Title: **Sudoku Solver**

Name:	**Student Number:**
Zhangwen Yan	101040231
Hanbo Zhang	101014678

Summary:

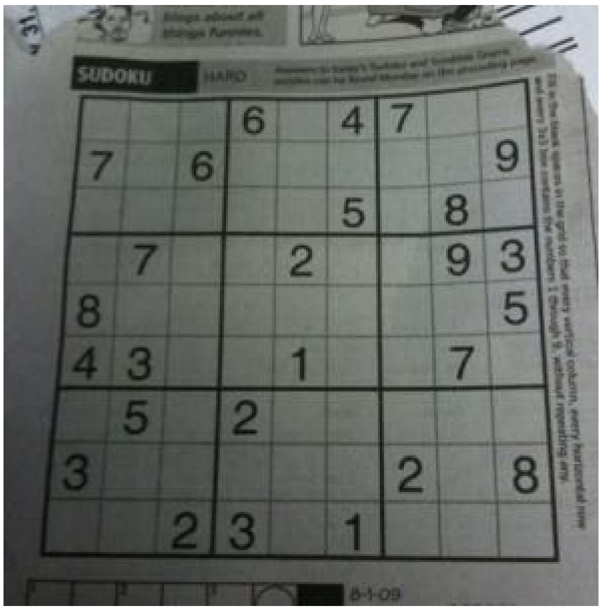
This project can quickly identify Sudoku games in photos (or video screenshots), convert the information on the picture into an array and automatically fill empty spaces. In our scenario, by using computer vision knowledge, we want to identify Sudoku photos that are not clear or having low resolutions or having bad lighting conditions or other non-Sudoku disturbances. We want to learn how to recognize shapes and numbers in images with this project. Identifying pictures with bad shooting conditions will be a challenge. In addition, interference items need to be excluded to ensure that Sudoku is correctly identified (not other squares in the same image). Another challenge is that we need to develop one or more sudoku solver algorithms that can solve different types of sudoku.The biggest challenge is processing pictures so that the computer can read the pictures and how to correctly identify the numbers in the picture.

Background:

Our idea comes from CHRIS(2009,07,19), How does it all work? IPHONE SUDOKU GRAB.from http://sudokugrab.blogspot.com/2009/07/how-does-it-all-work.html. This sudoku grab process only grabs normal shape of sudoku and solve it, what we are going to do in our project is trying to solve some different shapes of sudoku like Even Sudoku that places a digit from 1 to 9 into each of the empty squares so that each digit appears exactly once in each of the rows, columns and the nine outlined 3x3 regions. and there are some grey squares that must contain even digits. The solver will be developed on PC and the photos can be uploaded on PC from anywhere. We are looking for more shapes and types of sudoku to be solved and those sudoku can be recognized from newspaper or coloured magazines.
The members of our group are good at sudoku games and they have confidence in developing a sudoku solver algorithm.

Approach:

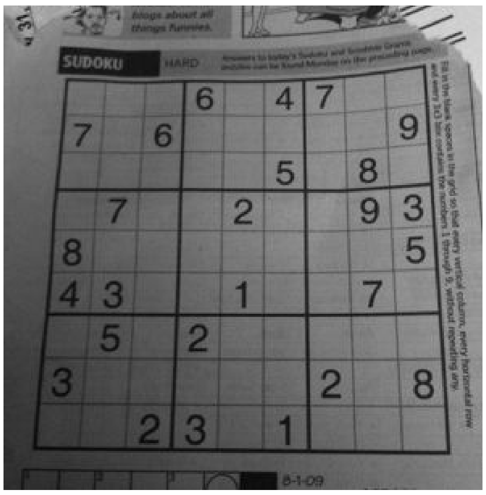
Sometimes, when people wait lines or drink coffee. they may read the magazine or newspapers nearby them and play the puzzle game on it, however when you need to go you may not finish the game and still want to know the answer. We decided to use python and opencv to solve those sudoku you took photos from anywhere like newspapers and magazines

