Mathable - Computer Vision Autoscoring

Ilinca-Ana Moraru

March 13, 2025

1 Introduction

This project aims to calculate the score of a player's round in the Mathable game, based on images of the board. Mathable is a board game similar to Scrabble, but instead of words, the players need to create mathematical equations. The added tiles need to be the result at least one addition, subtraction, multiplication or division of the neighbours of the respective tile. The neighbours ought to be both either on the left, right, up or down the new tile. The board contains constraints (+, -, *, /) - the new tile placed on the constraint needs to be the result of the operation indicated on the board at that position. For each correct equation, to the score of the player it is added the number on the new tile. There are also bonuses, X2, X3 that make the score of the move double or triple. I worked on dataset of 4 games of 50 moves each.

2 The General Approach

Firstly, the original images are cropped and remapped to contain only the board game. Then, the borders of the board games are removed, so the results are straight grids and finding boxes at any coordinates is precise. Next, the project iterates through the images and detects the places where tiles were placed. The value written on the tile is identified using a K-means model trained on Mathable digits. Finally, the score is calculated according to all the rules and the results are outputed in text files.

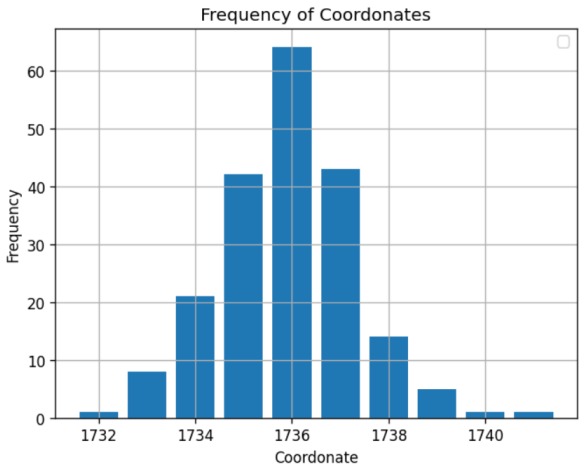

Figure 1: Example of one of the coordinates of the corners

3 The Preprocessing

3.1 Finding the corners of the board game

In order to have consistent corners for all board game images, I applied some filters on my dataset of 200 images that cropped the borders. First I applied a HSV filter alongside bluring and sharpening, in order to get sharp edges. The result is that the brown table got cropped out of the images that have now the same resolution. Next, on the big boards I targeted the yellow boxes that happen to be the corners of the playable space of boards with a HSV filter and plotted the most left-up, left-down, right-up, right-down coordinates of pixels that were targeted by the filter. As most had the same value, with variations of maximum 5 pixels, I decided to save the means of the coordinates. Therefore, the program will crop the borders in a consistent manner.

3.2 Constructing K-Means model

I analyzed the final images from four games provided in the dataset to identify the boxes containing placed tiles. Boxes were selected based on a grayscale pixel sum threshold of 400,000, which was determined to provide reliable results. After isolating the relevant boxes, I extracted the digits by cropping the regions defined by the leftmost, rightmost, topmost, and bottommost pixels of the digit contours. The extracted digits were then normalized to a consistent size using black padding. Finally, I applied K-means clustering from Scikit-learn to classify the digits and mapped the resulting clusters to the correct numerical values.

4 Computing the results

The images are processed game by game.

4.1 Cropping the board game

The big board is detected and mapped at a resolution of 2000x2000 after filtering out the brown table, providing resilience to changes in the position at which the images are taken. Next, the blue borders are cropped based on the coordinates found at preprocessing and all images are saved at a resolution of 1400X1400.

4.2 Computing the value of the tile

Images are compared to the one before and so, the coordinates at which a new tile has been placed are found. This is done by subtracting from the current image the one before and finding the biggest difference in grayscale format. The contours of the digits of that tile are found and each digit gets cropped. After normalizing the size of each image, the value of the digit is determined by the K-Means model trained previously. The value of the tile is then calculated and stored in a matrix of tile values so that the score can be computed. The boxes that are empty are stored as -1. The last step is writing in the file corresponding to the number of the move the coordinates and the value of the placed tile.

4.3 Computing the score

To compute the score of the current move there are 2 matrixes that store constant integers, one for the bonuses and one for the constraints. The first 2 neigbouring boxes that are to the left, right, up or down relative to the current tile are checked. If there are tiles there, it is checked if there is a valid mathematical equation between them and the current tile. If so, the value of the tile is added to the score. If the current box has a constraint, it is checked that the equation is of the specified type. Finally, if the box has a bonus, the score is multiplied by it. If the current move is the last of the set of moves of the current player, their added scores are written in the scores text file.

5 Conclusions

This project taught me that although the task seems easy, as the dataset is quite ideal as opposed to real world scenarios, it is time consuming to get to the final results as I found getting the right filtering tedious at times. If I were to change my approach, I would chose to use templates for the detection of digits rather than a K-Means model, as I realize it is unnecessary. Overall, I found working on this project an enjoyable way of learning the basics and I am glad to have some hands-on experience.