Human-AI Systems for Visual Information Access

CMU-HCII-20-105
August 2020

Anhong Guo

Human-Computer Interaction Institute
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, USA 15213
anhongg@cs.cmu.edu

Thesis Committee:
Jeffrey P. Bigham (Chair), HCII & LTI, CMU
Chris Harrison, HCII, CMU
Jodi Forlizzi, HCII & Design, CMU
Meredith Ringel Morris, Microsoft Research

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

Keywords: Human-AI systems, human-computer interaction, intelligent systems, crowdsourcing, artificial intelligence, computer vision, machine learning, accessibility, environmental sensing, 3D printing, reverse engineering, augmented reality.

To my parents.

And to Xu.

Abstract

In my work, I create hybrid human- and AI-powered intelligent interactive systems to provide access to visual information in the real world. By combining the advantages of humans and AI, these systems can be nearly as robust and flexible as humans, and nearly as quick and low-cost as automated AI, enabling us to solve problems that are currently impossible with either alone.

I develop and deploy human-AI systems for two application domains: accessibility and environmental sensing. To make physical interfaces accessible for blind people, I develop systems to interpret static and dynamic interfaces, enabling blind people to independently access them through audio feedback or tactile overlays. For environmental sensing, I develop and deploy a camera sensing system that collects human labels to bootstrap automatic processes to answer real-world visual questions, allowing end users to actionalize AI in their everyday lives.

AI systems often require a huge amount of up front training data to get started, but targeted human intelligence can bootstrap the systems with relatively little data. Although humans may be slower initially, quickly bootstrapping to automated approaches provides a good balance, enabling human-AI systems to be scalable and rapidly deployable.

Acknowledgments

I would not be here without the support from my mentors, collaborators, friends, and family. They have made this work possible, made my PhD journey incredibly rewarding, and shaped me into a better person.

First, I would like to thank my advisor, Jeff Bigham, for the invaluable mentorship and support over the years. He has always been there for me, and I have had the greatest honor working with him and learning from him about research, life, academia, and industry. I would like to thank my thesis committee members Jodi Forlizzi, Chris Harrison, and Meredith Ringel Morris for their insightful feedback and continuous support. I would also like to thank my mentors and champions Gregory Abowd, Anind Dey, Scott Hudson, Chris Le Dantec, Yang Li, Jennifer Mankoff, Andrés Monroy-Hernández, Tim Paek, Thad Starner, and Rajan Vaish for guiding me at various stages of my career.

I would like to thank my awesome collaborators of this work for their help and support, including: Chieko Asakawa, Xiang 'Anthony' Chen, Patrick Clary, Ronnie Ghose, Suman Ghosh, Cole Gleason, Ken Goldman, Danna Gurari, Jaylin Herskovitz, Yasha Iravantchi, Anuraag Jain, Sasa Junuzovic, Ece Kamar, Shaun Kane, Jeeeun Kim, Junhan (Judy) Kong, Gierad Laput, Saige McVea, Sujeath Pareddy, Haoran Qi, Michael Rivera, Abigale Stangl, Jennifer Wortman Vaughan, Hanna Wallach, Sam White, Jason Wu, Frank F. Xu, Tom Yeh, and Yu Zhong. This work would not have been possible without them.

I would like to thank the research sponsors who supported my work, including National Science Foundation, the National Institute on Disability, Independent Living, and Rehabilitation Research, Google, Bosch, Microsoft Research, Snap Research, and the CMU Swartz Center for Entrepreneurship.

I would like to thank my friends and colleagues from CMU HCII, the Big Lab and FigLab, and beyond, including: Karan Ahuja, Nikola Banovic, Ilter Canberk, Patrick Carrington, Minsuk Chang, Fanglin Chen, Yan Chen, Judeth Oden Choi, Sauvik Das, Sai Ganesh, Xianheng Guan, Nathan Hahn, Kotaro Hara, Ting-Hao (Kenneth) Huang, Haojian Jin, Anna Kasunic, Rushil Khurana, Queenie Kravitz, Hanchuan Li, Haozhe Li, Toby Jia-Jun Li, Franklin Mingzhe Li, Chao Liu, Fannie Liu, Michael Xieyang Liu, Zhengzhong Liu, Michael Nebeling, Amy Pavel, Huaishu Peng, Stephanie Valencia, Xiaolong Wu, Françeska Xhakaj, Robert Xiao, Zheng Yao, Cheng Zhang, Dingtian Zhang, Shikun Zhang, Xiaoyi Zhang, Yang Zhang, Siyan Zhao, and many others who have helped me along the way.

Finally, I would like to thank my family for their love and support. I am forever grateful to my parents, Hongguang Cheng and Wu Guo, for your unconditional love and limitless support. And thank you to my wife Xu Wang, the love of my life, who I met through the PhD program, for the incredible support over the years. The past six years would have been worth it if all I did was meeting you, and I look forward for the many years to come.

Contents

1 Introduction 1

1.1 Human-AI Systems for Physical Interface Accessibility . . . 2

1.2 Human-AI Systems for Environmental Sensing . . . 4

1.3 Characteristics of Human-AI Systems for Visual Access . . . 5

1.4 Document Organization . . . 7

2 Background 9

2.1 AI-Powered Systems for Visual Access . . . 9

2.2 Human-Powered Systems for Visual Access . . . 10

2.3 Hybrid Human-AI Systems for Visual Access . . . 11

3 VizLens: A Robust and Interactive Screen Reader for Interfaces in the Real World 13

3.1 Introduction . . . 13

3.2 Related Work . . . 15

3.2.1 Computer Vision for Accessibility . . . 15

3.2.2 Crowd-Powered Systems for Visual Assistance . . . 16

3.3 Formative Study . . . 17

3.4 VizLens . . . 18

3.4.1 Implementation . . . 18

3.4.2 Initial Crowdsourced Segmenting and Labeling . . . 19

3.4.3 Retrieving Visual Elements . . . 21

3.4.4 Providing Feedback and Guidance . . . 24

3.5 User Evaluation . . . 24

3.5.1 Apparatus and Participants . . . 25

3.5.2 Design . . . 25

3.5.3 Tasks . . . 26

3.5.4 Results . . . 27

3.6 Technical Evaluation . . . 30

3.6.1 Crowdsourcing Performance . . . 31

3.6.2 Interface Robustness . . . 31

3.7 VizLens Version 2 . . . 32

3.7.1 VizLens::State Detection . . . 33

3.7.2 VizLens::LCD Display Reader . . . 33

3.7.3 VizLens::Wearable Cameras . . . 34

3.8 Discussion and Future Work 35
3.9 Conclusion 37

4 Cursor-based Interactions for Supporting Non-Visual Explorations 39
4.1 Introduction 39
4.2 Related Work 41
4.2.1 Computer Vision for Visual Access 41
4.2.2 Window Cursor Applications 42
4.2.3 Finger Cursor Applications 42
4.2.4 Touch Cursor Applications 43
4.3 Cursor-based Interactions 43
4.3.1 Mobile Application 44
4.3.2 Window Cursor Mode 45
4.3.3 Finger Cursor Mode 45
4.3.4 Touch Cursor Mode 45
4.4 User Study 46
4.4.1 Participants and Apparatus 46
4.4.2 Procedure 46
4.4.3 Tasks 47
4.4.4 Methods 48
4.5 Results 49
4.5.1 Task: Locate an Object 49
4.5.2 Task: Interpret Documents and Signs 50
4.5.3 Task: Read Labels and Enter Data on an Appliance 51
4.5.4 Task: Learn About Surroundings 52
4.5.5 Feedback From Low Vision Users 53
4.5.6 Social Acceptability 54
4.5.7 Key Takeaways 54
4.6 Discussion and Future Work 55
4.6.1 Prerequisites for Cursor Usability 55
4.6.2 Auditory Feedback 56
4.6.3 Cursor Interactions 56
4.6.4 Social Acceptability 56
4.6.5 Combination of Multiple Cursor Methods 57
4.7 Conclusions 57

5 Facade: Auto-generating Tactile Interfaces to Appliances 59
5.1 Introduction 59
5.2 Related Work 60
5.3 Formative Study 62
5.3.1 Procedure 62
5.3.2 Results: Design Considerations 63
5.4 Facade 64
5.4.1 Capture and Perspective Transformation 65

5.4.2 Crowdsourced Segmenting and Labeling 65
5.4.3 Fabricating Accessible Augmented Layer 66
5.5 Design Iterations 67
5.5.1 Iteration #1: Design Probe 67
5.5.2 Iteration #2: Material Exploration 68
5.5.3 Iteration #3: Improved Legibility 69
5.6 User Evaluation 70
5.6.1 Apparatus and Participants 70
5.6.2 Procedure 71
5.6.3 Results 72
5.7 Technical Evaluation 74
5.7.1 Interface Capture 75
5.7.2 Model Production 76
5.8 Discussion and Future Work 76
5.9 Conclusion 77

6 StateLens: A Reverse Engineering Solution for Dynamic Touchscreens 79
6.1 Introduction 79
6.2 Related Work 82
6.2.1 Reverse Engineering User Interfaces 82
6.2.2 Improving Accessibility for Physical Interfaces 83
6.3 Formative Study 84
6.3.1 Design Considerations 84
6.4 Risk-Free Exploration 85
6.4.1 Thingiverse Survey 85
6.4.2 Finger Ring Prototype 86
6.4.3 Design Variations 86
6.5 StateLens 87
6.5.1 Generating the State Diagram 88
6.5.2 Accessing the State Diagram 91
6.6 Technical Evaluation 93
6.6.1 Dataset 93
6.6.2 Generating the State Diagram 94
6.6.3 Accessing the State Diagram 94
6.7 User Evaluation 97
6.7.1 Apparatus and Participants 97
6.7.2 Procedure 98
6.7.3 Results 98
6.8 Discussion and Future Work 101
6.8.1 Technical Approach to Accessibility 101
6.8.2 Generalizability 101
6.8.3 Assistive Hardware for Automatic Screen Actuation 102
6.8.4 Limitations 103
6.9 Conclusion 104

7 Zensors++: Human-AI Camera Sensing in the Real World 105
7.1 Introduction . . . 105
7.2 Related Work . . . 108
7.2.1 Environment Sensing . . . 108
7.2.2 Computer Vision and Crowd-Powered Systems . . . 108
7.3 Discovery Deployment . . . 109
7.4 Zensors++ . . . 110
7.4.1 Compute . . . 110
7.4.2 Cameras . . . 111
7.4.3 Question Sensor Authoring . . . 111
7.4.4 Notifications . . . 112
7.4.5 Privacy Preservation . . . 112
7.4.6 Redundant Images . . . 112
7.4.7 Crowd Interface . . . 113
7.4.8 Crowd Disagreement . . . 113
7.4.9 HIT Queues . . . 113
7.4.10 Crowd Reliability . . . 114
7.4.11 Machine Learning . . . 114
7.4.12 Promoting Good Question Sensors . . . 114
7.5 Evaluation Deployment . . . 115
7.6 Results & Discussion . . . 115
7.6.1 Scale . . . 117
7.6.2 Raw Crowd Accuracy . . . 117
7.6.3 Question Sensor Accuracy . . . 117
7.6.4 User-Defined Question Errors . . . 118
7.6.5 Crowd Latency . . . 119
7.6.6 Hashing Performance . . . 120
7.6.7 Cost . . . 120
7.6.8 Machine Learning Accuracy . . . 121
7.6.9 Participant Use of Data . . . 122
7.6.10 Types of Question Sensors . . . 122
7.6.11 Proxy Questions . . . 123
7.6.12 Perceived Value . . . 123
7.6.13 Privacy and Sharing . . . 124
7.7 Future Work . . . 124
7.8 Conclusion . . . 125

8 Conclusion and Future Directions 127
8.1 Thesis Contributions . . . 127
8.2 Research Vision and Approach . . . 128
8.3 Future Directions . . . 129
8.3.1 Accessibility as a Driving Force of AI . . . 129
8.3.2 Enabling Access in More Contexts . . . 130
8.3.3 AI Datasets and Fairness . . . 131

8.3.4 Privacy Implications 132
8.4 Conclusion 132

Bibliography **135**

List of Figures

1.1 VizLens is a screen reader to help blind people access static physical interfaces. . 2

1.2 Facade is a crowdsourced fabrication pipeline that enables blind people to make flat physical interfaces accessible by independently producing a 3D-printed overlay of tactile buttons. 3

1.3 StateLens uses a hybrid crowd-computer vision pipeline to dynamically generate state diagrams about interface structures from point-of-view usage videos, and using the diagrams to provide interactive guidance and feedback to help blind users access the interfaces (a). 3D-printed accessories enable "risk-free exploration" (b). 4

1.4 Eight example sensors created by our participants using Zensors++, with regions of interest highlighted on the full camera image. Many sensors directly complemented and augmented people's existing work practices. 5

1.5 The human-AI systems described in this dissertation also differ in the types of visual sensors required for input (local and remote), and in the types of output provided for end users (audio feedback, tactile overlays, and data streams). 6

3.1 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. 15

3.2 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, s/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. 19

3.3 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. 21

3.4 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. 22

3.5 *(a)* Defining interaction point relative to the detected fingertip location. *(b)* Rules for assigning feedback based on button layout. 23

3.6 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. 26

3.7 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. 27

3.8 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. 28

3.9 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. 29

3.10 Answers to Likert scale questions indicating that participants found VizLens useful (1, 2, 3) and wanted to continue using it (4). 30

3.11 VizLens works robustly with a wide range of interfaces, including microwaves, printers, copiers, water machines, thermostats, laser cutters, toasters, remote controls, vending machines, etc. 32

3.12 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. 33

3.13 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. 34

3.14 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. 35

4.1 We define and compare cursor-based interactions that support non-visual attention to items within a complex visual scene: *(a)* window cursor, in which the user moves the device itself to scan the scene and receives information about what is in the center of the image; *(b)* vertical window cursor, a variation of window cursor that sacrifices the granularity on the vertical axis, but potentially facilitates locating the direction of a specific object; *(c)* finger cursor, in which the user moves their finger on the real world object they want to access and receives information about details near (or under) their fingertip; and *(d)* touch cursor, in which the visual scene is brought onto the device screen and the user moves their finger on the touchscreen to receive information about what they touch on the live camera image. 40

4.2 Illustrations of cursor-based interactions: *(a)* window cursor, *(b)* vertical window cursor, *(c)* finger cursor, and *(d)* touch cursor. 44
4.3 Screenshots of cursor-based interactions in different use cases: *(a)* window cursor, *(b)* vertical window cursor, *(c)* finger cursor, and *(d)* touch cursor. Colored bounding boxes show the recognized OCR results, the cursor region is drawn as a transparent overlay, while the rest of the image is covered with a semi-transparent dark overlay. 44
4.4 Study setup comparing the cursor techniques across a series of tasks representative of blind people's daily routines, including *(a)* locating an object in the environment, *(b)* interpreting documents and signs, *(c)* manipulating an appliance interface, and *(d)* learning about their surroundings. 47

5.1 Facade is a crowdsourced fabrication pipeline that enables blind people to make flat physical interfaces accessible by independently producing a 3D-printed overlay of tactile buttons. From left to right, we demonstrate example applications including microwave, refrigerator door, copier, and another microwave. Insets shows close views of individual embossed buttons. 60
5.2 Facade users capture a photo of an interface they would like to use with a fiducial marker attached to it (we use a dollar bill). Using perspective transformation, the interface image is warped to the front view and absolute measurements are calculated. Then this image is sent to multiple crowd workers, who work in parallel to quickly label and describe elements of the interface. Blind users can then customize settings of the labeling strategy, and these labels and preferences are used to generate the 3D models of a tactile layer matching the original controls. Finally, an off-the-shelf 3D printer fabricates the layer, which is then attached to the interface using adhesives. 61
5.3 Shapes inform users of different functionalities. For example, half spherical buttons without Braille label indicates number buttons *(a)*, while rectangular buttons with Braille labels indicate function buttons *(b)*. Users are also able to change settings to use symbols *(c)*, Braille *(d)*, or embossed letters *(e)* for buttons labels such as plus and minus. 67
5.4 A design probe tested with 6 blind participants. An augmented button set with Braille labels *(a)* is attached to the microwave *(c)*, and buttons are connected with thin bridges to facilitate pressing *(b)*. 68
5.5 Example printed overlays and legends generated by Facade. *(a)-(d)* demonstrate the different material combinations we tested in the design iterations (NinjaFlex with Braille, Flex+PLA Braille label, Flex+PLA Braille cover, and Flex+PLA embossed letter cover). Facade users can choose to print a legend for the abbreviations *(e)*. If a user does not have a 3D printer at home, models can also be printed through commercial printing services and mail-ordered. *(f) and (g)* show two example prints ordered from 3D Hubs using PolyFlex and SemiFlex materials. 70
5.6 Examples of the attached overlay performed by the participants. For P4 and P5, slight offset did not affect using the appliance. An exact alignment is shown in the top left corner. 74

5.7 Examples for image localization and warping on photos taken by the participants using a dollar bill as the fiducial marker. Boxes show in green represents warping results that were good enough for generating a usable tactile overlay model, while those shown in orange and red represents failure cases. 75

6.1 StateLens is a system that enables blind users to interact with touchscreen devices in the real world by *(i)* reverse engineering a structured model of the underlying interface, and *(ii)* using the model to provide interactive conversational and audio guidance to the user about how to use it. A set of 3D-printed accessories enable capacitive touchscreens to be used non-visually by preventing accidental touches on the interface. 80

6.2 StateLens uses a hybrid crowd-computer vision pipeline to dynamically generate state diagrams about interface structures from point-of-view usage videos, and using the diagrams to provide interactive guidance and feedback to help blind users access the interfaces. 81

6.3 A set of 3D-printed accessories that prevent the wearer from accidentally triggering touches while exploring the interface. When desired, the wearer can activate a touch using either a tilt motion (B-C and E-F) or by touching a conductive trace on the accessory with a finger (H-I). These accessories elegantly add "risk-free exploration" to existing capacitive touchscreen devices without modifying the underlying hardware or software, which has been a major hurdle for past efforts. 3D models of these accessories are available at: `https://github.com/mriveralee/statelens-3dprints` 87

6.4 Visualization of how StateLens represents the coffee machine interface structure as a state diagram. Note only some edges are shown. 88

6.5 Sample interactions between a user and the coffee machine natural language conversational agent StateLens automatically generated. 92

6.6 We evaluated how well StateLens reconstructs state diagrams from point-of-view usage videos across a wide range of interfaces, including an ATM, coffee machine (both graphical and text only), printer, projector control, room reservation panel, treadmill, ticket kiosk, Coca-Cola machine, subway ticket machine, washer, and car infotainment system. 93

6.7 StateLens maintains a relatively stable processing time for state detection as the number of states increases, compared to the linear increase in the baseline approach. 96

6.8 StateLens maintains a relatively stable error rate for state detection as the number of states increases, compared to the increasing trend in the baseline approach. . . 96

6.9 BrushLens is a smart assistive hardware proxy that would locate and actuate external interface controls automatically. Pictures show our first prototype built using solenoid actuators. 102

7.1 Example question sensors created by our participants, with regions of interest highlighted on the full camera image. 106

7.2 The system architecture of Zensors++ consists of four main components: user touchpoints (blue), image processing pipeline (pink), labeling system (green), and data storage (grey). 110
7.3 Screenshots of user interfaces. (A) End users highlight a region of interest in a camera image and add an associated natural language question. (B) Crowd workers provide answers for user-defined "question sensors" given an image region of interest. (C) End users can view real-time and historical data through visualizations. 111
7.4 Detailed statistics on all 63 question sensors from our evaluation deployment. . . 116
7.5 Yes/No question sensor accuracy when well formed (gray bar), formulated as a proxy question (blue bar), and when exhibiting four different error types (red bars).118
7.6 Histogram of end-to-end crowd latency. Median time to first answer was 120 seconds, while median time to answer of record was 355 seconds. 119
7.7 Mean perceptual hash hit rate over 24 hours, starting at noon, far left. 120
7.8 Distribution of daily cost across all question sensors without machine learning. More than 60% of question sensors cost $2/day or less to operate. 121

8.1 Human-AI systems could accelerate the development lifecycle of building AI products. 128
8.2 Design of AR-based visual guidance for different types of interactions: *(a)* On a coffee machine interface, a floating circle is displayed around the target button "coffee 50-50"; *(b)* On a movie ticket kiosk, an animated circle moving towards the target swipe direction is displayed to guide the user to swipe to the next page of movie list; *(c)* On a snack vending machine interface, an animation of inserting bills is displayed with the area highlighted, in order to guide the user to complete the payment; *(d)* On a text-heavy coffee machine, an overlay with bigger fonts and higher contrast is displayed on top of the original interface to make it more readable. 130

List of Tables

1.1 The human-AI systems described in this dissertation are for two application domains, accessibility and environmental sensing; use different kinds of human (end user, online crowd, and on-site crowd) and machine intelligence (computer vision, machine learning, dialogue system, and 3D printing); and are bootstrapped in either a one-off or a continuous manner. 6

3.1 Participant demographics for our user evaluation with 10 visually impaired users. 25

5.1 Interface, material and reading medium combinations used in design iteration #2 to improve attacheability, legibility, and pressibility. * Flex refers to NinjaFlex or SemiFlex for flexible material printing. ** Required manual intervention for raised buttons. 69

5.2 Participant demographics for our user evaluation with 11 blind users. 71

5.3 We experimented with a variety of materials and printing techniques. Test prints were ordered from 3D Hubs. 76

6.1 Categorization of our Thingiverse survey results related to assistive technologies, touchscreens, and finger-based interactions. The number of items is shown next to each category name. 85

6.2 Aggregated precision, recall, and F1 scores of the state diagram generation process of StateLens using a combination of features – Screen Detection (SD), SURF, and OCR – and with stationary, hand-held, and web (with links) usage videos. Each cell shows the precision (top left), recall (top right), and F1 scores (bottom). Bold values identify the feature combination with the best performance. 95

6.3 Participant demographics for our user evaluation with 14 visually impaired users. Thirteen were blind, and one (P12) had low vision. 97

6.4 Results from the 3D-printed accessory study, showing mean and standard deviation (in parentheses). 99

6.5 Initial testing on several BrushLens hardware proxy designs. Using solenoid with conductive rubber seems to be the most promising solution. 103

Chapter 1

Introduction

The world is full of visual information that is not easily accessible. For blind people, frustrating accessibility problems because of vision are commonplace and pervasive. For example, they frequently encounter inaccessible physical interfaces in their everyday lives that are difficult, frustrating, and often impossible to use independently. For space owners, actionalizing camera streams into sensor data can help them better monitor, manage, and optimize the environment. Despite the advantages, these visual information are often left uncaptured, and cameras are merely used to view a remote area.

Two trends are converging that make solving these problems tractable: artificial intelligence (AI) and human computation. With recent and impressive advances, AI shows promise in understanding the visual world with computer vision. However, AI systems struggle in many real-world, uncontrolled situations, and do not easily generalize across diverse human environments. Humans, on the other hand, can be more robust and flexible in solving real-world problems that cannot be handled by AI. However, using human intelligence is slow and expensive, thus not scalable.

In my work, I create hybrid human- and AI-powered intelligent interactive systems to provide access to visual information in the real world. By combining the advantages of humans and AI, these systems can be nearly as robust and flexible as humans, and nearly as quick and low-cost as automated AI, enabling us to solve problems that are currently impossible with either alone. I developed and deployed human-AI systems for two application domains: accessibility and environmental sensing. To make physical interfaces accessible for blind people, I developed systems to interpret static and dynamic interfaces, enabling blind people to independently access them through audio feedback or tactile overlays [76, 78, 82]. For environmental sensing, I developed and deployed a camera sensing system that collects human labels to bootstrap automatic processes to answer real-world visual questions, allowing end users to actionalize AI in their everyday lives [79]. AI systems often require a huge amount of up front training data to get started, but targeted human intelligence can bootstrap the systems with relatively little data. Although humans may be slower initially, quickly bootstrapping to automated approaches provides a good balance, enabling human-AI systems to be scalable and rapidly deployable.

The goal of my research is to create intelligent interactive systems that solve AI-hard real-world problems. These systems collect data for users' immediate needs, in order to build a model to work in the moment. To the end users, these systems are always intelligent and smart. But under the hood, large-scale data can be collected, and automation can be achieved over time to support these user needs.

1.1 Human-AI Systems for Physical Interface Accessibility

The world is full of physical interfaces that are inaccessible to blind people, from microwaves and information kiosks to thermostats and checkout terminals. The VizWiz dataset [85] I helped release showed that many blind people sought assistance using such interfaces. Blind people cannot access these interface because the buttons are tactually indistinguishable, and the screens contain visual information that they cannot read. Creating new devices that are accessible could work, but is unlikely to make it into all devices produced due to cost, let alone the substantial legacy of inaccessible devices already in the world.

To make physical interfaces accessible, in Chapter 3, I first introduce *VizLens* [76], a robust and interactive screen reader for real-world static interfaces (Figure 1.1). To work robustly, VizLens combines on-demand crowdsourcing and real-time computer vision. When a blind person encounters an inaccessible interface for the first time, they use a smartphone camera 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 or other controls), and describes each element. Later, when the person wants to use the interface, they open the VizLens application, point it towards the interface, and hover a finger over it. VizLens uses SURF-based object matching techniques to match the crowd-labeled reference image to the image captured in real-time, and track the user's finger to retrieve and provide audio feedback and guidance. Deep CNNs may increase the robustness, but the beauty of our approach is that even simple computer vision techniques work. With such instantaneous feedback, VizLens allows blind users to interactively explore and use inaccessible interfaces. VizLens trades off

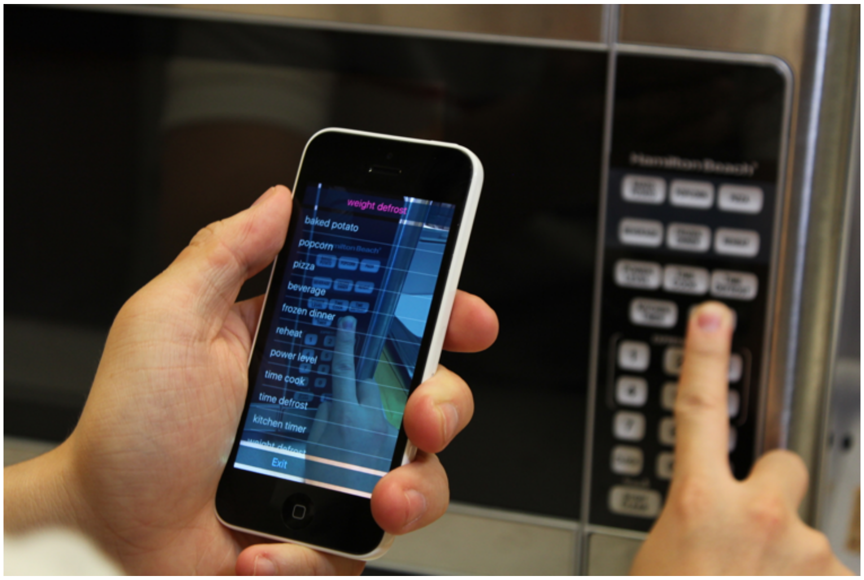


Figure 1.1: VizLens is a screen reader to help blind people access static physical interfaces.

the advantages of humans and computer vision to be nearly as robust as a person in interpreting the interface and nearly as quick and low-cost as a computer vision system to re-identify the interface and provide real-time feedback. In Chapter 4, I further explore cursor-based interactions to support non-visual explorations by blind users [80], integrating VizLens's real-world scene reader interaction as a type of finger cursor.

Blind people often label home appliances with Braille stickers, but doing so generally requires sighted assistance to identify the original functions and apply the labels. To address this challenge, in Chapter 5, I introduce *Facade* [78], a crowdsourced fabrication pipeline that enables blind people to independently create 3D-printed tactile overlays for inaccessible appliances (Figure 1.2). Blind users capture a photo of an inaccessible interface with a readily available fiducial marker for recovering size information using perspective transformation. This image is then labeled by crowd workers. Facade then generates a 3D model for a layer of tactile and pressable buttons that fits over the original controls, which the blind users can customize using the iOS app. Finally, a home 3D printer or commercial service can be used to fabricate the layer. We went through several design iterations to determine the most effective overlay design, material configuration, and printer settings to make the 3D-printed overlays easy to attach, read, and press. Facade makes end-user fabrication accessible to blind people, by shifting the sighted assistance to a virtual crowd working with computer vision. Facade combines a human-AI interpretation pipeline with an accessible 3D printing application.

VizLens and Facade enable blind users to access many *static* interfaces. To make *dynamic* touchscreens such as public kiosks and payment terminals accessible, in Chapter 6, I introduce *StateLens*, a three-part reverse engineering solution [82]. First, using a hybrid crowd-computer vision pipeline (Figure 1.3a), StateLens reverse engineers the underlying state diagrams of existing interfaces using point-of-view videos found online or taken by users. Second, using the state

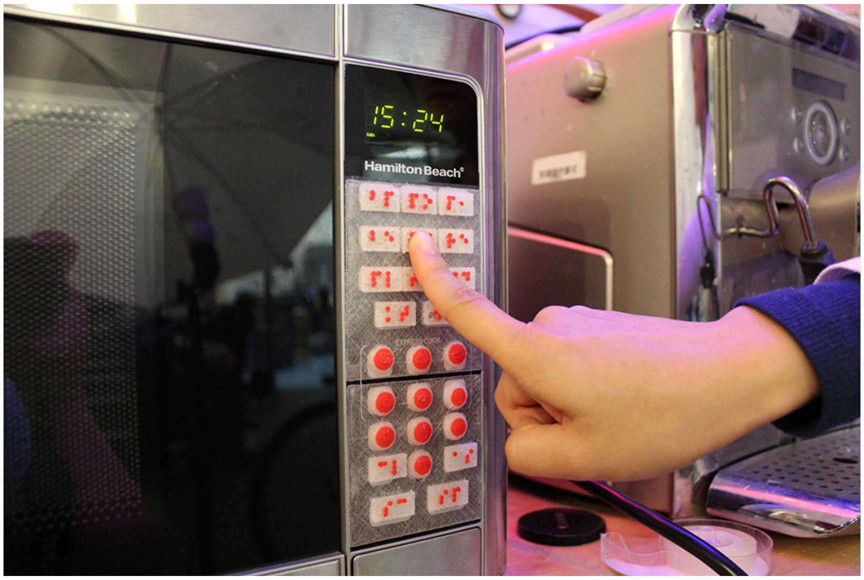


Figure 1.2: Facade is a crowdsourced fabrication pipeline that enables blind people to make flat physical interfaces accessible by independently producing a 3D-printed overlay of tactile buttons.

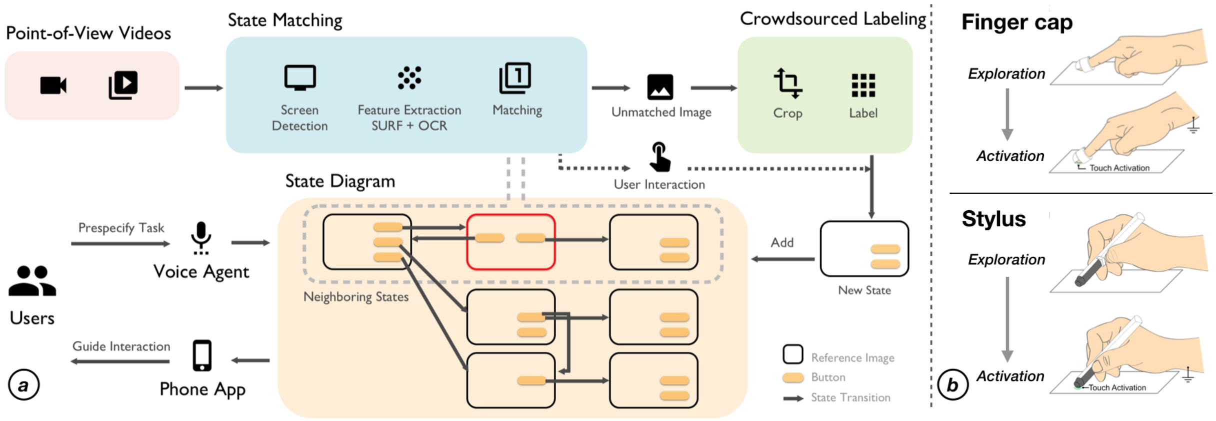


Figure 1.3: StateLens uses a hybrid crowd-computer vision pipeline to dynamically generate state diagrams about interface structures from point-of-view usage videos, and using the diagrams to provide interactive guidance and feedback to help blind users access the interfaces (a). 3D-printed accessories enable "risk-free exploration" (b).

diagrams, StateLens automatically generates conversational agents to guide blind users through specifying the tasks that the interface can perform, allowing the StateLens iOS application to provide interactive guidance and feedback so that blind users can access the interface. Finally, to address the "Midas touch problem" of accidental triggers during exploration, we designed a set of 3D-printed accessories (Figure 1.3b: finger cap and stylus) that allow users to explore without touching the screen, and perform a gesture to activate touch at a desired position. These accessories bring "risk-free exploration" to public capacitive touchscreens without modifying the underlying hardware or software, which is core to accessible touchscreen interaction. Our technical evaluation with 12 touchscreen devices and over 70K video frames showed that StateLens can accurately reconstruct interfaces from stationary, hand-held, and web videos; and through a user study with 14 blind participants, we showed that the complete system enables blind users to access otherwise inaccessible dynamic touchscreens.

StateLens addresses the very hard case in which blind users encounter a touchscreen in the real world that is inaccessible, which they cannot modify the hardware or software, and whose screen updates dynamically to show new information and interface components. Furthermore, StateLens takes advantage of different kinds of human intelligence: humans who provide access and collect videos at the interface to build up the training data, and online crowds who provide necessary labels to bootstrap automation.

