VizLens: A Robust and Interactive Screen Reader for Interfaces in the Real World

Anhong Guo[1], Xiang 'Anthony' Chen[1], Haoran Qi[2], Samuel White[1], Suman Ghosh[1], Chieko Asakawa[3], Jeffrey P. Bigham[1]

[1]Human-Computer Interaction Institute, [2]Mechanical Engineering Department, [3]Robotics Institute
Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA 15213
{anhongg, xiangche, chiekoa, jbigham}@cs.cmu.edu, {haoranq, samw, sumang}@andrew.cmu.edu

ABSTRACT

The world is full of physical interfaces that are inaccessible to blind people, from microwaves and information kiosks to thermostats and checkout terminals. Blind people cannot independently use such devices without at least first learning their layout, and usually only after labeling them with sighted assistance. We introduce *VizLens* —an accessible mobile application and supporting backend that can robustly and interactively help blind people use nearly any interface they encounter. VizLens users capture a photo of an inaccessible interface and send it to multiple crowd workers, who work in parallel to quickly label and describe elements of the interface to make subsequent computer vision easier. The VizLens application helps users recapture the interface in the field of the camera, and uses computer vision to interactively describe the part of the interface beneath their finger (updating 8 times per second). We show that VizLens provides accurate and usable real-time feedback in a study with 10 blind participants, and our crowdsourcing labeling workflow was fast (8 minutes), accurate (99.7%), and cheap ($1.15). We then explore extensions of VizLens that allow it to *(i)* adapt to state changes in dynamic interfaces, *(ii)* combine crowd labeling with OCR technology to handle dynamic displays, and *(iii)* benefit from head-mounted cameras. VizLens robustly solves a long-standing challenge in accessibility by deeply integrating crowdsourcing and computer vision, and foreshadows a future of increasingly powerful interactive applications that would be currently impossible with either alone.

Author Keywords

Non-visual interfaces; visually impaired users; accessibility; crowdsourcing; computer vision; mobile devices.

ACM Classification Keywords

H.5.2 Information interfaces and presentation: User Interfaces - *Input devices and strategies*; K.4.2 Computers and Society: Social Issues - *Assistive technologies*.

UIST '16, October 16-19, 2016, Tokyo, Japan
 ISBN 978-1-4503-4189-9/16/10...$15.00
DOI: http://dx.doi.org/10.1145/2984511.2984518

INTRODUCTION

The world is full of inaccessible physical interfaces. Microwaves, toasters and coffee machines help us prepare food; printers, fax machines, and copiers help us work; and checkout terminals, public kiosks, and remote controls help us live our lives. Despite their importance, few are self-voicing or have tactile labels. As a result, blind people cannot easily use them. Generally, blind people rely on sighted assistance either to use the interface or to label it with tactile markings. Tactile markings often cannot be added to interfaces on public devices, such as those in an office kitchenette or checkout kiosk at the grocery store, and static labels cannot make dynamic interfaces accessible. Sighted assistance may not always be available, and relying on co-located sighted assistance reduces independence.

Making physical interfaces accessible has been a long-standing challenge in accessibility [11, 26]. Solutions have generally either involved *(i)* producing self-voicing devices, *(ii)* modifying the interfaces (*e.g.*, adding tactile markers), or *(iii)* developing interface- or task-specific computer vision solutions. Creating new devices that are accessible can work, but is unlikely to make it into all devices produced due to cost. The Internet of Things may help solve this problem eventually; as more and more devices are connected and can be controlled remotely, the problem becomes one of digital accessibility, which is easier to solve despite challenges. For example, users may bring their own smartphone with an interface that is accessible to them, and use it to connect to the device [10, 32]. Computer vision approaches have been explored, but are usually brittle and specific to interfaces and tasks [11]. Given these significant challenges, we expect these solutions will neither make the bulk of new physical interfaces accessible going forward nor address the significant legacy problem in even the medium term.

This paper introduces *VizLens*, a robust interactive screen reader for real-world interfaces (Figure 1). Just as digital screen readers were first implemented by interpreting the visual information the computer asks to display [31], VizLens works by interpreting the visual information of existing physical interfaces. To work robustly, it combines on-demand crowdsourcing and real-time computer vision. When a blind person encounters an inaccessible interface for the first time, he uses a smartphone to capture a picture of the device and then send it to the crowd. This picture then becomes a reference image. Within a few minutes, crowd workers mark the layout of the interface, annotate its elements (*e.g.*, buttons

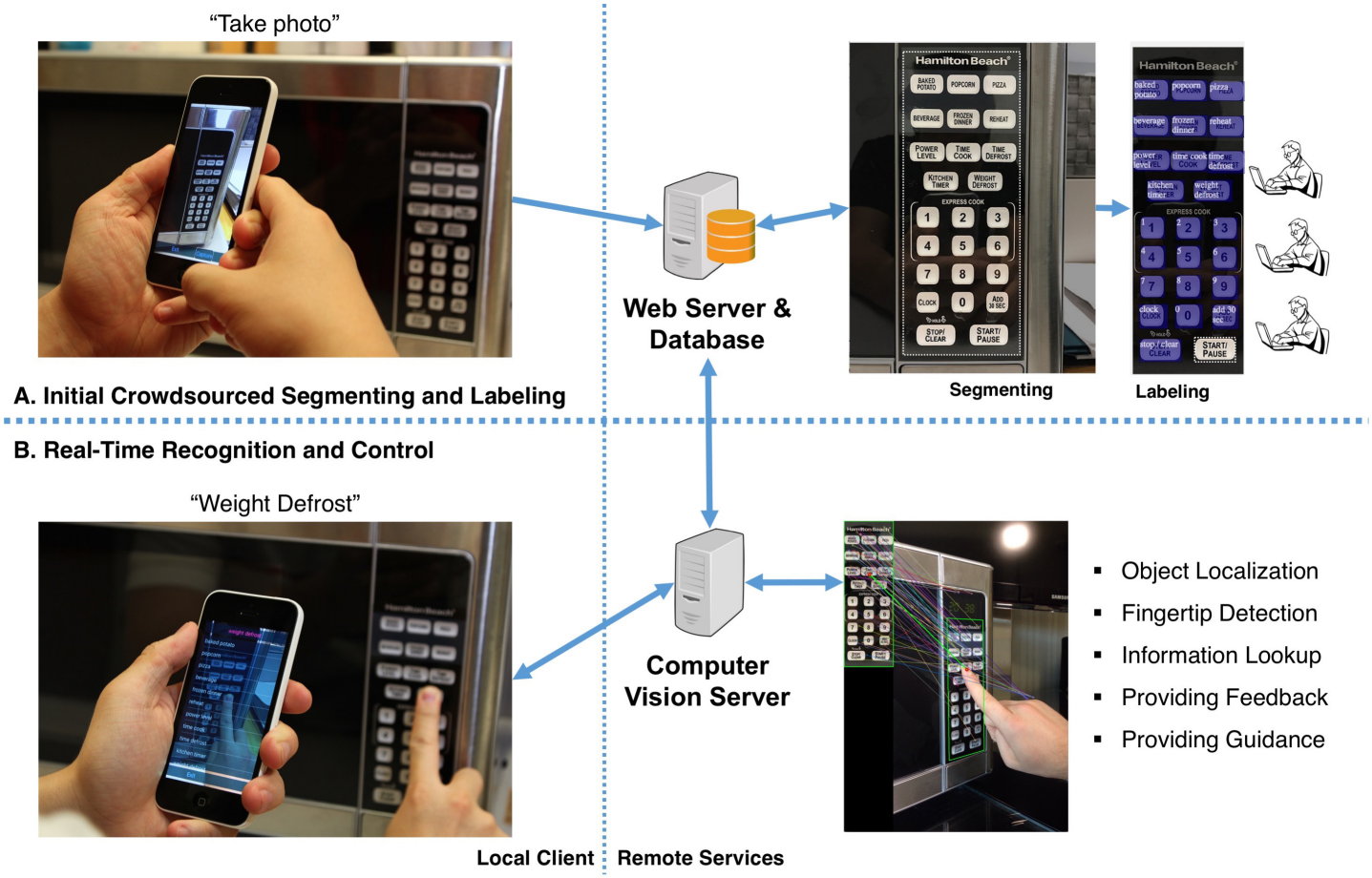


Figure 1. VizLens users take a picture of an interface they would like to use, it is interpreted quickly and robustly by multiple crowd workers in parallel, and then computer vision is able to give interactive feedback and guidance to users to help them use the interface.

or other controls), and describes each element (Figure 1A). Later, when the person wants to use the interface, he opens the VizLens application, points it toward the interface, and hovers a finger over it. Computer vision matches the crowd-labeled reference image to the image captured in real-time. Once it does, it can detect what element the user is pointing at and provide audio feedback or guidance (Figure 1B). With such instantaneous feedback, VizLens allows blind users to interactively explore and use inaccessible interfaces.

