Extracting visual information from Sudoku puzzles

Valentin Pătrașcu

May 2021

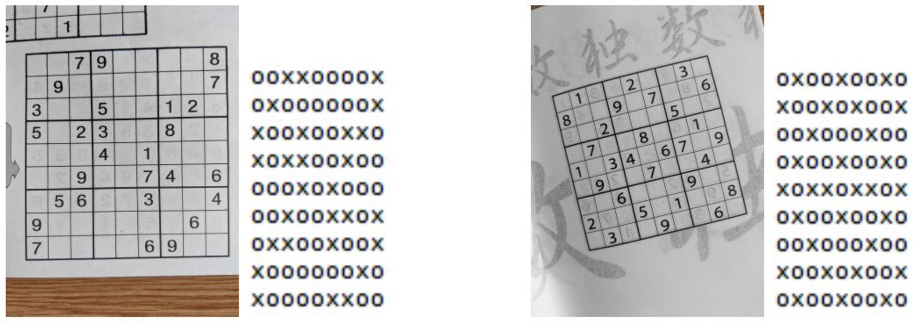
1 Introduction

The goal of this project is to develop an automatic system for extracting visual information from images containing different variations of Sudoku puzzles.

2 Task 1

2.1 Requirement

Write a program that processes an input image containing a Classic Sudoku puzzle and outputs the configuration of the puzzle by determining whether or not a cell contains a digit. Mark empty cells with letter 'o' and the filled in cells with letter 'x'. (see Figure 1)



```
ooxxoooox
oxoooooox
xooxooxxo
xoxxooxoo
oooxoxooo
ooxooxxox
oxxooxoox
xooooooxo
xooooxxoo
```

```
oxooxooxo
xooxoxoox
ooxoooxoo
oxooxooxo
xoxxoxxox
oxooxooxo
ooxoooxoo
xooxoxoox
oxooxooxo
```

Figure 1: Examples of two Classic Sudoku puzzles and their corresponding configuration for Task 1

2.2 Solution

The solution of Task 1 includes multiple steps:

- Change the perspective of the images in order to have an orthogonal alignment of the sudoku grid with the x and y axes
- Determine the horizontal and vertical lines of the sudoku grid using the previously calculated binary image of the edges with respect to the changed perspective.
- Classify the patches that are determined by the vertical and horizontal lines.

The function that takes the input image and change its perspective is called change_perspective and it does a handful of operations until the final image is determined.
The first operation is to increase the brightness by a factor of 1.2 to remove some of the black noise. (see Figure 2)

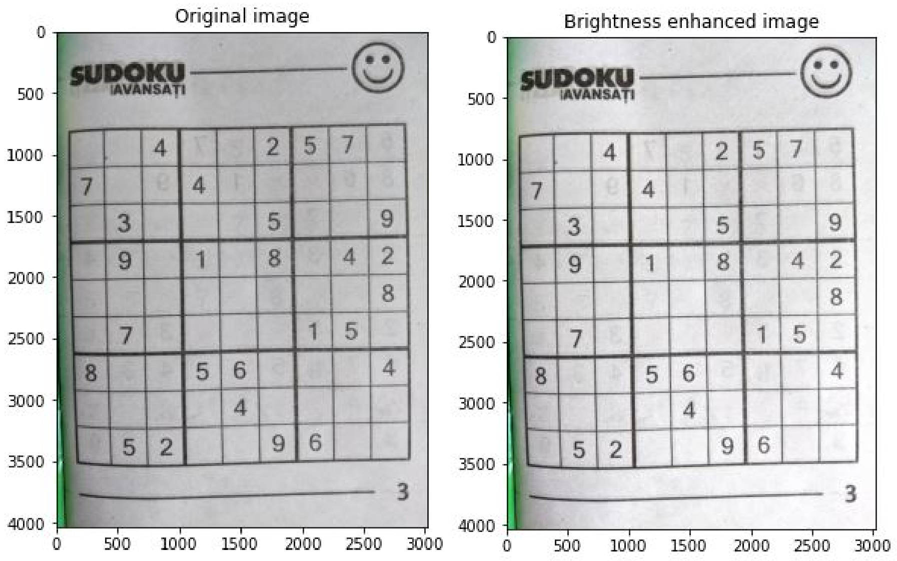


Figure 2: Left image the original image and the right figure is the image after applying the brightness enhancing procedure.

