Answering Visual Questions with Conversational Crowd Assistants

Walter S. Lasecki[1], Phyo Thiha[1], Yu Zhong[1], Erin Brady[1], and Jeffrey P. Bigham[1,2]

Computer Science, ROC HCI[1]
University of Rochester
{wlasecki,pthiha,zyu,brady}@cs.rochester.edu

Human-Computer Interaction Institute[2]
Carnegie Mellon University
jbigham@cmu.edu

ABSTRACT

Blind people face a range of accessibility challenges in their everyday lives, from reading the text on a package of food to traveling independently in a new place. Answering general questions about one's visual surroundings remains well beyond the capabilities of fully automated systems, but recent systems are showing the potential of engaging on-demand human workers (the crowd) to answer visual questions. The input to such systems has generally been a single image, which can limit the interaction with a worker to one question; or video streams where systems have paired the end user with a single worker, limiting the benefits of the crowd. In this paper, we introduce Chorus:View, a system that assists users over the course of longer interactions by engaging workers in a continuous conversation with the user about a video stream from the user's mobile device. We demonstrate the benefit of using multiple crowd workers instead of just one in terms of both latency and accuracy, then conduct a study with 10 blind users that shows Chorus:View answers common visual questions more quickly and accurately than existing approaches. We conclude with a discussion of users' feedback and potential future work on interactive crowd support of blind users.

Author Keywords

Assistive Technology, Crowdsourcing, Human Computation

ACM Classification Keywords

H.5.m. Information Interfaces and Presentation: Misc.

General Terms

Human Factors; Design; Measurement

INTRODUCTION

Blind people face a wide range of accessibility challenges in their daily lives, from reading the text on food packaging and signs, to navigating through a new place. Automatically answering general questions about users' visual surroundings requires not only effective machine vision and natural language processing, but also deep understanding of the connection between the two sources of information as well as the intent of the user. Each of these problems is a grand challenge of artificial intelligence (AI) individually, thus the combination remains well outside the scope of what fully automated systems are able to handle today or in the near future.

Figure 1. An set of questions asked by the pilot users of VizWiz.

In contrast, recent crowd-powered systems such as VizWiz [2] have shown the ability of on-demand human computation to handle many common problems that blind users encounter. Despite its usefulness, blind users input images to VizWiz one at a time, making it difficult to support users over the course of an entire interaction. In VizWiz, images are used because they represent small, atomic jobs that can be sent to multiple crowd workers for reliability of answers received. This model of question asking is largely ineffective for questions that span a series of sequential queries, which compose as much as 18% of questions asked to VizWiz. This difficulty is due to different workers answering each question, resulting in a loss of context and long completion times.

In this paper, we introduce Chorus:View, a system that is capable of answering sequential questions quickly and effectively. Chorus:View achieves this by enabling the crowd to have a consistent, reliable *conversation* with the user about a video stream from the user's phone. This speeds up the interaction by retaining workers instead of re-recruiting them, and by allowing workers to maintain context.

By keeping workers continuously connected to the task, we demonstrate that not only can Chorus:View handle problems for which single-image approaches are not designed to handle, but can complete tasks more efficiently than existing crowd-powered methods. We then discuss users' feedback and conclude with potential future work on supporting blind users in a variety of situations using the crowd.

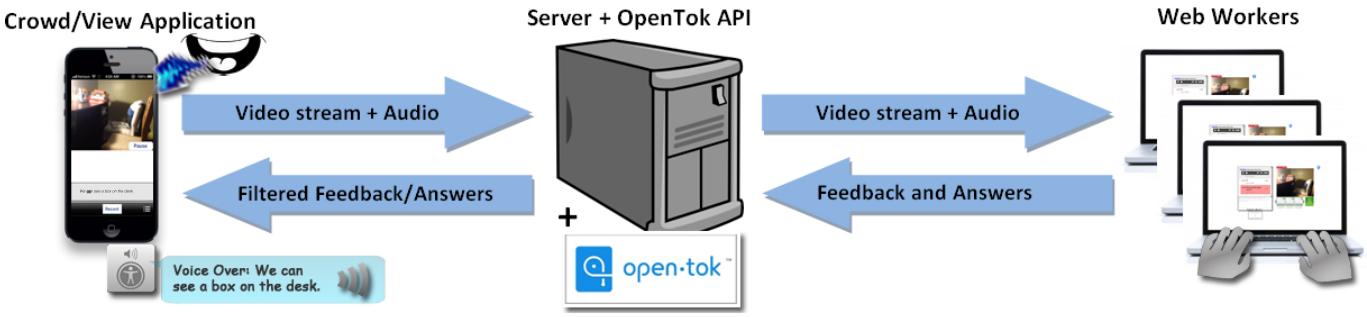

Figure 2. Chorus:View system architecture. The mobile app streams video captured by the user's phone using OpenTok, along with an audio message recorded by the user, to crowd workers. Workers watch the live video stream, and provide feedback and answer to the user's question. The server forwards selected feedback and answers to the mobile application, which is read by VoiceOver.

