Conundrum Unraveller using CNN/KNN

18BCE2237 Tanishq Padwal
18BCE2185 Nikhil Philip Chacko

Abstract

In the last decade, solving the Sudoku puzzle has become every one's passion. The simplicity of puzzle's structure and the low requirement of mathematical skills caused people to have enormous interest in accepting challenges to solve the puzzle.

Many puzzles such as Kakuro, Sudoku, Killer Sudoku etc are solved by people of various age groups on a daily basis. These puzzles come in a variety of difficulty levels and sizes that at times are not solvable or require help from different sources to be completed.

In this project, we have presented a way to solve these complicated puzzles using various Image recognition and preprocessing techniques and algorithms. The purpose is to implement an efficient algorithm and then use image recognition to take input of question and solve the puzzle using CNN/KNN which are machine learning algorithms.

Problem Statement :

Many puzzles such as Kakuro, Sudoku, Killer Sudoku etc are solved by people of various age groups on a daily basis. These puzzles come in a variety of difficulty levels and sizes that at times are not solvable or require help from different sources to be completed.

Thus to make solving such puzzles easier in absence of an internet connection, we will be implementing a puzzle solver using machine learning algorithms such as CNN/KNN and image processing techniques to extract puzzle from image using an user friendly Python based GUI.

Tools and algorithms used :

- **Python 3.8.2**
  - Keras-Applications==1.0.8
  - Keras-Preprocessing==1.1.0
  - numpy==1.17.2
  - opencv-contrib-python==4.1.1.26
  - Pillow==6.2.0
  - scikit-learn==0.21.3
  - scipy==1.3.1
  - sklearn==0.0
  - tensorboard==1.14.0
  - tensorflow==1.15.2
  - tensorflow-estimator==1.14.0

Image preprocessing techniques used :

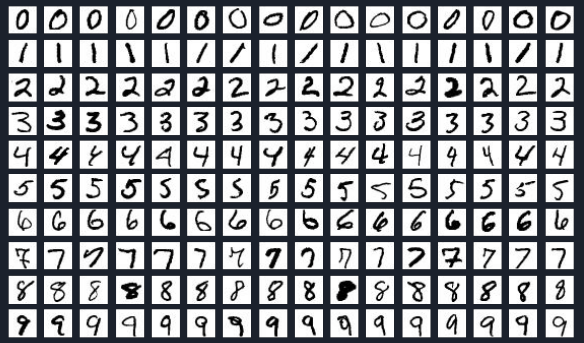
- Gaussian Blurring
- Adaptive Gaussian Thresholding
- Inverting
- Dilation
- Flood Filling
- Hough Line Transform
- Thresholding and Inverting the grid
- Slicing
- Blackfilling and centering the number.

Recognition :

- **Convolutional Neural Network** :
  - A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other.
- **K Nearest Neighbours :**
  - A supervised machine learning algorithm (as opposed to an unsupervised machine learning algorithm) is one that relies on labeled input data to learn a function that produces an appropriate output when given new unlabeled data.

Dataset Used :

MNIST handwritten digits dataset which has around 70,000 28X28 images which includes numbers from 0-9.



- Source :
  - http://yann.lecun.com/exdb/mnist/
  - https://www.kaggle.com/crawford/emnist

- CNN has around 98 percent accuracy on the test set.
- KNN uses K=3 and around 97 percent accuracy on the test set.

Architecture / Working :

Image Preprocessing :

- **Gaussian Blurring** : Blurring using a Gaussian function. This is to reduce noise and detail.
- **Adaptive Gaussian Thresholding** : Adaptive thresholding with a Gaussian Function to account for different illuminations in different parts of the image.
- **Inverting** to make the digits and lines white while making the background black.
- **Dilation** with a plus shaped 3X3 Kernel to fill out any cracks in the board lines and thicken the board lines.
- **Flood Filling** Since the board will probably be the largest blob a.k.a connected component with the largest area, floodfilling from different seed points and finding all connected components followed by finding the largest floodfilled area region will give the board
- The largest blob a.k.a the board is found after the previous step. Let's call this the outerbox

- **Eroding** the grid a bit to undo the effects of the dilation on the outerbox that we did earlier.
- **Hough Line Transform** to find all the lines in the detected outerbox.
- Merging related lines. The lines found by the Hough Transform that are close to each other are fused together.
- **Finding the Extreme lines** . We find the border lines by choosing the nearest line from the top with slope almost 0 as the upper edge, the nearest line from the bottom with slope almost 0 as the lower edge, the nearest line from the left with slope almost infinity as the left edge and the nearest line from the right with slope almost infinity as the right edge.
- **Finding the four intersection points.** The four intersection points of these lines are found and plotted along with the lines

- **Warping perspective.** We find the perspective matrix using the end points, correct the perspective and crop the board out of the original image.
- **Thresholding and Inverting the grid.** The cropped image from the previous step is adaptive thresholded and inverted.
- **Slicing** the grid into 81 slices to get images of each cell of the Sudoku board.
- **Blackfilling and centering the number.** Any white patches other than the number are removed by floodfilling with black from the outer layer points as seeds. Then the approximate bounding box of the number is found and centered in the image.

