Tactile Graphics with a Voice

CATHERINE M. BAKER, LAUREN R. MILNE, RYAN DRAPEAU, JEFFREY SCOFIELD, CYNTHIA L. BENNETT, and RICHARD E. LADNER, University of Washington

We discuss the development of Tactile Graphics with a Voice (TGV), a system used to access label information in tactile graphics using QR codes. Blind students often rely on tactile graphics to access textbook images. Many textbook images have a large number of text labels that need to be made accessible. In order to do so, we propose TGV, which uses QR codes to replace the text, as an alternative to Braille. The codes are read with a smartphone application. We evaluated the system with a longitudinal study where 10 blind and low-vision participants completed tasks using three different modes on the smartphone application: (1) no guidance, (2) verbal guidance, and (3) finger-pointing guidance. Our results show that TGV is an effective way to access text in tactile graphics, especially for those blind users who are not fluent in Braille. We also found that preferences varied greatly across the modes, indicating that future work should support multiple modes. We expand upon the algorithms we used to implement the finger pointing, algorithms to automatically place QR codes on documents. We also discuss work we have started on creating a Google Glass version of the application.

Categories and Subject Descriptors: H.5.2 [**User Interfaces**]

General Terms: Design, Human Factors

Additional Key Words and Phrases: Access technology, blind, camera, non-visual feedback, visually impaired, tactile graphics, QR codes

ACM Reference Format:
Catherine M. Baker, Lauren R. Milne, Ryan Drapeau, Jeffrey Scofield, Cynthia L. Bennett, and Richard E. Ladner. 2016. Tactile Graphics with a Voice. ACM Trans. Access. Comput. 8, 1, Article 3 (January 2016), 22 pages.
DOI: http://dx.doi.org/10.1145/2854005

1. INTRODUCTION

From pictures of plant cells to diagrams of parabolas, images are an integral part of most textbooks and frequently convey information that cannot be understood from text alone. Therefore, these images and the text contained within them should be accessible to all students, and there is a need to create alternative access methods for people with disabilities. The common solution when making a textbook accessible for blind students

This material is based on work supported by the National Science Foundation Graduate Research Fellowship under Grant Nos. DGE-0718124 and DGE-1256082 and National Science Foundation Grant No. IIS-1116051. This work was supported by the U.S. Department of Education, Office of Special Education Programs (Cooperative Agreement #H327B100001). Opinions expressed herein are those of the authors and do not necessarily represent the position of the U.S. Department of Education or the National Science Foundation.
Authors' addresses: C. M. Baker, L. R. Milne, R. Drapeau, J. Scofield, C. L. Bennett, and R. E. Ladner, Computer Science & Engineering, University of Washington, AC101 Paul G. Allen Center, Box 352350, 185 Stevens Way, Seattle WA 98195-2350; emails: {cmbaker, milnel2, drapeau, jeffsco, bennec3, ladner}@cs.washington.edu.

DOI: http://dx.doi.org/10.1145/2854005

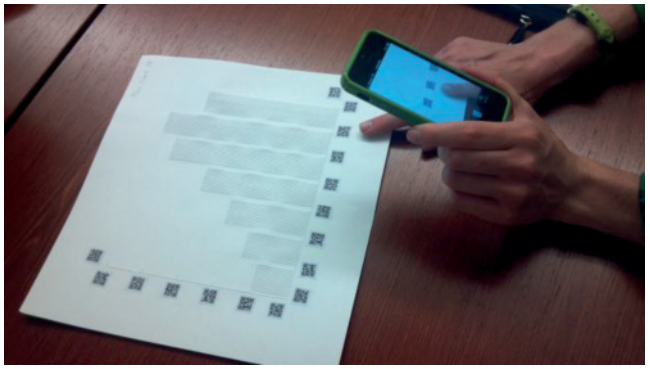

Fig. 1. The Tactile Graphics with a Voice system in use. The subject is using the Finger-Pointing mode to select which QR code to scan.

is to create tactile representations of the images, or tactile graphics. Tactile graphics can be low fidelity by using spaghetti glued on poster board or a high-fidelity graphic printed on an embossing printer. Studies have shown that tactile graphics are valuable for conveying graphical information [Holton 2013]. In a survey of 24 teachers who worked with visually impaired children, all indicated that there were situations where tactile graphics were important for and effective at teaching a lesson [Sheppard and Aldrich 2001]. Teachers also indicated that the ability to explore graphics, discover the information, and answer questions about the information independently was a fundamental part of the learning process [Rule et al. 2011; Sheppard and Aldrich 2001].

The text in tactile graphics is typically represented using embossed Braille. However, a 2009 report by the National Federation of the Blind states that less than 40% of the functionally blind population in the United States is fluent in Braille [National Federation of the Blind Jernigan Institute 2009]. Therefore, tactile graphics with Braille labels are not accessible to a significant number of blind people.

There have been a few solutions to this problem presented by the access technology community. Examples include a system where an overlaid tactile graphic on a tablet gives audio feedback when touched [Landau et al. 2004] and a talking pen to explore a tactile graphic [Landau and Neile 2010]. However, these solutions require using specialized devices, which can be expensive.

We present and discuss the development of a new system for embedding and accessing text in tactile graphics using QR codes, which are small codes that directly encode textual information (Figure 1). QR codes can be read by a smartphone and can easily be created by anyone with access to a computer. We created a smartphone application for blind users called Tactile Graphics with a Voice (TGV) that scans QR codes and provides feedback to help users aim the smartphone camera. We conducted interviews and surveys with people who are blind or have low vision to design the application and determine what types of nonvisual feedback are most helpful to aim the smartphone. In addition, we developed a finger-pointing method to help determine which QR code should be read when there are multiple QR codes in the camera view.

We evaluated our application in a longitudinal study and found that people who are blind or have low vision were able to successfully answer questions about tactile graphics by scanning QR codes. Key findings from the study are listed here:

(1) Four of our participants were able to correctly answer questions about the images using the QR codes but were not able to use the Braille equivalents as they were not fluent in Braille.

(2) Participants fluent in Braille spent an equivalent amount of time on tasks and had similar accuracy for both the QR codes and Braille equivalents.
(3) Preferences varied greatly among participants as to what kind of feedback from the smartphone application is most helpful. Four of our participants preferred the Silent mode, four preferred the Finger-Pointing mode, and two preferred the Verbal mode.

Based on feedback from the study, we have begun work on TGV for Google Glass, a hands-free version of the application. Our contributions are as follows:

(1) We discuss the development of TGV, a system to access text on tactile graphics using QR codes and a smartphone application.
(2) We present the findings from our study, which show that users who are blind or have low vision support having a variety of nonvisual feedback mechanisms to help aim a camera.
(3) We discuss how to use our findings to develop another iteration of the application using Google Glass.

1.1. Relationship to Conference Paper

This is an extended version of our conference paper [Baker et al. 2014], which appeared at *ASSETS 2014*. In this version, we have added additional sections regarding the integration of Tactile Graphics with a Voice with the Tactile Graphics Assistant [Jayant et al. 2007], additional details regarding the finger-pointing algorithms used in TGV, additional analysis of some results from the user study, and preliminary details about implementing TGV on Google Glass. Additionally, many of the figures have been improved.

2. RELATED WORK

We discuss prior work related to three areas of our system: (1) methods to embed textual information on tactile graphics, (2) methods to access the information, and (3) use of the finger-pointing technique as a means to select which information to be read aloud.

2.1. Accessing Textual Information on Tactile Graphics

Braille labels on educational tactile graphics present difficulties for both students and teachers. Sheppard and Aldrich [2001] found that both students and teachers had difficulties with Braille labels on tactile graphics in the classroom. Teachers, in particular, had issues placing the labels without text overlapping the figure. Students struggled with the meaning of a label when it stretched across the entire graphic.

Despite the issues with using only Braille for accessing text, there is little work in the HCI literature using alternative methods. There has been some progress made in the access technology community. Touch Graphics has developed the Talking Tactile Tablet (TTT) [Landau et al. 2004], a touch-sensitive tablet on which a user can place a tactile graphic and hear audio information upon touch. However, this method requires a large touch-sensitive surface ($\sim$12 $\times$ 15 inches, 6.5 pounds) and has to be connected to a computer via USB, which contains the information for the tactile graphic to be explored. Touch Graphics also created the Talking Tactile Pen (TTP) [Landau and Neile 2010], which allows blind users to access information on custom tactile graphics tagged with a proprietary code. The pen contains a small camera used to photograph the proprietary codes. When the pen contacts a tagged area, it reads aloud the corresponding file stored on the pen. Despite the pen's portability in comparison to the TTT, it is a specialized device and is only useful on properly tagged tactile graphics that have their information stored on the pen. TGV is a solution that attempts to solve the same problem by

using nonproprietary codes and a nonspecialized, portable, mainstream device like the smartphone.

