Yarmouk Private University
Faculty of Informatics and
Communications Engineering

جامعة اليرموك الخاصة كليّة
هندسة المعلوماتيّة و الاتصالات
قسم هندسة البرمجيات

Sudoku Solver

Applied Project Report

Prepared By:

Obada Abdul Hafeez Saleh.

Hamza Saeed Farhat.

Supervisor:

Eng. *Joud Khattab*

Semester 2017/2016 3

`

Yarmouk Private University
Faculty of Informatics and
Communications Engineering

جامعة اليرموك الخاصة كليّة
هندسة المعلوماتيّة و الاتصالات
قسم هندسة البرمجيات

Sudoku Solver

Applied Project Report

Prepared By:

Obada Abdul Hafeez Saleh.

Hamza Saeed Farhat.

Supervisor:

Eng. *Joud Khattab*

Semester 2017/2016 3

Page of committee:

Name of students:

- *Obada Abdul Hafeez Saleh*

- *Hamza Saeed Farhat*

Main advisor name: Eng. *Joud Khattab* **signature:**

Abstract

This project helps to solve *Sudoku* game with a simple way that requests a good quality image with no surroundings to deal with it in a high accuracy.

In other words the accuracy of the project depends on the quality of the inserted image.

The project will process the image and read the number of the filled cells and the empty cells. Then, the data will pass to the intelligence algorithm to find solution. Finally, the solution will print on the empty *Sudoku* image.

The Final output is a solved *Sudoku* image.

Table of Content:

Table of Figures:

[Chapter 1] Introduction

1-1 - Project Overview:

The idea of the project comes from the increasing difficulty for solve *sudoku* mystery by some people, learn some basics in *AI* (artificial intelligence), and image processing algorithms.

1-2 - Functional requirements:

1- Read Image from user
2- Process the *sudoku* image to be ready for OCR Engine (optical character recognition)
3- Read numbers from the *sudoku* image and prepare it to the solution algorithm
4- Solve the game using intelligence algorithm.
5- Create an empty *sudoku* image
6- Print the solution to the new empty *sudoku* image

1-3 - Non-functional requirements:

1- Graphical user interface.
2- Make the project as an *API*

1-4 - Time Line:

Executive Business	Gregorian month				
	8	8	9	9	9
Learn Python	X	X	X		
Learn how to work with tesseract library		X	X		
Analytical Study		X			
Design Study			X		
Implementation the algorithm			X	X	
Test			X	X	X
Document the project and the report			X	X	X

1-5 - Sudoku Game:

Sudoku is one of the most popular puzzle games of all time. The goal of *Sudoku* is to fill a 9×9 grid with numbers so that each row, column and 3×3 section contain all of the digits between 1 and 9. As a logic puzzle, *Sudoku* is also an excellent brain game.

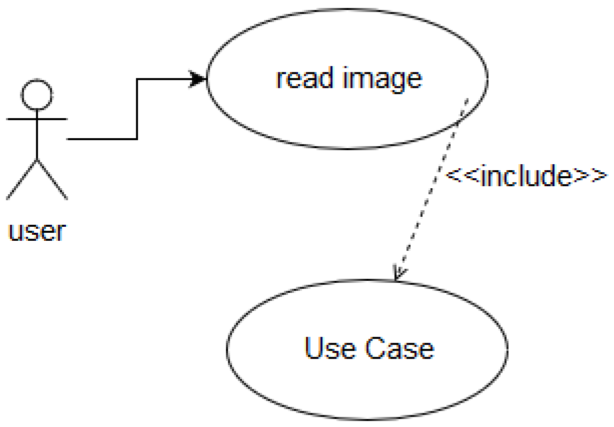
5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Fig (1-1): Example of *Sudoku* game

[Chapter 2]

Software Requirement

2-1 - UseCase Diagram:

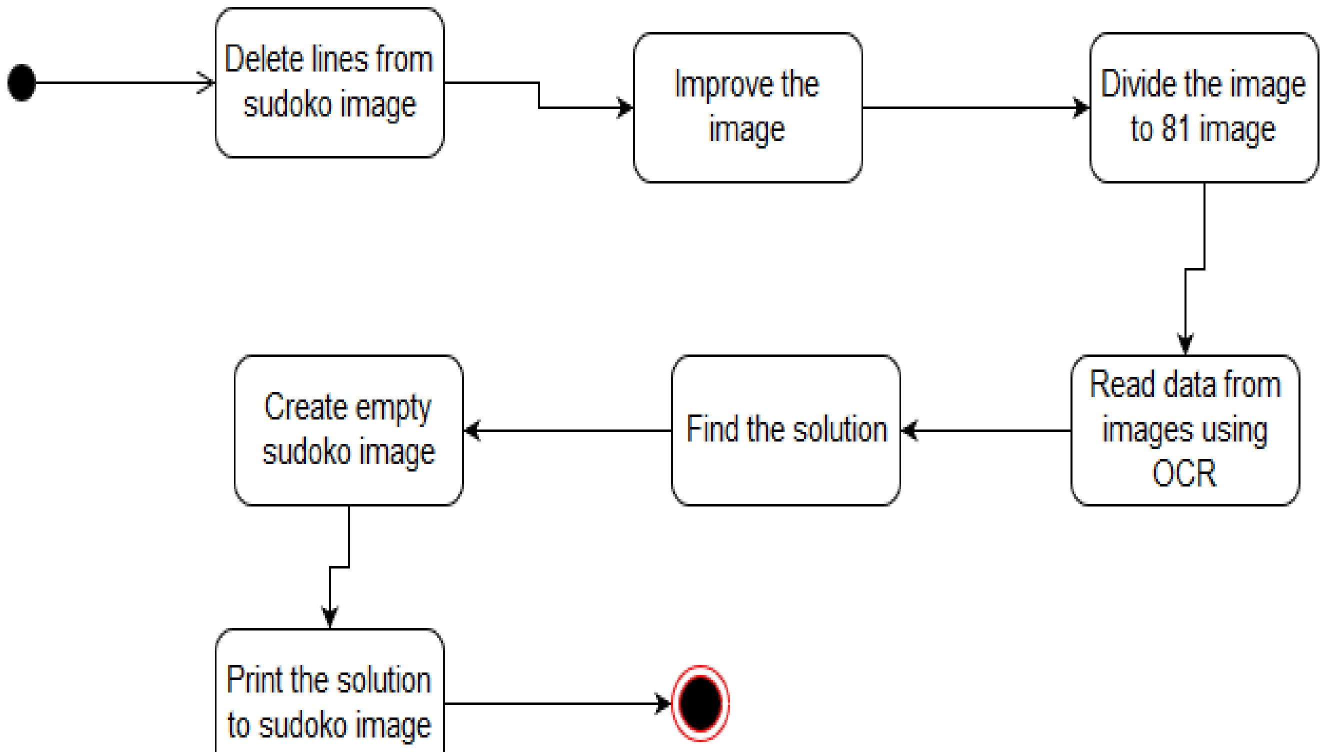


2-2 - UseCase Specification:

Name	**Solve Game**
Description	This *UseCase* describe how the program solve the game.
Actors	User.
Pre-Condition	The *sudoku* image must be inserted
Critical scenario	Read wrong numbers because of the accuracy
Post Condition	The game was solved in a new image
Flow of Event	1. System: processing the image 2. System: read the numbers of image 3. System: print the solution in a new image

Name	**Read Image**
Description	This *UseCase* describe how the program read image.
Actors	User.
Pre-Condition	X
Critical scenario	Insert wrong image
Post Condition	System have the *sudoku* image to solve

2-3 - Activity Diagram:


Delete lines from sudoko image
Improve the image
Divide the image to 81 image
Read data from images using OCR
Find the solution
Create empty sudoko image
Print the solution to sudoko image