Problem domain :

These are the fundamental domains that are used :

- **Image Acquisition**
- **Image Enhancement**
- **Image Restoration**
- **Morphological Processing**

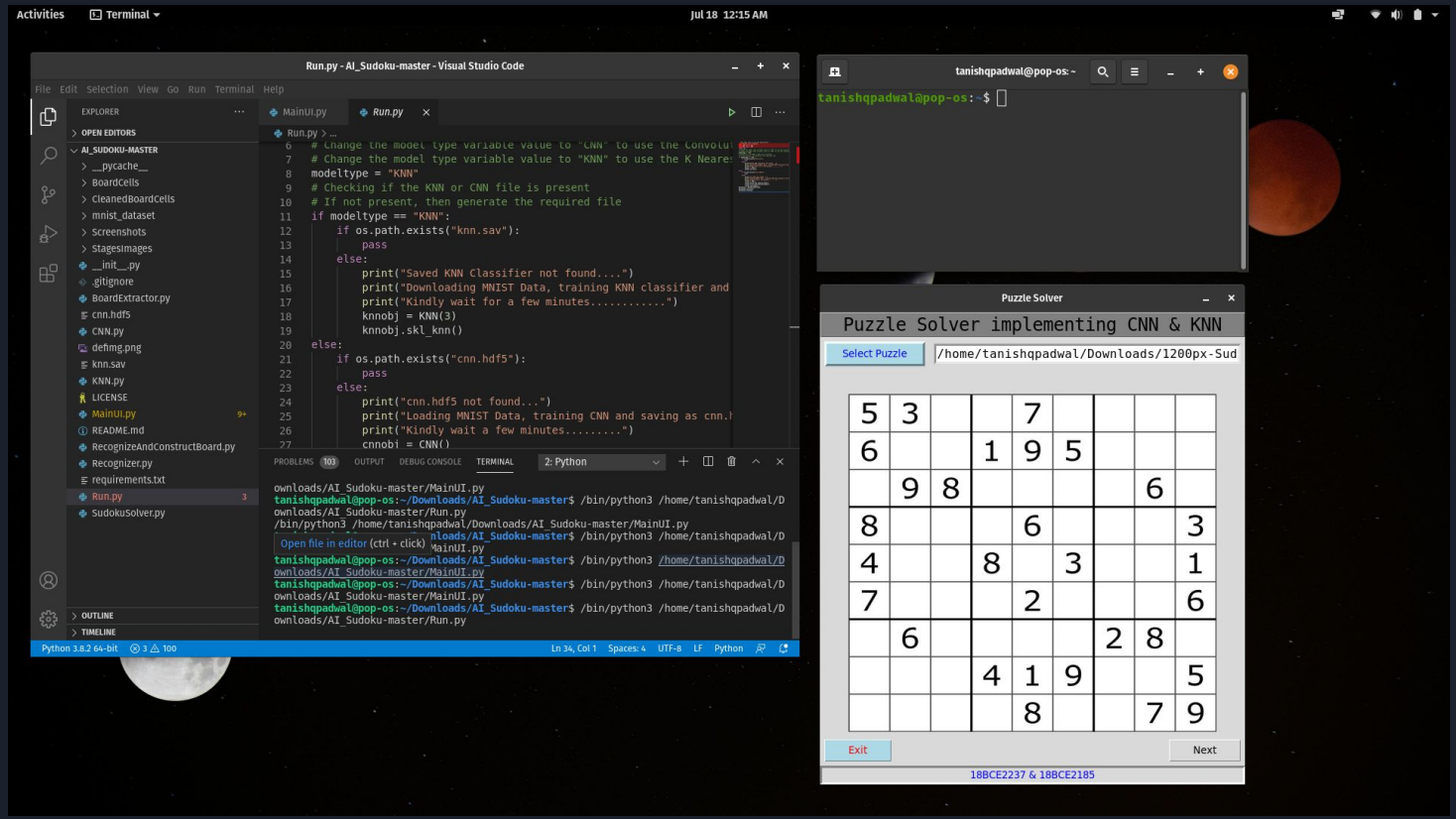
Input :

- The GUI Homepage that opens up as soon as you run the application.
- You need to select an image of a Sudoku Puzzle through the GUI Home Page.
- Once you press Next, a number of stages of image processing take place which are displayed by the GUI leading up to recognition
- For recognition, a CNN or KNN can be used. This option can be toggled. Once recognized, the board is displayed and you can rectify any wrongly recognized entries in the board.
- Finally click on reveal solution to display the solution.

Output :

- After pressing on reveal solution, the stated processing takes place which then returns the solution and missing values from the original Sudoku puzzle which was used.
- All solution values displayed are on the GUI.