In a user study, 10 participants effectively accessed otherwise inaccessible interfaces on several appliances. Based on their feedback, we added functionality to adapt to interfaces that change state (common with touchscreen interfaces), read dynamic information with crowd-assisted Optical Character Recognition (OCR), and experimented with wearable cameras as an alternative to the mobile phone camera. The common theme within VizLens is to trade off between the advantages of humans and computer vision to create a system that is nearly as robust as a person in interpreting the user interface and nearly as quick and low-cost as a computer vision system. The end result allows a long-standing accessibility problem to be solved in a way that is feasible to deploy today.

This paper makes the following contributions:

- We introduce VizLens, a system that combines on-demand crowdsourcing and real-time computer vision to make real-world interfaces accessible.
- In a study with 10 visually impaired participants, we show that VizLens can provide useful feedback and guidance in assisting them accomplish realistic tasks that involve otherwise inaccessible visual information or interfaces.
- We show that our crowdsourcing labeling workflow is fast (8 minutes), accurate (99.7%), and cheap ($1.15). Once the reference image is prepared, VizLens provides accurate, real-time feedback across many different devices.
- We produce VizLens v2, which adapts to state changes in dynamic interfaces, combines crowd labeling with OCR technology to handle dynamic displays, and benefits from head-mounted cameras.

RELATED WORK

Our work is related to prior work on *(i)* making visual information accessible with computer vision, and *(ii)* crowd-powered systems for visual assistance.

Computer Vision for Accessibility

A number of systems have been developed to help blind people access visual information using computer vision. Specialized computer vision systems have been built to help blind people read the LCD panels on appliances [11, 26, 30]. These systems have tended to be fairly brittle, and have generally only targeted reading text and not actually using the interface. Because it uses crowdsourcing, VizLens can adapt fluidly to new interfaces it has not seen before.

Several prior systems have been developed to help blind people take better photographs, since acquiring a high-quality photograph is often a prerequisite for further computer vision processing [15, 25, 33, 35, 41]. One of the challenges with systems supporting "blind photography" is that it is often unclear what the user is trying to take a picture of. VizLens solves this problem by first having the crowd assist users in capturing a clear picture of the interface, which can then be recognized again later when the user is receiving assistance.

Many systems have been developed to help blind people read visual text via OCR [36]. For instance, the KNFB Reader [18] is a popular application for iOS that helps users frame text in the camera's view, and then reads text that is captured. Camera-based systems such as Access Lens 'read' physical documents and lets a blind person listen to and interact with them [17]. OCR can do reasonably well in providing access to text that is well-formatted, but recognizing text in the real world can be difficult[1]. Even detecting that text exists in natural photographs can be difficult [16]. VizLens reads text using OCR in display areas marked by the crowd.

Recently, deep learning approaches have been applied to general object recognition, in products such as Aipoly[2] and Microsoft's "Seeing AI"[3]. These approaches can work reasonably well, although can only recognize a preset number of objects (*e.g.*, Aipoly recognizes approximately 1000 predefined objects). VizLens may eventually be used to collect data about physical interfaces that could be used to train deep learning models capable of replicating its performance.

Various projects have experimented with wearable computer vision approaches. Fingerreader [29] assists blind users with reading printed text on the go. One challenge that this approach has is that information beneath the fingertip can be occluded. This is a problem that VizLens does not have because it uses context to recognize occluded information based on its reference image. EyeRing similarly leverages a finger-worn camera to interpret immediate surroundings using computer vision [27]. OrCam is a product that uses a head-mounted camera to make available various computer vision applications targeting low vision people [28]. Foresee enables real-world objects to be magnified using a wearable camera and head-mounted display [40]. The form factors of these all introduce interesting opportunities that VizLens may eventually support; all are fundamentally limited by the performance of underlying computer vision.

Crowd-Powered Systems for Visual Assistance

Crowd-powered systems have been developed for various applications, *e.g.*, document editing and shortening [5], user interface control [22], real-time captioning [21]. These systems operate quickly by both lowering the latency to recruit workers [4, 6], and allowing workers to work synchronously together once recruited. At least two existing projects have explored the combination of computer vision and crowdsourcing. Zensors [20] fuses real-time human intelligence from online crowd workers with automatic approaches to provide robust, adaptive, and readily deployable intelligent sensors. Tohme [14] combines machine learning, computer vision, and custom crowd interfaces to find curb ramps remotely in Google Street View scenes, which performs similarly in detecting curb ramps compared to a manual labeling approach alone at a 13% reduction in time cost. VizLens is a crowd-powered system for making physical interfaces accessible.

A number of crowd-powered systems have been developed to make visual information accessible to blind people. One of the first projects in this space was VizWiz[4], a system that lets blind people take a picture, speak a question, and get answers back from the crowd within approximately 30 seconds [6]. More than 10,000 users have asked more than 100,000 questions using VizWiz[5]. Users often ask questions about interfaces [9], but it can be difficult to map the descriptions sent back, *e.g.*, "the stop button is in the middle of the bottom row of buttons," to actually using the interface. VizLens makes this much easier.

Other systems provide more continuous support. For example, Chorus:View [23] pairs a user with a group of crowd workers using a managed dialogue similar to [24] and a shared video stream. "Be My Eyes" matches users to a single volunteer over a shared video stream [3]. These systems could more easily assist blind users with using an interface, but assisting in this way is likely to be cumbersome and slow. VizLens specializes on the important subset of visual assistance tasks related to using physical interfaces and can assist with very low latency.

Other systems have expanded the capabilities of VizWiz. For example, VizWiz::LocateIt [7] allows blind people to ask for assistance in finding a specific object. Users first send an overview picture and a description of the item of interest to crowd workers, who outline the object in the overview picture. Computer vision on the phone then helps direct users to the specific object. This is somewhat similar to VizLens in that the robust intelligence is handled by the crowd, whereas the interactive refinding task is handled by computer vision. VizLens extends this to multiple objects and explicitly gives feedback on what is beneath the user's finger.

RegionSpeak [42] enables spatial exploration of the layout of objects in a photograph using a touchscreen. Users send

[1] `http://www.meridianoutpost.com/resources/articles/ocr-limitations.php`

[2] `http://aipoly.com`

[3] This exists currently as an unreleased system; a blog post about it is here: https://blogs.microsoft.com/next/2016/03/30/decades-of-computer-vision-research-one-swiss-army-knife/

[4] The "Viz" prefix comes from how some tech-savvy blind people refer to one another, *e.g.*, "are you viz?"

[5] `http://vizwiz.org/data/`

a photo (or multiple stitched photos), and the crowd labels all of the objects in the photo. Users can then explore the photo on a touchscreen. VizLens reuses some of these ideas in labeling all of the interface elements, although it extends RegionSpeak's functionality into real-world detection of interface elements that have been labeled.

FORMATIVE STUDY

We conducted several formative studies to better understand how blind people currently access and accommodate inaccessible interfaces. We first went to the home of a blind person, and observed how she cooked a meal and used home appliances. We also conducted semi-structured interviews with six blind people (aged 34-73) about their appliances use and strategies for using inaccessible appliances. Using a Wizard-of-Oz approach, we asked participants to hold a phone with one hand and move their finger around a microwave control panel. We observed via video chat and read aloud what button was underneath their finger.

We extracted the following key insights, which we used in the design of VizLens:

- Participants felt that interfaces were becoming even less accessible, especially as touchpads replace physical buttons. However, participants did not generally have problems locating the control area of the appliances, but have problems with finding the specific buttons contained within it.
- Participants often resorted to asking for help, such as a friend or stranger: frequently seeking help created a perceived social burden. Furthermore, participants worried that someone may not be available when they are most needed. Thus, it is important to find alternate solutions that can increase the independence of the visually impaired people in their daily lives.
- Labeling interfaces with Braille seems a straightforward solution but means only environments that have been augmented are accessible. Furthermore, fewer than 10 percent blind people in the United States read Braille [1].
- Participants found it difficult to aim the phone's camera at the control panel correctly. In an actual system, such difficulty might result in loss of tracking, thus interrupting the tasks and potentially causing confusion and frustration.
- Providing feedback with the right details, at the right time and frequency is crucial. For example, participants found it confusing when there was no feedback when their finger was outside of the control panel, or not pointing at a particular button. However, inserting feedback in these situations brings up several design challenges, *e.g.*, the granularity and frequency of feedback.

VIZLENS

VizLens is an accessible mobile application for iOS and a supporting backend. VizLens users capture a photo of an inaccessible interface and send it to multiple crowd workers, who work in parallel to quickly label and describe elements of the interface to make subsequent computer vision easier. The VizLens application helps users recapture the interface in the field of the camera, and uses computer vision to match new camera input to previously obtained crowd-labeled reference images to recognize and inform the user of the control he intends to use by providing feedback and guidance.

