Scrabble Assistant

David Koeplinger
Department of Electrical Engineering
Stanford University
Stanford, CA
dkoeplin@stanford.edu

***Abstract*—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. Various applications exist for helping players with forming words from their tiles, but few account for all of the possible positions on the board simply because manually entering the board state is tedious. This paper presents an image processing algorithm for extracting and recognizing characters from Scrabble game boards for use in general Scrabble backends. In the test images used to evaluate this algorithm, boards were digitized with 87% character accuracy on average.**

Keywords—Scrabble; character recognition

I. 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 needs to find a spot on the existing board to put their word given Scrabble's placement rules. Commercial applications rarely take the board state as input; entering the board manually is tedious for the user, but digitizing the board automatically is relatively complex and may be unreliable. This paper presents an image processing algorithm for digitizing Scrabble boards from a single image of the board. The algorithm is evaluated for use with a Scrabble "oracle" backend which takes as input a board state and a set of tiles and suggests the best possible scoring word to the user.

II. IMAGE PROCESSING ALGORITHM

A. Assumptions

This algorithm assumes that any perspective skew in the input image is small. In particular, it assumes that visible tiles on the board are approximately square and roughly the same size. It also assumed that the Scrabble board is a standard 15 x 15 grid and that most of the board is visible in the image. All figures used in this section are from processing steps for image 13. The results for the remaining images can be seen in the Results section and in the Appendix.

B. Tile Size Estimation and Normalization

The image is first converted to grayscale and adaptive histogram equalization is applied to account for local variations in image brightness, such as glare from the Scrabble board itself.

In order to determine which regions of the input image are relevant text, the size of the characters must first be estimated. This estimation is made using the size of empty squares on the board (characters on Scrabble tiles are roughly half the size of the tiles). To estimate tile size, maximally stable extremal regions (MSERs) are located. Regions are restricted to be no larger than 1/225th of the area of the image. MSERs with a width and height that differ by more than 0.1% are discarded, as are regions with less than 100 pixels and regions with a density less than 80%. The median width of the remaining regions is then taken to be the tile size. The image is then resized such that tiles are 30 pixels wide.

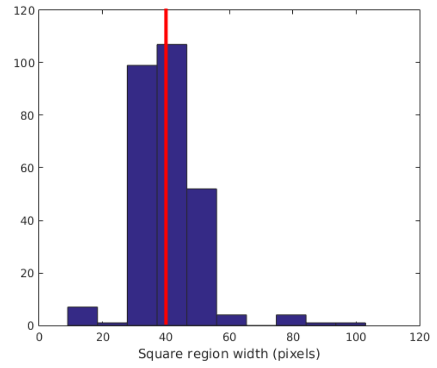


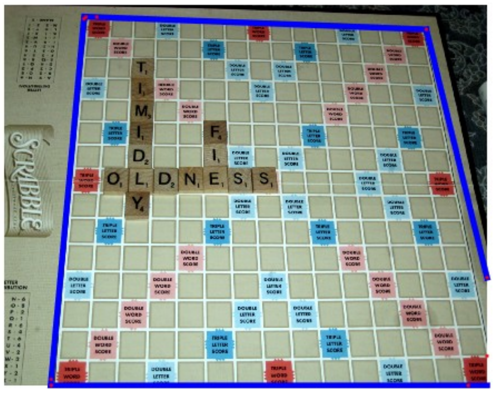
Fig. 1. Square MSER width distribution for Image 13. The vertical line is the calculated median width.

