University of Bucharest
Faculty of Mathematics and Computer Science
Department of Computer Science

Iuliana Georgescu
Bogdan Alexe
MSC in Computer Science
April 2021

Computer Vision - Project 1

Extracting visual information from Sudoku puzzles

Objective

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

Classic Sudoku

Sudoku is currently one of the most famous puzzles in the world. The most popular version consists on a 9×9 grid made up of 3×3 subgrids, but the general case, an $n^2 \times n^2$ grid with $n \times n$ subgrids is considered. Some cells contain numbers, which can be considered as input data. The goal is to fill in the empty cells, one number in each, so that each column, row, and subgrid contains the numbers 1 through 9 exactly once (numbers 1 to n^2 in the general case). If the input data are correct, the sudoku has one and only one solution. An example of a Classic Sudoku with its solution is given in Figure 1.

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

Figure 1: *A Classic Sudoku puzzle (left) with its coresponding solution (right).*

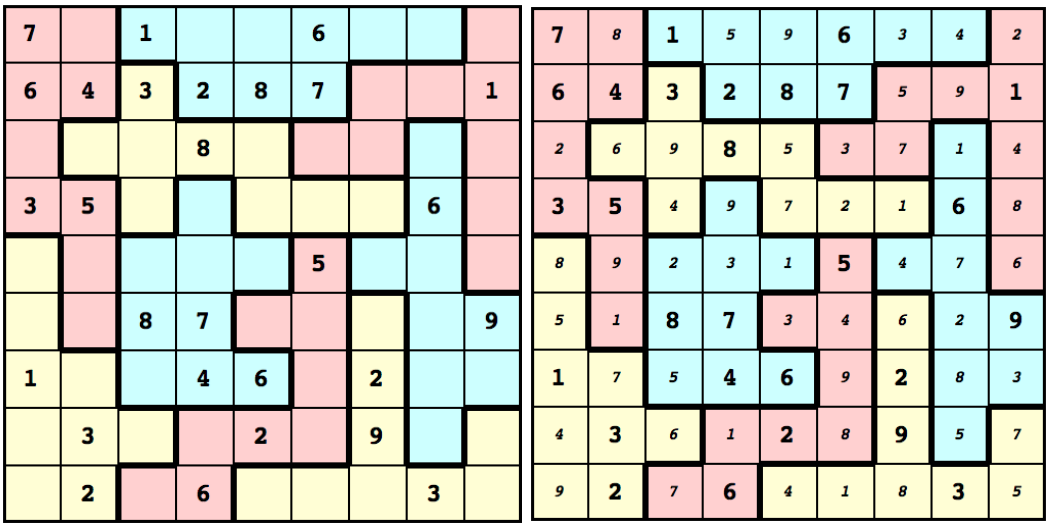


Figure 2: *A Jigsaw Sudoku puzzle (left) with its coresponding solution (right).*

Sudoku variants

Although the 9×9 grid with 3×3 regions is by far the most common, many other variations exist. In this project, we will also consider two such variations: Jigsaw Sudoku (Figure 2) and Sudoku Cube (Figure 3).

In Jigsaw Sudoku the goal is to fill in the empty cells, one number in each, so that each column, row, and irregular shape region contains the numbers 1 through 9 exactly once.

In Sudoku Cube, the goal is to fill in the empty cells of a cube (usually only three sides out of the six of the cube are shown) so that on all sides of the cube in every row, every column and every bold bordered region every digit from 1 through 9 appears exactly once. Additionally, the digit on the common edge of two sides must be the same number.

