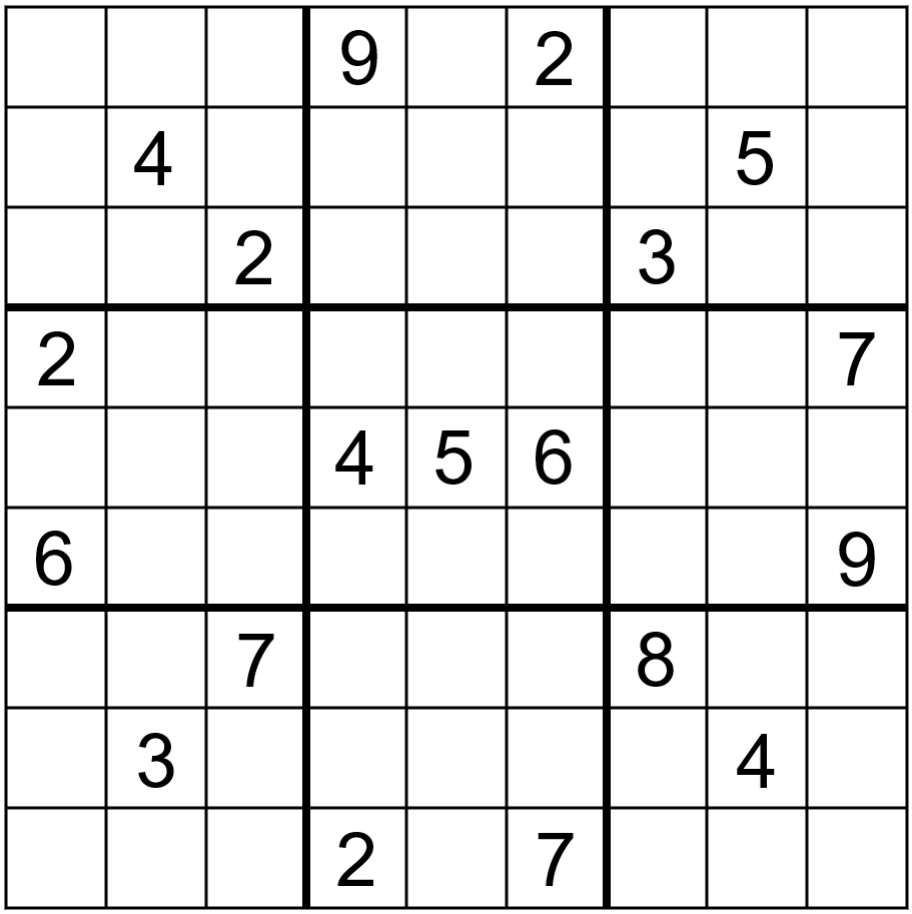
Sudoku Recognition Using OpenCV

MCEN 5228: Mechatronics and Robotics 2 Grad Project

Nicolas Garzione

May 3, 2022



1 Introduction

The goal of this project was to develop a Sudoku solving program that could solve a physical Sudoku board held up to a camera. The development of a Sudoku solving program requires three main parts: the computer vision, the numeric recognition AI, and the Sudoku solving algorithm. This project explores the first part and its implementation into such a program.

2 OpenCV

This project was developed in Python and OpenCV was used for the computer vision implementation. OpenCV can access the webcam of a laptop which allows accessibility to a camera for this project. OpenCV is also well documented and its multitude of libraries made the process of manipulating images easier. It can also overlay shapes on top of images, making the demonstration of its implementation easy to see.

3 Implementation

3.1 Approach

The computer vision aspect of this project had two main goals: identify a Sudoku board and alter it so a numerical recognition AI would be able to easily identify the numbers. Identifying the Sudoku board required finding its signature square shape and rejecting any other shape seen in the camera. Warping the imaging was more difficult, requiring the corners of the board to be identified and then snapped on to a perfect square shape to warp the image from a "bent" square to a perfect square. Accomplishing both of these should result in a Sudoku board similar to Figure

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

Figure 1: Expected Output

3.2 Contour Recognition

To allow the program to recognize contours, and in extension shapes, the original image must be modified. Raw images contain noise which causes contour-finding algorithms and results in imperfect tracing of what should be straight lines. A common method to alter an image for computer vision readability is as follows: grayscale, blur, and threshold.