Implementation

VizLens consists of three components: *(i)* mobile application, *(ii)* web server, and *(iii)* computer vision server.

Mobile App

The iOS VizLens app allows users to add new interfaces (take a picture of the interface and name it), select a previously added interface to get interactive feedback, and select an element on a previously added interface to be guided to its location. The VizLens app was designed to work well with the VoiceOver screen reader on iOS.

Web Server

The PHP and Python web server handles image uploads, assigns tasks to Amazon Mechanical Turk workers for segmenting and labeling, hosts the worker interface, manages results in a database and responds to requests from the mobile app. The worker interfaces are implemented using HTML, CSS, and Javascript.

Computer Vision Server

The computer vision pipeline is implemented using C++ and the OpenCV Library. The computer vision server connects to the database to fetch the latest image, process it, and write results back to the database. Running real-time computer vision is computationally expensive. To reduce delay, VizLens uses OpenCV with CUDA running on GPU for object localization. Both the computer vision server and the web server are hosted on an Amazon Web Services EC2 g2.2xlarge instance[6], with a high-performance NVIDIA GRID K520 GPU, including 1,536 CUDA cores and 4GB of video memory.

Overall Performance

Making VizLens interactive requires processing images at interactive speed. In the initial setup [13], VizLens image processing was run on a laptop with 3GHz i7 CPU, which could process 1280×720 resolution video at only 0.5 fps. Receiving feedback only once every 2 seconds was too slow, thus we moved processing to a remote AWS EC2 GPU instance, which achieves 10 fps for image processing. Even with network latency (on wifi) and the phone's image acquisition and uploading speed, VizLens still runs at approximately 8fps with 200ms latency.

Initial Crowdsourced Segmenting and Labeling

The first time a user encounters an interface, he uses VizLens to take a photo of the interface (Figure 2b), provide a name for it, and send the image to be processed and pushed to the crowd for manual labeling. This image is called the "reference image." In order to make the reference image most useful for computer vision algorithms, VizLens uses a two-step workflow to label the area of the image that contains the interface and then label the individual visual elements.

[6] `https://aws.amazon.com/ec2/instance-types/#g2`

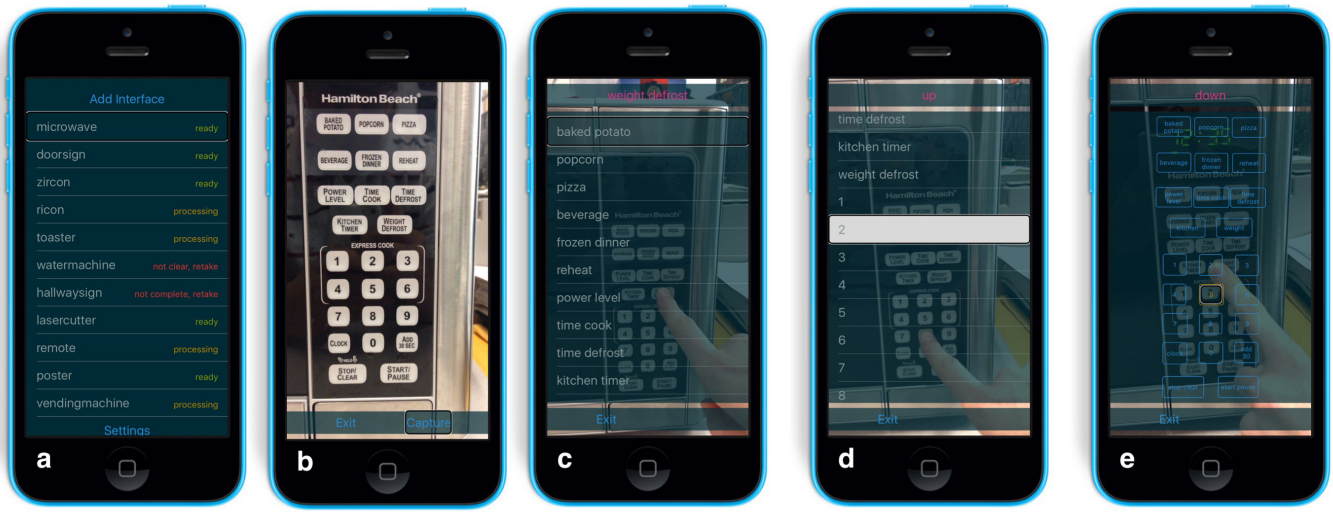


Figure 2. VizLens mobile app interfaces. ***(a)*** **App's main screen, listing all available interfaces and status, as well as Add Interface and Settings options.** ***(b)*** **Interface for Add Interface, where the user can take a photo and denote a name for it.** ***(c)*** **When user selects an interface, he can start to get feedback.** ***(d)*** **When selecting a visual element, the app start providing guidance.** ***(e)*** **a virtual interface layout that users can explore directly on the touchscreen.**

Step 1: Segmenting Interface Region

In step 1 (Figure 3a), crowd workers are asked whether the interface of the object is complete and clear in the photo. If the majority of workers agree that the photo contains a clear view of the complete interface, it proceeds to the next step; otherwise the user is prompted to take another photo (Figure 2a). Once an acceptable image is captured, crowd workers draw a bounding box around the interface, which will be cropped in the backend server and used for recognition later. In this step, the crowd workers are also asked to indicate the approximate number of visual elements, which will make it easier to distribute tasks and calculate compensation for the next step.

Step 2: Labeling Visual Elements

In step 2, crowd workers are instructed to draw bounding boxes around all of the individual visual elements (*e.g.*, buttons) within the interface area (Figure 3b); and provide a text annotation for each element (such as labeling buttons as 'baked potato', 'start/pause'). Similar to RegionSpeak [42], VizLens has multiple workers label in parallel to complete all of the visual element within a very short time, even if the interface is cluttered with visual elements (although we are currently not as aggressive in recruiting workers).

The workers interface shows labeled elements to other workers as they are completed. Over time, this allows the workers to completely label all of the elements. An initial challenge was that workers tended to label the interface in the same order at the same time, *e.g.*, from top to bottom. This resulted in redundant labels that increased the time required to completely label the interface. We cannot simply queue all the labeling tasks because we do not know *a priori* where the elements are. To encourage workers to label different buttons, we added an arrow that points to a random location (*e.g.*, up arrow in Figure 3b). Even though the arrow is placed randomly, it effectively directs workers toward different parts of the interface, encouraging them to work in different orders and reducing redundant work.

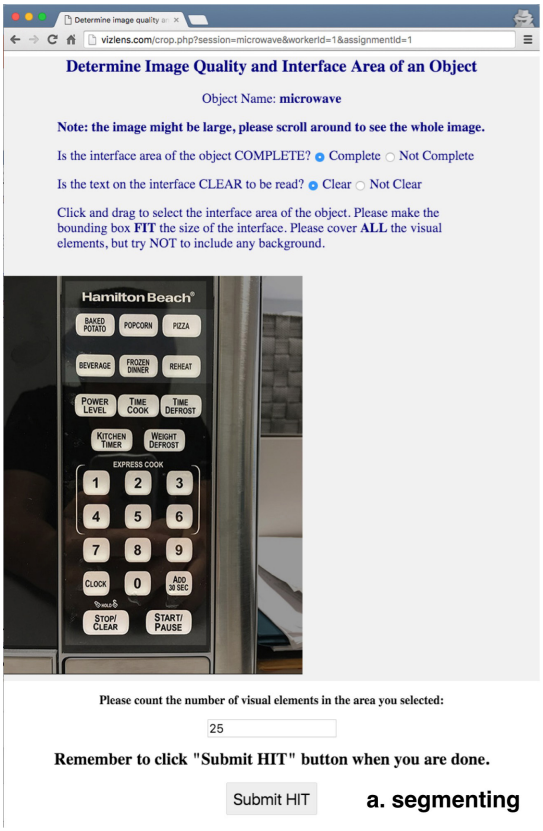
The VizLens backend monitors the progress of labeling, including aggregating overlapping labels, and counting the number of visual elements already labeled. Two bounding boxes are detected to be overlapping with each other if each one of the center points lies within the other. Once it reaches the expected number of visual elements from step 1, the interface will show an option for finishing labeling this image (the bottom option in Figure 3b). Once agreement is reached, this image then becomes the reference image (Figure 4a).

In the future, this labeling step could use automatic techniques as a first pass, *e.g.*, OCR or automatic button detection, in order to save crowd workers' time. Over time, the data collected as people use VizLens may allows robust recognizers to be trained. We do not expect automatic approaches to work perfectly in the near term, which is why we use the crowd.

