COMP4102 Group Project Write Up

Title: Sudoku Solver

Team Members:

Wenyu Zhang	100941511
Bowen Zuo	100951180

April 2020

Carleton University

Abstract:

Sudoku is a popular puzzle game that is commonly printed on newspapers. While more and more people would prefer electronic version than plain text, the demand of converting Sudoku from an image to digital format is desirable. There are some difficulties in recognizing and reading the Sudoku. More importantly, solving the puzzle in a short time is also a desired feature, as users may want to find out the solution immediately. Therefore, this project is designed to recognize any Sudoku from an image using OpenCV and display as digital format. Then, the program can solve the Sudoku and display answer on the original image. After the algorithm modification, this program can read and solve any Sudoku puzzle precisely.

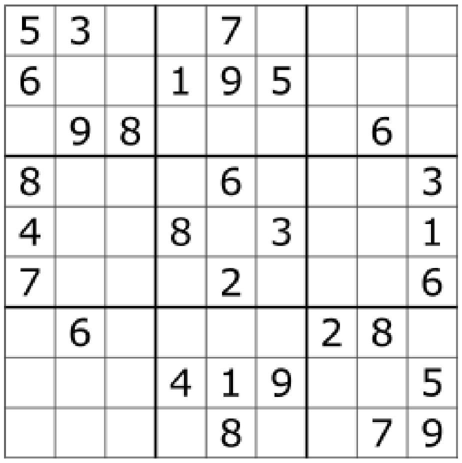
Introduction:

This application requires several steps to read and solve the puzzle. Firstly, the raw image needs to be converted to gray scale and applied some filters to remove any possible noises. Then, the processed image would be ready to recognize any potential Sudoku grid using the find contours feature in OpenCV. Once the Sudoku grid is located, the perspective of the sudoku will need to be transformed to a better shape. To read numbers from every single cell, the Sudoku grid is divided to 81 single cells. For each cell, number from the image will be transcript to text, while 0 will be assigned to any empty cell. Among all the steps above, OpenCV plays a very important role in image proceeding and querying. Several convenient features are pulled to precisely process and de-structure the image.

Move on to the solving part, the backtracking algorithm is applied with a recursive function. Specifically, for the first empty cell, some possible numbers are listed out for options. The recursive solving function starts with one of the numbers and moves on to the next empty cell. If the puzzle is solvable, the function will return the solved puzzle. If the puzzle is not solvable, then it will go back to the last point to seek for other solutions.

Background:

Modern Sudoku was firstly introduced and got popular in Japan in 1800s. This fun and logical game was soon spread to all over the world and started to present on newspapers and books. Standard Sudoku is a 9x9 grid comprised of nine 3x3 sub-grids. To complete the puzzle, each row, column and sub-grid should contain all of the digit from 1 to 9 (Fig. 1). This project is inspired by the existing mobile application called iPhone Sudoku Grab [1] which could search grid and read number in a standard Sudoku. However, this project will be built with more solving function that aim to help users to solve their Sudoku as quick as possible. The image processing part and the grid recognition part are implement using OpenCV. OpenCV is a well-developed software package for real-time computer vision [2].



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

Fig. 1. Illustration of standard Sudoku

Approach:

The challenge of this project is to recognize and solve the Sudoku correctly. Though there are some source of reference online, it would be still challengeable to recognize number correctly, especially for those pictures with low resolution. This project followed a stepwise implementation, where the raw image will experience an image processing, grid location, digit reading and puzzle solving (Fig. 2). The Sudoku recognition can be implemented by using some features in OpenCV, including cv2.findContours and cv2.getPerspectiveTransform. The solving algorithm is adapted from the backtracking algorithm. For the first empty cell, some possible numbers are listed out for options. The recursive solving function starts with one of the numbers and moves on to the next empty cell. If the puzzle is solvable, the function will return the solved puzzle. If the puzzle is not solvable, then it will go back to the last point to seek for other solutions (Fig. 3). Lastly, cv2.putText will be used to add solution back to the cropped Sudoku image.

