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

Alexandra Diaconu
Bogdan Alexe
MSC in Computer Science
April 2023

Computer Vision - Project 1

Mathable score calculator

Objective

The goal of this project is to develop an automatic system to count the score for the game Mathable.

Mathable

Mathable is a game based upon mathematical equations which must be formed on the playing board. It has been described as being like playing Scrabble but using numbers (Figure 1). In Mathable, the players make use of a playing board with normal squares (white squares), squares imposing some constraints (blue squares) and squares containing an award (purple squares marked 2× and orange squares marked 3×), as well as 106 tiles with numbered tokens. The game can be played by two, three of four players with the goal of forming valid mathematical equations and scoring as many points as possible.

Tokens

In completing mathematical equations on the board, players use tiles with numbered tokens. For simplicity, we refer to them as simply *tokens*. There are in total 46 different types of tokens, which usually can help in completing a mathematical equation. The tokens are:

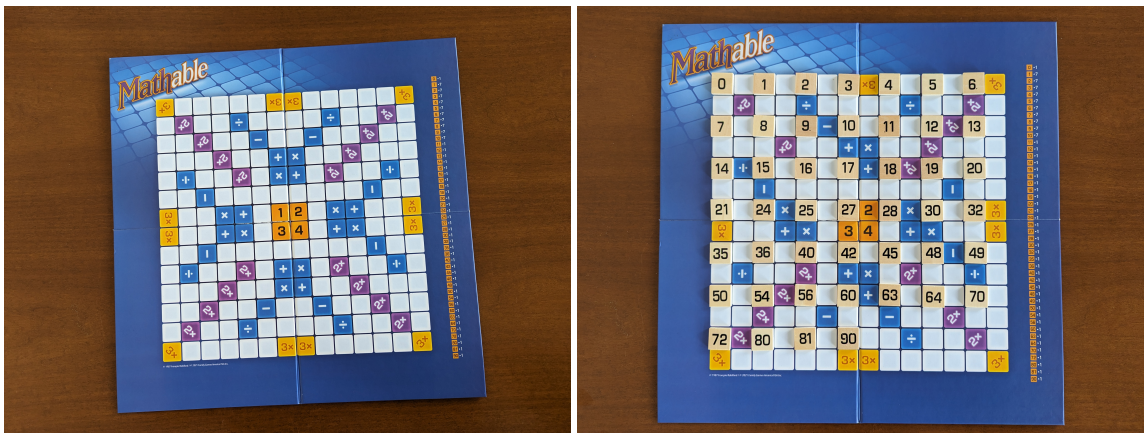

Figure 1: *The Mathable board (left) and all possible tokens placed on the board (right).*

- digits 0-9 (for a total of 10 tokens in interval [0,9]);
- numbers 10-19 (for a total of 10 tokens in interval [10,19]);
- numbers 20, 21, 24, 25, 27, 28 (for a total of 6 tokens in interval [20,29]);
- numbers 30, 32, 35, 36 (for a total of 4 tokens in interval [30,39]);
- numbers 40, 42, 45, 48, 49 (for a total of 5 tokens in interval [40,49]);
- numbers 50, 54, 56 (for a total of 3 tokens in interval [50,59]);
- numbers 60, 63, 64 (for a total of 3 tokens in interval [60,69]);
- numbers 70, 72 (for a total of 2 tokens in interval [70,79]);
- numbers 80, 81 (for a total of 2 tokens in interval [80,89]);
- number 90 (for a total of 1 token in interval [90,99]).

The specific number of tokens available in the game is specified on the right edge of the board. For each value between 1 and 10 there are 7 tokens in the game (for a total of 70 tokens) while for the rest of 36 values there is a single token with that value (for a total of 36 tokens). So, in total, there are 106 tokens in the game.

Board

The Mathable playing board (Figure 1) is divided into a 14×14 grid of squares. The board is marked with coloured squares, some of them containing an award and thus increasing the score and some of the imposing constraints. There are in total 28 coloured squares with an award:

- 16 purple squares for double points;
- 12 orange squares for triple points.