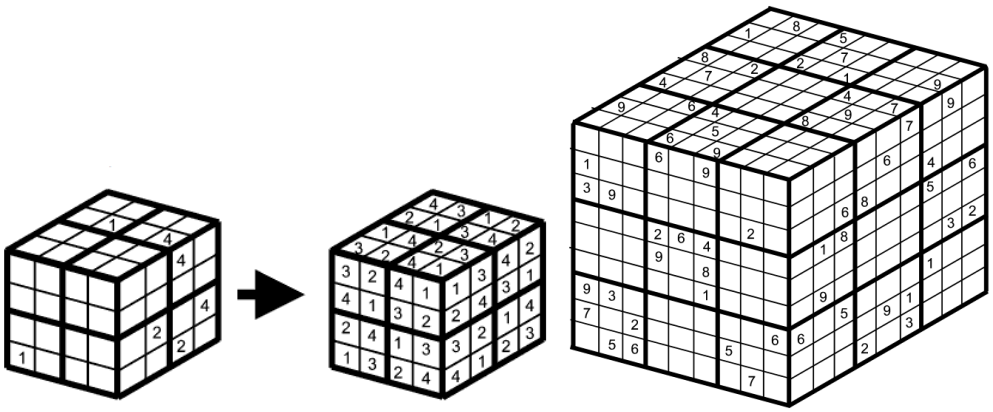


Figure 3: *Left: a Sudoku Cube puzzle for $n = 4$ digits and its coresponding solution. Right: a Sudoku Cube puzzle for $n = 9$ digits.*

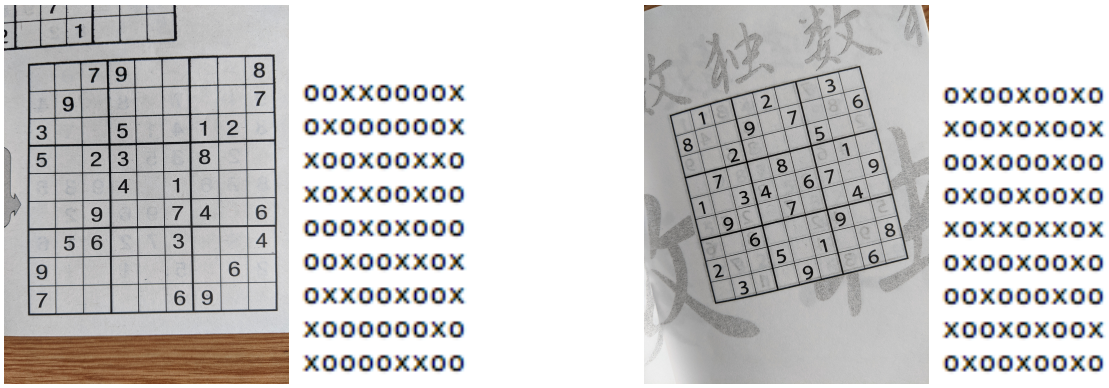


Figure 4: *Examples of two Classic Sudoku puzzles and their corresponding configuration.*

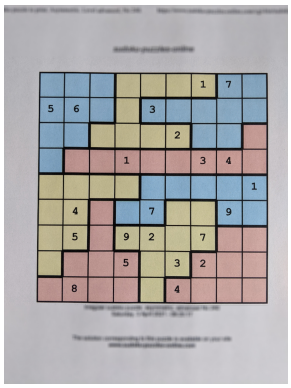
The goal of this project is to develop an automatic system for extracting visual information from images containing different variations of Sudoku puzzles. You can imagine the following scenario: you acquire images containing Sudoku puzzles using a mobile phone and then send the images to a server. On the server, your designed automatic system should process the images and return the extracted information. Based on the extracted information a separate system solving actually the Sudoku puzzle can be employed. In this project we are not concerned with actually solving the puzzle but only with extracting the visual information from images. For the first two tasks you will work with images that have the same resolution 4032 (height) $\times$ 3024 (width) pixels (they were acquired using Bogdan's mobile phone). For the last task you will work with syntetic images generated on computer that have resolution 1500 $\times$ 1500 pixels.

Task 1 - extracting configurations of Classic Sudoku puzzles

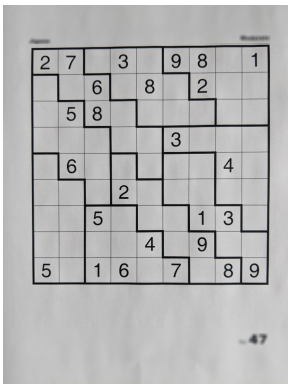
In the first task you are asked to write a program that processes an input image containing a Classic Sudoku puzzle and outputs the configuration of the puzzle by determining whether or not a cell contains a digit. We mark empty cells with letter 'o' and the filled in cells with letter 'x'. The training data consists of 50 training examples. Each training example (an image obtained by taking a photo with the mobile phone) contains one Classic Sudoku puzzle, centered, usually axis aligned or with small rotations with respect to the Ox and Oy axis. Figure 4 shows two training examples of Clasic Sudoku puzzles and their corresponding configurations.