The next step is to get the binary image of the important features in the image in order to determine the contour (in this step an adaptive thresholding is enough because we do not want a very accurate edge detection). The brightness enhanced image is firstly converted to grayscale, then a median filter is applied to remove a portion of the noise in the image and after that, the adaptive thresholding method is used. (see Figure 3)

After we have the binary image given by the adaptive threshold we take the biggest contour which represents our sudoku grid. Then we calculate the points of the smallest rectangle that comprises our contour. We will use these points as the source points of the warping. (see Figure 4)

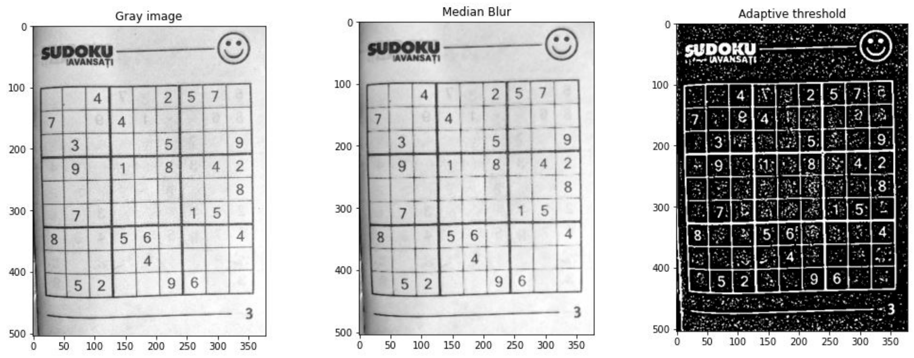


Figure 3: Left image is the image after converting to grayscale. The middle figure is the image after applying a 3x3 median filter. Right figure is the image after applying an adaptive thresholding.

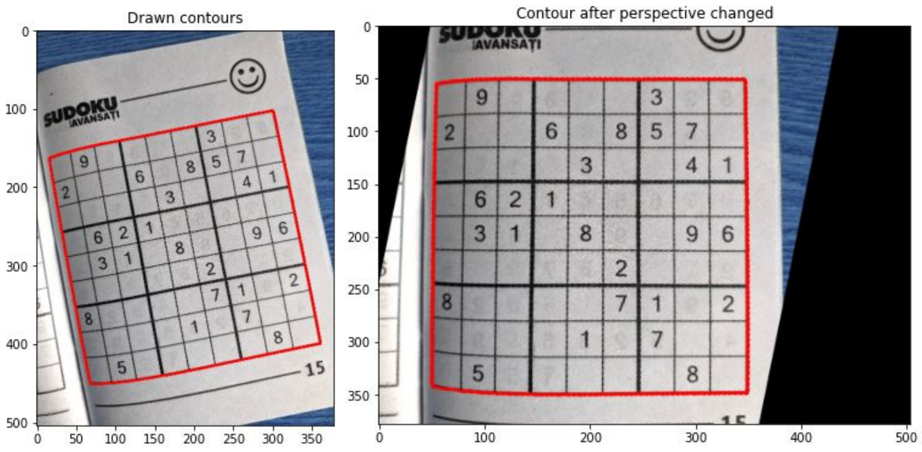


Figure 4: Left figure is the original image with the contour drawn. Right figure presents the contour after changing perspective.