After initial segmenting and labeling by the crowd, VizLens relies on computer vision. The reason computer vision is likely to work robustly now is that the problem has been simplified from the general problem (any interface in any context) to a much more specific one (this interface in a similar context, *e.g.*, lighting condition, placement, etc). This *hyperlocal context* argument is similar to that used to explain why computer vision worked better than expected in Zensors [20].

Retrieving Visual Elements

The core function of VizLens is to speak a description of the part of the interface that is beneath the user's finger. To do this, VizLens needs to be able to *(i)* find the interface in the input images, *(ii)* detect their finger, and *(iii)* retrieve and output the correct information based on the finger location.



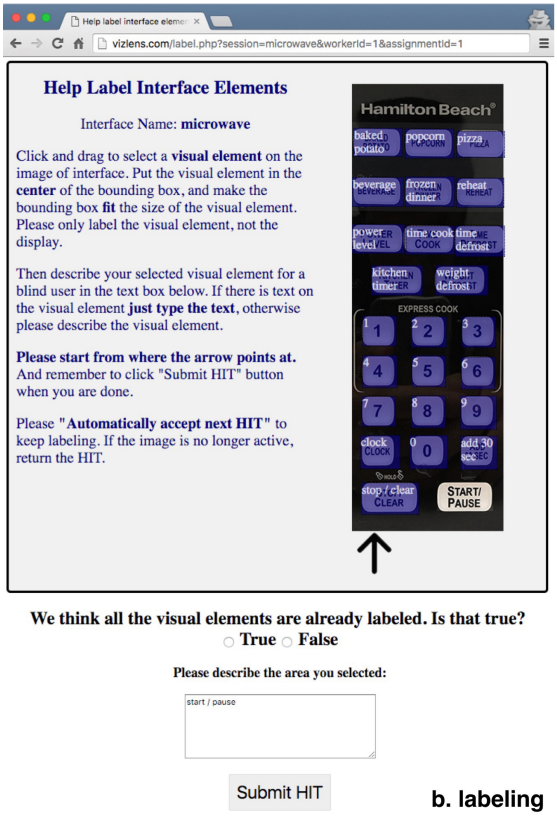


Figure 3. Crowd segmenting and labeling interfaces of VizLens: *(a)* crowd workers rate the quality of the initial photo of the interface, segment the interface area, and specify the number of visual elements on the interface, *(b)* other crowd workers then label individual visual elements on the interface in parallel.

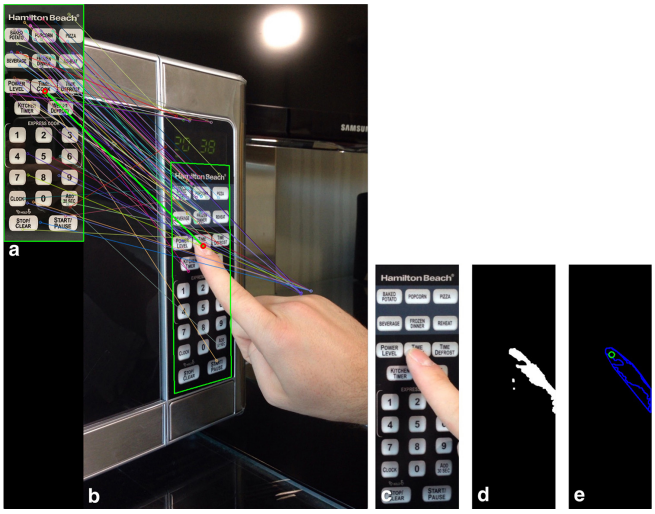


Figure 4. Real-time recognition and control using VizLens. Recognition result, showing *(a)* reference image and *(b)* input image. *(c)* Input image warped to reference image's frame allowing the coordinates of the elements previously labeled to be retrieved. *(d)* Result of skin color thresholding, and *(e)* calculated fingertip location.

Refinding the Desired Interface

Using the reference image obtained earlier, VizLens can first localize the interface in the input video stream in real-time. It uses SURF (Speeded-Up Robust Features) [2] feature detector with hessian threshold set to 400 to compute key points and feature vectors in both the reference (Figure 4a) and the input image (Figure 4b). Note that the reference image can be pre-computed once in advance to improve processing speed. The feature vectors are then matched using brute-force matcher with normalization type of L2 norms, which is the preferable choice for SURF descriptors. By filtering matches and finding the perspective transformation between the two images using RANSAC (Random Sample Consensus), VizLens is able to localize the reference image of the interface in the input image. In Figure 4b, the green bounding box is identified by transforming the corners of the reference image to corresponding points in the input image.

Fingertip Detection

VizLens then transforms the input image to the reference image frame using a warp function (Figure 4c), adjusts the lighting of the warped image to match the reference image, and detects the fingertip's location using skin color thresholding [34]. After performing Gaussian Blur with a 3-by-3 kernel to smooth the image and transforming it to HSV (Hue, Saturation, Value) color space, it uses a set of thresholds to segment the skin parts from the image (for [H, S, V] values respectively, the lower thresholds are [0, 90, 60], and upper thresholds [20, 150, 255]). Then it uses morphological operations *i.e.* eroding and dilating to filter out noises from the threshold image (Figure 4d). Then, VizLens uses the largest contour of the convex hull to detect the fingertip's location (Figure 4e). VizLens requires the user to use one finger to hover over the button, therefore the system recognizes the topmost fingertip location in the image if multiple exists. This approach also reduces the size of the image to process to only the reference image interface, reducing processing time.

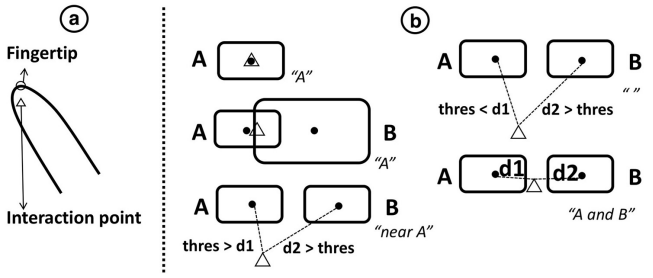


Figure 5. ***(a)*** **Defining interaction point relative to the detected fingertip location.** ***(b)*** **Rules for assigning feedback based on button layout.**

Information Lookup

Note that for interaction purposes, the topmost fingertip location is normally not the exact location of the finger pad that is used to, *e.g.*, press a button. Considering that most control buttons are designed to be similar in size with a finger width for ease of use, VizLens defines an "interaction point" by adding a fraction of the average button size to the y-position of the computed fingertip location (Figure 5a).

Then by looking up the coordinates of the interaction point in the database of the reference image's labeled visual elements, VizLens provides real-time feedback or guidance. To do this quickly, instead of looking up in the database every frame, we pre-compute a hash table of the resolution of the reference image associating pixel locations with feedback according to the following rules (Figure 5b):

- If the interaction point is within a button, assign that button's label, *e.g.*, power level;
- If the interaction point is within two buttons at the same time, assign the button's label whose center is closer to the interaction point;
- If the interaction point is not in any button, check its distance to the two closest buttons ($d_1, d_2; d_1 <= d_2$). If both are larger than a threshold (*e.g.*, average button size), assign an empty string;
- If only the closest distance (d_1) is within the threshold, assign "near" and the button's label, *e.g.*, near power level;
- If both distances (d_1 and d_2) are within the threshold, assign the two button labels separate by "and" with the closer one to start, *e.g.*, power level and time cook.

Providing Feedback and Guidance

Providing Feedback

After identifying the visual element, VizLens triggers the VoiceOver screen reader on iOS to read its description (Figure 2c). In our formative studies, participants found it hard to keep track of the feedback when their finger was moving quickly on the interface. Therefore, if the finger movement speed is over a threshold (*e.g.*, 1.5 buttons per second), VizLens will not confuse the user by providing feedback.

Pilot users were confused when the system did not provide any feedback, which happened when the object was not found or when no finger was on the interface. Providing no feedback was confusing, while having it repeat "no object" or "no finger" could get annoying. Pilot users also found it annoying when VizLens repeated the same label over and over again. Based on this feedback, we decided to only announce the instructions every second when it is not changing. On the other hand, a different instruction is immediately announced. As an option in the mobile app, users can select between announcing feedback using polite or interrupt mode. In polite mode, a new label will be announced only after the current one finishes. However, in interrupt mode, once a new label comes in, it will announce it right away and cut off the current one. As another preference option, besides saying "no object" or "no finger", VizLens also applies sonification techniques and uses low and high pitch sound as earcons [8] to identify a lack of object in view and a lack of finger while object is in view.

Providing Guidance

In our formative studies, participants wanted to know the direction to a button when unfamiliar with an interface. VizLens allows a user to specify a target in the app through speech or selection in a list of available visual elements, and then provides guidance to it (Figure 2d).