BACKGROUND

People with disabilities have relied on the support of people in their communities to overcome accessibility problems for centuries [3]. For instance, a volunteer may offer a few minutes of her time to read a blind person's mail aloud, or a fellow traveler may answer a quick question at the bus stop (e.g., "Is that the 45 coming?") Internet connectivity has dramatically expanded the pool of potential human supporters, but finding reliable assistance on-demand remains difficult. Chorus:View uses *crowds* of workers to improve reliability beyond what a single worker can be expected to provide.

Crowdsourcing has been proven to be an effective means of solving problems that AI currently struggles with by using human intelligence. Answering questions based on the context of visual data is difficult to do automatically because it requires understanding from natural language what parts of the image are important, and then extracting the appropriate information. At best, current automated approaches still struggle to accomplish one of these tasks at a time. Chorus:View allows users to leverage the *crowd*, recruited on-demand from online micro-task marketplaces (in our experiments, Amazon's Mechanical Turk), to interactively assist them in answering questions as if they were a single co-located individual. Chorus:View builds off of continuous crowdsourcing systems such as VizWiz [2], Legion [7], and Chorus [9]. In this section, we describe what made these systems unique, and how we leverage similar approaches to create an interactive accessibility tool powered by the crowd.

Automated Assistive Technology

Most prior mobile access technology for blind people involves custom hardware devices that read specific information then speak it aloud, such as talking barcode readers, color identifiers, and optical character recognizers (OCR). These systems are expensive, (typically hundreds of dollars [6]), and often don't work well in real-world situations. Thus, these devices have had limited uptake.

Recently, many standard mobile devices have begun incorporating screen reading software that allows blind people to use them. For instance, Google's Android platform now includes Eyes-Free (code.google.com/p/eyes-free), and Apple's iOS platform includes VoiceOver (www.apple.com/accessibility/voiceover). These tools make it possible for multitouch interfaces to leverage the spatial layout of the screen to allow blind people to use them. In fact, touchscreen devices can even be preferred over button-based devices by blind users [5]. We have developed our initial version of Chorus:View for the iPhone because it is particularly popular among blind users.

Because of the accessibility and ubiquity of these mobile devices, a number of applications—GPS navigation software, color recognizers, and OCR readers—have been developed to assist blind people. One of the most popular applications is LookTel (www.looktel.com), which is able to identify U.S. currency denominations. However, the capabilities of these automatic systems are limited and often fail in real use cases. More recently, the idea of "human-powered access technology" [3] has been presented. For instance, oMoby is an application that uses a combination of computer vision and human computation to perform object recognition (www.iqengines.com/omoby). A handful of systems have attempted to allow a blind user to be paired with a professional employee via a video stream (e.g. Blind Sight's "Sight on Call") but to our knowledge have not been released due to the expense of recruiting and paying for the sighted assistance. Chorus:View allows a dynamic group of whoever is available to help, and mediates input to ensure quality.

VizWiz

VizWiz [2] is a mobile phone application that allows blind users to send pictures of their visual questions to sighted answerers. Users can record a question and choose to route it to anonymous crowd workers, automatic image processing, or members of the user's social network. In its first year, over 40,000 questions were asked by VizWiz users, providing important information about the accessibility problems faced by blind people in their everyday lives [4]. Figure 1 shows an interaction from the pilot version of the application.

When questions are sent to crowd workers, answers are elicited in *nearly realtime* by pre-recruiting members of the crowd from services such as Mechanical Turk. Workers are asked to answer the user's question, or provide them with feedback on how to improve the pictures if the information needed to answer a question is not yet visible. Figure 3 shows that despite cases where individual feedback and answers can be obtained quickly, answering long sequences of questions using single-image interactions can be time consuming.

Properly framing an image can be challenging even for an experienced user because they must center an object at the right

distance without visual feedback. While some approaches have tried to provide feedback about framing automatically (e.g., EasySnap [14]), these can be brittle, and don't account for object orientation or occlusion within the frame (i.e. a label facing away, or important information being covered by the user's thumb). Chorus:View's interactive approach allows users to get open-ended feedback and suggestions from the crowd as if they were an individual helping in-person.

Legion

Legion [7] is a system that allows users to crowdsource control of existing interfaces by defining tasks using natural language. This work introduced the idea of continuous real-time interactions to crowdsourcing. Prior systems (such as [13, 10, 1]) focused on discrete micro-tasks that workers were asked to complete, whereas in Legion, workers are asked to participate in a closed-loop interaction with the system they are controlling. This model has also been used to power assistive technologies such as real-time captioning using the crowd [8].

