CrosScan: The Crossword Scanning App

Ryan Dorson*
Department of Computer Science
University of Maryland
College Park, MD 20742
rdorson@terpmail.umd.edu

Evan Golub
Department of Computer Science
University of Maryland
College Park, MD 20742
egolub@cs.umd.edu

David Jacobs
Department of Computer Science
University of Maryland
College Park, MD 20742
djacobs@cs.umd.edu

ABSTRACT
CrosScan: The Crossword Scanning App is an Android application that allows users to scan crossword puzzles into their mobile devices using the camera so they can complete the puzzles digitally. There are two aspects of this project: scanning puzzles from newspapers, magazines, and other physical media, and solving the scanned puzzles interactively in the application. Usability testing will allow me to base the design of the project on the preferences of the target audience, crossword enthusiasts. By asking participants to complete a few tasks with the application, I hope to learn what end users expect from a crossword scanning application, which features are necessary for the application, and how to improve those features. In this study with no known risks, human subjects will be protected by minimizing loss of confidentiality by storing data on a password-protected computer and assuring they are informed of their rights to withdraw from the study at any time.

1. INTRODUCTION
It is becoming more common to see people using mobile technology, rather than pen and paper, to take notes [8]. Whereas conventional note taking requires pens and pencils, which are often inconvenient to locate, mobile technology makes note-taking convenient and accessible, requiring only a readily available finger. In many similar cases, mobile applications are reinventing tangible technologies. However, with the reinvention of tangible technologies on the digital platform, it becomes difficult to bridge the gap between the physical and the digital.

The e-book market, for example, makes it convenient for users to access millions of titles on their mobile devices, but for a price [11]. If the users already own a collection of physical books, then it is unlikely that they will want to buy digital copies of the same collection. Hence, they must deal with the inconvenience of carrying the physical books with them, despite having an e-book reader. A system for bringing the physical books into digital media would help users to avoid such a problem.

With the recent proliferation of Quick Response (QR) codes, industry has found a solution for bridging the gap between tangible and digital technologies. QR codes are notable for their ability to store large amounts of information in comparison to traditional UPC barcodes, as well as their accessibility to mobile device users [6]. With QR codes affixed to many physical objects (e.g., for advertising and publicity reasons) anyone with a mobile device can bridge the gap between the physical and the digital, by scanning the code.

Crossword puzzle solvers find themselves in a similar predicament as e-book market consumers. If they have a physical crossword puzzle they want to solve (e.g. from a newspaper or magazine), there is no way to bring the puzzle into digital media. Thus, they must carry around the physical puzzle and pen despite having the capability to store and solve crossword puzzles on mobile devices. CrosScan: The Crossword Scanning App solves this problem by allowing crossword puzzle solvers to scan puzzles from physical media and complete them on their mobile devices.

CrosScan is an Android application that allows users to scan crossword puzzles into their mobile devices using the camera so they can complete the puzzles digitally. There are two aspects of this project: scanning puzzles from newspapers, magazines, and other physical media, and solving the scanned puzzles interactively in the application. Thus, once users take a picture of a crossword puzzle, CrosScan reproduces the grid and allows users to enter the answers to the clues. CrosScan seeks to answer the question of how to scan a crossword puzzle from physical media into a digital medium. The current study also seeks to answer the question of what potential CrosScan users would want from such an application.

2. REVIEW OF RELATED WORKS
Crosswords [2] is a crossword puzzle application that allows users to download crossword puzzles into the app from various sources and complete them digitally. This application partially solves the problem by bringing puzzles users can complete physically onto the digital platform. However, the method is indirect — users can only access the puzzles once publishers release them to the Crosswords app. In addition, users can only access a limited database of digitized cross-

*CS Honors Undergraduate Student

word puzzles, rendering them unable to complete puzzles digitally from a large number of other sources.

Sudoku Grab [4] is a Sudoku puzzle scanning application created by Dr. Chris Greening. It is similar in design to CrosScan in that it is capable of scanning Sudoku puzzles from physical media so that users can complete them digitally. It differs in that the scanner must recognize a 9x9 grid of numbers, whereas CrosScan must recognize a grid (black and white squares) of variable size and a list of clues. However, many of the image recognition methods implemented in the Sudoku Grab app are similar to those used in CrosScan.

3. COMPUTER VISION APPROACH

Several computer vision techniques lend themselves to the successful recognition of the crossword puzzle grid. In this paper, I use the OpenCV4Android implementations of these techniques, described below.

The first technique involves retrieving the photo (see Figure 1) and converting it to a grayscale image (see Figure 2). This step simplifies the processing of the image, since color information is not necessary for successful recognition of the crossword puzzle. In a grayscale image [10], pixel information is represented on a single channel, with each pixel ranging in intensity from 0 (black) to 255 (white).

