Batuhan Erden

batuhan.erden@ozu.edu.tr

CS 423/523 – Introduction to Computer Vision

Assignment 3

13.01.2018

Sudoku Puzzle Solver with Augmented Reality

Abstract

In this assignment, I have used Python and OpenCV to detect the Sudoku puzzle, solve it and display the solution on both the images and videos. I have used MNIST dataset to train my model. The model is trained with Convolutional Neural Network layers and Keras is used as a machine-learning library.

1 Introduction

In this report, I will include the steps I have taken to build this Augmented Reality based Sudoku Solver and difficulties I have faced with. First of all, the recognition accuracy on the images is 85.60%. In order to solve the Sudoku, you need to know every digit correct. Therefore, the algorithm could only solve the puzzles in 21 of the images. If we compare it with the number of images which is 203, 10% of the Sudoku puzzles are solved. There are some images where the Sudoku was still solved but the accuracy was not 100%. This is the result of storing the best 3 predictions for each digit. Instead of storing a 9x9 2D Matrix for Sudoku array, I stored a 9x9x3 3D Matrix for Sudoku array such that the best 3 predictions for each digit are stored. After recognizing the digits, the Sudoku is solved by a solver algorithm. Therefore, the Sudoku solution was ready to be shown on the videos/images. I have warped the Sudoku puzzle to bird eye view by applying Affine Transform, then use that warped image to detect and solve the Sudoku puzzle. After achieving that, I have drawn the Sudoku with its solution on the warped image. Lastly, I applied inverse transformation to display it on the original image.

2 Extracting the Bounding Box

The first step was to detect the bounding box of the Sudoku. By using that bounding box and the maximum approximation, I applied warp perspective to obtain the bird-eye view of the Sudoku puzzle. The example of a bird-eye view can be shown in the figures below. *Figure 1* shows the actual video frame/image while *Figure 2* shows the Sudoku puzzle warped.

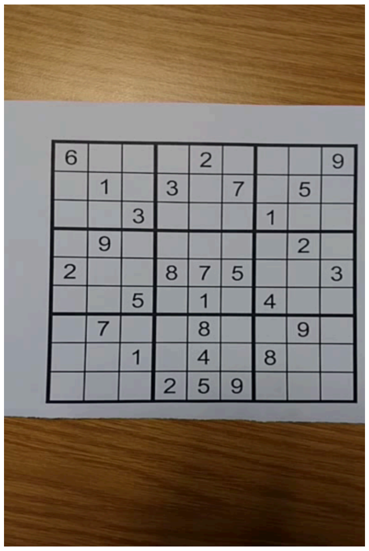


Figure 1: Actual Video Frame

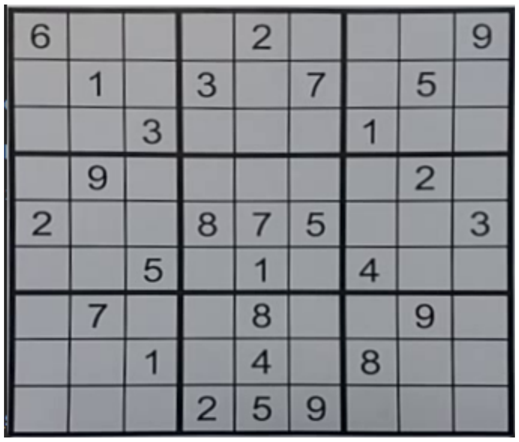
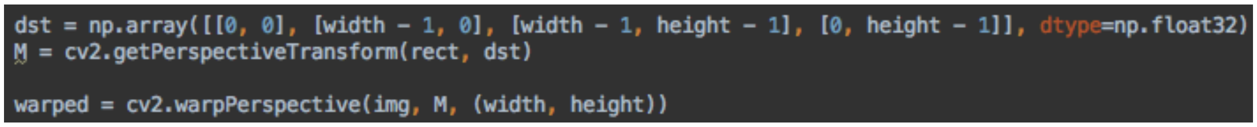


Figure 2: Warped Sudoku Puzzle

The process of warping an image is achieved by obtaining the transformation matrix. Thus, we need to have the 4 corners of the bounding box accurate. In the *Figure 3*, the most significant steps to apply warp perspective are shown.



```
dst = np.array([[0, 0], [width - 1, 0], [width - 1, height - 1], [0, height - 1]], dtype=np.float32)
M = cv2.getPerspectiveTransform(rect, dst)

warped = cv2.warpPerspective(img, M, (width, height))
```

Figure 3: The most crucial steps to obtain the warped image

3 Digit Extraction and Recognition

Finding the Sudoku and the rectangles within was the most accurate way of extracting the digit. That is, the boxes are free from the boundaries. They only contain the digits which makes the digit recognition easier. However, there were times when the number of rectangles found insufficient. In such cases, the algorithm was only recognizing the digits in the rectangles found and some of the images were left unrecognized. In this part of the project, by making the use of the warp perspective, that problem is solved. Since I get only the Sudoku puzzle after applying the warp perspective, dividing the warped Sudoku puzzle into 81 same-sized pieces solved the problem. Yet, in some images, the boundary was still a problem but it was better than before and a nicely warped image wasn't causing a problem. The extracted digit can be shown in *Figure 4*.