C. Grid Detection

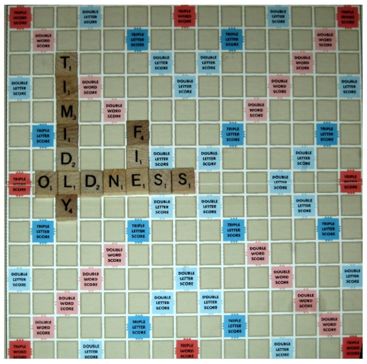
Following tile size normalization, the edges of the rescaled grayscale image are calculated using the Canny edge detector. Edges are then merged using a close operation with a disk structural element of radius six. This creates a connected region of tile borders forming most of the 15 x 15 grid, but it also can connect the board to extraneous image details, such as the nearby Scrabble logo on the board. To counteract this, the merged edges are then repeatedly dilated with a small (3 pixel) vertical structural element and eroded with a similarly sized horizontal element. This has the effect of eliminating horizontal bridges in the edge map while also preserving most of the features of the grid. The region with largest area within the edge map is then taken to be the board's grid. This area is then closed with a structural element approximately the size of a tile to convert the edge map into a blob.

D. Perspective Skew Correction

Although the input image is assumed to have small skew, even small viewing angles can affect the simple strategy later used to map character regions to a 15 x 15 matrix. To counteract this, corners are located within the previously isolated grid. The minimum bounding quadrilateral [2] is then computed to estimate the actual boundaries of the board (see Fig. 2). The affine transformation from the bounding quadrilateral's corners to an unskewed square are computed. The side of the square is calculated as the mean of all of the bounding quadrilateral's sides. The grayscale image is then transformed using this affine matrix and cropped.



A



B

Fig 2. Perspective skew correction of Image 13. A) Quadrilateral mapped to corners of resized image, shown overlaid in blue. B) Image after affine transformation.

E. Grid Fitting

Once the image has been transformed, the board's grid is now roughly aligned with the horizontal and vertical axes. However, the grid's alignment with the axes is not perfect due to small errors in the skew correction and other small irregularities such as curvature due to the board not being a completely flat surface. In order to determine the locations of characters on the board, a grid must be fit to the image. This is initially done by dividing horizontally and vertically by 15 in each dimension. The image is gamma corrected to sharply increase the contrast of bright regions, emphasizing empty squares. Square MSERs in the resulting image are again located as before. The image is then gamma corrected to strongly increase the contrast of dark regions, emphasizing tiles and characters. Square MSERs in this image are located as well, but with a width : height ratio threshold of 5% since most characters are not completely square. The resulting two sets of regions are concatenated and eroded to increase the distance between them. These regions are then classified by their location with the horizontal and vertical lines of the rough 15 x 15 grid previously created. These borders are then refined by moving vertical and horizontal boundaries to the mean of the distance between regions on either side of each boundary. Boundaries are not moved if no regions were found on at least one side. As seen in Fig 3., the resulting grid is then a close fit to the skew corrected image.

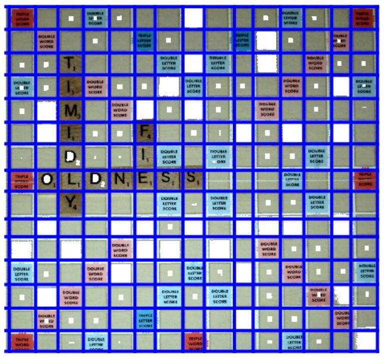


Fig 3. Grid fit for Image 13. Blue lines represent calculated horizontal and vertical boundaries. White regions show the MSERs used to refine the grid.

F. Character Detection

Now that the image has been discretized using grid boundaries, characters can be mapped to the grid based upon their centroid. Characters are recognized using the algorithm described by Chen in [1] to recognized and isolate characters in natural images. A brief outline of this algorithm follows.

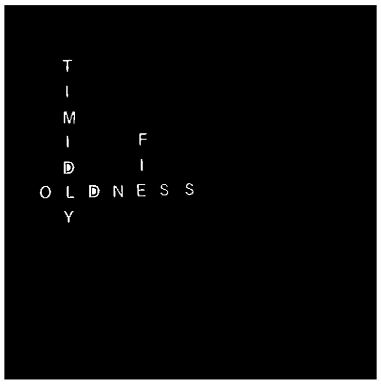


Fig 3. Isolated character candidate regions in Image 13.