Voiceye codes are also being used to encode text on graphics [Holton 2013]. While not used on tactile graphics, they are used in South Korea to make government forms accessible. These are similar to QR codes but may contain more information for a given area. Users scan these codes with a smartphone application and the corresponding text appears for reading aloud or visual magnification. However, users are not given feedback to assist in scanning the code and the codes must be created with expensive proprietary software. TGV provides a major benefit over current approaches because QR codes can be freely created.

2.2. Camera Use by Blind People

TGV requires the use of a smartphone camera, because it enables the use of QR codes. While aiming the smartphone is a challenging task for blind people, there are research efforts in the accessibility community to tackle this problem. Bigham et al. [2010] created an application called VizWiz::LocateIt, which allows blind users to locate objects using the camera on their smartphone. VizWiz::LocateIt uses crowdsourcing to identify the object in the photo and computer vision techniques to provide audio feedback about the proximity to the object. Our application uses similar audio feedback to guide users to the QR code but does not rely on crowdsourcing, thus providing quicker feedback.

Using computer vision techniques exclusively with a smartphone camera may enhance camera feedback. Jayant et al. [2007] created EasySnap, a camera application that assists users in taking pictures by providing audio feedback. EasySnap uses computer vision to locate people or objects in the viewfinder and relays information about their location and their size in proportion to the viewfinder. They followed with another application, PortraitFramer, which incorporates features of EasySnap and uses haptic feedback to communicate where in the viewfinder the people or objects are located. They found that it took little training for users to take better pictures. A similar feature has been built into the camera application on recent versions of iOS [Goransson 2011]. When text-to-speech is enabled, the camera application provides feedback about faces, such as "face at top of screen," to guide users in taking portraits. Our application also incorporates audio feedback, but because users are using their sense of touch to explore a tactile graphic, we decided not to use haptic feedback to avoid cognitive overload.

TGV utilizes audio feedback, but there are diverse options, such as tone and speech. Vasquez and Steinfeld [2012] were interested in learning what type of audio feedback was preferred among blind people using a camera. They considered speech, tone, and no feedback. People strongly preferred speech feedback and found it easier to use than either silent or tone feedback. As a result, we use speech feedback in TGV as opposed to tone.

The majority of the camera applications mentioned previously were focused on taking a quality picture of a person or a physical object. Another related space is in technology that allows blind users to scan barcodes, which are similar to QR codes. The majority of commercial applications, such as the i.d. mate[1] or Digit-Eyes,[2] do not provide feedback. However, Tekin and Coughlin [2010] experimented with different feedback modalities to help blind users scan barcodes on products. They used both verbal feedback and sonification, but their application was evaluated by a single user. TGV distinguishes itself in two ways: (1) our QR codes are labels that can be located by touch, and (2) multiple QR codes can be close together.

[1]http://www.envisionamerica.com/products/idmate/.
[2]http://www.digit-eyes.com/.

2.3. Finger Pointing

Because textbook images may have multiple text labels in close proximity of one another, the use of a finger may help select the preferred QR code when multiple codes are in the viewfinder. Thus, we present related work on the practicality of finger pointing as a method to select a preferred QR code.

There are numerous projects that use finger pointing to identify an object or information of interest. One example is the EyeRing [Nanayakkara et al. 2013], a camera worn on the finger that reads information aloud based on where the finger is pointing. Similarly, OrCam[3] also uses finger pointing for people who have low vision. The OrCam is a wearable camera that uses computer vision to identify objects to which a user is pointing and reads aloud information about that object. The manufacturers envision that OrCam can recognize faces, places, objects, and text.

Kane et al. [2013] developed Access Lens, a way for people who are blind or have low vision to access documents. This system uses a camera connected to a computer to read aloud the text on documents. Users can point to any element on the document to hear the associated information. This system brings promise to the accessibility of printed documents and demonstrates that finger pointing is an easy way for blind users to control what information they hear. However, Access Lens is not portable. In TGV, we capitalize on finger pointing as a simple means of selecting the information the user wants to hear.

3. FORMATIVE STUDIES

In order to determine the feasibility of substituting QR codes for text labels on tactile graphics, we conducted a survey and follow-up interviews with people who are blind or have low vision. We were motivated to learn about the current use of tactile graphics and cameras, and whether people would take interest in using QR codes as labels on tactile graphics.

The online survey was distributed to mailing lists for people who are blind or have low vision and inquired about their use of Braille, tactile graphics, and camera applications on the smartphone. Twenty-two people completed our survey, where 15 of our respondents were blind and seven had low vision. There were 12 females and 10 males with an average age of 38.18 (SD = 13.46). All of our respondents had taken some college courses, and nine respondents had a graduate degree. Sixteen respondents knew Grade 2 Braille, while only three respondents had little to no knowledge of Braille. All but one of the respondents owned a smartphone.

We conducted follow-up interviews with 10 of the survey respondents, six of them female. We selected a diverse subset of those who indicated they would be willing to be interviewed on the survey. The interviews provided more detail about their survey responses and provided feedback about our proposed system, TGV. The participants' ages ranged from 21 to 67, with an average age of 37.6 (SD = 13.95). Five participants identified as blind and five as having low vision. Five participants used Braille at work, three knew Braille but did not use it often, and two had little familiarity with Braille. Eight participants used tactile graphics in their education and work.

We found that many of our respondents frequently used cameras, especially on smartphones, and were interested in using tactile graphics with QR code labels. Seven of the 10 participants reacted positively to replacing Braille with QR codes. One participant noted: "You can fit a lot more information on a QR code than on a Braille label," a sentiment shared by five of our participants.

[3]http://www.orcam.com.

In addition, many of our survey respondents were familiar with using the camera on their smartphone, and thus have completed similar tasks to scanning QR codes. Fifteen respondents used an application that required the camera on a daily to weekly basis.

We learned that people found nonvisual feedback for aiming the camera to be helpful. Just over half the respondents used an application that gave them feedback to help aim the camera. The majority of those respondents indicated that the feedback was helpful, with only one respondent mentioning that he had received feedback that was not helpful as it was unclear what it meant.

In our follow-up interviews, we investigated preferences for feedback modalities on a smartphone camera application: verbal, tonal, haptic, and no feedback. While the participants had a variety of preferences, we found that most participants preferred having the option of a quiet mode. Participants wanted a quiet mode because they felt that expert users needed less feedback, and they would not want to disturb others, such as during a meeting.

4. TACTILE GRAPHICS WITH A VOICE

The first iteration of TGV is composed of tactile graphics with QR code labels and a smartphone application. The application provides multiple nonvisual feedback modalities and allows users to select which QR code they want to scan.

4.1. Tactile Graphics with QR Codes

The creation of tactile graphics for TGV requires a similar amount of work as traditional tactile graphics. Traditionally, converting a textbook graphic into a tactile graphic is a labor-intensive process. First, the text must be removed from the graphic. Generally, extra processing is needed or the image may need to be completely redesigned to make it understandable in a tactile form. Once the text is removed, it needs to be translated into Braille and be placed back on the image in a similar location to the original text. For TGV, instead of generating Braille labels, we created a QR code from text using a free online generator. We chose to have the label information directly encoded into the QR code, instead of a link to a database, so that users would not need an Internet connection. Because of this, the size of the QR code reflects the amount of text on the label. Because the embosser we used to create the tactile graphics cannot print ink, we printed QR codes on a separate sheet of paper and glued them onto the graphic (see Figures 1 and 5 for examples). It was not necessary to mark the QR codes with an embossed symbol because the height difference of the QR codes was sufficient to be felt, and the labels were easy to locate while exploring the tactile graphic. If you have an embosser capable of both embossing Braille and printing ink, the only difference from the traditional process is that you would place the QR code labels (with accompanying tactile markers) on the graphic in place of the Braille labels.

4.2. Smartphone Application

We created an accessible application for iOS that allows a user to scan a QR code easily, even if there are multiple QR codes close together. We designed the application to be used to scan a single QR code at a time. As users are exploring the graphic, they may come across a label that they would like to know what it says. At this point, they would pick up the phone and aim it in the direction of the QR code they would like to scan. After the QR code has been scanned, it will be read aloud and the user can either resume exploring the tactile graphic or scan another QR code.