[Chapter 3]
Implementation

3-1 - Environments

3-1-1 - Python:

Python is an interpreted high-level programming language for general-purpose programming.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Python interpreters are available for many operating systems. *CPython*, the reference implementation of *Python*, is open source software and has a community-based development model, as do nearly all of *Python's* other implementations. *Python* and *CPython* are managed by the non-profit Python Software Foundation. [5]

3-1-2 - OpenCV:

OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by *Willow Garage* then *Itseez* (which was later acquired by Intel). The library is cross-platform and free for use under the open-source *BSD* license.*OpenCV* is written in *C*++ and its primary interface is in *C*++, but it still retains a less comprehensive though extensive older C interface. There are binding sin *Python*, *Java* and *MATLAB/OCTAVE*. The *API* for these interfaces can be found in the online documentation. Wrappers in other languages such as *C#*, *Perl*, *Ch*, *Haskell* and *Ruby* have been developed to encourage adoption by a wider audience. All of the new developments and algorithms in *OpenCV* are now developed in the *C*++ interface. [6]

3-1-3 - OCR Library:

Optical character recognition (also optical character reader, OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast).

Widely used as a form of information entry from printed paper data records – whether passport documents, invoices, bank statements, computerized receipts, business cards, mail, printouts of static-data, or any suitable documentation – it is a common method of digitizing printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. *OCR* is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common, and with support for a variety of digital image file format inputs. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

Accuracy rates can be measured in several ways, and how they are measured can greatly affect the reported accuracy rate. For example, if word context (basically a lexicon of words) is not used to correct software finding non-existent words, a character error rate of 1% (99% accuracy) may result in an error rate of 5% (95% accuracy) or worse if the measurement is based on whether each whole word was recognized with no incorrect letters. [7]

3-1-4 - Python Imaging Library:

Python Imaging Library (abbreviated as **PIL**) (in newer versions known as Pillow) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different image file formats. It is available for Windows, Mac OS X and Linux. The latest version of PIL is 1.1.7, was released in September 2009 and supports Python 1.5.2–2.7, with Python 3 support to be released "later".

Development appears to be discontinued with the last commit to the *PIL* repository coming in 2011. Consequently, a successor project called ***Pillow*** has forked the *PIL* repository and added *Python* 3.x support. This fork has been adopted as a replacement for the original *PIL* in *Linux* distributions including *Debian* and *Ubuntu* (since 13.04) [8]

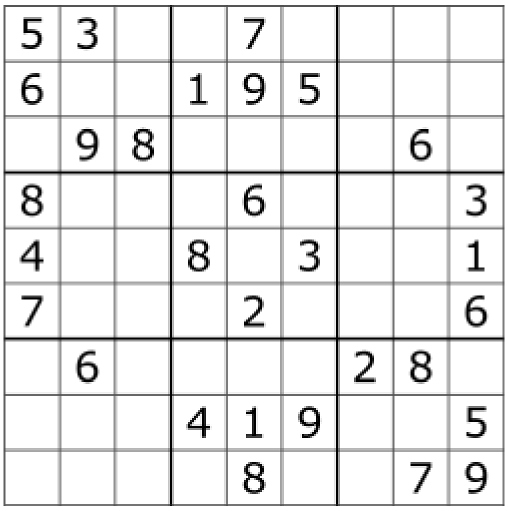
3-1-5 - NumPy Library:

NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. The ancestor of *NumPy*, Numeric, was originally created by *Jim Hugunin* with contributions from several other developers. In 2005, *Travis Oliphant* created *NumPy* by incorporating features of the competing *Numarray* into Numeric, with extensive modifications. *NumPy* is open-source software and has many contributors. [9]