Chorus:View uses a similar system of collective input to respond to users in real-time. The difference is that while Legion combines workers' inputs behind the scenes, Chorus:View explicit uses a custom interface and incentive mechanism designed to encourage workers to both generate and filter responses.

Chorus

Chorus is a crowd-powered conversational assistant. It elicits collaborative response generation and filtering from multiple crowd workers at once by rewarding workers for messages accepted by other workers (more for proposing a message than for just voting). Using this, Chorus is able to hold reliable, consistent conversations with a user via an instant messenger interface. Chorus not only provides a framework for the design of a conversational interface for the crowd, but demonstrates the power of crowds versus single workers in an information finding task. Chorus:View uses the idea of a conversation with the crowd, and focuses the interaction around a video stream displayed to workers in the interface.

Privacy

Chorus:View introduces privacy concerns by making both streaming video (potentially of the user or their home) and audio recordings of the user's voice available to people on the web. While a vast majority of workers in the crowd are focused on helping the user, the potential risks might impede adoption. Many problems are easily mitigated by educating users about the operation of the system. For instance, by not streaming video near potentially private information (such as letters with addresses or other documents), and disabling video between questions, the risk of disclosing information unintentionally can be reduced.

Prior systems have also looked at how video might be streamed to the crowd in a privacy-preserving way without reducing the workers' ability to complete the underlying task. For instance Legion:AR [12] helped mitigate privacy concerns by using automatically-generated privacy veils that covered a user's face and body, and by limiting a worker's involvement with a single user. In many ways, VizWiz has shown that this type of service can be embraced by users for its benefits, even in the presence of potential security issues.

MOTIVATION

While visual question answering tools like VizWiz allow users to get answers from single photographs, there remain scenarios where this model does not allow more complex question asking. In this section we discuss current issues with the photographs and questions being sent in by blind VizWiz users, and discuss additional use cases in which VizWiz may not be a suitable source for answering user's questions.

Current Uses

VizWiz users frequently ask one-off identification, description, and reading questions, which best suit its design where each worker interacts only with a single photograph and question, and does not have any of the previous context from recent questions. In contrast, Chorus:View focus on answering questions that require multiple contextually-related steps, such as finding a package of food, reading its type, then locating and reading the cooking instructions (an example of a user completing this task using VizWiz is shown in Figure 3), or locating a dropped object on the ground (which often requires searching an area larger than the default field of view of a single image taken at standing height). For these tasks, contextual information is critical to achieve the final objective and single-photograph based approaches may be ill-suited to answer due to the limited scope of a single photo.

In order to demonstrate that there is a need for tools that improve sequential question answering, we first analyzed a month of VizWiz data to determine how often users asked a *sequence of questions*, where multiple questions were asked by a user within a short period of time. These questions may show scenarios where users have multiple questions about an object that cannot be answered at once, or where their photography techniques must be corrected by workers before an answer can be found. 1374 questions were asked by 549 users during the analysis period (March 2013). In order to identify potential sequences, we looked for groups of 3 or more questions asked by users in which each question arrived within 10 minutes of the previously asked question. We identified 110 potential sequences from 89 users, with an average sequence length of 3.63 questions.

After identifying potential sequences, we narrowed our definition to *likely* sequences where the objects in three or more photographs from the sequence were related, and the questions asked were related over multiple images. For this task, our definition of related images focused on objects that were the same or similar types (e.g., two different flavors of frozen dinner), while related questions were either the same question asked repeatedly, questions where the user asked for multiple pieces of information about the same subject, or questions where no audio recording was received. A single rater performed this evaluation for all 110 potential sequences. We also validated our definitions of relatedness by having another rater evaluate 25 sequences (Cohen's kappa of $k = 0.733$).

In all, we found that 68 of our potential sequences (61.8%) contained a likely sequence, meaning that nearly 18% of

Figure 3. An example of a question asked by a VizWiz user. The sequence of pictures was taken to answer the question "How do I cook this?" Each question was answered in under a minute, but overall, the cooking instructions took over 10 minutes to receive as crowd workers helped the user frame the correct region of the box. Chorus:View supports a conversation between a user and the crowd so that corrections can be made quicker and easier.

VizWiz questions are likely sequences. The average sequence length was 3.70, with an average total time span of 10.01 minutes to ask all photographs (in comparison, a single VizWiz question is asked and answered in an average of 98 seconds).

These numbers serve as initial motivation, but are most likely under-representing the problem, since users who do not get correct answers to their questions or who had difficulty taking photographs have been less likely to continue using the VizWiz service after their first use [4].