The smartphone application is built on top of the ZXing software for scanning QR codes. This software identifies QR codes by looking for an area of black-and-white

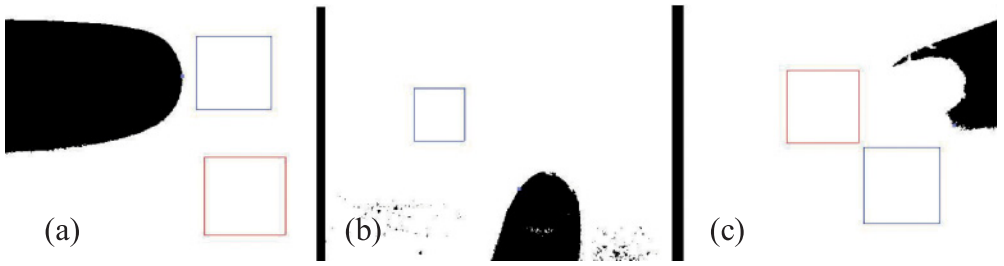

Fig. 2. Images captured by our application. They are thresholded for finger detection and have boxes around the QR codes detected by our algorithm. The blue box shows the QR code that was selected by the algorithm and whose label was read aloud. (a) A success case in which the algorithm identified the QR code at which the user was pointing. (b, c) Failure cases. In (b), the intended QR code was not found by the software (a box is missing above the finger), and in (c), the algorithm selected the incorrect QR code (blue) even though it had found the correct QR code (red), because the finger detection did not detect the fingernail of the user.

variation. We added verbal feedback to help users scan QR codes, as well as the option to use finger pointing to allow the selection of a QR code when many are visible in the viewfinder.

Based on our survey and interviews, we integrated feedback for aiming the camera. In addition, we determined that it was important to have a feedback mode and a silent mode. Because the participants' preferences on feedback modalities varied, we used verbal feedback based on prior work [Vásquez and Steinfeld 2012] and because most of our interview respondents indicated that verbal feedback was the easiest to learn.

We presented short, clear verbal feedback to assist a user in moving the phone. We based the feedback on the screen location of the QR code, based on related work [Vásquez and Steinfeld 2012]. For instance, if participants hold a smartphone in a nontraditional way (e.g., sideways), they will still hear relevant feedback because the navigational instructions to a QR code are based on the current phone orientation.

When multiple QR codes are visible, it is necessary to determine which QR code should be scanned. Therefore, we implemented finger pointing as a method to distinguish which label should be scanned. The selected QR code is the one with the shortest distance to the user's finger. To prevent the application from scanning the incorrect QR code, we set a maximum distance in which a finger can choose a QR code to scan. Unlike Kane et al. [2013], which selects the information at the tip of the finger, our application selects the QR code that is closest to any part of the finger. As a result, users needed to be aware of their hand placement to ensure a false positive does not occur. We identify the finger with color-based skin detection [Elgammal et al. 2009; Phung et al. 2005]. Because of the constrained black-and-white environment of tactile graphics, we can identify a painted fingernail by looking for colored pixels and group them as part of the finger.

Although our algorithm is generally successful, in testing it we have identified two main failure cases, as shown in Figure 2. In these images, the QR codes discovered by the software have a box around them. The first failure case is when the QR code scanning software does not identify a QR code that is visible. This can be seen in Figure 2(b), where a QR code is not identified (there should be a box at the tip of the finger), so the system will report back the nearest QR code as long as it is within a certain distance of the finger. The second failure case occurs when the fingernail is not detected and the person points at an angle to a QR code in a row of QR codes (Figure 2(c)). Both failure cases can be remedied by detecting the direction of pointing, which we have done in the second iteration of TGV on Google Glass.

4.3. Feedback Modalities

Because feedback and finger pointing are not appropriate in every situation, we created three modes for the application: Silent, Verbal, and Finger Pointing.

(1) *Silent mode* gives no feedback to help aim the camera. If multiple QR codes are visible in the viewfinder, the application does not scan. When it has successfully scanned a QR code, it chimes and then reads the scan aloud.
(2) *Verbal mode* provides spoken feedback to help aim the camera. As it does not matter where the QR code is on the screen, the spoken feedback relates what is visible on the screen. It will say the number of QR codes that are visible on the screen, saying either "zero," "one," "two," or "many." If only one partial QR code is visible, then the feedback will relate its location on the screen, saying either "top," "bottom," "left," or "right." If multiple QR codes are visible in the viewfinder, the application speaks this information and will not scan as there is no way to know which QR code the user intends to scan. When the application has successfully scanned a QR code, it chimes and then reads the scan aloud.
(3) *Finger Pointing mode* provides spoken feedback to help aim the camera. The application needs to detect the finger in order to scan. If the finger is not detected, the application says "no finger" and does not scan. It then looks for a QR code. If multiple QR codes are visible, the application will scan the QR code closest to the finger. When no QR code is found, it says "zero." When the application has successfully scanned a QR code, it chimes and then reads the scan aloud.

5. INTEGRATING WITH THE TACTILE GRAPHICS ASSISTANT

The process described for creating the tactile graphics is laborious. However, if the creators have access to printers that can do both embossing and ink, it may be easier, as we have done some work on integrating with the Tactile Graphics Assistant (TGA) [Jayant et al. 2007], which seeks to automate as much of the process of creating tactile graphics as possible. The current process to create tactile graphics works to automate all five steps of the process: (1) cleaning up the image, (2) identifying and removing the text, (3) resizing the image to fill the Braille paper (generally 10×10 or 11×11 inches), (4) translating the text into Braille, and (5) replacing the text with Braille in the new tactile image. Our new process replaces steps 4 and 5 with automated placement of QR codes.

Jayant et al. [2007] found that most time consuming of the nonautomated tasks was editing the placement of the Braille in the newly resized tactile images; therefore, we are interested in using a heuristic algorithm to automate the process and reduce the amount of human editing needed. We use a greedy algorithm to place the QR codes as near as possible to the original text location without overlapping each other or any graphical elements. We also compared output figures using various evaluation functions.

As input the placement algorithms took (1) a resized image from which the text had been removed, (2) a text file containing all the text that had been removed, and (3) an XML file that had information about the text (including alignment and original placement) and both the x- and y-scale factors for the resizing of the image as TGA allows you to select whether you would like to preserve the aspect ratio of the image or resize the image so it will take up the entire page.

The goal for the algorithm was to create a good placement of the QR labels, meaning (1) the labels should not overlap the image if possible, and (2) the label placement should be near to the placement of the original text labels in the image, as, presumably, that is the label placement that makes the most sense for the image.

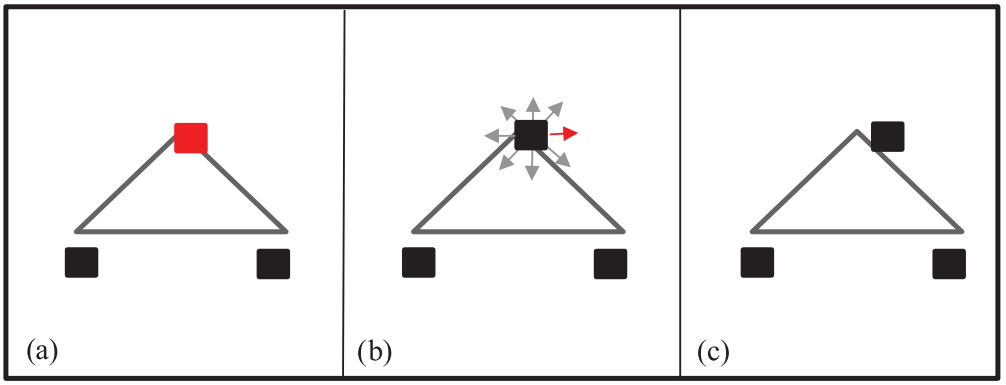

Fig. 3. The QR label placement algorithm (a) selects the QR label with the worst placement (highest score), highlighted in red; (b) attempts to move it 10 pixels in eight directions; (c) chooses the direction that gives the best placement (lowest score), and repeats until all label scores are within a threshold or the maximum number of moves has been reached.