The constraint squares are marked with blue and contain an addition, subtraction, multiplication or division sign. In order to occupy a specific square by placing a token there, the player must make an equation that corresponds to the sign of that square. There are in total 32 squares with constraints.

Valid and invalid configurations

A mathematical equation is completed by adding, subtracting, multiplying or dividing two adjacent numbers putting a token with the result on the next empty square, be in to the right or left, below or above the two original tokens, never diagonally or between tokens. Figure 2 shows valid and invalid configurations on a general Mathable board. The first player may choose one of the four mathematical operations using the numbers 1 to 4 positioned in the center of the playing board.

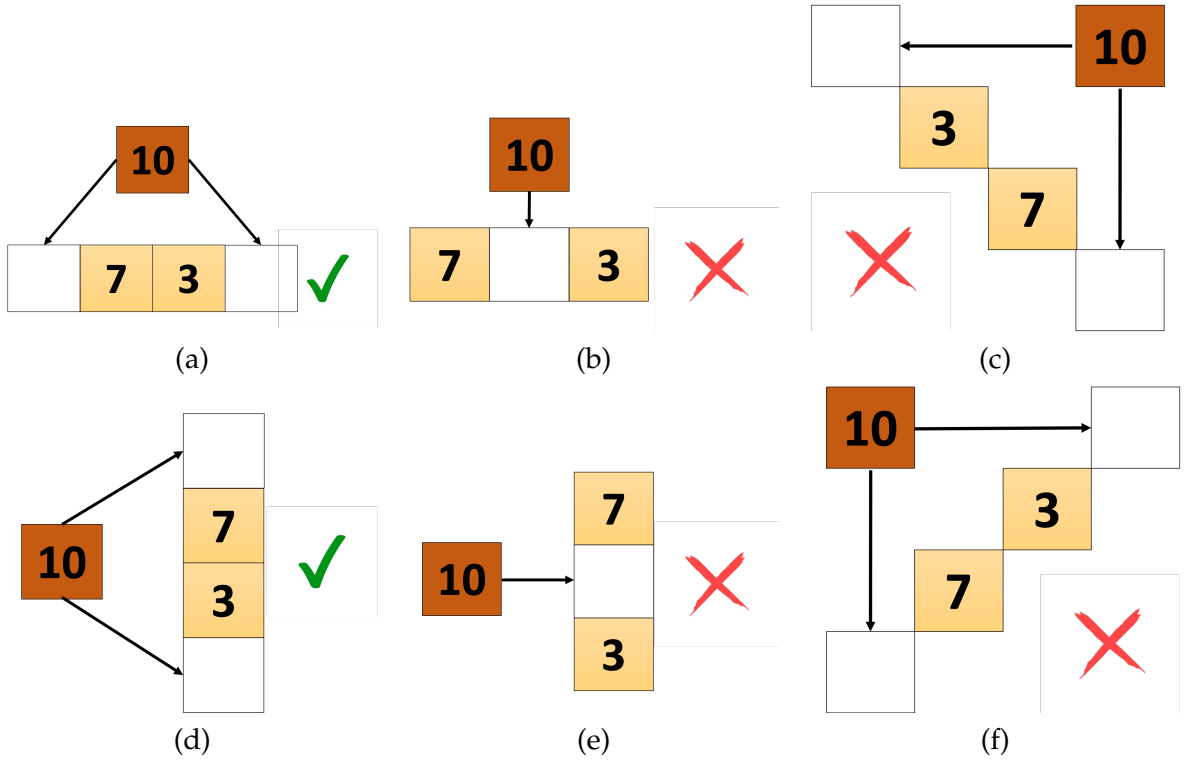

Figure 2: **Valid and invalid configurations in Mathable.** *Examples in (a) and (d) illustrate valid moves, where the token* 10 *can be placed either in the right or in the left (case (a)) or in below or above (case (d)) the two original tokens* 7 *and* 3 *completing the addition. Examples in (b) and (e) show invalid moves, where the token* 10 *cannot be placed horizontally between tokens (case (b)) or vertically between tokens (case (e)). Examples in (c) and (f) show invalid moves, where the token* 10 *cannot be placed diagonally wrt tokens* 7 *and* 3*.*

The play

We consider the scenario with only two players, Player 1 and Player 2. Each player receives at the beginning of the game 7 tokens drawn from a bag with all tokens (initially there are 106 tokens in the bag). At each round, the current player can make several moves by completing mathematical equations on the board. After each move, the current player gets a score. The maximum number of moves per round is limited to 7. At the end of his turn, each player draws tokens randomly from the bag, to bring the amount of his holder back to 7. The score obtained by the current player in the round is obtained by adding the scores obtained at each move.

