**Annexure 2**

# Introduction to Image Processing Lab Manual

# CSL 316

Project Report


Faculty name - Dr. Ashwini Rahangdale

**Students Name** -

Aarushi Dua (18CSU002)

Abhishek Bhatia (18CSU008)

Bhavya Kalra (18CSU045)

**Semester**: VI

Department of Computer Science and Engineering

The NorthCap University, Gurugram- 122001, India

Session 2020-21

# Table of Contents

## Project Description:

Sudoku is one of the most popular puzzles of all time. The goal of Sudoku is to fill a 9x9 grid such that each row, each column and 3x3 grid contains all of the digits between 1 to 9.In this project we aim to create a real time Sudoku solver which recognizes the elements of Sudoku puzzles and provides a digital solution using Computer vision.Sudoku Solver is the collection of very basic image processing techniques. A very good way to start is the OpenCV library which can be compiled on almost all the platforms.

## Problem Statement:

The intuition and knowledge of increasing your mental strength of this new generation is decreasing day by day to overcome one such problem and to provide answers instantly to users. We have made a Real-Life Sudoku Solver. It can be used to solve Sudoku in a smarter and more easy way with the help of Image processing and solve sudoku problems instantly for the enthusiasts, beginners and even professionals who consider this challenging. The need of the hour to make this project was due to the fact that in India most of the people who play sudoku use the newspaper platform to play it and eventually they have to wait for a whole day to get the answers of the previous day sudoku which they solved. But not anymore with the help of Computer vision we have created a way for them to get their answers instantly by using our technology.

## Analysis:

## 3.1 Hardware Requirements

Unsolved Sudoku (9*9) from a sheet of Paper or phone screen .

## 3.2 Software Requirements

We have used OpenCV for image processing. Therefore Basic knowledge of opencv regarding use applications and syntax are essential here. It is a library in python that helps with computer vision and is more convenient as it provides a large number of functions for conversions and processing. It can be learned on opencv.org.
Numpy for numeric handling. Basic knowledge of numpy including syntax and computational parts are essential to form image arrays. It is a library in python used to perform mathematical computation. it can be learned from numpy.org.
Tersseract for Python used in image to text conversion. This is also known as pytersseract and is used for object recognition. Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images. It can be learned from pype.org.

## Design:

## 4.1 Data/Input Output Description:

At the time of input we are going to pass a video frame of an unsolved sudoku. The frame acts like continuous real time images.It is passed into each image preprocessing steps mentioned in the next section to get our output of Solved sudoku. The output is going to be an image of the sudoku grid extracted from the video frame and the blanked boxes are now filled with numbers which are highlighted in green along with the given number of Unsolved sudoku.

## 4.2 Algorithmic Approach / Algorithm / DFD / ER diagram/Program Steps

Majorly there are **Three main steps** to solve our problem.

- **The first one** is to extract the Sudoku grid from our webcam Image and Fig 1 shows how to proceed step by step in achieving this task.