MSERs of the image around the expected size of characters are computed, then intersected with the original image's edge map. These intersections are then grown along the local gradient of the original image, then inverted and intersected with the edge-enhanced MSER map. This has the effect of “pushing” noise away from areas with strong edges, like characters. It also has the side effect of eroding characters slightly. Tile sizes were previously normalized to ensure that this erosion effect does not eliminate any characters.

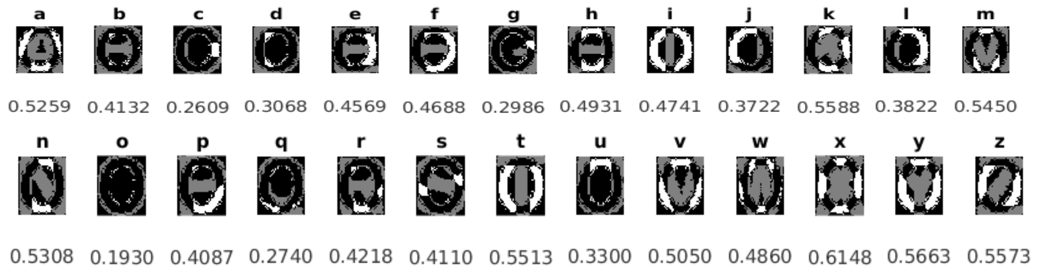


Fig 4. Template matching results for character 'O' in Image 13.

The remaining regions in the image are then filtered. The first pass removes regions based upon expected minimum and maximum area of the characters, as well as typical eccentricity and solidity of character regions. These parameters required a small amount of fine tuning to properly capture the standard Scrabble font. The second filtering pass eliminates regions based upon stroke width variation. A third filtering pass added for the Scrabble algorithm filters regions by height and width, since the expected range of dimensions for characters is well known in this context. An example of the final character candidates resulting from these filtering passes can be seen in Fig. 3. Note that characters which inscribe a closed area occasionally have regions within them. This is generally not an issue, since the area of those regions is usually smaller than the character's total area.

G. *Character Recognition*

After being isolated, the character candidate regions are categorized. The final character recognition method used template matching based upon a set of reference tiles, but other methods were attempted as well (see Discussion). The Hough transform of the reference tiles are precomputed during extraction and angles corresponding to the top twenty peaks are extracted. To compare a character candidate region to the templates, the Hough transform is first calculated for the region. The character region's rotation angle is estimated by computing average of the difference between its top twenty angles and each template's, individually. The region is then rotated separately for each template and fit to the template, counting the number of pixels which differ. Since angle estimation using the Hough transform is not always accurate, the same process is also done without any angular correction. The templates with the smallest error across all comparisons is chosen as the region's label. Note that this process assumes that non-character regions have already entirely been filtered out and makes no attempt to classify regions as non-characters.

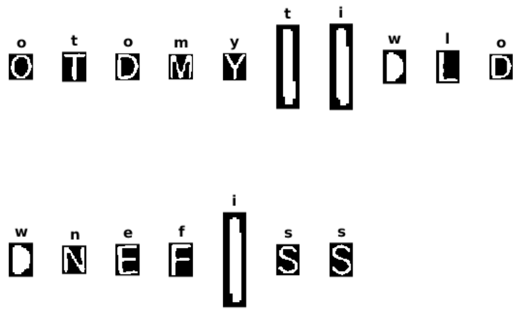


Fig 5. Template matching results for all character candidate regions in Image 13.

After labeling, the character regions are then classified by their center's relation to the grid previously fit to the warped image. Grid locations are classified by the most commonly occurring label within that grid. The label of each grid is then written to a file for processing

H. *Board Representation*

The matrix of characters is represented as a 15 x 15 grid, with dashes for empty squares and lowercase characters for letter tiles. Rows are separated by linebreaks and columns are separated by spaces. This file can of course be used in any backend of interest. For evaluation of the algorithm, the backend used was a Scrabble oracle written in Scala which determines the best possible word to play given the user's set of tiles and the current board state.