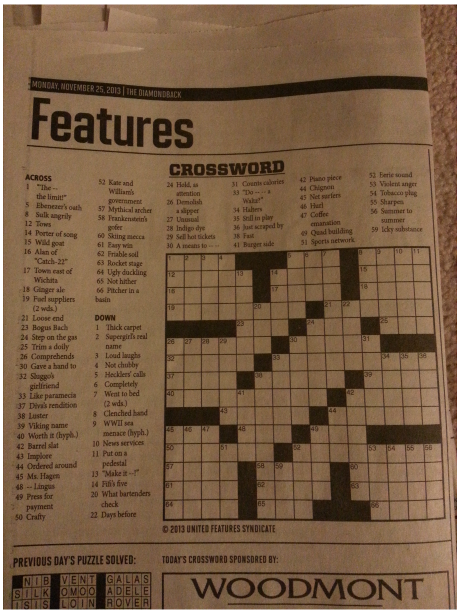

Figure 1: The original image.

Figure 2: The grayscale image after conversion.

The next technique, adaptive thresholding [9], detects edges in the grayscale image (i.e., edges of the crossword puzzle grid). In general, thresholding refers to producing a binary image of black and white pixels from a corresponding grayscale image based on a threshold. It involves choosing a threshold value between 0 and 255 and labeling each pixel in the binary image black or white based on whether the intensity of the corresponding pixel in the grayscale image is higher or lower than that value. However, this general technique does not detect edges well or take into account shadows and other anomalies in the image. Thus, in adaptive thresholding, the algorithm chooses a threshold for each pixel in the image based on its surrounding pixels. This threshold value often corresponds to the average intensity of surrounding pixels. Thus, it is possible to detect edges in the image, since the dark pixels have a lower intensity than surrounding light pixels. In the current case, adaptive thresholding produces a binary image where white pixels correspond to low intensity pixels in the grayscale image and black pixels correspond to high intensity pixels in the grayscale image (see Figure 3).

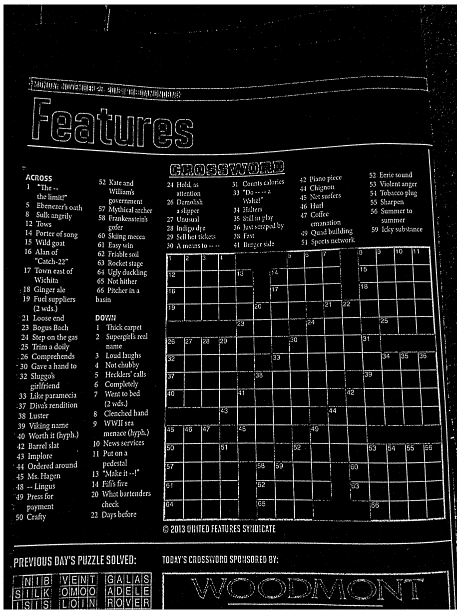

Figure 3: The binary image after adaptive thresholding.

Blob extraction, also known as connected-component labeling [5], allows for the detection and extraction of a list of "blobs" from the image. In the case of CrosScan, the blobs are regions of white pixels in the binary image. By taking this list of blobs and sorting it by blob area, the algorithm can extract the largest blob from the image. Given that the focus of the image is the puzzle grid, this method will extract the blob consisting of grid lines. Hence, it finds the region of the image containing all grid data (see Figure 4).

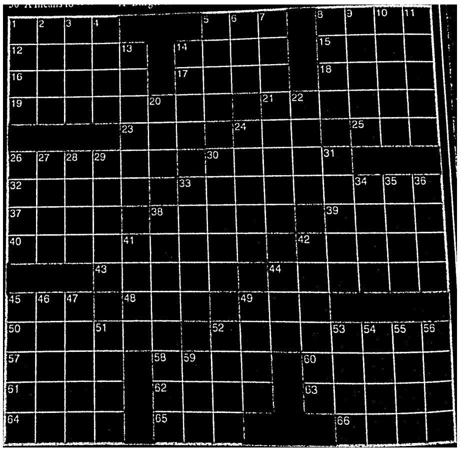

Figure 4: The edge image cropped to the region with the largest blob.

The next technique, the Hough line transform, works on the binary image limited to the portion discovered via blob extraction. The algorithm finds lines in the image by grouping detected edges in the image [3]. Using a voting procedure, it determines which lines are the most accurate candidates, throwing away lines with small numbers of votes. The Hough line transform finds the lines of the crossword grid well, but it also introduces many extraneous lines due to noise (see Figure 5). Thus, an algorithm for grid extraction from detected lines is necessary.

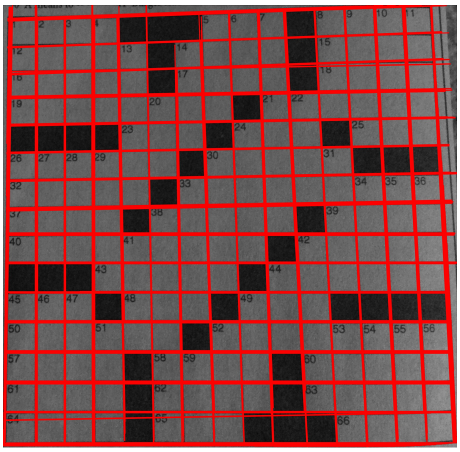

Figure 5: The line image with extraneous lines.

