**Software Support and Design Methodology**

**Software Support**

This part shall enumerate the softwares and libraries used and give a brief description about them and list the websites where one can find them. Later on the sequential installation procedure will be mentioned. We’re using Windows 7, 32-bit system and would provide links for the same. If you’re using any other operating system you’ll have search for the respective softwares/ libraries yourself.

**Softwares Used**

1. Python - We’ve used python as our choice of programming language since it’s simple to learn, can be used on all desktop computers and is the most used programming language as of 2015. We’re using 2.7.11 version and it can be downloaded from the link provided below.

Download link:

<https://www.python.org/ftp/python/2.7.11/python-2.7.11.msi>

Documentation link:

<https://docs.python.org/2/>

2. Sublime Text - This is simply a text editor, and is not in any way necessary for the code to run. But it is highly recommended by us because it being light weight and it provides better code readability and a ton of customization options. We’ll be using Sublime Text 2, specifically version (2.0.2a), and it can be downloaded from the link given below.

Download link:

<https://download.sublimetext.com/Sublime%20Text%202.0.2a%20Setup.exe>

Documentation link:

<https://www.sublimetext.com/docs/2/>

**Libraries Used**

1. OpenCV - OpenCV is released under a BSD license and hence it’s free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. We’ll be using OpenCV for image processing. We’ll be using the OpenCV version 2.4.12 and the link is provided below.

Download link:

<http://iweb.dl.sourceforge.net/project/opencvlibrary/opencv-win/2.4.12/opencv-2.4.12.exe>

Documentation link:

<http://docs.opencv.org/2.4.12/>

2. NumPy - NumPy is the fundamental package for scientific computing with Python. Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. We’ll be using NumPy for complex numerical computations. We’ll be using the NumPy version 1.8.0 and the link is provided below.

Download link:

<http://liquidtelecom.dl.sourceforge.net/project/numpy/NumPy/1.8.0/numpy-1.8.0-win32-superpack-python2.7.exe>

Documentation link:

<http://docs.scipy.org/doc/numpy-1.8.0/reference/>

3. SciPy - SciPy (pronounced “Sigh Pie”) is a Python-based ecosystem of open-source software for mathematics, science, and engineering. We’ll be using the SciPy version 0.16.1 and the link is provided below.

Download link:

<http://liquidtelecom.dl.sourceforge.net/project/scipy/scipy/0.16.1/scipy-0.16.1-win32-superpack-python2.7.exe>

Documentation link:

<http://docs.scipy.org/doc/scipy/reference/release.0.16.0.html>

4. dlxsudoku - Sudoku Solver written in pure Python with no dependencies. It solves Sudokus of sizes N x N by pure induction as far as is possible, and then uses an optional Dancing Links brute force solver, when the basic induction is not enough. We’ll be using the dlxsudoku version 0.10.1 and the link is provided below.

Download link:

<https://pypi.python.org/packages/source/d/dlxsudoku/dlxsudoku-0.10.1.tar.gz#md5=72cf8421327509e527aa4abe0d17b0c2>

Documentation link:

<https://github.com/hbldh/dlxsudoku>

**Installation Procedure**

The installation procedure is mentioned below and it is highly recommended that you follow it step by step as given and don’t mess up the order.

1. Download Python from the link provided and install it.
2. Check if the Python has been added to the path variable in Windows. You can do this by right-clicking on ‘My Computer’, selecting ‘properties’, then selecting ‘Advanced System Settings’, then clicking on the ‘Advanced’ tab, then selecting ‘Environment Variables’. Now in the ‘System variables’ box, double click on ‘Path’ and check if the following is present in the ‘Variable value’ field - “C:/Python27;C:/Python27/Scripts”. If it isn’t, add it.
3. Download and install NumPy from the link provided.
4. Download and install SciPy from the link provided.
5. Download and install (actually extract) OpenCV to a location of your choice.
6. Then inside the folder navigate to “opencv\build\python\2.7\x86” and copy the file named ‘cv2.pyd’. Now go to “C:\Python27\Lib\site-packages” and paste the file here.
7. Now we need to install pip so that we can download dlxsudoku package. For installing pip visit the following website: <https://bootstrap.pypa.io/get-pip.py> and copy and paste the entire code to a notepad file on your desktop. Now press ‘Ctrl + Shift’ on your keyboard and simultaneously ‘right-click’ on the desktop, and select “Open command window here”. Now type the following command to install pip: “python get-pip.py”.
8. Now again open command line. This can be done by pressing ‘Start’, typing “cmd”, and pressing ‘Enter’. Now type the command: “pip install dlxsudoku”, to install the package. You need not explicitly download this from the link provided above.

