Computer Vision - Project 1 - Double Double Dominoes

Mihail Feraru - 407

May 2023

1 Introduction

In this paper we will present the failed and successful approaches in solving the task at hand: automatically scoring a game of Double Double Dominoes. It also includes details about the bonus task, which is roughly the same problem from a computer vision perspective.

The following sections will describe each major step performed by the developed application to extract information from the dominoes board and solve the task. For a given dominoes board photo, the major steps performed are:

- crop the inner board (the green one) from a given image;
- split the board in a grid of patches (as in laboratory 2);
- detect circles in each patch roughly the same size as a dot on a domino piece;
- detect the median lines on each domino piece, their orientation and position;
- count the number of circles on each domino piece, thus completing the extraction.

After extracting information from each board photo, solving the task is quite rudimentary:

- computing the score requires iterating two boards and finding their different cells;
- detecting invalid placing of 2x2 domino squares requires a sliding window traversal over the matrix of a board;
- detecting invalid neighboring dominoes requires checking the orientation and two neighbours for each cell of the matrix.

2 Extracting board information from images

All methods below were tested for all the 150 images provided in the train set, so we can increase the confidence of their generality. The images were processed in grayscale format and at their full scale. The high resolution has an impact on the speed of the running application, but it assures us of the best results in terms of accuracy.

2.1 Inner board cropping using ORB

Firtly, we tried to extract the inner board using image filtering (eg. Gaussian blur, binary thresholds and morphological operations like erosion), Canny edge detection and finding the largest rectangular contours on the board. This approach proved to be tedious and prone to errors, so we abandoned it early.

Secondly, we used the best image of an empty board to create a nice top-down view template of the inner green grid. To accommodate for the slight inaccuracies of the photo (the board is not a perfect square because of the perspective), we applied a perspective transform using OpenCV, and the result can be observed in Figure 1.

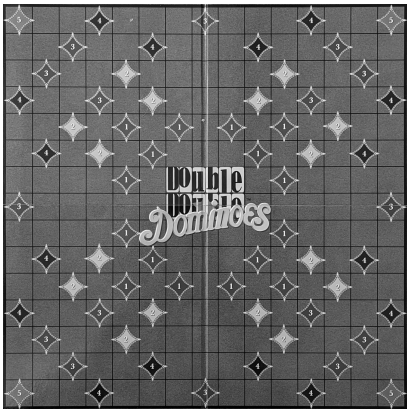

Figure 1: Template image of the inner dominoes board, shaped as perfect square using a perspective transform from OpenCV.

Finally, we tried to extract the board using feature matching with SIFT and FLANN on one hand, and with ORB and brute-forcing on the other. The SIFT approach showed to be slow and sometimes it was extracting a distorted board, so we stick with an ORB with 2000 features and a retention rate for matches of 15%.

Because the extracted image seemed to align almost perfectly with the template, we omitted any alignment step afterwards.

2.2 Failed approaches to detect dominoes

The first idea was to rely on the mean or dominant color in a given square in order to detect if it is a domino or not. It quickly became clear that the letters in the middle, the diamonds and variable number of dots on a domino are invalidating the approach.

Afterwards, we tried to detect all the circles of roughly the same radius as a dot on a domino in the entire image, then use this information together with the mean color to detect dominoes. Again, a lot of false positives were generated by the letters in the middle and the diamonds spread across the board.

The last failed approach to detect dominoes or their dots was to template match them. Template matching dots was unsuccessful, because dots lack feature complexity. Template matching entire dominoes was slow and required a lot of templates, which seemed unfeasible.

2.3 Multi-step domino detection

First, we generate the grid of patches over the extracted board in a procedural way by considering the margins of the board, the thickness of the lines of the grid and the number of rows and columns.

We apply a series of filters on the entire image (Gaussian blur, Otsu's thresholding, and morphological dilation using a kernel of 5x5) and for each patch in the grid, we apply Hough Circle Transform to obtain all the circles similar to the dots on a domino. The filtered image before the detection can be seen in Figure 2.