4. GRID EXTRACTION

The problem of grid extraction essentially comes down to finding the largest sets of approximately equally spaced horizontal and vertical lines in the image. Finding a set of equally spaced lines in the image allows the algorithm to discover the boundaries of the squares. Finding the largest set of equally spaced lines ensures that the algorithm discovers the whole puzzle.

The algorithm begins with finding the orientation of the grid by counting the frequency of each line angle. The most frequent angle becomes the horizontal line angle, and the vertical line angle is 90 degrees from this angle. The algorithm will swap the angles if one is closer to horizontal than the other angle.

Next, the algorithm adds all lines with angles close to the horizontal angle to a horizontal list and those with angles close to the vertical angle to a vertical list. The algorithm sorts the lines in the list based on the average vertical and horizontal positions, respectively, of the lines. By sorting, it is possible to find the largest set of horizontal and vertical lines in $O(n^3)$ steps, where n is the number of lines in the set.

The algorithm iterates through every pair of lines in $O(n^2)$ steps. For each pair of lines, it calculates the distance d between them and adds them to a set. If d is greater than a minimum square size threshold, then the algorithm iterates through successive lines, looking for ones that are distance d from each other (or within a small threshold of this distance). It adds each line that satisfies this property to the set in $O(n)$ steps. If the resulting set is larger than the current largest set, then it keeps the set. The result of the algorithm is the largest set of equally spaced lines in the image (see Figure 6).

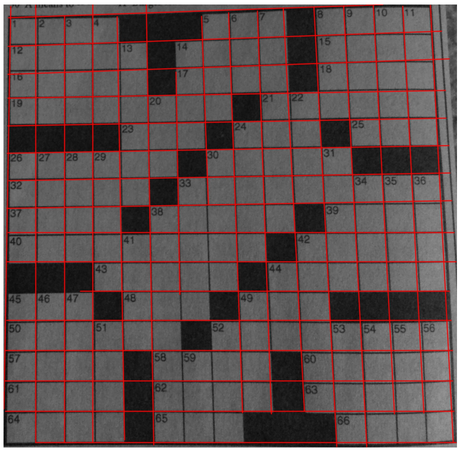

Figure 6: The resulting largest set of horizontal and vertical lines.

The next task involves recognizing where the squares are from the line data. The algorithm computes the intersection of each pair of horizontal and vertical lines (or projected intersection if the segments do not intersect). The algorithm normalizes these points so they are within the boundaries of the puzzle grid and considers the points to be the corners of the squares (see Figure 7).

The next task is to recognize whether the squares are black

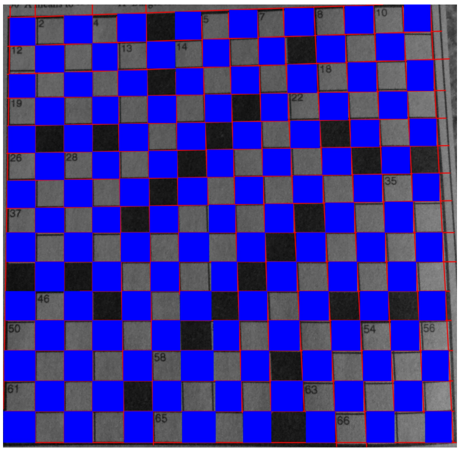

Figure 7: The blue checkerboard pattern designates detected squares.

or white. The algorithm finds the average intensity of each of square (by adding up pixel values in each square and dividing by the number of pixels). It builds an image where each pixel corresponds to each square and has a value equal to the average intensity for that square (see Figure 8).

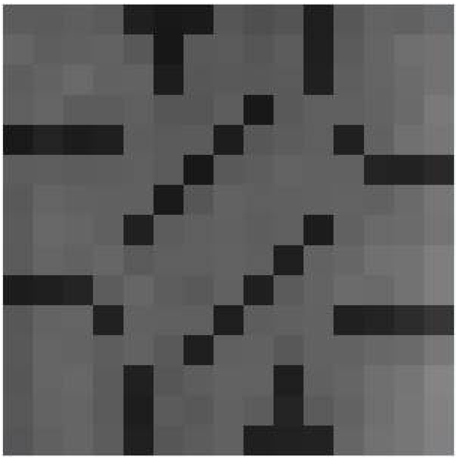

Figure 8: The pixel image based on average intensity.

It then applies adaptive thresholding to the image to produce a binary image where black pixels correspond to black squares and white pixels correspond to white squares (see Figure 9). Using this image, it assigns black or white to corresponding squares in the crossword grid. Finally, the algorithm serializes the grid data, containing grid size and locations of black and white squares, so CrosScan can read and display the puzzle.