The x- and y-coordinates of both the upper left and lower right corners of the original text labels are recorded during the initial processing of the graphic by TGA. To determine the initial placement of the QR labels, we multiplied the x- and y-coordinates of one of the corners of the original location of the text by the x- and y-scale factors. The dimensions of the QR labels were significantly different than those of the original text, so which corner we used depended on the alignment of the original text: if the text was left-aligned, the initial placement used the top left corner; if it was right-aligned, the top right corner was used; and if it was centered, the top center was used. If any part of the QR label was initially placed off of the image, it was moved until its edge was along the outside of the image. This initial placement was used as the starting point for all the algorithms.

For the greedy algorithm, we used a hill climbing search approach [Russell and Norvig 2009]. The QR labels were placed in a priority queue ranked by their evaluation score, with the highest score first. The algorithm, shown in Figure 3, removed the worst-placed QR label (the one that had the highest evaluation score) from the queue and evaluated moving it in eight directions (up, down, left, right, and along the diagonals) by 10 pixels. Along the diagonals, the label was moved 10 pixels along both the x- and y-axes. The algorithm picked the direction that improved the evaluation score of that QR label by the most. If movement only worsened the evaluation score, the QR label was put aside and rejoined the priority queue only after another QR label was moved. To prevent thrashing, we also limited the number of times a QR label could be moved to 50.

We explored a number of functions to evaluate the placement of a QR label. The functions were all of the form $score = \sum w_i f_i$, where the w_i's are weights and the f_i's are numeric features that we identified as possibly being important in the evaluation function. The features included in the evaluation functions were (1) distance from original placement (along both the x- and y-axes), (2) overlap over other QR labels, and (3) overlap over other features in the image. In determining the overlap over other features in the image, there were two related features: one computed the number of nonbackground image pixels that were eclipsed by the label, and the other computed the number of nonbackground image pixels weighted by the location of each pixel. Pixels near the center of the QR label had a higher weight than those on the outside. We used this weighting to move the QR labels off of image features that were larger than 10 pixels, where moving the QR label 10 pixels may provide no improvement

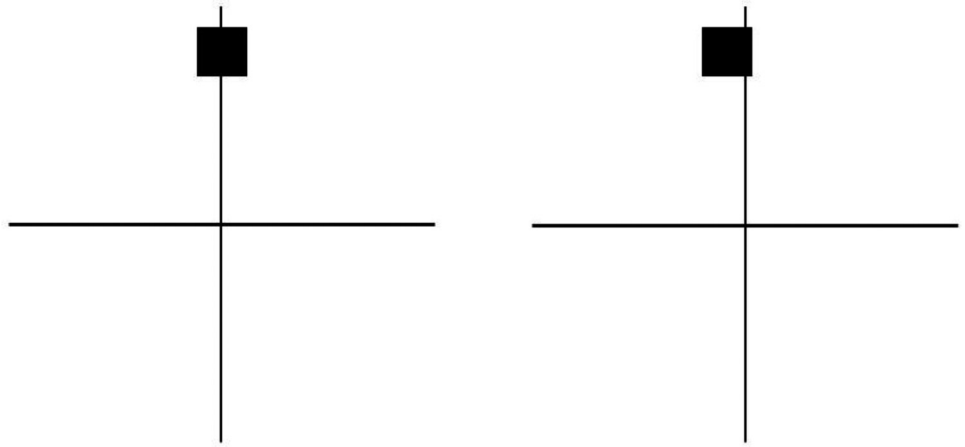

Fig. 4. This figure shows an example of how the weighted overlap feature may be helpful. The left image is the original placement and the right image is after a move. Both images have the same number of pixels overlapped, but the weighted overlap scores would be different with the right image having a lower score.

in terms of the number of pixels overlapped, but moving the QR label 20 pixels may improve the overall placement. An example of this can be seen in Figure 4, where the total number of pixels eclipsed is the same, but the weighted overlap is different and makes a difference. The left image is the original placement of the label and the right image is a potential move of the label. If you are only using the overlap feature, then the score will not improve and the move will not happen. However, with the weighted overlap, the move is beneficial. That move makes it possible for the one more move to happen, moving the label completely off the image.

We tested the placement algorithms using a number of different weights for the features on images acquired from a precalculus textbook [Gordon-Holliday et al. 1999] previously used for the TGA. The images had already been digitized and classified into different categories. We wanted to make sure we tested our algorithms on a variety of types of images to make sure that we were not overfitting our evaluation functions to one type of image. Therefore, we selected our images from two datasets: complex images on which it might be difficult to place the QR labels and images with clean lines that were mostly graphs, a very common image type in the textbook. In order to get a representative sample, we selected every third image from these two sets. After using these two datasets, we had 27 figures to use in our evaluation.

We had one of the authors (a tactile graphics expert) who did not participate in the placement of the QR labels evaluate the label placement. We used his feedback to assess the effectiveness of three different evaluation functions, which each weighted the features differently.

For each of the 27 figures, we created a task sheet for the expert to reference. We placed the original image from the textbook at the top of the page. This image included the English text so the expert could tell what text each QR label contained. Below the original image were the four images containing different QR label placements, one for each evaluation function we tried as well as one for the initial placement of the QR labels. The four images were placed in a randomized order and given labels A, B, C, and D so the expert was unaware which image went with which placement type to prevent bias.

In order to determine the amount of saved human labor, we used two different metrics. The first metric we looked at was what percentage of images required fewer

QR labels to be moved than would be moved using the initial placement. Ideally, we would like to have every image require fewer QR labels to be moved or at least none to require more labels to be moved. We found that based on the expert evaluation, all the images had the same or a fewer number of QR labels that needed to be moved, and 77.7% of the time the images required fewer QR labels to be moved.

Once we knew that the majority of the images did not require more QR label moves, we wanted to see what the effect was on the total amount of work that needed to be done. Therefore, we used the metric of the average number of QR labels that need to be moved per image. We found that the total amount of work also decreased when compared to the initial placement. When considering all the images, the number of QR labels that needed to be moved decreased slightly, but the decrease was much more dramatic when looking just at the images deemed solvable. Some images were deemed unsolvable, as with the current constraints of the image size, it was not possible to fit all the QR labels. This was most common with images that had axes with a large number of labels along the side. Either the image would need to be larger (or have more buffer around it so that the labels could be staggered) or some QR codes removed. If we look at only the solvable images using the best of the three evaluation functions, the number of QR labels that need to be moved went from an average of 1.9 in the initial placement to 0.3 per image in the placement provided by this algorithm.

This study of QR code label placement algorithms demonstrates that some human time in placing QR codes can be reduced by relatively simple heuristic algorithms. We explored three possible algorithms, but better ones may be possible.

6. LONGITUDINAL STUDY

To evaluate the efficacy of TGV, we conducted a six-session longitudinal study with 10 participants who either were blind or had low vision. Participants answered questions using TGV with the three modes of feedback (Silent, Verbal, and Finger Pointing). In the last session, we had participants who knew Braille complete the same tasks using tactile graphics with Braille labels.

6.1. Participants

We conducted the study with 10 participants (four male, six female), with ages ranging from 30 to 54 years, and an average age of 41.9 (SD = 8.1). Five had college degrees, three had some college education, and two had a high school education. Four participants identified as having low vision and the remaining six identified as blind. Six participants completed the Braille portion of the study, while four were not Braille literate or were not confident in their Braille skills. Overall, participants did not have much experience with tactile graphics, with five never using them, three rarely using them, one using them once per month, and one using them once per week. Nine participants had smartphones; seven had iPhones and two had Androids. Our smartphone users had used camera applications for varying frequencies: two used them daily, two weekly, two monthly, and three rarely. Finally, eight participants had no experience scanning QR codes with their smartphones and two had some experience.

6.2. Apparatus

The TGV application was run on an iPod Touch 4th generation and an iPhone 5, each running iOS 6. Each participant used the same device for all six sessions. The tactile graphics were printed on standard 11×11.5-inch Braille paper and embossed with a Tiger embosser.[4] QR codes were printed on standard printer paper and cut and pasted onto the tactile graphics in the appropriate places. Braille labels were embossed directly

[4]http://www.viewplus.com/.