Scoring

The score of the current player after each move is based on the token placed on the board and can increase in some cases.

Score values of a token. Each token has a score value equal to the number on the token.

Bonus for multiple equations. If a token, when being placed, completes more than one equation, the points are gained for each equation completed.

Squared with an award. A purple square marked 2× double the amount of points of the token on that square, an orange square marked 3× triples the amount of points.

Scoring example

Figure 3 shows twelve moves made at the beginning of a game. We list below the twelve moves and offer detailed explanations about computing the corresponding score of each player after each move. The two players in this example have the following tokens:
Player 1: 1, 2, 8, 12, 16, 17, 42;
Player 2: 3, 4, 6, 7, 7, 8, 21.

Move 1. Player 1 may choose one of the four mathematical operations using the numbers 1 to 4 positioned in the center of the playing board. Player 1, who has a token 12 on his holder, chooses to multiply the numbers 3 and 4. He may place token 12 either to the left of the 3 or to the right of the 4. He chooses to place it at the right. Since the equation is correct, the player earns 12 points.

Move 2. Player 1 places his second token. Since he has an 8, he decides to multiply 2 and 4 and chooses to place the result above the 2. An alternative move would have been to position the token 8 bellow the 4. Thus, player 1 earns another 8 points.

Move 3. Player 1 places his third token. He takes the token 2 from the holder, subtracts 1 from 3 and places the 2 below the 3. He could also place the 2 above the 1, but chooses not to do so. The player adds 2 points to his total.

Move 4. Player 1 places his fourth token. He subtracts 1 from 2 and places token 1 left of the 1 already on the board. He could also place it at the right side. Thus, player 1 earns another 1 point.

Move 5. Player 1 places his fifth token. At this point, the square to the right of the square occupied by token 12 is blue, marked with a + sign, meaning that the player has to add the two tokens if he wishes to occupy this square. The player takes token 16 from his holder and places it on the blue square, thus fulfilling the constraint. The player adds 16 points to his score. The player cannot place another token, and so his turn is over. He totals all points earned in this turn, and finds his total score is 39.

Move 6. Second player's turn. Player 2 decides to add 3 and 4, since he has a 7 on his holder. He places the token to the left of the 3. He receives 7 points.

Move 7. Player 2 places his second token. Subtracting 1 from 7 gives him a total of 6. He