3-2 - The proposed algorithm

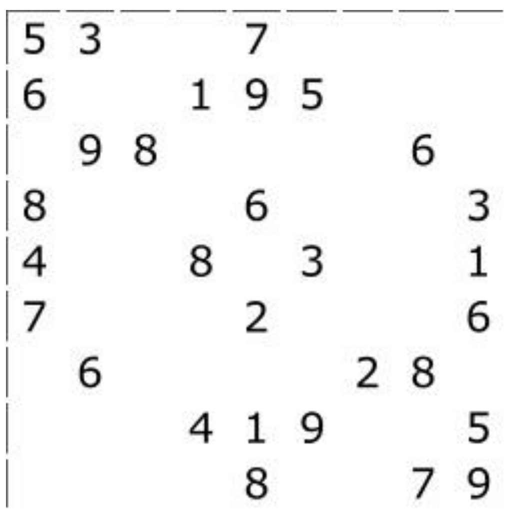
3-2-1 - Pre-processing:

First, Get *sudoku* image and delete the lines from it.



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

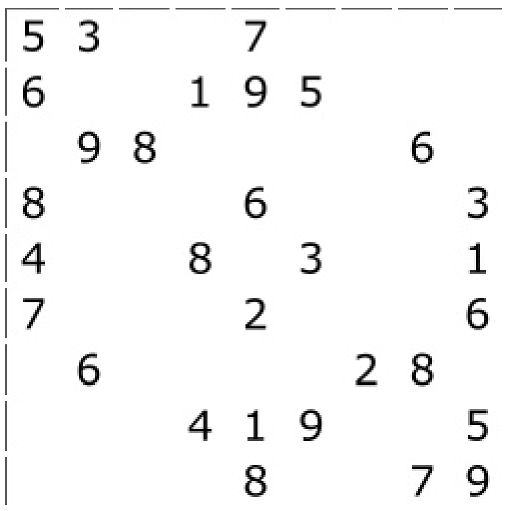
Fig (3-1): initial *Sudoku* image



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

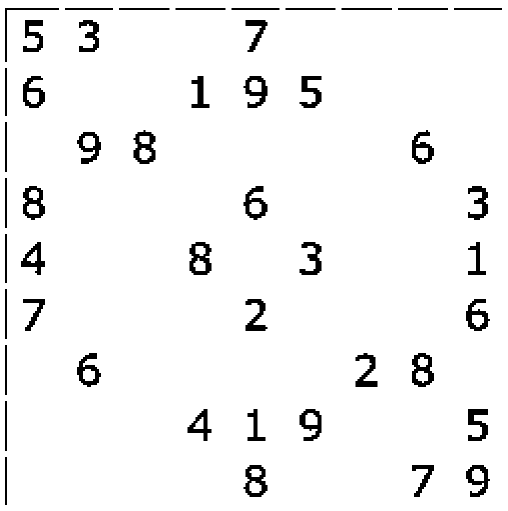
Fig (3-2): *Sudoku* image without lines

Next, remove the noise from image and convert it to binary (black and white).



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 (3-3): *Sudoku* image without lines



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 (3-4): Improved image

Next, divide the image to 81 image in which every number or blank cell will be in individual image

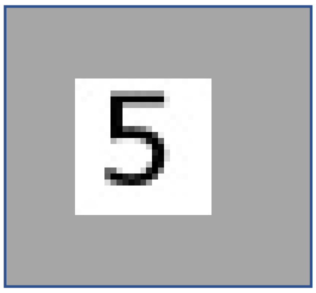


Fig (3-5): single cell image

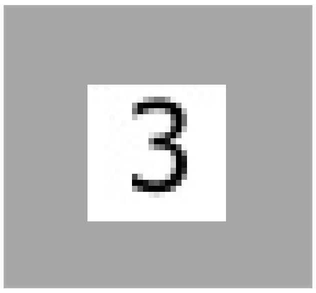


Fig (3-6): single cell image

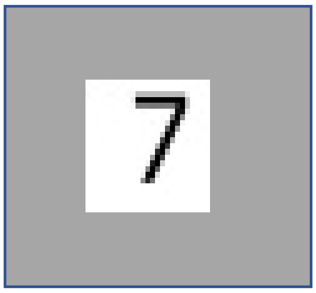


Fig (3-7): single cell image

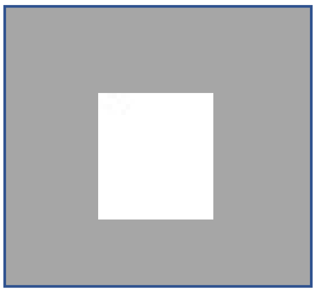


Fig (3-8): single cell image

Finally, read the number from each image to prepare it for the solution algorithm