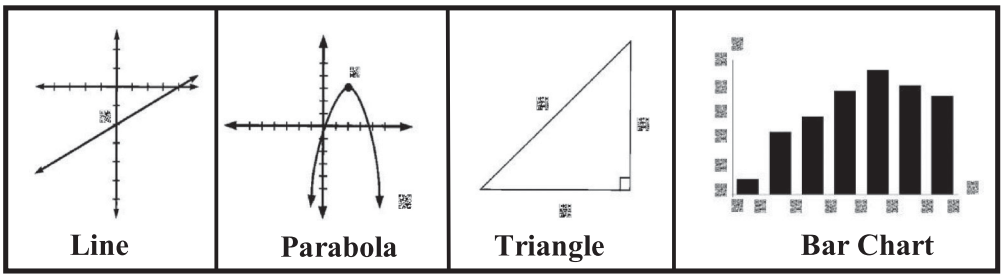

Fig. 5. This is an example of each of the tasks that the participants completed. The first task is to find the y-intercept of a line. The second task is to find the (x,y)-coordinates of the vertex of a parabola. The third task is to find the length of the hypotenuse of a right triangle. The fourth task is to find the range of the tallest bar on the bar chart. In each session, participants used similar graphics, but with different labels (e.g. the parabola might be the opposite direction and have a different vertex).

on the graphic using Nemeth code, the type of Braille usually found in math textbooks. Numbers in Nemeth code and Grade 1 and 2 Braille are similar; in Nemeth code the dots are shifted down a row [Nemeth Braille].

6.3. Procedure

We had each participant complete six sessions over a 2-week period. We wanted participants to interact with the application over time to emulate a real-world situation, such as using the application to complete schoolwork.

During the first session, we collected demographic information from the participants and taught participants how to use the three modes of the TGV application (Silent, Verbal, and Finger Pointing). We explained how each mode worked and provided basic information about using the application, such as the suggested scanning height and where the camera was physically located on the device. Participants had a chance to practice scanning a QR code with each mode.

During each session, participants completed a total of 12 tasks by using each mode of TGV (Silent, Verbal, and Finger Pointing) on the following four tasks (Figure 5):

(1) *Line.* The first task was to find the y-intercept on a line graph. The graphics always had one QR code representing the value of the intercept.
(2) *Parabola.* The next task was to find the (x.y)-coordinates of a parabola vertex. The graphics used in this task always had two QR codes, one for the coordinates of the vertex and one for the equation of the parabola.
(3) *Triangle.* The third task was to find the length of the hypotenuse of a right triangle. The graphics in this task always had three QR codes, as the lengths of all sides were labeled.
(4) *Bar Chart.* The final task was to find the range of the tallest bar in a bar chart (the QR codes on the left and right edges of the bar). For this task, the bar chart had seven bars, and there was a QR code marking the bounds of each bar on the x-axis and each tick mark on the y-axis as well as axes labels.

While the type of task remained the same for each session/mode, the images were slightly different each time. For instance, the parabola may be facing up or down, which bar is the tallest may change, and so forth. This prevented the participants from learning exactly where the QR code was as it was in a slightly different place in each version of the task. The images for each task were based on images taken from a precalculus textbook [Gordon-Holliday 1999]. At the beginning of each task, a tactile graphic was placed in front of participants, and they were instructed to begin.

The task ended when participants responded with their answer. A researcher recorded the task completion time and their answer. We video recorded participants to validate this data. For the first three tasks (Line, Parabola, and Triangle), participants can only receive 0% or 100% accuracy. On the final task (Bar Chart), participants can also receive 50% accuracy, as that task required finding both the left and right values of a range. Participants were not told whether or not their answers were correct to mimic a testing situation. We randomized the order of modes used in each session but kept the order of tasks consistent: Line, Parabola, Triangle, and Bar Chart. At the end of each session, we conducted a survey to gauge the participants' preferences for the feedback modes.

In the last session, participants who were proficient in Braille attempted to complete the same tasks using Braille labels in lieu of the QR codes. We choose to have the comparison to Braille only in the last session for two reasons. The first is that participants were already familiar with Braille, so we felt that they did not need the time to learn it. By completing the sessions with TGV, they would be familiar with the tasks by the sixth session, making the comparison from TGV to Braille more equal. The second is that we wanted to limit the length of the sessions to prevent fatigue. As some of our participants were Braille literate but not familiar with Nemeth code, we explained the difference between Nemeth and Braille to those participants.

7. DESIGN AND ANALYSIS

The study was a 6×3 within-subjects design with factors for *Session* and *Mode*. The levels of *Session* were (1–6); the levels for *Mode* were (Silent, Verbal, Finger Pointing). Each participant completed a total of 72 trials, for a total of 720 trials with the smartphone, and six participants additionally performed four trials with Braille at the end of the session. The other four participants were not comfortable enough with Braille to attempt those tasks. We measured completion time and accuracy for each task. If participants took longer than 180 seconds to complete a task, we stopped them and recorded that they had timed out on the task. Participants were still allowed to submit an answer if they timed out on the task.

While analyzing completion time for a task, we used a mixed-effects model analysis of variance with fixed effects of *Session* and *Mode*, with *Participant* modeled as a random effect. For accuracy and preference data, we looked at the descriptive statistics.

8. RESULTS

8.1. Accuracy

The accuracy for each task did not vary across the different modes (Silent mode: 88%, Verbal mode: 88%, Finger Pointing mode: 89%). Mode did not have a significant effect on the accuracy (mode: $F_{2,\,660} = 1.752$, $p = .202$); however, there was a trend that session had an effect on accuracy ($F_{5,\,660} = 3.947$, $p = .0782$). Figure 6 shows the change in accuracy of the bar chart task. As can be seen from Figure 6, participants appeared to improve in accuracy between Sessions 1 and 2, with little to no improvement over the next four sessions, and a decrease in the accuracy during Session 5. We believe this dip and the lack of a significant learning effect was due to the small sample size and the variation of tasks: although we attempted to create equally difficult tasks in all the sessions, the graphics used in Session 5 may have been more difficult than those in other sessions. We also saw that the accuracy tended to be lower on the Bar Chart task than the other tasks (Table I). We hypothesize that this was the case because this task had increased complexity and the most QR codes closest together.

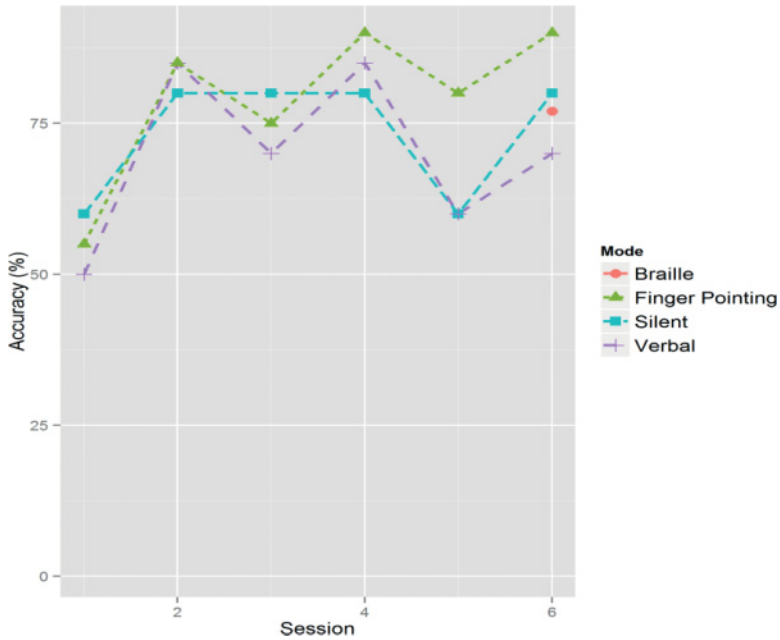

Fig. 6. A comparison of the average accuracy for the Bar Chart task across the six sessions for the three modes (n = 10) and Braille (n = 6) on the last session. Participants were asked to find the range of the tallest bar and their answer could be 0%, 50%, or 100% correct.

Table I. Comparison of Accuracy on Different Tasks with Different Feedback Modalities Across All Sessions for All Participants

	Silent	Verbal	Finger Pointing
Line	97%	97%	93%
Parabola	93%	95%	95%
Triangle	88%	88%	90%
Bar Chart	73%	70%	79%
All Tasks	88%	88%	89%

8.2. Time

If participants reached 180 seconds without answering the question and completing the task, this was counted as a time-out, and the time is not included in the average or the statistical analysis. Out of 720 tasks, the total number of tasks that timed out was 41, or 5.7%, and almost half of those time-outs (19) occurred in the first session. Additionally, over half of the time-outs (21) occurred during the difficult Bar Chart task. The time-outs occurred in all the modes, with 16 time-outs occurring in the Finger-Pointing mode, 16 occurring in the Silent mode, and nine occurring in the Verbal mode.