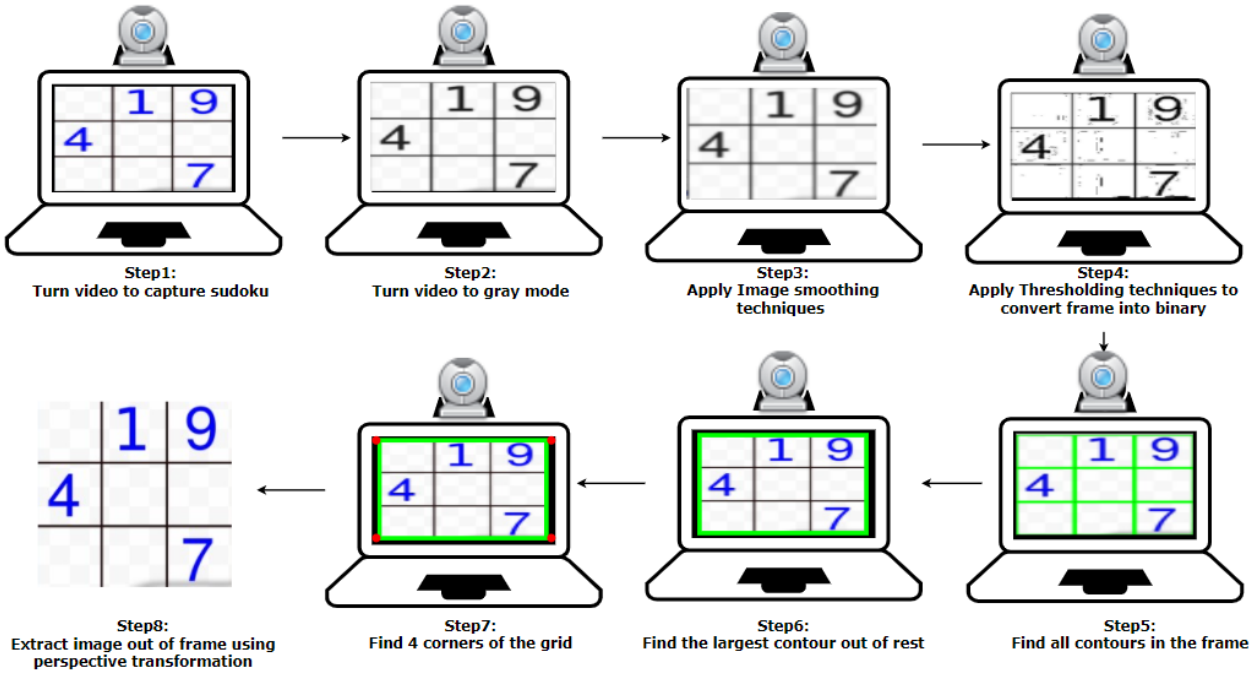



**Fig:1 Extracting Sudoku grid from our webcam Image.**

- **The second step** is to preprocess the extracted image in order to Detect the Digits using pytesseract. Fig 2 shows the steps involved to detect the grid numbers.

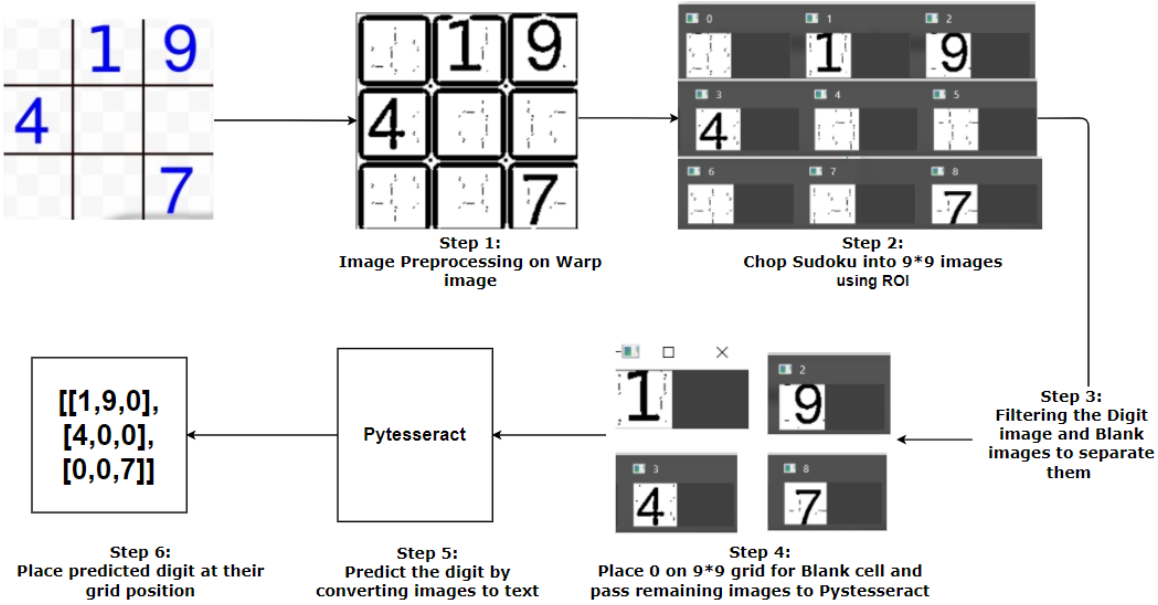



**Fig:2 Detecting Digits from Extracted Image**

- **The third step** is to solve the Sudoku puzzle and print the solution back on the warp image.To solve we have used Breadth First Search (BFS) Algorithm