New Uses

While examining the current use of VizWiz for sequential question asking provides us with a demonstrated need for longer term question asking, we believe that new tools may be more suited to support longer term, contextual interactions between workers and users. For instance, many VizWiz users initially asked follow-up questions of the form "How about now?", which assumed the crowd had context from the previous question. After getting responses from the crowd that made it clear the initial question had been lost, users would learn to ask future questions which contained the full original question. This shows that users expect the system to remember initially, which we suspect makes the continuous approach taken by Chorus:View more natural.

There are also situations that single-image approaches were never meant to handle. For instance, navigating using visual cues such as street signs, locating an object from a distance (i.e. finding a shirt, or trying to find a place in a park with benches). In these cases, we believe existing VizWiz users either immediately understand that the requirement of taking only single photos is not well-suited to these problems, or have learned not to use the application for this over time. As such, usage logs are not reasonable indicators for how much use there is for such capabilities. By developing Chorus:View, we introduce a way to meet a range of user needs that were not previously met using VizWiz.

PRELIMINARY TESTS

In order to test how users might benefit from our approach, we first designed a wizard-of-oz test using student workers. We first prototyped a version of VizWiz which had users submit a short video instead of a single image. However, it was very clear from initial tests that the down-side of this approach was the high cost of resubmitting a question in the event that a mistake was made in framing or focus (which was common).

Next, we developed a real-time streaming prototype, by using FaceTime (www.apple.com/ios/facetime/) to stream video from a users' phone to workers' desktop computer. We configured the computer used by the worker so that they could view the streamed video and hear the audio question from the user via FaceTime, but must reply via text, which was read by VoiceOver on the user's iPhone.

Experiments

To test the effect of our streaming prototype against VizWiz, we paired users and workers in randomized trials, and measured the time taken to complete each task. The user can ask question(s) to the worker, who will respond with either instructions to help frame the item or an answer to the question. We recruited 6 blind users (3 females) to determine if they would benefit from continuous feedback and if so, how they might use it. We also recruited 6 students as volunteer workers who answered questions from the blind users remotely. The first task was to find out specific nutrition content—for example, the sodium content—for each of three different food items that were readily found in the kitchen. The second task was to get the usage instructions from a shampoo bottle.

Results

Our results showed that users spent an average of 141 seconds getting the shampoo usage instructions via the streaming prototype versus an average of 492.6 seconds using VizWiz. In finding the nutrition content of three food items, the users spent 607.2 seconds to obtain satisfactory answers via the prototype whereas they spent 1754.4 seconds using VizWiz. Further analysis showed that system, task type (nutrition content vs. usage instruction) and their interaction together account for about 36.3% ($R^2 = 0.363$) of the variance in task completion time, and it is significant ($F(3, 18) = 3.418$, $p < 0.05$). Moreover, method has a significant effect on completion time (beta=0.293, $F(1, 18) = 8.393$, $p < 0.01$).

We asked the users to rate how easy it was to use of each method to accomplish different tasks based on a 7-point Likert scale where 1 was "strongly disagree" and 7 was "strongly agree". The median Likert rating for the prototype was 6.0 for both nutrition content and usage instruction tasks, while it was 3.0 and 2.0 respectively for corresponding tasks in VizWiz. Additionally, 4 out of 6 users indicated that they prefer to use the Chorus:View prototype if only one of the two approaches was available to them.

From our observation of the workers, some seemed hesitant to respond to questions if they were not completely confident about the answer, and some resisted giving corrective suggestions for the phone camera as they assumed their responsibility was solely to provide the answers. When we told them that they could help the users with positioning and framing the camera, most of them adapted quickly. In addition, we noted some instances where responses from the workers might have potentially caused friction between the user-worker interaction. In such incidents, the users were frustrated with the help they received from the workers. This indicates that training the workers is necessary to help them understand the system, and provide helpful and acceptable responses. As we discuss later, we use a combination of a video tutorial and an interactive tutorial to train workers in Chorus:View.

On the other hand, the users struggled to properly align images using both systems, but were able to get responses faster using the streaming prototype. They were also able to benefit from better scanning behavior compared to VizWiz due to immediate feedback from the workers. For example, the user panned the camera across the object of interest until they were told to stop. As a result, the answers came faster. VizWiz was mostly plagued by framing problems and users who had never used VizWiz previously struggled more noticeably.

Despite its advantages over VizWiz, our prototype system did not allow workers to view past content, which often made reading small details difficult. Based on this, Chorus:View gives workers the ability to capture static images of the video.

CHORUS: VIEW

In order to improve on visual question answering for the blind, Chorus:View adds the concept of a closed-loop question and answer interaction with the crowd. In this system, users capture streaming video and a series of questions via the mobile interface, and workers provide feedback to users, in the form of corrections and answers.