Once we have the warped image, the next procedure is to detect the edges using Canny detector. A morphological dilation is applied on the binary image resulted from Canny detection to expand the activated pixels. (see Figure 5)

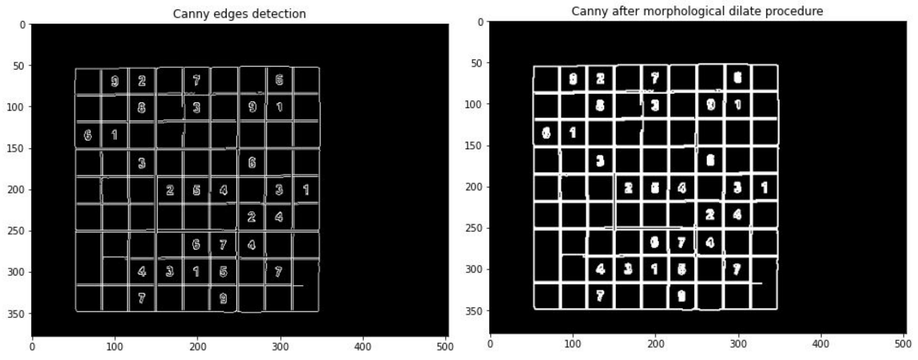


Figure 5: Left figure is the image resulted after Canny detection with low_threshold = 90 and high_threshold = 255. Right figure presents the image after dilation with 2x2 kernel.

After we have the pixels from the edges activated, we proceed to detect the lines that construct the sudoku grid. Hough transform is applied in order to determine the lines that unite the activated points. Two types of lines are considered, the horizontal and vertical ones (we assume that the perspective transform was done properly). To separate the lines, the slope was considered for each of the types. In order to have the lines exceeding the sudoku borders, a method to extend the lines was implemented (it takes into account the line equation given two points and generate new points on that line that exceed the limits of the grid). Also, lines that were too close to each other were removed. (see Figure 6)

The final step was to determine the content from each of the sudoku cells. In order to do this, the grid was divided into 9x9 patches, each of this patches being classified in order to detect if there is a digit present or not. The patches were extracted using the information given by the lines that separate one cell from its neighbors. The points of intersection between the two horizontal and two vertical lines that bounds the cell were calculated and then, the information between this irregular quadrilateral was extracted. To exclude the possibility of having the pixels from the bounding lines, a padding was used.

To classify the content inside each cell a simple method was used. We assumed the fact that the cells that contain digits have different values of the intensity of the pixels and we compute the standard deviation. If the standard deviation of the values of the pixels was higher than a threshold then that cell contains a digit. (see Figure 7)

The matrix with 'x' and 'o' as shown in (Figure 1) was generated for each of the images and the content was saved into a file corresponding to each image.

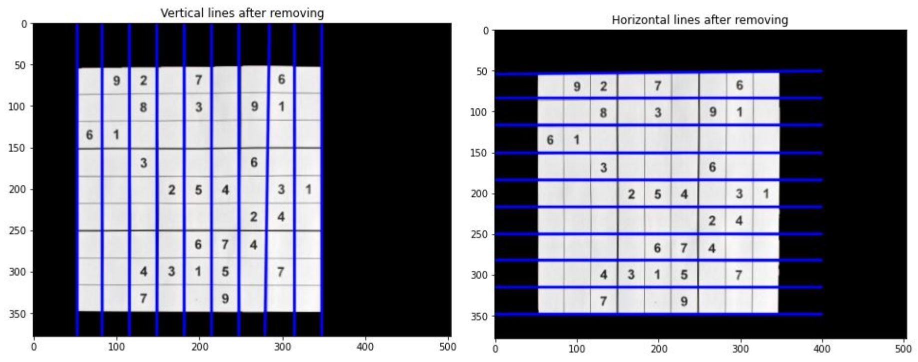

Vertical lines after removing
Horizontal lines after removing

Figure 6: Left figure is the warped image with the final vertical lines drawn (we considered a valid slope between -1.5 $\geq$ slope or 1.5 $\leq$ slope) . Right figure presents the warped image with the final horizontal lines drawn (we considered a valid slope using the equation -0.5 $\leq$ slope $\leq$ 0.5).