III. Results

The Scrabble image processing algorithm was evaluated on a series of thirteen images taken from an online Scrabble "sweepstakes" [5], which challenged players to find the maximum scoring word given an image of the board and image of their tiles.

The character recognition rates can be seen in Fig 6. These rates are calculated as the number of correctly recognized characters in the image, ignoring blank tiles. Character recognition is, on average, 87% accurate, spanning between 100% and 78% depending upon the image.

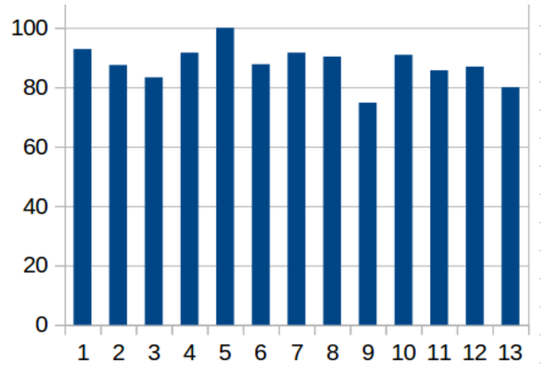


Fig 6. Character recognition rates for all images evaluated, in percentag points. Recognition rate is calculated as the number of correctly recognized characters divided by the actual total number of characters in the image (no blanks).

The output of the Scrabble algorithm was fed into a Scrabble oracle. Fig 7. outlines the results of the oracle.

TABLE I. TABLE TYPE STYLES

	Oracle Results					
	Human Best	***Pts***	***Best Possible***	***Pts***	***Scrabble Assist. Best***	***Pts***
1	UNITIES	73	INQUIET	74	INQUIET	74
2	SNARKIER	78	INSANER	83	SNARKIER	78
3	REALISM	79	REALISM	79	SIMILAR	67
4	OBLONGS	74	ENGLOBES	76	ENGLOBES	76
5	ICEBOAT	73	ICEBOAT	73	ICEBOAT	73
6	TURTLERS	71	ULSTER	74	ULSTER	74
7	TAWNIER	81	WANIEST	86	WANIEST	86
8	PROSING	91	PROSING	91	PROSING	106
9	AGONIES	75	OCEANIDS	75	OCEANIDS	75
10	SMOKING	75	SMOSING	82	SMOSING	82
11	QUESTING	77	QUESTING	77	QUEINGS	77
12	CLAMMER	99	CLAMMER	99	CLAMMER	99
13	LIFTGATE	74	LIFTGATE	74	GANEF	30

Fig 7. Table of results from Scrabble sweepstakes

IV. DISCUSSION

As can be seen in Fig 7., character recognition was not always reliable enough for the oracle to return words that scored the highest possible amount. In several cases, the words are actually invalid because they do not match the actual tiles on the board. This demonstrates how important character recognition accuracy is for this task – a single character wrong can mean missing the best possible word. For this algorithm to be commercially viable, future improvements should focus on improving character recognition accuracy. Most of the errors made during character recognition are immediately understandable – eg. recognizing a O for a D, an I for a T, etc. Improving the error classification beyond a simple count of differing pixels may go a long way towards improving character recognition accuracy. Grouping the differing pixels into regions and calculating the average area of each region, for example, may be a better error metric.

The character recognition method used in this paper was based upon template matching. Many other single character recognition techniques have been proposed. The Banagrams solver [5], for example, compared the first four Hu moments [4] of character candidates and as shown to be extremely reliable for that application. An algorithm for individual handwritten recognition is proposed in [6] but was not very accurate on the Scrabble text. There are also existing Optical Character Recognition (OCR) engines, but these tend not to do well on individual character recognition without manipulating the image extensively.