This completes the installation part of the project. Alternately if you opt to use Sublime Text 2, then the installation and setup of the same is given below.

1. Download and install Sublime Text 2 from the link provided above.
2. Open Sublime Text.
3. Go to ‘View’, and then to ‘Show Console’.
4. Visit this site: “<https://packagecontrol.io/installation#st2>” and copy the code snippet under the heading “SUBLIME TEXT 2”.
5. Paste it in the console of Sublime Text and press ‘Enter’. Package Control will be installed.
6. In Sublime press ‘Ctrl + Shift + P’
7. Type in “install”
8. Click on “Package Control: Install Package”.
9. Then select SublimeREPL. It will install it automatically.

This completes the installation part of Sublime Text.

Now for executing code using the default Python text editor namely Python IDLE press F5. And if you’re using Sublime Text, simply pressing “Ctrl + B” will do. However in case you need to give some input to the code then you’ll need the console, and will have to follow the following method. Go to “Tools > SublimeREPL > Python > Python” ,a new tab will open in Sublime Text, this will be your console. Now for executing a script go to the script tab and press “Ctrl + ,” and then “f” on the keyboard.

**Design Methodology**

In this section working of all the parts of the code will be explained. For explanatory purposes let’s assume we apply our script on the following image.

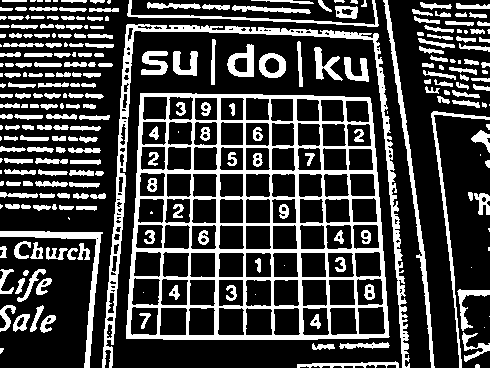
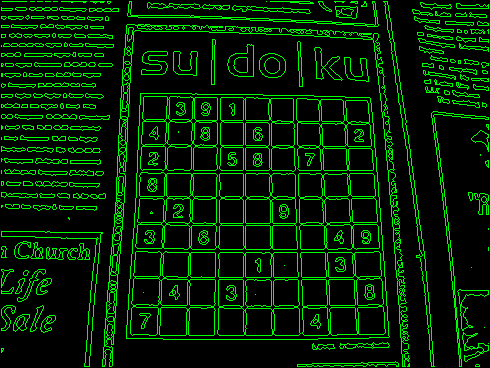


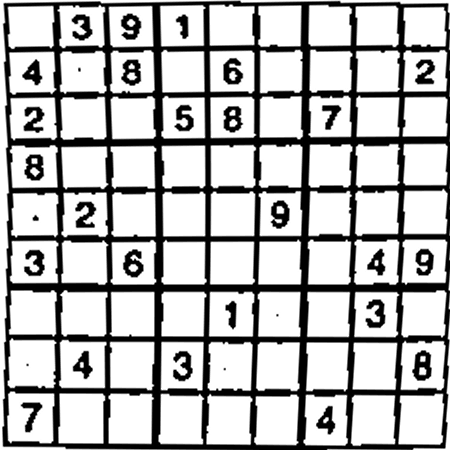
**Image Processing**

The first script written by us was “imgProcessing.py”. The procedural overview of the script is as follows.

1. Loads the image in Python in the form of integer matrix. Every pixel is represented by 3 integers, which denote the RGB content of the pixel respectively.
2. Now the cv2.cvtColor(Sudoku\_Image,cv2.COLOR\_BGR2GRAY) converts the colour image into a black and white image. Now the value of every pixel is represented by only 1 integer, denoting the intensity of that pixel, i.e. how black the pixel is. 0 being black and 255 being white.  
   
