Handwritten Sudoku Solver

Aiden Sullivan , Sam Avis, Jimmy Schaffer, Dolan Clahan, Spencer Webb

Current state of Vision in Sudoku

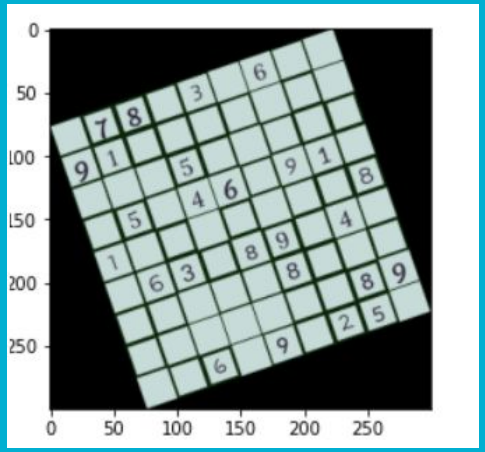
This problem was initially posed as a AI challenge on the website “AIcrowd.com” on 21st August, 2020 and submissions closed on September 11th. As of September 11th, the ‘challenge’ was won and there was an 8-way tie for first place all achieving 100% accuracy

Other image based solvers exist as well working from live camera

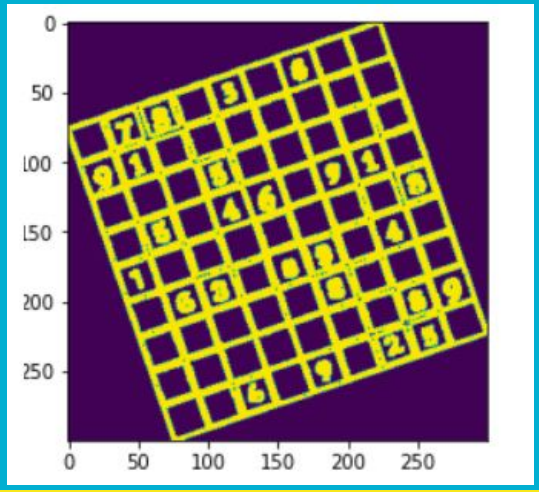
What we have achieved

Rotating and Cropping the Image

- Detection of edges via convolution of gradient kernels (used in lab 2)
  - Canny edge detection (where are the greatest changes in pixel intensity?)
  - Edge detection is straightforward given the datasets (binary coloring)

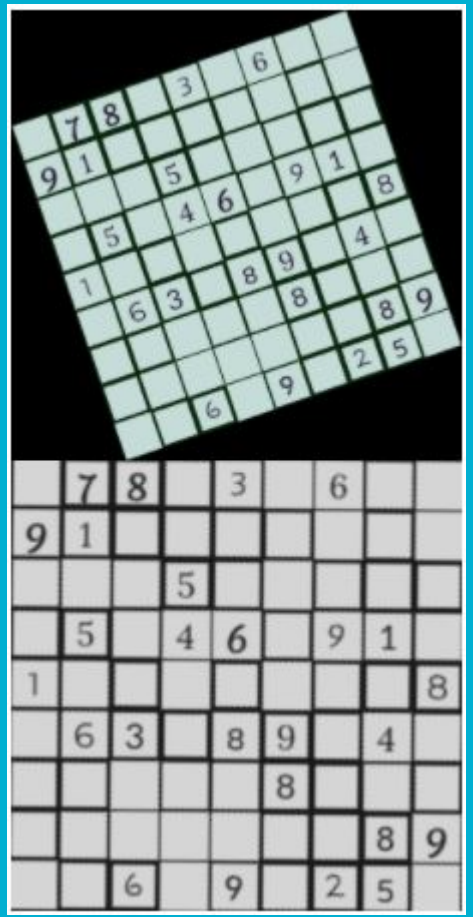


- Rotation of images using angles of lines detected via line detection algorithm
  - Hough line transformation (which lines are the most popular among the labeled edge pixels?)



