VizWiz: Nearly Real-time Answers to Visual Questions

Jeffrey P. Bigham[†], *Chandrika Jayant*[ϕ], *Hanjie Ji*[†], *Greg Little*[§], *Andrew Miller*[γ], *Robert C. Miller*[§], *Robin Miller*[†], *Aubrey Tatarowicz*[§], *Brandyn White*[‡], *Samuel White*[†], *and Tom Yeh*[‡]

[†]University of Rochester Computer Science
Rochester, NY 14627 USA
{jbigham, hji, rmill13, swhite24}@cs.rochester.edu

[‡]University of Maryland Computer Science
College Park, MD 20742 USA
{bwhite, tomyeh}@umiacs.umd.edu

[§]MIT CSAIL
Cambridge, MA 02139 USA
{altat, glittle, rcm}@mit.edu

[γ]University of Central Florida CS
Orlando, FL 32816 USA
amiller@ucf.edu

[ϕ]University of Washington CSE
Seattle, WA 98195 USA
cjayant@cs.washington.edu

ABSTRACT

The lack of access to visual information like text labels, icons, and colors can cause frustration and decrease independence for blind people. Current access technology uses automatic approaches to address some problems in this space, but the technology is error-prone, limited in scope, and quite expensive. In this paper, we introduce *VizWiz*, a talking application for mobile phones that offers a new alternative to answering visual questions in nearly real-time—asking multiple people on the web. To support answering questions quickly, we introduce a general approach for intelligently recruiting human workers in advance called *quikTurkit* so that workers are available when new questions arrive. A field deployment with 11 blind participants illustrates that blind people can effectively use VizWiz to cheaply answer questions in their everyday lives, highlighting issues that automatic approaches will need to address to be useful. Finally, we illustrate the potential of using VizWiz as part of the participatory design of advanced tools by using it to build and evaluate VizWiz::LocateIt, an interactive mobile tool that helps blind people solve general visual search problems.

ACM Classification: H5.2 [Information interfaces and presentation]: User Interfaces - Graphical user interfaces.

General terms: Human Factors, Design, Experimentation

Keywords: Real-Time Human Computation, Non-Visual Interfaces, Blind Users

INTRODUCTION

Our environment often assumes the ability to see. Food products otherwise indistinguishable are labeled with their contents, color relays semantics, and currency denominations are differentiable only by the writing on them[1]. A quick visual scan helps stop minor problems from turning into big frustrations—a sighted person can tell in a glance if their clothes match before an important job interview, spot an empty picnic table at the park, or locate the restroom at the other end of the room without having to ask. Blind people often have effective, albeit inefficient, work-arounds that render individual problems into mere nuisances. Collectively, however, small problems can lead to decreased independence.

Talking mobile devices from both research and industry have been designed to help blind people solve visual problems in their everyday lives, but current automatic approaches are not yet up to the task. Products designed for blind people are specialized for a few functions, are prone to errors, and are usually quite expensive. As an example, the popular Kurzweil knfbReader software ($1000 USD) uses optical character recognition (OCR) to convert text to speech in pictures taken by users on their mobile devices [16]. When it works, this product offers the independence of reading printed material anywhere, but unfortunately, OCR cannot yet reliably identify the text in many real-world situations, such as the graphic labels on many products, a hand-written menu in a coffee shop, or even the street name on a street sign. Other popular products identify colors and read barcodes with similar performance (and prices). Filling the remaining void are a large number of human workers, volunteers, and friends who help blind people address remaining visual problems.

In this paper we introduce *VizWiz*, a project aimed at enabling blind people to recruit remote sighted workers to help them with visual problems in nearly real-time. Blind people use VizWiz on their existing camera phones. Users take a picture with their phone, speak a question, and then receive multiple spoken answers. Currently, answers are provided by workers on Amazon Mechanical Turk [1]. Prior work has demonstrated that such services need to work quickly [22], and so we have developed an approach (and accompanying implementation) called quikTurkit that provides a layer of abstraction on top of Mechanical Turk to intelligently recruit multiple workers before they are needed. In a field deployment, users had to wait just over 2 minutes to get their first answer on average, but wait times decreased sharply when questions and photos were easy for workers to understand. Answers were returned at an average cost per question of only $0.07 USD for 3.3 answers. Given that many tools in this domain cost upwards of $1000 USD (the equivalent of

[1]Many currencies other than the US Dollar are tactually distinguishable.

UIST'10, October 3-6, 2010, New York City, USA.

nearly 15,000 VizWiz uses), we believe nearly real-time human services can not only be more effective but also competitive with, or cheaper than, existing solutions. When set to maintain a steady pool of workers (at a cost of less than $5 per hour), VizWiz receives answers in *less than 30 seconds*.

Because VizWiz uses real people to answer questions, the scope of questions it can help answer is quite large, and, as opposed to automatic approaches, users can phrase questions naturally assuming an intelligent system (or person) will answer their question. For instance, OCR programs require users to carefully center the text they want to read in the camera's view, and then return all of the text. In contrast, VizWiz users can simply ask what they really want to know, for instance, "How much is the cheeseburger?" Because blind people cannot see the picture they are taking, pictures are often not framed well or have other problems. Real people can give guidance as to how to take a better picture. With real people answering questions, we can first target tools to what blind people want and then try to automate, rather than create tools according to what can currently be done automatically and hope they are good enough.

Contributions

Our contributions are the following:

- We motivate and present VizWiz, a talking mobile application that lets blind people take a picture, ask a question, and receive answers from remote workers in nearly real-time.
- We introduce *quikTurkit*, an approach and accompanying implementation that shows how workers can be intelligently recruited in advance to reduce latency.
- We show that VizWiz can work in real-world situations and we demonstrate some of the primary problems that blind people have taking photographs in a field deployment.
- We introduce VizWiz::LocateIt, a VizWiz extension that combines remote human-powered vision with automatic computer vision, illustrating how VizWiz facilitates participatory design with prototype tools which cannot yet be made fully automatic.

MOTIVATING SURVEY

We first recruited 11 participants (5 female) to get an idea of how blind people might want to use a system like VizWiz. We presented the idea of VizWiz, and asked participants how they could imagine using it. Participants wanted to read text in a variety of situations - road signs, cafe menu boards, tags on clothes, and cooking instructions on a packaged meal. Several participants wanted information about their physical appearance. For instance, how does this outfit look or is this shirt stained or discoloured? Many participants just wanted feedback on how good the picture was that they took or general information on the photos they have taken. When asked how they solve problems like these now, participants mentioned various strategies for specific situations and how they fallback to asking a sighted friend. One participant said that VizWiz could be "...very useful because I get so frustrated when I need sighted help and no one is there."