1.2 Human-AI Systems for Environmental Sensing

Beyond enabling access to physical interfaces for blind people, I also explored environmental sensing platforms for understanding the visual world. In Chapter 7, I introduce the development and deployment of *Zensors++* [79], a human-AI camera sensing system to answer natural language user questions based on camera streams. To create a sensor, users select a camera, drag

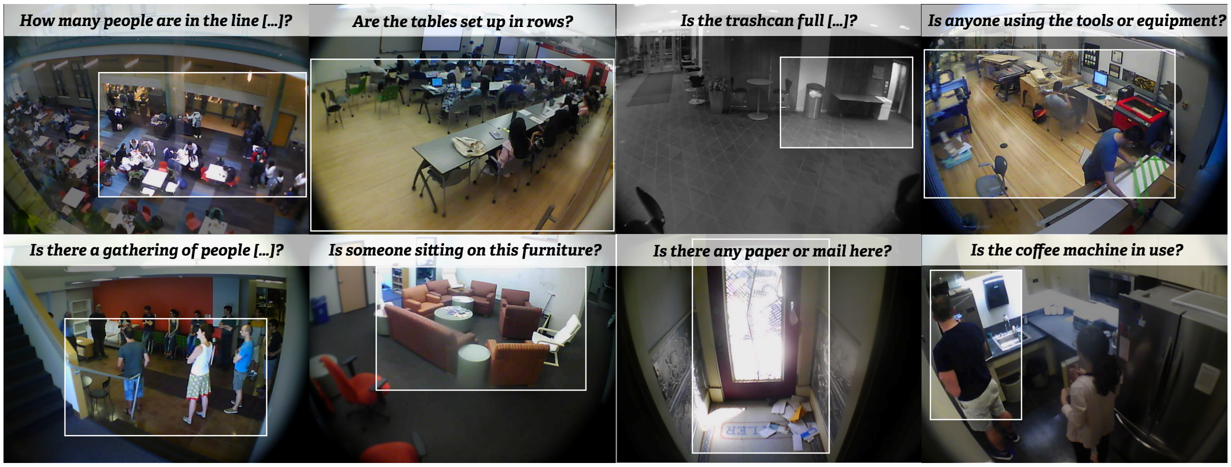

How many people are in the line [...]?
Are the tables set up in rows?
Is the trashcan full [...]?
Is anyone using the tools or equipment?
Is there a gathering of people [...]?
Is someone sitting on this furniture?
Is there any paper or mail here?
Is the coffee machine in use?

Figure 1.4: Eight example sensors created by our participants using Zensors++, with regions of interest highlighted on the full camera image. Many sensors directly complemented and augmented people's existing work practices.

a bounding box to select a region of interest, and ask a natural language question. At first, crowd workers provide near-instant answers for users' questions. Over time, Zensors++ relies on the crowd less, as answers can be automated through perceptual image hashing and continuously-evolving machine learning. We deployed Zensors++ in the wild, with real users, over many months and environments, generating 1.6 million answers for nearly 200 questions created by our participants, costing roughly 6/10ths of a cent per answer delivered. We demonstrated that crowd workers were able to provide labels quickly (~6s) and at scale, and that the system could hand-off to image hashing and machine learning classifiers for ~75% of all questions.

Participants created a wide range of sensors for their use cases (Figure 4). For example, for "Are the tables set up in rows?", the instructor used it to decide whether he needed to go to the classroom early to arrange the room before lecture. For "Is someone sitting on this furniture?", the program director was using Zensors++ to conduct physical A/B testing on different furniture arrangements. For "Is the trash can full?", the building manager was able to get email notifications when the trashcan is full, so he could better allocate resources to clean them up, rather than doing periodic checking manually. For "How many people are in line at the cash register?", a restaurant manager was interested in using the longitudinal data to identify consumption patterns to better plan and prepare food, while students and faculties were more interested in knowing how long the line is. Overall, our deployments demonstrated that human-AI, camera-based sensing can work at scale. Zensors++ relies on end users to define questions of interest and specify image region, as well as online crowd workers to provide labels when necessary. Then it relies on machine intelligence to automate over time to reduce the cost and latency.

1.3 Characteristics of Human-AI Systems for Visual Access

The human-AI systems developed and deployed in this dissertation are bootstrapped in either a one-off or a continuous manner (Table 1.1). One-off bootstrapping requires a one-time data input

System	Application Domain	Human Intelligence	Machine Intelligence	Bootstrapping
VizLens	Accessibility	End user, online crowd	Computer vision (CV)	One-off
Facade	Accessibility	End user, online crowd	CV, 3D printing	One-off
StateLens	Accessibility	Online crowd, on-site crowd	CV, dialogue system	Continuous
Zensors++	Environmental Sensing	End user, online crowd	CV, machine learning	Continuous

Table 1.1: The human-AI systems described in this dissertation are for two application domains, accessibility and environmental sensing; use different kinds of human (end user, online crowd, and on-site crowd) and machine intelligence (computer vision, machine learning, dialogue system, and 3D printing); and are bootstrapped in either a one-off or a continuous manner.

from the end user, whereas continuous bootstrapping suggests the system can continuously gather input to improve itself. VizLens and Facade require one-off examples from end-users to bootstrap the systems, while StateLens and Zensors++ employ more continuous methods using videos and continuous image streams across time.

For input, these systems also differ in the types of visual sensors they require (Figure 1.5). When end users are physically located in the environment to bootstrap the system, they serve as local visual sensors. When physically locating in the environment is not convenient or possible, remote visual sensors can be utilized. For example, VizLens and Facade employ local visual sensors for input, which are the end users' mobile cameras. While in StateLens and Zensors++, the systems also take input from remote visual sensors, such as videos contributed and cameras set up by other people.

For output, these systems provide the appropriate types of feedback and assistance to end users depending on the contexts and applications (Figure 1.5). For example, VizLens and StateLens

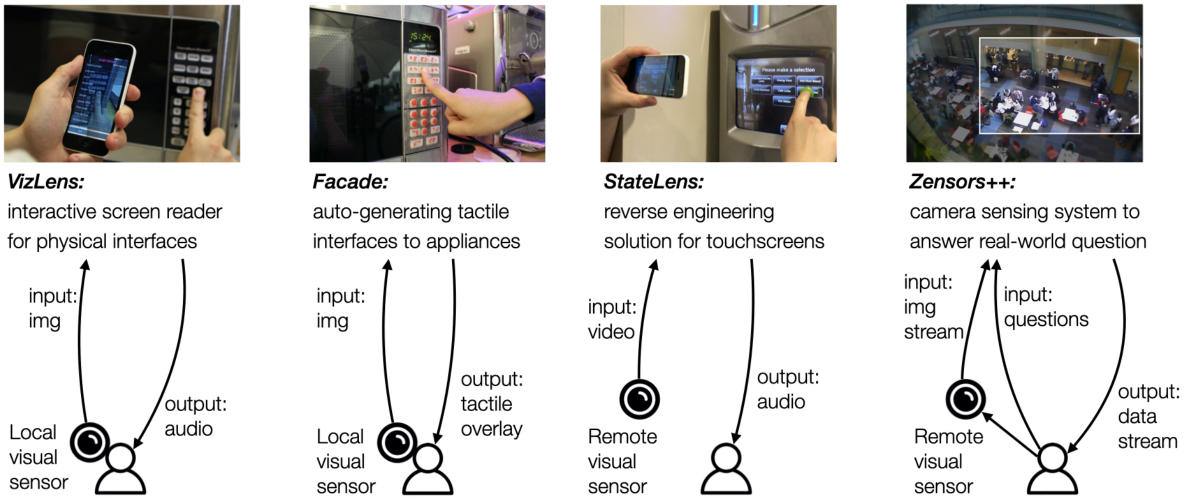


Figure 1.5: The human-AI systems described in this dissertation also differ in the types of visual sensors required for input (local and remote), and in the types of output provided for end users (audio feedback, tactile overlays, and data streams).

provide audio guidance and feedback as output to help blind people access physical interfaces especially in public spaces that cannot be labeled with Braille dots and stickers. However, for static interfaces in blind people's homes, physical tactile overlays generated by Facade are more appropriate, because requiring blind users to hold a device every time when using home appliances might be cumbersome. Finally, Zensors++ provides data stream as output in the forms of visualizations and notifications, for end users to better monitor, manage, and optimize the environment.

1.4 Document Organization

In this dissertation, I introduce several hybrid human- and AI-powered intelligent interactive systems to provide access to visual information in the real world. By combining the advantages of humans and AI, these systems can be nearly as robust and flexible as humans, and nearly as quick and low-cost as automated AI, enabling us to solve problems that are currently impossible with either alone. In Chapter 2, I outline prior work in this space, primarily focusing on human-powered and AI-powered systems for visual access, as well as existing hybrid approaches to combine human and machine intelligence.

Then, I introduce the human-AI systems I developed and deployed for two application domains: accessibility and environmental sensing. In Chapter 3, 5, and 6, I introduce human-AI systems to make physical interfaces accessible for blind people, enabling blind people to independently access them through audio feedback or tactile overlays. Additionally, in Chapter 4, I describe a set of cursor-based interactions to support non-visual explorations by blind users, integrating the real-world scene reader interaction as a type of finger cursor. Then, in Chapter 7, I introduce a human-AI camera sensing system that collects human labels to bootstrap automatic processes to answer real-world visual questions, allowing end users to actionalize AI in their everyday lives.

Finally, I discuss the contributions and impact of this work, my research vision and approach, as well as future research directions enabled by this dissertation.

Chapter 2

Background

Two trends are converging to provide better access and understanding to visual information in the real world: artificial intelligence (AI) and human computation. In this chapter, I outline prior work in this space, first focusing on AI-powered and human-powered systems for visual access. Then, I discuss existing hybrid approaches to combine human and machine intelligence, and situate my work at this intersection.

2.1 AI-Powered Systems for Visual Access

With recent and impressive advances, AI shows promise in understanding the visual world with computer vision.

For accessibility purposes, a number of systems have been developed to help blind people access visual information using computer vision. For example, many systems have been developed to help blind people read visual text via OCR, e.g., KNFB Reader [109] 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 [103]. OCR can do reasonably well in providing access to text that is well-formatted, but recognizing text in the real world can be difficult [132]. Even detecting that text exists in natural photographs can be difficult [100].

Since acquiring a high-quality photograph is often a prerequisite for further computer vision processing, several prior systems have been developed to assist blind people in taking better photographs [99, 128, 171, 176, 184]. One challenge with these systems supporting "blind photography" is that it is often unclear what the user is trying to take a picture of.

Furthermore, related to my work on making physical interfaces accessible for blind people, vision systems have been built to help blind people read the LCD panels on appliances [65, 137, 163]. These systems have tended to be fairly brittle, and have generally only targeted reading text and not actually using the interface.

Recently, deep learning approaches have been applied to general object recognition, in products such as Aipoly [168] and Microsoft's Seeing AI [133]. For example, Seeing AI [133] provides functionalities for blind users to take a picture and get an overview description of the captured scene. Other capabilities have been developed to help blind people recognize faces [133, 145], identify products [126, 133, 145], or count money notes [125, 145]. These approaches can work reasonably well, although can only recognize a preset number of objects, and often require a huge amount of up front training data to get started.

For environmental sensing purposes, computer vision approaches has also come close, as cameras offer high-fidelity data that can be processed to yield sensor-like feeds. Consumer home cameras have started offering rudimentary computationally-enhanced functions, such as motion [54] and intruder detection [140]. In commercial and municipal camera systems [127], computer vision has been applied to e.g., count cars and people [49, 135], read license plates [50], control quality [172], analyze sports [18], recognize faces [166] and monitor road surfaces [59]. In general, these computer-vision-powered systems require extensive training data and on-site tuning to work well. For example, FaceNet [152] achieved human-level face detection accuracy, but required a team of researchers to collect and prepare over 100 million images for training. This is obviously impractical for the long-tailed distribution of scenarios and the many bespoke questions users may wish to ask about their environments [22, 35, 85].

Despite the advantages of AI-powered systems, there are also huge challenges when applying them in the real world. For example, they often require huge amount of training data up front to get started. Whereas human can start off immediately given very few examples. AI systems also struggle in many real-world, uncontrolled situations, and do not easily generalize across diverse human environments. Achieving human-level abstractions and accuracy is a persistent challenge, leading to the creation of many human-powered systems for visual access.

2.2 Human-Powered Systems for Visual Access

Humans, on the other hand, can be more robust and flexible in solving real-world problems that cannot be handled by AI.

Researchers have explored using “the crowd” — people recruited from the Web to power real-time interactive systems. Crowdsourcing systems access “human intelligence” through online marketplaces such as Amazon Mechanical Turk [4]. Crowd-powered systems have been developed for various applications, *e.g.*, document editing and shortening [23], user interface control [119], real-time captioning [118]. These systems operate quickly by both lowering the latency to recruit workers [21, 26], and allowing workers to work synchronously together once recruited.

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, a system that lets blind people take a picture, speak a question, and get answers back from the crowd within approximately 30 seconds [26]. More than 10,000 users have asked more than 100,000 questions using VizWiz [35, 85]. Users often ask questions about interfaces [35], 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.

Other systems provide more continuous support. For example, Chorus:View [121] pairs a user with a group of crowd workers using a managed dialogue similar to [122] and a shared video stream. Be My Eyes [20] and Aira [2] matches users to a single person over a shared video stream. These systems could more easily assist blind users with using an interface, but assisting in this way is cumbersome and slow.

Existing crowd-powered systems provide promising solutions for many AI-hard problems — those that require human-level intelligence to solve. However, using human intelligence is slow and expensive, thus not scalable.

2.3 Hybrid Human-AI Systems for Visual Access

By combining the advantages of humans and AI, hybrid human-AI systems can be nearly as robust and flexible as humans, and nearly as quick and low-cost as automated AI, enabling us to solve problems that are currently impossible with either alone.

Regarding the disadvantages of using crowd only-powered systems such as VizWiz for visual assistance for blind people, other systems have expanded their utilities by integrating additional AI capabilities. For example, VizWiz::LocateIt [27] 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. RegionSpeak [185] enables spatial exploration of the layout of objects in a photograph using a touchscreen. Users send 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.

Researchers have also mixed computer vision and crowd-powered approaches to create systems that learn over time. For example, Legion:AR uses on-demand crowd labeling to train an HMM-based activity recognizer [120]. Likewise, Flock [48] and Alloy [43] train hybrid crowd-machine learning classifiers to enable fast prototyping of machine learning models that can improve on both algorithmic performance and human judgment, accomplishing tasks where automated feature extraction is not yet feasible. VATIC [174] uses crowd workers to annotate video with labels and object bounding boxes, providing critical training data to bootstrap machine learning. Zensors [116] fuses real-time human intelligence from online crowd workers with automatic approaches to provide robust, adaptive, and readily deployable intelligent sensors. Tohme [88] 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. JellyBean [151] introduce a suite of hybrid algorithms that combines the best of crowds and computer vision to count objects in images more accurately than either alone.

Mechanisms of combining human and machine intelligence have also been applied to automate critical tasks in other domains, including managing dialogue [94, 96], synthesizing online information [87], answering difficult queries [63], and organizing teams [150].

Building on prior literature on AI-powered, human-powered, and hybrid human- and AI-powered systems for visual access, this dissertation contributes novel hybrid human-AI intelligent interactive systems that combine human and machine intelligence to tackle accessibility and environmental sensing challenges in the real world, and in real time. These systems collect data for users' immediate needs, in order to build a model to work in the moment. To the end users, these systems are always intelligent and smart. But under the hood, large-scale data can be collected, and automation can be achieved over time to support these user needs.

The key observations are that AI systems often require a huge amount of up front training data to get started, but targeted human intelligence can bootstrap the systems with relatively little data. Although humans may be slower initially, quickly bootstrapping to automated approaches provides a good balance, enabling human-AI systems to be scalable and rapidly deployable.

Chapter 3

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

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.

3.1 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 [65, 137]. 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 [55, 169]. Computer vision approaches have been explored, but are usually brittle and specific to interfaces and tasks [65]. 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 work introduces *VizLens*, a robust interactive screen reader for real-world interfaces (Figure 3.1). Just as digital screen readers were first implemented by interpreting the visual information the computer asks to display [165], 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, s/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 or other controls), and describes each element (Figure 3.1A). Later, when the person wants to use the interface, s/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 3.1B). 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 work 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,

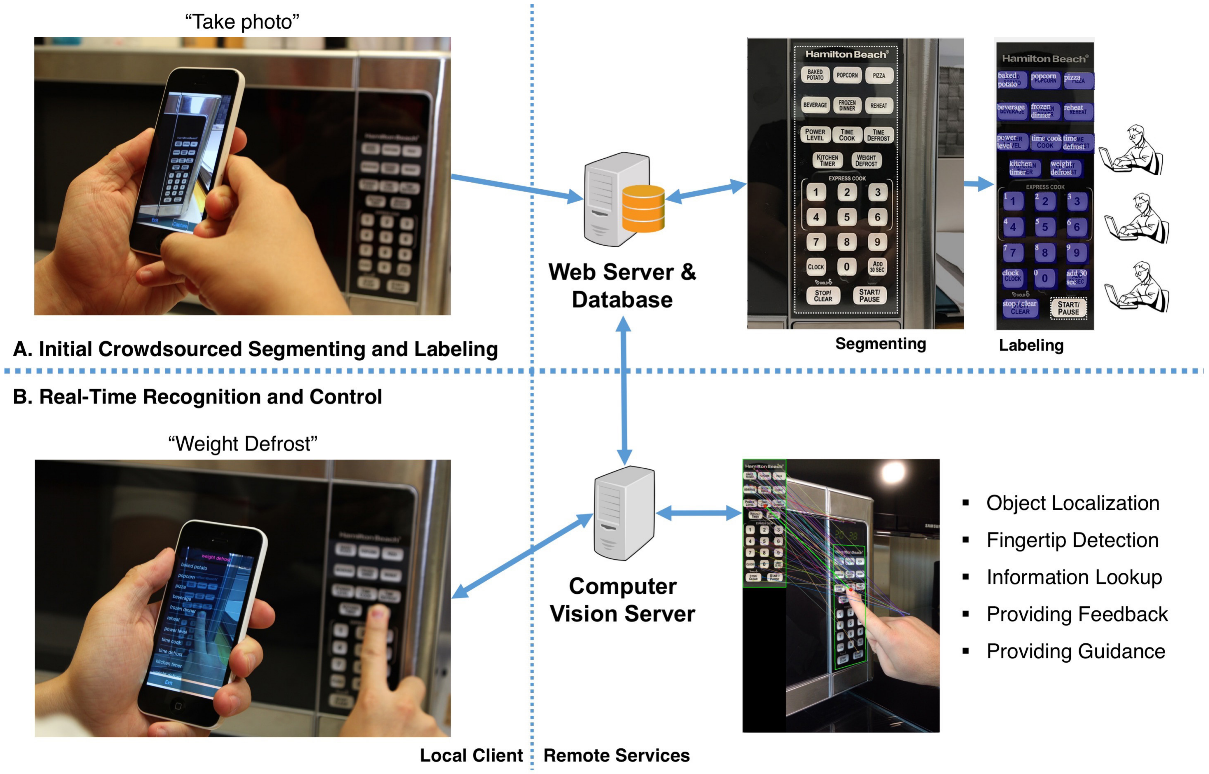


Figure 3.1: VizLens users take a picture of an interface they would like to use, it is interpreted quickly and robustly by multiple crowd workers in parallel, and then computer vision is able to give interactive feedback and guidance to users to help them use the interface.

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.

3.2 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.

3.2.1 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 [65, 137, 163]. 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 [99, 128, 171, 176, 184]. 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. For instance, the KNFB Reader [109] 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 [103]. OCR can do reasonably well in providing access to text that is well-formatted, but recognizing text in the real world can be difficult [132]. Even detecting that text exists in natural photographs can be difficult [100]. 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 [168] and Microsoft’s Seeing AI [133]. These approaches can work reasonably well, although can only recognize a preset number of objects (e.g., Aipoly recognizes approximately 1000 pre-defined 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 [156] 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 [139]. OrCam is a product that uses a head-mounted camera to make available various computer vision applications targeting low vision people [145]. Foresee enables real-world objects to be magnified using a wearable camera and head-mounted display [183]. 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.

3.2.2 Crowd-Powered Systems for Visual Assistance

Crowd-powered systems have been developed for various applications, e.g., document editing and shortening [23], user interface control [119], real-time captioning [118]. These systems operate quickly by both lowering the latency to recruit workers [21, 26], and allowing workers to work synchronously together once recruited. At least two existing projects have explored the combination of computer vision and crowdsourcing. Zensors [116] fuses real-time human intelligence from online crowd workers with automatic approaches to provide robust, adaptive, and readily deployable intelligent sensors. Tohme [88] 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 (The “Viz” prefix comes from how some tech-savvy blind people refer to one another, e.g., “are you viz?”), a system that lets blind people take a picture, speak a question, and get answers back from the crowd within approximately 30 seconds [26]. More than 10,000 users have asked more than 100,000 questions using VizWiz [35, 85]. Users often ask questions about interfaces [35], 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 [121] pairs a user with a group of crowd workers using a managed dialogue similar to [122] and a shared video stream. Be My Eyes [20] matches users to a single volunteer over a shared video stream. 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 [27] 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 [185] enables spatial exploration of the layout of objects in a photograph using a touchscreen. Users send 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.

3.3 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 [144].
- 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.

3.4 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 s/he intends to use by providing feedback and guidance.

3.4.1 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.

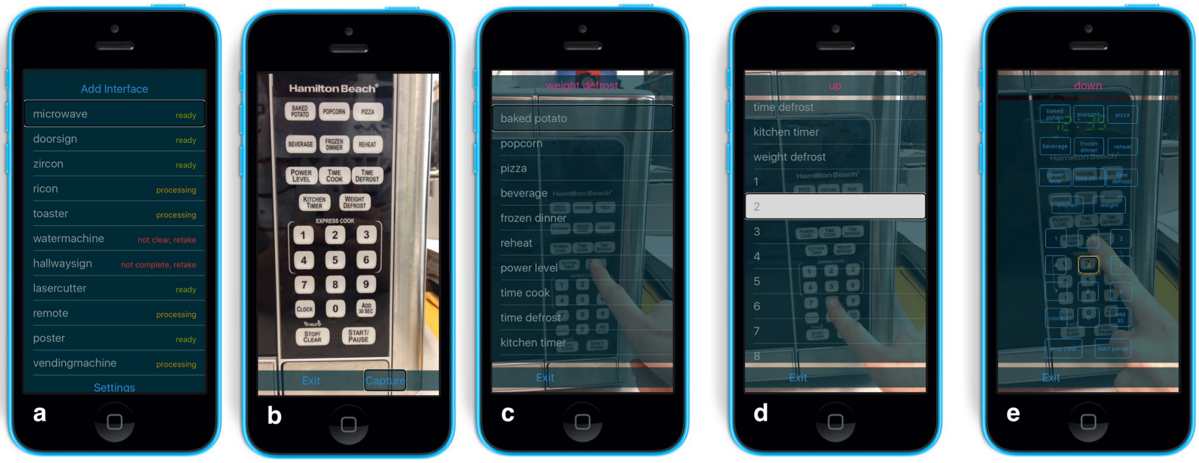


Figure 3.2: VizLens mobile app interfaces. *(a)* App's main screen, listing all available interfaces and status, as well as Add Interface and Settings options. *(b)* Interface for Add Interface, where the user can take a photo and denote a name for it. *(c)* When user selects an interface, s/he can start 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.

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, 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 [74], 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.

3.4.2 Initial Crowdsourced Segmenting and Labeling

The first time a user encounters an interface, s/he uses VizLens to take a photo of the interface (Figure 3.2b), 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.

Step 1: Segmenting Interface Region

In step 1 (Figure 3.3a), 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 3.2a). 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

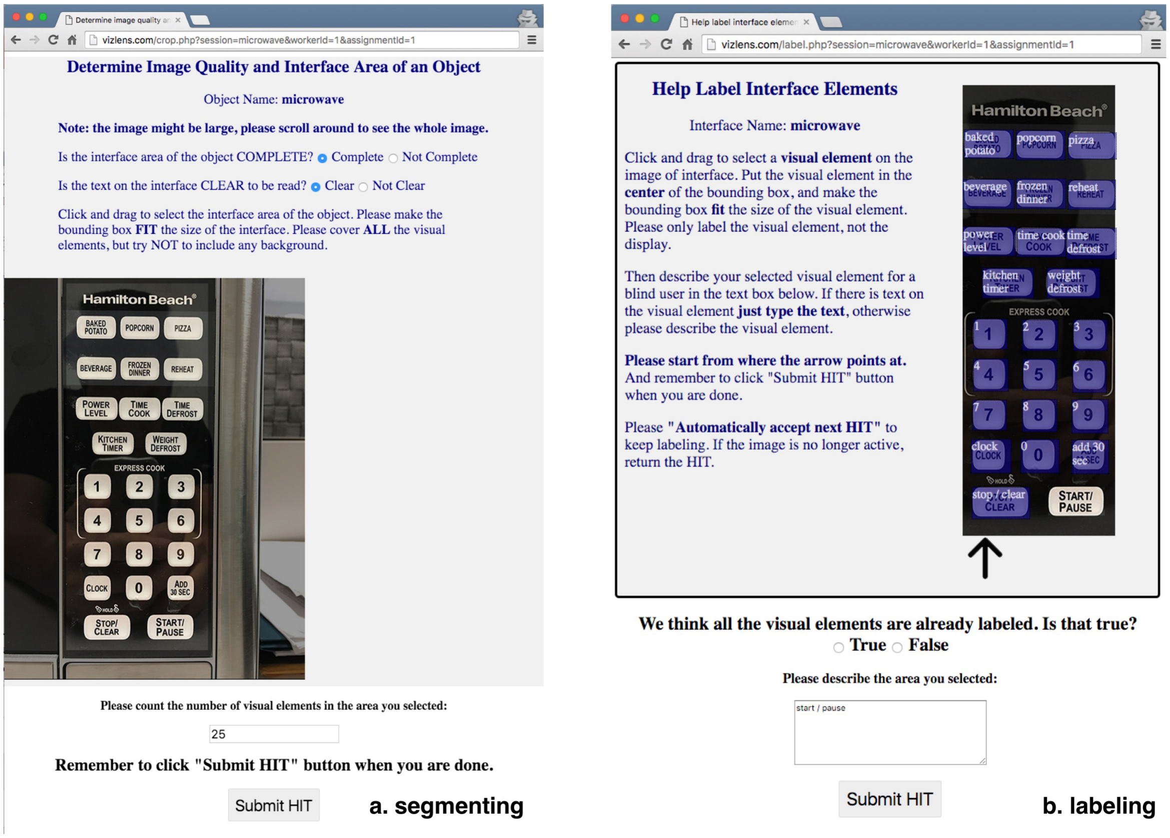
In step 2, crowd workers are instructed to draw bounding boxes around all of the individual visual elements (e.g., buttons) within the interface area (Figure 3.3b); and provide a text annotation for each element (such as labeling buttons as ‘baked potato’, ‘start/pause’). Similar to RegionSpeak [185], 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 3.3b). 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 3.3b). Once agreement is reached, this image then becomes the reference image (Figure 3.4a).

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


Determine image quality an
vizlens.com/crop.php?session=microwave&workerId=1&assignmentId=1

Determine Image Quality and Interface Area of an Object

Object Name: **microwave**

Note: the image might be large, please scroll around to see the whole image.

Is the interface area of the object COMPLETE? ● Complete ○ Not Complete

Is the text on the interface CLEAR to be read? ● Clear ○ Not Clear

Click and drag to select the interface area of the object. Please make the bounding box **FIT** the size of the interface. Please cover **ALL** the visual elements, but try NOT to include any background.

Please count the number of visual elements in the area you selected:

25

Remember to click "Submit HIT" button when you are done.

Submit HIT
a. segmenting
Help label interface elemen
vizlens.com/label.php?session=microwave&workerId=1&assignmentId=1

Help Label Interface Elements

Interface Name: **microwave**

Click and drag to select a **visual element** on the image of interface. Put the visual element in the **center** of the bounding box, and make the bounding box **fit** the size of the visual element. Please only label the visual element, not the display.

Then describe your selected visual element for a blind user in the text box below. If there is text on the visual element **just type the text**, otherwise please describe the visual element.

Please start from where the arrow points at. And remember to click "Submit HIT" button when you are done.

Please "**Automatically accept next HIT**" to keep labeling. If the image is no longer active, return the HIT.

We think all the visual elements are already labeled. Is that true?
○ **True** ○ **False**

Please describe the area you selected:

start / pause

Figure 3.3: 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.

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 [116].

3.4.3 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.

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) [19] feature detector with hessian threshold set to 400 to compute key points and feature vectors in both the reference (Figure 3.4a) and the input image (Figure 3.4b). Note that the reference image can be pre-

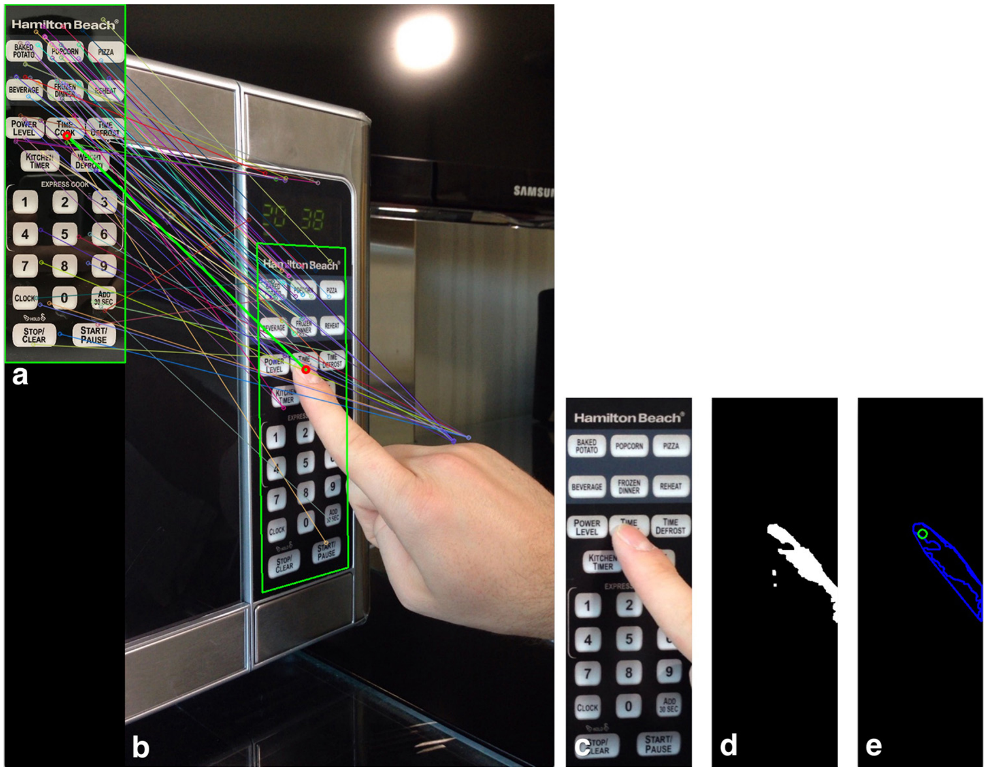


Figure 3.4: Real-time recognition and control using VizLens. Recognition result, showing *(a)* reference image and *(b)* input image. *(c)* Input image warped to reference image's frame allowing the coordinates of the elements previously labeled to be retrieved. *(d)* Result of skin color thresholding, and *(e)* calculated fingertip location.

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 3.4b, 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 3.4c), adjusts the lighting of the warped image to match the reference image, and detects the fingertip's location using skin color thresholding [173]. 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 3.4d). Then, VizLens uses the largest contour of the convex hull to detect the

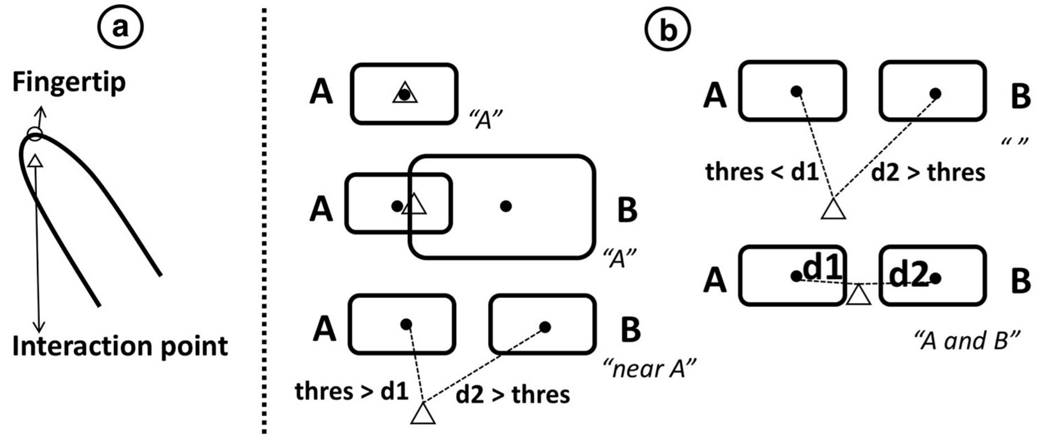


Figure 3.5: *(a)* Defining interaction point relative to the detected fingertip location. *(b)* Rules for assigning feedback based on button layout.

fingertip's location (Figure 3.4e). 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.

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 3.5a).

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 3.5b):

- 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.

3.4.4 Providing Feedback and Guidance

Providing Feedback

After identifying the visual element, VizLens triggers the VoiceOver screen reader on iOS to read its description (Figure 3.2c). 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 [30] 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 3.2d).

The path of navigation follows the Manhattan Distance [29] 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.

3.5 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”).

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

Table 3.1: Participant demographics for our user evaluation with 10 visually impaired users.

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).

3.5.1 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 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 3.1.

3.5.2 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 3.6.