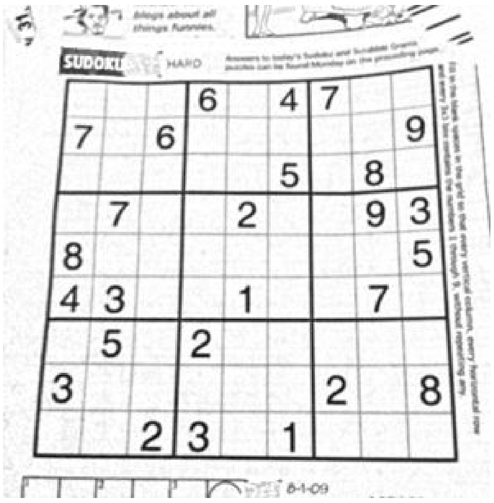


.

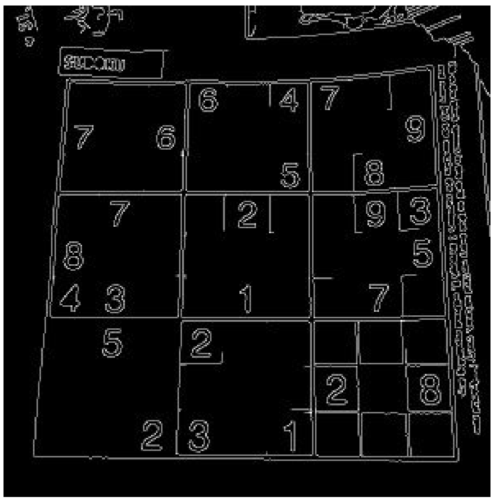
The first step is identifying the image. Because we can not do anything without a clear picture. So we need to use what we learn in class.

For identifying the image, we need to use a filter to turn the rgb image to gray image. By Gaussian filter which is a filter whose impulse response is Gaussian function and median filter, we can remove the noise. Also we thought that if the picture contains not only sudoku but also other images we don't need, we should cut them all, so we use a mask to blackout all other images and only keep the sudoku in the whole image.

