Scrabble Score Calculator

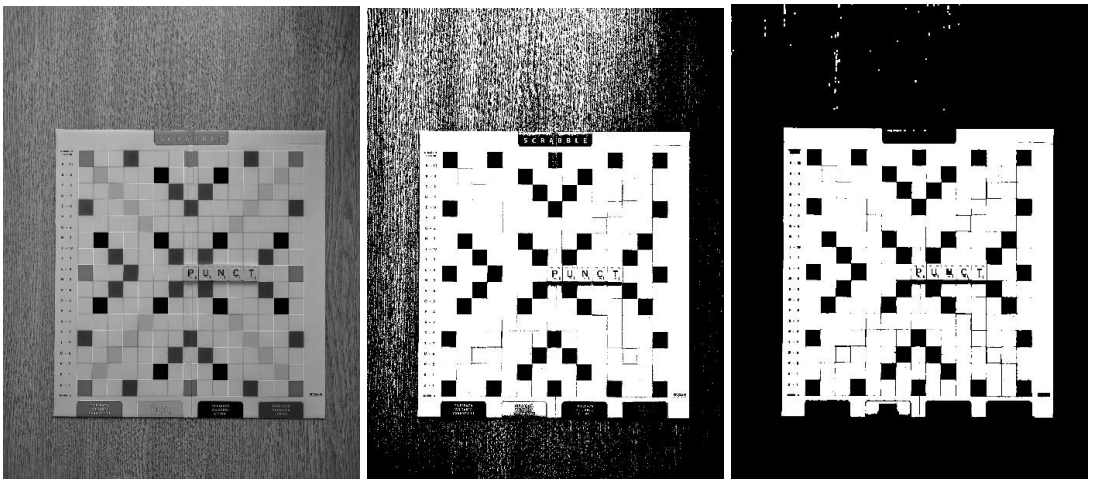
FMI UNIBUC **Burdușa Petru-Robert**
Computer Vision

The purpose of this project is to implement an automatic score calculation system for the word game Scrabble, using Python.

For training I had 100 images divided in 5 games, 20 rounds per game, and for each round I had to determine the score of the words that were formed with the newly added letters. For more clarity in code, I divided the project in multiple steps which have functions as counterpart in the code:

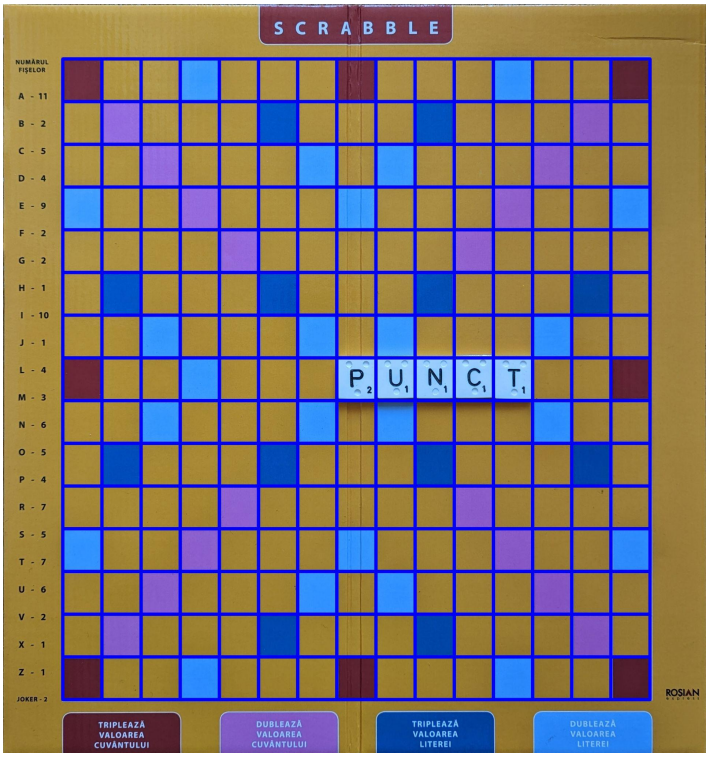
I. *Preprocessing the board images:*
```
imageProcessing(image)
```
II. *Extracting the board from the image:*
```
getBoard(original_image, image)
```
III. *Finding the cells on the board:*
```
findBoardCells(board)
```
IV. *Deciding for each cell if it has a letter or not:*
```
find_letter(cell)
```
V. *Preparing the templates:*
```
getTemplates()
```
VI. *Deciding what letter is in the cells that have one:*
```
classify_cell(cell, templates)
```
VII. *Calculating the score:*
```
score(board, letters_added)
```