3. Then the function cv2.GaussianBlur removes any noise from the image.



1. Next, onto the image, we apply the function cv2.adaptiveThreshold. This converts the gray scale image into a binary image, meaning each pixel is either black or white. So the value of each pixel is either 0 (meaning black) or 1 (meaning white).  
   
2. Next the script detects contours present in the image with the cv2.findContours function. The green colour in the image below indicates the boundaries of all the contours present in the image.  
   
3. Next our script goes through every contour and locates the contour enclosing the maximum area. This is basically the Sudoku grid more often than not.
4. Next we approximate the coordinate values of this contour using the cv2.approxPolyDP function. This function approximates the contour coordinates as 4 values only, representing the corners of the skewed square.
5. Next we create a new image of 450x450 by resizing this skewed square to our final region of interest and invert the colour scheme to get the following image



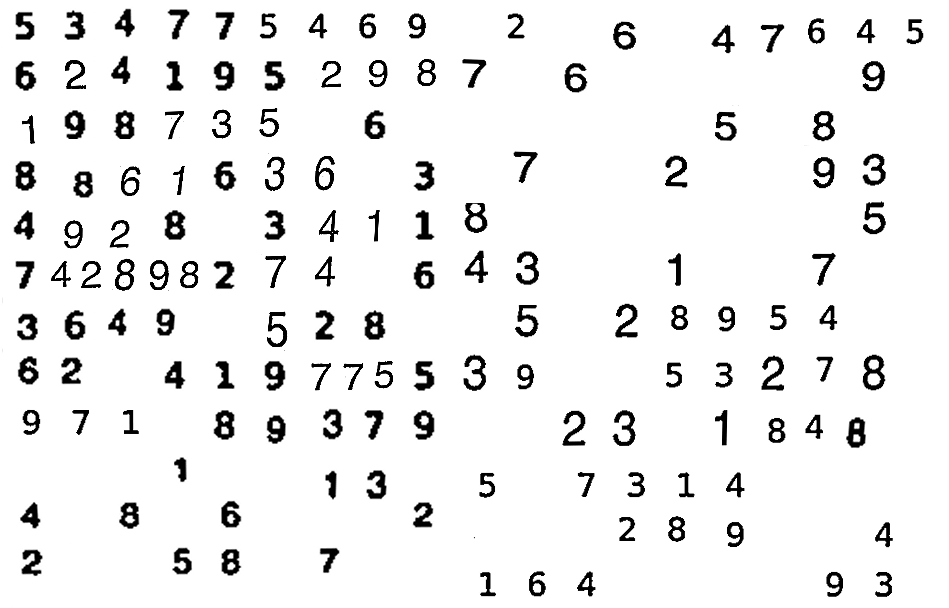
1. We save this image in the “Temp\_Storage” folder with the name “Warped\_Sudoku\_Image.png” for the next part of our script, i.e. the “ocrRecognizer.py” to access it.

This script requires roughly 2 seconds to execute.

**Manual OCR Trainer**

The next script is the “ocrTrainerManual.py”. The procedural overview of this script is as follows.

1. Firstly it loads the training image into Python. This image consists of all the numbers, appearing multiple times in various fonts as shown below.



1. Next it converts the image into a black and white image.
2. Then it applies Gaussian blur to remove noise, if any.
3. Then it applies thresholding to convert the image into a binary one.
4. Next it locates all the contours in the image.
5. Then it starts selecting the contours one by one, every contour basically represents a number, and draw a fitting rectangle around it, for the user to see.
6. Then it accepts a number from the user representing the number detected.
7. The pixel values inside the fitting are resized to 35x25 resolution, and saved in a list called Samples.
8. This procedure is repeated for all the numbers in the image.
9. Finally the samples list is stored in a file called “Samples.data”.
10. And the responses entered by the user are stored in another file called “Responses.data”.

**Automated OCR Trainer**

This script is named as “ocrTrainerAuto.py”. This script is useful while training using the image but with altered dimensions of the resized number selected. This is because the contours detected are always detected in the same order, so as to say the Responses.data file will not be altered in any way if we’re using the same image. So instead of wasting time on inputting the same values again and again we can use this script to auto train our code if we’re using the same image. There is basically no difference between this and “ocrTrainerManual.py” other than the fact that this doesn’t take response from the user.