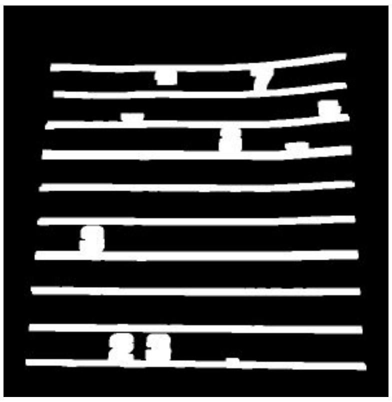


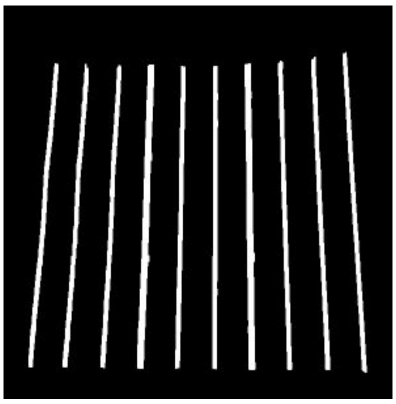


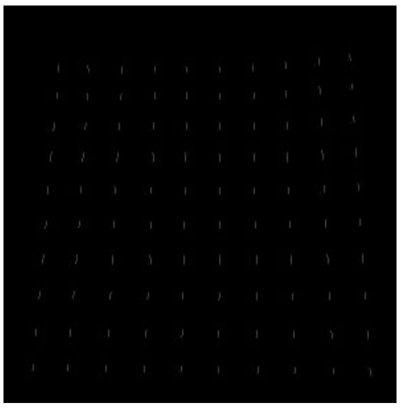
median filter   Gaussian filter

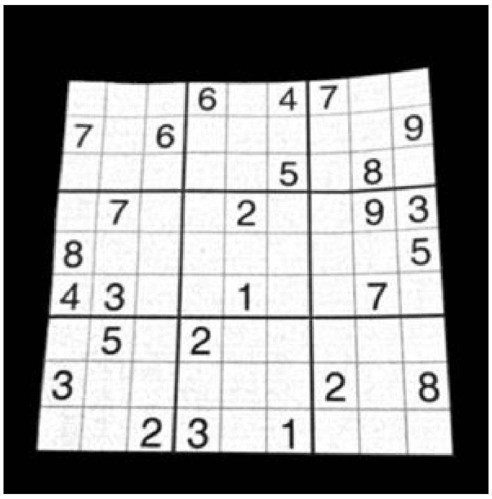
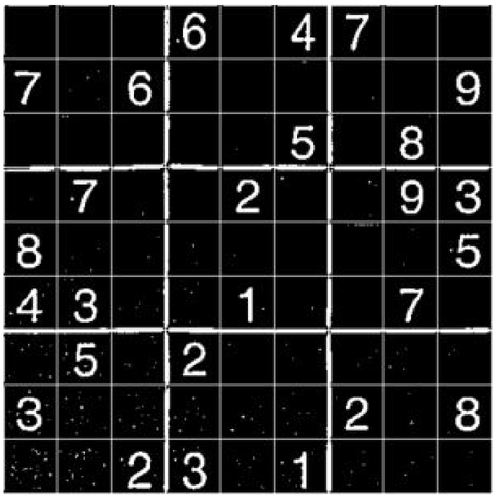

canny edge

After extracting the sudoku image and removing the noise, we use sobel to do image correction, use the sobel operator to extract vertical and horizontal lines. Use bitwise_and operation to get the intersection of horizontal and vertical lines. Perform perspective transformation and get the graph of sudoku. Also,binarization is a key to let row and column cross as a point to make the image clear. In the end, we convert the picture to a size of 540 * 540(A 9*9 squares that consisted of small squares with a side length of 60). At this point, we get a clear and standard Sudoku image.


sobel x


sobel y

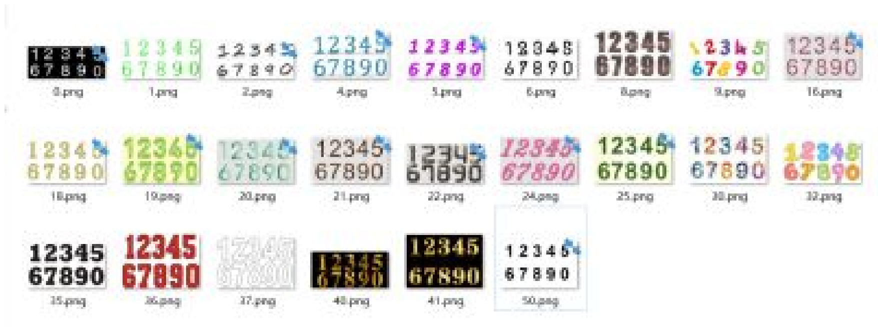

bitwise_and

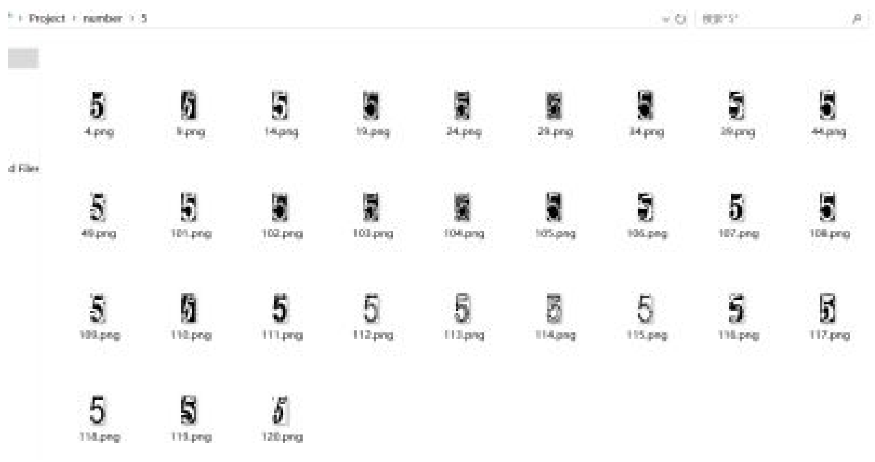
Mask   Final result

When we make the image clearly, what we are going to do is extract the numbers from the sudoku image, For a 540 x 540 image. We use 9x9 to cut them as 81 blocks that each one is 60x60 size. for a black-white image, if the white occupies 50% of the picture.we will judge that there are numbers in it and save their index.