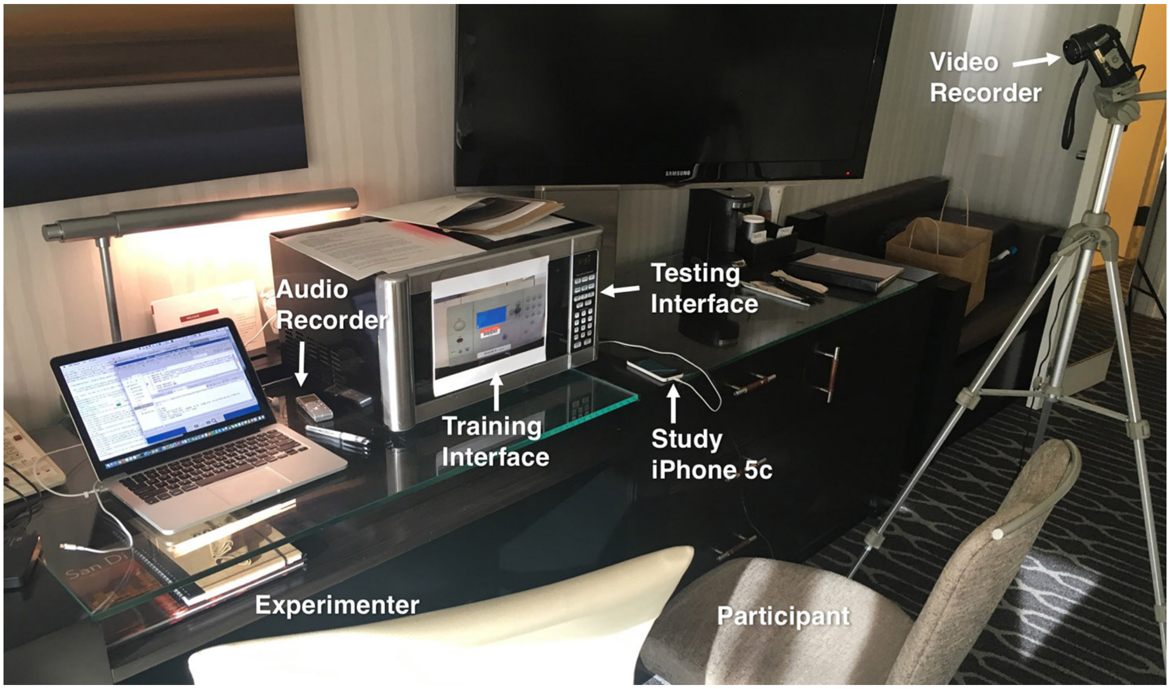
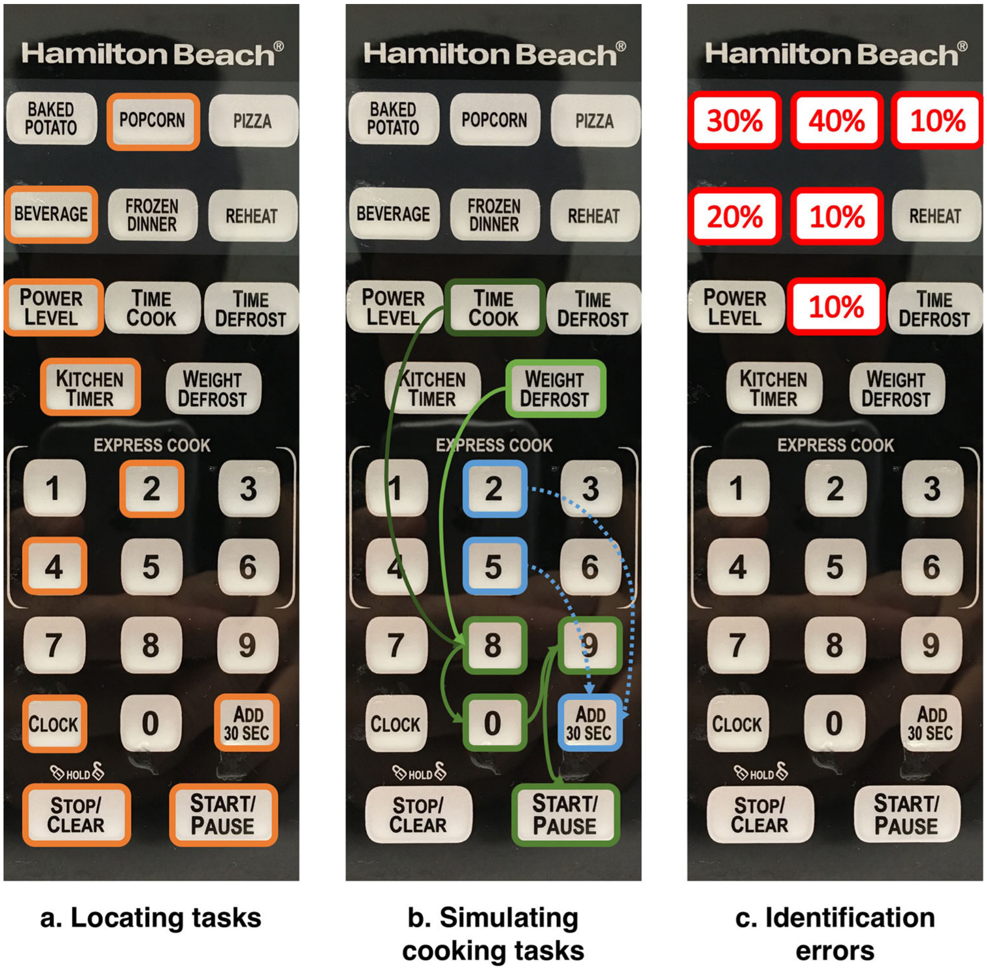


Figure 3.6: User study setup. A printer's interface is printed out on paper and used for training. The microwave interface was used for controlled testing, followed by more exploratory use of other interfaces nearby (e.g., remote control, thermostat, vending machine). The study was conducted in a hotel room and was video and audio recorded.

3.5.3 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 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 3.7a, 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 3.7b. 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.



a. Locating tasks

b. Simulating cooking tasks

c. Identification errors

Figure 3.7: 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.

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.

3.5.4 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.

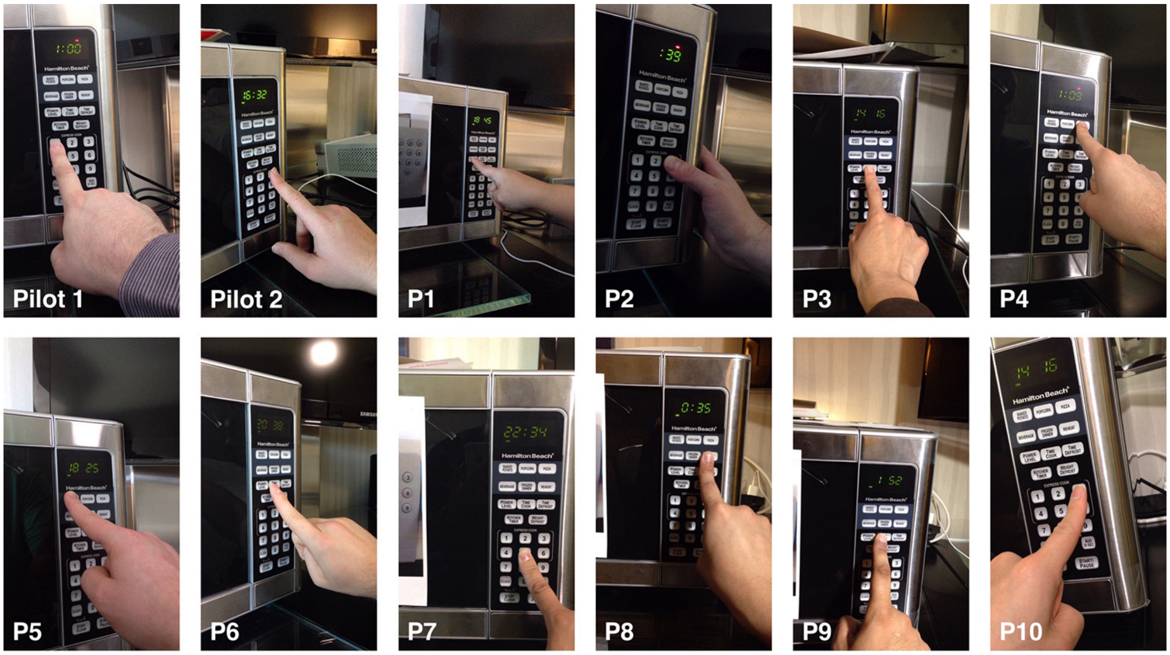


Figure 3.8: VizLens works robustly across various skin colors and lighting conditions. These are images from participants that were processed by computer vision and successfully identified the finger locations.

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 3.7c). 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 3.8. 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.

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

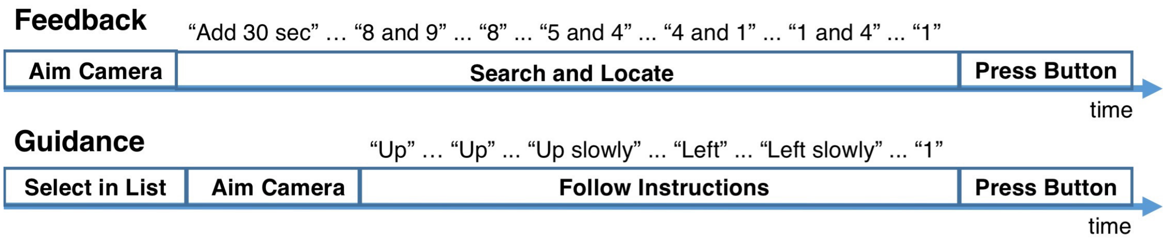

Feedback
"Add 30 sec" … "8 and 9" ... "8" ... "5 and 4" ... "4 and 1" ... "1 and 4" ... "1"
Aim Camera
Search and Locate
Press Button
time
Guidance
"Up" … "Up" ... "Up slowly" ... "Left" ... "Left slowly" ... "1"
Select in List
Aim Camera
Follow Instructions
Press Button
time

Figure 3.9: 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.

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 3.9 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).

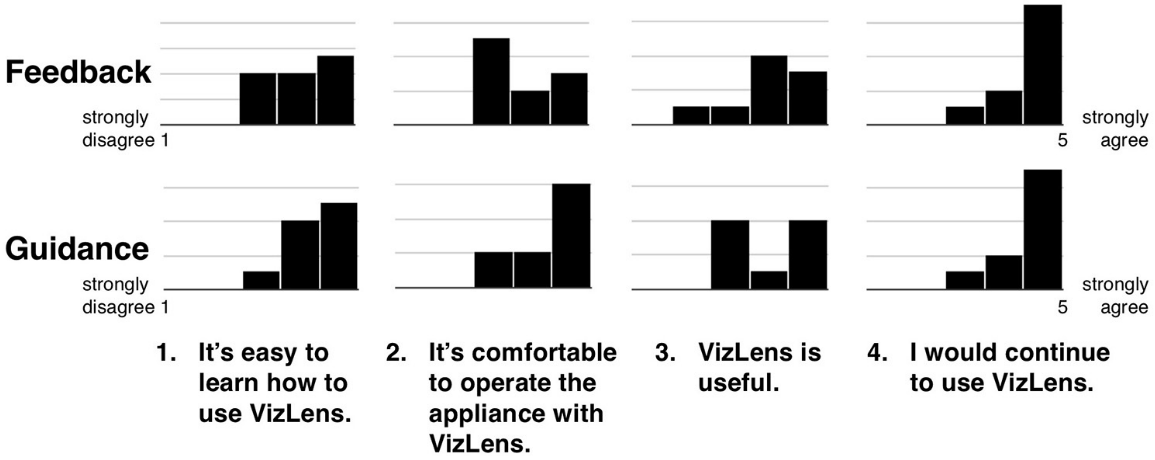


Figure 3.10: Answers to Likert scale questions indicating that participants found VizLens useful (1, 2, 3) and wanted to continue using it (4).

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 3.10). Several participants expressed their frustration with aiming and keeping good framing of the camera. Several participants tried to get a general idea of the button layout from the linear button list (Figure 3.2c) 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 [185]. 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.

3.6 Technical Evaluation

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

3.6.1 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 3.3a). For image quality, an agreement of three workers was required. If the image was determined to be complete and clear, $2 \times (NumberofButtons)$ of HITs were created for the labeling step (Figure 3.3b) 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.

3.6.2 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 3.11) 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 [66]. 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.

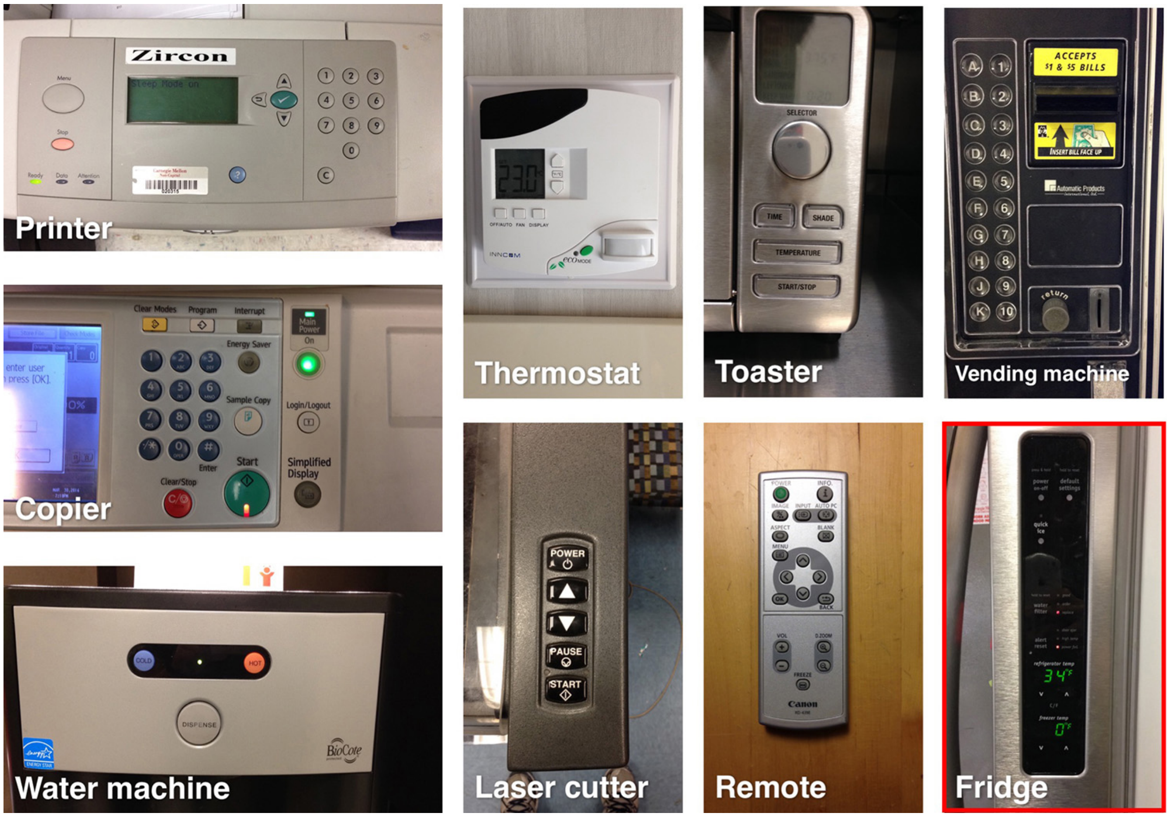


Figure 3.11: VizLens works robustly with a wide range of interfaces, including microwaves, printers, copiers, water machines, thermostats, laser cutters, toasters, remote controls, vending machines, etc.

3.7 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.

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 3.2e), similar to RegionSpeak [185]. 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

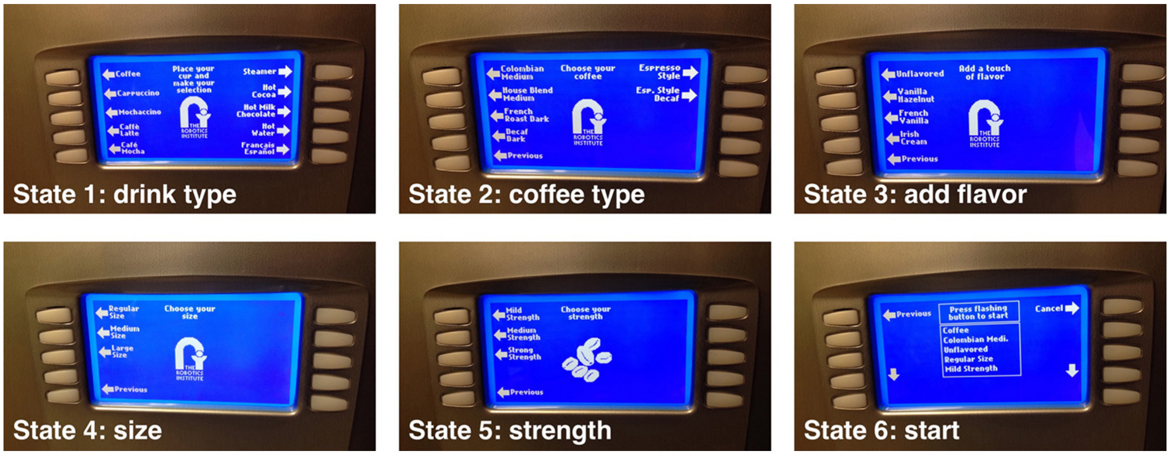


Figure 3.12: VizLens::State Detection detects screen state and adapts to it. In this example, VizLens figures out which of six states this fancy coffee machine is in, and provides feedback or guidance specific to that screen.

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.

3.7.1 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 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 video, we show VizLens helping a user navigate the six screens of a coffeemaker with a dynamic screen (Figure 3.12).

3.7.2 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

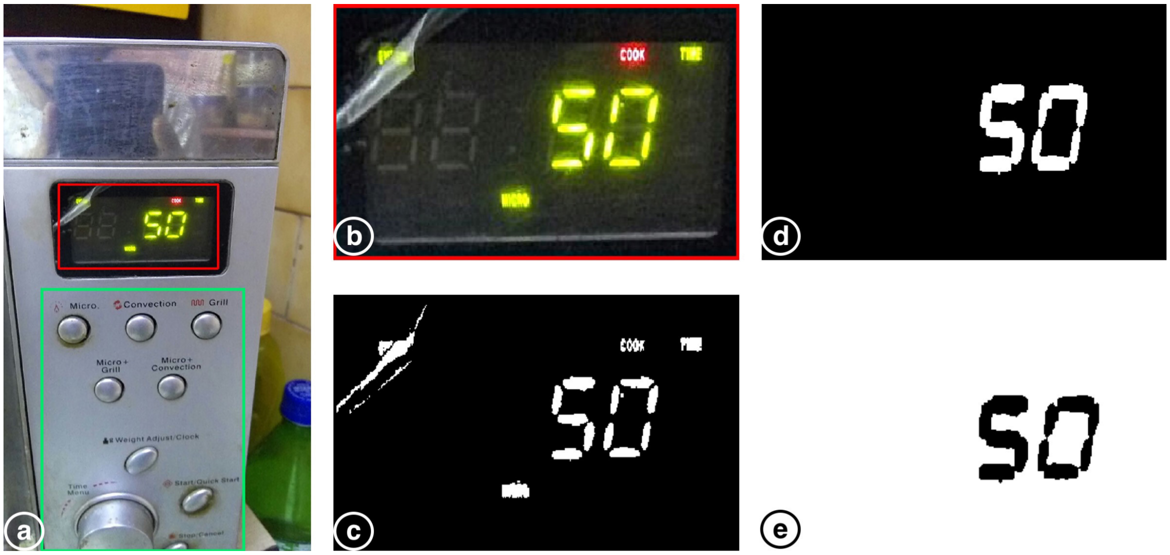


Figure 3.13: VizLens::LCD Display Reader applies OCR to recognize digits on the portion of the interface that is an LCD screen. *(a)* Separated dynamic and static regions. *(b)* Image sharpening using unsharp masking. *(c)* Intensity-based thresholding. *(d)* Morphological filtering and small blob elimination. *(e)* Selective color inversion.

(Figure 3.13a). 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 [148] for enhanced image quality (Figure 3.13b) and intensity-based thresholding to filter out the bright text (Figure 3.13c). 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 $height > 2 \times width$ to prevent adjacent characters from merging while forming single characters. Next, we applied small blob elimination to filter out noise (Figure 3.13d), and selective color invertion to create black text on a white background, which OCR performs better on (Figure 3.13e). Then, we performed OCR on the output image using the Tesseract Open Source OCR Engine [164]. When OCR fails to get an output, our system dynamically adjusts the threshold for intensity thresholding for several iterations.

3.7.3 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

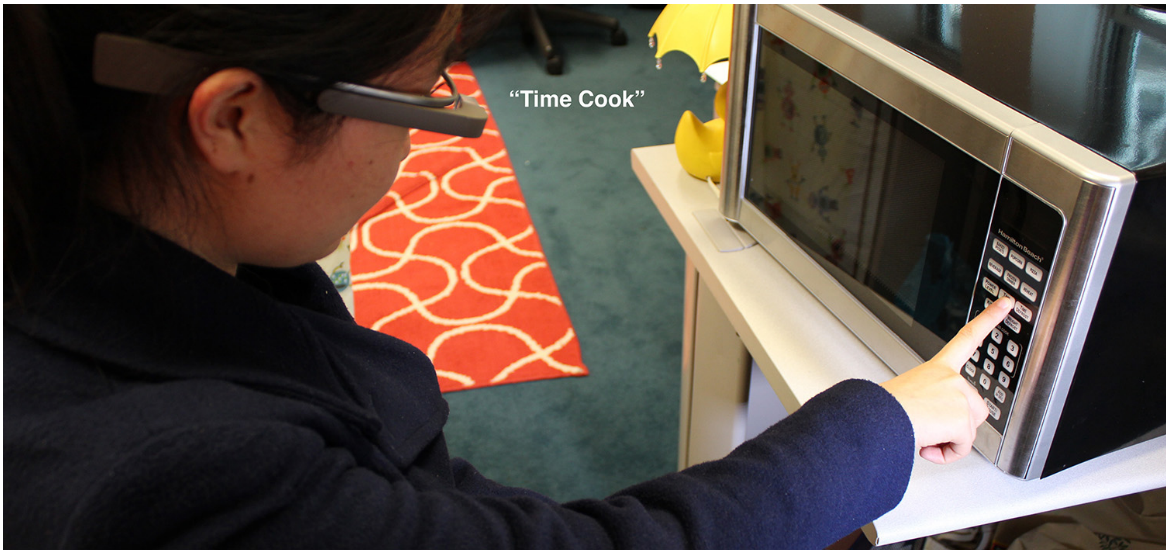


Figure 3.14: We migrated VizLens to run on Google Glass, which has the advantage of leaving the user's hand free, easier to keep image framing stable, and naturally indicating the field of interest.

(Figure 3.14), 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.

3.8 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 work 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 [113].

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 [185], 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 button's label in feedback mode will increase). The system can adapt to some of this, for example, in Figure 3.5b, the second rule on the left column shows that this labeling deviation can be fixed by the lookup rules. Furthermore, once the blind user's finger is on the button, s/he 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, s/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.

3.9 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.

VizLens is a human-AI system to enable blind people to access physical interfaces similar to using a screen reader. In this case, the human's role is to interpret the user interface in arbitrary settings, which would have been hard for machines to do. And then, the machine's job becomes simpler, it just needs to re-identify the interface and provide real-time feedback to users. VizLens trades off the advantages of humans and computer vision to be nearly as robust as a person in interpreting the interface and nearly as quick and low-cost as a computer vision system to re-identify the interface and provide real-time feedback.

The idea of VizLens's scene reader interaction of using a finger to access and explore real-world information can be applied more broadly. In the next chapter (Chapter 4), I explore cursor-based interactions to support non-visual explorations by blind users, integrating VizLens's scene reader interaction as a type of finger cursor.

Furthermore, VizLens is effective for static interfaces especially on public devices that cannot be labeled. However, for appliances in blind people's homes, requiring them to always hold or wear a device to use might be cumbersome, especially when they have the option to label them. To address this challenge, in Chapter 5, I introduce a complementary approach *Facade*, which is a crowdsourced fabrication pipeline that enables blind people to independently create 3D-printed tactile overlays for inaccessible appliances.

Chapter 4

Cursor-based Interactions for Supporting Non-Visual Explorations

The human visual system processes complex scenes to focus attention on relevant items. However, blind people cannot visually skim for an area of interest. Instead, they use a combination of contextual information, knowledge of the spatial layout of their environment, and interactive scanning to find and attend to specific items. In this work, we define and compare three cursor-based interactions to help blind people attend to items in a complex visual scene: window cursor (move their phone to scan), finger cursor (point their finger to read), and touch cursor (drag their finger on the touchscreen to explore). We conducted a user study with 12 participants to evaluate the three techniques on four tasks, and found that: window cursor worked well for locating objects on large surfaces, finger cursor worked well for accessing control panels, and touch cursor worked well for helping users understand spatial layouts. A combination of multiple techniques will likely be best for supporting a variety of everyday tasks for blind users.

4.1 Introduction

The development and prevalence of computer vision has brought tremendous changes to blind people's lives. For example, current computer vision systems can collect images taken by blind users as input, then analyze the images to produce an audio stream of information extracted (e.g., Seeing AI, OrCam, etc.) However, visual scenes often contain large amounts of information. While an audio overview such as a scene description can be helpful as a summary, humans often need detailed information about specific parts of the visual scene. For blind people, focusing is not straightforward. Because they cannot see the image, they do not know what is contained within the image, and cannot simply visually skim and point to an area of interest. Instead, blind people use a combination of contextual information, knowledge of the spatial layout of their environment, as well as interactive scanning to find and attend to specific items [157]. For example, blind people apply this strategy for locating an object on the table, reading and accessing buttons on an appliance control panel, interpreting documents and signs, or learning the spatial layout of a scene.

Cursor-based interactions are defined by how users indicate a region of the image to attend to.

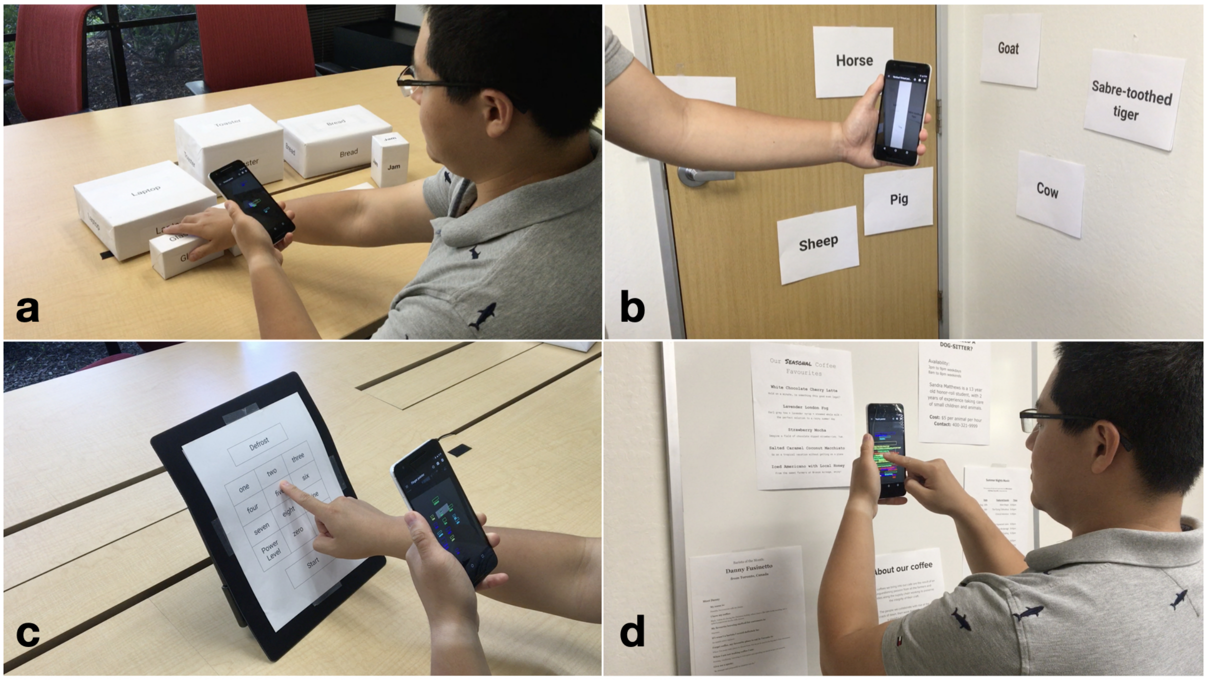


Figure 4.1: We define and compare cursor-based interactions that support non-visual attention to items within a complex visual scene: *(a)* window cursor, in which the user moves the device itself to scan the scene and receives information about what is in the center of the image; *(b)* vertical window cursor, a variation of window cursor that sacrifices the granularity on the vertical axis, but potentially facilitates locating the direction of a specific object; *(c)* finger cursor, in which the user moves their finger on the real world object they want to access and receives information about details near (or under) their fingertip; and *(d)* touch cursor, in which the visual scene is brought onto the device screen and the user moves their finger on the touchscreen to receive information about what they touch on the live camera image.

This can be done in many ways, e.g., by the current camera frame (or a region within it), by the location of a finger tip in the real world, or by a touch point on the device's touchscreen. Once a region is indicated, information and feedback relative to the cursor position are provided, e.g., by speaking out the names of items or text in the cursor region. Users can explore based on the feedback. The cursor affects how easily they can query for certain types of information and within which types of visual scenes.

To explore this concept, we implemented three cursor-based interactions to help blind users attend to items within a complex visual scene (Figure 4.1), including: *(i)* window cursor, in which the user moves the device to scan the scene and receives information at the center of the camera, similar to VizWiz::LocateIt [27]; *(ii)* finger cursor, in which the user moves their finger on the real world object they want to access and receives information near their fingertip, similar to VizLens [76]; *(iii)* and touch cursor, in which the user moves their finger on the touchscreen and receives information of the relative location on the live camera image, similar to RegionSpeak [185].

Prior work has explored the concepts of the three cursor-based interactions individually, and suggested that different cursors might be more or less appropriate for certain tasks. In this work, we contribute a study with 12 visually impaired participants where we first implemented the cursor-based interaction techniques, and formally compared the three techniques across a series of tasks that are representative of blind people's daily routines, including *(i)* locating an object in the environment, *(ii)* interpreting documents and signs, *(iii)* manipulating an appliance interface, and *(iv)* learning about their surroundings.

Our study results revealed that:

- Window cursor works well for locating objects on larger surfaces, but does not work well for small and fine-grained tasks. Blind users generally liked this one the most, as it was the simplest to use, and required the least amount of coordination.
- Finger cursor works well for accessing appliance control panels, but does not work well in pointing at remote objects in 3D space. Most users needed good instructions for this technique.
- Touch cursor works well for understanding the layout of a scene or document, but does not work well when mapping 2D screen locations is required to take actions on real-world objects (grabbing an object, pushing a button).
- A combination of multiple techniques will likely be best for supporting a variety of everyday tasks that blind users encounter.

The primary contributions of this work are: *(i)* empirical results from a user study that expose the pros and cons of each technique on a variety of tasks based on real-world use cases, and *(ii)* design implications to apply and combine these techniques to support non-visual exploration. The study contributes understanding on how to best support blind people to extract visual information from the real world.

4.2 Related Work

Our work is related to prior work on making visual information accessible with computer vision. The three cursor-based interactions that we define and study in this work have been incorporated in various ways in prior research studies and products, although their use and trade-offs in different contexts have not been previously studied.

4.2.1 Computer Vision for Visual Access

The development and prevalence of computer vision has brought tremendous changes to blind people's lives. For example, current computer vision algorithms can collect images that blind users take as input, analyze the images, and produce an audio stream of extracted information as output.

Many systems have been developed to help blind people read visual text via OCR [136]. For instance, the KNFB Reader [109] is a popular application for iOS that helps users frame text in the camera's view, and then reads text that is captured. Other systems have been built to help blind people recognize faces [133, 145], identify products [126, 133, 145], count money notes [125, 145], or read the LCD panels on appliances [65, 137, 163].

Recently, deep learning approaches have been applied to general object recognition and scene description, in products such as Aipoly [168] and Microsoft's "Seeing AI" [133]. For example, Seeing AI [133] provides functionalities for blind users to take a picture and get an overview description of the captured scene.

While an overview such as a scene description can be helpful as a summary, humans often need focused information about one or more parts of the visual scene, which often contains large amounts of information. Generally, prior approaches have assumed that there would be a primary target in the camera's field of view. However, the interaction to attend to specific targets has not been made explicit. In response to this, prior work has explored various ways in assisting blind users attend to specific items within a complex visual scene.

4.2.2 Window Cursor Applications

A natural way to capture the information users want in a photograph is to move the camera around until the intended part of the photograph is contained within the frame. 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 [99, 128, 171, 176, 184].

The challenge for these systems is both ensuring that some frame captured by the camera actually contains the object of interest, and developing an approach to alerting the user or automatically capturing the image when that occurs. For example, EasySnap [99, 176] reads out locations of faces and pre-registered objects in the field of view, and guides blind users to move their phone to take a better picture. VizWiz::LocateIt [27] 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. In [171], an image composition model is used to provide aiming feedback, and the system automatically saves the best image captured. Scan Search [184] automatically extracts key frames from a continuous camera video stream, and identifies the most significant object inside the picture. This way, blind users can scan for objects of interest and hear potential results in real time.

Our window cursor interaction is similar to these techniques in that visually impaired users hold and move their phone to scan the environment, then get real-time feedback about the objects and their locations in the field of view.

4.2.3 Finger Cursor Applications

Various projects have experimented with having visually impaired users use their fingers to access real world objects and information. In this interaction, the user's finger provides a direct connection to the item in the physical space.

Several projects use finger-worn cameras to read text and explore surroundings with computer vision. Fingerreader [156] assists blind users with reading printed text on the go with a finger-worn device. EyeRing similarly leverages a finger-worn camera to interpret immediate surroundings [139].

Other projects use cameras placed in the environment. Access Lens reads physical documents and lets a blind person listen to and interact with them [103]. In the direct touch interaction mode, Access Lens tracks the user's fingertip and speaks the text closest to it, which enables blind users to read previously inaccessible documents simply by touching them. Talkit [155] enables blind users to access 3D printed models with their finger and get audio cues about what is underneath their finger.

Using hand-held and head-mounted cameras, VizLens [76] fuses crowdsourcing and computer vision to interactively help blind people use inaccessible interfaces in the real world. Similar to a screen reader, VizLens provides feedback on what is beneath a user's finger. OrCam is a product that uses a head-mounted camera to make available various computer vision applications targeting low vision people [145]. Specifically, blind users point their finger at or on an object they want to recognize, then pull it away for a picture to be snapped.

The reason these projects used finger-based interactions is likely because the user's finger provides a direct connection to the item being explored and accessed in the physical space, and after locating the specific target, objects can be directly manipulated. Our finger cursor interaction is similar to these techniques, in that visually impaired users move their finger on the real world object they want to access, and get feedback about what is near their fingertip.

4.2.4 Touch Cursor Applications

Prior research has explored having visually impaired people use touchscreens to access mobile devices. In this interaction, the user drags a finger along the touchscreen or performs accessible gestures to navigate through information, or learn the spatial layout of documents and scenes.

Slide Rule developed multi-touch gestures that could control touchscreens non-visually [102], which informed the VoiceOver screen reader on iOS, and the TalkBack screen reader on Android. RegionSpeak [185] enables spatial exploration of the layout of objects in a photograph using a touchscreen. Users send a photo (or multiple stitched photos) to have the crowd label all of the objects in the photo. Users can then explore the photo on a touchscreen to learn about the spatial layout.