**OCR Recognizer**

This script is named as “ocrRecognizer.py”. It’ll be used for recognizing the digits present in the “Warped\_Sudoku\_Image.png” image. The sequential algorithm for this is as follows.

1. Firstly it loads the “Samples.data” file and the “Responses.data” file into two lists.
2. Next we train a model using the samples and responses we have using the KNearest function of OpenCV.
3. After that it loads the “Warped\_Sudoku\_Image.png” image.
4. Then it converts it into a grayscale image.
5. Then it applies thresholding to convert the grayscale image to a binary image.
6. Next it locates all the contours in the image and goes through them one by one.
7. Only those contours will be selected which are numbers. This sorting is based on the area of the contour selected.
8. A fitting rectangle is drawn around the number detected and it is then resized to 35x25 pixels.
9. This is then passed to the model we created earlier to detect which number has been selected.
10. Once the number is identified using the k-Nearest Neighbours algorithm based on the location of the contour detected we store the number in a list called “Sudoku\_Text” in its respective position.
11. This “Sudoku\_Text” is passed onto the next script namely, “sudokuSolver.py”

It requires roughly 0.6 seconds to execute this script.

**Sudoku Solver**

This script is named as “sudokuSolver.py” and is used for solving the Sudoku. There is not much to this script since we’ve not written our own algorithm (for solving the Sudoku) in this one. It relies on a python library called dlxsudoku to get things done.

1. It receives the “Sudoku\_Text” list from “ocrRecognizer.py”.
2. The “Sudoku\_Image” is loaded in Python.
3. Then it converts the list into a string of numbers and sends it to the dlxsudoku library. Every black space is represented with a 0, and a filled space is represented with its respective number.
4. This is then processed by the dlxsudoku library and if a solution is available then it returns the solved Sudoku\_Text.
5. In case a solution is not available it throws an error.
6. A non-availability of solution means that the numbers have been incorrectly recognized.
7. In such a case it displays the “Sudoku\_Image” with the detected numbers and prompts the user to enter the row number, column number and the correct digit. Then it modifies the “Sudoku\_Text” list accordingly and returns to step 3.
8. The numbers from the solved Sudoku\_Text are then overwritten onto the Sudoku\_Image at their respective places and then the Solution\_Image is displayed on the screen.

It requires roughly 0.4 seconds to execute this script.

**Main Control Code**

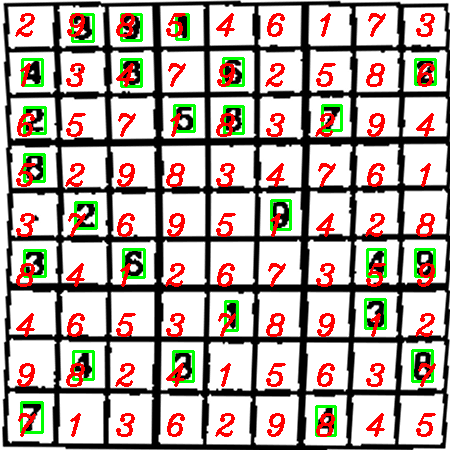
This script is named as “MP2.py”. This is only used as a control to shuffle between the various scripts mentioned above in sequential order.

1. imgProcessing.py
2. ocrRecognizer.py
3. sudokuSolver.py

And at the end of execution all the generated images, stored in Temp\_Storage folder are deleted. All in all the entire code requires a little over 2 seconds to solve and display the output for a given Sudoku.

**Simulation and Experimental Results**

A sample output for the image used above is shown below



**Conclusion**

**Conclusion**

In this project, we have successfully implanted Sudoku recognizer and solver in Python programming language using various libraries like OpenCV, and in the process learned a lot about the Pythonic coding style, computer vision and the vast potential it posses for the future, classification type of machine learning, implementation of algorithms specifically the kNN algorithm for OCR, calling one Python script using another and many more things which would be extremely useful for practical application of our knowledge in the industry.

**Future Scope**

Our project can be used for solving Sudoku puzzles appearing in newspapers in a matter of seconds. The future scope may include optimizing the current algorithm we use or rather coding in C++ as opposed to python for achieving greater speed, and using techniques other than kNN algorithm for OCR, like the SVM algorithm, since it’s more robust and will tend to give more accurate results for OCR.