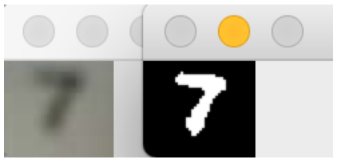


Figure 4: After thresholding the image

After obtaining a black-white image with a digit inside, I resized it to 28x28 and divided each element of the matrix with 255 to be recognized by MNIST data. I have also augmented the MNIST data so that it contained empty, blurred, rotated, shifted and zoomed images. It did definitely increase the accuracy.

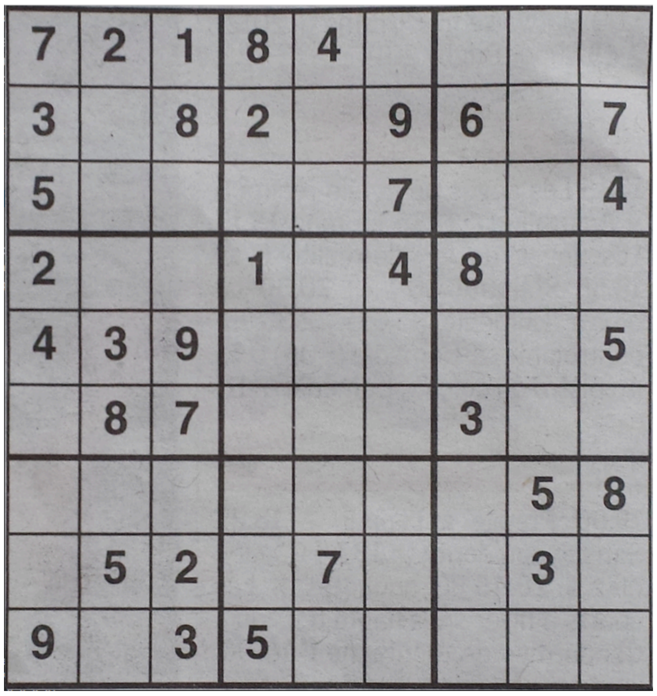
In recognizing the digits, the model was predicting 3 labels for each digit. That is, instead of storing a 9x9 2D Matrix for Sudoku array, I stored a 9x9x3 3D Matrix for Sudoku array such that the best 3 predictions for each digit are stored. This is used while solving the Sudoku puzzle. In the cases where the solution cannot be found, the algorithm simply changes the index of the prediction to try the next best possible prediction to solve the Sudoku.

4 Solving the Sudoku Puzzle

After obtaining the Sudoku Puzzle, we need to solve it. To do that, a simply Sudoku solving algorithm is used. I found the algorithm from the resources in the web and modified it a little bit to use. As described above, my Sudoku array was a 3D array with shape (9, 9, 3). In case the solution for the puzzle could not be found, the algorithm was trying the other 2 predictions of a particular digit to solve the puzzle again. On the other hand, in such cases where the prediction is less than 85-90%, the Sudoku puzzle could not be completely solved. So, I filled the unsolved grids with random numbers. As a result, most of the Sudoku results are wrong. (21/203 puzzles are solved correctly)

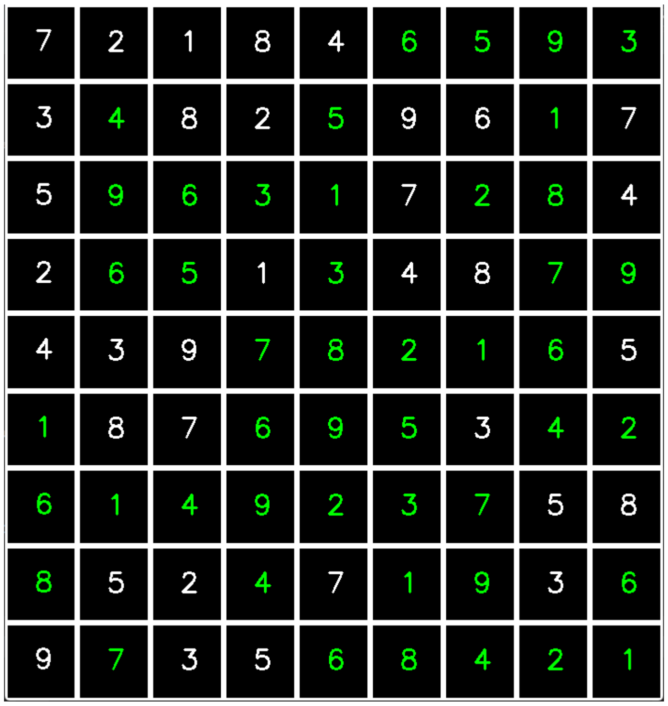
5 Drawing the Sudoku Solution on the Warped Image