Output :

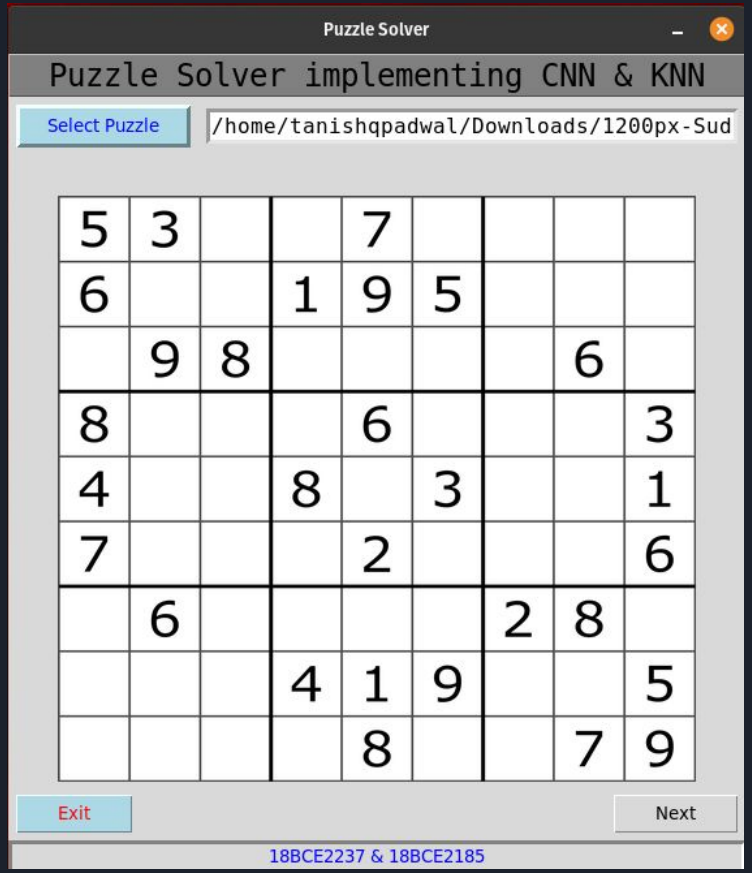

Activities
Terminal
Jul 18 12:15 AM
Run.py - AI_Sudoku-master - Visual Studio Code
tanishqpadwal@pop-os: ~
Puzzle Solver
Puzzle Solver implementing CNN & KNN
Select Puzzle
/home/tanishqpadwal/Downloads/1200px-Sud
Exit
Next
18BCE2237 & 18BCE2185

Select Puzzle :

Allows user to select puzzle

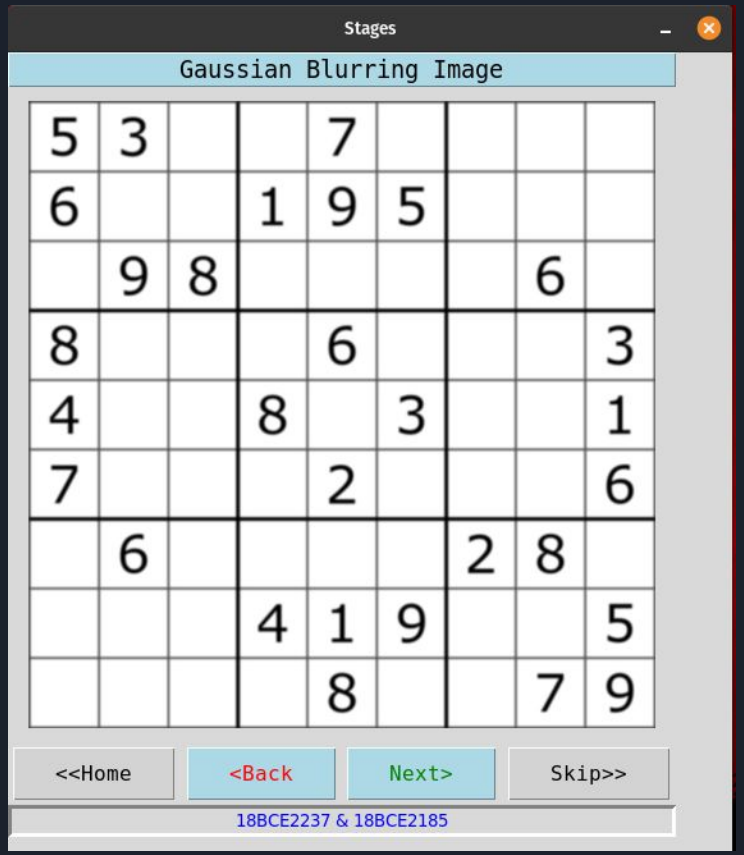
as needed.

CD into folder as needed.