The path of navigation follows the Manhattan Distance [37] between the current interaction point to the target location, which means only vertically and horizontally. In order to avoid frequent change of directions, VizLens guides the user to first move vertically along the y-axis (*i.e.*, up and down), and once settled within a threshold, it proceeds to horizontal directions (*i.e.*, left and right). VizLens repeats the instruction every second. Many participants overshot the target in our pilot studies. To address this problem, VizLens defines coarse and fine control areas, and the system will notify the user to move slowly when finger is near the target (*e.g.*, within 1.5 button sizes from the center of the target). When the finger is on the button, VizLens reads out the button label.

USER EVALUATION

The goal of our user study was to evaluate how VizLens performs in assisting visually impaired people accomplish realistic tasks that involve otherwise inaccessible interfaces. We evaluated it deeply on one appliance (an inaccessible microwave), with more shallow evaluations across many other devices. Further evaluation of its components is presented in the next section ("Technical Evaluation").

The microwave we chose was a Hamilton Beach 1.1 Cu Ft Microwave. The buttons on this microwave are flat and provide little (if any) tactile feedback. It contains some familiar buttons (0-9), and many that are likely to be less familiar (*e.g.*, time defrost, baked potato).

Apparatus and Participants

The VizLens iOS app was used in the study, installed on an iPhone 5c, runing iOS 9.2.1. For this particular evaluation, all the images were labeled by the experimenter as introducing the crowd would result in compound factors. The quality of the crowd's labeling was evaluated in a separate study.

We first conducted a pilot study with two visually impaired users to finalize the tasks, number of tasks, and fine-tuned

Table 1. Participant demographics for our user evaluation with 10 visually impaired users.

ID	Gender	Age	Occupation	Vision Level	Smartphone Use
P1	Female	33	AT consultant	Blind, since birth	iPhone, 4 years
P2	Male	37	Tech teacher for blind	Blind, since birth	iPhone, 3 years
P3	Female	47	Sales	Light perception, tunnel vision	Android, 5 years
P4	Male	24	Software developer	Blind, since birth	iPhone, 5 years
P5	Male	34	AT specialist	Light perception, since birth	iPhone, 5 years
P6	Male	21	Student	Light/color perception	iPhone, 5 years
P7	Male	40	Digital AT consultant	Blind, since birth	iPhone, 2.5 years
P8	Male	31	Scriptor, AT instructor	Light/color perception, since birth	iPhone, 5 years
P9	Male	26	AT instructor	Light perception, since birth	iPhone, 5 years
P10	Male	29	Project manager	Blind, later on	Mostly iPhone, 10 years

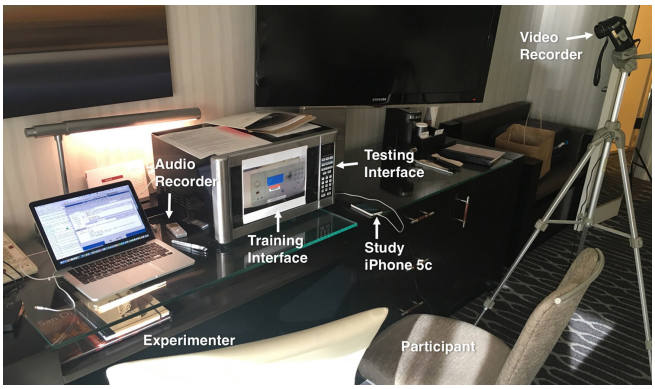


Figure 6. User study setup. A printer's interface is printed out on paper and used for training. The microwave interface was used for controlled testing, followed by more exploratory use of other interfaces nearby (*e.g.*, remote control, thermostat, vending machine). The study was conducted in a hotel room and was video and audio recorded.

some parameters in our system. We then recruited 10 visually-impaired users (2 female, age 21-47). The demographics of our participants are shown in Table 1.

Design

Our study consisted of an initial training phase, followed by a series of task using the microwave. There were two conditions in completing the tasks: *(i)* feedback - where the participants were provided with audio feedback of what is underneath their finger on the interface; and *(ii)* guidance - where audio directions were provided for them to move their finger to a specific target. After each condition, we conducted a semi-structured interview collecting subjective feedback for the methods. The order of conditions was counterbalanced for all participants. The study took about one hour and the participants were compensated for $50. The whole study was video and audio recorded for further analysis, and the study set up is shown in Figure 6.

Tasks

Following a brief introduction of the study and demographic questions, we first used a printer's interface printed on paper to familiarize the participants with the iOS app. In this training phase, we also asked for the participant's preferences on the polite/interrupt and sound/word settings. Then, participants were asked to take 10 photos of the microwave control panel, with feedback provided after each one to simulate the

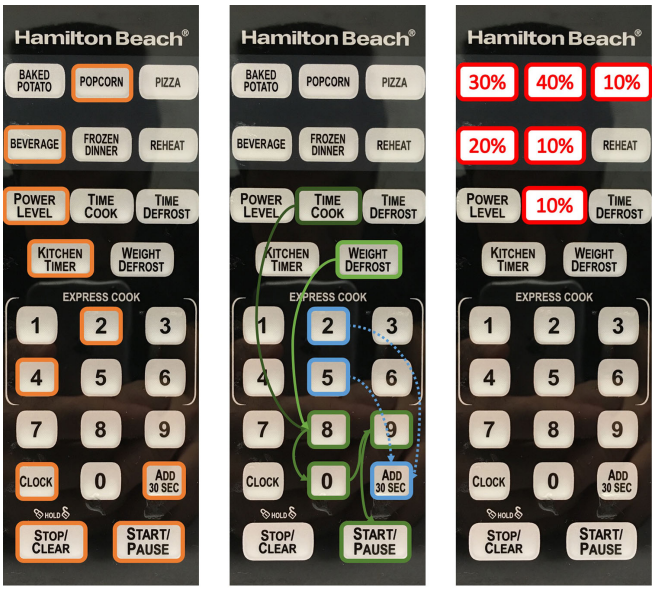


Figure 7. Visualization of user study tasks and identification results. *(a)* Locating tasks highlighted in orange. *(b)* Simulating cooking tasks highlighted in green and blue, and sequences shown with line and dashed arrows. *(c)* Identification errors visualized on the interface, where all errors are happening on the top region of the control panel.

crowd feedback for image quality. These images are used for evaluating the crowd-based labeling in a separate study.

Next, for each of the two conditions, participants were asked to complete five locating tasks and two simulating cooking tasks. For locating tasks, the participant was asked to locate a button with the assistance of the VizLens app, and then push to trigger the button. As shown in Figure 7a, the 10 buttons were selected so that they covered different areas on the control panel. For simulating cooking tasks, we designed more realistic tasks that involved a series of button presses. For example, a multi-button cooking task would require pressing a configure button (*e.g.*, weight defrost, time defrost, or time cook), followed by setting a time duration by pressing the number pads (*e.g.*, 2, 1, 0 for two minutes and 10 seconds, or two pounds and 10 oz), and finally pressing the 'Start' button. The specific tasks used are visualized in Figure 7b. For both locating and simulating cooking tasks, we measured

completion rate and time for successfully completing a task. After each condition, participants were asked a few subjective questions about that condition.

After the two conditions, we conducted a controlled test for identifying individual buttons. Participants were guided by the experimenter to rest his/her finger on each button of the interface. The system recognizes the button and the accuracy was recorded. Finally, we ended the study with a final semi-structured interview asking for the participant's comments and suggestions on the VizLens system.

Results

We detailed our user study results and performed t-tests to compare participant's task completion rate and time for the two methods. We also summarized users' feedback and preferences that informed our next design iteration of VizLens.

Identification Tasks

For identification tasks, only 10 of a total of 250 buttons were falsely identified across 10 participants, resulting in an accuracy of 96.0%. When taking a deeper look at the errors, all errors are happening on the top region of the control panel (Figure 7c). This is most likely because when interacting with buttons on the top region, the user's hand covers most of interface, making the object localization harder with fewer SURF features points left in the image. Furthermore, our user study demonstrated that VizLens works robustly in various lighting and skin color conditions, as shown in Figure 8. To further improve the robustness of the variety of skin color and lighting conditions, we could add a pre-calibration step for individual users in new environments.

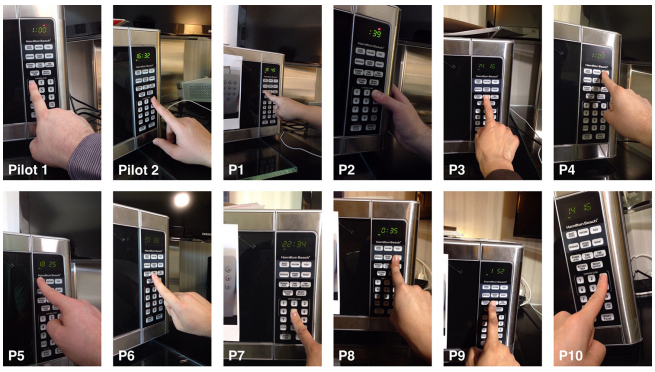


Figure 8. VizLens works robustly across various skin colors and lighting conditions. These are images from participants that were processed by computer vision and successfully identified the finger locations.