5. RESULTS

In order to measure the performance of the grid extraction algorithm, I analyzed a sample of 50 crossword puzzle images. Using the app, I scanned the 50 images from 10 different crossword puzzles in 5 different lighting and orientation environments. I measured the ability of the app to extract all the squares from the image on the first try, recording

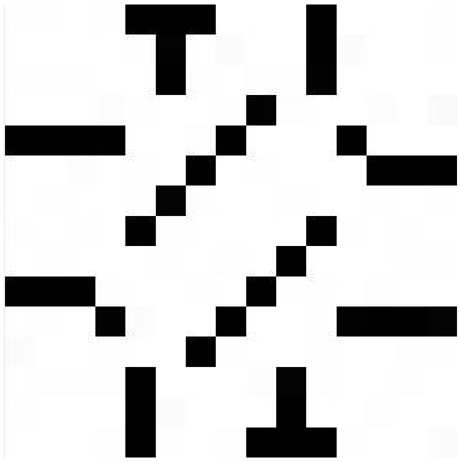

Figure 9: The binary pixel image after adaptive thresholding.

precision and recall to calculate the F-score.

The crossword puzzles came from a variety of sources, as detailed in Table 1.

The environments consisted in different lighting sources and orientations, as shown in Table 2.

Precision measures how many squares the algorithm detected correctly out of all the squares it detected. If the algorithm detects a non-existent square from outside the grid, then the precision will be less than 1. Recall measures how many squares the algorithm detected correctly out of all the squares it should have detected. If the algorithm does not detect all of the cells in the grid, then the recall will be less than 1.

$$Precision = \frac{|correctly\ detected\ squares|}{|correctly\ detected\ squares| + |incorrect\ squares\ added|}$$

$$Recall = \frac{|correctly\ detected\ squares|}{|correctly\ detected\ squares| + |correct\ squares\ missed|}$$

I used these values to calculate the F-score, F, which factors in both precision and recall. Then I went through the detected squares and found the percentage of square correctly labeled black or white. I multiplied this value by the F-score to generate a scaled F-score, SF, which factors in the precision and recall of square detection along with the percentage of correctly labeled squares.

$$F = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

$$SF = \frac{|correctly\ labeled\ squares|}{|squares\ in\ grid|} \times F$$

The algorithm performed fairly well, with an average F-score of 0.9530 across all puzzles and environments (see Table 3). Except for 2 cases, when the algorithm did not turn out an

Puzzle	Size	Format	Notes
crossword-1	15x15	Newpaper	None
crossword-2	15x15	Newpaper	None
crossword-3	21x21	Online print-out	None
crossword-4	15x15	Online print-out	None
crossword-5	15x15	Newpaper	Partially completed, eraser marks
crossword-6	15x15	Online print-out	None
crossword-7	15x15	Online print-out	British crossword puzzle
crossword-8	15x15	Online print-out	Landscape photo
crossword-9	15x15	Online print-out	Smudgy
crossword-10	15x15	Computer screen	None

Table 1: Puzzle details.

Environment	Description
Environment 1	Daytime inside, closed blinds
Environment 2	Daytime inside, open blinds
Environment 3	Daytime outside
Environment 4	Nighttime inside
Environment 5	Same as Environment 1, but rotated 10 degrees counter-clockwise

Table 2: Environment details.

		Crossword										
		1	2	3	4	5	6	7	8	9	10	Average
Environment	1	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	2	1.0000	0.8824	1.0000	1.0000	0.8333	1.0000	1.0000	1.0000	1.0000	1.0000	0.9716
	3	1.0000	0.9375	0.8400	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	0.8889	0.9666
	4	0.9822	1.0000	1.0000	1.0000	0.9677	1.0000	1.0000	1.0000	1.0000	1.0000	0.9950
	5	0.8065	1.0000	0.0000	0.9677	0.9269	1.0000	0.8824	0.8333	0.9653	0.9375	0.8320
Average		0.9577	0.9640	0.7680	0.9935	0.9456	1.0000	0.9765	0.9667	0.9931	0.9653	0.9530

Table 3: Scaled F-score results.

F-score of 1 for a given puzzle and environment, rescanning a second or third time resulted in an F-score of 1.

The first exception to this rule was crossword-10 in Environment 3, since the computer screen was too dim outside to recognize the color of the squares. However, it did detect the correct number of squares. The second exception was crossword-3 in Environment 5, since the puzzle was large and rotating it caused the size of the squares to be small. The algorithm failed by judging the width of a cell to be the width of two actual cells. Thus, the algorithm consolidated 2 squares into 1 several times.

Other than these issues, there were 3 main problems with recognition. The first was rotating the puzzle (as can be seeing in the Environment 5 results in Figure 10), causing the algorithm not to detect squares in the puzzle. The second was not detecting all the squares in the image (resulting in a lower recall) due to a variety of reasons. The last was shadows over the puzzles, causing the algorithm to mislabel the color of squares.