Converting an image to grayscale reduces the amount of data being handled by the algorithm and in return reduces the computation required. Figure 2 demonstrates this conversion. Color is not needed for this application, all the program needs to do is ind the simple black outline of the Sudoku board.

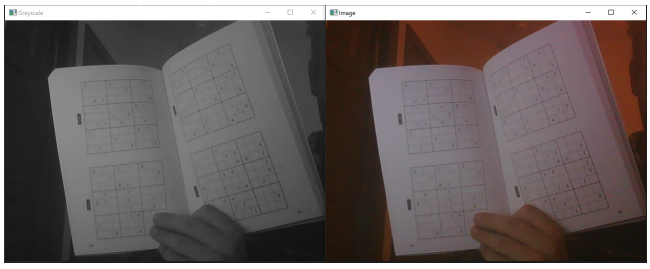


Figure 2: Grayscale Conversion

A Gaussian blur is applied to the grayscale image. By blurring the image, the equivalent of a low-pass filter is applied, filtering out some of the noise. This will help the algorithm better detect straight lines by reducing the likelihood of noise being added along the edges of the lines. This can be observed in Figure 3.

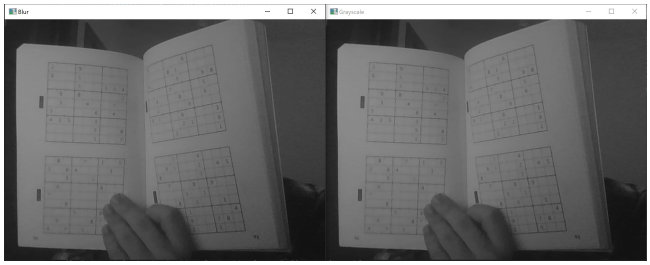


Figure 3: Blur Conversion

Finally, a threshold is applied to the image. Observing the previous steps, the steps so far have not increased the visibility of the lines being recognized. To increase their visibility, the image is converted to a white-on-black image. Observing Figure 4, it can be seen that the lines on the Sudoku board are much more accentuated. This will allow the algorithm to better detect these lines.

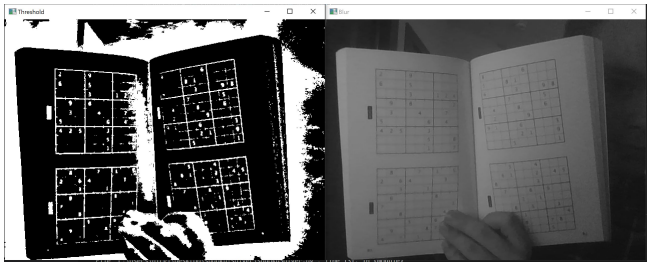


Figure 4: Threshold Conversion

Applying these steps allows the OpenCV contour recognition algorithm to be able to find clean, straight lines, when before there would just be noise. Figure 5 demonstrates this result, with a clear green square outlining the Sudoku board. Implementing a simple algorithm to find the largest complete contour on the image is the last step in the process. The Sudoku board has been successfully identified.

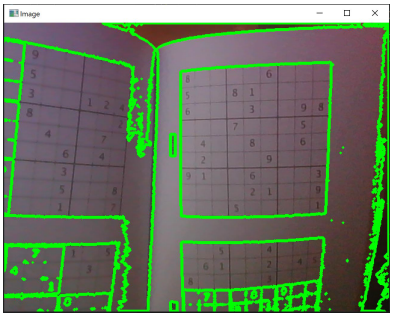


Figure 5: Contour Recognition

3.3 Shape Fitting

Figure **??**, while square-like, is not a square shape that can be used. To continue with the perspective warp step, a square must be fitted to this shape. The largest is taken and it is fitted to a shape of four sides. This shape replaces the contours originally defined and the corners of this shape can be used as reference points

due to their location accuracy. This also allows anything but a square to be rejected from the Sudoku solving algorithm further down the line. Figure refclip7 demonstrates the new fitted shape replacing the previous contour shape.