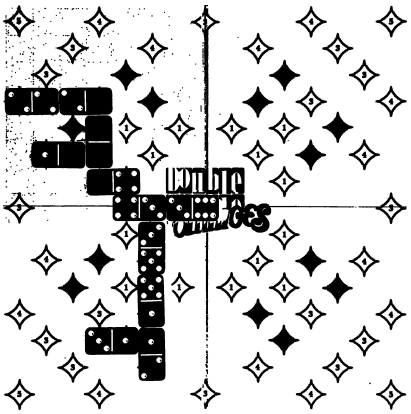

Figure 2: Board image with Gaussian blur, Otsu's thresholding inverted, and a dilation of 5x5 kernel. It is used to detect circles and contours.

After the Hough Circle Transform, we have detected a lot of false positives, but at this stage we are not going to bother with them.

After detecting the circles, we will use the same filtered board image to detect all rectangular contours which are very wide or tall and with an appropriate area. This way, we obtain all the contours which could be a median black line on a domino piece, but there are a lot of false positives too.

To eliminate the false positives among the median lines of dominoes pieces, we get back the the filtered image and draw on it with hard black all the detected circles in the previous steps. The result can be observed in Figure 3.

After filling all the circles on the image with black, we have all the domino pieces as solid black shapes with a slim white line in the middle. Now, we can filter all the false positive lines at the previous step by checking if they are surrounded by solid black rectangles.

Let's recap, we have a list of circles which are similar to the dots of domino pieces, we have a list of contours which are probably median lines of domino pieces and a nicely aligned grid on the board.

Now, we create two matrices to store vertical and horizontal detected median lines. We compute their position in those matrices and store them. We iterate over each of those matrices and remove any lines that are too close, for example, if we detected two lines closer than the width of a full domino piece, we are sure that one of them is a false positive.

After eliminating the false positives using the rule above, we take every median line detected and as we know that it is surrounded by two halves of a domino piece (up and down or left and right), we count the detected circles which are contained in each half, thus we find the value of each domino.

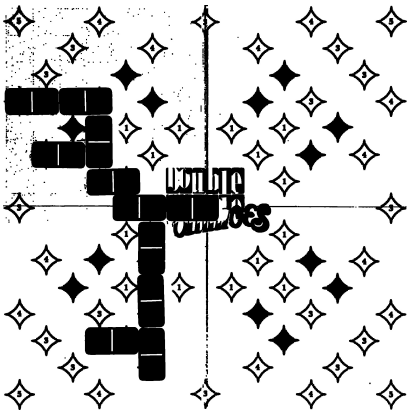

Figure 3: The same board as in Figure 2, but with all the circles filled with black. This is helpful for detecting median lines on domino pieces.

Based on the position of each median line, we can compute the position of each domino on the board (rows 1 to 15, columns A to O). In Figure 4 we see all the detected elements. Note that the false positive circles do not disturb our counting, because we counted only the circles near a median domino line.

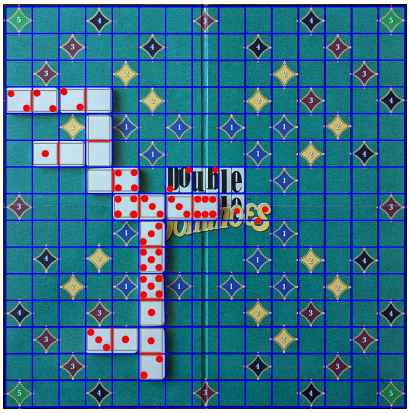

Figure 4: Example board after detecting all elements: dots, lines of the dominoes, and the grid of the board.

3 Solving the game

After extracting each board as presented in the previous step, we create a list of all boards associated with a game (the empty board being the first board of each game). We iterate among as transitioning through rounds, so we compare the current and the next state of the game. For each, only two cells in the matrix will be different (i.e. the placed domino piece). We update the score and output the newly placed domino as required.

To detect invalid squares of dominoes, we take each board and slide a window of 2x2 over it. If we find a window filled with values, we output the positions of those cells as being invalid.

To detect invalid touches of dominoes, we iterate over each cell on the board. If two neighboring cells are not empty, have different values and are not part of the same domino piece, we found an invalid position.