After solving the Sudoku, it is time to draw it on the warped image. Before starting the first assignment, I have seen a video of Sudoku Solver and it changed my perspective. In that video, the solution shown was designed perfect. So, I decided to design my own Sudoku. In my design, the background is white and the boundaries of each box are white. The original digits on the Sudoku are also white and the solution is green. The original warped image and solution image are shown in the *Figure 5* and *Figure 6*. Note that white digits are the original digits whereas green digits are the solved digits. In this picture, the recognition was 100%. Thus, the solution is correct.



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

Figure 5: Warped Sudoku Puzzle

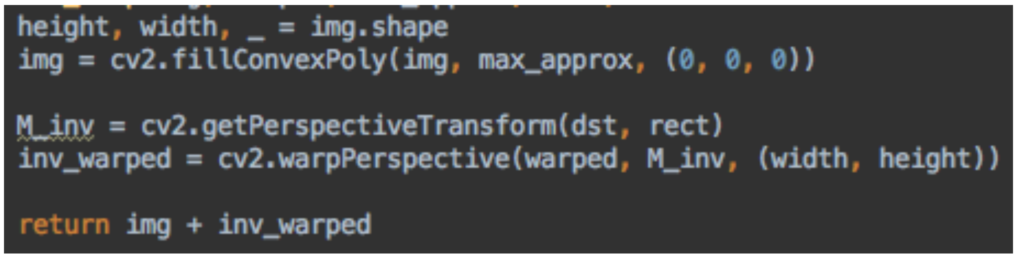


7	2	1	8	4	6	5	9	3
3	4	8	2	5	9	6	1	7
5	9	6	3	1	7	2	8	4
2	6	5	1	3	4	8	7	9
4	3	9	7	8	2	1	6	5
1	8	7	6	9	5	3	4	2
6	1	4	9	2	3	7	5	8
8	5	2	4	7	1	9	3	6
9	7	3	5	6	8	4	2	1

Figure 6: Warped Sudoku Puzzle Solution

6 Applying the Inverse Transformation

After solving the Sudoku, it is time to draw the solution on the original video/image. The algorithm was pretty simple. It was the inverse of warp transform. First of all, the area where the warped image is extracted from should be painted to black to accept another image. The **dst** matrix and the **rectangle** returned from the warp transform is used here. Only difference is that the ordering is reversed. Applying reversely applying them to perspective transform, I obtained the **M inverse**, which is then be used to apply warp perspective on the warped image to obtain the original one. At last, by a simple addition operation, the original image is obtained with a solution drawn in it. In *Figure 7*, the most crucial steps of the inverse transformation algorithm is shown and in *Figure 8*, the latest result on the original image is shown.



```
height, width, _ = img.shape
img = cv2.fillConvexPoly(img, max_approx, (0, 0, 0))

M_inv = cv2.getPerspectiveTransform(dst, rect)
inv_warped = cv2.warpPerspective(warped, M_inv, (width, height))

return img + inv_warped
```