Locating Tasks

For locating tasks, participants successfully completed 41/50 ($M = 82.0\%, SD = 0.175$) tasks under 200 seconds in feedback condition, which is significantly lower than 49/50 ($M = 98.0\%, SD = 0.063$) for guidance, $t(9) = -2.753, p = 0.022$ (two-tailed). However, there was no significant difference for average task completion time between feedback ($M = 52.5, SD = 52.6$) and guidance ($M = 54.4, SD = 40.4$), $t(88) = -0.198, p = 0.843$ (two-tailed). The difference in task completion rate is most likely because for guidance it is more independent of the user's mental model of the interface. While for feedback, it is hard to find a random button. Therefore, we hypothesized that it is easier to find function buttons (*e.g.*, power level, baked potato) using guidance than feedback mode, while it is easier to find number buttons (*i.e.*, 0 - 9) using feedback than guidance.

To validate our hypothesis, we took a deeper look into the data. For number buttons, with all tasks successfully completed for both conditions, the average task completion time for feedback ($M = 27.8, SD = 17.6$) was shorter than for guidance ($M = 36.3, SD = 17.0$), though this is not statistically significant, $t(9) = -1.138, p = 0.142$ (one-tailed). We think this is because using feedback mode, when the users found a number, they also knew the general location of other number buttons, making them easier to find. However, for guidance mode, it is harder for participants to take advantage of their mental model of the interface with the directional instructions. For all other buttons, even though there were no significant differences in task completion time between feedback ($M = 60.4, SD = 57.7$) and guidance ($M = 59.1, SD = 43.4$), $t(68) = 0.910, p = 0.910$ (two-tailed), the task completion rate for feedback was significantly lower ($31/40, M = 77.5\%, SD = 0.219$) compared with ($39/40, M = 97.5\%, SD = 0.079$) in guidance, $t(9) = -2.753, p = 0.011$ (one-tailed).

Figure 9 shows the time breakdown for feedback and guidance modes. In feedback mode, users aim the camera and search for the button repetitively, and press once they reach the button. In guidance mode, users first select a button from the list in the VizLens app, aim the camera, follow instructions to the button, and press. One challenge we observed is that sometimes VizLens would give correct feedback of a button's label, but users could not push it because their finger was not directly on the center of the button. This could be confusing, although users generally resolved it eventually.

Simulating Cooking Tasks

For simulating cooking tasks, there was no significant difference in task completion rate between feedback ($18/20, M = 90.0\%, SD = 0.211$) and guidance ($20/20, M = 100\%$), $t(9) = -1.500, p = 0.168$ (two-tailed), as well as in average task completion time between feedback ($M = 102.3, SD = 93.6$) and guidance ($M = 120.4, SD = 64.8$), $t(36) = -0.698, p = 0.490$ (two-tailed).

Subjective Feedback

During training, we asked for participant preferences on polite/interrupt and sound/word settings. 6 out of 10 participants preferred interrupt mode than polite mode, due to its instantaneous feedback. For sound/word setting, half the users preferred using words, while the other half preferred earcons. The users who preferred using words mentioned that the two earcons for "no object" and "no finger" were not distinctive enough for them to easily differentiate between the two.

We asked the participants to rate and compare the two method based on learnability, comfort, usefulness, and satisfaction (Figure 10). Several participants expressed their frustration with aiming and keeping good framing of the camera. Sev-

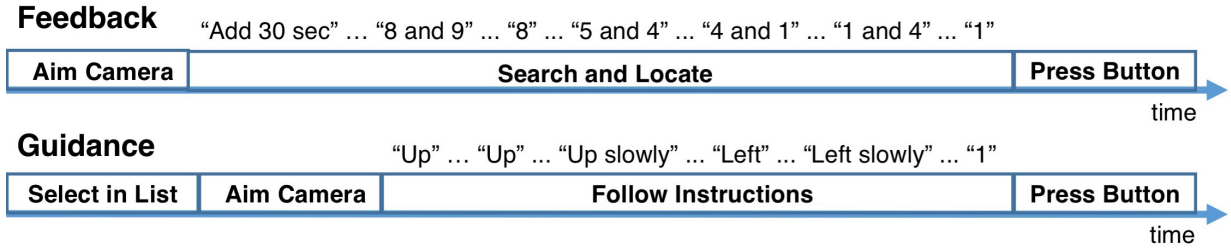


Figure 9. Time breakdown for feedback and guidance modes. For feedback, users aim the camera and search for the button repetitively, and press once they reach the button. For guidance, users first select a button in the list, aim the camera, then follow the instructions to the button, and press.

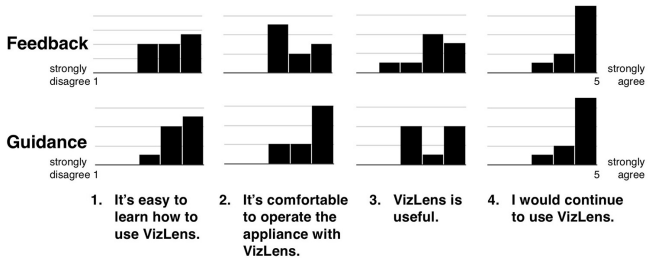


Figure 10. Answers to Likert scale questions indicating that participants found VizLens useful (1, 2, 3) and wanted to continue using it (4).

eral participants tried to get a general idea of the button layout from the linear button list (Figure 2c) and suggested to show the layout of the buttons of the interface on the touchscreen, so that they can explore and build a mental model first, and then use the system's feedback to locate the button they want to use, similar to RegionSpeak [42]. We address most of this feedback in VizLens v2 presented later. Overall, participants were excited about the potential of VizLens and several asked when the app can be available for download. One participant mentioned that when living alone and got a new appliance, he had to wait and ask someone to help put dots on it. Using VizLens, he could get oriented by himself and start using it right away, which is a big advantage.

TECHNICAL EVALUATION

We conducted a multi-part technical evaluation in order to understand how each component of VizLens performs across a range of interfaces.

Crowdsourcing Performance

We evaluated our crowdsourcing interfaces with the 120 images taken by the blind participants in the user studies just described. First, the experimenters manually labeled these images as ground truth. For each image, each segmentation step was completed by a different worker (Figure 3a). For image quality, an agreement of three workers was required. If the image was determined to be complete and clear, $2 \times (Number of Buttons)$ of HITs were created for the labeling step (Figure 3b) so that multiple crowd workers could work in parallel. Once a worker agreed with the system that the interface is completely labeled, this crowdsourcing segmenting and labeling process is completed.

A total of 251 crowd workers participated in this evaluation, providing 2,147 answers overall. For the 68 out of 120 images that failed the quality qualifications, it took an average of 134 seconds ($SD = 86$) for VizLens to provide this feedback. All of the feedback was correct. Each segmenting task paid \$0.15, which required ~40 seconds of work (\$13.5/hour). Each image costs an average of \$0.50 ($SD = 0.09$).

For the 52 out of 120 images that were complete and clear, it took an average of 481 seconds ($SD = 207$) before the VizLens interface was ready to be used, including time to upload the photo, workers to pick up the HITs, complete the tasks, and the system to aggregate labels. 99.7% ($SD = 1.3\%$) of the buttons were correctly labeled. Each labeling task paid \$0.02, which required less than 10 seconds of work (\$9/hour). Each interface costs an average of \$1.15 ($SD = 0.12$). We believe more aggressive recruiting of crowd workers could lead to even shorter latencies, but this was not our focus.

Interface Robustness

Similar to the identification tasks in the user evaluation, we conducted a controlled test for identifying individual buttons on another set of interfaces (Figure 11) to see when it succeeds and fails. For the thermostat, remote control, laser cutter, toaster and printer, VizLens successfully recognized all buttons. For the vending machine, button A on the top left failed, possibly also because of the hand covering a large portion of the interface. For the copier and water machine, even though all buttons were successfully recognized eventually, there were a lot of false-identifications initially caused by the buttons that confused with the skin color in HSV color space. To adapt for these situations, applying background subtraction method or pre-calibration of skin color for fingertip detection might improve performance. VizLens failed the fridge interface completely, mainly because there are very few features points that can be used for matching for the object localization algorithm. Similar for identification results in the user studies, where all errors happened near the top of the control panel, the requirement for feature points for object localization and matching is a limitation of VizLens. One possibility to adapt to interfaces with few feature points is to attach fiducial markers with specific patterns to introduce feature points into the field of view [12]. This would require modifying the interface, but, as opposed to labeling it, would not require the markers to be positioned in any particular place and could be done independently by a blind person.