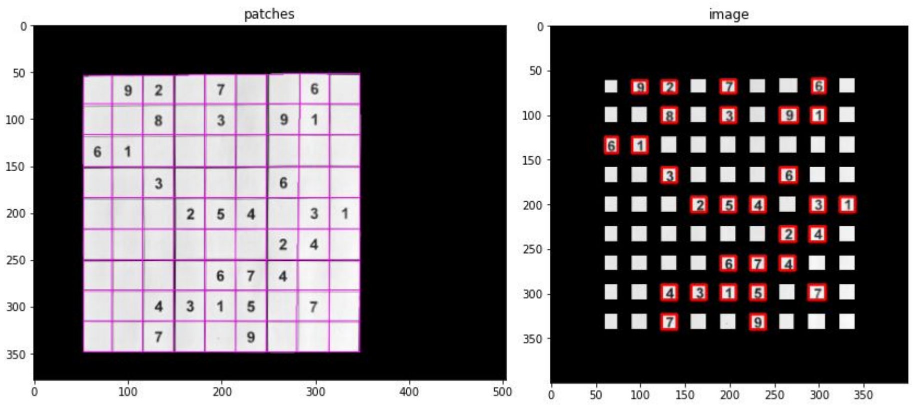
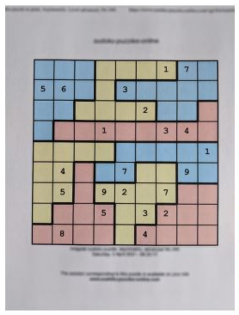


Figure 7: Left figure is the warped image with the patches drawn. Right figure presents the classified patches according to the standard deviation.

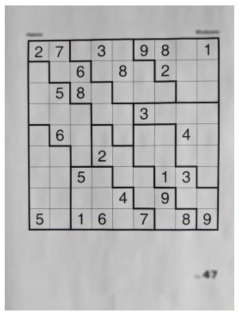
3 Task 2

3.1 Requirement

Write a program that processes an input image containing a Jigsaw Sudoku puzzle and outputs the configuration of the puzzle by: (1) determining whether or not a cell contains a digit; (2) determining the irregular shape regions in the puzzle. (see Figure 8)



```
1o1o1o2o2o2o2x3x3o
1x1x1o2o3x3o3o3o3o
1o1o2o2o2o2x3o3o4o
1o4o4o4x4o4o4x4x4o
5o5o5o5o6o6o6o6o6x
5o5x7o6o6x8o8o6x6o
5o5x7o8x8x8o8x9o9o
5o7o7o7x8o8x9x9o9o
7o7x7o7o8o9x9o9o9o
```



```
1x1x2o2x2o3x3x3o3x
4o1o1x2o2x2o3x3o3o
4o4x1x1o2o2o2o3o3o
4o4o4o1o1o5x5o5o5o
6o4x4o7o1o8o8o5x5o
6o6o4o7x7o8o8o5o5o
6o6o9x9o7o7o8x8x5o
6o6o9o9o9x7o7x8o8o
6x6o9x9x9o9x7o7x8x
```

Figure 8: Examples of two Jigsaw Sudoku puzzles and their corresponding configuration for Task 2

3.2 Solution

The point (1) was discussed in depth in 2.2
For the point (2) the approach is as it follows:

- Generate two Gabor filters and apply them on the warped image in order to determine where the thick lines are present.
- Create a mask of the responses given by the activated pixels from each Gabor filter.
- Check if the region (either vertical or horizontal) between two patches contains any activated pixels that corresponds to the thick lines. Store the information about the vertical and horizontal lines.
- Apply a depth first search on the stored information about thick lines to calculate the id of the cells in each of the 9 sudoku regions.