V. FUTURE WORK

Blank tiles are not detected in this version of the algorithm. One approach to detect these tiles might be to use gamma correction to find regions of the board which are covered, but which have no characters. This approach would of course have to be made robust to local variations in brightness across the image.

All evaluation images have a small amount of perspective skew, but the algorithm still needs to be evaluated on images with larger skew, as well as on images taken where characters are rotated by more than 35 degrees (from the opposite side of the board, for example). The algorithm could also be evaluated on and made more robust to a variety of Scrabble board styles.

This algorithm is currently implemented in Matlab using the Visual Processing Toolbox. In the future, it could be ported for use as a mobile application.

ACKNOWLEDGMENT

The author would like to thank Andre Araujo for his suggestions on improving the algorithm as well as Huizhong Chen for his extremely helpful tutorial on character detection and isolation in Matlab.

REFERENCES

[1] Chen, Huizhong, et al. “Robust Text Detection in Natural Images with Edge-Enhanced Maximally Stable Extremal Regions.” Image Processing (ICIP), 2011 18th IEEE International Conference on. IEEE, 2011.

[2] D'Errico, John. “A suite of minimum bounding objects” Matlab File Exchange. http://www.mathworks.com/matlabcentral/fileexchange/34767-a-suite-of-minimal-bounding-objects

[3] Hu, M. (1962). “Visual pattern recognition by moment invariants.” IRE Trans. Information Theory, vol. 8, no. 2, 179 - 187.

[4] Leung S., Perkins, S., and Rhoades, C. “Bananagrams Tile Extraction and Letter Recognition for Rapid Word Suggestion.” EE368 Winter 2013-2014, unpublished

[5] Paul Rivas, “Sunday Scrabble Sweepstakes.” One Sorry Blog. WordPress. 2007. https://onesorryblog.wordpress.com/category/sunday-scrabble-sweepstakes/

[6] Gaurav, D. D. and Ramesh, R. (2012). “A feature extraction technique based on character geometry for character recognition.” CoRR, vol. abs/1202.3884.