When the Chorus:View mobile application is started, the server recruits crowd workers from Mechanical Turk. The server then forwards the audio question and the video stream from the user to these workers, via the worker interface. Responses are aggregated using a collective (group) chat interface in order to respond to the user's questions using natural language. While workers will still reply in text, a screen reader on the user's phone performs text to speech conversion to make the text accessible. Chorus:View allows very rapid successions of answers (each within a matter of seconds) to complex or multi-stage questions, and lets workers give feedback to users on how to frame the required information.

VizWiz users have previously noted that they prefer a more continuous interaction [2]. We used an iterative, user-centered design process to develop the Chorus:View mobile application that allows users to stream video and send audio questions to our service, as well as the worker interface which elicits helpful responses. Instead of allowing only a single audio question per submission like VizWiz does currently, the application lets users record an initial question and then re-record a new one at any point, which are shown to workers as a playable audio file embedded in the chat window. Future versions of the system will also include a corresponding caption generated by either automatic speech recognition or a crowd-powered solution such as Legion:Scribe [8]. This allows users to modify a question to get the response they were looking for, or ask follow-up questions to the same crowd of workers who know the answers to previously asked questions.

We avoided using speech responses from the crowd because it adds latency during worker evaluation, not all workers have access to a high-quality microphone, and differences between workers' and users' English fluency (and accents) adds difficulty to understanding responses. We also tried adding preset responses to the worker interface to allow them to provide direction and orientation feedback to users quickly; however, we found that switching between multiple modes of interaction (typing and clicking) caused confusion among workers.

User Interface

When users start the Chorus:View mobile application, they are given the option to immediately begin streaming video to the crowd, or first record a question. When they decide to begin streaming, workers can view this content and provide feedback. Users receive this feedback in a text area that is automatically read by VoiceOver. Users can also read previous responses by using a history button that will allow them to select and listen to old messages. At any point in this process, users can choose to either pause the video stream (useful for when they want to complete a task in privacy but know there is another question that they will need to ask soon), or end the session completely, which will also end the worker's session and compensate them for their contributions.

Worker Interface

Workers are presented with an interface which has two main components: video streamed from the user and a chat window that allows them to jointly generate responses along with others synchronously. To stream and display video, we use OpenTok (www.tokbox.com/opentok). To enable a conversation between the user and multiple crowd workers, we use a chat interface similar to Chorus [9].

Viewing the Video Stream and Capturing Screenshots

During our preliminary tests, we found that most users could not hold the camera stable enough for workers to focus on small details. In response, we designed the worker interface to allow workers to capture still images of the video stream by clicking on the video window. This functionality enables workers to capture moments of interest from the video stream, and view them at a time of convenience to answer the questions. The interface also allows the workers to delete and

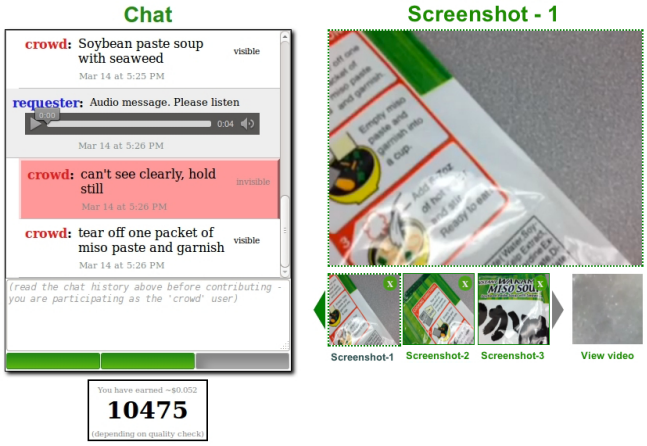

Figure 4. Chorus:View worker interface. Workers are shown a video streamed from the user's phone, and asked to reply to spoken queries using the chat interface. In this example, the recorded message asks workers "How do I cook this?" a question that often involves multiple image-framing steps and was often challenging using VizWiz.

browse all the captured images, and switch back to viewing the live video stream at any moment. Alternatively, future versions of Chorus:View could include video playback controls instead of capturing select images.

Feedback Selection Process

In order to select only a set of responses that is beneficial to the user, our system uses an approach similar to Chorus, in which workers can both vote on existing responses and propose new ones. In Chorus:View, workers need to agree quickly on what feedback to give users, while still being consistent (receiving both 'move item left' and 'move item right' at the same time as a response would be confusing). Chorus:View chat system helps do that by introducing an incentive mechanism that rewards workers for quickly generating answers, but also for coming to agreement with others. Once sufficient consensus is found, the text response is forwarded to the user to be read aloud. The latency of this approach is low enough that it does not interfere with the overall speed of the interaction (around 10-15 seconds per response).