6. PROCESS

I created the Android application in various iterations. The first step involved a low-fidelity prototype of the application, created using mockups of Android device screens, called "activities." The prototype allowed me to design each of the necessary activities and determine the layout of the application. Next, I created a "shell" of the application without most of the current functionalities. This shell allowed me to program placeholder activities and the transitions between them. It also allowed me to work with Android built-in devices, such as the camera and input method (keyboard).

Next, I investigated model applications, such as Crosswords and Sudoku Grab. The open source Sudoku puzzle app, OpenSudoku [7], served as a starting point for my application. OpenSudoku contained interactive grid activities, puzzle lists, and keyboard input methods, all of which were vital to my application. By tweaking these implementations, I was able to implement the storage and completion of crossword puzzles.

CrosScan stores serialized puzzles in a SQLite database, allowing it to read and modify existing puzzles and create new ones. The app displays the puzzles in an activity that draws the grid of black and white squares, includes a clue bar, and contains a keyboard for user input.

After completing the puzzle completion aspect of the app, I began to implement the computer vision component. My initial methods were similar to the ones described above, but with a few key differences that rendered the process less accurate. The original implementation used the Canny edge detector rather than adaptive thresholding, and it did not include a blob detection stage.

Whereas the adaptive thresholding technique detects the

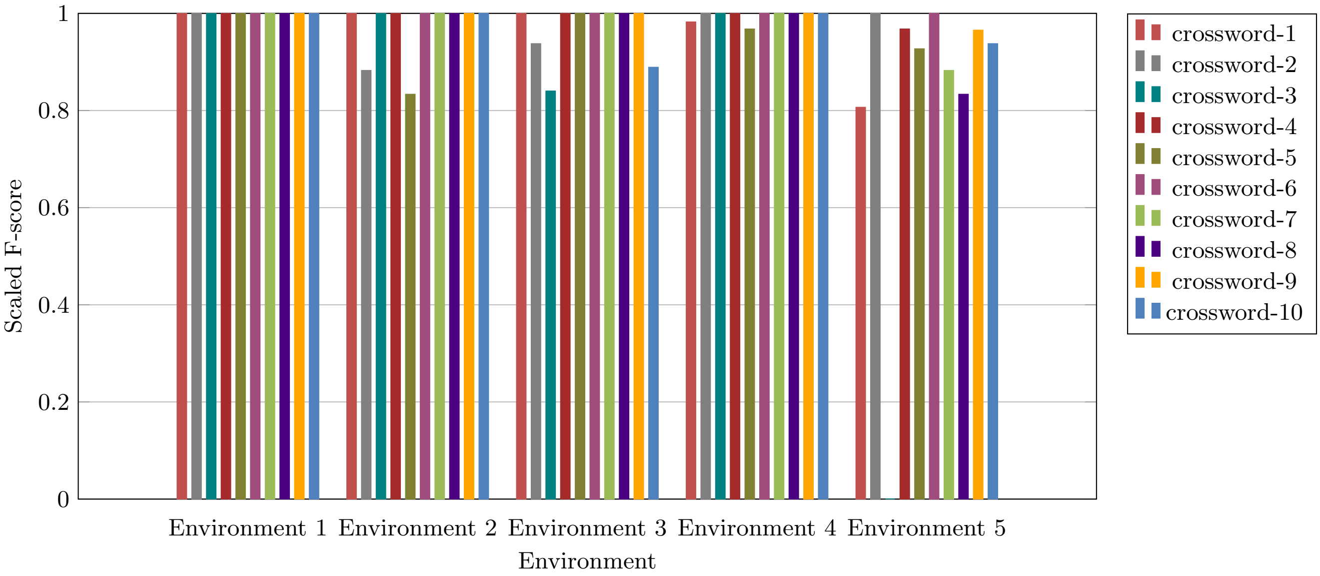

Figure 10: Scaled F-score by Environment.

whole edge of the puzzle, the Canny edge detector only found the boundaries of each edge. Because of the high resolution of the Android camera, each line on the crossword puzzle page would be multiple pixels wide. The Canny edge detector would detect the outer boundaries of each of these lines, causing the Hough line transform to detect lines on each boundary. This caused more noise than running the Hough line transform with the binary image from adaptive thresholding.

Without blob detection, the app struggled to locate the grid in the photo. Though the algorithm usually noticed a few of the grid squares, it rarely found all of them. This issue resulted in incomplete grids, as well as rows of completely white squares found outside of the puzzle grid due to noise (see Figure 11). By focusing on the portion of image containing the grid via blob extraction, the algorithm began detecting the grid more reliably and processing time improved.

7. CURRENT VERSION

The current version of CrosScan has most of the intended features implemented. Since the app has two main functions, the main activity consists of two buttons, allowing users to navigate easily between the two (see Figure 12). When the user selects the “Scan Puzzles” option, it navigates to a camera activity with a button for taking pictures. In addition, tapping the screen causes the camera to autofocus.