After the process knows which block contains numbers, it needs to know which numbers are, so we search lots of numbers to build a database then use knn training to identify the numbers from 0 to 9. Finally we will have an array contain every number we got, if there is no number in it, it will be 0. Now, we convert the information in the picture into an array of sudoku



.
Training library



Traning result

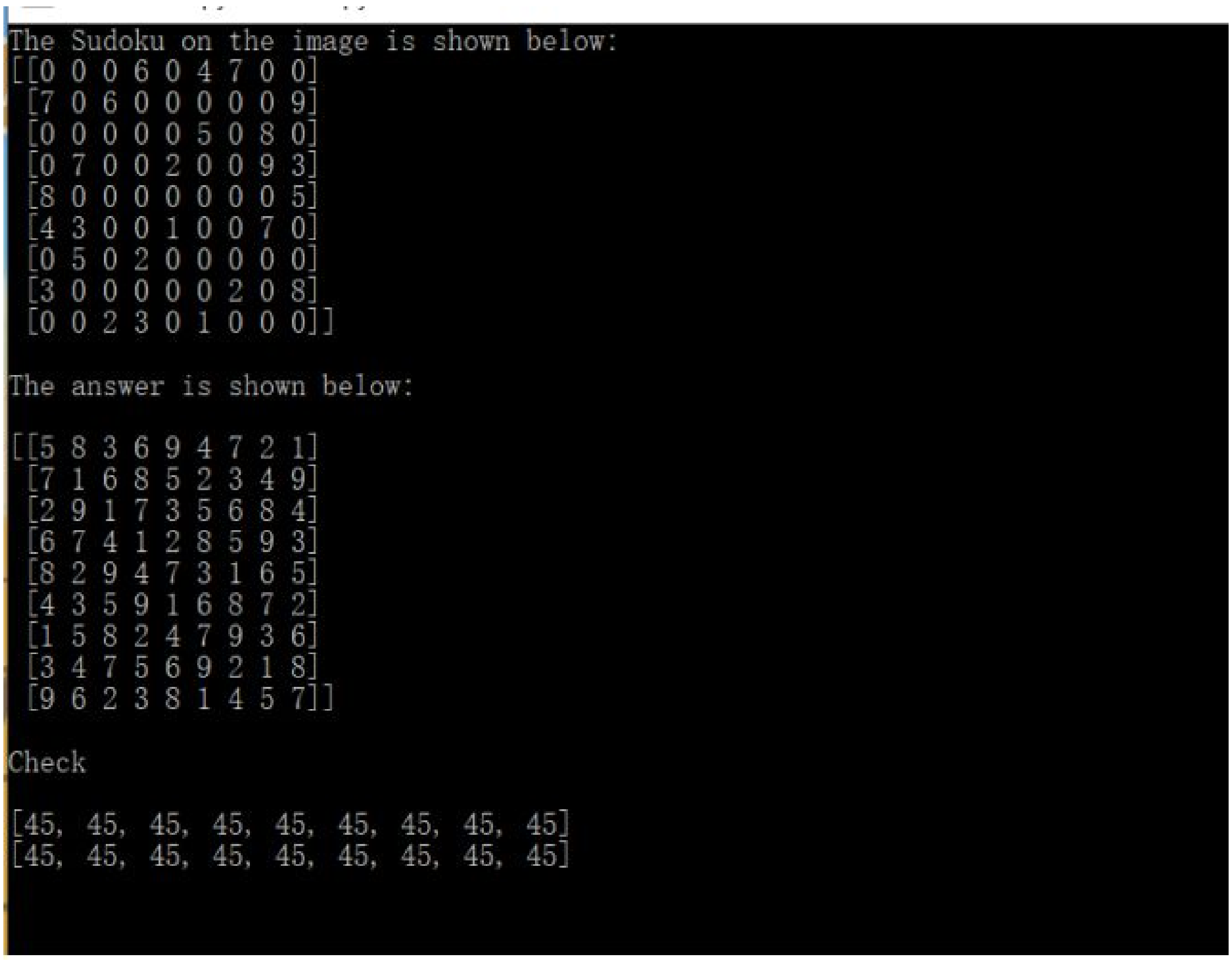
After all these things, we use the algorithm that we built to solve the sudoku. Also, check the answer by letting every number in row or column when they add all together with the result equal to 45. At the end, we output the results to users.