The two Gabor filters were built using two theta parameters 0 and $\pi/2$ (because we want to determine the horizontal and vertical activated thick lines). (see Figure 9)

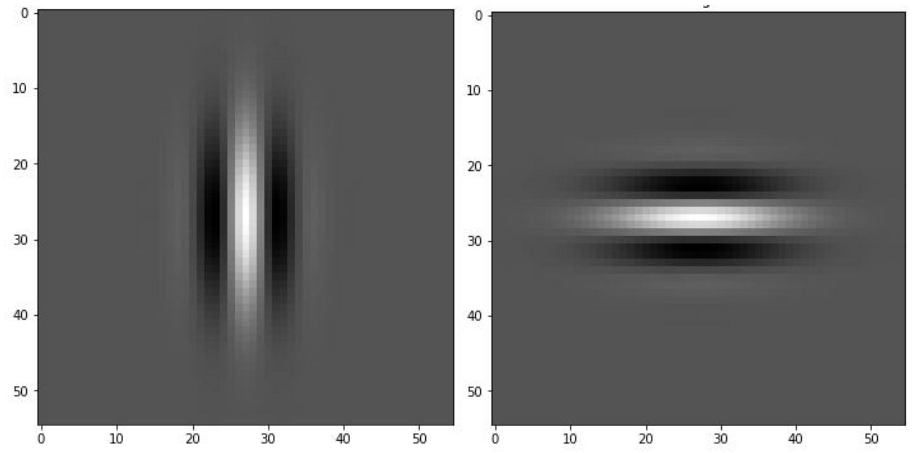


Figure 9: Gabor filters with theta = [0, $\pi/2$] and lambda = 10.

The results of each filtering was added together and the final mask was determined. (see Figure 10)

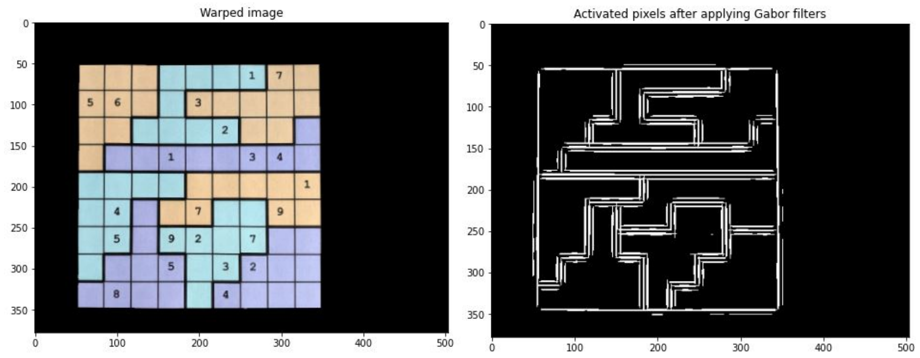


Figure 10: Initial warped image and the activated mask resulted from applying the Gabor filters.

The next procedure was to use the patches that were calculated at the step (1) and see if there are activated pixels in the regions between two patches, If more than two activated pixels were found between two patches then a thick line is present there. (see Figure 11)

The final operation was to apply a recursive depth first search on the grid ids of the cells having the knowledge about the position of the thick lines.