1.Firstly check wether the given grid is violating sudoku or not
2.In the loop, start with box with least possible choice and then fill it with remaining option
3.Before assigning a number, check whether it is safe to assign. Check that the same number is not present in the current row, current column and current 3X3 subgrid. After checking for safety, 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 try the next number for the current empty cell. Loop till all grid becomes non zero.

## Implementation and Testing ( Stage/Module wise)

**We have carried out our Implementation in Three Steps**

**Steps :**

1. We take the Sudoku grid from our webcam Image.
2. We Extract and Detect the Digits.
3. Solve the puzzle and Print the solution.

**<u>STEP 1: Grab the Sudoku grid from Webcam Image.</u>**

- Capture the video frame continuously(cv2.videoCapture)
- We take the input frame and convert it to gray scale(cv2.cvtColor).
- Blur the Image using Gaussian Blurring (cv2.gaussianBlur).
- Apply Adaptive Thresholding (cv2.adaptiveThreshold).
- We get a binary Image.
- Then we Find all contours(cv2.findContours).
- Extract the contour with the biggest area (cv2.contourArea).
- Locate 4 corners of the contour and determine the upper left, upper right, bottom left, bottom right corners.
- We have used 2 Criteria for Qualifying a Sudoku grid i.e. 4 angles must be approximately 90 degrees and 4 sides must have approximately equal lengths .
- If the biggest contour is a square(required), from 4 corners, we apply cv2.wrapPerspective to get a nice and able Sudoku Grid Image.