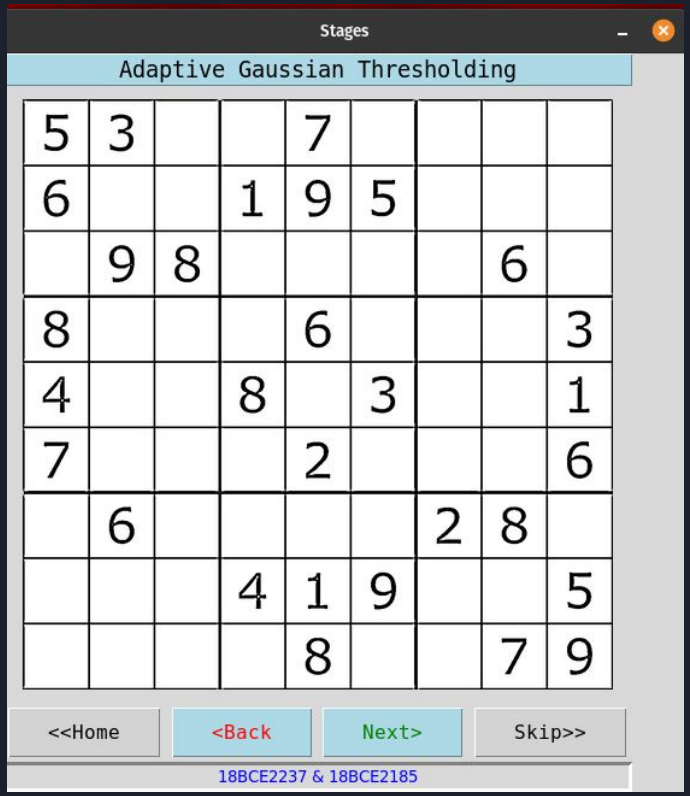
Gaussian Blurring :

- Blurring using a Gaussian function
  This is to reduce noise and detail.
- As we can see, the image looks slightly blurred.



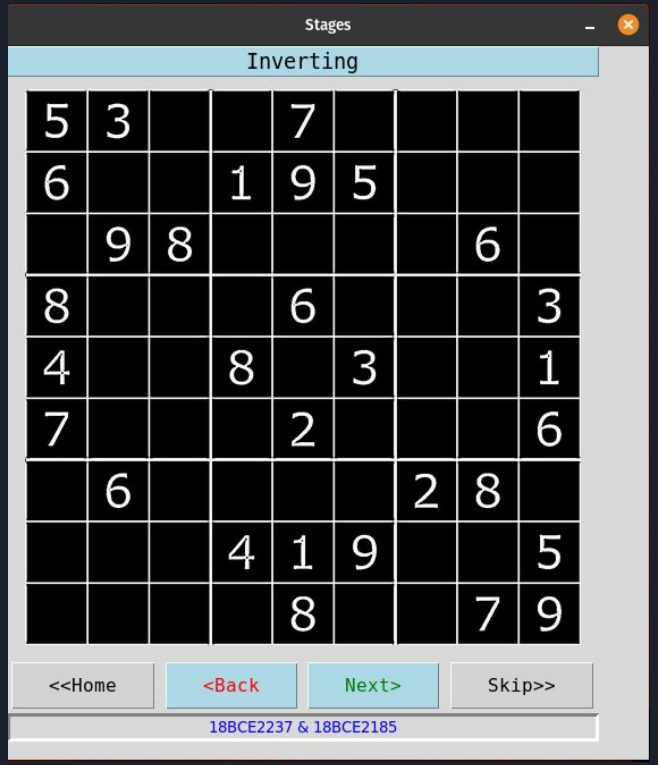
Adaptive Gaussian :

- In Simple Thresholding, a global value of threshold was used which remained constant throughout. So, a constant threshold value won't help in the case of variable lighting conditions in different areas. Adaptive thresholding is the method where the threshold value is calculated for smaller regions. This leads to different threshold values for different regions with respect to the change in lighting.



Inverting :

image inversion refers to where light areas are mapped to dark, and dark areas are mapped to light.



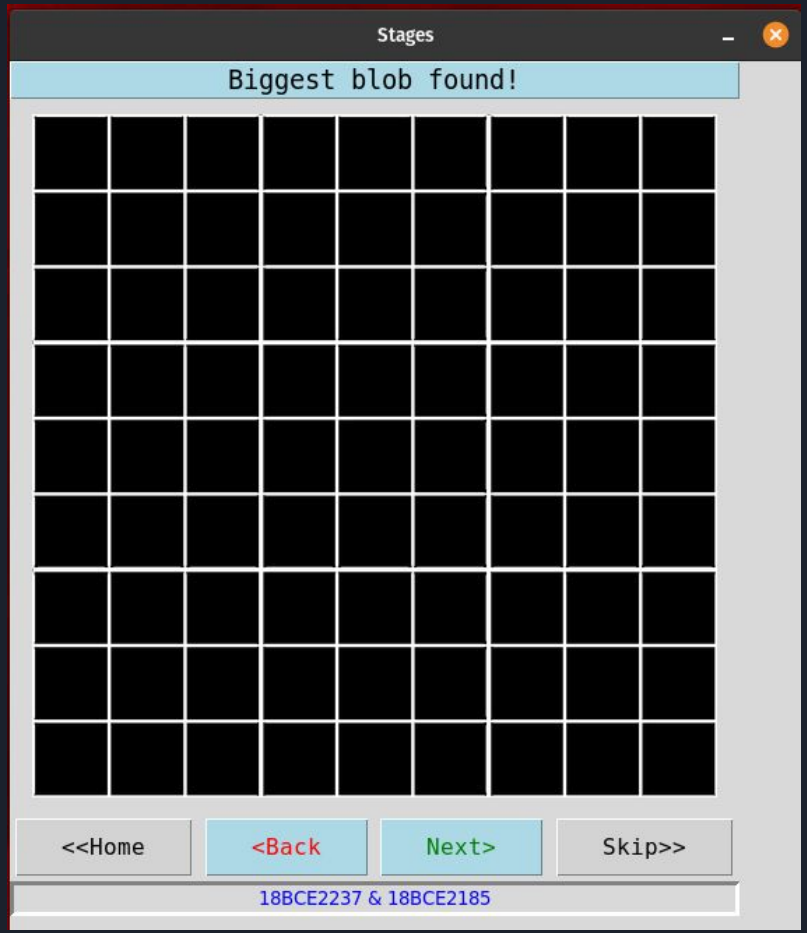
Flood filling :

