Sudoku Solver

By:

Name	ID
***Mohamed Medhat**	77887
Mohamed Salah	77884
Hisham Osama Nabih	77864

Contents

PROBLEM DEFINITION AND IDEA

Problem is solving Sudoku games for people who struggle at some puzzles since some of them are hard or requires sudoku experts to solve, our Solution is to automatically solve the sudoku by uploading a picture of the sudoku and importing it to the application.

MODULES AND FUNCTIONS USED

opencv:

- imread
- resize
- findContours
- drawContours
- getPerspectiveTransform
- warpPerspective
- cvtColor
- addWeighted

numpy

- zeros
- float32
- asarray
- where
- array_split

Tensorflow & keras

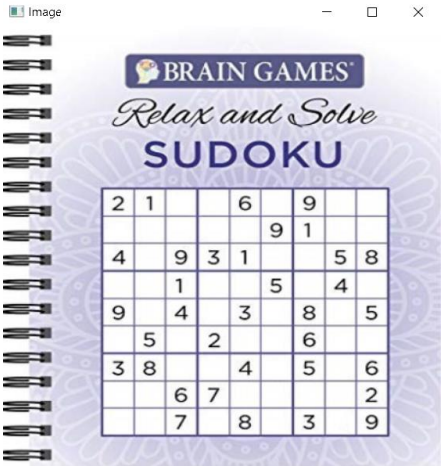
- Load_model
- Predict
- Predict_classes

Custom Functions

- InitializePredictionModel
- preProcess
- reorder
- biggestContour
- splitBoxes
- getPredictions
- drawgrid
- displayNumbers
- SudokuSolver custom Module

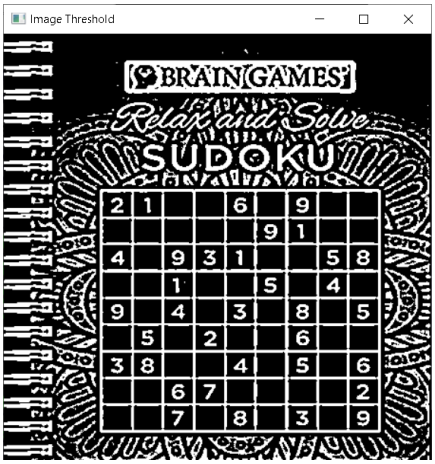
HOW?

Our solution consists of 6 Steps.
we need our sudoku to be a 9x9 grid.



1st Step is to Preprocess our image in 3 steps

1- Convert it to grey scale image
2- Add gaussian blur
3- Add adaptive threshold



Preprocessing the image helped showing the contours clearly which is essential for the next step

2nd step is to find contours of the preprocesses image

Finding contours is easy since it's a built-in function in opencv library after finding contours our image would look something like this:

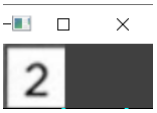
As you can see every Contour is highlighted in Green

3rd step is to find the biggest contour which will be the board itself

In this step I had to create a biggerContour function to find the biggest contour the function will check for the area of each contour and return the biggest one's dimensions + it's area. Applying this function will return the 4 points of the biggest contour (the board) but not in the correct order, so a function called reorder is then called to order the biggest contour points.

4th Step is to split the image.

We'll be splitting the image into 81 different images (our boxes since it's a 9x9 grid) and applying a Digit detection model on them to detect each digit and add them to an array. Sample of printing index[0] "2"

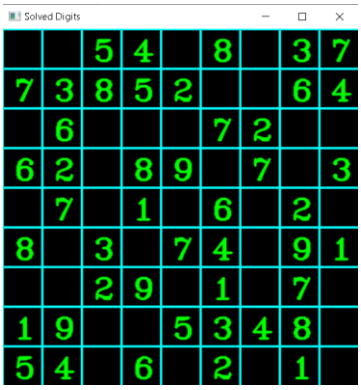


5th step is to print the classified digits and find the solutions.

Since we used a model to detect the image and store them in an array we need a way to print this array in a grid so with the help of the function Drawgrid we was able to print both grids as follows

Board

Solutions



To get solutions we used an online Sudoku algorithm which takes an array with board numbers as input and outputs the answers in an array we then used that array to print the previous grid.

6th step is Printing them all together

We needed to print the boards on each other to be readable to the user and much easier to understand, to do that we used 3 open cv functions (getPerspectiveTransform, warpPerspective, addWeighted)

getPerspectiveTransform is to get the matrix of the sudoku and save it in a variable called matrix which will then be used in the next functions

warpPerspective Applies perspective transformation to an image, the function uses the matrix that we got in the previous step to do the transformation

addWeighted this is the function which adds the solution board numbers to the original image

Results