EE368 Project Proposal

David Koeplinger

dkoeplin@stanford.edu

Scrabble Assistant

Introduction

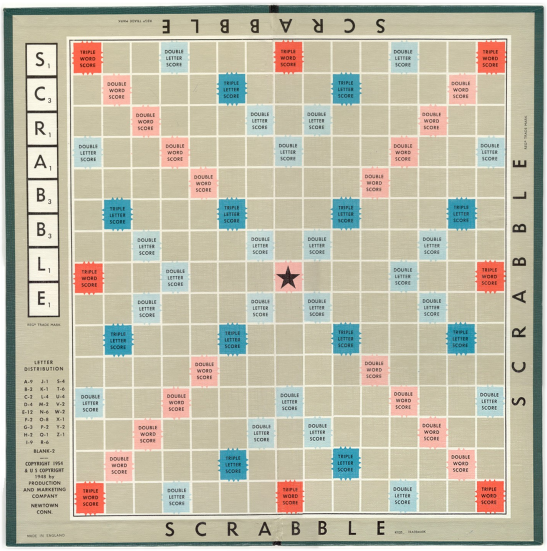
Scrabble is a commonly played word game in which players take turns forming words using a set of seven letter tiles and placing them onto a grid, following placement rules similar to a crossword puzzle. Points are awarded based on the sum of values of individual letters, along with letter and word multipliers positioned regularly throughout the board. While turns are not usually timed, creativity and vocabulary are both important to achieving a high score. With the increasing popularity of Scrabble-like online games like Words with Friends, various websites like wordfind.com have been created to help players create words from their set of tiles. However, the most challenging (and often most frustrating) part of the game still remains: the player still needs to find a spot on the existing board to put their word given Scrabble's placement rules.

To help augment these hint websites, I propose creating an image processing application which identifies both a player's set of letters and the current board state using a single image of the board and the player's tiles. At a high level, the solver application should be able to do the following:

1. Account for small amounts of blurring during image capture
2. Identify tiles within the input image, invariant to perspective skew, rotation, and scale.
3. Extract letters from the tiles.
4. Superimpose a grid on the Scrabble board to identify potential spots for the player's new word.
5. Query a hint server to find possible words given these openings.
6. Return a list of words, ranked by score.

Challenges

In classifying letters, the system will have to ignore extraneous characters in the scene, including characters that are part of the game board design. For example, in the board design on the right, the characters in “Scrabble” are approximately the same size, font, and orientation as those on the game's tiles. The stylized “Scrabble” on the left is printed to look like it was created using game tiles, even including each letter's score on the bottom left.



[http://imgkid.com/scrabble-board-game-template.shtml]

Another challenge in detecting letters in Scrabble is the wildcard tile, which has no letter or point value on its face. Distinguishing this tile will require several assumptions:

1. The tiles on the board are always connected relative to the board's grid (per Scrabble rules)

2. All words on the board can be found in an English dictionary.
3. Players usually have exactly 7 tiles at the beginning of their turn.
4. Only two blank tiles exist in the game.

Players typically keep their tiles on a rack so that opponents are unable to tell which positions on the board they might be hoping will remain open. This effectively means in a single image, the tiles on the board and the player's tiles will be skewed relative to each other. This project will begin by assuming that the player's tiles and the board are on the same plane, then move on to local skew correction if necessary and time allows.

Algorithm

The procedure for identifying letters on the Scrabble board and in the player's “hand” will roughly be as follows:

1. Correct for blur during image capture [3].

2. Correct for perspective skew using a method similar to that described by Cao, Wang and Li [1]. This method should be very effective for the Scrabble board, since it has a large number of easily identifiable straight lines.

3. Use local thresholding to segment the image, then detect character regions and identify characters using the pattern comparison method described in [2].

4. Establish a grid on the skew corrected board image by using character regions' central moments. Insert blank tiles in the grid for disconnects, keeping in mind the above assumptions for blanks.

5. Identify the set of characters off the grid which represents the player's tiles.

6. Identify all grid openings and corresponding sizes (in number of tiles).

7. Submit these combinations to a hint server, filter the results by those that can fit on the board, sort the resulting list by score, and return the top results to the user.

References

[1] Cao, Y., Wang, S., & Li, H. (2003). Skew detection and correction in document images based on straight-line fitting. *Pattern Recognition Letters, 24*(12), 1871-1879.

[2] Ohya, J., Shio, A., & Akamatsu, S. (1992). Recognizing characters in scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 214-220.

[3] Tian, Y., & Ming, W. (2009). Adaptive Deblurring for Camera-Based Document Image Processing. *Advances in Visual Computing*, *5876*, 767-777.