RELATED WORK

VizWiz builds from prior work in (i) talking mobile devices designed for blind people and (ii) the use of remote humans as part of computational processes.

Mobile Devices for Blind People

Most mainstream cellphones are not accessible to blind people. Smartphones often provide the best access through separate screen reading software like Mobile Speak Pocket (MSP) [25]. Though somewhat popular, the uptake of such software among blind users has been limited due to its high price (an additional $500 after the cost of the phone). Google's Android platform and the Apple iPhone 3GS now include free screen readers [36, 6]. The iPhone has proven particularly popular among blind users, which motivated us to concentrate on it for VizWiz. Apple's stringent controls on the applications available on its online store and tighter integration of its screen reader (VoiceOver) with the operating system has resulted in a large number of accessible applications. Touchscreen devices like the iPhone were once assumed to be inaccessible to blind users, but well-designed, multitouch interfaces leverage the spatial layout of the screen and can even be preferred by blind people [12].

Applications for general-purpose smartphones are beginning to replace special-purpose devices, but blind people still carry devices such as GPS-powered navigation aids, barcode readers, light detectors, color identifiers, and compasses [13]. Some accessible applications that use the camera on existing phones include currency-reading applications and color identifiers [20, 37]. Because VizWiz connects users to real people, it can potentially answer all of the questions answerable by many costly special-purpose applications and devices.

Talking OCR Devices: Of particular interest to blind people is the ability to read text, which pervasively labels objects and provides information. Both the kNFBReader [16] and the Intel Reader [11] are talking mobile OCR tools. VizWiz has an advantage over tools such as these because humans can still read more text written in more variations than can automatic approaches. When OCR works, however, it is faster and can be used to transcribe large text passages. Human workers are slower but this may be partially offset by their ability to take instructions that require intelligence. For example, an OCR program can read an entire menu, but cannot be asked, "What is the price of the cheapest salad?"

Other Automatic Computer Vision for Mobile Devices: Several research projects and products expose automatic computer vision on mobile devices. Photo-based Question Answering enables users to ask questions that reference an included photograph, and tackles the very difficult problems of automatic computer vision and question answering [39]. Google Goggles enables users to take a picture and returns related search results based on object recognition and OCR [8]. Although these projects have made compelling progress, the state-of-the-art in automatic approaches is still far from being able to answer arbitrary questions about photographs.

Interfacing with Remote Services: Most mobile tools are implemented solely as local software, but more applications are starting to use remote resources. For instance, TextScout [33] provides an accessible OCR interface, and Talking Points delivers contextually-relevant navigation information in urban settings [7]. VizWiz also sends questions off for remote processing, and these services suggest that people are becoming familiar with outsourcing questions to remote services.

Human-Powered Services

VizWiz builds from prior work in using human computation to improve accessibility. The ESP Game was originally motivated (in part) by the desire to provide descriptions of web images for blind people [38]. The Social Accessibility project connects blind web users who experience web accessibility problems to volunteers who can help resolve them, but 75% of requests remain unsolved after a day [31]. Solona started as a CAPTCHA solving service, and now lets registered blind people submit images for description [29]. According to its website, "Users normally receive a response within 30 minutes." VizWiz's nearly real-time approach could be applied to other problems in the accessibility space.

Prior work has explored how people ask and answer questions on their online social networks [26]. While answers were often observed to come back within a few minutes, response time varied quite a lot. The "Social Search Engine" Aardvark adds explicit support for asking questions to your social network, but advertises that answers come back "within a few minutes." [28] VizWiz and quikTurkit explore how to use microtask marketplaces like Amazon's Mechanical Turk to get answers back even faster.

Mechanical Turk makes outsourcing small paid jobs practical [1] and has been used for a wide variety of purposes, including large user studies [15], labeling image data sets [30], and determining political sentiments in blog snippets [10]. Amazon Remembers lets users take pictures of objects and later emails links to similar products that Amazon sells [2]. It is widely suspected that Amazon outsources some questions to Mechanical Turk. The TurKit library on which our quikTurkit is built lets programmers easily employ multiple turkers using common programming paradigms [19]. To our knowledge, quikTurkit is the first attempt to get work done by web-based workers in nearly real-time.

Connecting Remote Workers to Mobile Devices

Some human-powered services provide an expectation of latency. ChaCha and KGB employees answer questions asked via the phone or by text message in just a few minutes [5, 14]. VizWiz often provides answers faster, although the information necessary to answer a VizWiz question is embedded in the photo, whereas ChaCha and KGB are often used to ask questions that might require a web search. Other common remote services include relay services for deaf and hard of hearing people (which requires trained employees) [27], and the retroactive nearly real-time audio captioning by dedicated workers in Scribe4Me [23]. A user study of Scribe4Me found that participants felt waiting the required 3-5 minutes was too long because it "leaves one as an observer rather than an active participant." The VizWiz living laboratory of non-expert workers may help explore the perceived time sensitivity of visual questions versus audio questions.

Existing Use of Photos and Video for Assistance: Several of the blind consultants whom we interviewed mentioned using digital cameras and email to informally consult sighted friends or family in particularly frustrating or important situations (e.g., checking one's appearance before a job interview). LookTel is a soon-to-be-released talking mobile application that can connect blind people to friends and family members via a live video feed [21]. Although future versions of VizWiz may similarly employ video, we chose to focus on photos for two reasons. First, mobile streaming is not possible in much of the world because of slow connections. Even in areas with 3G coverage, our experience has been that the resolution and reliability of existing video services like UStream [35] and knocking [17] is too low for many of the questions important to blind people. Second, using video removes the abstraction between user and provider that VizWiz currently provides. With photos, questions can be asked quickly, workers can be employed for short amounts of time, and multiple redundant answers can be returned. As we will see, our field deployment showed that the low latency of VizWiz still supports some of the back and forth between user and worker that video would directly support.

VIZWIZ

VizWiz is an iPhone application designed for use with the VoiceOver screen reader included on the iPhone 3GS. Its wizard interface (Figure 1) guides users through taking a picture, speaking a question that they would like answered about the picture, and receiving answers from remote workers.