When the user takes a photo, it performs the image processing. The app does not yet recognize clues, so it generates default clues, such as “1a. ACROSS CLUE HERE,” and “1d. DOWN CLUE HERE” (users can obtain clues from the photo when solving puzzles). Once processing is complete, it will direct the user to the puzzle naming activity, where the user can view the scanned puzzle grid and name the puzzle (see Figure 13). If any grid square has the wrong color, the user can tap the square to toggle the color between black and white. If the grid is the incorrect size, it prompts

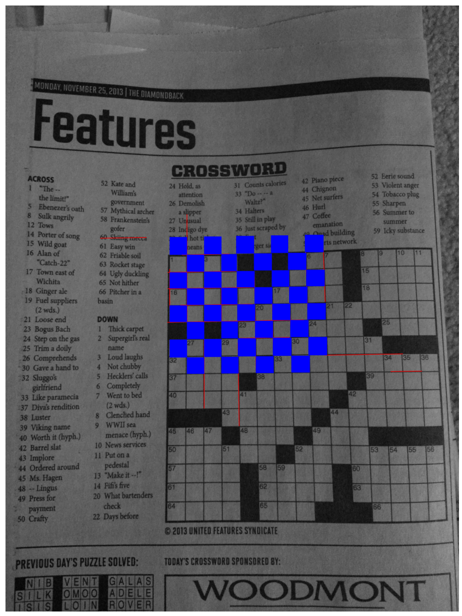

Figure 11: An incomplete grid with squares detected outside of grid boundaries, due to lack of blob detection.

the user to use the back button to return to the scanning screen.

Once the user names the puzzle, it opens the puzzle solving activity for the user to interact with the puzzle. It also adds the puzzle to the list of scanned puzzles, which the user can view in the puzzle list activity. This activity contains all puzzles that users have scanned and named (see Figure 14). By tapping the puzzle in the list, the user opens the puzzle in the solving activity. By long-pressing the puzzle, the user opens a menu with “Play Puzzle,” “Puzzle Info,”

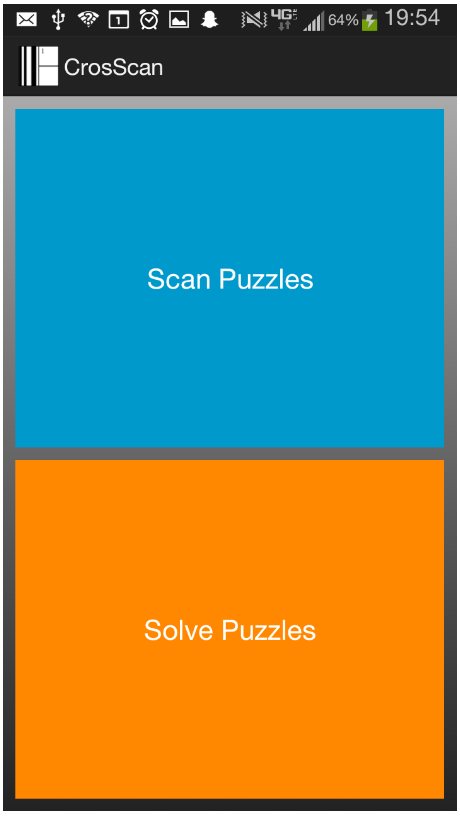

Figure 12: The main activity of the app.

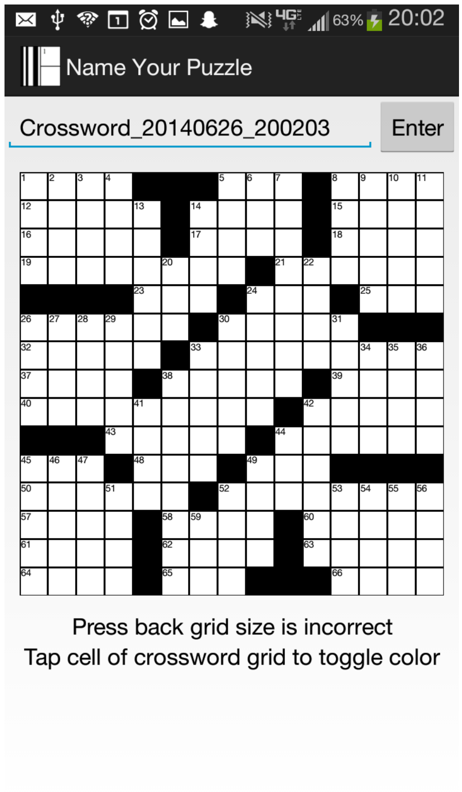

Figure 13: The puzzle naming activity.

and "Delete Puzzle" options. The first and third options are self-explanatory. The third option opens an activity that contains puzzle information, such as title, date of puzzle creation, and percentage completion (see Figure 15).