Server

The Chorus:View server handles the streaming media management, worker recruitment, and input mediation. When a user starts the application, the server automatically generates a session ID and begins recruiting workers from Mechanical Turk using quikTurKit [2]. Once requested, streaming video is then sent from the user's mobile device to the server using OpenTok. The server manages all current connections using the session ID parameter, publishes the video stream and forwards audio messages to the appropriate set of workers. Once workers respond, the server tracks the votes and forwards accepted messages back to the user's device.

EXPERIMENTS

The goal of our experiments was to better understand Chorus:View's performance on question types that often require asking multiple repeated or related questions.

Worker Study

We began our experiments to measure the ability for workers to answer questions using Chorus:View. Our chat interface supports multiple simultaneous workers collaboratively proposing and filtering responses, but a key questions is if we really *need* multiple workers to effectively answer users' questions. To test this, we had users ask a set of objective questions about information contained on an object. Using 10 household objects (8 food items), a unique question was asked regarding a single fact discernible from text on each object selected. The study was run over the span of 2 days to get a wider sample of crowd workers.

We tested 2 conditions with each object. In the first condition, a single worker was used to provide answers. In the second, a group of 5 workers were recruited to answer questions simultaneously. We randomized the order of the items being tested, the order of the conditions, and separated tests using the same item to be on different days to help prevent returning workers from recalling the answers from previous trials.

We recruited all of our workers from Mechanical Turk. For this series of tests, the 'user' role was played by a researcher who followed a predefined set of guidelines determining how he would respond to given types of user feedback (i.e. 'turn the can to the right' would be performed the same as 'turn the can 90 degrees counter-clockwise'). The condition used for each trial—single versus multiple workers—was also made unknown to the researcher. For any situation that was not in the predefined set, an interpretation was made as needed and used consistently for the rest of the trials (if the situation arose again). Our goal was to normalize for the user's behavior within these tests to more accurately measure the workers.

Worker Results

We had a total of 34 distinct workers contribute to our task. Of these, 29 proposed at least one new answer, and 20 voted for at least one answer generated by another worker.

We observed a clear difference between single workers and groups of workers in terms of speed and accuracy. Using a single worker, the mean time to first response was 45.1 seconds (median of 47.5 seconds), most likely due to workers trying to orient themselves to the task with no other input provided from other workers. When using a group of workers, there was a drop of 61.6% to 17.3 seconds (median of 16.5 seconds), which was significant ($p < 0.05$). Likewise, the average time until final response decreased from 222.5 seconds (median of 190.5 seconds) to 111.7 seconds (median of 113.5 seconds), a significant difference of 49.8% ($p < 0.05$).

User Study

The second part of our experiments focused on testing blind users' ability to use Chorus:View. We recruited blind users to complete a set of experiments similar to the ones in our preliminary trial, which are broken up into three sets:

- **Product Detail:** One of the most common problems observed in VizWiz is that, using a one-off interaction with workers, it is difficult or even impossible to maintain context and get the most useful responses to a series of re-

peated questions, such as when users re-ask a question after adjusting their camera based on previous feedback from the crowd. We compared Chorus:View to VizWiz in these situations by using each to find a specific piece of information about a product. Users were asked to select a household object that has an expiration date, then ask the systems to tell them the date. This task is a common example of one that requires careful framing of information with no tactile indicators meaning that users often require a lot of feedback from workers to frame the date properly.

- **Sequential Information Finding:** We next compared the ability of both systems to find sequences of related information by asking users to find a package of food which is not identifiable by shape (e.g., cans of different types of vegetables), then find first the type of food, and then the cooking instructions. While finding each progressive piece of information, the crowd observes more of the item, potentially allowing them to get the next piece of information more quickly than a new set of workers could.
- **Navigation:** Chorus:View's continuous interaction also allows for new applications beyond the scope of VizWiz. We explored Chorus:View's ability to help users navigate to visual cues by asking participants to simulate an accidentally dropped item by rolling up a shirt and tossing it gently. They were asked to face away from the direction that they threw the shirt, and then use Chorus:View to locate it and navigate to the shirt. While users had a rough idea where the item landed (as is often the case when looking for something), they do not know exactly where to begin and thus often have difficulty.

Each of these conditions was randomized, as was the order of the systems used. Because many workers return to the VizWiz task posted to Mechanical Turk, there is an experienced existing workforce in place. To avoid bias from this set of workers, we used a different requester account to post the tasks (since many workers return to repeated tasks based on the requester identity). In order to ensure the number of responses was on a par with Chorus:View, we increased the number of responses requested by the system from 1-2 (in the deployed version of VizWiz) to 5. We also increased the pay to \$0.20–\$0.30 per task (roughly \$10–\$20 per hour), instead of the usual \$0.05, to better match the \$10 per hour anticipated pay rate of Chorus:View.