When users start VizWiz, it starts to ping its remote server to indicate that a question may be asked soon, so that the server can start recruiting workers if necessary. Once the picture is taken and the question is asked, VizWiz sends these to the remote server. The picture is compressed on the phone in a background process while the sound is being recorded to reduce the latency required to send the image. Sound is recorded in mp4 format using hardware compression on the phone. When the server receives the question and image, it calls a speech recognition service (currently the Windows Speech Recognition Engine) to convert the question to text, and adds the image and question to its database. Importantly, speech recognition does not need to work (we found it highly unreliable) because workers can also listen to the original question in an included Flash player.

quikTurkit, which is described in detail in the next section, is tasked with maintaining a pool of workers to answer questions. Recruited workers are shown a web page that presents the image and the text of the recognized question, and plays the original question (Figure 1). Workers are required to answer multiple previously-asked questions to keep workers around long enough to possibly answer new questions.

quikTurkit

quikTurkit denotes a general approach for achieving low- latency responses from Mechanical Turk along with a specific script that we have developed for this purpose[2]. quikTurkit is an abstraction layer on top of the TurKit Mechanical Turk API [19]. quikTurkit primarily achieves low latency by queuing workers before they are needed, but also encapsulates heuristics to help its Human Intelligance Tasks (HITs) be quickly picked up by workers.

The chart below illustrates the components of the time for a single worker to answer a question. The total time that a worker spends equals the sum of the time for the worker to find the posted task t_r plus the time to answer each question

[2]quikTurkit is available at quikturkit.googlecode.com

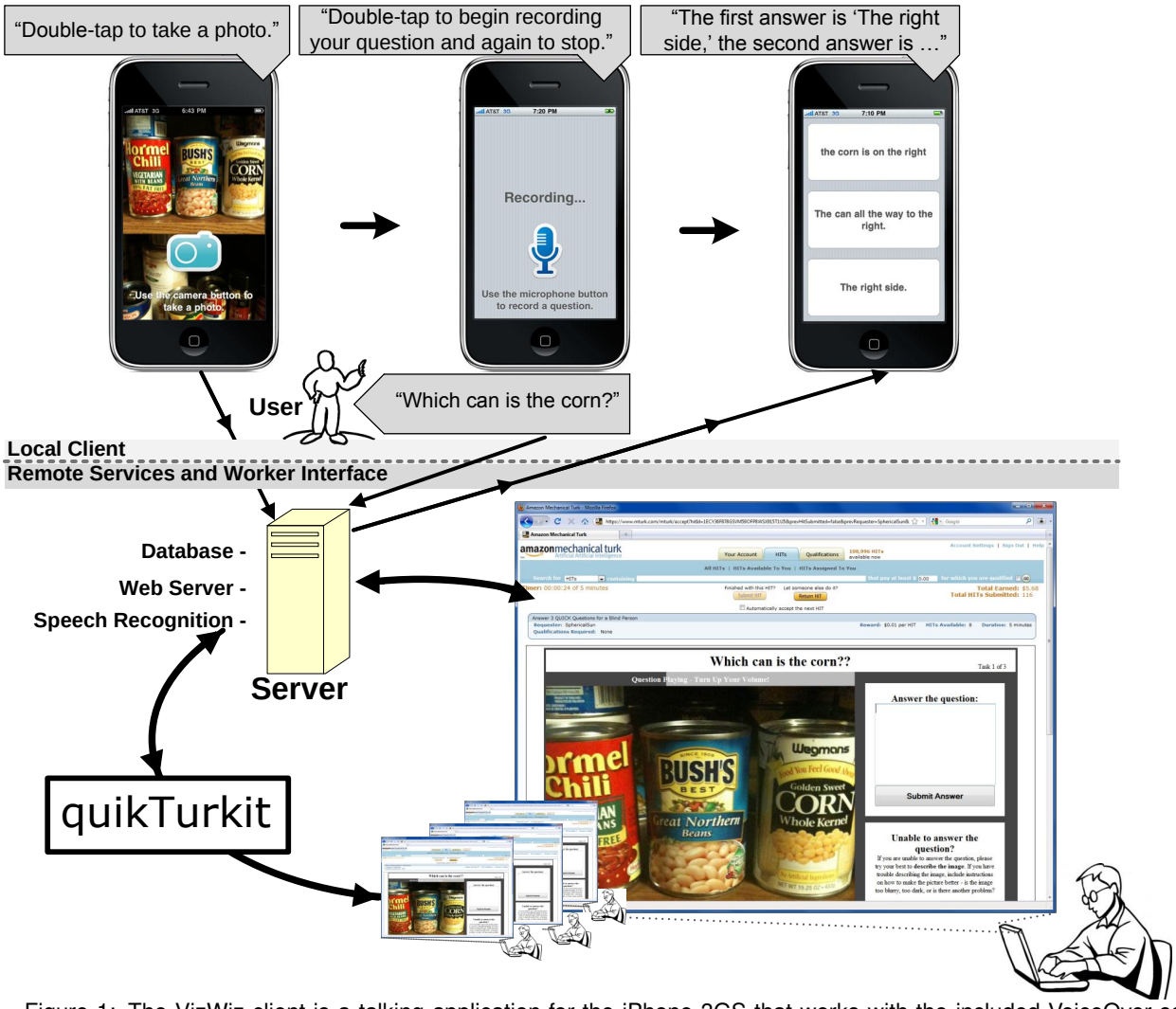
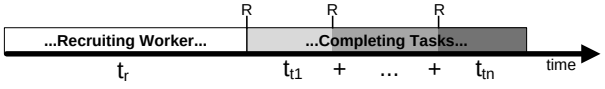


Figure 1: The VizWiz client is a talking application for the iPhone 3GS that works with the included VoiceOver screen reader. VizWiz proceeds in three steps—taking a picture, speaking a question, and then waiting for answers. System components include a web server that serves the question to web-based workers, a speech recognition service that converts spoken questions to text, and a database that holds questions and answers. quikTurkit is a separate service that adaptively posts jobs (HITs) to Mechanical Turk in order to maintain specified criteria (for instance, a mininum number of answers per question or a pool of waiting workers of a given size).

in a batch of n questions ($t_{t1}+...+t_{tn}$). quikTurkit has workers complete multiple tasks to engage the worker for longer, ideally keeping them around until a new question arrives. In the following figure, points marked with an R represent when the worker is available to answer a new question.



To use quikTurkit, requesters create their own web site on which Mechanical Turk workers answer questions. The interface created should allow multiple questions to be asked so that workers can be engaged answering other questions until they are needed. VizWiz currently has workers answer three questions. Importantly, the answers are posted directly to the requester's web site, which allows answers to bypass the Mechanical Turk infrastructure and answers to be returned before an entire HIT is complete. As each answer is submitted (at the points labeled R above), the web site returns the question with the fewest answers for the worker to complete next, which allows new questions to jump to the front of the queue. Requests for new questions also serve to track how many workers are currently engaged on the site. If a worker is already engaged when the k^{th} question is asked, then the time to recruit workers t_r is eliminated and the expected wait time for that worker is $\frac{t_{t(k-1)}}{2}$.