With time-outs removed, the average QR code task completion time for all sessions and all modes was 40.9 seconds (SD = 36.3). However, the participants were faster in the sixth session than in the first (see Figure 7) and the effect of *Session* on time was statistically significant ($F_{5, 640} = 2.268$, $p < .05$). In Session 6, the average Silent mode completion time was 25.3 seconds (SD = 18.7), Verbal mode completion time was 30.5 seconds (SD = 29.8), and Finger-Pointing mode completion time was 40.3 seconds (SD = 29.4). The effect of *Mode* on time was not statistically significant ($F_{2, 640} = 0.619$, $p = .5391$).

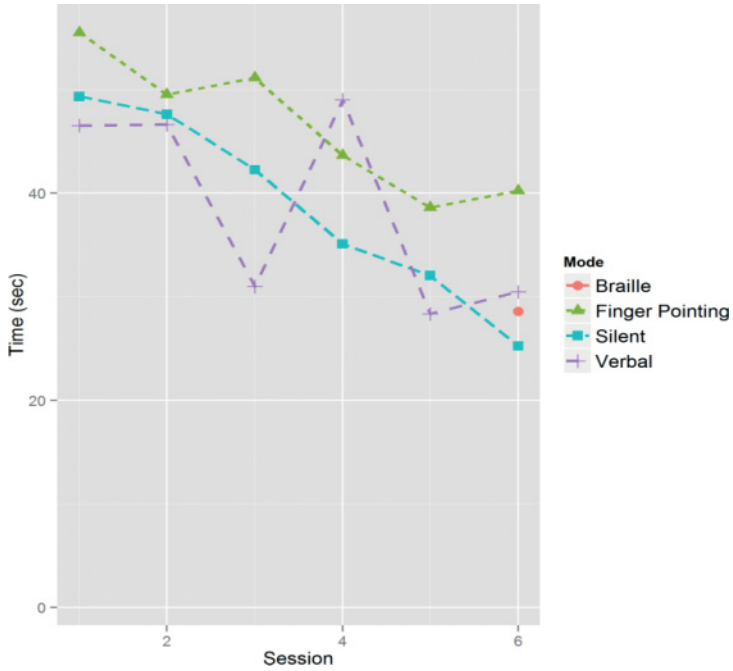

Fig. 7. A comparison of the average time for each participant to give the answer for a task for the three modes (n = 10) across the six sessions as well as for Braille (n = 6) on the final session.

Table II. Feedback Modality Preferences of Users Averaged Across All Sessions on 7-Point Scale, Where a 7 Indicates That They Did Not Like the Mode or Found It Unhelpful or Difficult to Use

	Silent	Verbal	Finger Pointing
Liked	3.98 (SD = 2.11)	2.87 (SD = 1.57)	3.63 (SD = 1.72)
Helpfulness	4.13 (SD = 2.05)	2.85 (SD = 1.38)	3.33 (SD = 1.71)
Ease of Use	3.53 (SD = 2.16)	2.68 (SD = 1.26)	3.33 (SD = 1.89)

8.3. Feedback Modality

At the end of each session, we asked participants to indicate their feedback modality preference by ranking the different modes. In addition, participants rated how much they liked using each mode, how helpful they found each mode, and how easy to use they found each mode on a 7-point semantically anchored scale. For each scale, a 1 was better and it meant that they strongly liked the mode, found it very helpful, or found it very easy to use. A 7 was the worst and it meant that they strongly disliked it, found it very unhelpful, or found it very hard to use. Like our survey and interview, we found a wide range of preferences. Table II shows the mean for each mode on the scales, and Figure 8 shows the distributions of the responses.

However, the ranking of each method varied strongly between participants and over time. In the final session of the study, four out of 10 participants ranked Silent mode as their favorite mode, four ranked the Finger Pointing mode as their favorite, and two ranked the Verbal mode as their favorite. Interestingly, for many participants, their least favorite mode in the first session became their favorite mode by the last session or vice versa. Half of the participants had this change in preference between the first and last session, with three participants selecting Silent mode as their least favorite mode in the first session and their favorite mode in the last session.

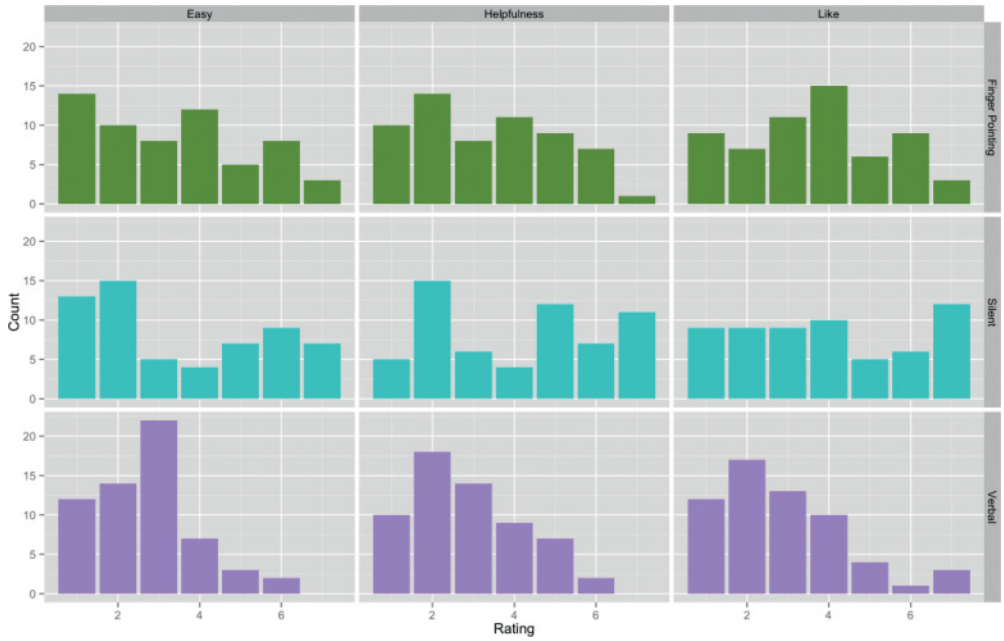

Fig. 8. The distribution of the Likert responses by mode and Likert question for all sessions. The columns from left to right represent the Likert questions: ease of use, helpfulness, and how well they liked using the mode. The rows from top to bottom represent the modes: Finger Pointing, Silent, and Verbal. For all the Likert scales, 1 is the worst rating and 7 is the best.

Participants who preferred the Finger Pointing mode generally thought it was more accurate. Participant 1 stated:

> *I like the concept of the finger pointing. I feel more confident that since it looks for a finger it's getting the right QR code if you have multiple on the same page.*

People who did not like the Finger Pointing mode thought it was difficult to use. Participant 3 stated that "I haven't been able to see my finger point in years, so knowing where my finger is isn't useful," but thought that it might be useful for others:

> *I did like that you're - that it's trying to branch out and give people options for identifying things like with a finger. It's a pretty neat touch. I like that. I could see that turning into something useful. I think my preference was still for just taking it with a simple picture with the camera.*

Participants who preferred the Silent mode were fatigued of audio feedback, as participant 3 said: "To be honest, I use screen readers every day and I am so sick of electronic noise."

Most participants disliked Silent mode because they felt that they needed feedback to know what was happening in the application. In the words of Participant 1: "the lack of feedback makes it harder to use because you don't know whether it sees a QR code," and Participant 9: "I still prefer having more versus less feedback."

Participants who preferred the Verbal mode liked it because it provided feedback but was less of a cognitive load than Finger Pointing. In the words of Participant 9: "the other thing I like about Verbal mode is that every time I hear a zero I think so I need to move it a little bit," as opposed to the Finger Pointing mode, which "presents more issues to deal with you already have to deal with how many labels are here and then I

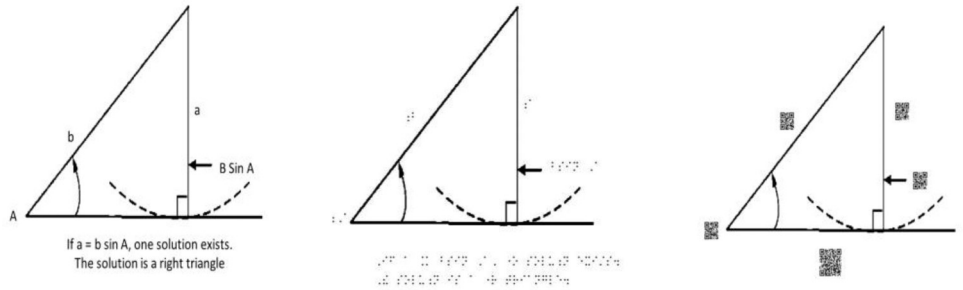

Fig. 9. A comparison of the same image which is similar to one from a precalculus textbook [Gordon-Holliday et al. 1999] in its original form, tactile graphic form with the labels in Braille and tactile graphic form with labels as QR codes. The bottom text is a good example where the QR codes can be smaller than the equivalent Braille.

got this finger issue this finger needs to be there, but it can't be too close [and] it can't be too far away."