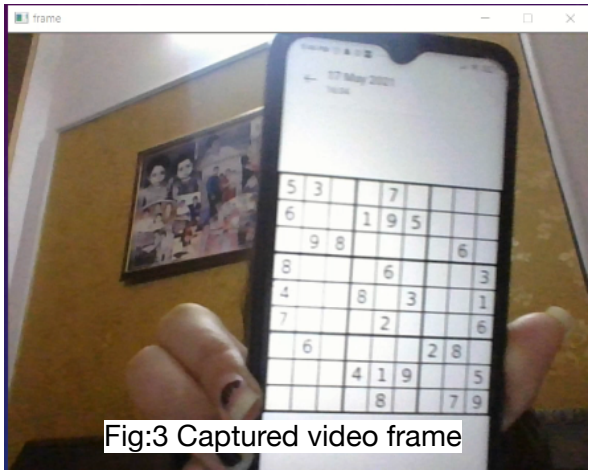


Fig:3 Captured video frame

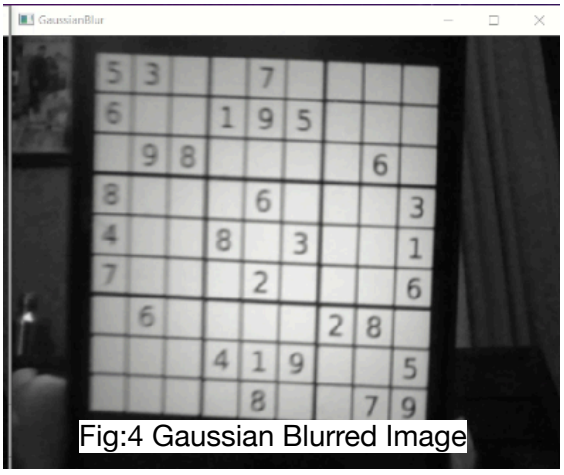


Fig:4 Gaussian Blurred Image

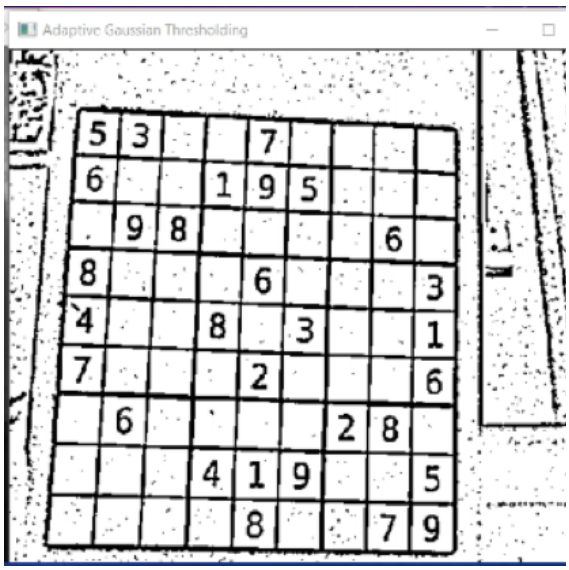



Fig:5 Adaptive Thresholding

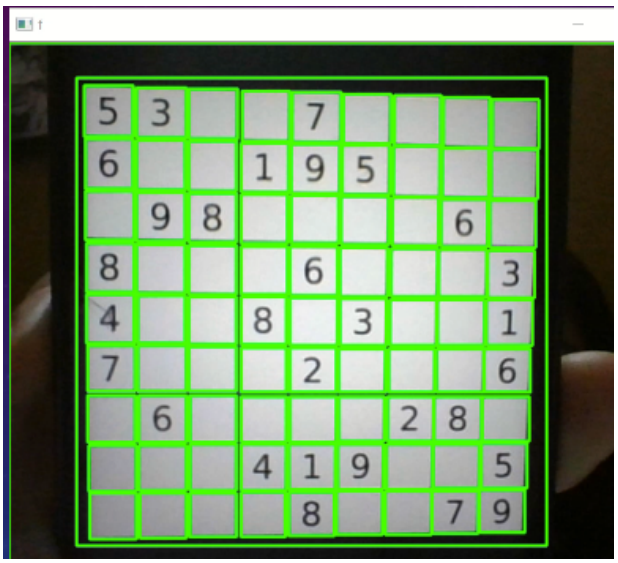