Recruiting workers before they are needed: quikTurkit allows client applications to signal when they believe new work may be coming for workers to complete. VizWiz signals that new questions may be coming when users begin taking a picture. quikTurkit can then begin recruiting workers and keep them busy solving the questions that users have asked previously. This effectively reduces t_r by the lead time given by the application, and if the application is able to signal more than t_r in advance, then the time to recruit is removed from

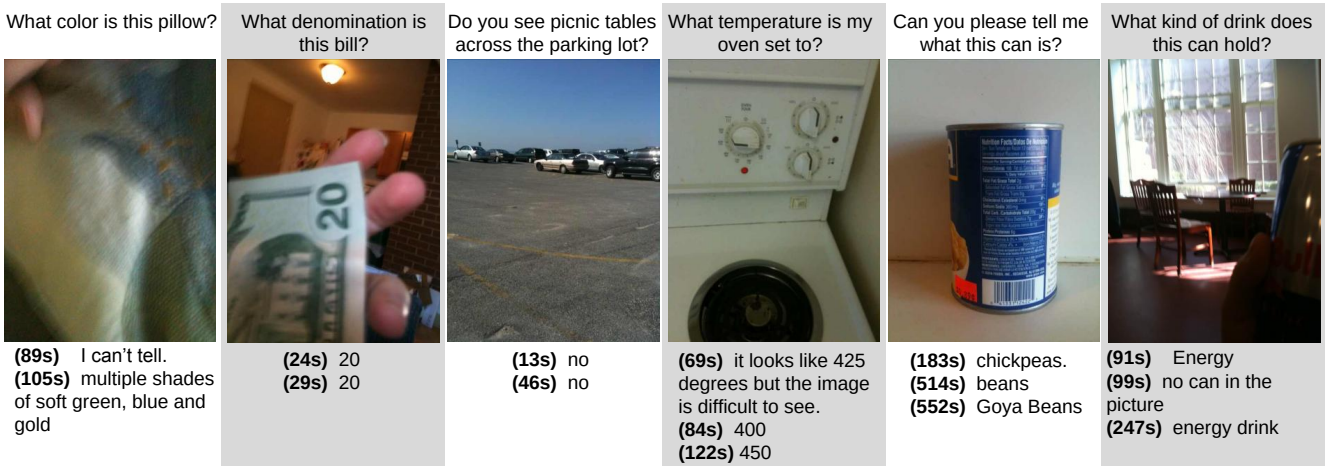


Figure 2: Six questions asked by participants, the photographs they took, and answers received with latency in seconds.

the total time required to answer a question. quikTurkit also makes it easy to keep a pool of workers of a given size continuously engaged and waiting, although workers must be paid to wait. In practice, we have found that keeping 10 or more workers in the pool is doable, although costly.

Most Mechanical Turk workers find HITs to do using the provided search engine[3]. This search engine allows users to view available HITs sorted by creation date, the number of HITs available, the reward amount, the expiration date, the title, or the time alloted for the work. quikTurkit employs several heuristics for optimizing its listing in order to obtain workers quickly. First, it posts many more HITs than are actually required at any time because only a fraction will actually be picked up within the first few minutes. These HITs are posted in batches, helping quikTurkit HITs stay near the top. Finally, quikTurkit supports posting multiple HIT variants at once with different titles or reward amounts to cover more of the first page of search results.

VizWiz currently posts a maximum of 64 times more HITs than are required, posts them at a maximum rate of 4 HITs every 10 seconds, and uses 6 different HIT variants (2 titles $\times$ 3 rewards). These choices are explored more closely in the context of VizWiz in the following section.

FIELD DEPLOYMENT

To better understand how VizWiz might be used by blind people in their everyday lives, we deployed it to 11 blind iPhone users aged 22 to 55 (3 female). Participants were recruited remotely and guided through using VizWiz over the phone until they felt comfortable using it. The wizard interface used by VizWiz speaks instructions as it goes, and so participants generally felt comfortable using VizWiz after a single use. Participants were asked to use VizWiz at least once a day for one week. After each answer was returned, participants were prompted to leave a spoken comment.

quikTurkit used the following two titles for the jobs that it posted to Mechanical Turk: "3 Quick Visual Questions" and "Answer Three Questions for A Blind Person." The reward distribution was set such that half of the HITs posted paid $0.01, and a quarter paid $0.02 and $0.03 each.

Asking Questions Participants asked a total of 82 questions (See Figure 2 for participant examples and accompanying photographs). Speech recognition correctly recognized the question asked for only 13 of the 82 questions (15.8%), and 55 (67.1%) questions could be answered from the photos taken. Of the 82 questions, 22 concerned color identification, 14 were open ended "what is this?" or "describe this picture" questions, 13 were of the form "what kind of (blank) is this?," 12 asked for text to be read, 12 asked whether a particular object was contained within the photograph, 5 asked for a numerical answer or currency denomination, and 4 did not fit into these categories.

Problems Taking Pictures 9 (11.0%) of the images taken were too dark for the question to be answered, and 17 (21.0%) were too blurry for the question to be answered. Although a few other questions could not be answered due to the photos that were taken, photos that were too dark or too blurry were the most prevalent reason why questions could not be answered. In the next section, we discuss a second iteration on the VizWiz prototype that helps to alert users to these particular problems before sending the questions to workers.

Answers Overall, the first answer received was correct in 71 of 82 cases (86.6%), where "correct" was defined as either being the answer to the question or an accurate description of why the worker could not answer the question with the information contained within the photo provided (i.e., "This image is too blurry"). A correct answer was received in all cases by the third answer.

The first answer was received across all questions in an average of 133.3 seconds (SD=132.7), although the latency required varied dramatically based on whether the question could actually be answered from the picture and on whether the speech recognition accurately recognized the question (Figure 4). Workers took 105.5 seconds (SD=160.3) on average to answer questions that could be answered by the provided photo compared to 170.2 seconds (SD=159.5) for those

[3]Available at mturk.com

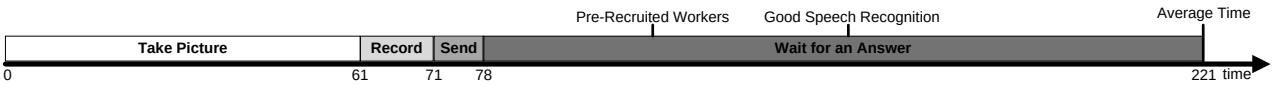


Figure 3: Average time required to take a picture, record a question, send it, and receive an answer for three different cases. The first 78 seconds (on average) is lead time during which workers can be recruited.