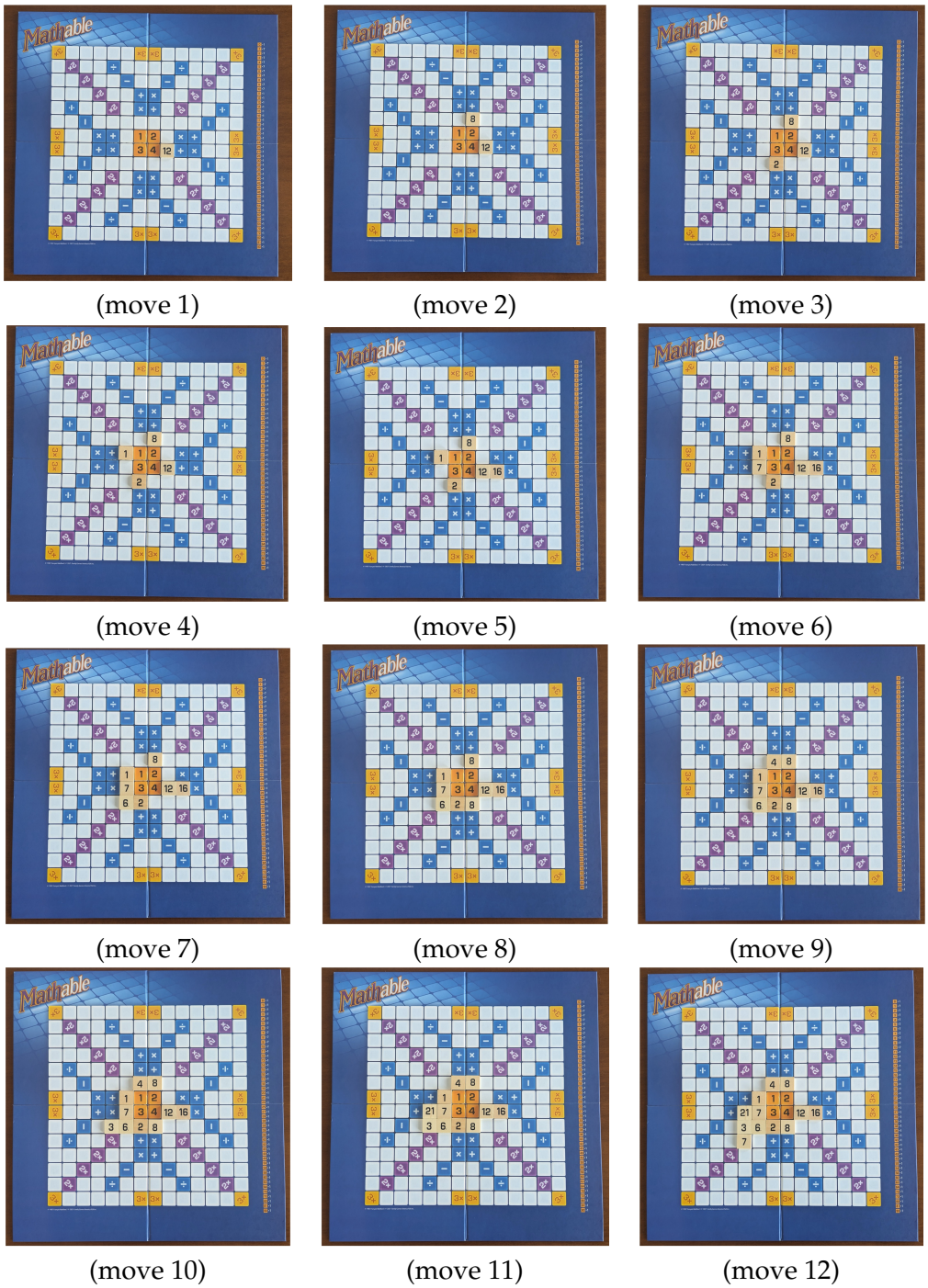

Figure 3: *We illustrate twelve moves made at the beginning of a game, starting from the empty board. We describe in detail in the main text the computation of the corresponding scores to each of the two players after each move.*

places the token 6 under the 7 (he could also have placed it above the 1). He adds 6 points to his total.

Move 8. Player 2 places his third token. The player decides to add 6 and 2, totalling 8. He has two possibilities: Placing the 8 to the left of the 6, giving him 8 points or placing the 8 to the right of the 2, and thus profiting from the Bonus for Multiple Equations, since the vertical 2×4 also results in 8. The player opts for the second choice, adding another 8 points (a total of 16) to his score. REMARK: If the two tokens above this square do not complete the equation, the player may still place a token there, through only counting 8 points for the addition.

Move 9. Player 2 places his fourth token. This time he uses token 4, adding 3 and 1. He places the token above the 1. He earns 4 points.

Move 10. Player 2 places his fifth token. To make use of token 3 he divides 6 by 2. The result 3 is placed next to the 6. He adds 3 points.

Move 11. Player 2 places his sixth token. Multiplying 7 by 3 totals 21. This token is placed to the left of the 7. He adds 21 points. Please also note that, again, the current player fulfills the constraint of the blue square on which he has placed his token.

Move 12. Player 2 places his seventh token. He places the 7 below the 3 since 21 divided by 3 is 7. He cannot place the token above the 21, since a constraint square does not allow him to divide. Placing the 7 below the 3, he lands on an award square (2×) which doubles the amount of points earned in this move to $14 = 7 \times 2$. The total points in his turn is thus 71.