In order to limit interactions to a reasonable length, we set a maximum task time of 10 minutes. If the correct information was not found by the time limit, the task was considered incomplete. For these tasks, the user expects an answer immediately (classified "urgent" in [4]).

User Results

We recruited a total of 10 blind users and 78 unique workers from Mechanical Turk to test Chorus:View. Our users ranged from 21–43 years old and consisted of 6 males and 4 females—6 of them use VizWiz at least once per month, but only 2 used it more than once per week on average.

For the product detail task, Chorus:View completed 9 out of 10 tasks successfully, with an average time of 295 seconds, while VizWiz completed only 2 trials within the 10 minute limit, with an average time of 440.7 seconds. For the sequential information finding task, Chorus:View completed all 10 of the trials within the 10 minute limit, with an average time of 351.2 seconds, while VizWiz completed only 6 trials with an average time of 406.8 seconds. For the navigation task, Chorus:View was able to complete all 10 trials, with an average time of 182.3 seconds. VizWiz was not tested in this condition since its interaction is not designed to support this case, and thus in preliminary tests, it was not possible to complete the task in a reasonable amount of time in many cases.

The completion rate of Chorus:View (95%) is significantly higher than VizWiz (40%). A two-way ANOVA showed significant main effects of the difference between the two applications ($F(1, 36) = 22.22, p < .001$), as well as the two task types ($55\% vs. 80\%, F(1, 36) = 4.59, p < .05$), but there was no significant interaction. Therefore blind users were more likely to successfully complete both sequential information finding tasks and more difficult tasks which involve exploring a large space for a tiny bit of information with Chorus:View. However, the improvement in completion time of Chorus:View over VizWiz was not significant. This is due to the small number of VizWiz trials that actually completed within the allotted amount of time.

DISCUSSION AND FUTURE WORK

Our tests showed that Chorus:View's continuous approach achieved a significant improvement in terms of response time and accuracy over VizWiz in cases where sequential information is needed. While both VizWiz and Chorus:View rely on images from the user's phone camera, there is a clear difference in final accuracy. Interestingly, the lower resolution source (video in Chorus:View) outperforms the higher resolution images from VizWiz. We believe this is due to VizWiz workers being biased towards providing an answer over feedback if unsure. Additionally, video allows workers to give incremental feedback over the user's camera positioning, potentially improving the quality of the images in the video stream.

We also observed a number of other effects on the quality of responses. For instance, some VizWiz workers assumed the task was simple object identification and did not listen to the questions, resulting in unhelpful responses. In Chorus:View, workers can see one another's contributions, and get a better understanding of the task. Some of the workers used "We" to refer to themselves, which hints to us that they collectively shoulder the responsibility of helping the user. In some trials, we observed that the workers discussed among themselves—that is, without voting on the chat submissions—whether certain information is correct or not. A few users also thanked the workers when they received the correct answers, which is the sort of interaction not possible in VizWiz. These interactions lead us to believe that Chorus:View system engenders a collaborative and mutually-beneficial relationship between the user and the workers. On the other hand, overenthusiastic workers might trigger negative effects, such as inundating the user with multiple simultaneous—and potentially contradicting—feedback.

The way a user asked a question influenced the helpfulness of the responses they received, especially when an answer was not readily apparent from the current view. For example, asking "What is the expiration date of this food?" received short unhelpful answers when the expiration date was not in view, e.g. "Can't see." Slight variations on this question, e.g. "Can you guide me to find the expiration date of this food?", generated more helpful responses like "The label is not in the picture. Try turning the box over to show the other side." Future work may explore either training users to ask questions more likely to elicit helpful responses, or emphasizing to workers how to provide a helpful response even if they cannot answer the specific question asked by a user.

User Feedback

Users were generally very enthusiastic about the potential of Chorus:View, while noting that VizWiz could be also be helpful for many non-sequential tasks (which fits its design). One user noted, "For common tasks such as identifying a product I would use VizWiz, but for more specifics like obtaining instructions on a box or can, I would use the Chorus:View application most certainly." In other cases though, the more rapid feedback loop of answer and framing help was seen as a major benefit. The video is also helpful because it does not force users to achieve challenging maneuver such as getting camera focus right on the first try. A user commented, "...[Chorus:View] is more convenient to use and you don't have to take a 'perfect' picture and wait for the results. It's just more practical when someone can tell you how to move the camera differently." However, one user raised a concern about Chorus:View application potentially using up the data quota when on carrier's network. To prevent this, we plan to update the app to use wireless networks when possible.