The puzzle solving activity uses the Crosswords app as a model, containing a timer, puzzle grid, clue bar, and keyboard. The activity highlights the cell selected by the user in yellow, and its corresponding Across or Down entry in blue (see Figure 16). It also allows the user to navigate between clues by tapping squares in the grid and to toggle between Across and Down by tapping the selected cell again or tapping the clue bar. The activity also skips over squares already occupied by letters and moves to the next clue once the user completes the current one. The arrows in the clue bar also allow the user to navigate between clues in the puzzle.

Users may view the original photo of the crossword puzzle by selecting the camera icon in the keyboard. This feature allows users to view the crossword clues, since the app does not yet extract clues from the image. Doing so opens an image activity that allows users to zoom and pan the image. Users may read the clues from the original photo and return to the solving activity to fill in the answer. Finally, users may delete the puzzle or restart the puzzle from the solving activity menu. Restarting the puzzle clears the squares, returns the cursor the clue, and restarts the timer.

8. USER TESTING

User testing will allow me to base the design of the project on the preferences of the target audience, crossword enthusiasts. By asking participants to complete a few tasks with the application, I hope to learn what end users expect from a crossword scanning application, which features are necessary for the application, and how to improve those features.

The study will take place on the UMCP Campus in the HCI user testing room in Hornbake. I will first give participants a summary of CrosScan, "CrosScan allows users to scan crossword puzzles into their mobile devices using the camera so they can complete the puzzles digitally. There are two main functions of the app, scanning a puzzle by taking a photo, and completing the puzzle with the device keyboard. CrosScan also allows users to edit puzzles in case they do not scan correctly." Then, I will provide an Android device with the CrosScan app and two paper crossword puzzles (labeled #1 and #2) to participants.

As the first task, I will ask participants to use the app to scan in puzzle #1, correct the puzzle if necessary, name the puzzle, and save it. When the user takes the photo, the phone will wait for processing, and then generate the correct crossword puzzle, pre-programmed into the phone. Thus, the app will generate the puzzle in a controlled manner to simulate intended app use.

For the second task, I will ask participants to begin completing puzzle #1 by navigating to the puzzle and entering a given answer to a given clue (e.g., "The answer to 59 Down

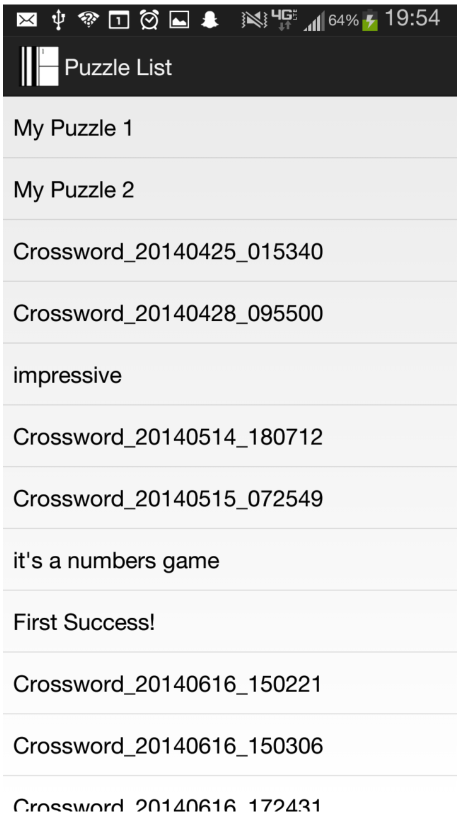

Figure 14: The puzzle list activity.

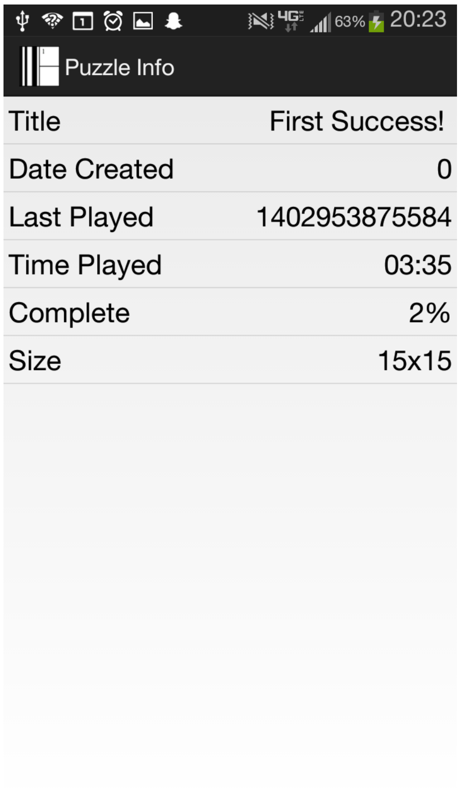

Figure 15: The puzzle info activity.

is ‘SPOTS’. Please fill in the answer.”