Data description

The release data directory (available here https://tinyurl.com/CV-2024-Project1) contains four directories: *board+tokens*, *train*, *test* and *evaluation*. All directories contain images taken with Bogdan's phone in different scenarios. The directory *board+tokens* contains several images with the empty board from different viewpoints and also with the board on which we placed all possible 46 tokens (from 0-21, 24, 25, 27, 28, 30, 32, 35, 36, 40, 42, 45, 48, 49, 50, 54, 56, 60, 63, 64, 70, 72, 80, 81, 90). You can use these images to better understand the problem and to extract data for your solution. The directories *train* and *test* have the same structure, although the *test* data will be made available after the deadline. In total there are 200 images and 200 annotations files. For each game, there is a .txt file that annotates the beginning of a turn for a player (*g_turns.txt* where *g* is the game number). The .txt file *g_scores.txt*, where *g* is the game number, contains the scores of each player after each round, expanding thus the file *g_turns.txt*. The training image *i* corresponding to game *g* is denoted by the file 'g_i.jpg', where $g \in \{1, 2, 3, 4\}$ and $i \in \{01, 02, 03, ..., 50\}$. The corresponding annotation file has the extension '.jpg' replaced by '.txt'.

The first game consists of images with regular views, the images are taken with the phone

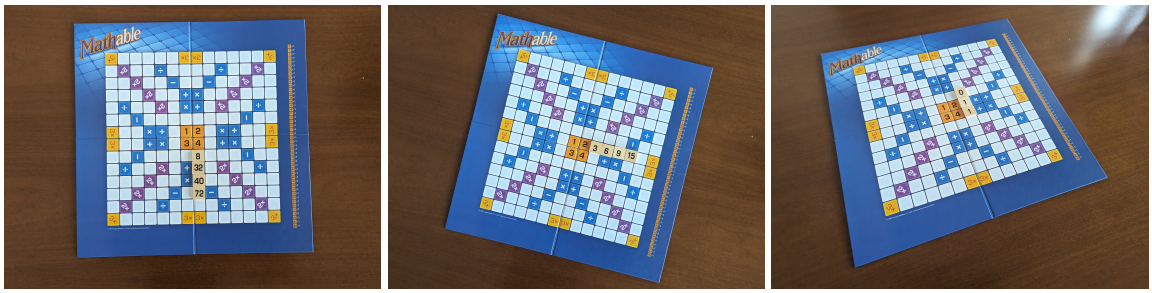

Figure 4: *Different views of the board: regular view (left), rotated view (centre) and perspective view (right).*

placed parallel above the board, with minor scale changes and rotations. The second game consists of images with rotated views, the images are taken with the phone placed parallel above the board, but at some non negligible rotation with respect to the board. The third game consists of images with perspective views, the images are taken with the phone tilted with respect to the board. The fourth game consists of images with mixed views, either regular, rotated or perspective. Figure 4 illustrates these scenarios.

The ground-truth annotations files contain for each move the following information:

- the position of the token added to the board, taken in the order from left to right and from top to down. We specify the position using numbers 1-14 for rows and letters A-N for columns. Take into consideration that only one token is placed at each move.
- the number on the token at the corresponding position on the board.

Figure 5 exemplifies a ground-truth annotation file for image 1_01.jpg

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. Notice that we include here only 1 game = 50 images. The test data will contain 4 games = 200 images.
- *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*. Notice that the current archive correspond to the data released in the *fake_test* directory. For the test stage your archive should contain around 200 files.
- *code* - this directory contains code that we will use to evaluate your results using the ground-truth data. Make sure that this code will run on your submitted .txt files. The ground-truth data will be released after you send us your results.