Results:

Sudoku is a square with numbers from 1 to 9 so that for each row and column, each number only appears once. So it is easy to test if the software is working: just check if there is a number that appears twice or more in the same row or column.

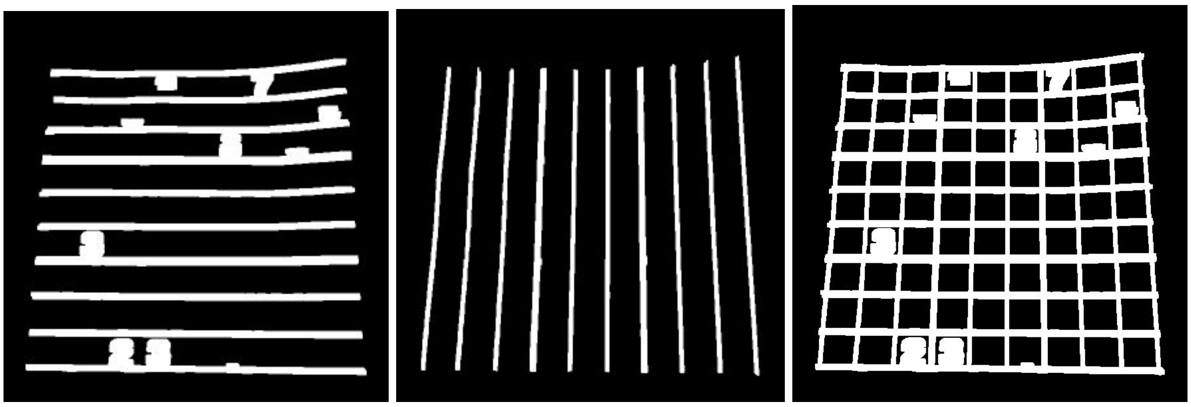
Our software will take a picture and recognize if there is sudoku on it and then identify the numbers on it, solving the sudoku is the last step. So before checking if the sudoku is solved, the user should check if there is a sudoku square on the picture, then check if the numbers on the sudoku solver are the same ones as the ones on the picture. Finally, add the numbers in each vertical column and horizontal row to see that it is equal to 45.

Here is the result, the first array is the sudoku from the original image, the second array is complete sudoku and the third one is a quick check that shows the sum of each column and row.



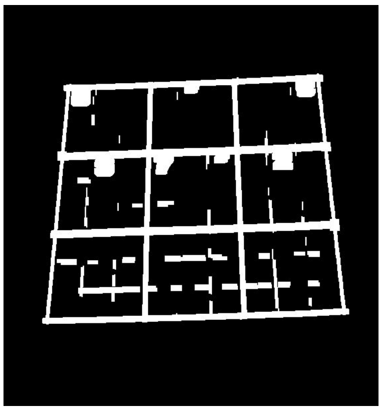
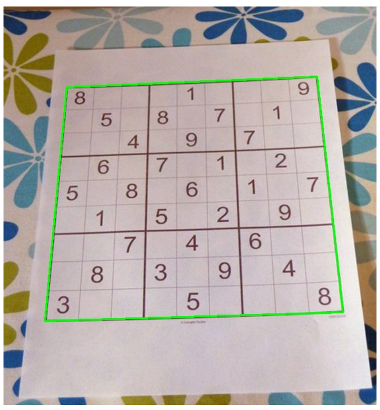
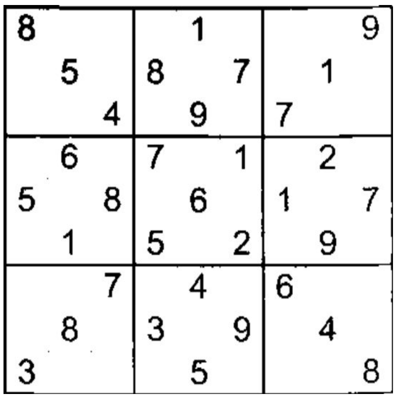
```
The Sudoku on the image is shown below:
[[0 0 0 6 0 4 7 0 0]
 [7 0 6 0 0 0 0 0 9]
 [0 0 0 0 0 5 0 8 0]
 [0 7 0 0 2 0 0 9 3]
 [8 0 0 0 0 0 0 0 5]
 [4 3 0 0 1 0 0 7 0]
 [0 5 0 2 0 0 0 0 0]
 [3 0 0 0 0 0 2 0 8]
 [0 0 2 3 0 1 0 0 0]]

The answer is shown below:

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

Check

[45, 45, 45, 45, 45, 45, 45, 45, 45]
[45, 45, 45, 45, 45, 45, 45, 45, 45]
```