For the third task, I will ask participants to reset the puzzle to clear it, and then delete the puzzle so that it no longer appears in the list of scanned puzzles.

For the fourth task, I will ask participants to use the app to scan in puzzle #2, correct the puzzle if necessary, name the puzzle, and save it. When the user takes the photo, the phone will wait for processing, and then generate the incorrect crossword puzzle, pre-programmed into the phone, which I expect participants to correct. The app will generate the errors in a controlled manner for easy observation of steps taken to correct the puzzle grid.

Finally, I will ask participants to complete a survey in order to gauge their thoughts about CrosScan. After they finish, I will inform them that the app contains pre-programmed puzzles, which they did not actually scan in real time during the experiment, but the behavior should be similar in a final released version.

I will observe and store participant actions, remarks, and survey responses on a password-protected computer. I will not record participants via audio or video and the overall procedure will take no more than 30 minutes.

9. COPYRIGHT CONCERNS

Since many crossword puzzle authors copyright their puzzles, the reproduction of these puzzles in my application comes into question. In itself, my app does not violate copyright law, as it does not contain or distribute copyrighted puzzles. In addition, there are no provisions in the app that facilitate violation of the Fair Use Doctrine [1].

In order to determine whether the use of copyrighted material is a case of fair use, there are four considerations. The first consideration is “the purpose and character of the use, including whether such use is of a commercial nature or is for nonprofit educational purposes.“ Since the app will not be sold, there is no commercial nature to CrosScan. In addition, I will use crossword puzzles in my app for nonprofit educational purposes during my research. Also, there are no provisions for using the app to distribute crossword puzzles that users have scanned, restricting such puzzles to personal use.

The fourth consideration is “the effect of the use upon the potential market for or value of the copyrighted work.“ Since users of the app must first obtain a copy of the puzzle in order to use the app, the market for the copyrighted work is not affected by use of the app. Therefore, the hard copy of the puzzle does not lose its value, as a user must obtain it for the app to be useful.

10. FUTURE WORK

The main goal for the future of CrosScan is to implement optical character recognition (OCR) of the clues in a scanned puzzle. By performing OCR in conjunction with grid recognition, it will be possible to pair the two together so the user

Figure 16: The puzzle solving activity.

can browse clues and navigate the grid more easily. Having clues readily available improves the efficiency of solving the puzzle by allowing the user to avoid navigating to the image activity to view the original image and read each clue.

I also plan to continue improvement of the grid extraction algorithm. Since rotation poses a problem to the algorithm, I can improve the algorithm to transform the image based on the most common line angle so that puzzle is upright in the image.

Some additions to the user interface will also improve usability. During image processing, the app will ideally present a processing dialog with a cancel button for the user's convenience. Ideally, there will be two buttons at the bottom of the activity, one labeled "Edit Puzzle," and the other, "Rescan Puzzle." The Edit Puzzle button will lead to a puzzle editing activity that the user will be able to access anytime in the application. The Rescan Puzzle button will return the user to the scanning activity for easy access. In the puzzle editing activity, there will be options to toggle grid squares as already implemented. There will also be options for adding and removing rows and columns and for adding, removing, and modifying clues.

11. CONCLUSIONS

Considering the results of the grid extraction algorithm, CrosScan successfully extracts the grid with an average success rating of 95.30% given by the scaled F-score. However, there are certain cases, particularly involving rotation, where the grid extraction algorithm fails. In addition, user testing will allow me to target features users expect from CrosScan in order to improve the app.

12. REFERENCES

[1] Copyright Office (US). *The Copyright Law of the United States and Related Laws Contained in Title 17 of the United States Code*. U.S. Government Printing Office, 2012.

[2] Crosswords for android, June 2014.

[3] R. O. Duda and P. E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, Jan. 1972.

[4] C. Greening. iphone sudoku grab: How does it all work?, July 2009.

[5] L. He, Y. Chao, K. Suzuki, and K. Wu. Fast connected-component labeling. *Pattern Recognition*, 42(9):1977 – 1987, 2009.

[6] T.-W. Kan, C.-H. Teng, and M. Y. Chen. Qr code based augmented reality applications. In B. Furht, editor, *Handbook of Augmented Reality*, pages 339–354. Springer, 2011.

[7] R. Mašek. opensudoku-android, Mar. 2011.

[8] A. Schepman, P. Rodway, C. Beattie, and J. Lambert. An observational study of undergraduate students' adoption of (mobile) note-taking software. *Computers in Human Behavior*, 28(2):308 – 317, 2012.

[9] L. Shapiro and G. Stockman. *Computer Vision*. Prentice-Hall, Upper Saddle River, NJ, 2001.

[10] C. Ungureanu, A. Tigora, I. Bucur, and M. Zaharescu. Reversible color to grayscale conversion. *Romanian Economic Business Review*, 8(1):105–113, May 2014.

[11] R. Wischenbart. *The Global eBook Market: Current Conditions & Future Projections*. O'Reilly Media, 2013.