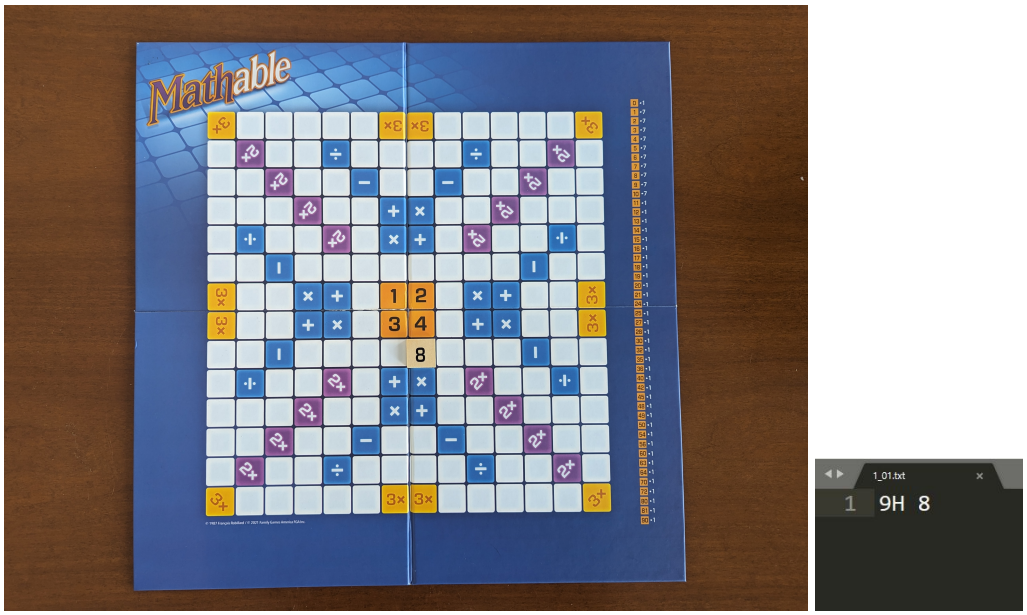

Figure 5: *For the current move, the token 8 is placed on the board. The ground-truth annotation file specifies the positions of newly added token and the token.*

Requirements

Your job is to write a program in Python/Jupyter notebook that automatically solves the task of extracting information of the current move depicted in a test image. For each test image you have to output the corresponding information similar to the annotation files, thus specifying the position of the newly added token to the board and the token. Additionally, you have to provide a similar .txt file of the form (*g_scores.txt*, where *g* is the test game number, containing the scores of each player after each round, expanding thus the file *g_turns.txt*.

This project worths 5.5 points. We will grade your project based on the performance achieved by your algorithms on each of 200 test images.

You will receive a test set containing 200 testing images organized in 4 games of 50 moves. The distribution of images in the test data follows the distribution of train data, meaning that the images were acquired in the same conditions and also that the first game will contain images with regular views, the second game will contain images with rotated views, the third game will contain images with perspective view, the fourth game will contain images with mixed views. For each test image you have to output a .txt file containing information similar to the annotation files and for every game a .txt file with scores after each round. Each correctly solved test image worths 0.0175 points. For correctly specifying the position of the added token you receive 0.01 points per image (move), for correctly specifying the token placed at the corresponding position on the board you receive 0.0075 points per image (move) and for correctly specifying the total score of the current player after a turn you receive 0.02 point per turn. You receive 0.5

points from *ex officio* conditioned on the fact that you respect the format of the submitted results, such that our evaluation script works smoothly on your provided results.

The oral presentation of this project (face-to-face or online) will be scheduled in the weeks 14 – 17 or 20 – 24 of May. It will take around 10-15 minutes in which Alexandra or Bogdan will ask question regarding implementation. The oral presentation will count for 0.5 points and is mandatory for each student submitting his solution for this project.

Deadlines

Submit a *zip archive* containing your code (python files or Jupyter notebook files), all auxiliary data that you are using (templates, models, etc.) and a pdf file describing your approach until Tuesday, 14^{th} of May using the following link https://tinyurl.com/CV-2024-PROJECT1-SUBMISSIONS. Please do not include in your zip archive any unuseful data (like training images, we already have them!!!). 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 and where to look for the output file. Students who do not describe their approach (using a pdf file) will incur a penalty of 0.5 points.

On Wednesday 15^{th} of May we will make available the test data. You will have to run your solution 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-2024-PROJECT1-RESULTS.