Fig:6 Find All Contours

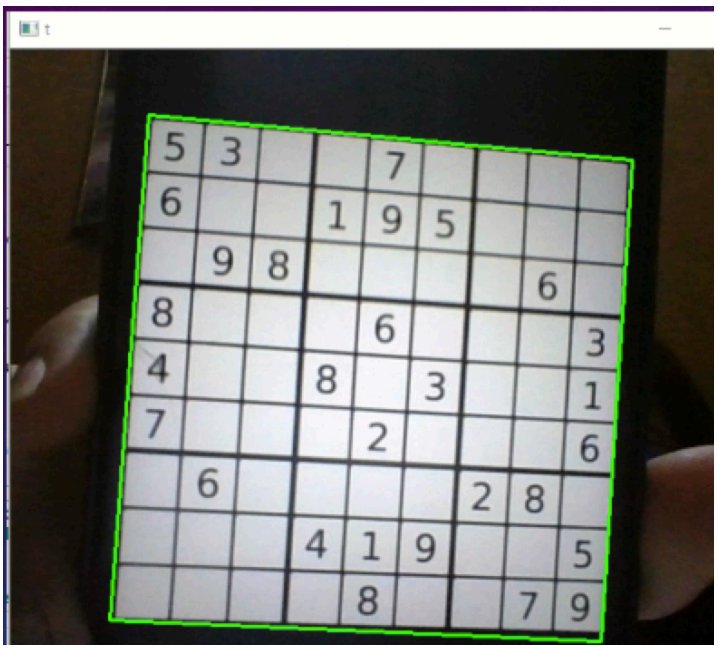



Fig:7 Contour with Biggest area

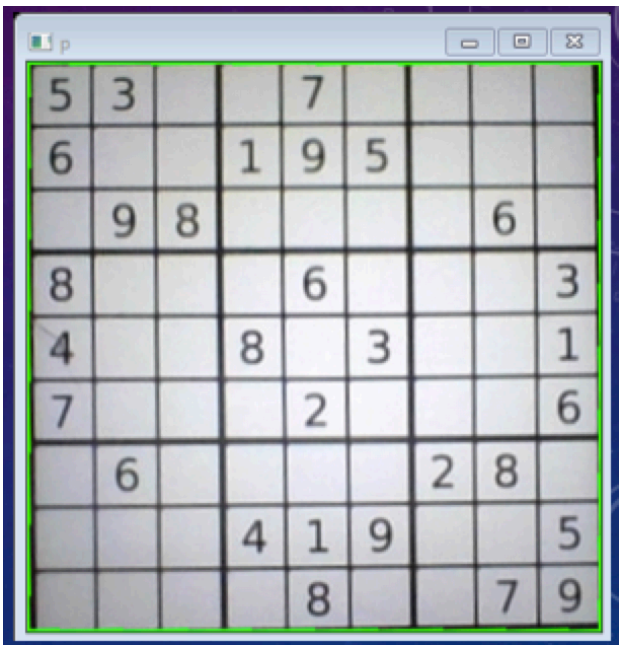



Fig:8 Extracted Grid out of Frame

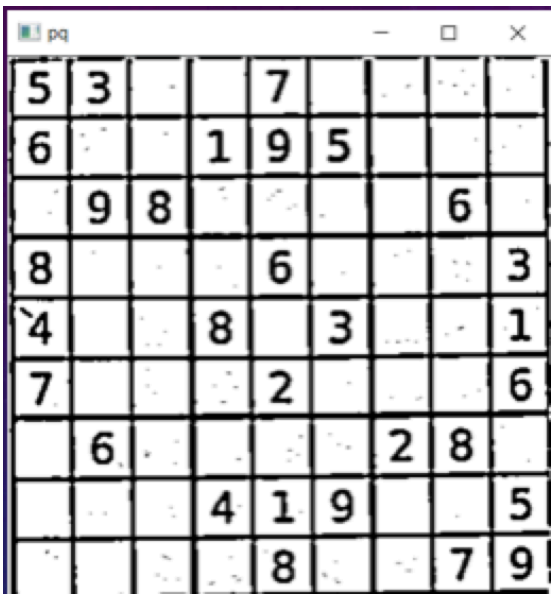
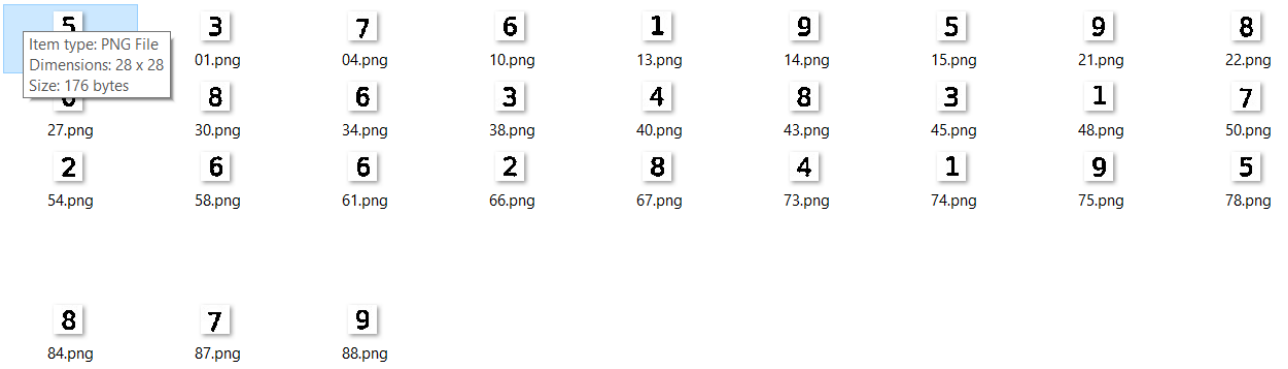