In the process of processing pictures, we need to find vertical and horizontal lines to get their intersection to determine each small grid. Such as:



sobel x sobel y small grid

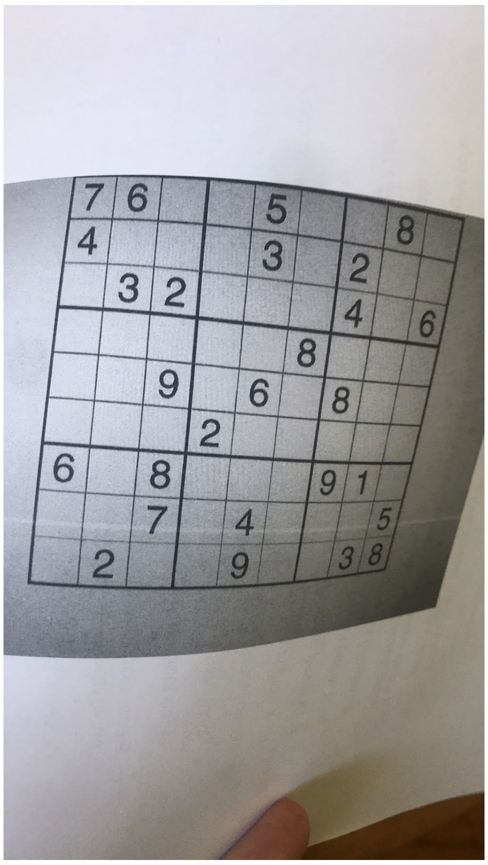
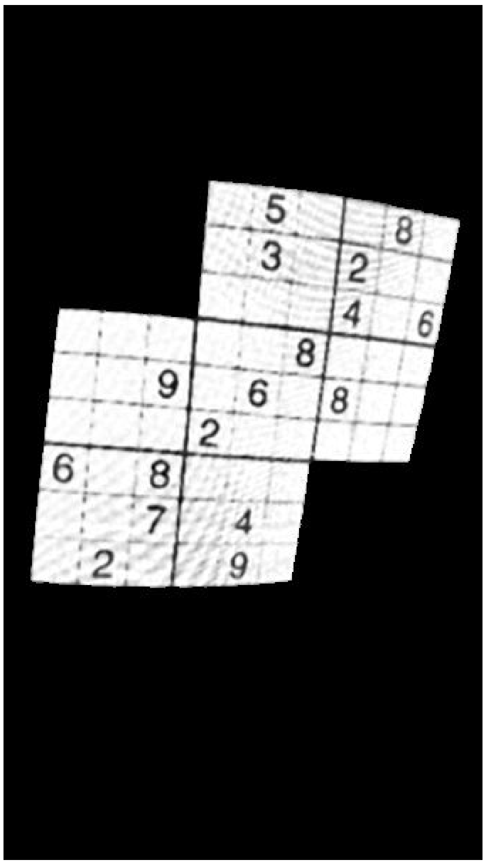
But for some pictures with inconsistent lighting conditions or pictures with insufficient grid lines will not be recognized well, so we use polygon approximation, polygon filtering to get the biggest square. Picture from another comp 4102 sudoku group: https://github.com/BHartford/COMP4102-Project