3-2-2 - Solution Algorithm:

First, section our patch to 81 Square.The columns take the name from Characters, and the rows take name from number.

Like this:

```
A1 A2 A3| A4 A5 A6| A7 A8 A9
B1 B2 B3| B4 B5 B6| B7 B8 B9
C1 C2 C3| C4 C5 C6| C7 C8 C9
--------+---------+---------
D1 D2 D3| D4 D5 D6| D7 D8 D9
E1 E2 E3| E4 E5 E6| E7 E8 E9
F1 F2 F3| F4 F5 F6| F7 F8 F9
--------+---------+---------
G1 G2 G3| G4 G5 G6| G7 G8 G9
H1 H2 H3| H4 H5 H6| H7 H8 H9
I1 I2 I3| I4 I5 I6| I7 I8 I9
```

Then, Customize for all square the squares which peers for help to solve the game.

For example, the peers for C7:

```
--------| --------| A7 A8 A9
--------| --------| B7 B8 B9
C1 C2 C3| C4 C5 C6| C7 C8 C9
--------+---------+---------
--------| --------| D7 -----
--------| --------| E7 -----
--------| --------| F7 -----
--------+---------+---------
--------| --------| G7 -----
--------| --------| H7 -----
--------| --------| I7 -----
```

Finally, lean for solve game on tow rules:

1- If a square has only one possible value, then eliminate that value from the square's peers.
2- If a unit has only one possible place for a value, then put the value there.

3-2-3 - Display the Solution (Output):

