

# Example - Processing Image

In this document we will go through the different parts of the data pipeline and visualize what happens with the image in this process. This document does not contain any code, please refer to the source code under ./project at <https://github.com/EmpanS/Project-Sudoku>.

Simplified, the data pipeline (the processing) can be split into 5 consecutive parts as shown below.

1. Gaussian smoothing and adaptive thresholding followed by an inversion of color.
2. Find contours and corner points of the grid.
3. Warp and crop the Sudoku board.
4. Divided the board into 81 images. For each image check it if contains a number:
  - it contains a number, extract the number and center it in a new image
  - if it does not contains a number, create a totally black image
5. Join all images of centered number and black images in a list.

We will go through each part above and visualize what happens with the image after each steps. We will process the image shown in Figure 1.

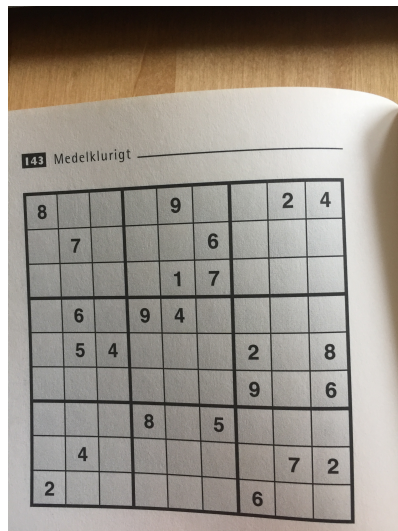


Figure 1: Image of a Sudoku board.

## 1 Gaussian smoothing, adaptive threshold, inversion of color

When we apply Gaussian smoothing and adaptive threshold, we get a much cleaner image. We have also inverted the colors as seen in Figure 2.

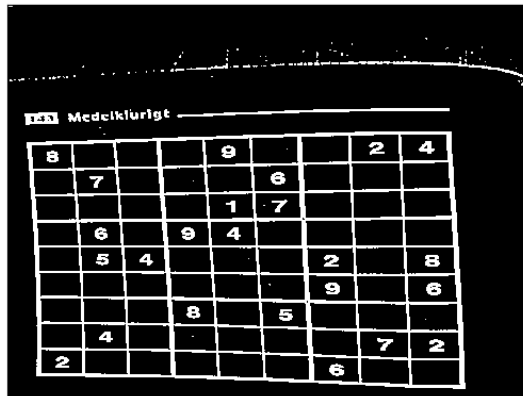


Figure 2: Image with applied Gaussian smoothing, adaptive threshold and inverted colors.

## 2 Find corner points of the grid

We can see that the program finds the corner points in Figure 3 below.

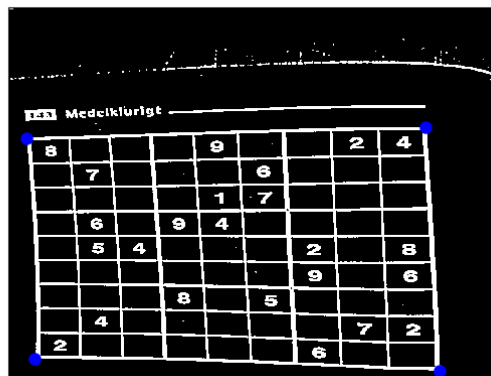
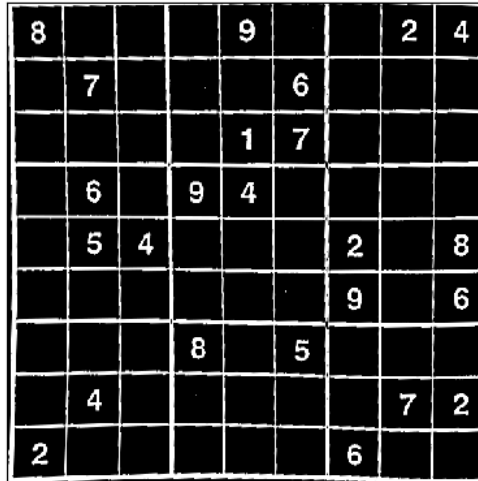


Figure 3: The image in Figure 2 but with the corners of the grid marked.

### 3 Warp and crop the Sudoku board.

Then, using the corners, the grid gets warped and cropped resulting in an image shown in Figure 4.



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

Figure 4: The resulting image after the image in Figure 3 had been warped and cropped.

### 4 Divided the board into 81 images and check if each image contain a number

Then, the image in Figure 4 gets divided into 81 images of equal size. In each image, everything but the most centered feature gets removed. I have made a demonstrative example in a jupyter notebook, see the jupyter-notebook *Example - extracting feature.ipynb* in the folder `./docs` at <https://github.com/EmpanS/Project-Sudoku> that explains the algorithm behind. Following, each image gets analyzed based on the two conditions defined in the beginning of this document. In Figure 5 and Figure 6 we can see the square of the first condition. There has to be at least 5 non-black pixels in the area between the four red dots. We can see that the image in Figure 5 fulfills this requirement but that the image in Figure 6 does not, just as we want.

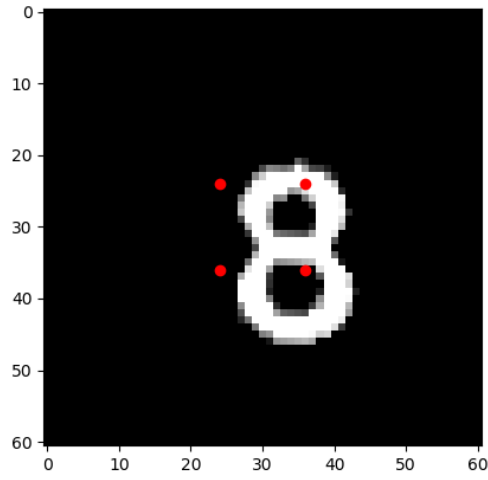


Figure 5: An image containing a number and that have at least fives non-black pixels within the square formed by the four red dots.

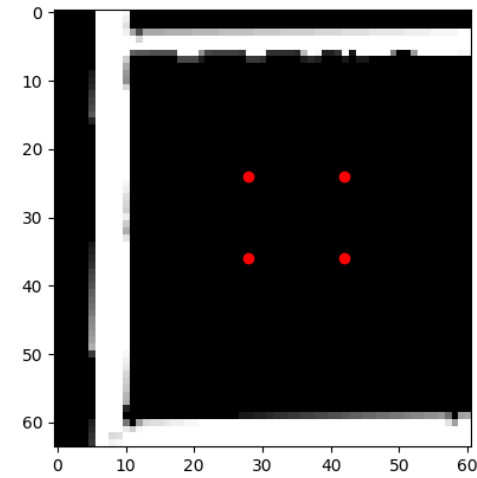


Figure 6: An image containing a number and that do not have five or more non-black pixels within the square formed by the four red dots and hence do not fulfill the first requirement.

## 5 Join all images of centered number and black images into a list

Each one of the 81 images get validated as described above. If the image contains a number given these conditions, the image is added to a list, if not, a black image with the same dimension is added to the list instead. This creates a list containing 81 clean images as shown in Figure 7.

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

Figure 7: 81 individual images. The images containing numbers are ready to be put into a model.