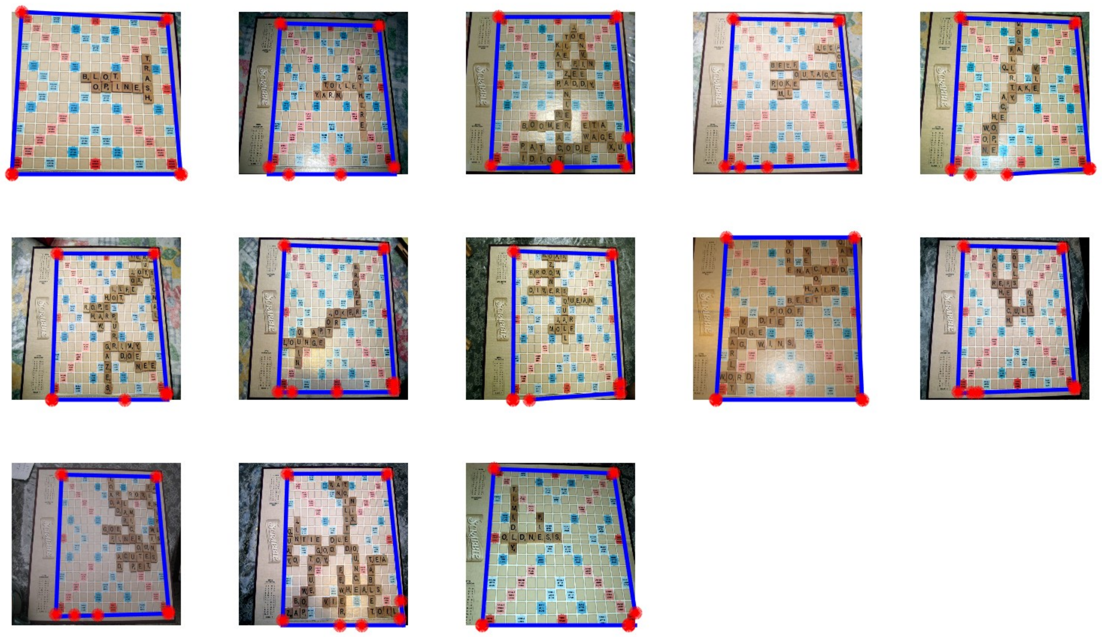
SUPPLEMENTAL MATERIALS

Koeplinger_Scrabble_Assistant.zip

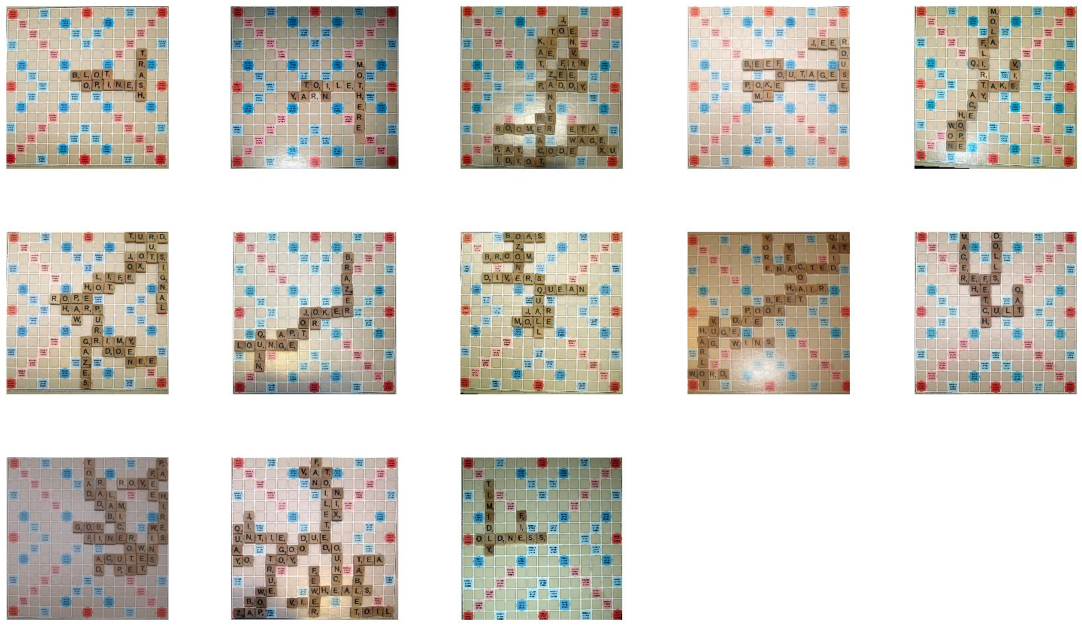
This compressed file includes the Scala Scrabble oracle and the Matlab implementation of the algorithm. It also includes the 13 images used for evaluation and the corresponding tile images, as well as the “gold” files for each.

A slightly larger Scrabble board and tile rack image database was collected for this project, but is not included with the source code. If you are interested in building upon this project, feel free to contact the author for the larger database (see email in paper heading).