9. COMPARISON TO BRAILLE

While our system was designed primarily for blind users who are unable to read Braille, there are benefits for people who are Braille literate as well.

9.1. Difficulties in Creating Tactile Graphics with Braille Labels

To assess the difficulties in producing tactile graphics, we spoke with three tactile graphics experts. All three had extensive experience in creating tactile graphics and had encountered a variety of problems with the creation of tactile graphics.

From our expert interviews, one common problem was how to place Braille labels on tactile graphics. Because of the limits of human tactile perception, Braille cannot be resized to fit into a small area Braille Cell Dimensions 2009. This means that labels with a large amount of text have to be moved. One technique for mitigating this problem is to create a key and legend. A short code is placed on the graphic where the label should be and the corresponding label is placed on a separate page. One of the experts estimated that the key and legend system is necessary for a quarter to a third of all the images he produces. Another tactile graphics expert mentioned that three quarters of tactile graphics require an explanation in order for them to be understood, and the explanation would not fit on the original graphic, requiring a second page.

9.2. Size of Braille Versus QR Codes

We did a size comparison between Braille and QR code labels and found that the QR codes are able to encode 45% more text in the same area (Figure 9). This calculation was completed by looking at 82 images from a precalculus textbook [Gordon-Holliday et al. 1999]. We calculated the estimated size of the Braille label using the product of the number of characters in the text and the size of a Braille cell, which is the standard size of all Braille characters [Braille Cell Dimensions 2009]. While many math symbols require multiple Braille characters, our conversion from text to Braille provides a good approximation. Unlike Braille, which has a standard size, QR codes vary in size based on the amount of text they encode and the distance from which they are meant to be scanned. By assuming a scan distance of 6 inches, we calculated the size of a QR code label based solely on the number of characters it encoded [Rule et al. 2011]. We found that the average QR code label size is 225 mm^2 and the average Braille label size is 327 mm^2.

9.3. Study

The main goal of the study was to determine if the TGV system was a feasible solution to making labels accessible to those who did not know Braille, and we feel our study demonstrates this fact. Here, we will explain the results from our comparison to Braille in the last session of our longitudinal study that was done with the six participants that did know Braille.

(1) *Accuracy:* Across all participants, the average accuracy for TGV using any mode was higher than the average accuracy using Braille (Silent mode: 88%, Verbal mode: 88%, Finger-Pointing mode: 89%, Braille: 77%). For the Bar Chart task, the TGV accuracies are similar to the average accuracy for Braille (Figure 6). While this finding goes against our hypothesis, this finding is likely the result of two things. First, in the Braille tasks the labels were written in Nemeth code. Even though we explained how to read Nemeth-coded numbers, some participants made mistakes. Second, some of the Braille-literate participants indicated that they were out of practice reading Braille.
(2) *Time:* The average completion time with the Braille graphics was 28.6 seconds (SD = 19.0). This was faster than the average time of TGV with all of the different modes. However, after the participants learned to use the application, the times were similar. This can be seen in Figure 7, where the dot representing the Braille mode was faster than Verbal and Finger-Pointing modes but slower than Silent mode in Session 6.
(3) *Preference:* Four of the six participants who used the Braille labels on the graphics stated that it was their favorite. One reason was because of ease of use, as Participant 6 stated that (with Braille) "it's already there and you can just read it." Additionally, people were more comfortable with Braille and thought it was more accurate. In the words of Participant 4: "I'm very comfortable with Braille. It feels more reliable."

The other two participants who preferred TGV to Braille did not feel comfortable with their Braille literacy skills. In the words of Participant 1:

> *I guess if you're reading a textbook in Braille you're probably up on your Braille so you wouldn't need a smartphone or anything to access that.*

Participant 5 said that:

> *I wish I had learned Braille when I was in school because that might that may have made a world of difference and I would be a lot more successful than I am right now so I do I really enjoy the Braille a lot.*

10. DISCUSSION

Errors on the first three tasks (Line, Parabola, and Triangle) were a result of misidentifying which label to scan or timing out. In contrast, with the Bar Chart, errors occurred when a participant attempted to scan the correct QR code but really scanned a different QR code. This issue occurred because of the small distances between the labels on the axes. Participants developed strategies to avoid this problem in later sessions, such as covering the neighboring QR codes. This technique helped increase the accuracy for all three modes from 55% (SD = 35) for the first session to 80% (SD = 30) for the last session. Figure 6 displays the changes in accuracy across the sessions by mode.

Although Finger-Pointing mode was the most accurate, many participants had difficulty using the mode. If users put their finger too close to the QR code, it would not recognize the QR code. There has to be a small gap between the finger and the QR code for both the finger and QR code to be recognized. This caused difficulties because

participants did not realize that their fingers were obscuring the QR code. Participant 2 expressed frustration:

> *With the pointing with the finger it kept not registering cause either my finger wasn't in the right spot or it kept picking up the wrong one somehow.*

We believe this is the main reason that Finger Pointing took significantly longer than the other modes and led to more time-outs.

11. GOOGLE GLASS

In user studies, we found that it was often difficult for participants to read the tactile graphic with one hand and aim the camera phone to scan the QR codes with the other hand. Often, users preferred to use both hands to explore the tactile graphic. Because of this feedback, we have begun to look at using a head-mounted camera, such as the one in Google Glass, to facilitate the scanning of the QR codes and exploring tactile graphics.

In preliminary work with Google Glass, we quickly found that because of the wide-angle lens camera, it was difficult to scan the small QR codes (1 to 2 inches wide) we used in the longitudinal study. Because of this difficulty, we have decided to use a different approach for Google Glass. Instead, we place a larger (4- to 5-inch) QR code on the back of the graphic. This QR code contains all of the information about each of the labels: location in x- and y-coordinates on the graphic, as well as the textual information of the label. On the front of the graphic, we print small tactile markers at the location of each label. Because we need to fit only a small tactile marker (equivalent in size to a single Braille character), the label markers are much easier to position in this version of Tactile Graphics with a Voice, and we are able to place the markers using the algorithm described in previous work to place Braille labels on tactile graphics [Jayant et al. 2007]. To explore the tactile graphic, a user would first scan the QR code on the back of the document using the Glass and the information about all the labels on the graphic is stored in the Glass. The user would then turn the graphic over to explore the front of the tactile graphic. When the user finds a tactile marker, he or she would point to it and use the voice command "Read Label" while looking in the general direction of the graphic. The Glass would take a picture, locate the tip of the user's finger, and read aloud the closest label. In Section 11.1, we describe the algorithm to first convert the skewed picture taken by the Glass so that the coordinates of the label can be calculated, and in Section 11.2, we describe the algorithm used for the identification of the finger. We used Open CV to do all of the image transformations.

11.1. Transforming the Skewed Image

The Glass converts the skewed image of the tactile graphic into rectangular coordinates by using the four corners and performing a perspective transform. Because tactile graphics are typically square (often 11×11 inches) with a standard width, we can convert from the skewed graphic to the rectangular graphics, even in an image that contains only two corners of the graphic. In order to do the perspective transform, we first reduce the size of the image by some constant factor to save computation time and space. We then convert the image from RGB to grayscale and apply a Gaussian blur to the image to remove small artifacts that are present. Next we run canny edge detection on the blurred grayscale image to find edges in the image (using 75 and 200 as our threshold values) [Szeliski 2011]. We then find the contours in the edged image and take the four-sided polygon that has the largest area, as this should be the piece of paper. Using the four corners of the polygon that we just found, we can use these four points to compute a four-point perspective transformation of the image [Szeliski 2011].

11.2. Identifying Where the User Is Pointing

We then use computer vision to identify the tip of the user's finger. In order to do this, we blur the image, convert the image to Hue-Saturation Value color space, and threshold (using the values (2, 50, 50) to (15, 255, 255), which were determined experimentally) the image to detect skin. We then pass the image through three dilations and blurs to remove artifacts. We find the contours in the thresholded image, and for each of these contours, we compute the contour area and choose the biggest one. Finally, we compute the convex hull for the largest contour, and the top finger can then be found by looking at the "highest" vertex in the convex hull [Szeliski 2011]. We now have the location on the paper that a finger is pointing at, as well as a top-down perspective of the paper, which we can use to determine which label's coordinates (known from scanning the QR code on the back of the image) are closest to the finger.