Task 2 - extracting configurations of Jigsaw Sudoku puzzles

In the second task you are asked to write a program that processes an input image containing a Jigsaw Sudoku puzzle and outputs the configuration of the puzzle by: (1) determining the irregular shape regions in the puzzle; (2) determining whether or not a cell contains a digit. For this task, we mark all cells with a string of length two: the digit (1 to 9) corresponding to the irregular shape region where the cell is positioned and a letter ('o' or 'x') specifying whether or not the cell is empty. The irregular shape regions from the puzzle are separated by bold borders and sometimes (in the colored puzzles, see Figure 5) contain



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



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

Figure 5: *Examples of two Jigsaw Sudoku puzzles and their corresponding configuration.*

cells with the same color. For determining the digit corresponding to a cell in an irregular shape regions in a Jigsaw puzzle we use the following simple algorithm: (i) we process the cells from left to right and top to bottom; (ii) the top left cell gets digit 1 as it is part of region 1; (iii) we assign the same digit for all cells in the same region; (iv) the first cell in the next region gets the increased digit (we move to the next region). The training data consists of 40 training examples (20 colored and 20 black and white Jigsaw Sudoku puzzles). Each training example (an image obtained by taking a photo with the mobile phone) contains one Jigsaw Sudoku puzzle, either colored or black and white, centered, usually axis aligned or with small rotations with respect to the Ox and Oy axis. A colored Jigsaw Sudoku puzzle will always contain regions of three possible colors: blue, yellow and red. Figure 5 shows two training examples of Jigsaw Sudoku puzzle and their corresponding annotation.

Task 3 - assembling a Sudoku Cube

In the third task you are asked to 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

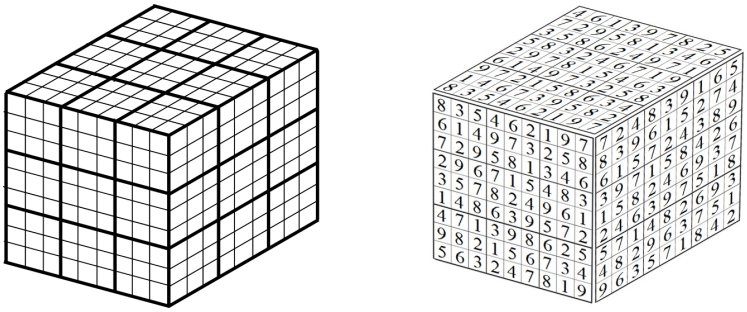
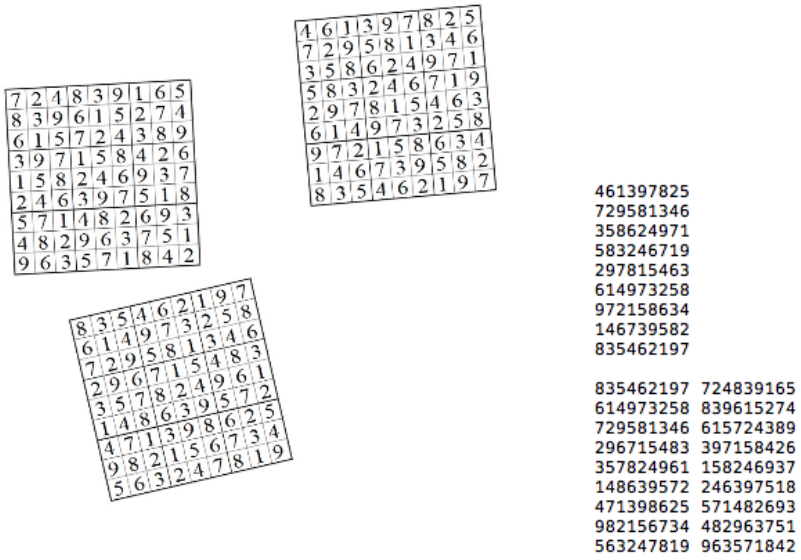


Figure 6: *Left: Sudoku Cube template. Right: The assembled Sudoku Cube base on the input image from Figure 7.*



```
461397825
729581346
358624971
583246719
297815463
614973258
972158634
146739582
835462197

835462197 724839165
614973258 839615274
729581346 615724389
296715483 397158426
357824961 158246937
148639572 246397518
471398625 571482693
982156734 482963751
563247819 963571842
```

Figure 7: *Left: An image containing three sides of a Sudoku Cube, each side is a sudoku puzzle. Right: The corresponding configuration after matching the sides. This correspond to the configuration in the assembled Sudoku Cube from Figure 6.*