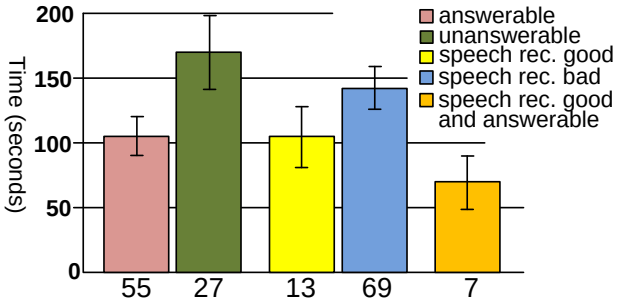


Figure 4: Time to first answer for questions which were answerable, which were not answerable, on which the speech recognition succeeded, on which speech recognition failed, and which were both answerable and speech recognition succeeded. The x axis reports the number of questions in each category, and error bars shows standard error.

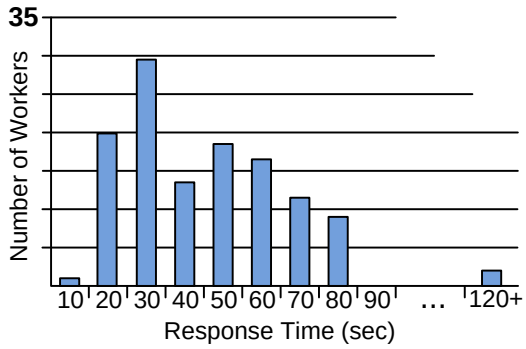


Figure 5: A histogram of the 122 workers who answered at least 5 queries based on the mean time they required per question.

that could not, which was a statistically significant difference ($F_{1,81}$=8.01, p<.01). The 13 questions for which the speech recognition succeeded took 104.5 seconds on average (SD=81.9) to answer compared with 142.7 seconds (SD=142.1) for questions on which the speech recognition failed. This difference was not detectably significant, perhaps because speech recognition succeeded so rarely and the time to answer was heavily influenced by whether the question was answerable from the photo. If we instead consider only those instances in which the question asked could be answered from the photo and the speech recognition succeeded, the average response time was 67.1 seconds (SD=57.1), which is significantly different from the photos in this group on which speech recognition failed ($F_{1,54}$=4.32, p<.05).

quikTurkit was set to ensure that each question received at least two answers (it continued to post HITs until this was true). Since many extra HITs were posted to reduce response time, more answers were often received. On average, participants received 3.3 (SD=1.8) answers for each question asked at a total cost of $0.07 USD, indicating that quikTurkit may

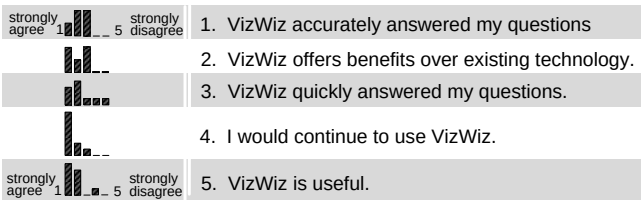


Figure 6: Answers to Likert scale questions on our survey indicating that participants found VizWiz useful (1, 3, 5) and wanted to continue using it (4).

be more aggressive. Future work will look to fine-tune the tradeoff between getting answers back quickly and keeping the number of responses low to minimize cost.

Of the 589 unique workers who answered questions, the average response time per question was 36 seconds (SD=30) for a total of 3987 answers. To complete the HIT, workers answered 3 questions and so we expect most workers to be engaged for 108 seconds, but able to answer new questions for only 72 seconds. As a consequence, we can expect to have to wait 18 seconds on average until an already-engaged worker is free to answer a query and an additional 36 seconds for that worker's answer. If we limit to those workers who answered at least 5 questions, the fastest worker had an average response time of 7 seconds and the slowest 94 seconds (Figure 5). Recruiting multiple workers can reduce the expect wait time as we will see in the next section, although we could alternatively recruit faster workers or encourage workers to respond more quickly.

Post-Deployment Survey Ten participants completed a survey about their experiences (Figure 6). We asked how much participants would be willing to pay for the service if it were offered as a monthly subscription and the average was just over $5, although two participants said they would prefer to pay $0.05 per use. We also asked what participants did while waiting for answers to return. For the most part, they just waited. Several participants suggested that we use push notifications to alert them of their answer later, so they did not have to actively wait. Participants also relayed their frustration with being unable to take good pictures, and often blamed the quality of the iPhone's camera. One participant suggested that using other phones with higher-quality cameras and even a flash may alleviate many of the photo-quality problems. Nevertheless, participants seemed uniformly excited about the potential of VizWiz - one said, "I would love for VizWiz to be made publicly available!"

VIZWIZ VERSION 2

A common problem experienced by participants in our field deployment was that the photos they took were either too dark or too blurry. To help identify these problems before questions were sent, we implemented both darkness and blur detection on the iPhone. Darkness detection takes prece-

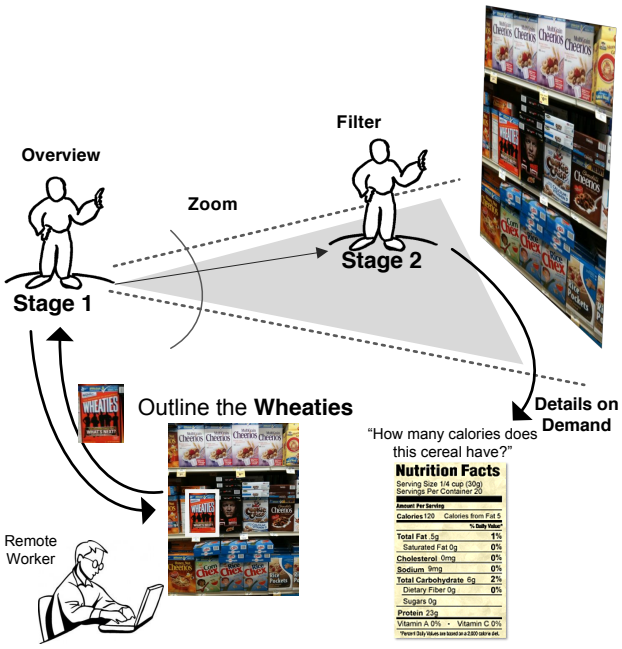


Figure 7: **VizWiz::LocateIt**. A user first takes an overview picture and asks a localization question. A remote worker outlines the target object. LocateIt uses this information to orient the user to move toward (zoom) the object. Once the user is close to the object, LocateIt uses computer vision to help the user filter and identify the target object. Optionally, the user can ask additional questions about the object.

dence and works by creating a grayscale histogram to check for any large concentrations of dark pixels indicating insufficient lighting. Blur is estimated by computing the mean and standard deviation of an image from its binary map and evaluating these values using a set of pre-built covariance matrices created from images known to be blurry or sharp [18]. At the user interface level, users are warned by way of a modal dialog box after taking a picture that it might be too blurry or too dark, and are given the option to either retake the picture or to continue. It is important to note that neither darkness nor blur detection can work all the time, such as when the user takes a picture of a black sheet and asks its color.