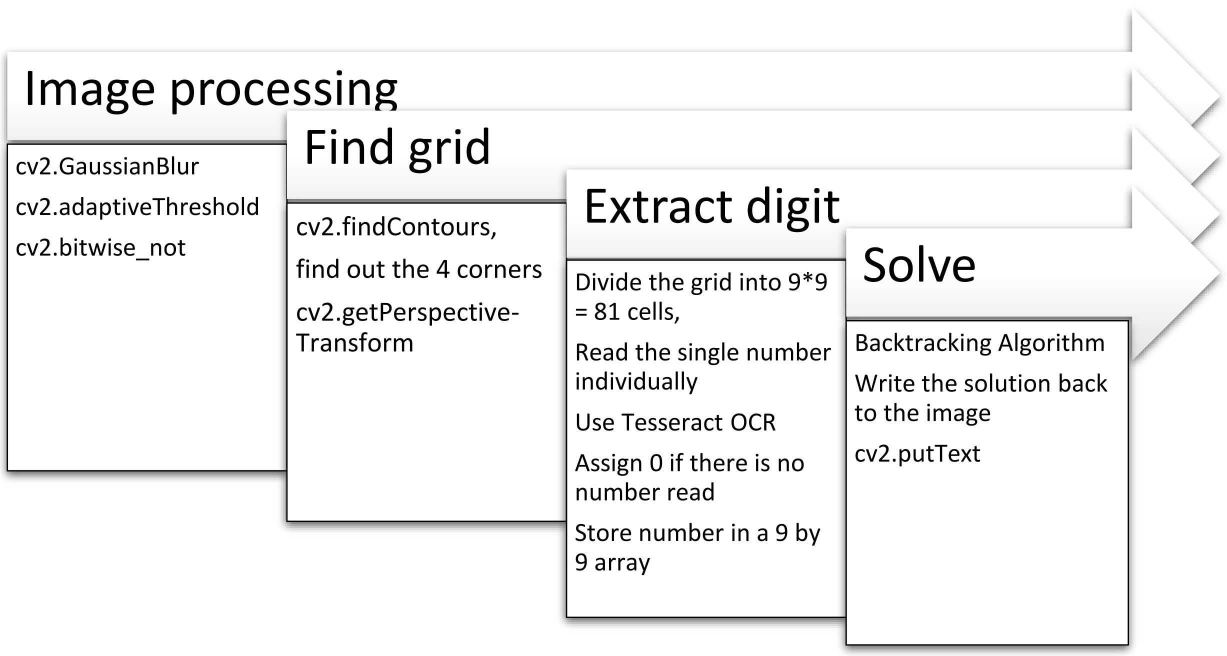
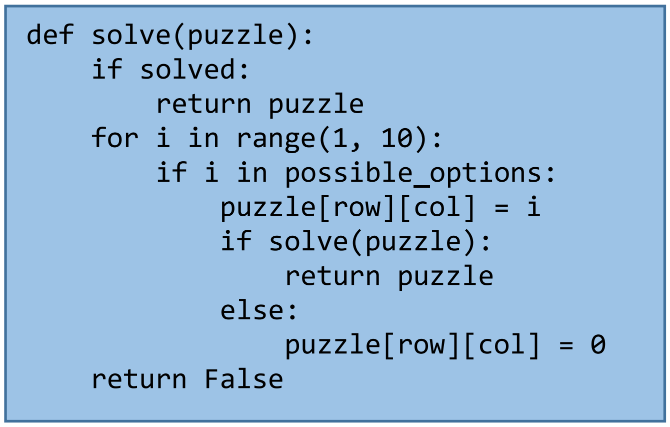


Image processing
cv2.GaussianBlur
cv2.adaptiveThreshold
cv2.bitwise_not
Find grid
cv2.findContours,
find out the 4 corners
cv2.getPerspective-Transform
Extract digit
Divide the grid into 9*9 = 81 cells,
Read the single number individually
Use Tesseract OCR
Assign 0 if there is no number read
Store number in a 9 by 9 array
Solve
Backtracking Algorithm
Write the solution back to the image
cv2.putText

Fig. 2. Flow chart of the image processing and puzzle solving steps



```
def solve(puzzle):
    if solved:
        return puzzle
    for i in range(1, 10):
        if i in possible_options:
            puzzle[row][col] = i
            if solve(puzzle):
                return puzzle
            else:
                puzzle[row][col] = 0
    return False
```

Fig. 3. Demonstration of solving algorithm

Results:

This OpenCV program can successfully recognize Sudoku grid from an image and label the 4 corners (Fig.4). After correcting the perspective (Fig.5a), it can read numbers from the image and store the matrix in an array (Fig. 5b). Finally, it can solve the puzzle and show the correct solution on the original image (Fig. 5c).