The reason these projects used touchscreen-based interactions is likely because the information can be easily represented digitally on the mobile device, fine-grained touch movements or accessible gestures can be used, and physically touching the object is not necessary or possible (due to proximity) for accessing the information. Our touch cursor interaction is similar to these techniques, in that visually impaired users drag one finger around the touchscreen to explore the content captured by the camera and mapped to the touchscreen dimensions.

To summarize, we implemented three cursor-based interaction techniques inspired by prior work for visually impaired users to get filtered and focused feedback from raw computer vision output. Different from prior work that focused on individual techniques to solve specific usage scenarios, we performed a thorough study to compare these techniques and understand their effectiveness for a variety of real-world tasks.

4.3 Cursor-based Interactions

The cursor-based interaction concept involves: *(i)* indication of a cursor region, *(ii)* more focused information and feedback relative to the cursor, and *(iii)* further user exploration based on the

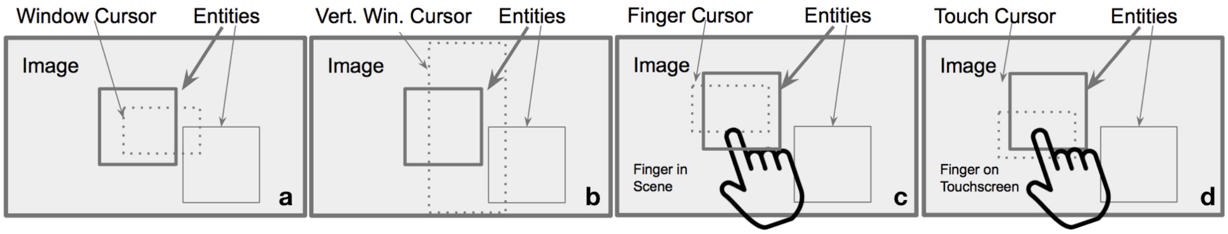


Figure 4.2: Illustrations of cursor-based interactions: *(a)* window cursor, *(b)* vertical window cursor, *(c)* finger cursor, and *(d)* touch cursor.

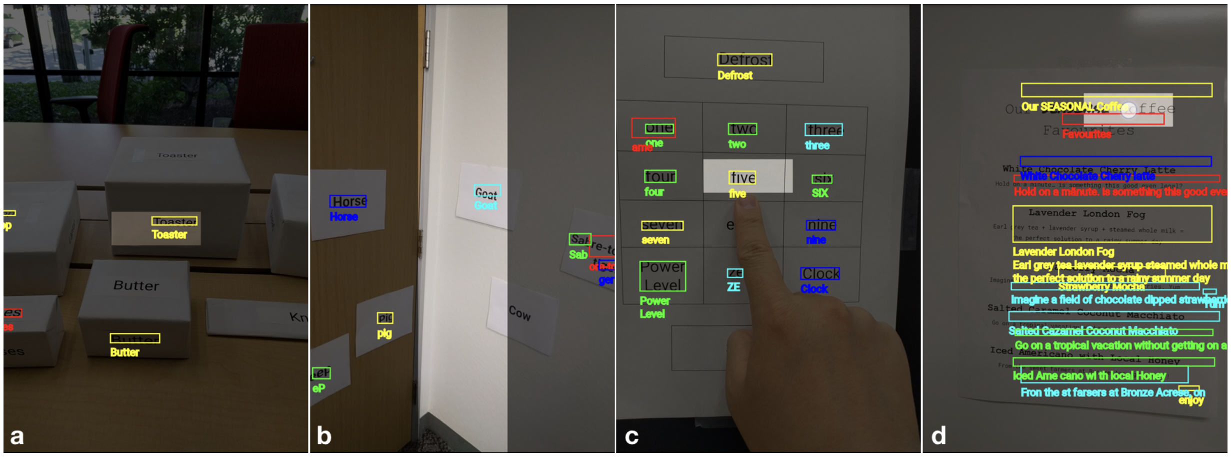


Figure 4.3: Screenshots of cursor-based interactions in different use cases: *(a)* window cursor, *(b)* vertical window cursor, *(c)* finger cursor, and *(d)* touch cursor. Colored bounding boxes show the recognized OCR results, the cursor region is drawn as a transparent overlay, while the rest of the image is covered with a semi-transparent dark overlay.

feedback. We introduce three cursor-based interaction modes for different usage scenarios, created based on prior work. More specifically, a cursor is defined by three features: *(i)* cursor center location, *(ii)* cursor shape, which defines the region of the cursor, also drawn on the camera view, and *(iii)* cursor affinity function, which defines the relationship between the cursor/entity bounding boxes produced by computer vision models, and the conditions for entities to be read out.

4.3.1 Mobile Application

The mobile application is implemented in Java for the Android platform. The backend of the app runs a series of computer vision recognizers including object identification, face detection, landmark detection, food identification, optical character recognition (OCR), etc. These recognizers output entity bounding boxes with their labels, which are then read out sequentially through an audio stream. Since the speed of individual recognizers is quite slow (e.g., OCR is ~1fps), optical trackers are used to maintain the rough locations of the bounding boxes between two processing frames.

We also included two earcons [30], which are brief and distinctive sound cues that provide additional context around the cursor region. When there are entities in the camera's field of view but are not worth being read out through Text-To-Speech (TTS), the "entity-in-view" earcon is played to indicate that the camera is generally aimed in the right direction. When the user's finger is in the field of view, the "finger-in-view" earcon is played to indicate that the camera and their finger are generally aimed in the right direction.

4.3.2 Window Cursor Mode

Window cursor mode announces entities in the center of the image. This potentially helps users find objects by moving their phone to scan the scene.

For this mode, the cursor center location is the center of the camera image, and the shape of the cursor is a small rectangle proportional to the camera image size. Entities are read out if the center of the entity is within the cursor bounding box. As illustrated in Figure 4.2a, the left entity is read out, while the right one is not. Note that the right entity also overlaps with the cursor bounding box — if there is one entity bounding box that is much larger than the others and overlaps the others, then it is undesirable to always say the large one; while the center of focus should be the smaller ones. For example, if there are multiple objects (laptop, mouse, water bottle) on a table, when the user scans their device over the table, always speaking "table" may not be relevant to the task of locating the mouse. On the camera view (Figure 4.3), the cursor region is drawn as transparent, while the rest of the image is covered with a semi-transparent dark overlay.

Another variation of the window cursor shape is a vertical slit box in the middle of the screen (Figure 4.2b). This change sacrifices the granularity of entities on the vertical axis, but potentially makes it easier for the user to scan the scene in one direction to quickly locate the direction of a specific object the user wishes to find.

4.3.3 Finger Cursor Mode

Finger cursor mode announces entities near the user's fingertip in the scene. This potentially helps users identify objects, read documents, and use appliance control panels. This interaction is similar to OrCam's MyEye [145], and Access Lens [103].

For this mode, the center location of the finger cursor is above the user's topmost fingertip location in the camera image, and the shape of the cursor is a small rectangle proportional to the camera image size. The reason for the cursor location being above the fingertip is, when the user's finger is covering an object or a piece of text, our system will not be able to read it. For use cases such as appliance access, it might be more natural to provide direct feedback of what is underneath the finger, similar to how a screen reader works. Techniques such as keeping a memory of the entities in the scene and using reference images for computing homography would help to solve this problem (e.g., VizLens [76]).

Entities are read out if the center of the entity is within the cursor bounding box. As illustrated in Figure 4.2c, the left entity is read out, while the right one is not.

4.3.4 Touch Cursor Mode

Touch cursor mode announces entities at the user's touch point on the screen. This potentially helps users explore and understand the spatial layout of a scene or document. The interaction is similar to the Explore by Touch in VoiceOver, TalkBack and RegionSpeak [185].

For this mode, the center location of the touch cursor is the user's touch point location on the touchscreen mapped to the camera image coordinate system, and the shape of the cursor is a small rectangle proportional to the camera image size. Entities are read out if the center of the entity is within the cursor bounding box. As illustrated in Figure 4.2d, the left entity is read out, while the right one is not.

4.4 User Study

The goal of the user study was to better understand how the cursor-based interactions perform for various use cases. The user study sought to answer the following research questions:

- What are the strengths and limitations of each cursor?
- Which cursor is the most or least appropriate for each task?
- How well does the user build a mental model of the 3D visual scene from the auditory feedback?

4.4.1 Participants and Apparatus

We recruited 12 participants (6 male, 6 female) through online postings. Among the 12, 9 of them were blind and 3 were low vision users; 5 were in the age range of 25-34 and 7 were in the age range of 35-54; 11 had at least a bachelor's degree, and the other one had a professional diploma; 6 were currently working at a tech company, 4 were working at schools or banks, and 2 were not currently employed; 11 had used screen reader before; and 11 had been either blind or low vision for 18+ years, while the other one had been blind for 3-6 years.

We implemented the three cursor-based interaction modes and installed the application on an Android device. To control for recognizer performance in the study, we only included tasks that involved text labels, and only used an OCR recognizer to provide feedback to the user. We also kept the cursor regions of the techniques the same size for direct comparison. The users were provided with a plastic pouch to carry the device around their neck. However, they would decide whether or not to use it. We compared cursor methods within the same overall system, controlling for many variables, including cursor size, affinity function, and recognizer performance. Though they may not be optimal, our implementations were sufficient to generate useful in-depth insights about user experience.

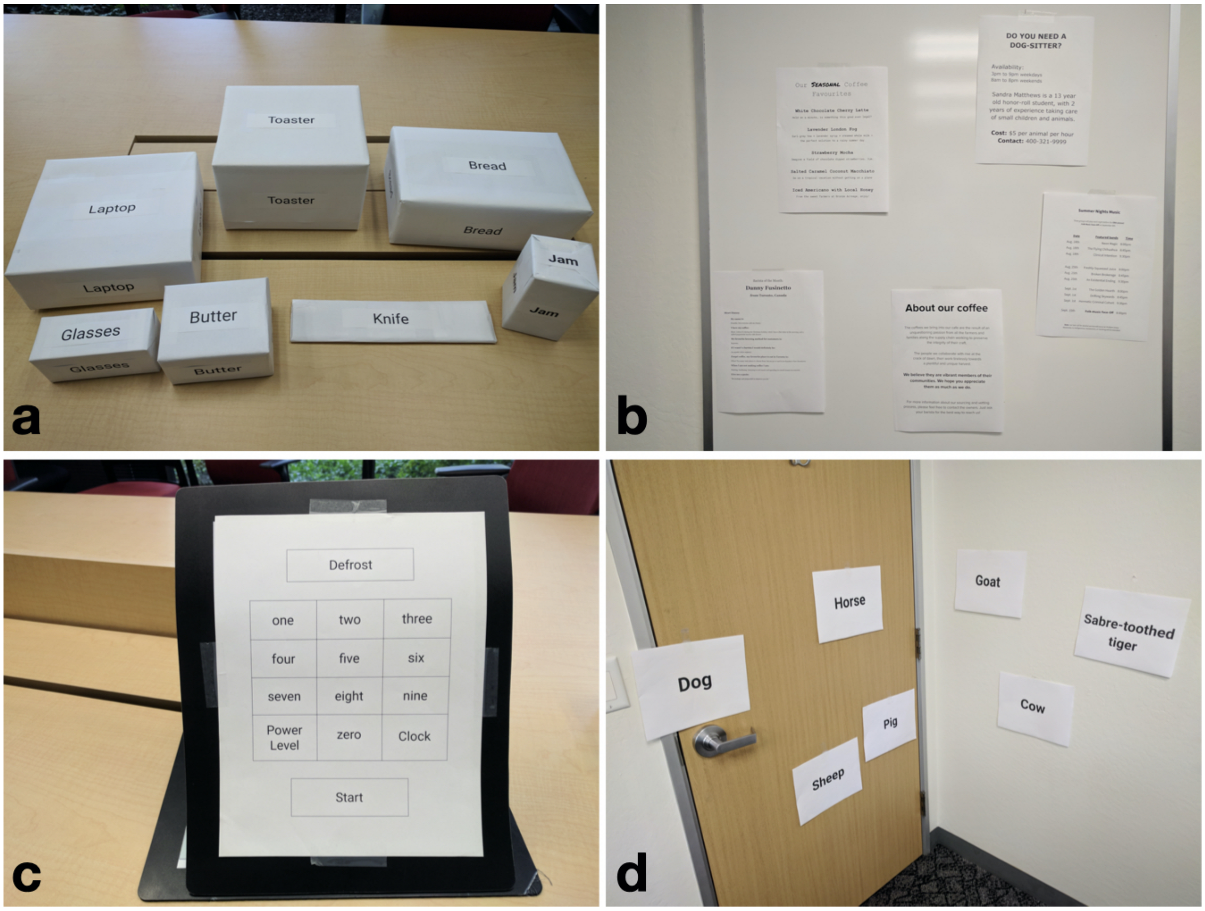
4.4.2 Procedure

The user study contained three stages. In the first stage, we conducted discovery interviews to better understand each participant's background and needs. In the second stage, we asked each user to complete four tasks with the three cursor modes to observe the usability and pain points of each method. Following each task, we asked users about their experience completing the task and asked them to rate "Of the 3 methods for this task, which one did you most prefer? Least prefer?" In the third stage, after users had completed all tasks, we conducted semi-structured interviews to ask them about their overall experience. The interview sessions were audio-recorded and the task sessions were video-recorded.

4.4.3 Tasks

The task session took 35-45 minutes. We designed four tasks representative of daily activities people engage in using eyesight (Figure 4.4). Tasks were designed based on prior work [26, 76, 103, 128, 185], and in consultation with blind participants through pilot interviews, where we asked about daily tasks that were difficult to complete without sighted help, information they felt they were missing out on, as well as situations that made them feel curious about the environment. The scenarios were also confirmed by participants in study stage 1.

We then carefully designed the tasks to be authentic and involving edge cases. In the first task, participants were asked to locate a specific object on the table, such as glasses. We also intentionally included food and knife on a kitchen table so that participants would not use their



a
Toaster
Bread
Laptop
Jam
Butter
Glasses
Knife
b
Our SEASONAL Coffee Favourites
White Chocolate Cherry Latte
Lavender London Fog
Strawberry Mocha
Salted Caramel Coconut Macchiato
Iced Americano with Local Honey
DO YOU NEED A DOG-SITTER?
Availability:
3pm to 9pm weekdays
8am to 8pm weekends
Sandra Matthews is a 13 year old honor-roll student, with 2 years of experience taking care of small children and animals.
Cost: $5 per animal per hour
Contact: 400-321-9999
Summer Nights Music
Danny Fusinetto
About our coffee
We believe they are vibrant members of their communities. We hope you appreciate them as much as we do.
c
Defrost
one
two
three
four
five
six
seven
eight
nine
Power Level
zero
Clock
Start
d
Horse
Goat
Sabre-toothed tiger
Dog
Pig
Cow
Sheep

Figure 4.4: Study setup comparing the cursor techniques across a series of tasks representative of blind people's daily routines, including *(a)* locating an object in the environment, *(b)* interpreting documents and signs, *(c)* manipulating an appliance interface, and *(d)* learning about their surroundings.

hands to directly touch and explore as they normally do. In the second task, participants were asked to interpret documents and signs, e.g., find the time and date of an event on a printed poster. For the third task, participants were asked to manipulate an interface, e.g., press a button sequence on a flat and unlabeled appliance control panel. For the last task, participants were asked to learn about their surroundings, e.g., explore an unfamiliar environment and identify what and where are the objects around them. These tasks generally require sighted assistance.

When blind people first come to an unfamiliar space, they need to learn about the surroundings. To effectively operate in a space, blind people need to first locate objects, and then interpret the objects or interact with them. Among the four tasks, environment exploration and object location requires 3D understanding and navigation, interface manipulation requires 2D understanding and navigation, and document interpretation requires the least spatial navigation.

We asked participants to complete the tasks with each of the three cursor methods (window cursor, finger cursor, and touch cursor). The sequence of the three cursor methods were counter-balanced across four tasks. All users completed the tasks in the same order. The task instructions are listed below.

Task 1: You are at a family gathering, and your aunt calls in after she left because she thinks that she forgot her glasses on the kitchen table. Could you find them for her?

Task 2: You've just stepped into a cafe for an iced drink, and the barista mentions they will be hosting live music this month to the person ahead of you. The schedule is on the bulletin board. Can you find and read the schedule?

Task 3: You've purchased a variety pack of fancy popcorn and each flavor needs to be warmed in the microwave for a very precise amount of time. The first requires 2 minutes and 49 seconds of cooking. Can you enter 2-4-9 on the microwave panel, then click the start button? Repeat for [3:17 + start], and again for [5:08 + start].

Task 4: A friend of yours is babysitting her niece for the day, and she invited you to join them at the petting zoo. You are standing in the centre of the park, and you hear her niece giggling. What animals are in the zoo? Where are they?

4.4.4 Methods

We took a qualitative approach when analyzing participants' responses in stage 2 and 3 of the study. Quantitative measures were used in prior work to evaluate individual cursors. We instead complement prior work with qualitative approaches, which is vital for gathering in-depth insights into user experience necessary for generating meaningful design implications.

We transcribed the video and audio recordings. We used a line-by-line coding approach [31] and synthesized the main concepts from the task sessions and subsequent interviews. We also selected user feedback quotes which are indicative of issues in the system and could inform future designs. Furthermore, we considered the diversity of participants (i.e., not all quotes should come from P2), the diversity of problems addressed, and the clarity of meaning when selecting these quotes.

4.5 Results

We now detail the user study results. We first present findings regarding the 9 blind users. For each task, we discuss user feedback and report preference rating responses for each cursor method. We then present results regarding the 3 low vision users. Finally, we summarize the key takeaways.

4.5.1 Task: Locate an Object

In this task, we aim to answer the question, "Can users leverage window cursor, touch cursor, or finger cursor to locate an object?" The results suggest that window cursor is the best among the three, with 7 most prefer, 1 neutral, and 1 least prefer in response to the preference rating.

Window Cursor

Users enjoyed that window cursor only required one hand, and rated it as the most comfortable. However, we noticed that users generally had a poor sense of angular alignment, making the task of inferring real-world position difficult. For example, P2 found the window to be small; P4 and P7 found it hard to aim the camera at a correct angle.

> *Why is it not telling me anything when it sees text? (Because the window is not over the text) Oh... could they make that window bigger?* (P2)
>
> *It's like you have really narrow vision... You have to scan systematically left and then right.* (P7)
>
> *It's hard to tell if I'm actually tilting it or not... So technically speaking, holding it flat is tough.* (P4)

Finger Cursor

Finger cursor was not found to be successful for this task. Users thought the involvement of their finger was unnecessary.

> *That's a foolish way of doing things, I'm sorry to say this. It's just redundant. Have you ever seen a blind person pointing their finger at something?* (P11)

Many blind users had a tough time aligning both the objects and their finger in the camera's field-of-view. The "finger-in-view" earcon was generally not trusted, since the false positives were frustrating for many participants.

> *It's tougher because getting my finger in the camera view is what I'm finding to be hard... I would have to move both simultaneously.* (P4)
>
> *I think it's also very tiring... I would have to be super motivated to find out whether that was jam or not.* (P5)

Touch Cursor

Touch cursor was found to be the most difficult to use among the three. Only patient users with a systematic approach completed the task. A number of users wanted to freeze the live view. Even after discovering an object, users struggled to keep the device steady enough to locate it.

> *Actually, touching the screen is pretty good. It's a way to say, 'Hey, stop chattering, I'm looking for something right now.'... So in this case, if you were to freeze the image, that would be good.* (P11)
>
> *While I'm moving the phone around, the live view is changing, so I might miss the text on the screen.* (P8)

Sometimes, the user's grip would interfere with their success. For example, one participant never explored the bottom part of the screen. Participants also expressed the difficulty of looking for information on the screen when they had no notion of where the information was.

> *I think it would be much easier, if instead of me sliding my fingers around on the phone... that it would just read it out. Because I have no idea where the text is.* (P2)
>
> *It would be nice to get a little more feedback for hot and cold... or if the phone would vibrate the closer I get to something... because right now, it's just a tiny little screen in the dark!* (P3)

4.5.2 Task: Interpret Documents and Signs

In this task, we aim to answer the question, "Can users leverage window cursor, touch cursor, or finger cursor to interpret documents and signs?" The results suggest that touch cursor is the best among the three, with 5 most prefer, 2 neutral, and 2 least prefer.

Window Cursor

Window cursor was not found to be very effective for this task. Users mentioned that it was not clear whether the camera was capturing the whole page, or only a fraction of the page. The high density of entities recognized made the users' attempts to interpret documents ineffective.

> *You have to balance between density and truncating the text with the edge of the window, then you have to figure out how to scan it.* (P3)

Moving along a row to gather table data was nearly impossible for most users, and they frequently had to guess the date/time based on ordering of audio output.

> *If it somehow can build a summary of what it thinks is important, that's good, but otherwise it has to read the whole thing.* (P11)

Users were generally better at holding the device upright rather than perfectly flat. Users became fatigued by this task, and came up with creative ways to stabilize the device.

Finger Cursor

Users struggled not to occlude what they were attempting to read (especially if the document was not at chest height).

> *It seems to see my finger, but it's only reading small parts of stuff.* (P8)

The desire to physically touch the flyers also caused users to stand too close to the bulletin. Users saw no value in pointing at specific sections of flyers because they did not know or care about page layout. Pointing at blank spaces and the edges of documents was a common mistake. It was not obvious that aiming for the top-center of a page might read the title.

> *If it takes too long to figure out, I'll probably just ask somebody... It's about getting things done, not about proving to the rest of the world that I can do this.* (P5)

Touch cursor

Framing the task in a public setting caused users to worry about negative social judgment, and they reiterated the desire to capture a photo then walk away.

> *Is there a way to freeze the image? Then you could sit down and quietly explore... I would like to just take a screenshot and not take other people's time.* (P5)

> *It's a little different in a coffee shop - people are going to wonder what the heck she's doing all hunched over.* (P12)

Because text density was high, multiple entities would be selected and read out sequentially. Output was often garbled because of errors with OCR and text truncated by the camera's field-of-view. Shifts in users' body position compounded problems caused by latency. Such reasons add additional difficulty to reading documents and signs.

> *You have to move and hold still, it takes a lot of patience, because you move and you stop... then you move again, and you stop. It's like traffic: stop and go.* (P1)

> *I find that moving my finger on the screen is throwing the orientation off, so because I'm fighting with it so much, I'm not noticing how much I'm moving the device.* (P3)

4.5.3 Task: Read Labels and Enter Data on an Appliance

In this task, we aim to answer the question, "Can users leverage window cursor, touch cursor, or finger cursor to read labels and enter data?" The results suggest that finger cursor is the best among the three, with 6 most prefer and 2 neutral.

Window cursor

Users did not trust the mental model created only by proprioception (kinesthetic awareness).

> *Something blind people don't understand is how much or how little the camera can see, so we don't know how much we need to move the camera.* (P12)

Latency caused scanning such a small area to be very difficult, and multiple buttons would often be read at once. Holding the device closer to the panel yielded the best results, but users who were unfamiliar with the device model did not know on which side the camera was located.

> *I'm having a hard time figuring out what the camera is capturing when I move it like that [waves device around in the air].* (P4)

> *You have to find the exact place, hold still, then find the button.* (P7)

Finger Cursor

Unsurprisingly, users were the most confident and satisfied with this interaction for appliance usage.

> *I'm more sure I'm pressing the right button.* (P7)

> *It's more tactile... Doing stuff on the screen, you still have to work through the screen to get to what you want, but if you're actually touching it, and it's telling you what you're interacting with, it's immediately much more useful.* (P3)

It was the preferred method, but users still felt lost at times. Alignment and occlusion were the biggest hindrances to usability. Users did not instinctively realize that OCR could not recognize text directly beneath their finger.

> *How do I know if I'm covering the thing that I want to press?* (P11)

> *On some panels, you can't really touch the buttons... since you might activate something. So that would be dangerous. And of course, because your finger will have to be below the actual control... I guess it's just something we would have to figure out... app should give you instructions...* (P5)

Touch Cursor

Most users thought this method would be helpful in creating a mental blueprint of the panel.

> *Because I have to translate what's on the screen to what's on the board, a lot of that relational information is going to be lost..., but that would give a good overview of how the control panel is laid out.* (P1)

The active window created by touching the screen was larger than the entities themselves, so the output was sometimes interpreted as contradictory and untrustworthy. Users also found it hard to capture the entire panel using the camera, and the mapping between screen and panel was not intuitive. Users had slightly more success if they rested the edge of the device on the table to prevent unintentional shifts in the field-of-view.

> *Oh. Now I need to figure out where it is on the control panel? That is going to be totally impossible.* (P5)

> *I could find everything, but it was hard to know the relationship between the screen and the panel.* (P7)

4.5.4 Task: Learn About Surroundings

In this task, we aim to answer the question, "Can users leverage window cursor, touch cursor, or finger cursor to learn about their surroundings?" Exploring a 3D space was very slow because users needed to scan both vertically and horizontally. In addition to the original window cursor, we added a variation in which the window cursor shape is a vertical slit box in the middle of the screen (Figure 4.2b and Figure 4.3b). We also included the vertical window cursor as a fourth method in this task. Users preferred the vertical window cursor the most among the four methods. Finger cursor yielded the worst results because of latency, occlusion, and lack of confidence in the technique. Users hoped that the cursors could be used in combination with other environmental cues to provide additional information. We also observed that the "entity-in-view" earcon was less helpful for entities at a distance, and that holding the phone in one place was very hard.

> *The beep-beep-beep thing is not useful... It's telling me that there is text when I can't find the text. It seems to find text everywhere!* (P11)

> *Too slow and not enough feedback as to hot or cold, getting closer or away from whatever information that is causing the screen to beep... so I now have to hunt for that, and then once I find it, I have to interpret the position of the screen with what's in front of me, so... yea, I didn't like that interaction at all.* (P3)

4.5.5 Feedback From Low Vision Users

We observed high completion rates for low vision users. All users were able to complete the tasks using at least one method. Window cursor was the least preferred, touch cursor the most, and finger cursor had the highest potential. Reading text was inherently the most difficult task for low vision users. Both finger cursor and touch cursor were perceived as highly promising for reading documents and signs as well as for confirming control panel layout.

Latency became a more obvious issue for low vision users. Finger cursors are relatively intuitive for low vision users to pick up. In contrast to blind users, the feedback provided by the finger cursor is used for confirmation rather than information. Low vision users did not encounter issues related to aligning the camera, avoiding occlusion, or correctly targeting the field-of-view.

Feedback on the Visual Interface

Low vision users thought the overlay was too dark, or completely unnecessary. Users also wanted to change the size, shape, or scale of the window.

> *Oh, I found one... That's purely by chance though, because when I look through this, it's so much darker that I can't see anything at all.* (P9)

On-screen text was too small to be helpful, and it made entity-dense views more difficult to understand. The contrast of some entities was not sufficient. Users found that inconsistency in the assignment of colors to entities added to the visual noise. False positives with finger pointing were found to be problematic.

> *I can barely read that... The contrast of these make a big difference, so whatever the blue one is, I can't see it at all.* (P9)

> *There are so many boxes and so many colors that are overlapping one another. When I try to touch one, my finger is too fat, and I can't get the one [I want].* (P6)

Feedback on the Auditory Output

Users reported that the "entity-in-view" earcon was unnecessary since entity boxes appeared on the screen. "Finger-in-view" earcon was further redundant since the user could see the window, but not their finger. Users wanted more instruction on occlusion and framing.

> *The sounds are clear, but they're not helpful.* (P10)

> *It would be nice if it gave you a little more guidance... any time you were doing something that's not optimal.* (P9)

Feedback on the Physical Interaction

Window cursor was the least preferred method.

> *It's less efficient because there are multiple rectangles on the screen, and now I have to get the white rectangle over one of the colored boxes. It seems like it's redundant.* (P6)
>
> *It might be nice to zoom in a bit, especially if the words were smaller.* (P10)
>
> *I would like to be able to choose the whole paper, and not just one area.* (P9)

All of the low vision users thought pinching the screen to zoom might improve usability. Similar to blind users, low vision users also expected the number read out to be the number beneath their finger. False positives and latency while pointing their fingers were frustrating for users.

For touch cursor, users needed to be instructed to touch and hold, rather than just tap the boxes.

> *Until you instructed me to hold down longer than I'm used to, it was a bit confusing. I thought, just tap and let go.* (P6)

Because entities move on the screen as the camera moves, selection of the desired entity with touch cursor was difficult.

> *I can see the dates, but because these keep jumping around so much, I can't actually get them to be read out. For a person who's blind, that's totally unusable.* (P10)

4.5.6 Social Acceptability

Social acceptability emerges as a theme from the task sessions and interviews. Our participants also expressed concerns about safety when adopting new tools. Some users revealed that they will not adopt a tool that would make them appear significantly different from their peers.

> *Not to mention, you're going to look weird to other people. If a blind person randomly starts pointing their finger, people are going to think, 'Oh, what is this guy doing?'... You don't want to stand out from the crowd for making strange gestures.* (P5)
>
> *It's also a safety issue. I would be concerned if I were a blind person, and I was walking down the street, that someone would just grab [my phone] and steal it.* (P9)
>
> *People would stare...I don't know if I'm pointing at a person or what. They would be like, 'Why is this person pointing at me?'... I would be afraid of that.* (P11)

4.5.7 Key Takeaways

In the study we found that different cursor methods are more effective for different tasks. Window cursor works well for locating objects on larger surfaces, but does not work well for small and fine-grained tasks. Blind users generally liked this one the most, because it was the simplest to use, and required the least amount of coordination. Finger cursor works well for accessing appliance control panels, but does not work well for pointing at remote objects in the 3D space. Most blind users needed good instructions for this method, though finger cursor is the most intuitive for low vision users. Touch cursor works well for understanding the layout of a scene or document, but does not work well when mapping the on-screen locations is required to take actions on real-world objects (e.g., grabbing an object, pushing a button). We also summarize the key concepts emerged from the task sessions and subsequent interviews:

- Familiarity with concepts of visual perception correlates to improved technique, greater patience, and higher success. Users who recently lost their sight find these interactions the most intuitive.
- Users believe the cursor methods are complementary and context-specific. Users believe a single task or scenario could benefit from multiple cursor methods used together.
- All cursor methods are potentially useful. But without guidance, they are too difficult to be valuable.
- Not all cursor methods should be triggered or detected automatically. Users enjoy a sense of control over their technology, and view cursors as an actively-triggered tool.
- The effectiveness of the interaction model depends on the form factor, and needs further validation. Most users expressed concerns for negative social judgment and personal safety while using the cursor methods.

4.6 Discussion and Future Work

The study revealed how blind people interacted with the three cursor methods. We identified system limitations, blind people's pain points and concerns. This informs future design that could better leverage and combine cursor-based interactions to support access to visual information in the real world.

4.6.1 Prerequisites for Cursor Usability

Blind participants found the system delay slowed them down and prevented them from interacting with the objects more intuitively. For blind users to interact with real world objects using cursor techniques in real time, recognition latency needs to be reduced, e.g. 0.1s according to approximate response time limits for instantaneous feedback[1].

The OCR recognizer's performance is quite poor for angled text, and often result in truncated and nonsensical output. In order for cursor interactions to work when the user is actively moving the device at different angles, robust recognition of angled and tilted text is necessary. Furthermore, other computer vision recognizers described earlier will be explored in future prototypes.

Consistent with prior work, providing additional feedback on the aiming of the camera would make the interactions more usable. One approach is to add edge detection for signs, documents, and panels, and provide feedback when the camera is not properly aligned. Alternatively, proper images can be automatically selected and used for recognition.

[1] https://www.nngroup.com/articles/response-times-3-important-limits/

4.6.2 Auditory Feedback

Users found the earcons to be confusing. To improve the audio feedback for novice users, the "entity-in-view" earcon could be replaced with verbal hints, potentially with different styles of TTS or earcons of varied frequencies to indicate the distances of entities rather than the binary choice of playing or not playing an earcon. The "finger-in-view" earcon, from users' feedback, should be replaced with "finger-not-in-view," since warning the user when their finger is not properly aligned in the camera view might be more helpful. Furthermore, earcons should generally be more pleasant and expressive.

4.6.3 Cursor Interactions

Cursors help users actively seek information based on their needs, instead of passively receiving information of the entire environment. Future designs can enable blind people to have more control over what information they can get, or what they want to learn about their surroundings. For window cursor, the current "window" box could be replaced with a pinch-to-zoom or resizable window model, or by a "window" with a larger viewfinder frame. (We kept the cursor region the same for direct comparison in our study.)

Since users reported that finger cursor is less intuitive, we could provide a tutorial, and verify user's technique before enabling the cursor mode. Skin tone calibration could also be added to the finger detection algorithm to increase accuracy. To solve the occlusion problem of fingers, techniques such as keeping a memory of the entities in the scene, and using reference images for matching would help (e.g., VizLens [76]).

For touch cursor, we found that it is not natural for the user to map the position of the entity on the screen to the location of the real world object, in order to take actions such as pushing a button or grabbing an object. However, this method does help with understanding the relative layout of different objects on the same document or scene. Therefore, it might be beneficial to apply touch cursor on still photographs or scans, and offer a touch-to-freeze or touch-to-save model. For example, after the user scans a document, they could use touch cursor to navigate the document. Furthermore, the user could take a picture or a panorama of a scene, and use touch cursor to explore the layout of different objects, similar to RegionSpeak [185].

For many tasks, users' ideal output would be an intelligent and well-formed summary, which they could probe for more details if needed, as mentioned by P1, P3, P9, P11, and P12. This points to the need of providing both target information and context-relevant information at the same time [158]. We could explore adding a search feature in the tool that enables users to look for specific information within a larger context.

4.6.4 Social Acceptability

An important theme in participant comments was social acceptability. The findings suggest that when designing tools for blind and low vision users, in addition to technical feasibility and efficiency, social acceptability is also a key factor. This was not emphasized in prior work. We see research or commercial systems fail to consider social acceptability when designing interaction methods. For example, OrCam [145] users point their finger at or on an object they want to recognize, then pull it away for a picture to be snapped, which may not be appropriate for targeting remote objects in public. In future work, we need to further investigate this adoption issue, and ensure blind and low vision users do not feel awkward, obtrusive, or self-conscious.