We deployed VizWiz 2.0 to three of the study participants who participated in the previous field deployment and asked them to use five times in just one day. During the study period, we also set quikTurkit to maintain a worker pool of 8 workers so that participants would not have to wait while workers were recruited to answer their questions, but otherwise all settings were the same as in our prior field deployment. During the 24 hour period of the study, quikTurkit maintained between 4 and 10 workers in its pool.

Our participants asked a total of 15 questions during the one day study period, and received a correct answer back in an average of 27.0 seconds (SD=19.5). In the three cases in which speech recognition worked and the question was answerable, answers were returned even faster in an average

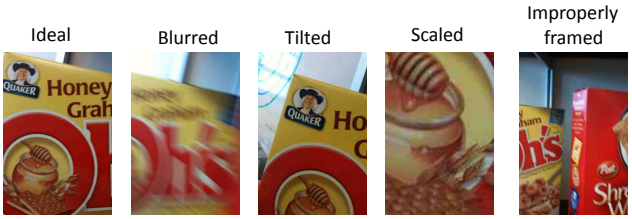


Figure 8: Frames captured by blind users during the Filter stage exemplifying computer vision challenges.

of 18 seconds (SD=1). Maintaining our pool of workers cost $4.67 USD per hour on average. Although pricey for a single individual, this cost could be shared by many VizWiz users. As this was an initial experiment, these results should be seen as an upper bound. As more people use VizWiz, it naturally converges to engaging a number of workers simultaneously despite not explicitly maintaining this pool.

VIZWIZ::LOCATEIT

While VizWiz has demonstrated its potential to support a wide variety of visual questions important to blind users, we found that certain types of identification questions are actually motivated by users' desire to *locate* a particular object. For example, suppose there are several soup cans on the shelf. A blind person may serially ask for the label of one can after another until they hear the name (e.g., tomato soup), thereby locating it. It would be more efficient if this question could be phrased as a single localization quesiton such as "Where is the tomato soup can?" instead of as a series of identification questions. In this section, we present our work on VizWiz::LocateIt, a prototype system that combines the VizWiz approach with automatic computer vision to help blind people locate arbitrary items in their environments.

System Description

We follow the information visualization mantra—overview, zoom and filter, and details on demand—to develop a system for helping blind users visualize their environment and locate objects within it (see Figure 7). Overview and details on demand are already supported by VizWiz. We created two extensions to VizWiz to support object localization: a web interface to let remote workers outline objects, and the VizWiz::LocateIt subsystem to the mobile interface consisting of the *Sensor* (zoom and filter) and *Sonification* modules.

Sensor Module: In the zoom stage (stage 1), the Sensor module estimates how much the user needs to turn in the direction of the target object (left, right, up, or down). It first uses the object's image location (u, v) indicated by the remote worker to calculate the 3D position (x, y, z) of the object relative to the user's current position. The construction of such a mapping function typically requires knowledge of a set of camera parameters that are extrinsic (e.g., camera orientation) and intrinsic (e.g., focal length, lense distortion). We estimate intrinsic parameters by camera calibration once per device and compute extrinsic camera parameters directly from the device's built-in compass (heading angle) and accelerometer (gravity vector) once per camera movement. Note that extrinsic parameters change whenever the camera moves whereas the intrinsic parameters stay constant and only need to be computed once per device. Once the

3D position of the target object is known, we can also compute the 3D position (x', y', z') toward which the camera is currently pointing using a similar procedure. The angular cosine distance between the two resulting vectors indicates how much the user needs to turn. This difference is measured as an angular cosine distance, and is passed to the Sonification module to generate appropriate audio cues.

In the filter stage (stage 2), the Sensor module uses computer vision to determine how close the current camera view is to the object outlined by the remote worker. This task is non-trivial because input images are often blurred, tilted, varied in scale, and improperly framed due to blind users being unable to see the image they are capturing (Figure 8). We implemented two visual matching schemes based on invariant local speeded up robust features (SURF) and color histograms respectively. In the first scheme, a homography between each captured frame and the overview image is computed based on the correspondences of SURF features [3]. Based on the homography, the object's location (u, v) in the overview image is mapped to a location (u', v') in the current frame. The distance between (u', v') and the center of the current frame is computed. The smaller the distance, the more "centered" the target object is in the current frame.

We found that local features were quite susceptible to problems related to lighting and blur and so we also created a visual matching scheme based on color histograms that is more robust to these problems. A color histogram h of the object outlined by the remote helper is computed. Then we divide each input frame up into N blocks and compute a color histogram h_i for each block i, which improves robustness to improper framing. We then compare the computer histogram to the target color histogram h, and calculate a distance measure d_i using L1. The total distance D is the minimum distance of contiguous subsets of the N individual block differences. The smaller the D, the more "similar" the object in the current frame is to the target object. To provide users with a consistent sense of distance, the distance is normalized by the smallest D observed during the k most recent interactions with the system. The normalized distance is then passed to the Sonification module to generate audible feedback.

Sonification Module: The Sonification Module inputs computed distances from the *Sensor* component and generates audio feedback to inform the user how close she is to the goal. In the zoom stage, the goal is a particular direction to "zoom in" (i.e., walk closer to). In the filter stage, the goal is a particular object for which we have implemented three different sonification options. The first two are based on the pitch of a tone and frequency of clicking, respectively. The third scheme is a voice that announces a number between one and four, which maps to how close the user is to the goal.

User Study Setup

We conducted a within-subjects lab-based user study in which we asked participants to find a desired cereal box using (i) LocateIt (color-histogram version) and (ii) a commercially-available barcode scanner with a talking interface (Figure 9(b)). We prepared three shelves each with five cereal boxes

(a) (b)

Figure 9: (a) Our mock grocery store shelf stocked with 15 different cereal boxes. (b) The ID Mate II talking barcode scanner from Envision America.

(Figure 9(a)). All cereal boxes were unique and unopened[4], and they reflected a diversity of sizes, colors, and weights. We recruited seven participants (two females, four totally blind, three low vision) aged 48 years on average (SD=8.7). One participant owned an iPhone, four others had experience with an iPhone, and five had previously taken photographs on inaccessible cell phone cameras either for taking photos of family and friends or for recognizing text.