Rotating and Cropping the Image

- Fit 300x300 pixel solid square to the original image by testing all degree variations until the minimum square difference between the two images is found
- Calculate the angle of rotation then calculate the corners of the new image
- Transform and crop the old image to straighten it



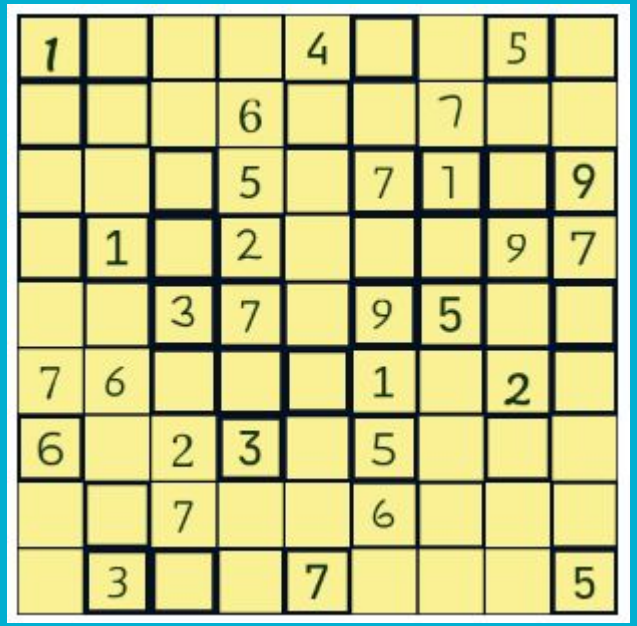
Getting Individual Cells

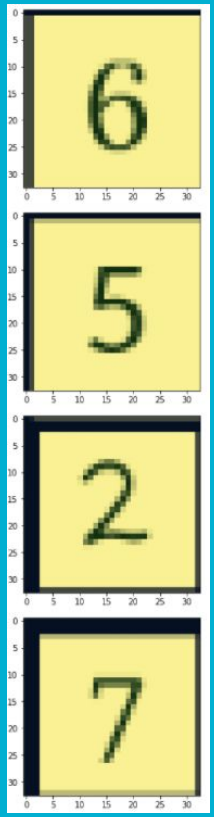
Our solution:

- Crop the puzzle into squares by pixel lengths
- Get rid of border noise/re-center
- Read number

Alternate:

- Crop puzzle using semi-random numbers or machine learning to find better fit for each square





Solving the puzzle

The simple naive brute force approach turned out to be a bit too slow

We transitioned to a package called py-sudoku for solving the puzzle

The package uses a different structure to represent a board so we had to write a function to transform the data from and then back into a string in row major form

```
sudokuPuzzle = "2__3_4___68_9__2_33_56__9__4_7_5_3___52___71___8_9_4_6__4__65_95_6__7_31___5_9__4"

findSolution(sudokuPuzzle)

'279384165681975243345621987467158392952463718138792456714836529596247831823519674'
```

Next steps

- Find faster way to import images
- Start training neural network on image data to identify numbers
- Finalize Straightening Algorithms

UPDATE 8/20

We have been able to properly implement:

- Rotate the image to its intended 300x300 pixel vertical sudoku puzzle
  - Multiple solutions found, optimized from ~30 seconds to <1 second for rotation.
- Crop the individual cells, accounting for differences in line thickness and number centering.
- Determine whether or not a given cell from a sudoku board is empty or contains some number by binarizing the tensor image and scanning for the presence of pixels in a given cell.
- Training a Neural Network on MNIST dataset for digit recognition.
- Reading the numbers from the individual cells with pre-trained model and formatting into solvable Sudoku() object.
- Training our own model using digits from non-empty cells from sudoku boards. Validating based on training data answers.
- Solved Sudoku puzzle! Accuracy? So-so. Any one mis-read digit and the sudoku solver is left with an unsolvable board, high margin for error.
- We were always aware of how integrating the many working parts of this project would be a challenge, and can confirm that that is the case.

extra

For pictures of demo, go to last 3 slides of our first presentation
https://docs.google.com/presentation/d/1HjlWsEZQKpxdhk0GJT732612pwbxkVHVC7MSoACqJQ0/edit?usp=sharing