First, build an empty *sudoku* image

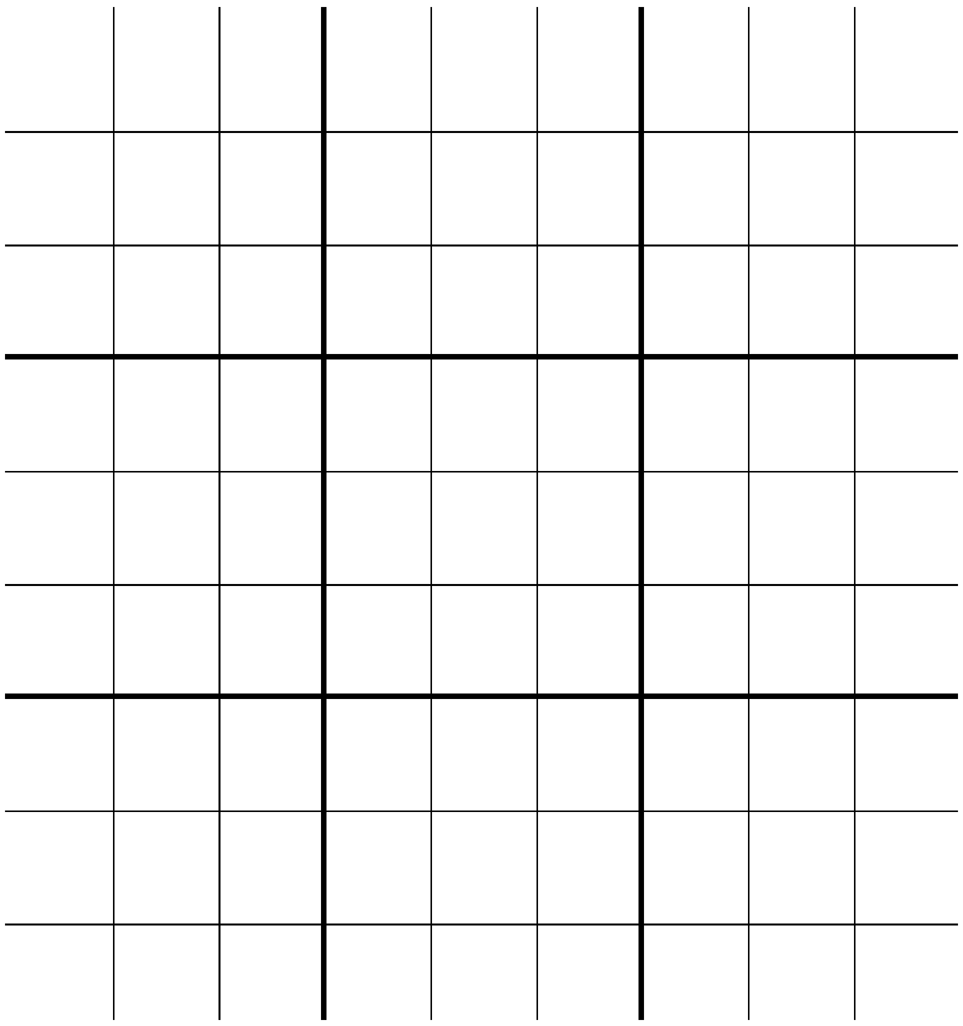
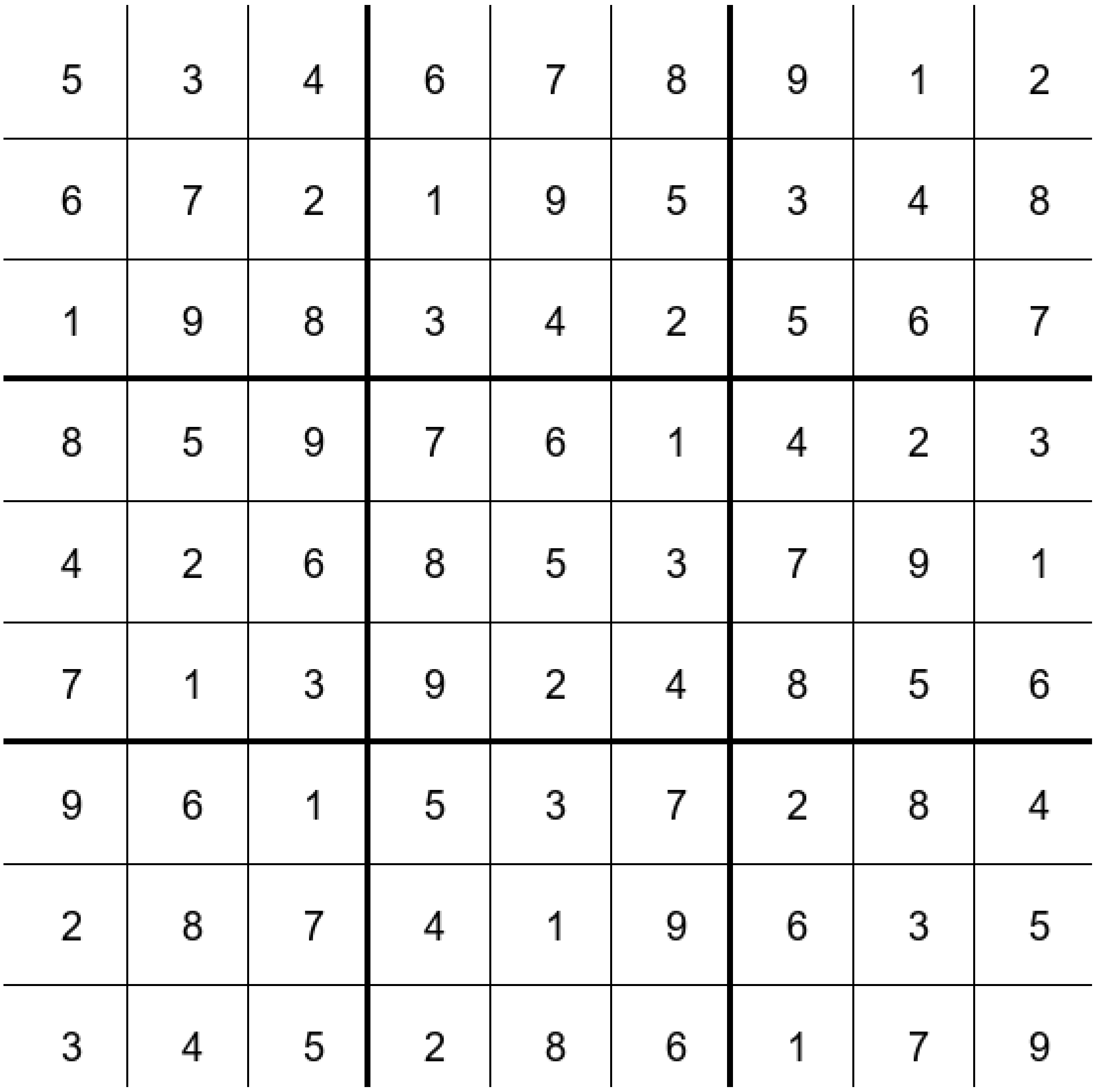