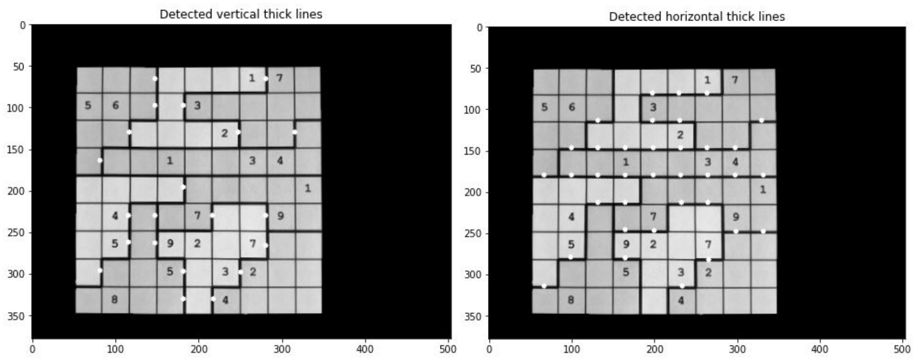
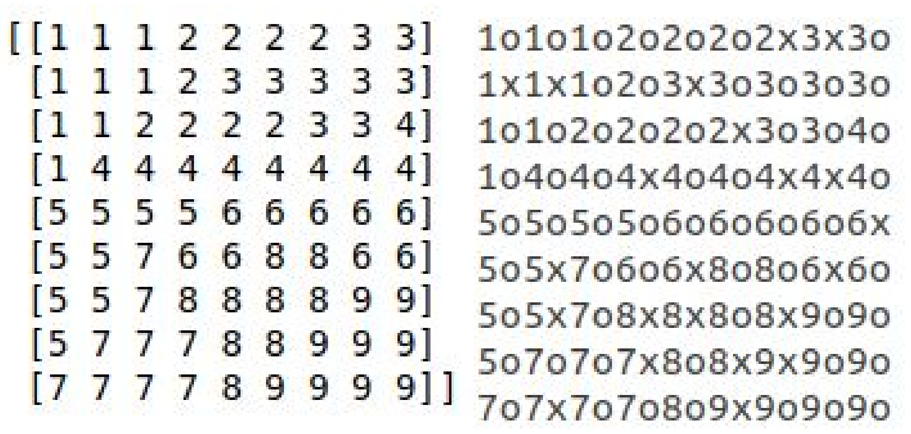


Figure 11: The detected thick lines. The left figure shows the vertical thick lines. The right figure shows the horizontal thick lines.

The ids given by the dfs and the presence of the digits in each cell given by the (1) step were stored into a matrix and saved to disk.



```
[[1 1 1 2 2 2 2 3 3]
 [1 1 1 2 3 3 3 3 3]
 [1 1 2 2 2 2 3 3 4]
 [1 4 4 4 4 4 4 4 4]
 [5 5 5 5 6 6 6 6 6]
 [5 5 7 6 6 8 8 6 6]
 [5 5 7 8 8 8 8 9 9]
 [5 7 7 7 8 8 9 9 9]
 [7 7 7 7 8 9 9 9 9]]
```

```
1o1o1o2o2o2o2x3x3o
1x1x1o2o3x3o3o3o3o
1o1o2o2o2o2x3o3o4o
1o4o4o4x4o4o4x4x4o
5o5o5o5o6o6o6o6o6x
5o5x7o6o6x8o8o6x6o
5o5x7o8x8x8o8x9o9o
5o7o7o7x8o8x9x9o9o
7o7x7o7o8o9x9o9o9o
```

Figure 12: The left image corresponds with the results given by the depth first search and the right image represents the final output.

4 Task 3

4.1 Requirement

Write a program that processes an input image containing three sides (each side is a sudoku puzzles) of a Sudoku Cube and outputs the coresponding Sudoku Cube by: (1) localizing the three sudoku puzzles in the image that form the sides of the Sudoku Cube; (2) inferring their position in the Sudoku Cube using the constraint that the digits on the common edge of two sides must be the same number; (3) warping each side on the corresponding side of a given template for the Sudoku Cube.

4.2 Solution

The proposed solution consists on a set of defined steps:

- Extract edges from the initial image of the three sudoku grids.
- Extract the contour corner points for each of the grids.
- Apply the perspective transform for each of the grids.
- Classify each cell according to the digit present in it.
- Check the matching between the grids in order to respect the given requirements for a valid Sudoku Cube.
- Add each sudoku grid into the template according to the template cube shape.

To extract the edges, contours and to apply the perspective transforms the same procedures as in the 2.2 were used. (see Figure 13)