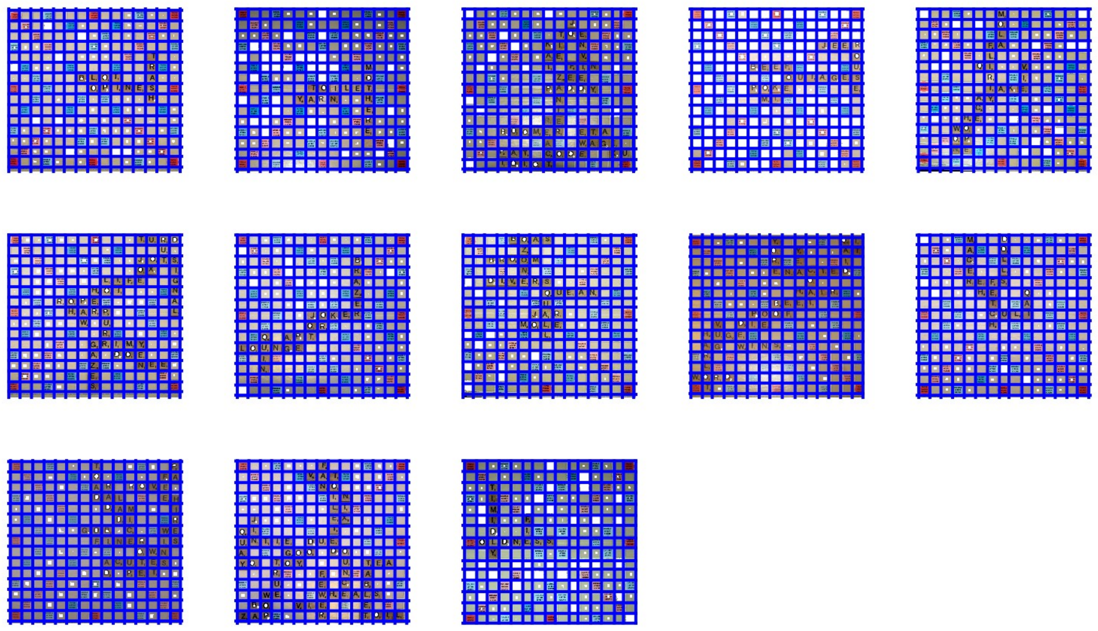
Appendix



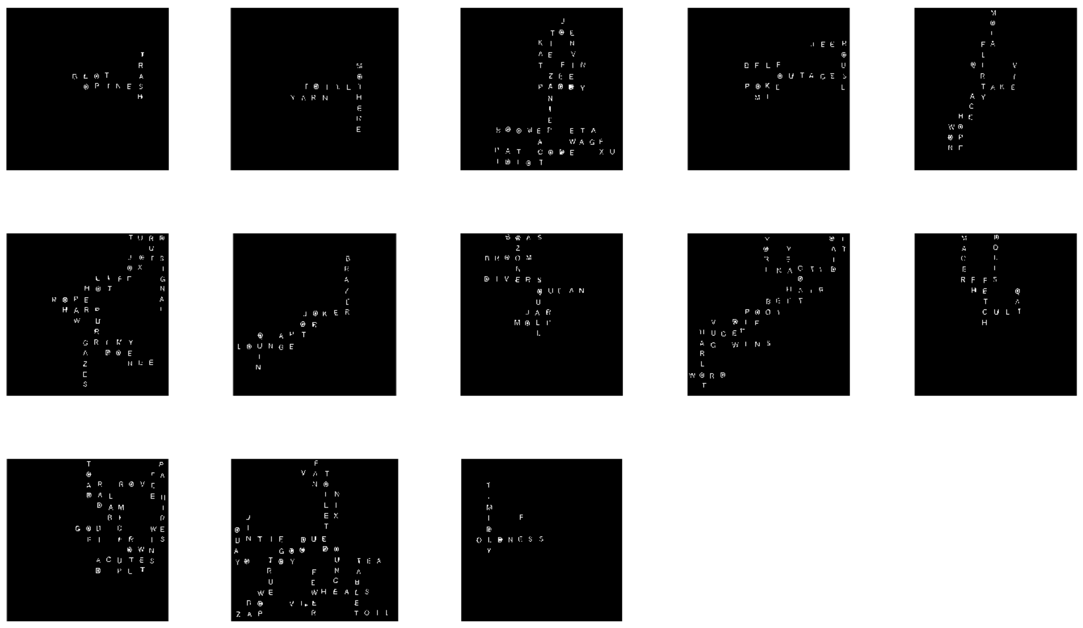
Quadrilateral mappings for all thirteen images



Images after affine mapping



Images after grid mapping



Images after character isolation. Note that the small size adds some distortions to the image.