I. Even if the images were taken in similar positions and lighting, I had to preprocess the images for a better result. The point was to choose what differentiates the board from the background, and I found that the red channel was more important for that than the rest. I applied a threshold to get a binary image where the board is white and the background is black, but there was a lot of noise so I applied an erosion filter, and then a dilatation filter to get the board to the initial size.

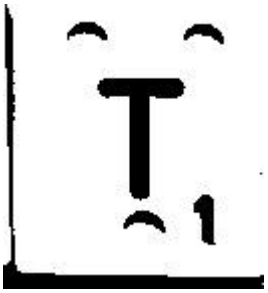


II. With the findContours() function from OpenCV I get the contours of the image, and choose the biggest. Then I find the corners of that contour that represents the board.

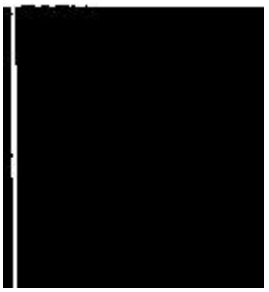
III. The next step is to split the board into a 15 by 15 grid. I took the measurements for this step manually, measuring the number of pixels of the board, the cells and the margins.



IV. The grid is now a matrix and I can go through it with 2 indices to get the cells. For each cell I get the green channel and apply a threshold, so the cells with no letter should now be mostly black. I calculate the mean value of the cell and if it is low(under 100), that means the cell does not have a letter in it.



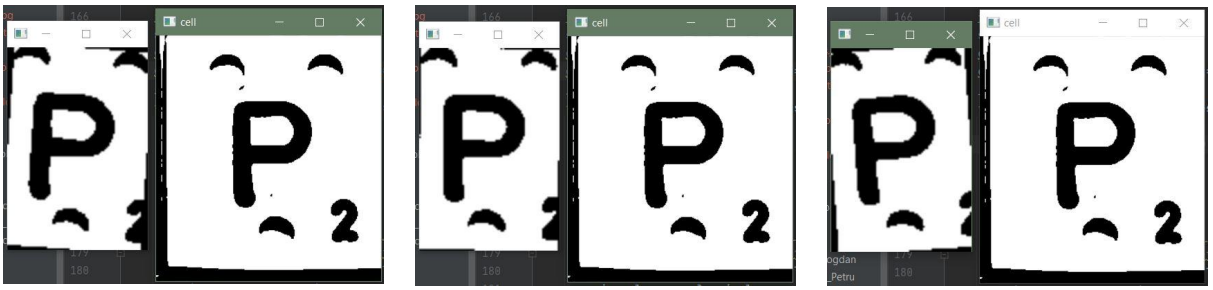
cell with letter



cell without letter

V. I extracted the cells with letters on them from an image that had all the letters on it, then manually cropped and straightened each one. After that I extracted the green channel and applied a threshold to get a binary image where the letter is black and the background is white. For each template I generate 8 more slightly rotated ones.

VI. For deciding what letter is in a cell I iterate through all letter templates(each letter with rotation variants) and make template matching between the cell and the template. Then I look for the biggest value and choose that as the result.



VII. Now I go through all the newly added letters and find the words that are formed with them. If just one letter was added at the current step, I go from that letter on the X and Y axis and find the words. If multiple letters were added, they will form a main word on one of the axis because they all have a common coordinate. Then I look at the newly added letters and search on the other axis for secondary words.

With this approach I was able to get right 99 from the 100 training images. In the image that I failed I only misclassified one letter.