Fig (3-9): Empty *Sudoku* image

Then, print the solution to the empty image

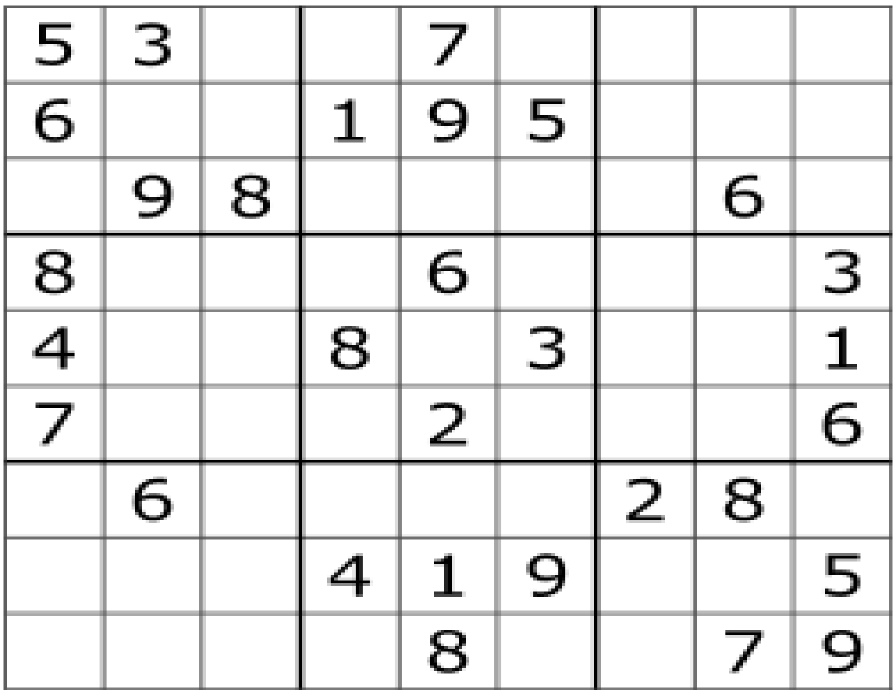


5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

Fig (3-9): solved *Sudoku* image

3-3 - Summary:

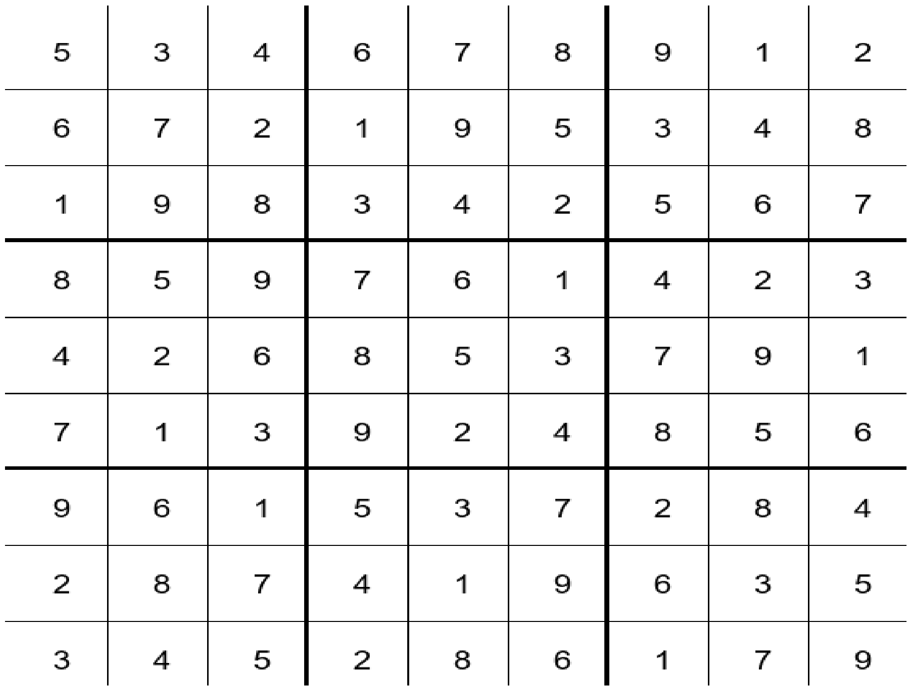
User insert *sudoku* image:



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

Fig (3-10): Inserted image

System give solved *Sudoku* image:



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

Fig (3-11): The resulting image

[Chapter 4]
Conclusion & feature plan

4-1 - Conclusion

We find that this intelligence algorithm is functional by to aspects high quality and processing speed.

Other thing, the OCR (optical character recognition) algorithm is very powerful in character reading but it does not serve the project in a high quality.

Finally, the image processing is the best step in the project because of the resulting image.

4-2 - Feature Plan

- Increase the speed of solution algorithm
- Increase the accuracy of data fetching from image
- Improve the interface design
- Develop the project to be parallel instead of sequential

References:

1. ***openCV* tutorial**. Available from: https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_tutorials.html
2. ***Python* tutorial**. Available from:

https://www.w3schools.com/python/default.asp

3. https://www.geeksforgeeks.org
4. https://stackoverflow.com
5. ***Python* language**. Available from: https://en.wikipedia.org/wiki/Python_(programming_language)
6. ***OpenCV* library**. Available from: https://en.wikipedia.org/wiki/OpenCV
7. **Optical Character Recognition (OCR).** Available from: https://en.wikipedia.org/wiki/Optical_character_recognition
8. **Python Imaging Library (PIL).** Available from: https://en.wikipedia.org/wiki/Python_Imaging_Library
9. **NumPy Libarary.** Available from: https://en.wikipedia.org/wiki/NumPy

ملخص المشروع

يساعد هذا البرنامج بحل لعبة سودوكو بطريقة بسيطة , حيث يتطلب البرنامج صورة بدقة جيدة و يجب ان تحتوي هذه الصورة على واجهة لعبة السودوكو فقط

بإيجاز تعتمد دقة البرنامج على جودة الصورة المدخلة

سيعالج البرنامج الصورة و يقرأ منها الارقام ,و يحدد الخلايات الفارغة ,ثم ستمرر هذه البيانات الى خوارزمية ذكية لإجاد الحل المناسب للصورة المدخلة

في الخطوة الأخيرة سيطبع البرنامج الحل على صورة فارغة لواجهة لعبة السودوكو