VIZLENS VERSION 2

Based on participant feedback in our user evaluation, we developed VizLens v2. Specifically, we focus on providing better feedback and learning of the interfaces.

Figure 11. VizLens works robustly with a wide range of interfaces, including microwaves, printers, copiers, water machines, thermostats, laser cutters, toasters, remote controls, vending machines, etc.

For VizLens to work properly it is important to inform and help the users aim the camera centrally at the interface. Without this feature, we found the users could 'get lost' —they were unaware that the interface was out of view and still kept trying to use the system. Our improved design helps users better aim the camera in these situations: once the interface is found, VizLens automatically detects whether the center of the interface is inside the camera frame; and if not, it provides feedback such as "Move phone to up right" to help the user adjust the camera angle.

To help users familiarize themselves with an interface, we implemented a simulated version with visual elements laid out on the touchscreen for the user to explore and make selection (Figure 2e), similar to RegionSpeak [42]. The normalized dimensions of the interface image as well as each element's dimensions, location and label make it possible to simulate buttons on the screen that react to users' touch, thus helping them get a spatial sense of where these elements are located.

We also made minor function and accessibility improvements such as vibrating the phone when the finger reaches the target in guidance mode, making the earcons more distinctive, supporting standard gestures for back, and using the volume buttons for taking photos when adding a new interface.

We also explored functional extensions of VizLens that allow it to *(i)* adapt to state changes in dynamic interfaces, *(ii)* combine crowd labeling with OCR technology to handle dynamic displays, and *(iii)* benefit from head-mounted cameras.

VizLens::State Detection

Many interfaces include dynamic components that cannot be handled by the original version of VizLens, such as an LCD screen on a microwave, or the dynamic interface on self-service checkout counter. As an initial attempt to solve this problem, we implemented a state detection algorithm to detect system state based on previously labeled screens. For the example of a dynamic coffeemaker, sighted volunteers first go through each screen of the interface and take photos. Crowd workers will label each interface separately. Then when the

Figure 12. VizLens::State Detection detects screen state and adapts to it. In this example, VizLens figures out which of six states this fancy coffee machine is in, and provides feedback or guidance specific to that screen.

Figure 13. VizLens::LCD Display Reader applies OCR to recognize digits on the portion of the interface that is an LCD screen. *(a)* Separated dynamic and static regions. *(b)* Image sharpening using unsharp masking. *(c)* Intensity-based thresholding. *(d)* Morphological filtering and small blob elimination. *(e)* Selective color inversion.

blind user accesses the interface, instead of only performing object localization for one reference image, our system will first need to find the matching reference image given the current input state. This is achieved by computing SURF keypoints and descriptors for each interface state reference image, performing matches and finding homographies between the video image with all reference images, and selecting the one with the most inliers as the current state. After that, the system can start providing feedback and guidance for visual elements for that specific screen. As a demo in our submission video, we show VizLens helping a user navigate the six screens of a coffeemaker with a dynamic screen (Figure 12).

VizLens::LCD Display Reader

VizLens v2 also supports access to LCD displays via OCR. We first configured our crowd labeling interface and asked crowd workers to crop and identify dynamic and static regions separately (Figure 13a). This both improves computational efficiency and reduces the possibility of interference from background noises, making it faster and more accurate for later processing and recognition. After acquiring the cropped LCD panel from the input image, we applied several image processing techniques, including first image sharpening using unsharp masking [39] for enhanced image quality (Figure 13b) and intensity-based thresholding to filter out the bright text (Figure 13c). We then performed morphological filtering to join the separate segments of 7-segment displays (which are commonly used in physical interfaces) to form contiguous characters, which is necessary since OCR assumes individual segments correspond to individual characters. For the dilation's kernel, we used $height > 2 \times width$

Figure 14. We migrated VizLens to run on Google Glass, which has the advantage of leaving the user's hand free, easier to keep image framing stable, and naturally indicating the field of interest.

to prevent adjacent characters from merging while forming single characters. Next, we applied small blob elimination to filter out noise (Figure 13d), and selective color invertion to create black text on a white background, which OCR performs better on (Figure 13e). Then, we performed OCR on the output image using the Tesseract Open Source OCR Engine [38]. When OCR fails to get an output, our system dynamically adjusts the threshold for intensity thresholding for several iterations.

VizLens::Wearable Cameras

56.7% of the images took by the blind participants for crowd evaluation failed the quality qualifications, which suggests there is a strong need to assist blind people in taking photos. In our user evaluation, several participants also expressed their frustration with aiming and especially keeping good framing of the camera. Wearable cameras such as the Google Glass have the advantage of leaving the user's hand free, easier to keep image framing stable, and naturally indicating the field of interest. We have ported the VizLens mobile app to Google Glass platform (Figure 14), and pilot tested with several participants. Our initial results show that participants were generally able to take better framed photos with the head-mounted camera, suggesting that wearable cameras may address some of the aiming challenges.

DISCUSSION AND FUTURE WORK

VizLens enables access and exploration of inaccessible interfaces by providing accurate and usable real-time feedback and guidance. While VizLens is not the first system to combine crowdsourcing and computer vision, we believe its robustness and focus on interactive tasks differentiate it from prior work in this area. This paper targets making physical interfaces of the type found on electronic appliances accessible. VizLens might be extended to other tasks that involve the presentation and interaction with spatial information. For instance, VizLens could be useful in helping blind users access inaccessible figures or maps [19].

Even after access to the content of an interface is available, designing good feedback remains challenging. In comparing feedback and guidance in our user studies, we found that some visual elements are laid out in a way that promotes "wayfinding", *e.g.*, number pads, when feedback is better; while some are less intuitive, *e.g.*, the functional buttons, and in these cases guidance is better. We could ask crowd workers to provide more structural information of the interface, and dynamically adjust between the two modes when navigating their finger on the interface. Note that we tried to merge the two methods together by providing feedback and guidance at the same time, *e.g.*, "time cook and kitchen timer, up". However, it was difficult for users to deal with so much information, especially when the user is also focusing on moving their finger to locate certain button. VizLens opens up new opportunities and relevance for the design of audio feedback to support interaction with otherwise inaccessible interfaces.

Our crowdsourcing evaluation results show that our crowdsourced segmenting and labeling workflow was fast (8 minutes), accurate (99.7%), and cheap ($1.15) for a very visually cluttered microwave interface with 25 buttons, demonstrating the practicality of VizLens in the real world. If VizLens were a product, a full time staff might reasonably be employed to provide interface labeling. It is likely possible that we could push the initial latency of creating the reference image down to a minute or two [42], although it is unclear how important this will be in practice, given that feedback from computer vision is nearly instantaneous once labeled. Future work may look to have the crowd provide more information regarding the interface for various information need, such as details of usage of each visual element rather than only a label, structural information, dynamic and static components, etc.

Built-in quality control (*e.g.*, checking that the size and aspect ratio of the button bounding box is reasonable, spell checking text labels, etc.) and redundancy mechanisms in VizLens improve the quality of answers. For the vision-based system components, refinding the desired interface and fingertip detection would not be affected by errors of crowd labeling. On the other hand, the information lookup might be affected if the boundary of the button is smaller or larger than its actual size, (*e.g.*, if the button is labeled to be larger, the region where the system will read the buttons label in feedback mode will increase). The system can adapt to some of this, for example, in Figure 5b, the second rule on the left column shows that this labeling deviation can be fixed by the lookup rules. Furthermore, once the blind users finger is on the button, he or she can generally push around to activate it.

An immediate future goal is to deploy VizLens to see how it performs over time in the everyday lives of blind users. Supporting such a deployment will require substantial engineering in order to scale the backend system. Currently the computer vision is run remotely because it needs a relatively high-power GPU in order to perform at interactive speeds. Yet, we expect before long the necessary computing power will be available on consumer phones. Over time, we expect data collected from deployments will allow the training of general models of physical interfaces, which may reduce or eventually eliminate crowd labeling.

We also plan to explore tighter integration between the end user, crowd, and computer vision. We imagine algorithms will monitor and predict the performance of the computer vision techniques. When the input images cause uncertain recognition results, it will provide the user with the option to

'ask the crowd'. This approach will inevitably take a longer wait time but the returned crowd-labeled image can be added to the library of reference images and improve the robustness of the recognition. If a similar situation occurs in the future, this new reference image could be a close match and the answers can be directly obtained from its labels. Collectively, these reference images can also benefit a broader range of users when it comes to interfaces in publicly shared spaces. When a blind user enters an unfamiliar office building and tries to use an interface, he can simply benefit from the reference images previously collected by someone else. When the images are geo-tagged, they can also help visually impaired users locate the interfaces they wish to use.