4.6.5 Combination of Multiple Cursor Methods

From the study, we realized that for many use cases, applying a combination of multiple modes might be more helpful for the user. For example, when trying to access an appliance control panel, they could first scan the control panel, and use touch cursor to explore and familiarize themselves with the layout of the interface, then use the finger cursor mode to access the buttons. When trying to find an object on a table, it would be helpful to first get an idea of the available objects and their relative layout with touch cursor, then use window cursor to locate a specific one. When trying to read a bulletin board, it would be helpful to first find the specific poster they are interested in with window/finger cursor, then scan it and use touch cursor to explore the document. Finally, when walking in an unfamiliar environment, it would be helpful to first know an overview of the things around them, then use vertical window cursor to guide them to the direction of a specific one. In this case, they could wear the device in a lanyard so they do not need to hold it.

If applying a combination of multiple cursor modes would be more helpful, having the users manually switch between them might be cumbersome, especially when they are wearing the device and their hands are busy holding a cane or guide dog. Therefore, automatically switching between cursor-based interaction modes could be interesting to explore. For example, as soon as they take the device out of the lanyard and hold it up with their hand, window cursor could be automatically selected. Then, if they show their hand in the field of view of the camera, finger cursor could be launched, if the user touches the screen or start dragging their finger across the screen, touch cursor could be triggered.

4.7 Conclusions

In this work, we implemented three cursor-based interactions that have been explored in prior work individually, to help blind users access targeted information from visual scenes. We conducted a thorough study to evaluate and compare the three cursors across four tasks that are representative of daily routines. The study reveals that different cursor methods are effective for different tasks. More specifically, we found that window cursor works well for locating objects on large surfaces; finger cursor works well for accessing appliance control panels; and touch cursor works well for understanding spatial layouts. A combination of multiple techniques will likely be best for supporting a variety of everyday tasks for blind users.

Chapter 5

Facade: Auto-generating Tactile Interfaces to Appliances

Common appliances have shifted toward flat interface panels, making them inaccessible to blind people. Although blind people can label appliances with Braille stickers, doing so generally requires sighted assistance to identify the original functions and apply the labels. We introduce *Facade* — a crowdsourced fabrication pipeline to help blind people independently make physical interfaces accessible by adding a 3D printed augmentation of tactile buttons overlaying the original panel. Facade users capture a photo of the appliance with a readily available fiducial marker (a dollar bill) for recovering size information. This image is sent to multiple crowd workers, who work in parallel to quickly label and describe elements of the interface. Facade then generates a 3D model for a layer of tactile and pressable buttons that fits over the original controls. Finally, a home 3D printer or commercial service fabricates the layer, which is then aligned and attached to the interface by the blind person. We demonstrate the viability of Facade in a study with 11 blind participants.

5.1 Introduction

Flat touchpads have proliferated on common appliances, making them inaccessible for blind people. The task of creating an appropriate tactile overlay to adapt to inaccessible appliances currently requires in-person sighted help and a labeling device that can print embossed labels. However, sighted assistance is not always available, and a labeling device doesn't solve issues such as layout and size of labels. Automatically generated tactile overlays could address both issues. We present an end-to-end crowdsourced fabrication pipeline that can be done independent of in-person sighted help, and costs less than $10 per appliance.

To identify the existing challenges of using inaccessible interfaces of home and work appliances, we conducted a formative study with six blind participants. We identified four design requirements for a system to augment physical interfaces for non-visual access: *(i)* the solution for tactile labeling should enable blind users to independently augment and access their appliances without in-person sighted assistance, *(ii)* the augmented labels should be customizable to address individual needs, *(iii)* the solution should allow for learning and memorization of the interface,

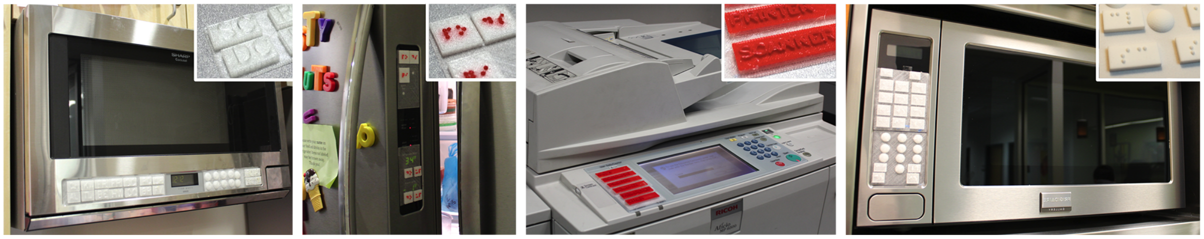


Figure 5.1: Facade is a crowdsourced fabrication pipeline that enables blind people to make flat physical interfaces accessible by independently producing a 3D-printed overlay of tactile buttons. From left to right, we demonstrate example applications including microwave, refrigerator door, copier, and another microwave. Insets shows close views of individual embossed buttons.

and *(iv)* the tactile labels should support easy attachment and reproduction for repeated use.

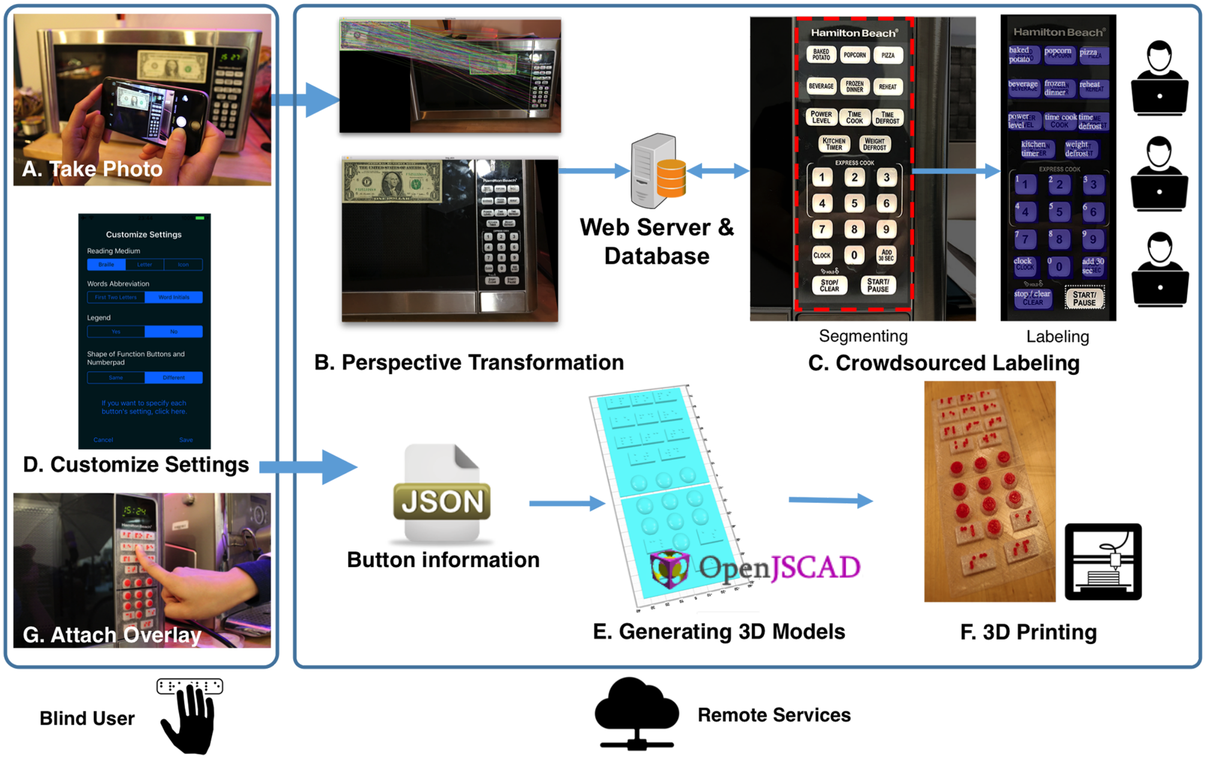
We introduce *Facade*, a crowdsourced fabrication pipeline to make physical interfaces accessible by adding a 3D printed layer of tactile buttons overlaying the original panel (Figure 5.2). When a blind person encounters an inaccessible appliance for the first time, s/he uses the Facade iOS app to capture a photo of the interface using a dollar bill as a fiducial marker for recovering size information (Figure 5.2A and B). Within a few minutes, crowd workers mark the layout of the interface, annotate its elements (e.g., buttons or other controls), and describe each element (Figure 5.2C). These labels are then used to generate 3D models of a layer of tactile and pressable buttons matching the original controls (Figure 5.2E), which the blind users can customize by changing the shape and labels of the buttons using the Facade iOS app (Figure 5.2D). Finally, an off-the-shelf 3D printer can be used to fabricate the layer (Figure 5.2F). The printed button facade is designed to be easily aligned and attached to its appliance using adhesives (Figure 5.2G). Although consumer-grade 3D printers might not be readily available to blind people at home, many printing services are available from which a print can be mail-ordered. In addition, we can expect that consumer-grade printers will continue to improve in speed and robustness. Even with mail-order costs, Facade is an inexpensive ($10 from a service such as 3D Hubs [1]) and more accessible alternative solution.

This work makes the following contributions:

- In a user study, we identify existing challenges and design requirements for augmenting physical interfaces with tactile markers.
- We introduce Facade, a crowdsourcing and fabrication pipeline to augment inaccessible physical interfaces with overlaid 3D printed tactile buttons.
- Our validation shows that Facade enables blind people to independently augment appliance interfaces, and that fabricated overlays provide rich and usable tactile feedback for accessing otherwise inaccessible appliances.

5.2 Related Work

Recent advances in consumer-grade 3D printers and the do-it-yourself (DIY) movement have changed the audience of 3D printing. It has already been established as a tool that enables



Customize Settings

Reading Medium: Braille | Letter | Icon (Braille selected)

Words Abbreviation: First Two Letters | Word Initials (Word Initials selected)

Legend: Yes | No (No selected)

Shape of Function Buttons and Numberpad: Same | Different (Different selected)

If you want to specify each button's setting, click here.

Cancel Save

Figure 5.2: Facade users capture a photo of an interface they would like to use with a fiducial marker attached to it (we use a dollar bill). Using perspective transformation, the interface image is warped to the front view and absolute measurements are calculated. Then this image is sent to multiple crowd workers, who work in parallel to quickly label and describe elements of the interface. Blind users can then customize settings of the labeling strategy, and these labels and preferences are used to generate the 3D models of a tactile layer matching the original controls. Finally, an off-the-shelf 3D printer fabricates the layer, which is then attached to the interface using adhesives.

amateurs to create a wide variety of assistive technologies [39, 41, 91, 131]. However, the barriers to entry for 3D modeling custom assistive technologies are high, which has lead to research on tools that can support amateurs without requiring mastery of modeling (e.g., [46]). In addition, assistive technology must typically interoperate with existing objects in the real world, which brings new challenges such as attachment [45] and interoperation [149].

In terms of accessibility for blind users, 3D printing has been used to produce custom labels on 3D printed objects [154], generate tactile maps [38, 69, 70, 162], support literacy skills through the creation of tactile picture books [106], teach design [130], mathematics [40], programming [101] and deliver tactile visualizations [37, 159]. These applications of 3D printing share a focus on 3D representations that can be customized beyond what is possible with the current state of the art (thermal printing or Braille labeling). However, this body of work assumes a sighted person who designs and produces the 3D printed artifact, which may limit a blind person's ability to access 3D printed solutions as needed.

One potential way of reducing the barriers to accessing sighted assistance is to shift the work to a virtual crowd [28, 33]. A number of crowd-powered systems have been developed to make visual information accessible to blind people [34]. VizWiz lets blind people take a picture, speak a question, and get answers back from the crowd within approximately 30 seconds [26]. Chorus:View [121] pairs a user with a group of crowd workers using a shared video stream. Be My Eyes [20] matches users to a single volunteer over a shared video stream. VizWiz::LocateIt [27] allows blind people to ask for assistance in finding a specific object. RegionSpeak [185] enables spatial exploration of the layout of objects in a photograph using a touchscreen. VizLens [74, 76] fuses crowdsourcing and computer vision to robustly and interactively help blind people use inaccessible interfaces in the real world, similar to a screen reader. Recently, physical crowds have been organized to construct pre-designed large-scale structures [115]. However, crowds have not in the past been used to create custom 3D printed objects. Facade combines a crowd interpretation pipeline with an accessible 3D printing application [77].

Another approach is to create new devices that are accessible, but this is unlikely to make all devices accessible due to cost. As more and more devices are connected to the Internet and can be controlled remotely, the problem becomes one of digital accessibility, which is easier to solve. For example, users may bring their own smartphone with an interface that is accessible to them, and use it to connect to the device [55, 142, 169]. Facade handles the legacy of inaccessible devices, which neither approach does.

To summarize, 3D printing can produce customized physical augmentations, and crowdsourcing can release the constraints of in-person sighted help through online and always-available visual assistance. Both have been applied with success in the domain of accessibility, including addressing the needs of blind users. Facade's novel contribution is in bringing these threads of research together to solve the important problem of making everyday appliances accessible.

5.3 Formative Study

To better understand how blind people currently use and accommodate home and office appliances, we conducted a formative study with 6 blind participants (all female, age 34-73). Four of the participants were congenitally blind, and the other two had light perception. All were Braille readers.

5.3.1 Procedure

We first went to the home of a blind individual, and observed how she cooked a meal and used home appliances. We then conducted semi-structured interviews with all participants. We asked questions about home appliance use, whether these appliances were accessible, if not, the ways employed to use these appliances, and strategies to label them. The studies were documented with video and audio recordings, as well as handwritten notes. We extracted key quotes and themes that reflected participants' personal strategies and challenges.

5.3.2 Results: Design Considerations

Participants remarked that interfaces are becoming much less accessible as flat digital touch pads replace physical buttons, which can at least be easily found by fingers once the locations of different functions were memorized. Appliances mentioned by participants were very diverse, and their interfaces differed in size, label, type of functions and number of buttons.

We identified four design requirements for a system to generate augmented physical interfaces for non-visual access. We refer to the participants in our formative study as F1 - F6 below (and also include related comments gathered later from our evaluation study participants P1 - P11).

Independence

Blind users often depend on in-person sighted assistance to identify the original functions and apply the labels on home appliances. When they bought a new appliance, they needed to wait for sighted help before being able to use the appliance.

> *My brother let me and my husband know what buttons are, we decide what buttons matter for us. And we write the Braille to label them, he again help us to stick onto buttons.* (P2)

The problem of existing solutions of applying Braille stickers, is that blind people cannot independently make appliances accessible. To address this, our solution should enable blind users to independently augment appliance interfaces, without needing to wait for help from sighted people.

Custom Settings

Participants had their own preferences and strategies for labeling. Simple dots (which could easily be counted and felt at a glance) were a popular choice on number buttons. Although not identical to Braille numeric characters, Braille readers also liked this strategy and only used Braille labels on more complex features, such as soft/melt, cook time, reheat, defrost, cancel and start on the microwave.

However, participants said they don't need all the buttons to be labeled. Some of them put bump dots or easily recognizable marker on frequently used buttons.

> *I put bump dots on only the 'add 30 seconds' button that I frequently use.* (F1)

When all the buttons are labeled with the same Braille dots, it's harder for them to find the number pad. Some of them mark only one of the number buttons (e.g., 0 or 5) as the reference to identify all others. F5 suggested that differentiating the number pad from other buttons could make interacting with the microwave faster.

> *Please indicate where the number starts, and that is enough. I can identify where other buttons are, it will make tasks quicker.* (F2)
>
> *I do not need to mark the entire number pad. 0, left and right are enough to get where number buttons are.* (F3)

Our solution should accommodate different preferred labeling strategies and reading mediums (Braille, printed letters, or dots). It should also use different shapes for functional buttons and number pads to reduce searching time.

Memorization Strategy

Since blind people were not familiar with the appliance functions, when using in-person sighted help for identifying the original functions and applying the labels, it was hard for them to remember the abbreviations and functions for more than a few buttons [134]. Therefore, they only tended to label a few buttons with only one or two Braille letters due to the limited size of the buttons, which limited their access to the appliances. There are also appliance interfaces that are hard to label. F1 reported making legend for a toaster oven since the buttons are hard to add labels on. Related to it, P5 stated:

> *I have an index card for a washer in my apartment, what normal hot and normal warm buttons are. I had my mom to help me to label when I moved in long time ago.* (P5)

To address this, our solution should better support learning and memorization of the appliance functions through the use of in-app support, or physical legend.

Robustness

The Braille labels applied to the interface will wear out over time. When it happened, blind people lost access to the specific buttons, and required sighted help again to reapply the labels.

> *We use microwave in the kitchen with dirty hands. Braille stickers are so easily fall off.* (P2)

To solve this problem, our solution should allow blind users to easily do the attachment independently. Furthermore, it should support easy reproduction and decrease the amount of effort required for the repeated work on the same appliance.

To summarize, our findings indicate that a solution for tactile labeling should allow blind users to independently augment and access their appliances. The solution should also support rich tactile feedback, diverse labeling strategies and preferences to address a wide range of individual needs. Furthermore, the solution should allow for learning and memorization of the interface, as well as easy attachment and reproduction.

5.4 Facade

Assisted by the Facade iOS app, blind users capture a photo of an inaccessible interface with a readily available fiducial marker (a dollar bill) for recovering size information. The web server transforms the image to the front perspective then feeds this image to multiple crowd workers, who work in parallel to quickly label and describe elements of the interface. These labels are then used to generate a 3D model for a layer of tactile and pressable buttons matching the original controls, which blind users can customize by changing the shape and labels of the buttons using the Facade iOS app. Finally, a home 3D printer or service fabricates the layer, which is then aligned and attached to the interface by blind users. Facade works as a pipeline, and is fully automated. Users do not need to attend to its full complexity.

5.4.1 Capture and Perspective Transformation

The first time a user encounters an interface, s/he uses the Facade iOS app to take a photo of the interface with a dollar bill (Figure 5.2A), and sends the image to be processed and pushed to the crowd for manual labeling. The dollar bill is used to produced an image of the interface warped to appear as if from the front perspective, and to recover size information. We use a dollar bill as an example to demonstrate the utility of using currency bills as fiducial markers because of its ubiquity, its standard size and appearance, and its richness in details and texture to provide sufficient feature points for tracking. We expect that a deployed version of Facade would allow users to choose their preferred bill in their local currency.

Facade uses SURF (Speeded-Up Robust Features) [19] feature detector to compute key points and feature vectors in both the standard image of the dollar bill and the input image. Then the feature vectors are matched using FLANN (Fast Library for Approximate Nearest Neighbors) [138] based matcher. By filtering matches and finding the perspective transformation [52] between the two images using RANSAC (Random Sample Consensus) [62], our system is able to localize the standard dollar bill image in the input image, and warp the input image to the front perspective for further labeling. Figure 5.2B shows the results of perspective transformation using a dollar bill. Using a system similar to VizLens [76], the Facade app streams images to the backend server, which then localizes either side of the dollar bill in the image and provides real-time feedback on the aiming of camera relative to the dollar bill to blind users. By reading out instructions such as “not found”, “move phone to left/right/up/down/further” and “aiming is good”, the app guides the blind user to more easily take a photo from the front perspective, which will result in better warped image after the perspective transformation. The computer vision components are implemented using C++ and the OpenCV Library.

Facade only has knowledge of the dollar bill and provides guidance based on its location, without knowing where the interface is. Blind users use this guidance, combined with their knowledge of the relative location of the interface and the dollar bill, to aim the camera and take photos. However, if the appliance interface is partially cropped in the photo, in the next step, crowd workers will provide feedback to the user for taking another photo. Using a second marker could help, but appliances might not have enough space to fit two markers. In the future, we could use more advanced techniques for helping blind users take photos [99, 128, 171, 176, 184].

5.4.2 Crowdsourced Segmenting and Labeling

Facade uses a two-step workflow to label the area of the image that contains the interface and then label the individual visual elements (Figure 5.2C), similar to those in VizLens [76]. Crowd workers are first asked to rate the image quality, and segment the interface region. Results are combined using majority vote. To assist with later attachment, we ask crowd workers to segment the interface region aligned with the physical boundaries of the appliance interface, so that blind people can feel that boundary and align the overlay themselves.

Crowd workers are then instructed to draw bounding boxes around all of the individual buttons within the interface area, and provide a text annotation for each element (such as labeling buttons as ‘baked potato’, ‘start/pause’). In this step, crowd workers work in parallel, and the worker interface shows labeled elements to other workers as they are completed.

5.4.3 Fabricating Accessible Augmented Layer

Labels are used to generate a 3D model for a tactile and pressable button layer, matching the original controls. After labeling by crowd workers, the blind user can use VoiceOver to customize the preferences for the tactile layer to be fabricated using the iOS app (Figure 5.2D). Blind users specify customizations using a virtual version of the interface displayed on their iPhone. Informed by our study, we allow individual buttons to be customized using Braille, embossed letters, or embossed symbols. Although embossed capital letters were not mentioned in our study, blind participants did mention using shared machines at home and at work with sighted people, which embossed letters allows for co-located access. Embossed letters also improve access for non-Braille readers, who can recognize capital letters almost as well as Braille readers recognize Braille [42]. Finally, users can customize the abbreviation strategy (i.e., which letters are used to represent a word or phrase); request a legend; edit the tactile label of individual buttons; set which buttons to label or remain flat; and customize the shape of buttons (useful for differentiating special buttons such as numbers).

Based on the results from our formative studies, we decide by default to detect and use different shapes for function (rectangular) and number (spherical) buttons when generating the 3D tactile overlay. Following common numeric keyboard or button pad accessibility conventions [167], by default we only label number 5 with a dot on the spherical button for the numbers.

The settings and the crowd-generated labels are then passed to our automated design tool. We implemented an OpenJSCAD script to generate the final STL files of 3D models of the augmented buttons for printing (Figure 5.2E). The input to the program is a generated JSON object including the dimensions of the tactile overlay, average button size, as well as the dimensions, positions, labels and preferences of each button. With this data, the script first generates groups of labeled 3D buttons. We determine the depth of the buttons to be proportional to the size of the buttons. To get the scale factor for Braille and letters, we first divide the button width by two to situate two characters, and then divide each area to hold two columns and three rows of dots including spacing. Compared to the standard dot radius and spacing size [15], the proportion is defined by this scale factor, and applied to determine the size of letters and symbols.

If short acronyms are not provided for each button label, the program automatically generates the abbreviations. By default, when adding Braille on top of the buttons, we use two characters for each button due to the limited surface area and the size of Braille characters: a word (e.g., ‘Clock’) is abbreviated by the first two letters (e.g., ‘CL’); and multiple words (e.g., ‘Power Level’) are abbreviated by the initial letters of the first two words (e.g., ‘PL’). When requested, a separate STL file is generated containing a legend (Figure 5.5e) detailing the abbreviations of the button labels, with the first column being acronyms, and the second column being the full words.

Our automated design tool then places buttons on top of a thin (2 layers in Gcode, 0.8mm) flat sheet, which creates a flat surface below the buttons that is easily attached to appliances with adhesives. Then, the program splits the tactile overlay into separate groups according to the 3D printer’s print bed size limit, and combines all sheets, buttons and embossed labels in each group into one piece for printing. The script can also merge multiple pieces as one print job based on print bed size to reduce print time. Our system exports files in ready-to-print STL format, which can be printed at the blind user’s home or through a commercial 3D printing service. An example 3D printed tactile overlay for a microwave is shown in Figure 5.2F. The overlay design

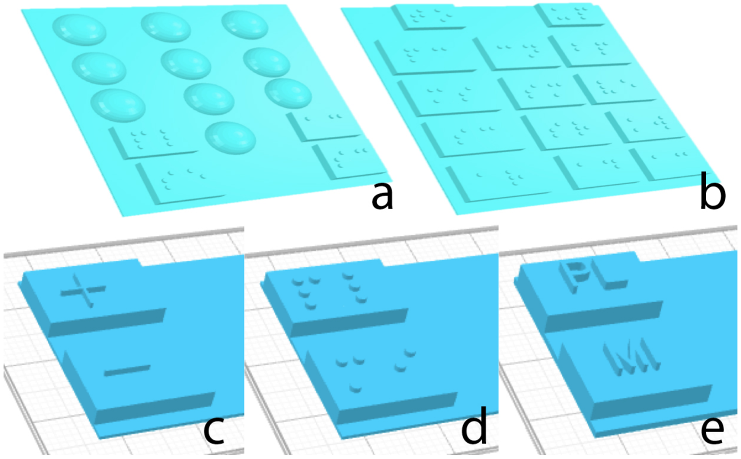


Figure 5.3: Shapes inform users of different functionalities. For example, half spherical buttons without Braille label indicates number buttons *(a)*, while rectangular buttons with Braille labels indicate function buttons *(b)*. Users are also able to change settings to use symbols *(c)*, Braille *(d)*, or embossed letters *(e)* for buttons labels such as plus and minus.

in Figure 5.3 was finalized after several design iterations as we detail in the next section.

5.5 Design Iterations

To produce the most effectively functioning tactile overlay, we went through several design iterations. The microwave we chose as the testing device was a Hamilton Beach 1.1 Cu Ft Microwave (Figure 5.4c). Similar to most common microwaves, 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). All of our tactile overlays used in design iterations and user evaluations were produced with off-the-shelf consumer grade 3D printers using the FDM (fused deposition modeling) technique of 3D printing.

5.5.1 Iteration #1: Design Probe

To test the 3D printed Facade overlay, we first created a design probe—a 3D printed sheet in PLA plastic of buttons labeled with Braille acronyms, attached to the microwave (Figure 5.4). We used an inverted cone shape for buttons, with the radius of the top surface corresponding to the actual size of the original button, and the radius of the bottom surface smaller (Figure 5.4a). Thus, the design reduces the pressure required for blind users to press on the top surface to activate the original buttons on the microwave. To minimize assembly time, we attached the buttons in a grid with connectors between buttons (Figure 5.4b), so that they could be batch printed, and also attached to the physical interface as a whole. We also made the connectors very thin so that the plastic buttons deform more easily when pressed. All of this design work was done by hand, but in a style that can be automatically generated for Facade.

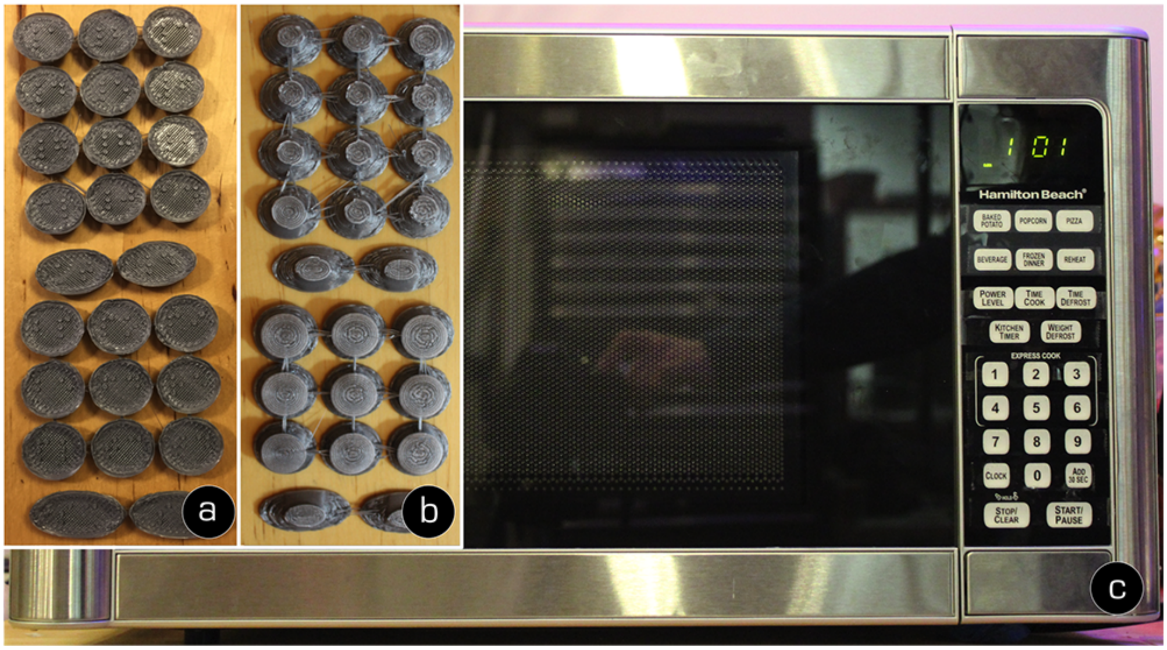


Figure 5.4: A design probe tested with 6 blind participants. An augmented button set with Braille labels *(a)* is attached to the microwave *(c)*, and buttons are connected with thin bridges to facilitate pressing *(b)*.

We tested this design with the same participants from our formative study, and identified the following issues:

- Some unexpected 3D printed artifacts on the edges of the top surface made the Braille dots feel overly rough, reducing legibility.
- Due to print resolution, Braille dots had different heights, reducing legibility.
- The plastic buttons were too hard to push.
- The button set did not attach to the microwave panel well and fell off after several times of use, due to the small contact regions.
- Because PLA does not deform, the connector bridges broke after pressing for a few times.

5.5.2 Iteration #2: Material Exploration

Informed by the participants' feedback to our initial design probe, we modified the design of the tactile overlay, and tested different combinations of materials (Figure 5.5a-d) to improve attacheability, legibility, and pressibility. Using a flat and thin sheet printed in flexible NinjaFlex [143] as the base of the overlay can make the augmentation much easier to attach to the appliance interface with adhesive (We used 3M removable double-sided Scotch Tape). The flexible material also made it much easier to press than using only PLA for the design probe.

While using NinjaFlex can improve attachability and pressability, it sometimes leaves undesired artifacts in the form of fine threads between Braille dots (think of melted cheese threads between pizza slices). These threads could reduce Braille legibility. One solution is to print Braille

Device	Material	Label
Hamilton Beach microwave	Flex*	Braille (Fig. 5.5a)
	Flex+PLA label	Braille (Fig. 5.5b)
	Flax+PLA cover	Braille (Fig. 5.5c)
	Flex+PLA cover	Letters (Fig. 5.5d)
	PLA legend	Braille (Fig. 5.5e)
Frigidaire Gallery microwave	Flex	Braille
Frigidaire fridge	Flex+PLA label	Braille
KitchenAid fridge	Flex	Letters + Symbol
Sharp microwave	Flex	Braille
Richo Alficio MP 6500 Copier**	Flex+PLA cover	Embossed letters Printed full words

Table 5.1: Interface, material and reading medium combinations used in design iteration #2 to improve attacheability, legibility, and pressibility. * Flex refers to NinjaFlex or SemiFlex for flexible material printing. ** Required manual intervention for raised buttons.

using hard material such as PLA (which we denote as Flex+PLA label), as shown in Figure 5.5b. A problem that occurred with this design is that these Braille dots may become dislodged from the button surface over time, due to the combination of heterogeneous materials. Another solution for this is to print several layers of the button together with Braille dots in PLA, while printing the rest of the bottom layers in NinjaFlex, resulting in a larger contact area between the two materials to allow them to stick together nicely (which we denote as Flex+PLA cover). Table 5.1 summarizes our experiments on various printing mediums and material combinations for a wide variety of home appliances.

We then obtained formative feedback of the examples shown in Table 5.1 from one blind individual (female, 24 years old, college student). In three different settings (i.e., pure NinjaFlex, Flex+PLA label, and Flex+PLA cover), the participant said all three testing material combinations were equally legible. Interestingly, she was most comfortable with reading the pure NinjaFlex version of the tactile overlay, despite the fine threads across dots. Unfortunately, both Flex+PLA label and Flex+PLA cover versions required her to press the button a lot harder to trigger the original interface. Overall, the NinjaFlex version of the tactile overlay had the best pressability and attachability among all material combinations we explored.

5.5.3 Iteration #3: Improved Legibility

Since the NinjaFlex version of printed Braille has enough detail and so is easily legible by a user, we printed the entire overlay in pure NinjaFlex including Braille. As guided by the user who tested our second design above, we also improved our design of the embossed letter version to make the letters thinner with larger gaps between letters for distinction.

For this improved design, we further tested the legibility of the Braille labels with two blind

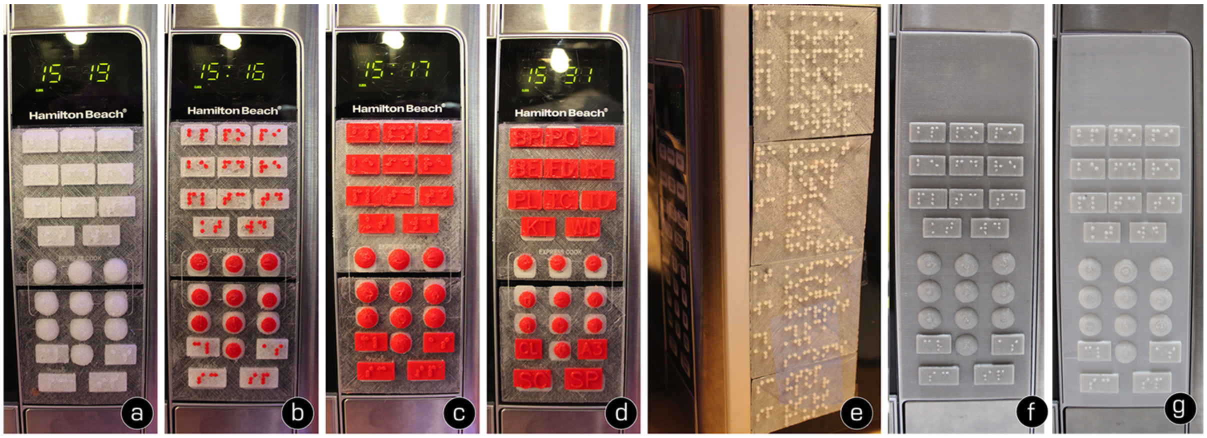


Figure 5.5: Example printed overlays and legends generated by Facade. *(a)-(d)* demonstrate the different material combinations we tested in the design iterations (NinjaFlex with Braille, Flex+PLA Braille label, Flex+PLA Braille cover, and Flex+PLA embossed letter cover). Facade users can choose to print a legend for the abbreviations *(e)*. If a user does not have a 3D printer at home, models can also be printed through commercial printing services and mail-ordered. *(f) and (g)* show two example prints ordered from 3D Hubs using PolyFlex and SemiFlex materials.

individuals (one female), both of whom provided formative feedback on the design. One suggested making the Braille dots more distinctive by raising the dots higher or reducing the button height. The other Braille expert, who works for a Braille publisher, suggested that Braille dots with a convex top are easier to read by touch than with a flat cylindrical top because convex tops provide a more salient separation between adjacent dots. Therefore, we changed the Braille dots from cylinders to domes, and finalized our design for the user evaluation we present next.