It connects parts and determines the area connected to a given node in a multi-dimensional array.

Used in bucket fill tool on MSPaint

We find the largest blob

Biggest blob found after flood filling

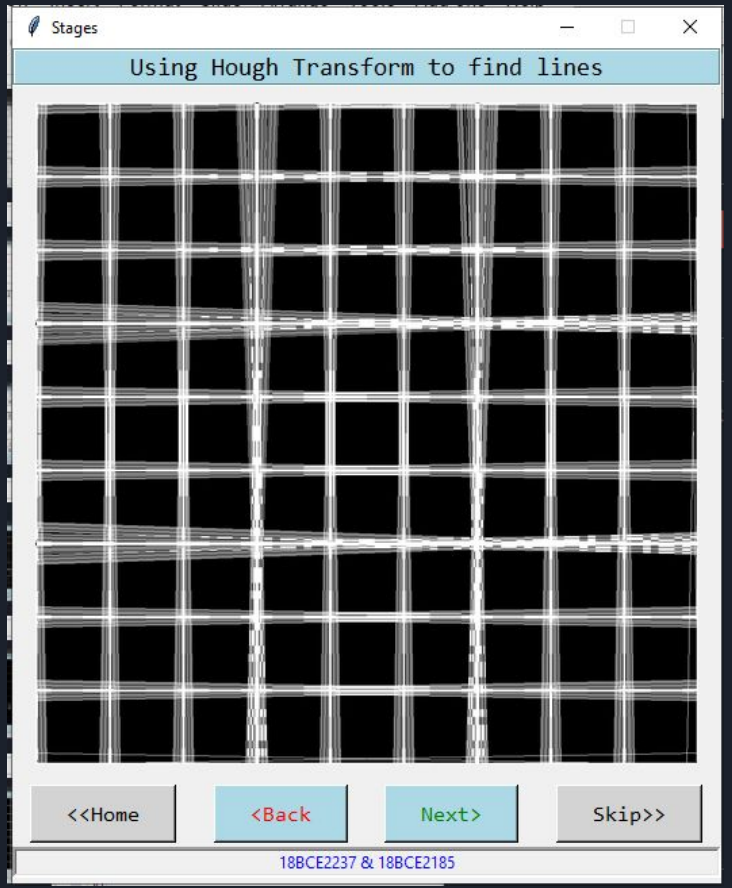


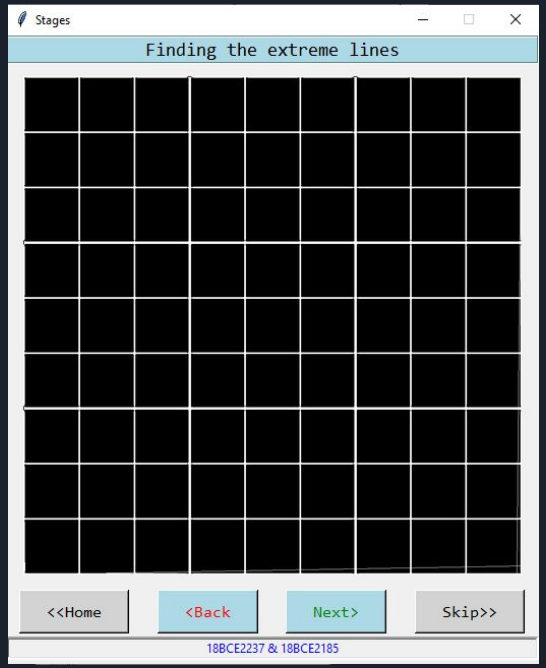
Hough transform :

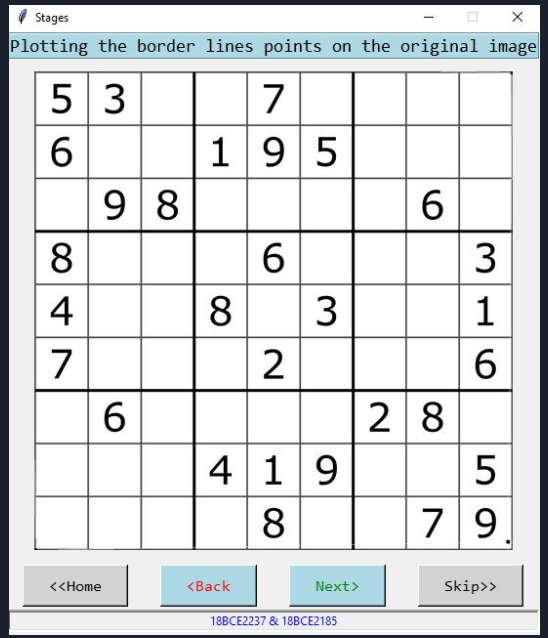
Hough transform is a feature extraction technique