Fig:9 Sudoku Grid Image

## STEP 2: Extract and Detect Digits.

- This Step will begin by Image processing and chopping Sudoku create images into 9x9 Square blocks .
- As there is so much noise in these images and if we feed them into Model now the accuracy will be about 10% (which is random).
- So Before feeding them into for Digit Recognition we need to clean them up first .
- Remove Black lines near 4 edges (if any) .
- Take only the largest connected component (cv2.connectedComponentWithStats) , which should correspond to digit and turn the rest into white pixels .
- Resize these images to 28 x 28
- We leave out any white cells
- Criteria 1: The image has too little black pixels (sum of all pixels are too big)
- Criteria 2: The image has a huge white area in the centre .
- After applying these two criteria we should be able to remove all white cells
- Now we centred upon - empty images by its **centre of mass** we will do the same thing on the training data set to make sure the image structure is similar .
- To get the best result I centred train image by **Centre of Mass.** I did the same thing (as we mentioned before) to our Sudoku digital images. This improve the accuracy to **more than 99% .**
- Finally , let's recognise those digits !
- With the advanced technology we have today , this was a simple task .
- We wanted to use Convolutional Neural Network (CNN) but were not able to feed in images so we have used pytesseract .
- Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and “read” the text embedded in images.Python-tesseract is a wrapper for Google’s Tesseract-OCR Engine. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.




```
grid
```

```
[[0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0],
 [0, 0, 0, 0, 0, 0, 0, 0, 0]]
```

```
grid
```

```
[[8, 0, 0, 0, 4, 0, 0, 0, 4],
 [0, 5, 0, 8, 0, 4, 0, 4, 0],
 [0, 0, 6, 0, 4, 0, 7, 0, 0],
 [0, 6, 0, 4, 0, 4, 0, 2, 0],
 [8, 0, 8, 0, 6, 0, 4, 0, 4],
 [0, 4, 0, 5, 0, 2, 0, 2, 0],
 [0, 0, 8, 0, 2, 4, 6, 0, 0],
 [0, 2, 0, 3, 0, 8, 0, 4, 0],
 [3, 0, 4, 0, 5, 0, 0, 0, 8]]
```

**STEP 3: Solve the puzzle and Print the solution.**

- The third step is to solve the Sudoku puzzle and print the solution back on the warp image.
- After the Digits are being Recognised as Text , in the Solving part we have used Breadth First Search as our sudoku algorithm.
- The simplest way to solve a Sudoku puzzle would be to simply search for the answer one cell at a time. The two most basic methods of search are Depth First(DFS) and Breadth First Search(BFS). From which we have Chosen BFS to solve .This algorithms make use of backtracking once they have explored a branch in their search path sufficiently to go back and expand other paths.
- Then we just Print the solved Sudoku that we got as an output on warp image using result=write_solution_on_image() function .

```
solver.solve_sudoku(predicted_digits)
```

```
predicted_digits
```

```
[[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]]
```

## Output (Screenshots):


This is the Final output of the Solved Sudoku printed on the image taken as first input

## Conclusion and Future Scope:

We have made a Real-Time Sudoku solver to instantly solve and provide answers to people making them learn and check their knowledge of sudoku solving . As most of the people in India play sudoku on the newspapers and eventually have to wait for a whole day to get the answers this software will help them getting the Solution Instantly .We have created this with the Help of Computer vision. When we play and solve sudoku problem in front of the camera it detects the image and with the image filtering techniques convert the image into greyscale and then apply Gaussian blurring and Thresholding so that we get a binary image. We then find all the contours at them to get the biggest area which is the sudoku grid . The main task then is to extract and detect digits which we have done using pytesseract , it recognises and read the text in images . After the extraction of digits we apply the solving algorithm to get the solution of the sudoku problem and then print and on the warp image .

For Future scope we can develop an application based on this which will be user-friendly and also provide better user experience .