Finally, the large number of images collected as the user operates the interface could be used to improve the system over time. Using information of where the user pushes the button can help with determining more accurate location of the fingertip and fix errors over time. Furthermore, usage information can be collected to learn about the common functionalities accessed, and used to inform a new user of usage patterns.

CONCLUSION

We have presented *VizLens*, an accessible mobile application and supporting backend that can robustly and interactively help blind people use inaccessible interfaces in the real world. We introduced the design of the system and its technical architecture, evaluated it in a user study with 10 blind participants, and evaluated each component separately to understand its limitations. Based on feedback from these studies, we developed VizLens v2, which improved on the user interface and explored how VizLens might adapt to changing or dynamic interfaces. VizLens introduces a workflow that leverages the strengths of the end user (knowledge of the problem and context, and access to the interface), the crowd (sight and general intelligence), and computer vision (speed and scalability), and tightly integrates them to robustly solve a long-standing challenge in accessibility.

ACKNOWLEDGMENTS

This work has been supported by the National Science Foundation, Google, and the National Institute on Disability, Independent Living, and Rehabilitation Research. We also thank Yang Zhang for his help in the preparation of the user studies.

REFERENCES

1. The braille literacy crisis in america. facing the truth, reversing the trend, empowering the blind. National Federation of the Blind, Jernigan Institute, March 2009.
2. Bay, H., Tuytelaars, T., and Van Gool, L. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*. Springer, 2006, 404–417.
3. Be My Eyes. `http://www.bemyeyes.org`, 2015.
4. Bernstein, M. S., Brandt, J., Miller, R. C., and Karger, D. R. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 33–42.
5. 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. *Communications of the ACM 58*, 8 (2015), 85–94.
6. Bigham, J. P., Jayant, C., Ji, H., Little, G., Miller, A., Miller, R. C., Miller, R., Tatarowicz, A., White, B., White, S., et al. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 333–342.
7. Bigham, J. P., Jayant, C., Miller, A., White, B., and Yeh, T. Vizwiz:: Locateit-enabling blind people to locate objects in their environment. In *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*, IEEE (2010), 65–72.
8. Blattner, M. M., Sumikawa, D. A., and Greenberg, R. M. Earcons and icons: Their structure and common design principles. *Human–Computer Interaction 4*, 1 (1989), 11–44.
9. Brady, E. L., Zhong, Y., Morris, M. R., and Bigham, J. P. Investigating the appropriateness of social network question asking as a resource for blind users. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, ACM (New York, NY, USA, 2013), 1225–1236.
10. de Freitas, A. A., Nebeling, M., Chen, X. A., Yang, J., Karthikeyan Ranithangam, A. S. K., and Dey, A. K. Snap-to-it: A user-inspired platform for opportunistic device interactions. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, ACM (New York, NY, USA, 2016), 5909–5920.
11. Fusco, G., Tekin, E., Ladner, R. E., and Coughlan, J. M. Using computer vision to access appliance displays. In *ASSETS/Association for Computing Machinery. ACM Conference on Assistive Technologies*, vol. 2014, NIH Public Access (2014), 281.
12. Garrido-Jurado, S., Muñoz-Salinas, R., Madrid-Cuevas, F. J., and Marín-Jiménez, M. J. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition 47*, 6 (2014), 2280–2292.
13. Guo, A., Chen, X., and Bigham, J. P. Appliancereader: A wearable, crowdsourced, vision-based system to make appliances accessible. In *Proceedings of the 33rd Annual ACM Conference Extended Abstracts on Human Factors in Computing Systems*, ACM (2015), 2043–2048.
14. Hara, K., Sun, J., Moore, R., Jacobs, D., and Froehlich, J. Tohme: detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning. In *Proceedings of the ACM symposium on User interface software and technology* (2014), 189–204.
15. Jayant, C., Ji, H., White, S., and Bigham, J. P. Supporting blind photography. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, ACM (2011), 203–210.

16. Jung, K., Kim, K. I., and Jain, A. K. Text information extraction in images and video: a survey. *Pattern recognition 37*, 5 (2004), 977–997.

17. Kane, S. K., Frey, B., and Wobbrock, J. O. Access lens: a gesture-based screen reader for real-world documents. In *Proc. CHI*, ACM (2013), 347–350.

18. KNFB Reader. `http://www.knfbreader.com/`, 2015.

19. Ladner, R. E., Ivory, M. Y., Rao, R., Burgstahler, S., Comden, D., Hahn, S., Renzelmann, M., Krisnandi, S., Ramasamy, M., Slabosky, B., et al. Automating tactile graphics translation. In *Proceedings of the 7th international ACM SIGACCESS conference on Computers and accessibility*, ACM (2005), 150–157.

20. Laput, G., Lasecki, W. S., Wiese, J., Xiao, R., Bigham, J. P., and Harrison, C. Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM (2015), 1935–1944.

21. Lasecki, W., Miller, C., Sadilek, A., Abumoussa, A., Borrello, D., Kushalnagar, R., and Bigham, J. Real-time captioning by groups of non-experts. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, ACM (2012), 23–34.

22. Lasecki, W. S., Murray, K. I., White, S., Miller, R. C., and Bigham, J. P. Real-time crowd control of existing interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, ACM (2011), 23–32.

23. Lasecki, W. S., Thiha, P., Zhong, Y., Brady, E., and Bigham, J. P. Answering visual questions with conversational crowd assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ACM (2013), 18.

24. Lasecki, W. S., Wesley, R., Nichols, J., Kulkarni, A., Allen, J. F., and Bigham, J. P. Chorus: A crowd-powered conversational assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, ACM (New York, NY, USA, 2013), 151–162.

25. Manduchi, R., and Coughlan, J. M. The last meter: blind visual guidance to a target. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM (2014), 3113–3122.

26. Morris, T., Blenkhorn, P., Crossey, L., Ngo, Q., Ross, M., Werner, D., and Wong, C. Clearspeech: A display reader for the visually handicapped. *IEEE Transactions on Neural Systems and Rehabilitation Engineering 14*, 4 (Dec 2006), 492–500.

27. Nanayakkara, S., Shilkrot, R., Yeo, K. P., and Maes, P. Eyering: a finger-worn input device for seamless interactions with our surroundings. In *Proceedings of the 4th Augmented Human International Conference*, ACM (2013), 13–20.

28. OrCam. `http://www.orcam.com`, 2016.

29. Shilkrot, R., Huber, J., Liu, C., Maes, P., and Nanayakkara, S. C. Fingerreader: A wearable device to support text reading on the go. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems*, ACM (2014), 2359–2364.

30. Tekin, E., Coughlan, J. M., and Shen, H. Real-time detection and reading of led/lcd displays for visually impaired persons. In *Applications of Computer Vision (WACV), 2011 IEEE Workshop on*, IEEE (2011), 491–496.

31. Thatcher, J. Screen reader/2: Access to os/2 and the graphical user interface. In *Proceedings of the First Annual ACM Conference on Assistive Technologies*, Assets '94, ACM (New York, NY, USA, 1994), 39–46.

32. Vanderheiden, G., and Treviranus, J. Creating a global public inclusive infrastructure. In *Universal Access in Human-Computer Interaction. Design for All and eInclusion*. Springer, 2011, 517–526.

33. Vázquez, M., and Steinfeld, A. An assisted photography framework to help visually impaired users properly aim a camera. *ACM Transactions on Computer-Human Interaction (TOCHI) 21*, 5 (2014), 25.

34. Vezhnevets, V., Sazonov, V., and Andreeva, A. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, vol. 3, Moscow, Russia (2003), 85–92.

35. White, S., Ji, H., and Bigham, J. P. Easysnap: real-time audio feedback for blind photography. In *Adjunct proceedings of the 23nd annual ACM symposium on User interface software and technology*, ACM (2010), 409–410.

36. Wikipedia. Optical character recognition — wikipedia, the free encyclopedia, 2016. [Online; accessed 18-March-2016].

37. Wikipedia. Taxicab geometry — wikipedia, the free encyclopedia, 2016. [Online; accessed 14-March-2016].

38. Wikipedia. Tesseract (software) — wikipedia, the free encyclopedia, 2016. [Online; accessed 12-April-2016].

39. Wikipedia. Unsharp masking — wikipedia, the free encyclopedia, 2016. [Online; accessed 12-April-2016].

40. Zhao, Y., Szpiro, S., and Azenkot, S. Foresee: A customizable head-mounted vision enhancement system for people with low vision. In *Proceedings of the 17th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '15, ACM (New York, NY, USA, 2015), 239–249.

41. Zhong, Y., Garrigues, P. J., and Bigham, J. P. Real time object scanning using a mobile phone and cloud-based visual search engine. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ACM (2013), 20.

42. Zhong, Y., Lasecki, W. S., Brady, E., and Bigham, J. P. Regionspeak: Quick comprehensive spatial descriptions of complex images for blind users. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ACM (2015), 2353–2362.