Figure 7: The most crucial steps of inverse transformation

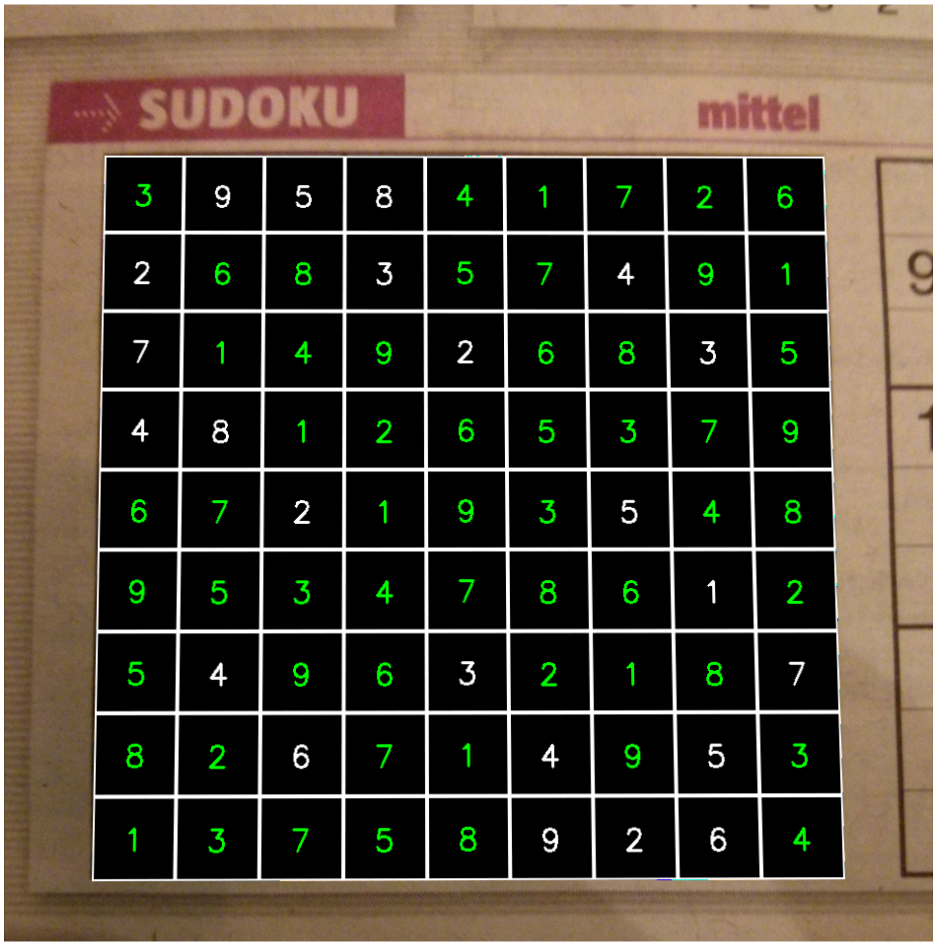


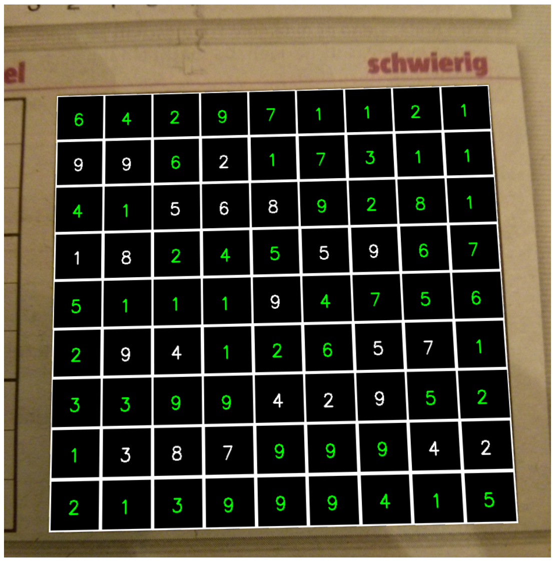
Figure 8: Final Sudoku Puzzle

7 Results and Discussion

In conclusion, this project has helped me a lot in learning augmented reality and processing video frame in real time. In the future, I intend to develop my algorithm even further and make it perform well even in the hardest environment. This 3 part project series was very fun to develop and test. I hope I could improve my computer vision skills in the future.

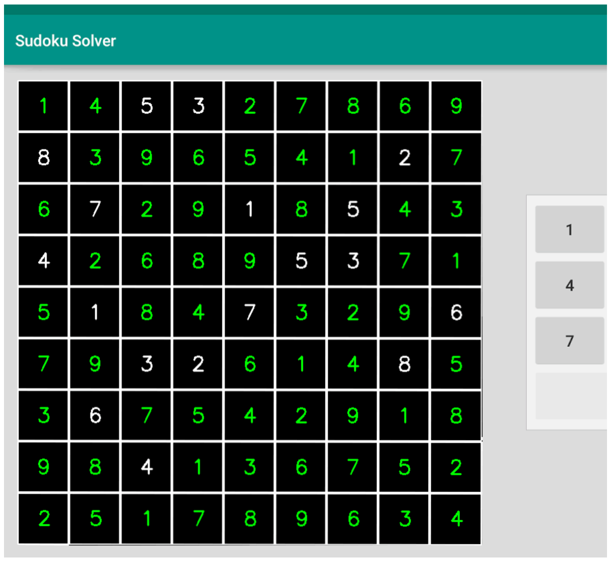
For the results, the results on the images were very nice. The digit recognition accuracy was 85.60% and in the images where the accuracy is above 95%, most of the solutions are correct. An example showing this case can be shown in *Figure 9* (Recognition accuracy was 94% but the solution is wrong). Furthermore, *Figure 10* demonstrates a fully solved Sudoku Puzzle.