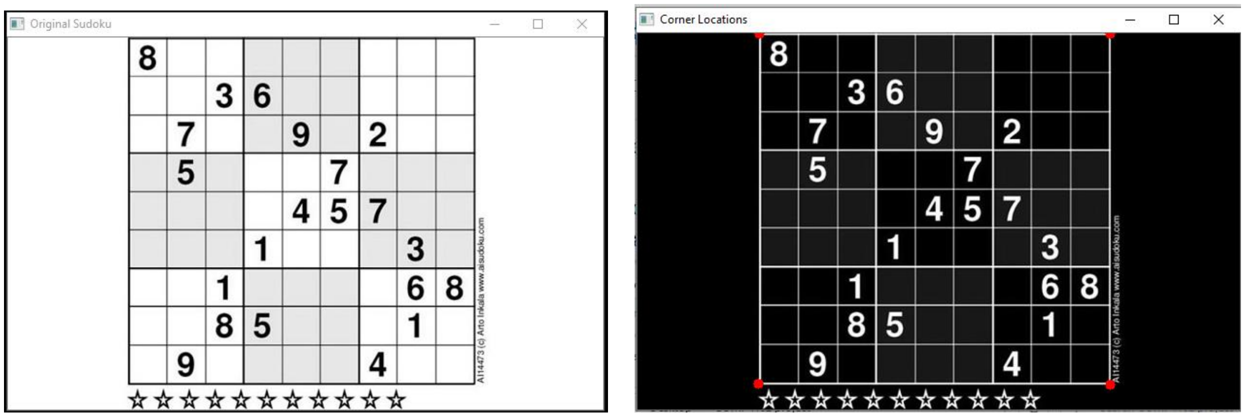
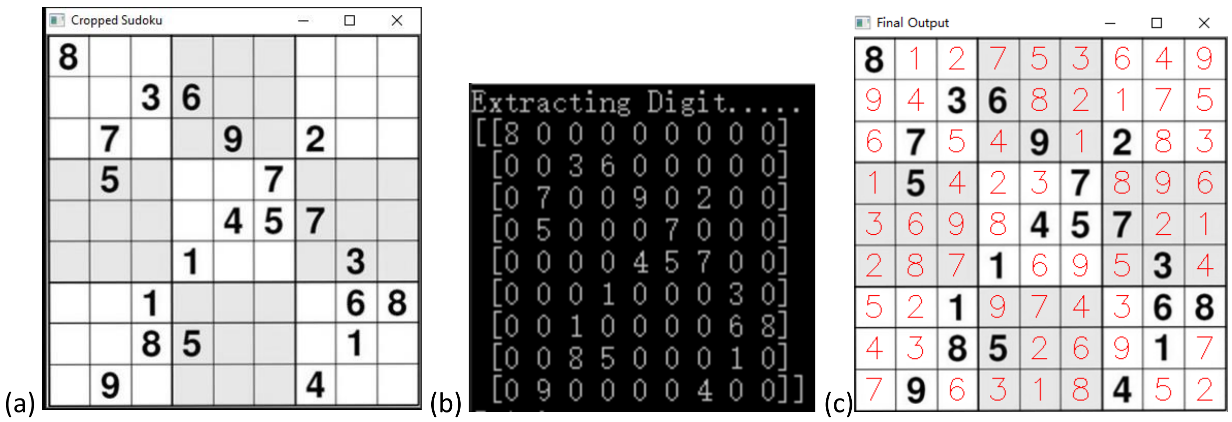


Fig. 4. The original Sudoku image (left) and the 4 corners recognition (right).



```
Extracting Digit.....
[[8 0 0 0 0 0 0 0 0]
 [0 0 3 6 0 0 0 0 0]
 [0 7 0 0 9 0 2 0 0]
 [0 5 0 0 0 7 0 0 0]
 [0 0 0 0 4 5 7 0 0]
 [0 0 0 1 0 0 0 3 0]
 [0 0 1 0 0 0 0 6 8]
 [0 0 8 5 0 0 0 1 0]
 [0 9 0 0 0 0 4 0 0]]
```

Fig. 5. The cropped Sudoku image (a), Sudoku in array format (b) and the solved Sudoku (c).

List of Work:

	Team members	
Week	**Wenyu Zhang**	**Bowen Zuo**
Feb 3, 2020	Determine UI language, find the library in OpenCV for gridding.	Research on available source and find tools for processing the image.
Feb 10, 2020	Write draft coding for grid recognition	Write draft coding for grid recognition
Feb 17, 2020	Implement gird recognition	Implement number recognition
Feb 24, 2020	Prepare for midterm	Prepare for midterm
Mar 2, 2020	Design user interface	Implement Sudoku solving
Mar 9, 2020	Add features	Optimize the solving algorithm
Mar 16, 2020	Fix bugs and polish syntax	Fix bugs and polish syntax
Mar 23, 2020	Prepare for demo + testing	Prepare for demo + testing
Mar 30, 2020	Demo + Report editing	Demo + Report editing
Apr 6, 2020	Fix minor errors and record	Finish writing report

GitHub Page:

https://github.com/ivoryzh/project

References:

[1] http://sudokugrab.blogspot.com/2009/07/how-does-it-all-work.html

[2] https://opencv.org/