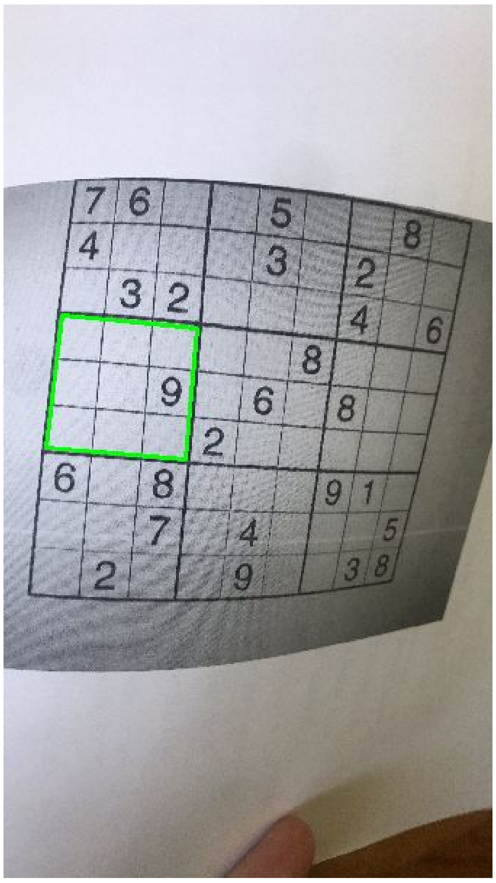




8				1				9
	5		8		7		1	
		4		9		7		
	6		7		1		2	
5		8		6		1		7
	1		5		2		9	
		7		4		6		
	8		3		9		4	
3				5				8

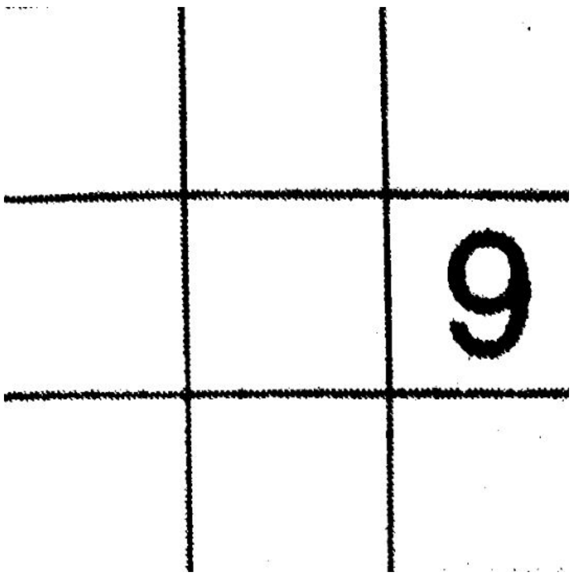
small grid Maximum border result

In another case, if the paper is bent more than a certain degree when shooting, the frame is not a straight line, or the outer frame is not fully in the camera. Our program cannot handle the pictures correctly, and no good solution has been found so far.



The frame is not straight enough with mask



find frame



result

In addition, due to the limitations of knn training and the size of the database is too small, it is not so perfect for number recognition. Handwritten digits are also not recognized, we next plan to use MNIST for better computer training (this is our first experience about computer learning).

List of Work:

Zhangwen Yan: Processing pictures, extract the picture with numbers and binarize the numbers on a picture, and extract every hierarchy of numbers, then we will use the Knn (K-Nearest Neighbor) algorithm to identify the numbers.

Hanbo Zhang: speech context, most of the paperwork and search and develop an efficient algorithm to solve the puzzle.

Github:
https://github.com/wengehuihuang/-4102Sudoku

References:
https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html
https://web.stanford.edu/class/cs315b/assignment1.html

https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123
https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm
https://docs.opencv.org/master/d4/d73/tutorial_py_contours_begin.html
https://stackoverflow.com/questions/10196198/how-to-remove-convexity-defects-in-a-sudoku-square
https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_geometric_transformations/py_geometric_transformations.html#geometric-transformations