5.6 User Evaluation

The goal of our user study was to evaluate whether Facade allows blind people to independently augment appliance interfaces, and how the fabricated overlay performs in assisting blind people accomplish realistic tasks that involve otherwise inaccessible interfaces. Our user evaluation included each step that the blind user needs to do in Facade.

5.6.1 Apparatus and Participants

We used the same inaccessible microwave as detailed in design iterations. The Facade iOS app was used in the study, installed on an iPhone 5c, runing iOS 9.3.4. For this particular evaluation, all the images were labeled by the experimenter as introducing the crowd would result in compounding factors. The tactile overlays used in the study were generated and 3D printed beforehand to save time. The quality of perspective transformation, crowdsourced labeling, and model production is presented in the next section ("Technical Evaluation"). We recruited 11 blind users (6 female, age 40-82). The demographics of our participants are shown in Table 5.2.

ID	Gender	Age	Occupation	Vision Level	Reading Medium	Smartphone Use
P1	Male	63	Retired	Blind, since birth	Braille, 60 years	iPhone, 7 years
P2	Female	68	Retired	Blind, since birth	Braille, 62 years	iPhone, 8 years
P3	Female	75	Retired	Light perception	Braille, 20 years	No
P4	Male	82	Retired	Blind, since 8 years old	Braille, 75 years	No
P5	Female	46	Unemployed	Blind, since birth	Braille, 42 years	iPhone, 2 months
P6	Male	40	IT professional	Light perception	Mostly audio	iPhone, 10 months
P7	Female	42	AT consultant	Blind, since birth	Braille, 22 years	iPhone, 2 years
P8	Male	43	Rehab counselor	Blind, since birth	Braille, 36 years	iPhone, 10 years
P9	Female	58	Retired	Blind, since 1 year old	Braille, 50 years	iPhone, 5 years
P10	Female	61	Retired	Light perception	Braille, 35 years	iPhone, 6 years
P11	Female	68	Retired	Blind, since birth	Braille, 62 years	iPhone, 1.5 years

Table 5.2: Participant demographics for our user evaluation with 11 blind users.

5.6.2 Procedure

Following a brief introduction of the study and demographic questions, participants were asked to attach a one dollar bill next to the interface with the goal of facilitating photo taking in the next step to include both the dollar bill and the complete interface into the field of view. Then, participants were asked to take five photos of the microwave control panel with the assistance of the Facade iOS app, followed by another five photos taken with the built-in camera app on iOS. After each photo was taken, simulated crowd feedback on image quality was provided. These images were used to evaluate the perspective transformation and crowdsourced labeling.

Next, the labels for the microwave were entered into the app to simulate it having been crowd labelled, and participants were asked to explore the customization interface for identifying their reading medium and other preferences. Then, based on the reading medium chosen by the participants, the fabricated overlay of the microwave was presented to the participants, and they were asked to attach the overlay onto the microwave with double-sided tapes by aligning the edges. Images of the attached overlay was taken, and experimenters tested individual buttons to evaluate whether the alignment was sufficient for activating the microwave controls.

Next, we used a sheet of Braille or embossed letters in randomized order to familiarize participants with the shape and feeling of the tactile labels. Then, participants were asked to identify and read out the label of each button of the microwave. Accuracy was recorded, and participants were told the meaning of each abbreviation, e.g., BP for Baked Potato.

Next, participants were asked to complete 11 locating tasks and 4 simulated cooking tasks. For locating tasks, the participant was asked to locate a button with the assistance of the tactile overlay, and then push to trigger the button. Tasks included Power Level, Baked Potato, Frozen Dinner, Kitchen Timer, Clock, Popcorn, Time defrost, 0, 2, 4, and 8. For simulated 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. For both locating and simulated cooking tasks, we measured accuracy and time.

After performing tasks on the microwave with the tactile overlay using their reading medium preferences, we also tested overlays we printed out with other settings, such as the same microwave augmented with Braille or embossed letters, and a fridge interface augmented with embossed letters and symbols.

After each step of the study, we collected Likert scale ratings and subjective feedback from the participants. Finally, we ended the study with a semi-structured interview asking for the participant's comments and suggestions on the Facade system. The study took about one and a half hours and the participants were compensated for $50. The whole study was video and audio recorded for further analysis.

5.6.3 Results

We now detail our user study results and summarize user feedback and preferences. For all Likert scale questions, participants were asked to rate along a scale of 1 to 5, where 1 is very negative and 5 is very positive, e.g., 1 for very hard to perform, and 5 for very easy to perform.

Participants spent an average of 30.3 seconds $(SD = 19.1)$ to attach the dollar bill and found it very easy to perform $(M = 4.8, SD = 0.41)$. For taking photos assisted with the Facade mobile app, participants took an average of 33.6 seconds $(SD = 24.5)$ to take each photo, and rated neutral for the difficulty of taking photos $(M = 3.2, SD = 1.3)$. The reason why it was not easy was mainly because it required users to hold the device very stable, and there was no direct feedback of where the interface was. However, participants mentioned that after taking a few photos, feedback became easier to follow.

For applying the tactile overlay onto the microwave control panel, it took participants an average of 117.1 seconds $(SD = 83.0)$ to attach the overlay, including 2 of the 11 participants failed to attach the overlay correctly (Figure 5.6). Participants rated it relatively easy to attach the overlay $(M = 3.8, SD = 1.9)$. Participants applied the strategy of aligning from the top and using gravity to keep the overlay flat and align towards the bottom. P7 suggested that making the edge of the tactile overlay more distinctive can make it easier to align with the interface. Depending on the size of the buttons, slight offset will not affect using the appliance (such as Figure 5.6 P5). Furthermore, if the buttons are physically raised, they will also help with aligning the tactile overlay.

Identification Tasks

Ten out of 11 participants chose Braille as their primary reading medium and used our tactile overlay augmented with Braille labels, while P6 used the embossed letter version of tactile overlay. In order to compare participants' performance and report on our Braille quality, we only report the performance of the ten participants who used Braille. For identification tasks, it took participants an average of 112.6 seconds $(SD = 44.1)$ to read through all 25 buttons of the microwave, with an accuracy of 98.3% $(SD = 0.018)$. Errors happened to letters including C, D, and P. Participants rated reading the Braille as easy $(M = 4.2, SD = 0.92)$. The errors were mostly caused by the limited resolution of the printer and some remaining artifacts on the print. We believe this will be further resolved with improvements in 3D printers and printing materials, as detailed in the survey of different materials and printing techniques in the next section. Participants also mentioned that when sitting in front of the microwave in our study, their hands needed to be flipped backward when reading the Braille, which affected the accuracy. This effect will be reduced when they place the microwave at the position they prefer, and as they get familiar with the functions over time as they use them.

Locating and Simulated Cooking Tasks

For locating tasks, it took participants an average of 6.7 seconds ($SD = 4.6$) to locate and push to activate each of the function buttons, while it all took less than 1 second for the number buttons. The overall accuracy was 97.3% ($SD = 0.044$). We asked participants to locate the number pad in a separate task to evaluate whether the different shapes of function and number buttons facilitate locating, and all participants rated it as very easy (5). Participants also found it very easy to remember the button name by acronyms ($M = 4.9, SD = 0.32$), locate the buttons ($M = 4.6, SD = 0.70$), push the buttons ($M = 4.5, SD = 0.53$), as well as operate the microwave with the tactile overlay ($M = 4.8, SD = 0.42$).

Specifically for pushing the buttons, we observed there were 6 times across all 110 locating tasks participants needed to push the button more than once to activate it. Participants commented that with the overlay, they needed to apply slightly more force to activate the button than the original microwave, but it was still very easy to perform. P3 suggested making the buttons thinner and closer to the interface to reduce the force required, similar to a Dymo label tape.

For simulated cooking tasks, it took participants an average of 17.2 seconds ($SD = 10.1$) to complete each sequence, with an accuracy of 92.5% ($SD = 0.169$).

Embossed Letters and Symbols

Though none of the participants use embossed letter or symbol as their primary reading medium, they have mostly encountered them in everyday lives, such as in elevators, doorways, hotel rooms, or restrooms.

For identification tasks on the embossed letter version of tactile overlay for the microwave control panel, it took participants an average of 218.1 seconds ($SD = 132.9$) to complete all 25 buttons. And for a fridge overlay that contains both embossed letters and symbols, it took them an average of 142.0 seconds ($SD = 76.0$) to read through 10 buttons.

Subjective Feedback

When asked which of the three reading medium they preferred (i.e., Braille, embossed letter, symbol), all participants chose Braille, mostly because it aligns with their primary reading medium. P5 mentioned that if living with sighted people or people with partial vision, he could also accommodate with embossed letters.

When asked to compare Facade with the traditional method of applying Braille labels, participants commented *"I like [Facade] much better. I can do it myself, to me it's huge. I don't need to wait for someone to come over and label things for me. If template gets damaged, then I can*

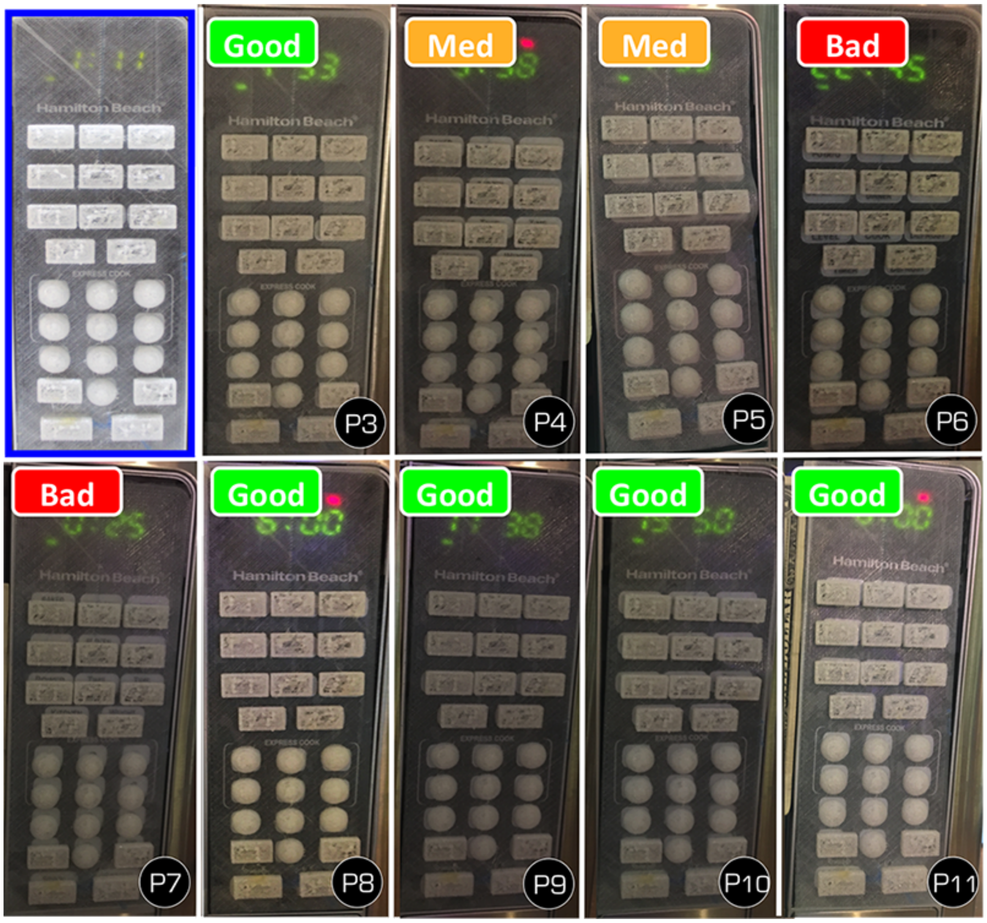


Figure 5.6: Examples of the attached overlay performed by the participants. For P4 and P5, slight offset did not affect using the appliance. An exact alignment is shown in the top left corner.

create a new one. With the [traditional] labels I made, things start get peeled off soon. I think this is neat (P1)", and *"This makes a lot more sense. Dymo easily fall off. I like this better (P6)."*

Participants also provided suggestions to make Facade work with interface widgets of other shapes, such as circular knobs (P9). P8 suggested to add a simple/advanced mode in the customization interface in the mobile app for people who prefer labeling the complete panel versus only a small set of buttons. P1 suggested that the feedback provided in the app should be more specific, such as saying "dollar bill in focus", "images are too close", etc.

Overall, participants were excited about the potential of Facade and several asked when they can use it on their appliances.

5.7 Technical Evaluation

We conducted a multi-part technical evaluation in order to understand how each component of the Facade pipeline performs across a range of interfaces and usage scenarios.

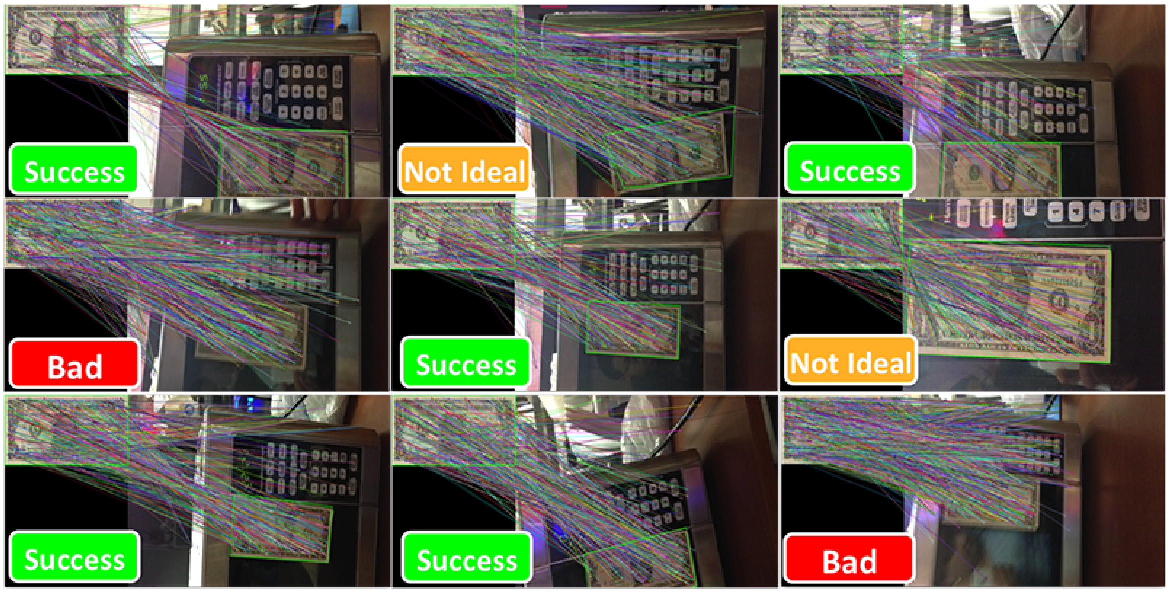


Figure 5.7: Examples for image localization and warping on photos taken by the participants using a dollar bill as the fiducial marker. Boxes show in green represents warping results that were good enough for generating a usable tactile overlay model, while those shown in orange and red represents failure cases.

5.7.1 Interface Capture

We first evaluated how well the Facade iOS app performs in assisting blind people in capturing photos containing both the dollar bill and the interface of the device, and how well our perspective transformation component performs in warping images to a front view of the interface.

We used photos of the microwave taken by the participants from our user evaluations (described in the previous section). Out of the 55 images taken when the Facade iOS app provided feedback, the perspective transformation component was able to identify the dollar bill and successfully warp the image to a front perspective for 34 cases (61.8%). In 4 of 55 cases, it identified the dollar bill, but the resulting warping was not suitable for further labeling and printing. In the remaining 17, the dollar bill failed to be localized. Pictures taken with the Camera app built into iOS were worse. Only 18 (32.7%) were successful, 3 were not ideal, and 34 did not localize the bill.

These image are then labeled by crowd workers, and labels are used to generate 3D models of the tactile overlay. Each segmenting task paid $0.15 (~40sec of work, $13.5/hr). Each labeling task paid $.02 (<10sec, $9/hr). We evaluated the crowdsourcing workflow, and generated analogous results to [76] as they share a similar crowdsourcing workflow. In prior work, the crowdsourcing labeling workflow was fast (8 minutes), accurate (99.7%), and inexpensive ($1.15). Accuracy is high because tasks are simple, and we perform automatic and redundancy checks on button size, aspect ratios, and labels. For the 34 successful pictures taken using the Facade app, Facade was able to generate a usable tactile overlay model for 21 of them (61.7%). On the other hand, 16 out of 18 images taken with the built-in Camera app were able to generate a usable overlay model. Figure 5.7 shows examples for image localization and warping on photos taken by the participants.

Material	**Printer**	**Resolution**	**Price**
NinjaFlex	Lulzbot TAZ5	Medium	$10.00
SemiFlex	Lulzbot TAZ5	Medium	$10.00
PolyFlex	Lulzbot TAZ5	Medium	$10.00
Nylon	ProJet 660 SLS	High	$31.40
Flexible Resin	FormLab SLS	High	$21.70

Table 5.3: We experimented with a variety of materials and printing techniques. Test prints were ordered from 3D Hubs.

The results show our Facade iOS app allowed participants to take better photos for generating the tactile overlay. It is important to note that since the Facade iOS app streams all images to the backend server when aiming the camera, we could configure our system to automatically pick several images where the dollar bill can be found before the user clicks the "take photo" button. This would also require adding another step in the crowdsourcing workflow for the crowd workers to select the best warped image.

5.7.2 Model Production

We next tested our pipeline on several other appliance interfaces, including three different microwaves, two refrigerators, and a printer, some of these are shown in Figure 5.1. In addition to varying appliances, we also experimented with a variety of printing techniques. Figure 5.5(a-d) shows tactile overlays printed with PrintrBot Simple Maker's kit (FDM) in our lab, each costs less than $5 (printed in 15% infill, 0.4mm layer thickness, 3 solid layers for top/bottom). Figure 5.5fg shows two example prints ordered from 3D Hubs using PolyFlex and SemiFlex materials. As shown in Table 5.3, we also tried out Nylon and Flexible Resin printed with SLS (selective laser sintering) printing technique, which generated much higher resolution Braille labels.

5.8 Discussion and Future Work

Facade enables blind people to access flat physical interfaces by auto-generating a tactile overlay. We focused on augmenting inaccessible buttons in this work, while this concept can be extended to work with other types of interface elements. For example, for a mechanical knob, tactile markings can be attached to the main interface, leaving the knob area empty, and a separate pointer printed and attached. Another approach is to fabricate a knob that supports internal movement.

Once the original physical interface is covered by the tactile buttons, sighted users living with blind users, or external caretakers cannot easily identify the original functions [36]. We chose transparent filament to print the overlaid buttons to see through the background. Yet, the button had multiple layers, which reduced transparency. To address this, we can support both sighted and blind people to access the appliances by applying different colors of materials to make the text labels visually salient, similar to Figure 5.5b. For appliances where visual elements are not as cluttered as the examples we show, we could place the tactile labels around the interactive elements and leave a hole for the button to directly make them accessible to sighted people (similar to Thingiverse thing: 1415446). Other attachment strategies can also be investigated with mechanical structures, such as hinge, flip door, sliding, etc.

For buttons that are not flat, our current approach of using a flat sheet for attachment wouldn't work. As an initial investigation, we show in the copier example in Figure 5.1 that additional measurements are required for creating concave structures to fit the embossed buttons. We have implemented this feature as an input parameter in our fabrication design tool. However, more advanced approaches need to be integrated to support the acquisition of this value. For example, instead of asking the blind person to take a photo, using a depth camera could better capture the convex properties of the physical interface.

For interfaces or buttons that are too small or cluttered, putting tactile labels on top of the buttons wouldn't work due to the fixed size of Braille. To mitigate this problem, we could configure Facade to print an overlay with minimal markings to attach to the interface, while generating another legend detailing the interface layout and labels on the side.

Our 3D printed augmentation is designed to overlay an interface which is triggered by manual force. If the augmented sheet covers a capacitive touchscreen, it would likely disrupt operation of the interface. One possible solution to address this problem is to print the button with conductive material that connects human skin's conductivity through the 3D printed objects. While this is an interesting approach and needs to be investigated to expand the range of interface that Facade can support, we leave this for future work.

The cost of Facade is rapidly changing and it may soon be competitive with creating labels with tape. 3D printing material is quickly getting cheaper, and approaching that of embossed labels using Dymo tape. In our current pipeline, the interface layout and labels are generated from the crowdsourcing workflow. However, these could also be acquired from remote friends or family, provided by the appliance manufacturers, or automatically retrieved from online manuals. Collectively, labels for common appliances may also benefit a new user. Furthermore, similar approaches can be used for other tasks, such as for a sighted partner or a building manager to quickly collect images and automatically produce tactile labels and augmentations to make a space accessible, which is likely more efficient than manual labeling.

Our evaluation demonstrates the viability of Facade by deeply evaluating each component. This will guide us (and others) in future work to understand how each component is likely to work in deployment and how we might usefully improve the system (e.g., using more advanced blind photography, using more robust crowd labeling workflows, and applying other fabrication techniques or materials)

5.9 Conclusion

We have presented *Facade*, a crowdsourced fabrication pipeline for blind users to augment inaccessible physical interfaces by 3D printed tactile overlays. Our system empowers blind users to access physical interfaces in everyday lives in an independent and inexpensive way. We introduced the design of the system and its technical architecture, evaluated it in a user study with 11 blind participants, and evaluated each component separately to understand its limitations.

Compared with traditional embossed labelers, Facade does not require in-person sighted assistance, provides richer tactile feedback using different reading mediums and button shapes, and reduces memory load by providing a legend and in-app support: embossed labels do none of these. Our research envisions a future when 3D printers are faster and more ubiquitous in people's homes. Facade can benefit blind users by generating tactile overlays to home appliances in minutes, complementing or replacing in-home embossed labelers.

VizLens and Facade enable blind users to access many *static* interfaces. However, many interfaces include *dynamic* components, such as an LCD screen on a microwave, or the dynamic interface on self-service checkout counter. Interacting with dynamic touchscreens is difficult non-visually because the visual user interfaces change, interactions often occur over multiple different screens, and it is easy to accidentally trigger interface actions while exploring the screen. To address these problems, in the next chapter (Chapter 6), I introduce *StateLens*, a three-part reverse engineering solution that makes existing dynamic touchscreens accessible.

Chapter 6

StateLens: A Reverse Engineering Solution for Dynamic Touchscreens

Blind people frequently encounter inaccessible dynamic touchscreens in their everyday lives that are difficult, frustrating, and often impossible to use independently. Touchscreens are often the only way to control everything from coffee machines and payment terminals, to subway ticket machines and in-flight entertainment systems. Interacting with dynamic touchscreens is difficult non-visually because the visual user interfaces change, interactions often occur over multiple different screens, and it is easy to accidentally trigger interface actions while exploring the screen. To solve these problems, we introduce *StateLens* — a three-part reverse engineering solution that makes existing dynamic touchscreens accessible. First, StateLens reverse engineers the underlying state diagrams of existing interfaces using point-of-view videos found online or taken by users using a hybrid crowd-computer vision pipeline. Second, using the state diagrams, StateLens automatically generates conversational agents to guide blind users through specifying the tasks that the interface can perform, allowing the StateLens iOS application to provide interactive guidance and feedback so that blind users can access the interface. Finally, a set of 3D-printed accessories enable blind people to explore capacitive touchscreens without the risk of triggering accidental touches on the interface. Our technical evaluation shows that StateLens can accurately reconstruct interfaces from stationary, hand-held, and web videos; and, a user study of the complete system demonstrates that StateLens successfully enables blind users to access otherwise inaccessible dynamic touchscreens.

6.1 Introduction

Inaccessible touchscreen interfaces in the world represent a long-standing and frustrating problem for people who are blind. Imagine sitting down for a 12-hour flight only to realize that the entertainment center on the seatback in front of you can only be controlled by its inaccessible touchscreen; imagine checking out at the grocery store and being required to tell the cashier your pin number out loud because the checkout kiosk is an inaccessible touchscreen; and, imagine not being able to independently make yourself a coffee at your workplace because the fancy new coffee machine is controlled only by an inaccessible touchscreen. Such frustrating accessibility

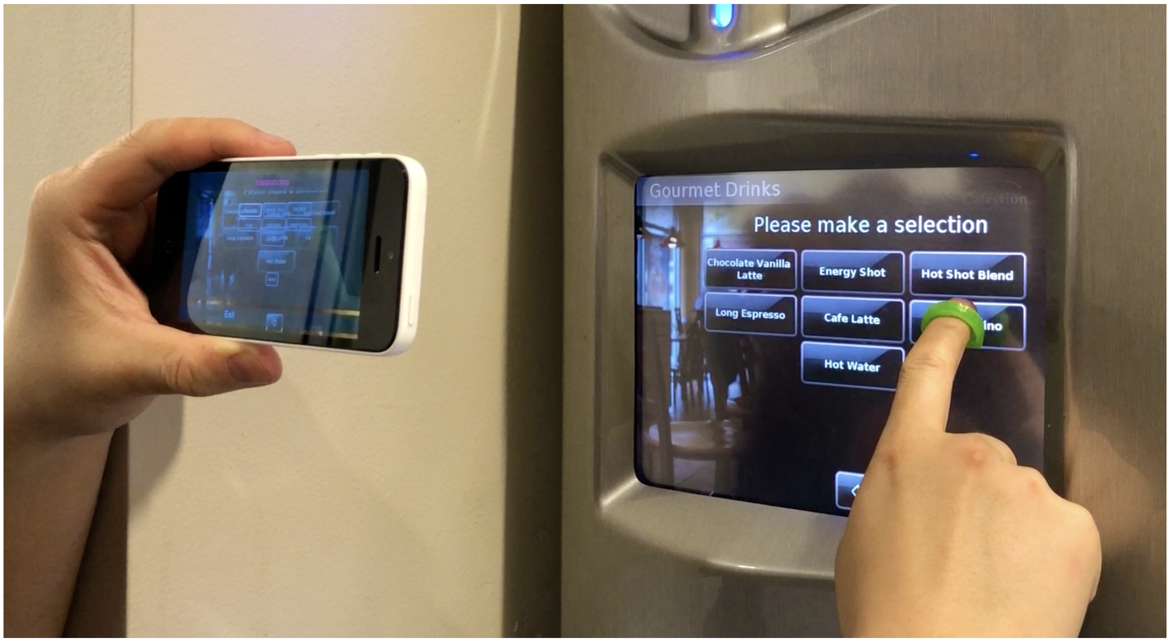


Figure 6.1: StateLens is a system that enables blind users to interact with touchscreen devices in the real world by *(i)* reverse engineering a structured model of the underlying interface, and *(ii)* using the model to provide interactive conversational and audio guidance to the user about how to use it. A set of 3D-printed accessories enable capacitive touchscreens to be used non-visually by preventing accidental touches on the interface.

problems are commonplace and pervasive.

Making touchscreen interfaces accessible has been a long-standing challenge in accessibility [65, 76, 137], and some current platforms are quite accessible (e.g., iOS). Solving all of the challenges represented by the combination of difficult issues for public touchscreen devices has remained elusive: *(i)* touchscreens are inherently visual so a blind person cannot read what they say or identify user interface components, *(ii)* a blind person cannot touch the touchscreen to explore without the risk of accidentally triggering something they did not intend, and, *(iii)* a blind person does not have the option to choose a different touchscreen platform that would be more accessible and cannot get access to the software or hardware to make it work better. This work is about enabling blind people to use the touchscreens they encounter *in-the-wild*, despite the fact that nothing about how these systems are designed is intended for their use.

Most prior work on making touchscreens accessible has assumed access to change or add to the touchscreen hardware or software. For example, physical buttons were added to the side of the screen to provide a tactile way to provide input [169, 170]. Slide Rule developed multi-touch gestures that could control touchscreens non-visually [102], which have informed the popular VoiceOver screen reader on the iPhone. In the real world, users cannot control the touchscreens they encounter, and many are not accessible. In response, recent work has considered making existing interfaces accessible using computer vision and crowdsourcing to interpret the interfaces on-the-fly and provide immediate feedback to users [76]. This approach can work for many static interfaces, but struggles when the interface changes dynamically (as most touchscreens

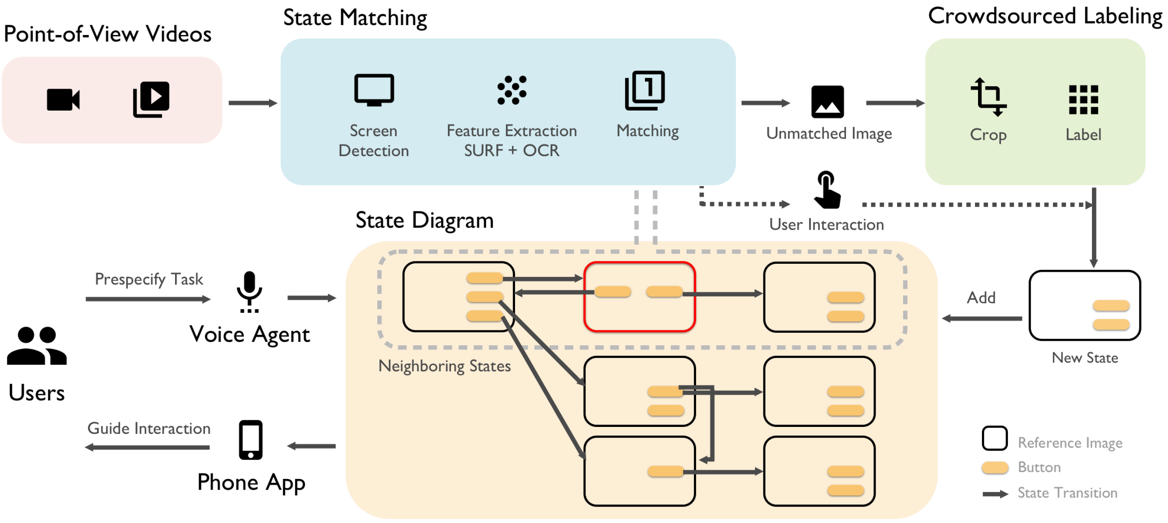


Figure 6.2: StateLens uses a hybrid crowd-computer vision pipeline to dynamically generate state diagrams about interface structures from point-of-view usage videos, and using the diagrams to provide interactive guidance and feedback to help blind users access the interfaces.

do), and cannot solve the problem of how a blind user could interact with a touchscreen without accidentally triggering touches.

This work introduces *StateLens*, a reverse engineering solution for making existing dynamic touchscreens accessible. StateLens works by reverse engineering state diagrams of existing interfaces from point-of-view usage videos using a hybrid crowd-computer vision pipeline (Figure 6.2). Using the state diagrams, StateLens automatically generates conversational agents that guide blind users to prespecify tasks (Figure 6.5). The StateLens iOS application then provides interactive guidance and feedback to help blind users access the interfaces (Figure 6.1). StateLens is the first system to enable access to dynamic touchscreens in-the-wild, that addresses the very hard case in which blind users encounter a touchscreen that is inaccessible and unfamiliar, which they cannot modify the hardware or software, and whose screen updates dynamically to show new information and interface components.

A known challenge for touchscreen interfaces is that they cannot easily be explored non-visually without the risk of accidentally triggering functions on the screen. Slide Rule developed the notion of “risk-free exploration” to counter this problem [102], but their solution (requiring multiple taps instead of just one) requires being able to modify how the touchscreen operates. StateLens is intended to work on touchscreens already installed in the world that are not possible to be modified. To do this, we introduce a set of simple 3D-printed accessories that allow users to explore without touching the screen with their finger, and perform a gesture to activate touch at a desired position. These accessories add “risk-free exploration” to existing touchscreen devices without modifying the underlying hardware or software.

In a formative study, we first identified key challenges and design considerations for a system to provide access to dynamic touchscreen interfaces in the real world. Our technical evaluation showed that StateLens can accurately reconstruct interface structures from stationary, hand-held, and web usage videos. Furthermore, the generated state diagrams effectively reduced latency and prevented errors in the state detection process. Then through a user study with 14 blind participants, we showed that the conversational agent, the iOS application, and the 3D-printed accessories collectively helped blind users access otherwise inaccessible dynamic touchscreen devices effectively. StateLens represents an important step for solving this long-standing accessibility problem, and its technical approach may find applications broadly for augmenting how people interact with the touchscreens they encounter.

6.2 Related Work

Our work is related to prior work on *(i)* reverse engineering user interfaces, and *(ii)* improving the accessibility of existing physical interfaces. StateLens is intended to solve a long-standing and hard problem at the intersection of these spaces.

6.2.1 Reverse Engineering User Interfaces

A core feature of StateLens is its ability to reverse engineer user interfaces *in-the-wild* based on videos of their use. Substantial prior work exists in reverse engineering user interfaces using computer vision from "pixels." This approach has been recognized as one of the most universally applicable methods for understanding a user interface's components, which is somewhat surprising given that at some level most user interfaces have been created with libraries that in some way had knowledge of their semantics. Unfortunately, that information is often either lost or inaccessible once the user interface makes it into a running system. StateLens is intended to make user interfaces accessible that are on public touchscreen devices, to which access is purposefully restricted.

Prior work on reverse engineering of user interfaces has mainly used sceenshots or screencast videos. These approaches have looked to automatically extract GUI components from screenshot images in order to decouple GUI element representation from predefined image templates [17, 44, 56, 97, 178], to augment existing interfaces through understanding of GUI components [17, 56], and to extract interaction flows from screencast videos and screen metadata [107, 123, 124, 182]. Prefab [56] identifies GUI elements using GUI-specific visual features, which enables overlaying advanced interaction techniques on top of existing interfaces. Sikuli [178] uses computer vision to identify GUI components in screen captures for search and automation in the interfaces. Hurst *et al.* [97] combine a number of useful computer vision techniques with mouse information to automatically identify clickable targets in the interface. Chang *et al.* [44] propose an accessibility and pixel-based framework, which also allow for detecting text and arbitrary word blobs in user interfaces. Waken [17] recognizes UI components and activities from screencast videos, without any prior knowledge of that application.