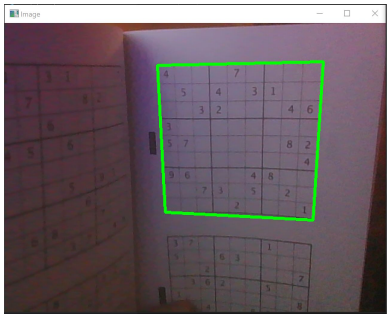


Figure 6: Shape Fitting

3.4 Perspective Warp

Perspective warp is the process of stretching a shape so it fits in to another shape. In the process, this evenly stretches the contents of the shape. In this case, it is used to take a "crooked" square and stretch it in to a perfect square so the numeric digits can be read accurately and precisely.

To perform this operation, the four corners of the warped square must be identified. More importantly, their order needs to be identified. The perspective warp algorithm requires the corners to be inputted in the order of: bottom left, bottom right, top right, and top left. A corner-identifying algorithm was implemented for this purpose. These corners are shown in Figure 7.

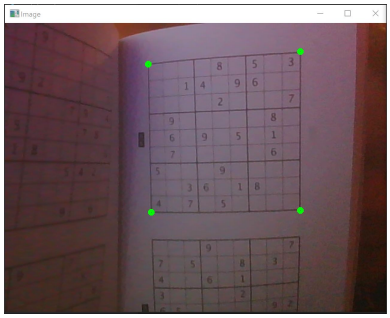


Figure 7: Corner Recognition

With these operations performed, the Sudoku board has been successfully identified and it can be warped in to a square shape for easy readability.

4 Results

Thee resulting warped board is shown in Figure 8. This is an average result for this algorithm. As can be observed, there is a lot left to desire from this algorithm. The perspective shifting needs to be improved to completely eliminate the bent lines within the Sudoku board. This could be accomplished by taking reference points within the board, at the intersecting lines. However, this would require a higher resolution camera and better lighting. This test was conducted on a subpar camera and with no back lighting. Adding these two components, and adding more reference points, could help improve the effectiveness of the perspective warping.

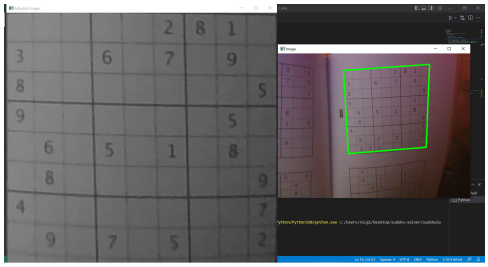


Figure 8: Final Result of Detection Algorithm

5 Conclusion

OpenCv implemented in Python can be successfully used to recognize a Sudoku board. The ability to unwrap the board on to a perfect square needs improvement and these changes must be implemented before any future steps can be taken.

6 Future Work

The next steps for this project will be to fix the perspective warping. This must be perfected for the numerical recognition to he effective. If the computer is not able to read straight across or down a line, it will not be able to fully encapsulate every number for its recognition. After this is fixed, the numeric recognition AI can be implemented. This will either use a pre-trained model or a model custom trained for this purpose. This will allow the digits on the board to be recognized and stored in a matrix. This matrix can then be fed in to a back-tracking algorithm to solve the Sudoku puzzle itself. A stretch goal for this project will be to project the finished puzzle back on to the original image so the user can view the numbers within the squares they should be in for the solved puzzle.

Appendix A: Python Code

DESCRIPTION

`https://github.com/nicolasgarzione/sudoku-solver`

Appendix A: Demonstration Video

DESCRIPTION

`https://youtu.be/ZARNxXP7sAQ`