Participants were trained using both methods (approximately 10 minutes). Participants then completed three timed trials using each method. For the LocateIt trials, the zoom and filter stages were timed separately. For the purposes of this study, researchers answered requests in order to concentrate on the user interaction with the application, although our experience has been that workers on Mechanical Turk can quickly answer questions requiring them to outline objects. For all six trials, participants started 10 feet in front of the shelves, and boxes were randomized after each trial.

Study Results

Participants used LocateIt and the barcode scanner in very different ways. LocateIt enabled users to zero in on the right part of the shelf much like a visual scan, whereas the barcode scanner required them to serially scan each box. The time required for each tool was similar, although LocateIt produced many more errors. LocateIt took an average of 92.2 seconds (SD=37.7) whereas the barcode scanner took an average of 85.7 seconds (SD=55.0), although the researchers answered questions in approximately 10 seconds as compared to the almost 30 seconds that we would expect workers on Mechanical Turk to require. Participants found the correct box in all cases using the barcode scanner (since it clearly spoke the name of each box), whereas using LocateIt participants found the correct box on their first try in 12 of 21 cases and in their second try in 7 out of 21 cases.

Interestingly, the zoom stage of LocateIt correctly led users to the correct area of the wall in only 30.7 seconds on average (SD=15.9). We informally tried using the first stage of LocateIt to direct users to approximately the right part of the wall, and then had them switch to the barcode scanner for identification. This ended up being slower, primarily because of how cumbersome it was to switch between devices.

[4]Until the end of the study.

In future work, we will explore how to better integrate both human-powered and automatic services together. For example, Trinetra connects a portable barcode reader to a phone via Bluetooth [34]. None of the participants wanted to carry around a bulky barcode reader, or even a smaller portable one, because of their high prices and inconvenience. All participants said, however, that they would use an accessible barcode reader on their phone if one was available.

In summary, our first LocateIt prototype was comparable to barcode scanner in terms of task completion time but produced more errors. However, LocateIt is useful for general visual search problems, does not require objects to be tagged in advance, and may scale better. From the observations and results we draw three lessons related to cues and orientation that will inform this work as we go forward:

External Cues Participants used many cues other than the audio information from LocateIt and the barcode scanner, including shaking the boxes, having prior knowledge of box size, or using colors (low vision participants).

Interaction Cues All participants liked the clicks used in the zoom stage of our application. For the second stage, many alternatives were brought up, including vibration, pitch, more familiar sounds (e.g., chirps and cuckoos crosswalk signal sounds), verbal instructions, or a combination of output methods, many of which are used in other applications for blind people [9, 32, 24].

Phone Orientation and Space Three participants had difficulty walking in a straight line from their beginning position to the shelf once a direction was indicated, desiring a more continuous noise to keep them on track. Participants also experienced difficulties keeping the phone perpendicular to the ground. In the up-close stage, all fully blind participants had trouble judging how far back from each cereal box they should hold the phone and framing each cereal box.

DISCUSSION

VizWiz enables visual questions to be answered in nearly real-time by recruiting multiple workers from existing online marketplaces. Research often focuses on automatic approaches to addressing the problems that blind people face, but VizWiz suggests that balancing human and automatic services may be not only more effective but also cheaper. Nevertheless, blind people have been living their lives without VizWiz, and it is unrealistic for them to come to rely on it in a week-long study. They label food cans or put them in a known place, fold money by denomination, and keep clothes in matching sets. VizWiz is useful when plans break down and may eventually reduce the need for prior preparation.

quikTurkit can already scale using only Mechanical Turk. The pool of workers in our 2nd deployment cost $4.67/hour resulting in 700 answers/hour, so VizWiz can already handle many more users. By raising the number of tasks per HIT from 3 to 5-10 or by increasing the reward, quikTurkit could get even more answers. The price per user goes down as more users ask questions. Human services introduce new concerns that will need to be considered as they are adopted for nearly real-time purposes.

A Human Is Answering My Question: Because participants knew that humans were answering their questions, they often built a requirement for human intelligence into their queries. For instance, one participant asked "What color is this neck pillow?" The supplied picture contained two pillows of different colors, but the participant named the pillow for which she wanted the color. As users become accustomed to conversing with humans, transitioning to imperfect automatic techniques might be more difficult.

Interestingly, however, several participants reported frustration stemming from the fact that a person was answering their questions. One participant said he "need(ed) to have more feedback with the person" and that "conversations while working with the item in question would make the service much more useful." It is unclear if participants would have felt this same frustration had VizWiz been an automatic service. Current automatic services often fail, but they tend to do so much more quickly so that users can try again quickly.

Workers on Mechanical Turk often provided feedback to the user on how to improve the picture they took, and participants clearly wanted more interactive collaboration with remote workers. This feedback can be expressed in different ways, and we plan to investigate options for facilitating efficient and usable information exchanges.

VizWiz::LocateIt: VizWiz::LocateIt combines automatic and human-powered computer vision, effectively offloading the vision not yet possible to do automatically to humans, while retaining the benefit of quick response times offered by automatic services. This allowed us to prototype an interaction that would not have been possible otherwise and easily begin a participatory design process to see if this type of interaction is desirable or even useful, highlighting the potential of the VizWiz approach to influence early designs.

FUTURE WORK

VizWiz demonstrates a new model for assistive technology in which human workers assist users in nearly real-time. VizWiz currently targets assisting blind and low vision users in the real world, but future work may explore how to extend these benefits to other domains (like the web) or to other populations. One compelling future direction is to use the VizWiz approach to help reduce the latency of transcription and description of audio for deaf and hard of hearing individuals. More generally, we believe that low-cost, readily-available human computation can be applied to many problems.

Expanding to New Worker Pools Although workers are currently recruited from Mechanical Turk, a rich area for further work is to consider recruiting workers from elsewhere. One option may be to send questions to one's social network [26] or even to more personal contacts via picture text messaging. At first, these options may be seen as preferable for those asking sensitive questions, but users may also prefer the relative anonymity of sending their questions to workers on the web whom they do not know.

Expanding to new services may also reduce the cost of using VizWiz. We have shown that we can get humans to answer questions about photographs in nearly real-time, but can

we get users to answer questions in nearly real-time using a game, on a volunteer site, or via text messages for their friends? Techniques introduced in VizWiz may be adapted to new sources of workers. For instance, workers likely need to be recruited in advance for low latency and interfaces need to consider how to encourage fast responses when questions cannot be answered or are ambiguous. Many variables influence response times that could be studied in future work.