Some of the prior work has gone beyond the task of identifying individual GUI components from static photos, and looked instead to extract interaction flows from screencast videos and screen metadata provided by the system API. For instance, FrameWire [124] automatically extracts interaction flows from video recordings of paper prototype user tests. Using Android's accessibility API, Sugilite [123] and Interaction Proxies [182] extract the screen structures, in order to create automation and improve mobile application accessibility. Kim *et al.* [107] apply a crowdsourcing workflow to extract step-by-step structure from existing online tutorial videos.

StateLens builds on this rich literature, and applies a hybrid crowd-computer vision pipeline to automatically extract state diagrams about the underlying interface structures from point-of-view usage videos. In contrast to prior work, StateLens is a solution for reverse engineering existing physical interfaces through much noisier point-of-view videos rather than screenshots or prototyped GUIs.

6.2.2 Improving Accessibility for Physical Interfaces

Many physical interfaces in the real world are inaccessible to blind people, which has led to substantial prior work on systems for making them accessible. Many specialized computer vision systems have been built to help blind people read the LCD panels on appliances [65, 137, 163]. These systems have tended to be fairly brittle, and have generally only targeted reading text and not actually using the interface.

Crowd-powered systems robustly make visual information accessible to blind people. VizWiz lets blind people take a picture, speak a question, and get answers back from the crowd within approximately 30 seconds [26]. More than 10,000 users have asked more than 100,000 questions using VizWiz [85]. Users often ask questions about interfaces [35], but it can be difficult to map the answers received, e.g., “the stop button is in the middle of the bottom row of buttons”, to actually using the interface because doing so requires locating the referenced object in space (e.g., place a finger on the button).

Other systems provide more continuous support. For example, Chorus:View [121] pairs a user with a group of crowd workers using a managed dialogue and a shared video stream. “Be My Eyes” matches users to a single volunteer over a video stream [20]. These systems could more easily assist blind users with using an interface, but assisting in this way is likely to be cumbersome and slow. RegionSpeak [185] and Touch Cursor [80] enable spatial exploration of the layout of objects in a photograph using a touchscreen. This can help users understand the relative positions of elements, but they still have the challenge of physically locating the elements in space on the real interface in order to use it.

Static physical interfaces can be augmented with tactile overlays to make them accessible. Past research has introduced fabrication techniques for retrofitting and improving the accessibility of physical interfaces. For example, RetroFab [149] is a design and fabrication environment that allows non-experts to retrofit physical interfaces, in order to increase usability and accessibility. Facade [78] is a crowdsourced fabrication pipeline to help blind people independently make physical interfaces accessible by adding a 3D-printed augmentation of tactile buttons overlaying the original panel.

VizLens [76] is a screen reader to help blind people use inaccessible static interfaces in the real world (e.g., the buttons on a microwave). StateLens goes beyond VizLens by enabling access to dynamic touchscreens. Without the 3D-printed accessories introduced in this work, VizLens would not work for touchscreens. VizLens users would also need to take pictures when the screen changes, which is difficult. With VizLens, at each step, a good picture must be taken, labeled, and only afterwards can users explore the buttons on the single screen. Each screen iteration would take several minutes, making it cumbersome to use for dynamic interfaces.

VizLens::State Detection is able to do limited adaptation to dynamic interfaces by matching against every possible state and providing feedback based on the best match. However, because of changing display states and screen layouts, exploring and activating UI components across multiple screens is difficult (analogous to finding one's way in a new city). By generating and using state diagrams, StateLens enables a crucial interaction of previewing and prespecifying tasks through a conversational agent (analogous to using map applications to plan trips and follow turn-by-turn directions). The 3D-printed accessories make exploration possible by bringing risk-free exploration to touchscreens.

6.3 Formative Study

We conducted a formative study to identify the key challenges and design considerations for a system to provide access to dynamic touchscreen interfaces in the real world. We conducted semi-structured interviews with 16 blind people about their experiences and challenges with public touchscreen appliances, and their strategies for overcoming these challenges. Then using a Wizard-of-Oz approach, we asked two participants to try using a touchscreen coffee machine with verbal instructions given by the researchers. We extracted key insights that reflected participants' challenges and strategies, which we used in the design of StateLens.

6.3.1 Design Considerations

Participants remarked that interfaces are becoming much less accessible as flat touch pads and touchscreens replace physical buttons. Touchscreen appliances mentioned by participants were very diverse, and their interfaces differed in size, type of functions and number of buttons.

Supporting Independence

Participants often resorted to sighted help when accessing public touchscreen appliances, and raised serious privacy concerns when asking others (often strangers) to help with entering sensitive information, e.g., using credit card machines to complete financial transactions, or using sign-in kiosks at pharmacies and doctors' offices. Participants also mentioned sighted people giving incorrect or incomplete information because of a lack of patience or experience helping blind people. Our solution should enable blind people to independently access touchscreen devices without needing sighted assistance.

Reducing Cognitive Effort

For unfamiliar dynamic touchscreen devices, the amount of time and cognitive effort needed for blind people to explore, understand, and activate functions became quite heavy. Participants noted that if it were for a one-time use, it would not be worthwhile to invest the time and effort to learn the interface, which is much easier for sighted people. Our solution should support more fluid

Category	Example Thingiverse Items
Styluses (17)	iPad drawing pencils (thing:8976); capacitive stylus (thing:2870398, thing:225001); resistive stylus (thing:1582974, thing:577056); mouth sticks (thing:1321021); wrist-cuff stylus (thing:1315004); Nintendo 3DS Stylus (thing:798010)
Prosthetic Accessories (10)	prosthetic hands (thing:1717809, thing:380665, thing:242639); prosthetic finger (thing:2527421, thing:2840850)
Finger Caps (6)	thimbles around or over the fingertip (thing:612664, thing:1044791); thumb protectors (thing:28722); adapter to hold another object on finger (thing:2133318)
Buttons (4)	button grid for mouse input (thing:2745606); mechanical triggers for mobile phone games (thing:2960274); assistive button via phone's microphone input jack (thing:1471760); braille button input for phone (thing:1049237)
Joysticks (2)	touchscreen mounted capacitive joystick (thing:2361676, thing:2361676)

Table 6.1: Categorization of our Thingiverse survey results related to assistive technologies, touchscreens, and finger-based interactions. The number of items is shown next to each category name.

interactions to reduce blind users' cognitive effort in exploring the interface layout and accessing functions on complex and unfamiliar touchscreen devices.

Enabling Risk-Free Exploration

Participants shared their concerns and fears of accidentally triggering functions on inaccessible touchscreens. For example, a participant mentioned that once in a few weeks she would accidentally hit the settings button on her fridge's touchscreen panel, then she needed to call someone to come and check on it, which has been a huge burden.

When attempting to use existing inaccessible touchscreen devices, participants found holding their fingers in mid-air while trying to explore and locate the buttons to be very awkward and unusable, which also often resulted in accidental touches. Therefore, our solution should support "risk-free exploration" to enable blind users freely explore without accidentally triggering functions on the screen.

6.4 Risk-Free Exploration

Risk-free exploration allows blind users to freely explore without accidentally triggering functions on the screen, all without modifying the underlying hardware or software of the device.

6.4.1 Thingiverse Survey

We first conducted an exploratory search on Thingiverse to understand what openly available solutions exist for people to interact with touchscreens and see if they can enable risk-free exploration for blind people. We created a list of 11 search terms including: touchscreen accessibility; touchscreen stylus; screen stylus; capacitive screen input; resistive screen; input assistive; assistive finger cap; finger cap; 3D printed accessibility; conductive PLA accessibility; and prosthetic finger. These search terms resulted in a total of 103 existing designs. We then filtered results that were not related to accessibility or assistive technology (*e.g.*, raspberry pi and/or touchscreen cases), leading to a total of 39 relevant items.

Using an approach akin to affinity diagramming [24, 47, 92], we classified these items into five main categories of devices: styluses, prosthetic accessories, finger caps, buttons and joysticks. We show each of these categories with example items in Table 6.1. Although the Thingiverse designs are closely related to assistive usage for touchscreens, none of them satisfy our need to enable blind users risk-free access to an existing touchscreen device. We used these categories to inspire design ideas for prototypes that take on familiar forms used in the Thingiverse accessibility community but also support risk-free exploration (Figure 6.3).

6.4.2 Finger Ring Prototype

Inspired by the finger cap designs from Thingiverse, we first created a 3D-printed ring that allows users to explore without touching the screen, and tilt their finger forward to perform a touch at a desired position (Figure 6.3A-C).

We tested this design in a pilot study with two blind participants (one female, age 48; one male, age 57). While the 3D-printed finger ring enabled our participants to explore without accidental triggers, participants also identified issues related with the design and suggested other solutions. For example, the location of the ring on the finger may vary for different users and different sessions during use, thus changing the actual position of touch. Furthermore, when pressing the finger and finger ring on the touchscreen, it was uncomfortable for the participants for certain angles and postures. This is worsened when they are asked to only use one finger to interact with the interfaces, in order to prevent accidental touches.

6.4.3 Design Variations

Informed by the participants' feedback to our initial prototype, we designed variations of 3D-printed accessories (Figure 6.3DG) that focus on improving stability and comfort during use. The designs aim to reduce the change of "touchpoint" when the user moves from exploration to interaction (*i.e.*, touch activation), and maintain consistency across sessions. We also focused on capacitive touchscreens rather than resistive touchscreens, since resistive screens usually require some pressure to activate so the issue of accidental activation is not as severe compared to capacitive touchscreens.

Our design variations consist of a finger cap (Figure 6.3D) and a conductive stylus (Figure 6.3G). The finger cap prevents accidental touches by shielding undesirable areas of the finger from touching the touchscreen. The cap has an opening on the finger pad that allows the user to tilt their finger to activate a touch (Figure 6.3DEF). Compared to the ring design, the finger cap's enlarged shielding area and top cover prevent accidental touches more effectively and ensure consistency across sessions. This finger-worn design also incorporates a slit so that when 3D printed with a flexible material (e.g., thermoplastic polyurethane – TPU), it can fit around fingers of different sizes. The stylus uses a conductive trace to trigger touches at the tip of the stylus when touched by a finger (Figure 6.3GHI). It provides a physical affordance to prevent accidental

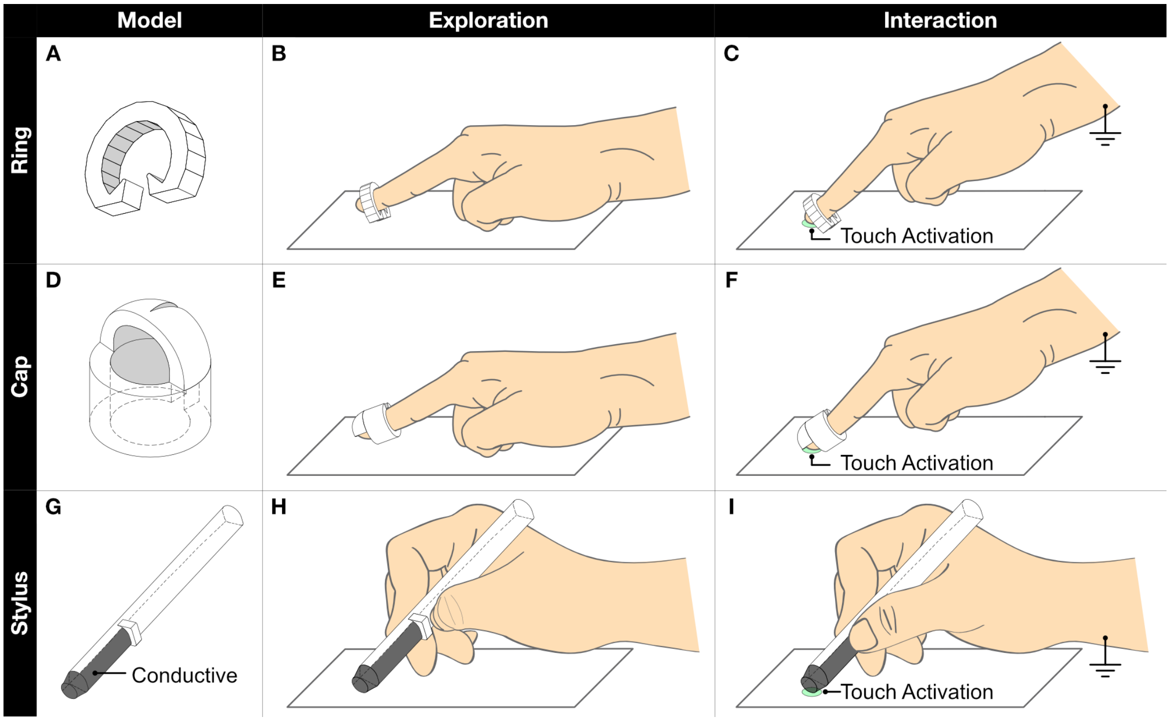


Figure 6.3: A set of 3D-printed accessories that prevent the wearer from accidentally triggering touches while exploring the interface. When desired, the wearer can activate a touch using either a tilt motion (B-C and E-F) or by touching a conductive trace on the accessory with a finger (H-I). These accessories elegantly add "risk-free exploration" to existing capacitive touchscreen devices without modifying the underlying hardware or software, which has been a major hurdle for past efforts. 3D models of these accessories are available at: `https://github.com/mriveralee/statelens-3dprints`

touches, by delineating the conductive and non-conductive regions with a rectangular bumper located on the side of the stylus. Conductive traces can be applied using conductive paint or printed with conductive PLA on a dual extrusion 3D printer. We had success with both techniques, though conductive PLA was more durable, while conductive paint can come off after repeated use.

6.5 StateLens

StateLens uses a hybrid crowd-computer vision pipeline to dynamically generate state diagrams about interface structures from point-of-view usage videos, and to provide interactive feedback and guidance to help blind users access the interfaces through these diagrams. We use the coffee machine in Figure 6.4 as a running example.

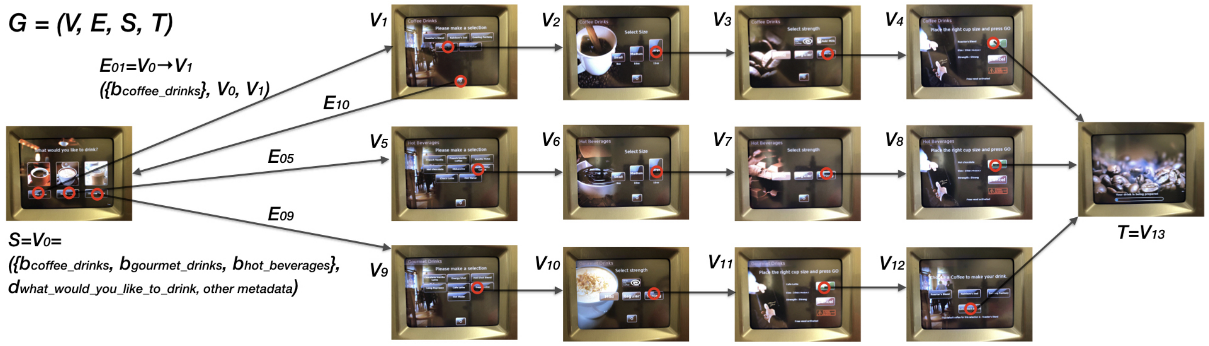

$G = (V, E, S, T)$
$E_{01}=V_0 \rightarrow V_1$
$(\{b_{coffee\_drinks}\}, V_0, V_1)$
E_{10}
E_{05}
E_{09}
$S=V_0=$
$(\{b_{coffee\_drinks}, b_{gourmet\_drinks}, b_{hot\_beverages}\},$
$d_{what\_would\_you\_like\_to\_drink}$, other metadata$)$
V_1
V_2
V_3
V_4
V_5
V_6
V_7
V_8
V_9
V_{10}
V_{11}
V_{12}
$T=V_{13}$

Figure 6.4: Visualization of how StateLens represents the coffee machine interface structure as a state diagram. Note only some edges are shown.

6.5.1 Generating the State Diagram

The architecture of StateLens to generate state diagrams (Figure 6.2) involves capturing point-of-view usage videos from a variety of sources, representing state diagrams, detecting screen regions, identifying existing and new states, soliciting labels from the crowd, as well as recognizing user interactions.

Capturing Point-of-View Usage Video

StateLens takes point-of-view usage videos of dynamic interfaces from various sources as input to build up state diagrams about interface structures. These videos can be collected in many ways, including through existing IoT and surveillance cameras, through motivating sighted volunteers to contribute videos using mobile and wearable cameras, by encouraging manufacturers to share videos as a low-cost way to make their systems accessible to more people, and by mining existing demo and tutorial videos in online repositories. For example, a search on YouTube for "coca cola freestyle machine demo" produces many usage videos. In the current work, we demonstrate StateLens with videos captured from stationary cameras, hand-held mobile phones and web video repositories.

Representing State Diagram

StateLens represents the interface structure with a state diagram, as shown in Figure 6.2 and the instantiation of the coffee machine shown in Figure 6.4. We represent a state diagram as a directed graph $G = (V, E, S, T)$ where S is the start state and $T = \{T_1, T_2, ..., T_n\}$ contains the end states where tasks are accomplished. Each node (state) $V_i \in V$ can be represented as $V_i = (\{b_1, b_2, ..., b_n\}, \text{descriptions, coordinates, other metadata})$, where b_n is one of the interactive elements (e.g., buttons) in state V_i. Each edge (transition) from state V_i to state V_j is $E_{ij} \in E$ that can be represented as $E_{ij} = (\{b_1, b_2, ..., b_m\}, V_i, V_j)$. Note that here b_m represents the button identifier in the metadata of *"from state"* that caused the state transition into *"to state."* Following our running example, the transition from the initial state $S = V_0 = (\{b_{\text{coffee\_drinks}}, b_{\text{gourmet\_drinks}}, b_{\text{hot\_beverages}}\}, \text{other metadata})$ to the coffee drink type state

V_1 can be represented as: $E_{01} = V_0 \rightarrow V_1 = (\{b_{\text{coffee\_drinks}}\}, V_0, V_1)$, stating that by interacting with button "Coffee Drinks" in the initial state, we could get to the desired state for coffee drinks type selection. Similarly, the transition to go back to the initial state can be represented as: $E_{10} = V_1 \rightarrow V_0 = (\{b_{\text{back}}\}, V_1, V_0)$.

Detecting the Screen

StateLens detects whether a screen is present and its bounding box in the camera's field of view to filter out irrelevant video frames and random background content. Since there is no existing models for detecting touchscreen interfaces, we re-purpose state-of-the-art object detection models' output for this task. Using the Amazon Rekognition Object Detection API [14], StateLens first detects bounding boxes of object categories related to electronics and machines. If such bounding boxes exist and their sizes are above 10% of the image size (aiming to filter out objects that are not the one of interest), StateLens crops the image using the bounding box to remove background noises for further processing. If not, StateLens checks whether the output labels with high confidence scores (above 55%) appears in the above categories. If so, the full video frame is retained and used for further processing. If not, the frame is determined irrelevant and discarded. StateLens is quite lenient in this step to prevent accidentally removing relevant frames, in order to maintain a high recall.

Identifying Existing States

StateLens extracts two kinds of features and intelligently combines them (Figure 6.2): SURF (Speeded-Up Robust Features) [19] and OCR. StateLens first uses SURF feature detectors to compute key points and feature vectors in both the existing state reference images (Figure 6.4) and the input image. 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 [52] between the reference-input image pairs using RANSAC (Random Sample Consensus) [61], StateLens is able to compute the ratios of inliers to the number of good matches for each existing state. It then uses the reference state with the distinctly highest ratio as the candidate matched state.

If the highest matched ratio across existing reference images is not high enough, meaning the match using only SURF features is not so confident, StateLens then uses the Google Cloud Vision API [68] to compute OCR results for the input image and compares to the pre-computed OCR results of the state reference image. Similarity is defined as the ratio of longest common sequence (LCS) edit distance to the length of the OCR output results, and if above a threshold, the candidate matched state is finalized as the matched state. For example, matching V_1 against V_5 results in low confidence with SURF, then with additional information provided by OCR, StateLens is able to differentiate them. On the other hand, if both the matched inlier ratio and the OCR similarity score are below a certain threshold, StateLens determines it as not a match.

Adding New States

When a transition happens on the dynamic interface, the new state might not have been seen before. If an input image is not a match with the existing states, StateLens adds it to a candidate pool. Then, for the next images which are also added into this pool, they are matched against the existing candidates. Using this candidate pool approach, only when the same image is seen continuously across multiple frames, StateLens is confident enough to register it as a new state. Among the candidates identified as the same state, StateLens automatically selects the last one added to the pool as the reference image for this new state. We do so because the first few candidates often include transition residuals from the previous state, such as animations. We use a time window of 1 second for this process. On the other hand, if continuous unmatched states in the pool do not reach the window size to qualify as a new state, they are considered noise and the candidate pool will be cleared. Once a new state is registered, StateLens then sends it to the crowdsourced labeling pipeline to acquire more information such as the interface region, interaction components, and description (Figure 6.2).

Soliciting Labels from the Crowd

StateLens builds upon the crowdsourcing workflow in VizLens [76], and uses a two-step workflow to label the area of the image that contains the interface assisted with screen detection results, and then label the individual interaction components assisted with OCR output (Figure 6.2). Crowd workers are first asked to rate the image quality, segment the interface region (with the generated screen bounding box as a start when available), indicate the approximate number of interaction components, and additionally provide a description of the interface state. Results are combined using majority vote.

Crowd workers are then instructed to provide labels to the individual interaction components (e.g., buttons) assisted with OCR output. Rather than requiring crowd workers to draw bounding boxes around all buttons and provide text annotations, the OCR-assisted crowd interface allows them to simply confirm or reject OCR-generated labels, and revise any errors. In this step, crowd workers also work in parallel, and the worker interface shows labeled elements to other workers as they are completed.

Recognizing User Interaction

Finally, StateLens captures the interaction component that triggered a state transition, e.g., a button b_n that contributes to the transition $E_{ij} = V_i \rightarrow V_j = (\{b_\mathrm{n}\}, V_i, V_j)$. Essentially, StateLens uses the last image of the previous state V_i before the state transition, transforms the input image to the reference image frame through warping, and detects the touchpoint location using skin color thresholding and other standard image processing techniques [173].

In the next section of Accessing the State Diagram, using the user interaction information, StateLens predicts the state that the interface could be transitioning to, and reduces the processing latency and errors by narrowing down the search space. Furthermore, StateLens aggregates these interaction traces to provide ranked usage suggestions to assist novice users. Note that recognizing finger touchpoint locations in naturalistic usage videos is not always possible or accurate, such as under extreme lighting conditions, or when users are wearing gloves. In those cases, StateLens will fallback to only using the state transition without the detailed interaction component as the triggering event, e.g., $E_{ij} = V_i \rightarrow V_j = (\emptyset, V_i, V_j)$.

6.5.2 Accessing the State Diagram

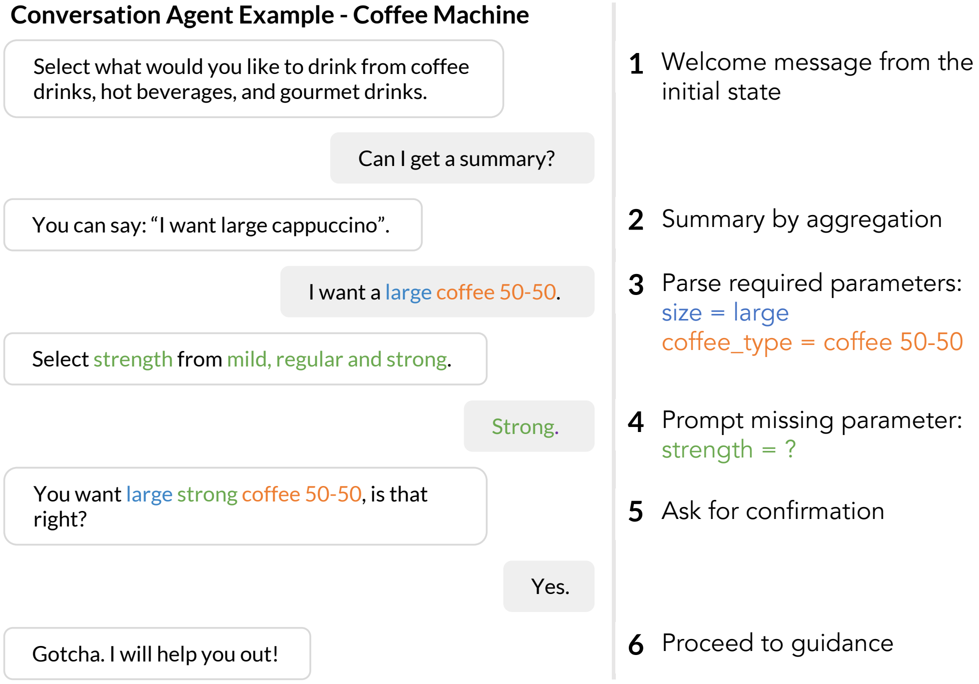
To help blind users access the dynamic interfaces, StateLens takes advantage of the state diagram to efficiently identify states, integrates natural language agents, and interactively provides feedback and guidance (Figure 6.1).

Identifying States Efficiently and Robustly

StateLens employs several techniques to enable efficient searching of states to reduce latency and prevent errors. First, when available, StateLens utilizes user's fingertip location to infer from the state diagram about the state that the interface has transitioned to, e.g., using the button that the finger was on. Second, StateLens searches the neighbors of previously identified state for the best match, in case when the inferred state from the fingertip location matches poorly with input image. Third, in case the matching results with neighbor states are poor, StateLens gradually expands the search space to other states of the interface according to the distance, calculated as the shortest path in the state diagram. Fourth, StateLens applies a similar approach to the candidate pool for smoothing, and only when a new state has been seen continuously across multiple frames, it is confident enough to determine a state transition. Finally, the reference images can be pre-computed once in advance to improve processing speed. These techniques effectively reduces the search space, speeds up the state detection process, and improves the robustness of state detection, which we will validate in technical evaluation. Note that for performance reasons, only SURF features are used when detecting states to provide real-time feedback for blind users. This is because the screen detection and OCR processes have longer delays (~1 second). However, in the future, these processes can be sped up and the produced bounding boxes can be tracked across frames to offer better performance.

Enabling Natural Language Queries

StateLens allows users to interact with a natural language conversational agent to prespecify the task they want to achieve. Inspired by our formative study, the goal of the conversational agent is to reduce the time and effort of the blind users to explore, understand, and activate functions on inaccessible and unfamiliar touchscreen interfaces. To do this, StateLens transforms all the possible paths (interaction traces) from S to T in the generated state diagram into different *intents* (e.g., to make coffee drinks, to make gourmet drinks), and the interactive element values in the edges E_i along the path into required *entities* for the intent and their attributes/values (e.g., size: large/medium/small). Using the Google Dialogflow API [67], StateLens automatically creates an agent for each device using these mappings. StateLens uses the description text from state S as the welcome prompt and adds confirmation prompts at the end of intents. StateLens heuristically generates training samples for the intents and prompts to the required entities from the descriptive texts along different paths aforementioned. Because Dialogflow only requires a small number of user utterance samples for training, StateLens uses a random sample of entity values and



Conversation Agent Example - Coffee Machine

Select what would you like to drink from coffee drinks, hot beverages, and gourmet drinks.

Can I get a summary?

You can say: "I want large cappuccino".

I want a large coffee 50-50.

Select strength from mild, regular and strong.

Strong.

You want large strong coffee 50-50, is that right?

Yes.

Gotcha. I will help you out!

1 Welcome message from the initial state

2 Summary by aggregation

3 Parse required parameters:
size = large
coffee_type = coffee 50-50

4 Prompt missing parameter:
strength = ?

5 Ask for confirmation

6 Proceed to guidance

Figure 6.5: Sample interactions between a user and the coffee machine natural language conversational agent StateLens automatically generated.

concatenates with phrases such as "Select ..." to create training sentences. The created agent then guides the user through each required parameter needed to complete an interaction trace. Once all required entities are fulfilled, the StateLens iOS application will proceed to guiding the users to activate each button on the predefined interaction trace. A sample user-agent interaction is shown in Figure 6.5.

Generating Natural Language Summary

StateLens uses the state diagram and the associated aggregation of interaction traces to automatically generate a natural language summary of the devices' popular use cases. This is designed to assist novice users get familiar with the device. To do this, StateLens ranks the aggregated interaction traces, then generates prompts for each trace based on the involved state and transition metadata as well as the corresponding interaction components. StateLens uses simple heuristic template-based generation methods that concatenate words like "I want ..." with most frequently selected button options, i.e. entities, as well as the descriptive text of the intent. This natural language summary is also integrated in the conversational agent (Figure 6.5), and users can simply ask, e.g., "tell me a summary."

Providing Interactive Feedback and Guidance

StateLens identifies the current state of the dynamic interface, and recognizes the user's touchpoint location to provide real-time feedback and guidance for blind users through the iOS application.

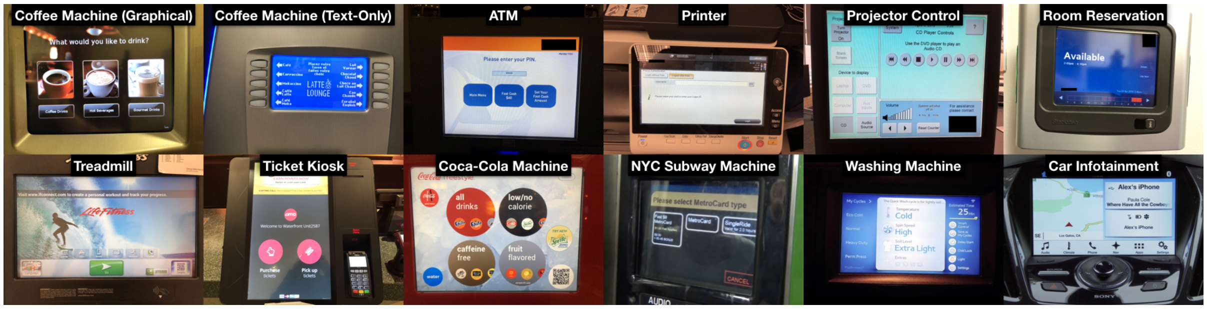


Figure 6.6: We evaluated how well StateLens reconstructs state diagrams from point-of-view usage videos across a wide range of interfaces, including an ATM, coffee machine (both graphical and text only), printer, projector control, room reservation panel, treadmill, ticket kiosk, Coca-Cola machine, subway ticket machine, washer, and car infotainment system.

For blind users accessing the interface with a 3D-printed accessory, a color marker on the accessory can be used to identify the touchpoint location. To make sure the touchpoint does not change from exploration to activation (i.e., the problems Slide Rule [102] addressed with split tap, and VizLens [76] addressed with shifting the interaction point), we measured the ground truth touchpoint location and placed the color marker on the accessory accordingly.

StateLens then looks up the coordinates of the touchpoint in the current state's labeled interaction components, and announces feedback and guidance to the blind user, e.g., "state: coffee drinks, select strength; target: regular", "move up", "move left slowly" and "at regular, press it." StateLens also provides feedback to users when the interface is partially out of frame by detecting whether the corners of the interface are inside the camera frame. If not, it provides feedback such as "move phone to right." Similarly, it provides feedback when it does not detect the interface or does not see a finger (using words or earcons [30] for "no object" or "no finger").

6.6 Technical Evaluation

We conducted a multi-part technical evaluation in order to understand how each key component of StateLens performs across a wide range of interfaces and usage scenarios.

6.6.1 Dataset

We collected a total of 28 videos from a diverse set of eight dynamic touchscreen interfaces, in different lighting conditions, and with both stationary and hand-held cameras, resulting in a total of 40,140 video frames. We also manually selected web videos of four touchscreen interfaces, resulting in a total of 32,610 video frames. All of these videos for our evaluation were collected by sighted people. The list of interfaces is shown in Figure 6.6, and summarized in Table 6.2.

6.6.2 Generating the State Diagram

We first evaluated the effectiveness of StateLens in reconstructing interface structures from stationary, hand-held, and web usage videos. After StateLens generated the states, researchers manually coded them as correct, missing, or redundant in order to calculate precision and recall. A high precision indicates that most of the extracted states are unique screens of the actual interface (few duplicates). A high recall indicates that most of the screens of the interface are captured in the extracted states (good coverage).

For each interface and video source, we computed the precision, recall, and F1 scores for the extracted states using four configurations of features: *(i)* SURF features only, *(ii)* Screen Detection and SURF features, *(iii)* SURF and OCR features, and *(iv)* Screen Detection, SURF, and OCR features. The results are shown in Table 6.2. Overall, the combination of Screen Detection+SURF+OCR features achieved high performances across a wide range of interfaces, and were often the best in the four feature configurations.

Regarding the effect of our screen detection approach, a combination of Screen Detection+SURF+OCR features generally yielded higher performance compared to SURF+OCR features. The advantages were mostly observed in the precision differences and especially for web videos, as irrelevant frames and noisy background were filtered out. The screen detection technique did not work well for the Coca-Cola machine, as the object detection model would not classify it as electronics or machines. To address this problem, special-purpose models for detecting screens could be built.

Regarding OCR features, a combination of Screen Detection+SURF+OCR features generally had better performance compared to Screen Detection+SURF features. The advantages were mostly observed in the recall differences, and specifically for interfaces that had many similar screens in graphical layout with only text changes, e.g., coffee machine (graphical), coffee machine (text-only), projector control, and room reservation interfaces. Regarding the different video sources, stationary videos generally performed better compared to hand-held ones for the same interface, because state matching is more robust with less camera blur, changing background noise and other uncertainty from camera motion.

Parameters can be chosen to further maximize recall (sacrificing some precision), as post-hoc crowd validation can be applied in the future to further filter out duplicates. Duplicate states require more manual effort to clean up, but have less impact on user experience compared to missing states.

6.6.3 Accessing the State Diagram

We next evaluated the effectiveness of using state diagrams to reduce latency and prevent errors in the state detection process.