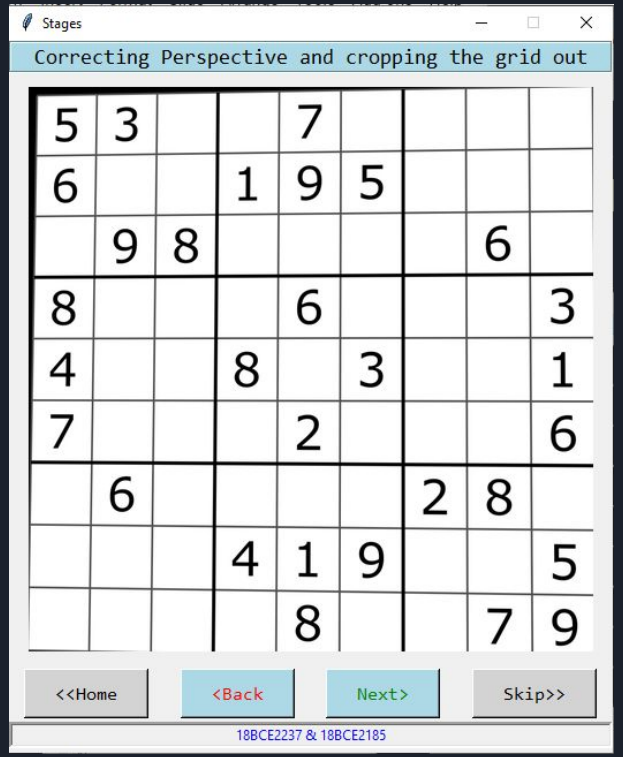
The Hough Line Transform is a transform used to detect straight lines.

To apply the Transform, first an edge detection pre-processing is desirable.



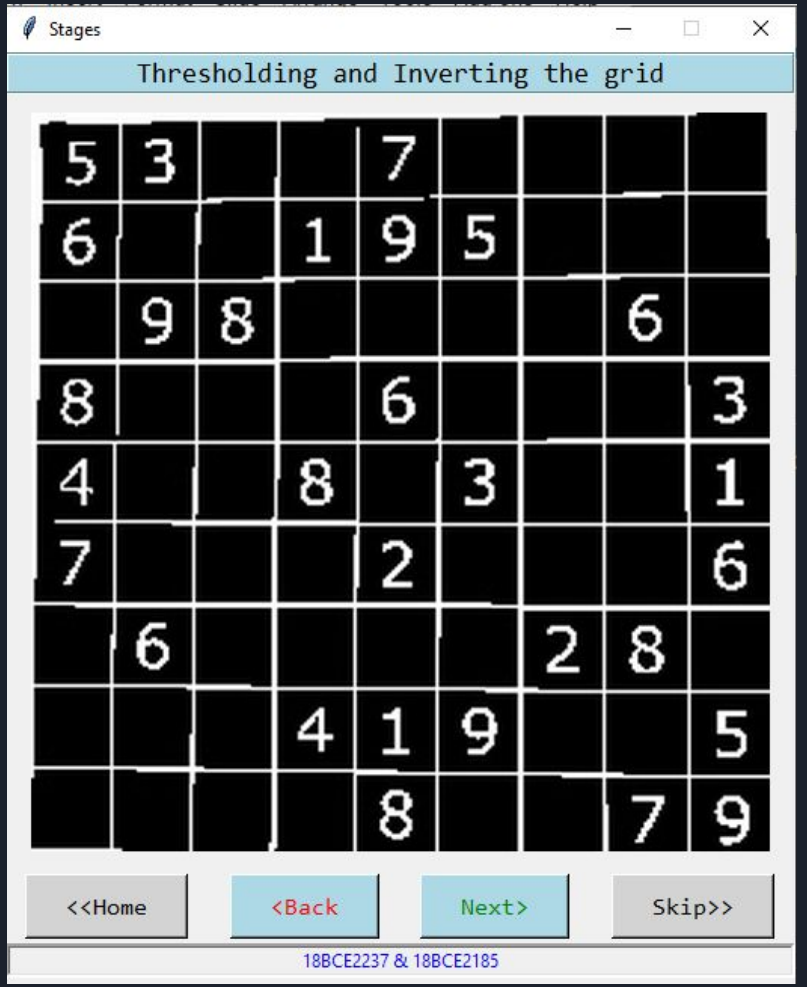

Stages
Finding the extreme lines
<<Home
<Back
Next>
Skip>>
18BCE2237 & 18BCE2185


Stages
Plotting the border lines points on the original image
5 3 7
6 1 9 5
9 8 6
8 6 3
4 8 3 1
7 2 6
6 2 8
4 1 9 5
8 7 9
<<Home
<Back
Next>
Skip>>
18BCE2237 & 18BCE2185


