applications. Vol. 55.

[6] Xiuqin Deng, Junhao Li, Guangqing Li. March 2013. *"Research on Sudoku Puzzles Based on Metaheuristics Algorithm"*. Journal of Modern Mathematics Frontier. Vol. 2, Issue. 1.

[7] Rohit Iyer, Amrish Jhaveri, Krutika Parab. May 2013. *"A Review of Sudoku Solving using Patterns"*. In the proceedings of International Journal of Scientific and Research Publications, Vol. 3, Issue. 5.

[8] Prof. S. T. Khandare, Mr. Akshay D. Isalkar. Jan 2014. *"A Survey Paper on Image Segmentation with Thresholding"*. In the proceedings of International Journal of Computer Science and Mobile Computing, Vol. 3 Issue. 1, pp. 441-446.

[9] Arnab Maji, Rajat Pal. Feb 2014. *"Sudoku solver using minigrid based backtracking"*. In the proceedings of 2014 IEEE International Advance Computing Conference (IACC).

[10] Sankhadeep Chatterjee, Saubhik Paladhi, Raktim Chakraborty. Oct 2014. *"A Comparative Study On The Performance Characteristics Of Sudoku Solving Algorithms"*. In the proceedings of IOSR Journal of Computer Engineering (IOSR-JCE). Vol. 16 Issue. 5.

[11] Honglei Zhao, X. S. Ding. Dec 2014. *"On the generation and evaluation of a complete sudoku puzzle"*. In the proceedings of Computer Science and Applications – HU (Ed.) 2014 Taylor & Francis Group, London.

[12] Allam Shehata Hassanein, Sherien Mohammad, Mohamed Sameer, Mohammad Ehab Ragab. Jan 2015. *"A Survey on Hough Transform, Theory, Techniques and Applications"*. IJCSI Vol. 12 Issue. 1.

[13] Prof. R. N Mandavgane, Prof. D. M. Khatri, Ms. Anjum Sheikh. Feb 2015. *"Review On Canny Edge Detection"*. IJAICT Vol. 1 Issue. 9.

[14] Snigdha Kamal, Simarpreet Singh Chawla, Nidhi Goel. Dec 2015. *"Detection of Sudoku puzzle using image processing and solving by Backtracking, Simulated Annealing and Genetic Algorithms: A comparative analysis"*. In the proceedings of 2015 Third International Conference on Image Information Processing (ICIIP).

[15] Akhil S Nair. Dec 2016. *"Overview of Tesseract OCR engine"*. In the proceedings of National Institute of Technology, Calicut.

[16] Paritosh Pujari, Saurabh Thorat , Suraj Thopate , Yash Shah , Prasad Kharade. Dec 2017. *"Sudoku Puzzle Detection using Image Processing Techniques and Solving: A Literature Survey"*. In the proceedings of International Journal of Innovative Research in Computer and Communication Engineering, Vol. 5 Issue. 12.

[17] Onokpasa Eva, Bakwa Dunka. Feb 2019. *"A Comparison of Three Sudoku Solving Methods"*. In the proceedings of International Journal of Computer Applications, Vol. 181.