11.3. Comparison to the TGV Smartphone Application

Like the TGV system we tested in our user study, we still do not have to rely on an external database, as all the information is self-contained in the graphic. In addition, it has some advantages over the smartphone-based system: users can explore tactile graphics with two hands, it is easier to capture pictures of the graphics because of the wide-angle lens of the camera, and when making the tactile graphics, you only have to print and place one QR code per image as opposed to multiple QR codes. There are some drawbacks to using the head-mounted camera. Even though it does have a wide-angle lens, a user has to tilt his or her head down to scan an entire document that is placed on a table in front of him or her; in our preliminary studies, we have found that you must tilt your head at an angle that could get uncomfortable for long periods of time. However, we believe that with our current design this will not be a problem, as you only have to take a picture once to scan the QR code at the beginning and then once each time you have found a label and want to scan it.

12. CONCLUSION

We have presented Tactile Graphics with a Voice, a method to access the text labels in tactile graphics using a smartphone application that provides feedback to help blind users scan QR codes. To the best of our knowledge, our system is the only solution for people who are blind or have low vision and cannot read Braille to access the text on an image that does not require the use of a specialized device or access to proprietary software. We discuss the development of the system and algorithmic details of how we created graphics and enabled finger pointing. We conducted a longitudinal study and found that even for people who can read Braille, our system was comparable in terms of time and accuracy to Braille labels. We have begun to expand on this work using the head-mounted camera of Google Glass to enable someone to explore the graphic with both hands, and we discuss how we changed the system to accommodate the new form factor. Ensuring that blind people can quickly and accurately access the text labels on tactile graphics is an important part of making educational materials accessible to all.

13. FUTURE WORK

There are opportunities to improve both the smartphone application and the Google Glass application. One opportunity for improvement for both modes is to make finger pointing easier by improving our finger-pointing algorithms. There are tradeoffs, as the finger detection must be done quickly to ensure that the application is able to take multiple scans.

Additionally, as Google Glasses are currently a more expensive (and less accessible) alternative to smartphones, our smartphone application could be modified to work with

the setup used for the Google Glass application. A user would use his or her smartphone to scan a single QR code that contains all of the label information and location and then place the phone in a stand above the graphic so the entire graphic is in the field of view. The phone would then be able to detect where the user is pointing in terms of the graphics' coordinate system.

The longitudinal study allowed us to assess the feasibility of TGV on a smartphone in a controlled setting. We look forward to continuing to develop the application on Google Glass and conducting a user study to evaluate its efficacy.

REFERENCES

Frances K., Aldrich and Linda Sheppard. 2001. Tactile graphics in school education: Perspectives from pupils. *British Journal of Visual Impairment*, 19, 2, 69–73.

Catherine M. Baker, Lauren R. Milne, Jeffrey Scofield, Cynthia L. Bennett, and Richard E. Ladner. 2014. Tactile graphics with a voice: Using QR codes to access text in tactile graphics. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility (ASSETS'14)*, 75–82. DOI: http://doi.org/10.1145/2661334.2661366

Jeffrey P. Bigham, Chandrika Jayant, Andrew Miller, Brandyn White, and Tom Yeh. 2010. VizWiz::LocateIt - Enabling blind people to locate objects in their environment. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Workshops (CVPRW'10)*, 65–72. DOI: http://doi.org/10.1109/CVPRW.2010.5543821

Braille Cell Dimensions. 2009. http://www.tiresias.org/research/reports/braille_cell.htm.

Ahmed Elgammal, Crystal Muang, and Dunxu Hu. 2009. Skin detection-a short tutorial. *Encyclopedia of Biometrics*, 1–10. Retrieved from http://pdf.aminer.org/000/312/166/finding_facial_features_using_an_hls_colour_space.pdf.

Daniel Goransson. 2011. New VoiceOver Features in iOS 5. Retrieved from http://axslab.com/articles/new-voiceover-features-in-ios5.

Berchie Woods Gordon-Holliday, L. E. Yunker, G. Vannatta, and F. J. Crosswhite. 1999. *Advanced Mathematical Concepts: Precalculus with Applications*. Glencoe/McGraw-Hill.

Bill Holton. 2013. Voiceye: A breakthrough in document access. *AFB AccessWorld Magazine*.

Chandrika Jayant, Hanjie Ji, Samuel White, and Jeffrey P. Bigham. 2011. Supporting blind photography. In *Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'11)*, 203–210. DOI: http://doi.org/10.1145/2049536.2049573

Chandrika Jayant, Matt Renzelmann, Dana Wen, Satria Krisnandi, Richard Ladner, and Dan Comden. 2007. Automated tactile graphics translation. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS'07)*, 75. DOI: http://doi.org/10.1145/1296843.1296858

Shaun K. Kane, Brian Frey, and Jacob O. Wobbrock. 2013. Access Lens: A gesture-based screen reader for real-world documents. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI'13)*, 347–350. DOI: http://doi.org/10.1145/2470654.2470704

Steven Landau, Steven Holborow, and Jane Erin. 2004. The use of the talking tactile tablet for delivery of standardized tests. In *Proceedings of the Annual International Technology and Persons with Disabilities Conference*.

Steven Landau and Joshua Miele. 2010. Talking Tactile Apps for the Pulse Pen: STEM Binder. Retrieved from https://docs.google.com/presentation/d/1Ylscpk6QKX7Y5WCW3CFnhZDJJlL4X5ki8gHdcJvBp70/present#slide=id.i0.

Suranga Nanayakkara, Roy Shilkrot, Kian Peen Yeo, and Pattie Maes. 2013. EyeRing: A finger-worn input device for seamless interactions with our surroundings. In *Proceedings of the 4th Augmented Human International Conference on AH'13*. 13–20. DOI: http://doi.org/10.1145/2459236.2459240

National Federation of the Blind Jernigan Institute. 2009. *The Braille Literacy Crisis in America: Facing the Truth, Reversing the Trend, Empowering the Blind*. Retrieved from https://www.nfb.org.

Nemeth Braille. http://www.braillebug.org/nemeth_braille.asp.

Son Lam Phung, Abdesselam Bouzerdoum, and Douglas Chai. 2003. Skin segmentation using color and edge information. In *Proceedings of the 7th International Symposium on Signal Processing and Its Applications (ISSPA'03)*, 1 (July), 525–528. DOI: http://doi.org/10.1109/ISSPA.2003.1224755

Son Lam Phung, Abdesselam Bouzerdoum, and Douglas Chai. 2005. Skin segmentation using color pixel classification: Analysis and comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27, 1, 148–154. DOI: http://doi.org/10.1109/TPAMI.2005.17

2011. QR Code Minimum Size. http://www.qrstuff.com/blog/2011/11/23/qr-code-minimum-size.

Audrey C. Rule, Greg P. Stefanich, Robert M. Boody, and Belinda Peiffer. 2011. Impact of adaptive materials on teachers and their students with visual impairments in secondary science and mathematics classes. *International Journal of Science Education* 33, January 2015, 865–887. http://doi.org/10.1080/09500693.2010.506619

Stuart Russell and Peter Norvig. 2009. *Artificial Intelligence: A Modern Approach*, 3rd ed. DOI:http://doi.org/10.1017/S0269888900007724

Linda Sheppard and Frances K. Aldrich. 2001. Tactile graphics in school education: Perspectives from teachers. *British Journal of Visual Impairment*, 19, 3, 93–97. http://doi.org/10.1177/026461960101900303

Richard Szeliski. 2010. Computer vision: Algorithms and applications. *Computer* 5, 832. DOI:http://doi.org/10.1007/978-1-84882-935-0

Ender Tekin and James M. Coughlan. 2010. A mobile phone application enabling visually impaired users to find and read product barcodes. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *6180 LNCS* (Part 2), 290–295. DOI:http://doi.org/10.1007/978-3-642-14100-3_43

Marynel Vázquez and Aaron Steinfeld. 2012. Helping visually impaired users properly aim a camera. In *Proceedings of the 14th International ACM SIGACCESS Conference on Computers and Accessibility*, 95–102. http://doi.org/10.1145/2384916.2384934

Received April 2015; revised October 2015; accepted October 2015