Stages
Correcting Perspective and cropping the grid out
5 3 7
6 1 9 5
9 8 6
8 6 3
4 8 3 1
7 2 6
6 2 8
4 1 9 5
8 7 9
<<Home
<Back
Next>
Skip>>
18BCE2237 & 18BCE2185

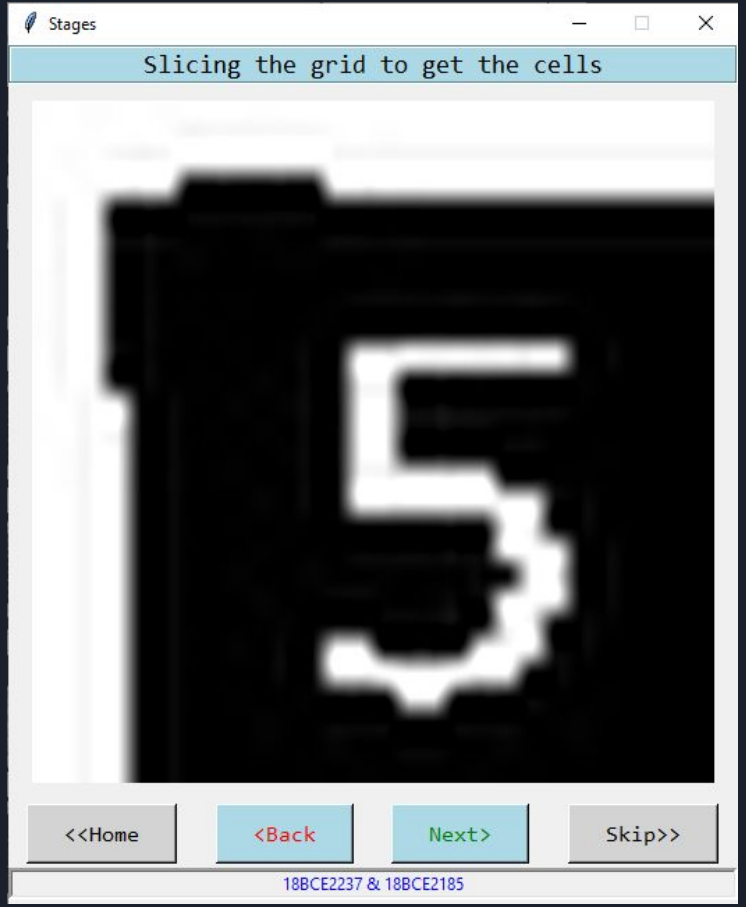
Thresholding Inv :

- The cropped image from the previous step is adaptive thresholded and inverted.



Slicing :

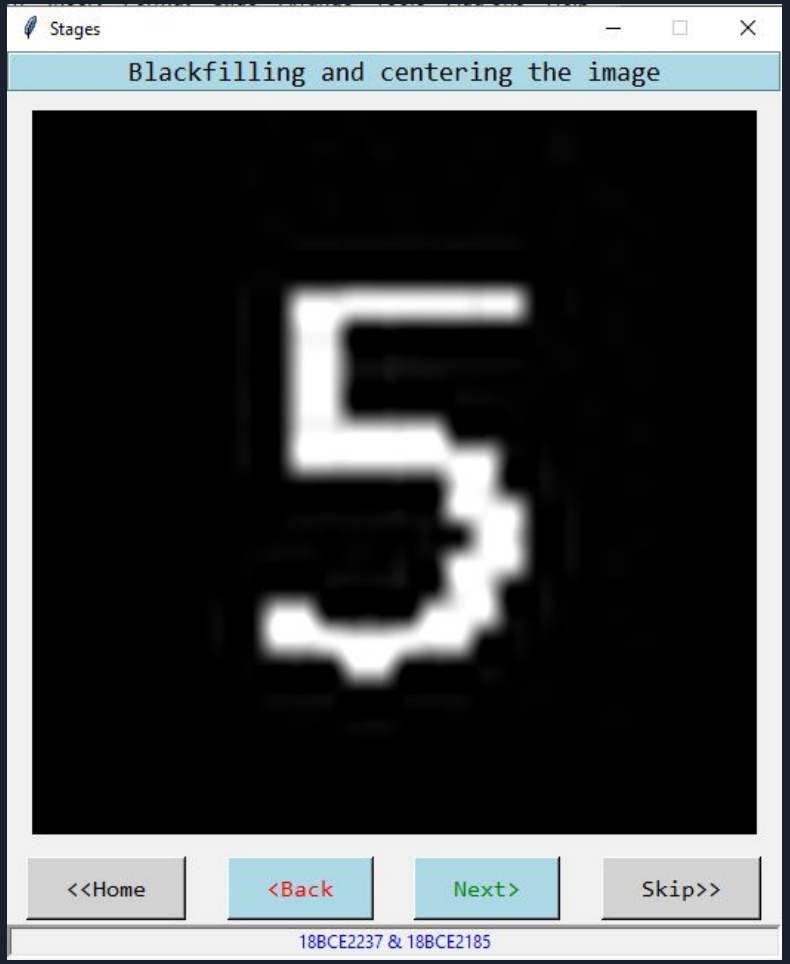
Sliced into 81 parts and each cell acquired.