Improved Interfaces for Blind People In future work, we plan to study in more depth how to best enable blind people to take pictures appropriate for the questions they seek to ask. This is of general interest, as many blind people want to share photographs with friends and family, just like everyone else. Taking pictures, and in particular framing and focusing photographs can be difficult. This, however, has not stopped blind photographers from taking and sharing photographs [4]. We might be able to provide software support to help them take pictures more easily.

Improved Response with Automatic Services Due to inherent human delay, human-powered services will never be as fast as completely automatic services. We plan to augment the human-powered services of VizWiz with automatic approaches. For instance, a tool for blind people could always run an OCR program over submitted photographs and only ask humans to describe the image if no text was found. We hope that by building services like VizWiz which enable blind people to ask questions that are not yet possible to answer with automated approaches, we might help motivate research in supporting the types of questions that blind people actually want answered.

CONCLUSION

We have presented VizWiz, a talking mobile application that enables blind people to solve visual problems in nearly real-time by recruiting multiple human workers for small amounts of money. VizWiz is directly inspired by how blind people overcome many accessibility shortcomings today—ask a sighted person—but our approach keeps users in control and allows questions to be asked whenever needed. VizWiz is useful as both a tool for answering visual questions and as a general approach for prototyping new tools before the necessary automatic computer vision has been developed.

REFERENCES

1. Amazon Mechanical Turk. http://www.mturk.com/. 2010.
2. Amazon Remembers. http://www.amazon.com/gp/. 2010.
3. Bay, H., A. Ess, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. *Proc. of CVIU 2008*, v. 110, 346–359, 2008.
4. Blind with camera: Changing lives with photography. http://blindwithcamera.org/. 2009.
5. Chacha. http://www.chacha.com/. 2010.
6. Eyes-free. http://code.google.com/p/eyes-free/. 2010.
7. Gifford, S., J. Knox, J. James, and A. Prakash. Introduction to the talking points project. *Proc. of ASSETS 2008*, 271–272, 2008.
8. Google Goggles, 2010. http://www.google.com/mobile/goggles/.
9. Hong, D., S. Kimmel, R. Boehling, N. Camoriano, W. Cardwell, G. Jannaman, A. Purcell, D. Ross, and E. Russel. Development of a semi-autonomous vehicle operable by the visually-impaired. *IEEE Intl. Conf. on Multisensor Fusion and Integration for Intelligent Systems*, 539–544, 2008.
10. Hsueh, P., P. Melville, and V. Sindhwani. Data quality from crowdsourcing: a study of annotation selection criteria. *Proc. of the HLT 2009 Workshop on Active Learning for NLP*, 27–35, 2009.
11. Intel reader. http://www.intel.com/healthcare/reader/. 2009.
12. Kane, S. K., J. P. Bigham, and J. O. Wobbrock. Slide rule: making mobile touch screens accessible to blind people using multi-touch interaction techniques. *ASSETS 2008*, 73–80, 2008.
13. Kane, S. K., C. Jayant, J. O. Wobbrock, and R. E. Ladner. Freedom to roam: a study of mobile device adoption and accessibility for people with visual and motor disabilities. *ASSETS 2009*, 115–122, 2009.
14. KGB, 2010. http://www.kgb.com.
15. Kittur A., E. H. Chi, and B. Suh. Crowdsourcing user studies with mechanical turk. In *Proc. of the SIGCHI Conf. on Human Factors in Computing Systems (CHI 2008)*, pages 453–456, 2008.
16. kNFB reader. knfb Reading Technology, Inc., 2008. http://www.knfbreader.com/.
17. Knocking live video. ustream, 2010. http://knockinglive.com/.
18. Ko, J., and C. Kim. Low cost blur image detection and estimation for mobile devices. *ICACT 2009*, 1605–1610, 2009.
19. Little, G., L. Chilton, M. Goldman, and R.C. Miller. TurKit: Human Computation Algorithms on Mechanical Turk. *UIST 2010*, 2010.
20. Liu, X. A camera phone based currency reader for the visually impaired. *ASSETS 2008*, 305–306, 2008.
21. Looktel, 2010. http://www.looktel.com.
22. Matthews, T., S. Carter, C. Pai, J. Fong, and J. Mankoff. Scribe4me: Evaluating a mobile sound transcription tool for the deaf. *UbiComp 2006*, 159–176, 2006.
23. Matthews, T., J. Fong, F. W.-L. Ho-Ching, and J. Mankoff. Evaluating visualizations of non-speech sounds for the deaf. *Behavior and Information Technology*, 25(4):333–351, 2006.
24. Miniguide us. http://www.gdp-research.com.au/minig_4.htm/.
25. Mobile speak screen readers. Code Factory, 2008. http://www.codefactory.es/en/products.asp?id=16.
26. Ringel-Morris, M., J. Teevan, and K. Panovich. What do people ask their social networks, and why? a survey study of status message q&a behavior. *CHI 2010*, 1739-1748, 2010.
27. Power, M. R., Power, D., and Horstmanshof, L. Deaf people communicating via sms, tty, relay service, fax, and computers in australia. *Journal of Deaf Studies and Deaf Education*, v. 12, i. 1, 2006.
28. Rangin, H.B. Anatomy of a large-scale social search engine. *WWW 2010)*, 431–440, 2010.
29. Solona, 2010. http://www.solona.net/.
30. Sorokin, A., and D. Forsyth. Utility data annotation with amazon mechanical turk. *CVPRW 2008*, 1–8, 2008.
31. Takagi, H., S. Kawanaka, M. Kobayashi, T. Itoh, and C. Asakawa. Social accessibility: achieving accessibility through collaborative metadata authoring. *ASSETS 2008*, 193–200, 2008.
32. Talking signs. http://www.talkingsigns.com/, 2008.
33. Testscout- your mobile reader, 2010. http://www.textscout.eu/en/.
34. Lanigan, P., A. M. Paulos, A. W. Williams, and P. Narasimhan. Trinetra: Assistive Technologies for the Blind Carnegie Mellon University, CyLab, 2006.
35. UStream. ustream, 2010. http://www.ustream.tv/.
36. Voiceover: Macintosh OS X, 2007. http://www.apple.com/accessibility/voiceover/.
37. voice for android, 2010. www.seeingwithsound.com/android.htm.
38. von Ahn, L., and L. Dabbish. Labeling images with a computer game. *CHI 2004*, 319–326, 2004.
39. Yeh, T., J. J. Lee, and T. Darrell. Photo-based question answering. *MM 2008*, 389–398, 2008.