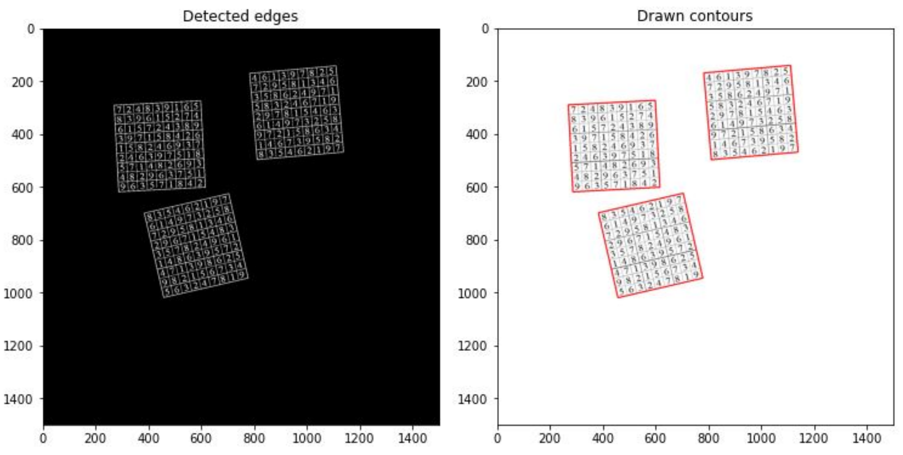


Figure 13: Detected edges on the initial image and the drawn contours after they were found in the image.

The method used to classify the digits present in each cell was to compare the image from each cell with the mean image of 10 images of each of the 1-9 digits. In order to know which digit is present in the cell, the minimum euclidean distance to one of the mean images was considered. (see Figure 14 to see examples of mean images previously classified)

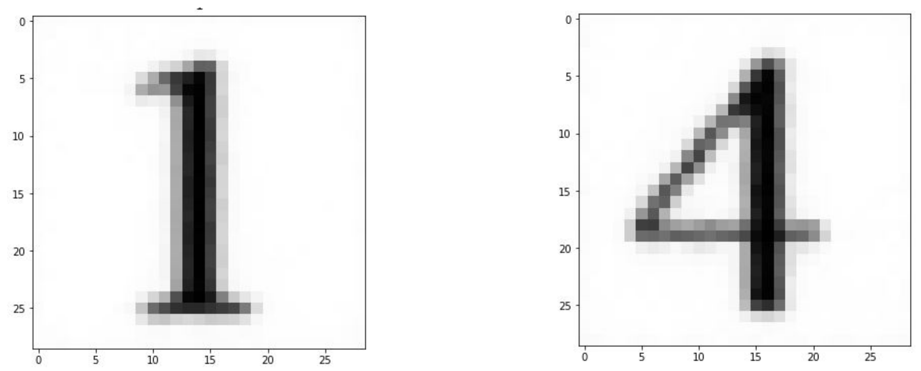


Figure 14: The mean image of the digit 1 and 4.

After the detection of the digits present in each grid, the a matching method was implemented that took into consideration the configuration of each of the grids and the requirements for a valid Sudoku Cube. After the matching process was done, the next step was to add each of the grids to the template image. (see Figure 15)

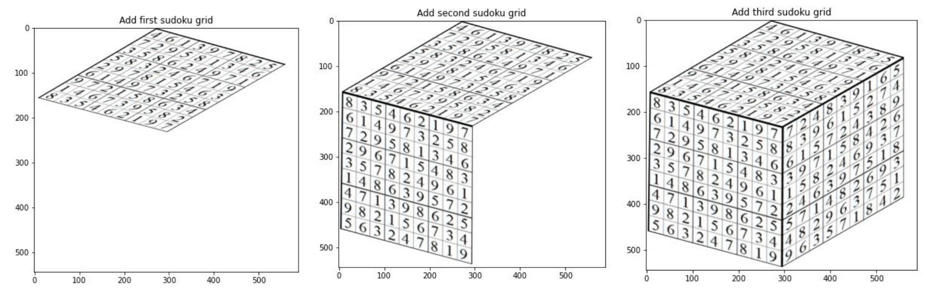


Figure 15: Add the three sudoku grids to their corresponding positions according to the cube configuration.

5 Conclusion

The project presented is intended to highlight the usage of different Computer Vision techniques to extract visual information from a real life scenario: a Sudoku game. The CV techniques used in this project varied from edge detection, contour extraction and application of different filters (adaptive threshold, morphological and Gabor) to the usage of Hough transforms and perspectives changes.