Black filling :

Any white patches other than the number are removed by floodfilling with black from the outer layer points as seeds.

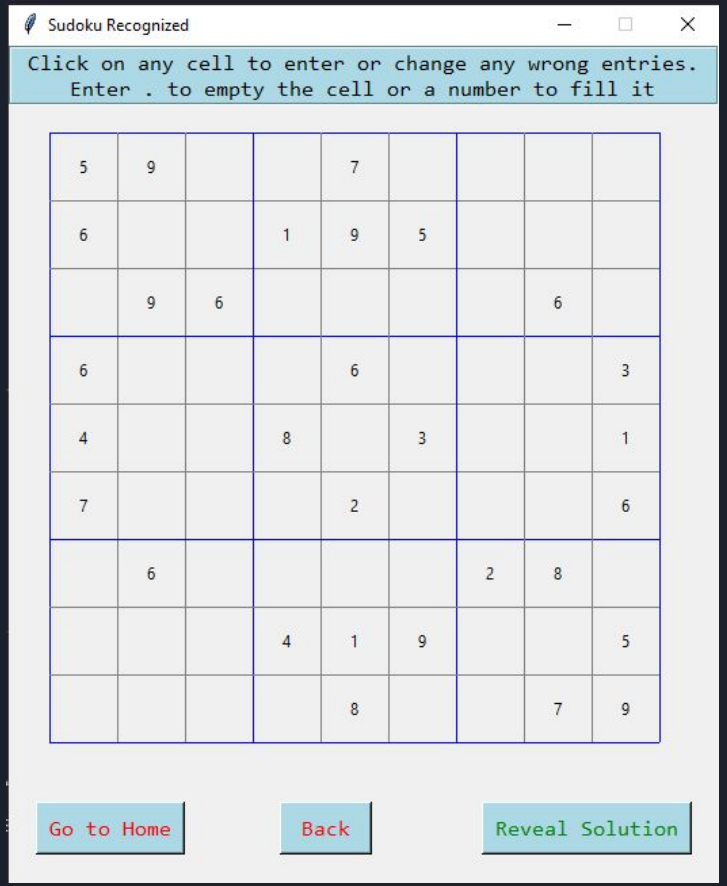
Then the approximate bounding box of the number is found and centered in the image.



Final Check :

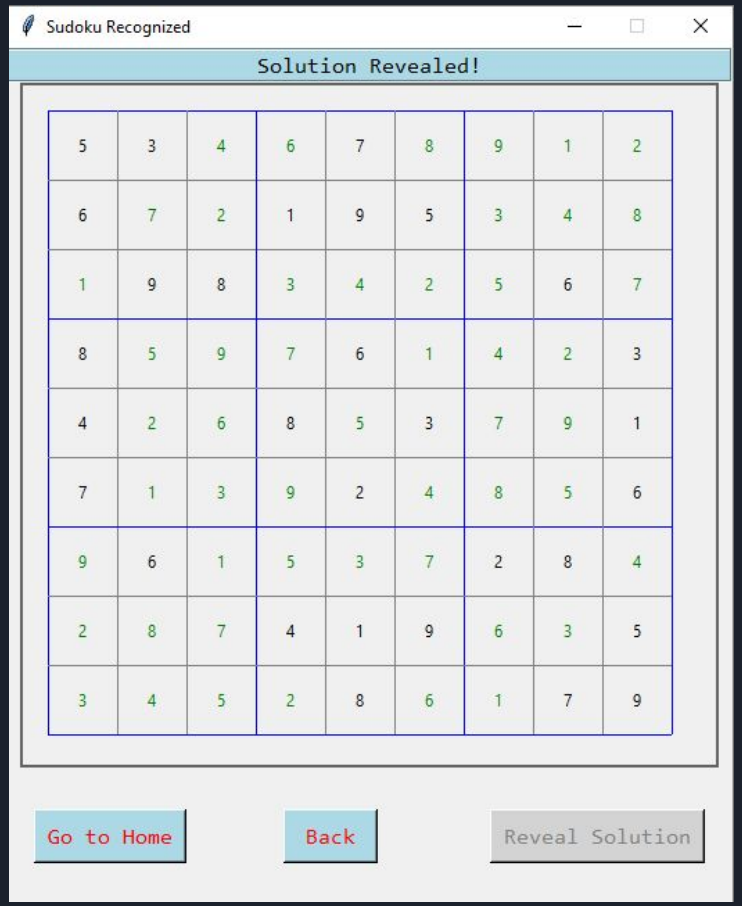
Check if all digits given are correct

If yes, proceed.



Final solution :

Once all cells are detected, they are solved as Needed and final solution is given.



References :

Image Acquisition and Techniques to Perform Image Acquisition

A Review on Image Enhancement Techniques

The research on image restoration algorithm based on improved total variation model

Morphological Image Processing

A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way

Machine Learning Basics with the K-Nearest Neighbors Algorithm