Worker Feedback

During our user studies, we also got a significant amount of feedback from crowd workers. While positive feedback from Mechanical Turk workers is very rare, we received emails from 10 workers saying the enjoyed doing the task and wanted to continue participation because of the benefit to the end user. We expect this to help us eventually train a workforce of experienced workers who feel good about contributing to the task, much in the same way the deployed version of VizWiz benefits from this same effect.

FUTURE WORK

Our work on Chorus:View points to a number of exciting future directions of research. First, there is still room to improve the speed of the interaction, as well as determine when users are best served by Chorus:View's continuous interaction, versus VizWiz's asynchronous interaction (where users have the ability to do other things while they wait for a response). Once released to the public, Chorus:View will provide an unmatched view into how low-vision users interact with a personal assistant who can guide them throughout their daily lives, and help answer questions that were not feasible before even when using VizWiz. More generally, Chorus:View will provide insight into how people interact when they are able to query a personal assistant that remains contextually aware using the sensors available in common mobile devices. This applies to both low-vision and traditional users, who would benefit from applications such as a location-aware navigation assistant that can find a route and provide navigation instructions based on a complex but intuitive sets of personal preferences that the user conveyed in natural language.

CONCLUSION

In this paper we have presented Chorus:View, a system that enables blind users to effectively ask a sequence of visual questions to the crowd via real-time interaction from their mobile device. Our tests indicate that the continuous interaction afforded by Chorus:View provides substantial benefit as compared to prior single-question approaches. Chorus:View is an example of a continuous approach to crowdsourcing accessibility. Its framework may help to enable future crowd-powered access technology that needs to be interactive.

ACKNOWLEDGMENTS

This work is supported by National Science Foundation awards #IIS-1116051 and #IIS-1149709, and a Microsoft Research Ph.D. Fellowship.

REFERENCES

1. Bernstein, M. S., Little, G., Miller, R. C., Hartmann, B., Ackerman, M. S., Karger, D. R., Crowell, D., and Panovich, K. Soylent: a word processor with a crowd inside. In *Proc. of UIST 2010*, 313–322.
2. Bigham, J. P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R. C., Miller, R., Tatarowicz, A., White, B., White, S., and Yeh, T. Vizwiz: nearly real-time answers to visual questions. In *Proc. of UIST 2010*, 333–342.
3. Bigham, J. P., Ladner, R. E., and Borodin, Y. The design of human-powered access technology. In *Proc. of ASSETS 2011*, 3–10.
4. Brady, E., Morris, M. R., Zhong, Y., White, S. C., and Bigham, J. P. Visual Challenges in the Everyday Lives of Blind People. In *Proc. of CHI 2013*, 2117–2126.
5. Kane, S. K., Bigham, J. P., and Wobbrock, J. O. Slide rule: making mobile touch screens accessible to blind people using multi-touch interaction techniques. In *Proc. of ASSETS 2008* (2008), 73–80.
6. Kane, S. K., Jayant, C., Wobbrock, J. O., and Ladner, R. E. Freedom to roam: A study of mobile device adoption and accessibility for people with visual and motor disabilities. In *Proc. of ASSETS 2009*.
7. Lasecki, W., Murray, K., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *Proc. of UIST 2011*, 23–32.
8. Lasecki, W. S., Miller, C. D., Sadilek, A., Abumoussa, A., Borrello, D., Kushalnagar, R., and Bigham, J. P. Real-time captioning by groups of non-experts. In *Proc. of UIST 2012)*, 23–33.
9. Lasecki, W. S., Wesley, R., Nichols, J., Kulkarni, A., Allen, J. F., and Bigham, J. P. Chorus: A Crowd-Powered Conversational Assistant. In *Proc. of UIST 2013*, To Appear.
10. Little, G., Chilton, L. B., Goldman, M., and Miller, R. C. Turkit: human computation algorithms on mechanical turk. In *Proc. of UIST 2010*, 57–66.
11. Singh, P., Lasecki, W.S., Barelli, P. and Bigham, J.P. HiveMind: A Framework for Optimizing Open-Ended Responses from the Crowd. URCS Technical Report, 2012
12. Lasecki, W. S., Song, Y. C., Kautz, H., and Bigham, J. P.. Real-time crowd labeling for deployable activity recognition. In *Proc of CSCW 2013*, 1203–1212.
13. von Ahn, L., and Dabbish, L. Labeling images with a computer game. In *Proc. of CHI 2004*, 319–326.
14. White, S., Ji, H., and Bigham, J. P. Easysnap: real-time audio feedback for blind photography. In *Proc. of UIST 2010 - Demonstration*, 409–410.