On the other hand, for the videos, the results of the AR were also nice but the recognition was worse. Due to the quality and hardness of the video, the accuracy was not that good most of the times. The difficulties I faced while developing the AR part on the videos is the rotation of the camera. I have failed handling the rotation. There were some other times where the puzzle's boundaries are left out of screen and these also prevented me to draw the Sudoku. However, I think the main problem is that the digit recognition was not that good on the videos as it was in the images. I think, if the video quality was better and camera was stationary while detecting the Sudoku and recognizing the digits, my algorithm would perform better and solving the Sudoku puzzles on videos. An example Sudoku solution in video frames can be seen in both *Figure 11* and *Figure 12*. Although the recognition and the solution is not that good, I think the AR part looks fine most of the times.



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

Figure 9: Incorrect Sudoku Solution



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

Figure 10: Correct Sudoku Solution

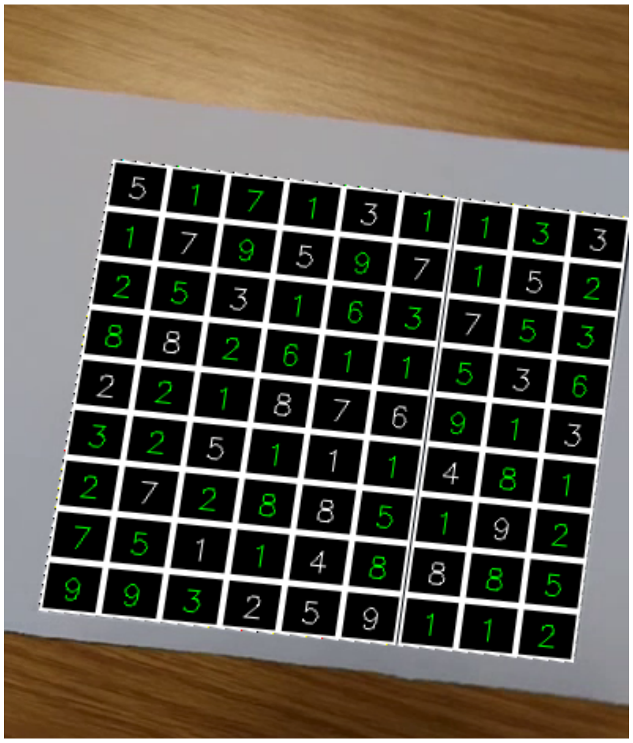


Figure 11: Solution of Sudoku Puzzle on test-08.mp4

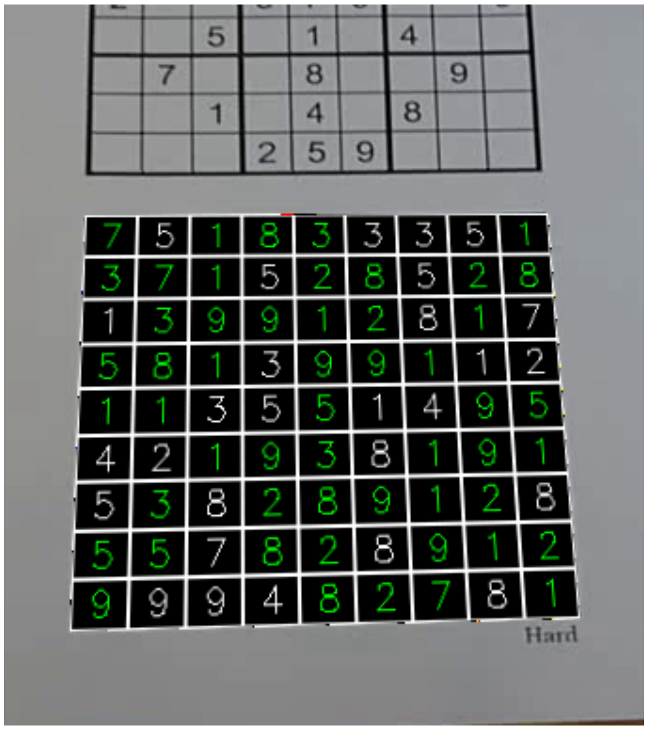


Figure 12: Solution of Sudoku Puzzle on test-01.mp4