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. The training data consists of 10 training examples. Each training example (an image 1500×1500 generated on the computer) contains three sides of a Sudoku Cube (see Figure 7). They are scattered around the image and usually rotated with respect to the axis. First, you have to find the three sudoku puzzles in the image, recognize the digits in each puzzle and solve the simple problem of matching the sides in the Sudoku cube. Please note that there is one solution. Then you have to warp the puzzles on the template in order to obtain the desired result. For warping, you are allowed to manually annotate points on the template for warping (but, of course, your are not allowed to to the same thing on the test images as we want your method to be automatically) such that it is easy to map each puzzle found in the image on the corresponding side of the Cube. Figure 7 shows the input image and the corresponding configuration (listed as side one, two and three; we consider side one of the Cube to be the side seen from above, side two of the Cube to be the side seen from the front and side three of the Cube to be the side seen from right). Figure 6 shows the Sudoku Cube template and the desired output after warping.

Data description

The release data directory (available here https://tinyurl.com/CV-2021-Project1) contains three directories: *train*, *test* and *evaluation*. The directories *train* and *test* have the same structure, although the *test* data will be made available after the deadline. The *train* directory contains data organized in three subdirectories as follows:

- *classic* - this directory contains 50 training examples with Classic Sudoku puzzles and their annotated configurations. Figure 4 shows two training examples of Clasic Sudoku puzzles and their corresponding configurations.
- *jigsaw* - this directory contains 40 training examples with Jigsaw Sudoku puzzles and their annotated configurations. Figure 5 shows two training examples of Jigsaw Sudoku puzzles and their corresponding annotation.
- *cube* - this directory contains 10 training examples with Sudoku Cube puzzles, their annotated configurations and the desired output. Figures 6 and 7 show the Sudoku Cube template, the input image containing the three sides, the corresponding configurations after matching and the desired output.

The directory *evaluation* shows how the evaluation will take place on the test data after the deadline. It contains the following subdirectories:

- *fake_test* - this directory exemplifies how the test data will be released, keeping the structure of the previously described *train* directory;
- *submission_files* - this directory exemplifies the format of the results data that we expect from you to submit in the second stage. You will have to send your results in this format, uploading a zip archive of a folder similar with the one called *Alexe_Bogdan_407*;
- *evaluation* - this directory contains code that we will use to evaluate your results using the ground-truth data. Make sure that his code will run on your submitted files. The ground-truth data will be released after you send us your results.

Requirements

Your job is to write a program in Python that automatically solves Task 1, Task 2 and Task 3. This project worth 5 points but you can gain a bonus for a total of 6 points. We will grade your project based on the performance achieved by your algorithms on each of the three tasks. The grading scheme is as follows:

- **Task 1** - you receive a test set containing 50 testing examples with Classic Sudoku puzzles. For each test image you have to output the corresponding configuration. Each correctly labeled configuration will worth 0.05 points for a total of **2.5 points**;

- **Task 2** - you receive a test set containing 40 testing examples with Jigsaw Sudoku puzzles (20 colored and 20 black and white). For each test image you have to output the corresponding configuration. Each correctly labeled configuration will worth 0.05 points for a total of **2 points**;
- **Task 3** - you receive a test set containing 10 testing examples with Sudoku Cube puzzles. For each test image you have to output the corresponding configuration (see Figure 7right) and the desired output (see Figure 6right). Each correctly labeled configuration will worth 0.05 points and each correct warped image will worth 0.05 points for a total of **1 point**;
- ex officio **0.5 points**;

Deadlines: Submit a zip archive containing your code and a pdf file describing your approach until Saturday, 8^{th} of May using the following link https://tinyurl.com/CV-2021-PROJECT1-SUBMISSIONS. Notice that this is a hard deadline, no projects will be accepted after the deadline. Your code should include a README file (see the example in the materials for this project) containing the following information: (i) the libraries required to run the project including the full version of each library; (ii) indications of how to run the solution for each task and where to look for the output file. Students are allowed to submit solution in the format of Jupyter Notebbok files with the code being commented. This can replace the pdf file describing their approach. Students who do not describe their approach (using comments thoughout the code or a pdf file) will incur a penalty of 0.5 points.

On Sunday 9^{th} of May we will make available the test data. You will have to run your system on the test images provided by us and upload your results in the same day as a zip archive using the following link https://tinyurl.com/CV-2021-PROJECT1-RESULTS.