Using State Diagram to Reduce Search Time

We evaluated the efficiency of our techniques in identifying states compared to the naive approach in VizLens::State Detection [76] which compares against every possible reference image. We varied the total number of states involved from one to all 14, and plotted the amount of processing

	Coffee Machine (G)				Coffee Machine (T)				ATM				Printer				Projector Control			
	Stationary		**Hand-held**		**Stationary**		**Hand-held**		**Stationary**		**Hand-held**		**Stationary**		**Hand-held**		**Stationary**		**Hand-held**	
# of states	14		13		11		10		11		12		10		10		13		9	
# of frames	4,980		2,580		3,060		2,310		2,910		2,340		1,380		1,980		3,540		1,530	
SURF	0.47	0.57	0.67	0.62	0.88	0.64	0.78	0.70	0.85	1.00	0.52	1.00	1.00	0.10	0.82	0.90	1.00	0.46	0.50	0.56
	0.52		0.64		0.74		0.74		0.92		0.69		0.18		0.86		0.63		0.53	
SD	0.58	0.50	0.73	0.62	1.00	0.64	0.80	0.80	0.69	1.00	0.63	1.00	1.00	0.20	0.82	0.90	0.86	0.46	0.63	0.56
+SURF	0.54		0.67		0.78		0.80		0.81		0.77		0.33		**0.86**		0.60		0.59	
SURF	0.72	0.93	0.65	0.85	0.73	1.00	0.67	1.00	1.00	1.00	0.75	1.00	1.00	0.40	0.63	1.00	1.00	0.54	0.67	0.67
+OCR	0.81		0.73		0.85		0.80		1.00		0.86		0.57		0.77		0.70		0.67	
SD+SURF	1.00	0.93	0.85	0.85	0.91	0.91	0.77	1.00	1.00	1.00	0.75	1.00	0.86	0.60	0.63	1.00	1.00	0.62	0.70	0.78
+OCR	**0.96**		**0.85**		**0.91**		**0.87**		**1.00**		**0.86**		**0.71**		0.77		**0.76**		**0.74**	

	Room Reservation				Treadmill				Ticket Kiosk				Coca-Cola		Subway		Washer		Car Control	
	Stationary		**Hand-held**		**Stationary**		**Hand-held**		**Stationary**		**Hand-held**		**Web**		**Web**		**Web**		**Web**	
# of states	7		8		10		10		11		14		9		16		11		24	
# of frames	1,560		1,260		1,260		4,500		1,470		3,480		2,100		6,630		5,940		17,940	
SURF	0.83	0.71	0.33	1.00	1.00	0.10	1.00	0.80	0.47	0.73	0.53	0.57	0.23	1.00	0.48	0.94	0.46	0.55	0.79	0.46
	0.77		0.50		0.18		**0.89**		0.57		0.55		0.37		0.64		0.50		0.58	
SD	0.86	0.86	0.50	0.75	1.00	0.10	1.00	0.50	0.58	0.64	0.64	0.50	0.33	0.67	0.71	0.94	0.78	0.64	0.73	0.67
+SURF	0.86		0.60		0.18		0.67		**0.61**		0.56		0.44		**0.81**		0.70		0.70	
SURF	0.54	1.00	0.39	0.88	0.75	0.60	0.83	0.50	0.50	0.73	0.60	0.64	0.20	0.89	0.47	1.00	0.53	0.82	0.65	0.83
+OCR	0.70		0.54		**0.67**		0.63		0.59		0.62		0.33		0.64		0.64		0.73	
SD+SURF	0.78	1.00	0.47	1.00	0.58	0.70	0.83	0.50	0.50	0.73	0.65	0.79	0.40	0.67	0.65	0.94	0.75	0.82	0.73	0.92
+OCR	**0.88**		**0.64**		0.64		0.63		0.59		**0.71**		**0.50**		0.77		**0.78**		**0.81**	

Table 6.2: Aggregated precision, recall, and F1 scores of the state diagram generation process of StateLens using a combination of features – Screen Detection (SD), SURF, and OCR – and with stationary, hand-held, and web (with links) usage videos. Each cell shows the precision (top left), recall (top right), and F1 scores (bottom). Bold values identify the feature combination with the best performance.

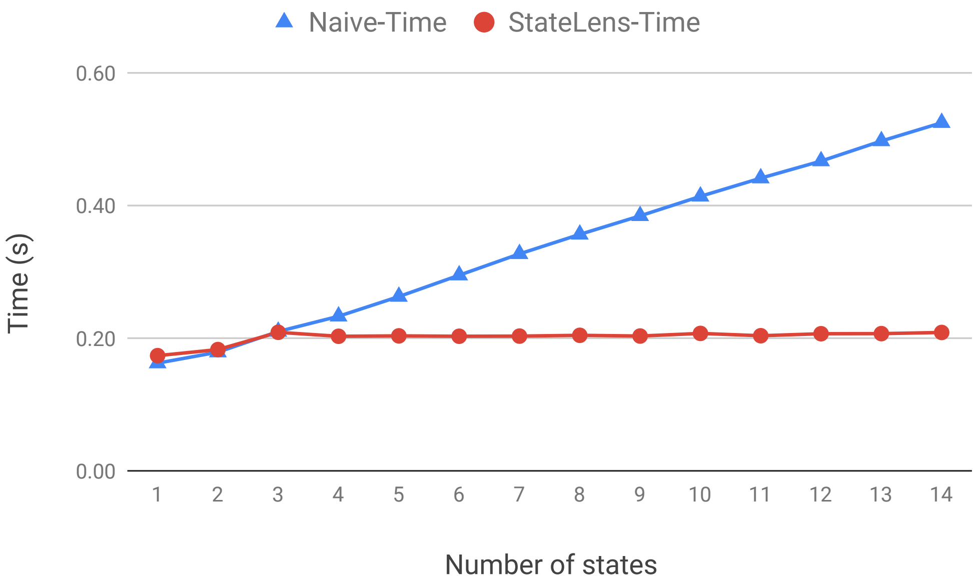


Figure 6.7: StateLens maintains a relatively stable processing time for state detection as the number of states increases, compared to the linear increase in the baseline approach.

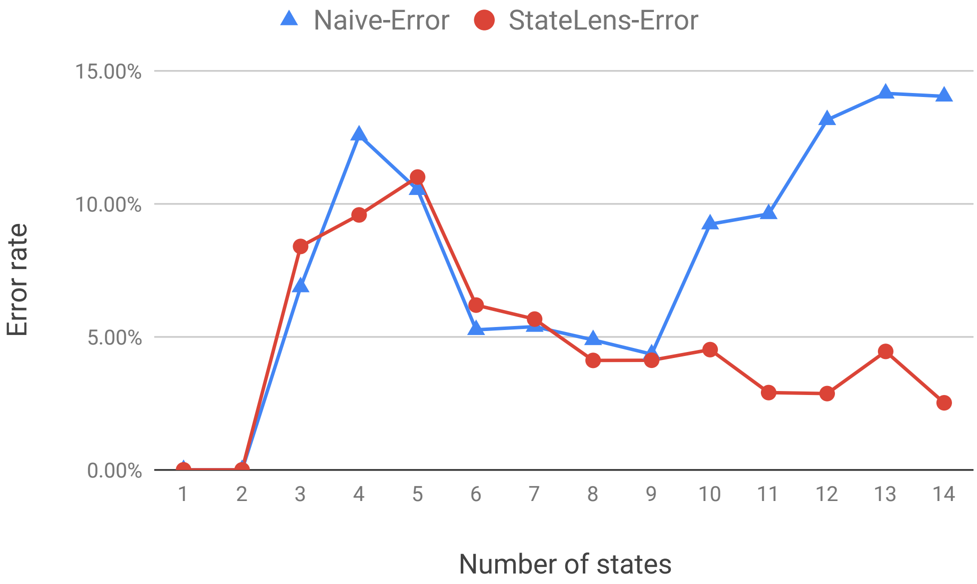


Figure 6.8: StateLens maintains a relatively stable error rate for state detection as the number of states increases, compared to the increasing trend in the baseline approach.

time required for identifying the current state. The results show that as the number of states increases, StateLens achieved a relatively stable processing time compared to the linear increase in the baseline approach (Figure 6.7). Furthermore, using the coffee machine with all 14 states, StateLens can still maintain sufficient speed for audio-guided interaction (~5fps), while the baseline approach dropped to ~2fps and became unusable.

ID	Gender	Age	Occupation	Vision Level	Hearing	Smartphone Use
P1	Female	64	Retired	Light perception, since 10	Normal	iPhone, 9 years
P2	Female	77	Retired	Light perception	Normal	iPhone, 2 years
P3	Female	34	Unemployed	Blind, since birth	Normal	iPhone, 6.5 years
P4	Female	46	AT consultant	Blind, since birth	Normal	iPhone, 5 years
P5	Male	43	IT consultant	Light/motion perception	Slight loss	iPhone, 3.5 years
P6	Male	67	Business Rep.	Blind, since birth	Normal	iPhone, 5.5 years
P7	Female	64	Retired	Blind, since birth	Mild loss	iPhone, 7.5 years
P8	Male	85	Retired	Blind, since 8 years old	Normal	No
P9	Female	37	AT Director	Light/shape perception	Normal	iPhone, 6 years
P10	Female	73	Retired	Blind, since birth	Normal	iPhone, 2 years
P11	Female	71	Retired	Blind, since childhood	Slight loss	iPhone, 7.5 years
P12	Male	71	Retired	Low vision (20/200), color blind	Normal	iPhone, 9 years
P13	Female	51	Unemployed	Blind, since birth	Moderate loss	iPhone, 8 years
P14	Male	71	Retired	Light perception	Slight loss	iPhone, 4 years

Table 6.3: Participant demographics for our user evaluation with 14 visually impaired users. Thirteen were blind, and one (P12) had low vision.

Using State Diagram to Reduce Search Error

We then evaluated the robustness of our techniques in identifying states compared to the baseline approach. We varied the total number of states involved from one to all 14, and plotted the percentage of errors in identifying the current state. The results show that as the number of states increases, StateLens achieved a relatively stable error rate of ~5% compared to the increasing trend in the baseline approach (Figure 6.8). Next in user evaluation, we further demonstrate how the generated state diagrams power interactive applications to assist blind users access existing dynamic touchscreen devices.

6.7 User Evaluation

The goal of our user study was to evaluate how the components of StateLens (the 3D-printed accessories, the conversational agent, and the iOS application) perform in enabling blind people to accomplish realistic tasks that involve otherwise inaccessible dynamic touchscreen interfaces.

6.7.1 Apparatus and Participants

In order to enable repeated testing without wasting coffee, we built a simulated interactive prototype of the coffee machine in Figure 6.4 with InVision [98], which we displayed on an iPad tablet of similar size as the coffee machine's interface (iPad Pro 3rd generation, 11-inch, running iOS 12.2 without VoiceOver enabled). The conversational agent and the iOS application were installed on an iPhone 6, running iOS 12.2 with VoiceOver enabled. The finger cap and the conductive stylus in Figure 6.3 were fabricated and used. We recruited 14 visually impaired users (9 female, 5 male, age 34-85). The demographics of our participants are shown in Table 6.3.

6.7.2 Procedure

Following a brief introduction of the study and demographic questions, participants first completed tasks using the 3D-printed accessories. For each of the three screen placements (in the order of 90° vertical at chest-level, 45° tilted at chest-level, and 0° flat on the table), participants completed five trials using both the finger cap and the conductive stylus. The order of accessories was counterbalanced for all participants. For each trial, participants were first instructed to *explore* by placing the accessory on the touchscreen and move according to the researcher's verbal instructions without activating touches. Participants were then asked to *activate* a touch. The number of accidental triggers during exploration, and the number of attempts during activation were recorded.

Next, participants were asked to talk to the conversational agent to prespecify drinks they want to order from the coffee machine for three times. Participants were instructed to order from a general category (e.g., coffee drinks), but can freely choose the other properties (e.g., coffee type, strength, size). Task completion rate and time were recorded.

Next, according to the three interaction traces prespecified through the conversational agent, participants were asked to use the 3D-printed accessories to perform the tasks following the guidance and feedback of the iOS application. These realistic tasks involved a series of button pushes across many states, e.g., select gourmet drinks, cafe latte, strong strength, then confirm, auto-select default coffee bean, and end on the drink preparation screen. The iPad Pro simulating the inaccessible coffee machine was placed tilted at chest level, and the iPhone 6 running the iOS application was mounted on a head strap to simulate a head-mounted camera. Task completion rate and time were recorded.

After each step of the study, we collected Likert scale ratings and subjective feedback from the participants. Finally, we ended the study with a semi-structured interview asking for the participant's comments and suggestions on the StateLens system. The study took about two hours and participants were each compensated for $50. The whole study was video and audio recorded for further analysis.

6.7.3 Results

We now detail our user study results and summarize user feedback and preferences. For all Likert scale questions, participants rated along a scale of 1 to 7, where 1 was extremely negative and 7 was extremely positive.

Exploration and Activation with 3D-Printed Accessories

All participants except P12 completed tasks using the 3D-printed accessories. P12 had low vision, and was able to hover his finger above the target and then activate by himself. The aggregated results are shown in Table 6.4. Using the conductive stylus to explore touchscreens

Screen Placement		**90 degrees**	**45 degrees**	**0 degree**
Stylus	Triggers	0 (0)	0.05 (0.21)	0.03 (0.17)
	Attempts	2.63 (1.13)	2.52 (1.08)	2.29 (0.99)
	Learnability	6.0 (0.9)	6.3 (0.9)	6.3 (0.9)
	Comfort	5.8 (1.4)	6.0 (1.1)	6.1 (0.9)
	Usefulness	6.0 (1.2)	6.3 (0.6)	6.5 (0.7)
	Satisfaction	5.8 (1.3)	6.5 (0.5)	6.7 (0.5)
Cap	Triggers	0.09 (0.34)	0.06 (0.24)	0.05 (0.21)
	Attempts	2.12 (1.10)	1.75 (0.95)	1.81 (0.96)
	Learnability	6.1 (0.6)	6.2 (1.2)	6.5 (0.7)
	Comfort	5.3 (1.4)	6.5 (0.7)	6.1 (0.9)
	Usefulness	6.3 (0.6)	6.6 (0.7)	6.5 (0.8)
	Satisfaction	6.2 (0.8)	6.6 (0.7)	6.5 (0.7)
Preference (S/C)		54% / 46%	38% / 62%	31% / 69%

Table 6.4: Results from the 3D-printed accessory study, showing mean and standard deviation (in parentheses).

generally resulted in fewer accidental triggers $(M = 0.03, SD = 0.16)$ compared to using the finger cap $(M = 0.07, SD = 0.27)$. On the other hand, the average attempts of using the stylus $(M = 2.48, SD = 1.07)$ was more than that from using the finger cap $(M = 1.90, SD = 1.01)$. This is likely because the conductive material is less sensitive compared to fingers.

In general, participants found both accessories to be comfortable to use $(M = 5.9, SD = 1.1)$ and highly useful $(M = 6.4, SD = 0.8)$. However, there were differences across the various screen placements. Participants slightly preferred using the stylus to explore and activate touchscreens in the 90° screen placement (54% vs. 46%), since holding the hand in the upright position using the finger cap was not as comfortable $(M = 5.3, SD = 1.4)$, and the stylus felt more natural. Others preferred the finger cap since it provided better control over the stylus. On the other hand, participants preferred the finger cap much more than the stylus (65% vs. 35%) in the 45° and 0° screen placements, since the finger cap became more comfortable to use in these positions $(M = 6.3, SD = 0.8)$.

We observed that participants sometimes held the accessories in awkward postures, likely due to unfamiliarity. This can be improved with practice, as participants generally found the accessories to be very easy to learn $(M = 6.2, SD = 0.9)$. Better affordances could further improve learnability as one participant (P14) noted that a conductive stylus design which incorporates a physical button to trigger, instead of a conductive region, would be beneficial.

Another interesting observation was that 8 of 13 participants who completed the tasks for the printed accessories would occasionally perform a “double-click”, or two taps in quick succession to activate the screen. Almost all of this subset of participants (7) had a strong familiarity with using VoiceOver on an iPhone or iPad, suggesting their habitual use of this technology may influence their interactions using the accessories.

Prespecifying Tasks with the Conversational Agent

Participants spent an average of 53.7 seconds ($SD = 11.6$) to prespecify tasks with the conversational agent, with an overall task completion rate of 100%, and found it to be extremely easy to learn ($M = 6.6, SD = 0.6$), comfortable to use ($M = 6.8, SD = 0.4$), and useful ($M = 6.7, SD = 0.6$). Several participants tried specifying multiple parameters in one sentence (e.g., I want a large coffee 50-50, shown in Figure 6.5). Note that the task completion time is likely to reduce in practice, since the agent's speaking rate is dependent on the users' screen reader setting, and after repeated usage, the users will get familiar with the functions.

Completing Realistic Tasks

Participants spent an average of 122.3 seconds ($SD = 41.9$) completing the first task, 110.4 seconds ($SD = 36.9$) for the second, the 97.6 seconds ($SD = 30.7$) for the third, as they got familiar with the audio feedback and guidance. The overall task completion rate was 94.7%. For five of the tasks, participants accidentally selected the wrong option and had to go back or start over. Because our smoothing approach requires a new state to be seen continuously across multiple frames in order to determine a state transition, there may be a delay in determining if a button press was successful. In this case, some users may accidentally press again at the same location triggering an incorrect selection on the next screen state. This issue may be alleviated by providing more immediate feedback such as a tentative audio confirmation that a button press has been successful.

In subjective ratings, participants found the StateLens iOS application to be easy to learn ($M = 5.5, SD = 0.9$), comfortable to use ($M = 5.6, SD = 1.2$), and very useful ($M = 6.1, SD = 1.1$). They felt the audio feedback provided by the app was in real-time and accurate ($M = 6.1, SD = 0.9$). Participants mentioned that the head mount can be made more comfortable using a lighter setup, e.g., glasses.

Overall, participants were very excited about the potential of StateLens, and felt that it could help them access other inaccessible interface in the future ($M = 6.6, SD = 0.9$):

> *It'll be a thing, I will actually use it.* (P1)
>
> *[StateLens] gives much more flexibility, so that if the machine itself doesn't have speech, this can cover the instances where you have to interact with a touchscreen. There are more tools to access them. This combination opens up more accessibility. ... I can't wait to see this in action!* (P6)
>
> *I really like the idea of using the phone to make screens accessible and give feedback in real time. That's really impressive. I would use it. It would be helpful and useful.* (P9)
>
> *I would welcome more opportunities to use interfaces with [StateLens], like operating the cable company box. It would be great if interfaces could also show up on my phone screen and read it to me or let me explore it there.* (P12)

A low vision user (P12) mentioned that even though he might not always need assistance, if the interface's contrast or brightness is poor, a system like StateLens would be greatly helpful as a confirmation. Furthermore, he would like to get more information beyond the text labels on the buttons by using StateLens as a cognitive assistant. He would find it useful if, for example, a button for a coffee selection labeled “Rainbow’s End” could further be described as “a coffee blend containing tasting notes of nuts and citrus” even though the display does not provide that information.

6.8 Discussion and Future Work

In this section, we discuss how the approaches used in StateLens might generalize to extract information from existing online videos to, for instance, assist sighted users and construct a queryable map of devices. We also discuss limitations of our work, which represent opportunities for future research.

6.8.1 Technical Approach to Accessibility

StateLens is not the ideal solution. In a perfect world, post-hoc fixes like StateLens would not be needed (because all technologies would be inherently accessible), but in practice access technology like StateLens plays a vital role. Even with the existing laws, there are still many cases where “reasonable accommodation” is not enough. For example, a vending machine could be labeled with Braille, but the checkout credit card machine is not accessible. StateLens is a stopgap measure to make access possible (as are many access technologies), and introduces ideas that might find purchase in other access and accessible technologies.

People who are blind were involved throughout the research, including several people with visual impairments on our extended research team, and multiple sessions of design and study with a total of 30 outside participants. While we strove to make this work self-contained, it builds on our long history of work involving thousands of blind people as students, researchers, participants, and users.

6.8.2 Generalizability

In this work, we developed a hybrid crowd-computer vision system to enable access to dynamic touchscreens in-the-wild. One unique contribution of this work is that we demonstrated the possibility of extracting state diagrams from existing point-of-view videos instead of screenshots or screencast videos [17, 114, 175]. For existing physical devices whose underlying hardware or software cannot be modified, point-of-view videos are more prevalent and easier to acquire compared to screencast videos, which makes our approach generalizable to a large variety of devices and scenarios.

We motivated our approach as a benefit to improve accessibility for blind users. However, this approach could be beneficial to sighted people and people with cognitive disabilities in many ways as well. For example, medical devices can be hard to configure, and devices that are in foreign languages are hard to operate. Through understanding of the state diagrams of devices with readily available or user-taken point-of-view videos, our approach can provide additional information to the user as they interact with the devices (e.g., augmented reality applications for translation services, and interactive tutorials such as those in [110]).

Using StateLens, we envision building a queryable map of state diagrams for many of the devices in the world using existing point-of-view videos that have been shared online. As users start to use a device, it can be geo-located, automatically recognized, or added into the system. Additional states can be added to the existing diagram as users interact with the device. Changes to the devices can be automatically detected over time to update the interface state diagram. Furthermore, similar but slightly different models of a device may reuse another state diagram and enable transfer learning.

6.8.3 Assistive Hardware for Automatic Screen Actuation

Our 3D-printed accessories elegantly add "risk-free exploration" to existing capacitive touchscreen devices without modifying the underlying hardware or software, which has been a major hurdle for past efforts. In our user study, we discovered issues around holding the accessories in certain angles, and "the last (centi-)meter" problem to accurately activate the exact button once. If the screen is cluttered, it could still be quite difficult to operate.

Furthermore, although in the user studies of StateLens and VizLens, participants were able to complete the tasks of using otherwise inaccessible interfaces, we observed that the time of completing realistic tasks involving a series of actions was still relatively long (~110s), especially compared to that in Facade (~17s). This is likely because, without the tactile feedback provided by the interface or the Facade-generated overlays, participants had to move their fingers very slowly to follow the audio instructions, and then activate interface elements when they were sure.

To address the above problems, we have started to design BrushLens, a smart assistive hardware proxy that can locate and actuate external interface controls automatically. Blind users could brush a "phone case" on the external touchscreen, then the built-in camera would capture, recognize, and instruct actuators contacting the external screen to trigger functions at the right place and time. This would further enhance the "risk-free exploration" user experience to existing physical interfaces without modifying the underlying hardware or software.

To achieve this, the hardware proxy will be designed to work with regular push buttons,

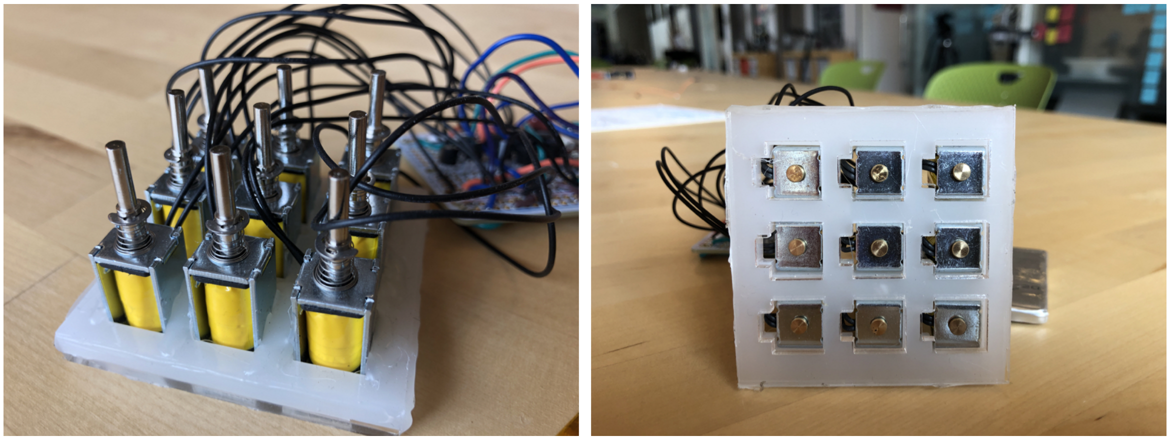


Figure 6.9: BrushLens is a smart assistive hardware proxy that would locate and actuate external interface controls automatically. Pictures show our first prototype built using solenoid actuators.

	Digital	**Piezoelectric**	**Pneumatic**	**Solenoid Only**	**Solenoid+Rubber**
Push button	No	No	Yes	Yes	Yes
Flat touchpad	No	No	Yes	Yes	Yes
Resistive touchscreen	No	Yes	Yes	Yes	Yes
Capacitive touchscreen	Some	Yes	Yes	Some	Yes
Mobile/Unibody	Yes	Yes	No	Yes	Yes

Table 6.5: Initial testing on several BrushLens hardware proxy designs. Using solenoid with conductive rubber seems to be the most promising solution.

flat touchpad buttons (like those in VizLens and Facade), resistive touchscreens, capacitive touchscreens (like those in StateLens), etc. The device itself should also be mobile and unibody. We have conducted initial testing on several designs, including using completely digital solutions, as well as using piezoelectric, pneumatic, or solenoid actuators. So far, our prototype built using solenoid with conductive rubber is the most promising (Table 6.5). Piezoelectric actuators are too expensive, and do not work well for cases that require high displacement. Pneumatic ones require bulky supporting hardware. The solenoid-only design works for most cases, but fails on some non-sensitive capacitive touchscreens. However, after adding conductive rubber on the actuator tips, the results seem to have improved.

Other components of the hardware will include a micro-controller, a hardware board hosting a grid of actuators, a BLE/WiFi connection module, a battery, and an outer casing that fits on users' mobile devices. The software running on the smartphone needs to take input from the user to prespecify tasks (using speech similar to that in StateLens, or storing sensitive information on device to preserve privacy), localize the position of the camera relative to the external touchscreen in real time, calculate the positions of the actuators relative to the intended button to press, send commands to the hardware component to actuate the right button at the right time, and repeat. For localizing the camera's position, we will investigate techniques such as real-time image stitching because the distance from the camera to the screen would be relatively small, thus limiting the field of view and increasing the difficulty of localization.

To evaluate the system, we will first conduct technical evaluations to understand the power consumption, and the effectiveness of the actuators' grid layout (e.g., using 1 vs. 3 vs. 9 pins). We will then conduct user studies with blind participants in the lab, with multiple device types, and heuristically compare with previous study results in VizLens and StateLens.

6.8.4 Limitations

As with most systems, StateLens currently has some limitations, which we believe could be explored in future work. For instance, StateLens has limited capability in noticing and differentiating minor interface changes such as toggle buttons or color indicators. One solution may be to detect and factor in UI widgets that are expected to change using approaches like those in PreFab [56] and TapShoe [160]. Furthermore, StateLens cannot currently handle major updates and layout changes of the interface, as well as list menus, slide bars or other gestures (e.g., scroll, swipe, pinch).

The completeness of the state diagram is limited by the coverage of the videos collected for the device. Even if videos only capture a subset of possible tasks, these would likely be frequently used paths of action, thus still providing reasonable functionality in many cases. If a blind user needs to access an unseen state, StateLens could add it to the state diagram on-the-fly, asking the user to wait for that screen to be labeled and then added to the full state diagram. Other approaches include generalizing based on the existing states or other machines, and relying more on OCR.

We evaluated StateLens across a number of touchscreen interfaces and with blind users in the lab, but we did not deeply study how StateLens works in the real world, which is often much more complicated and messier than in-lab studies. Our next step is to harden our implementation to scale to many users, and deploy it to understand how it performs in the everyday lives of blind people.

6.9 Conclusion

We have presented *StateLens*, a reverse engineering solution that makes existing dynamic touchscreens accessible. Using a hybrid crowd-computer vision pipeline, StateLens generates state diagrams about interface structures from point-of-view usage videos. Through these state diagrams, StateLens provides interactive feedback and guidance to help blind users prespecify task and access the touchscreen interfaces. A set of 3D-printed accessories enable capacitive touchscreens to be used non-visually by preventing accidental touches on the interface. Our formative study identified challenges and requirements, which informed the design and architecture of StateLens. Our evaluations demonstrated the feasibility of StateLens in accurately reconstructing the state diagram, identifying interface states, and giving effective feedback and guidance.

More generally, StateLens demonstrates the value in a hybrid, reciprocal relationship between humans and AI to collaboratively solve real-world, real-time accessibility problems. It takes advantage of different kinds of human intelligence: humans who provide access and collect videos at the interface to build up the training data, and online crowd workers who provide necessary labels to bootstrap automation. Then it relies on machine intelligence to generate the state diagram and the conversational agent to provide interactive guidance to the user.

By combining human and machine intelligence, I also explored environmental sensing platforms for understanding the visual world. In the next chapter (Chapter 7), I introduce *Zensors++*, a human-AI camera sensing system to answer natural language user questions based on camera streams. Zensors++ collects human labels to bootstrap automatic processes to answer real-world visual questions, allowing end users to actionalize AI in their everyday lives.

Chapter 7

Zensors++: Human-AI Camera Sensing in the Real World

Smart appliances with built-in cameras, such as the Nest Cam and Amazon Echo Look, are becoming pervasive. They hold the promise of bringing high fidelity, contextually rich sensing into our homes, workplaces and other environments. Despite recent and impressive advances, computer vision systems are still limited in the types of sensing questions they can answer, and more importantly, do not easily generalize across diverse human environments. In response, researchers have investigated hybrid human- and AI-powered methods that collect human labels to bootstrap automatic processes. However, deployments have been small and mostly confined to institutional settings, leaving open questions about the scalability and generality of the approach. In this work, we describe our iterative development of *Zensors++*, a full-stack human-AI camera-based sensing system that moves significantly beyond prior work in terms of scale, question diversity, accuracy, latency, and economic feasibility. We deployed Zensors++ in the wild, with real users, over many months and environments, generating 1.6 million answers for nearly 200 questions created by our participants, costing roughly 6/10ths of a cent per answer delivered. We share lessons learned, insights gleaned, and implications for future human-AI vision systems.

7.1 Introduction

Cameras are becoming pervasive in civic and commercial settings, and are moving into homes with devices like the Nest Cam and Amazon Echo Look. Owing to their high resolution and wide field of view, cameras are the ideal sensors to enable robust, wide-area detection of states and events without having to directly instrument objects and people. Despite this unique advantage, camera streams are rarely actionalized into sensor data, and instead are merely used to view a remote area.

This trend is slowly changing with consumer home cameras offering rudimentary computationally-enhanced functions, such as motion [54] and intruder detection [140]. Perhaps most sophisticated among these consumer offerings is the Amazon Echo Look, which can offer fashion advice [3]. In commercial and municipal camera systems [127], computer vision has been applied to e.g., count cars and people [49, 135], read license plates [50], control quality [172], analyze sports [18],

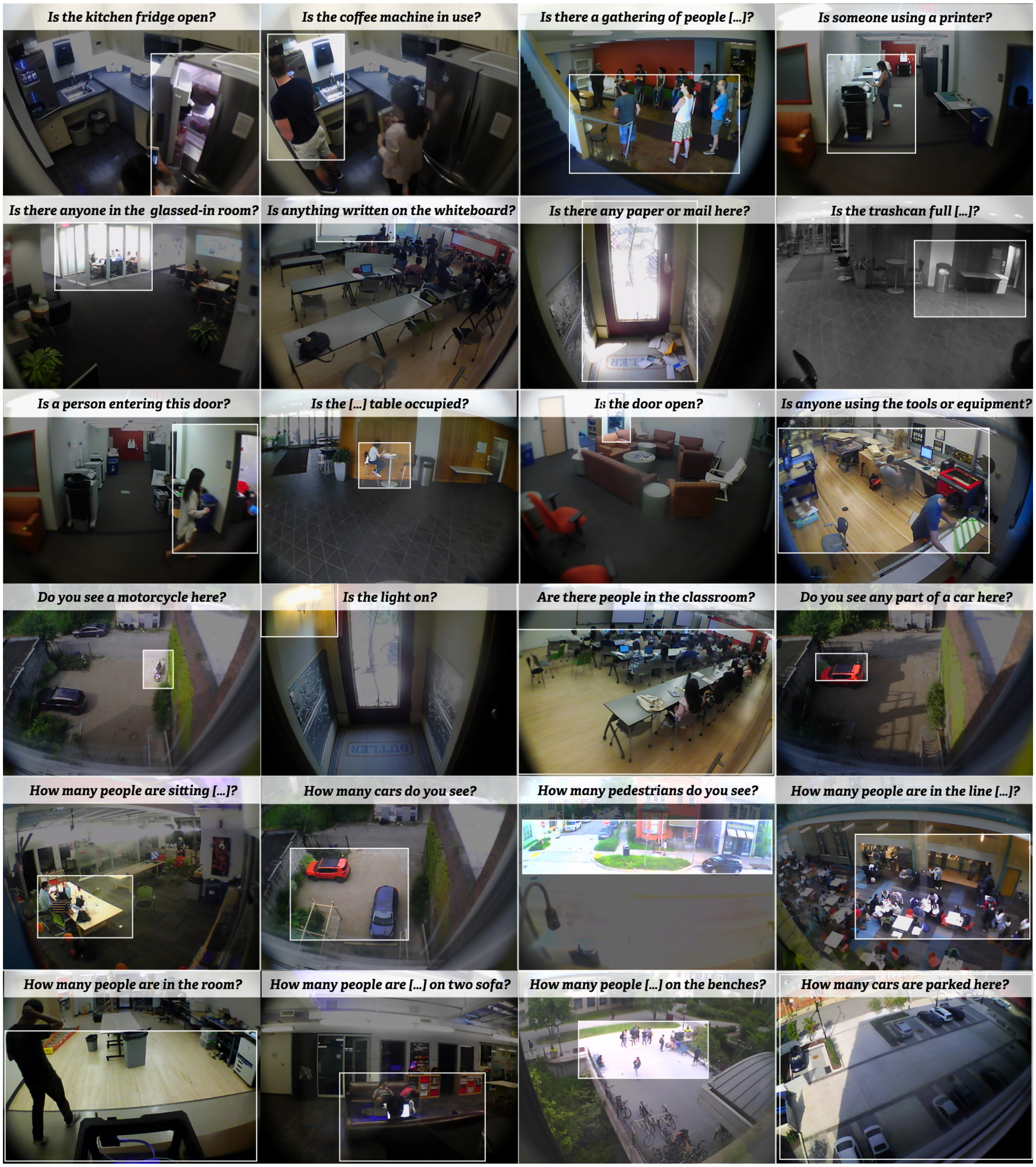

Is the kitchen fridge open?
Is the coffee machine in use?
Is there a gathering of people [...]?
Is someone using a printer?
Is there anyone in the glassed-in room?
Is anything written on the whiteboard?
Is there any paper or mail here?
Is the trashcan full [...]?
Is a person entering this door?
Is the [...] table occupied?
Is the door open?
Is anyone using the tools or equipment?
Do you see a motorcycle here?
Is the light on?
Are there people in the classroom?
Do you see any part of a car here?
How many people are sitting [...]?
How many cars do you see?
How many pedestrians do you see?
How many people are in the line [...]?
How many people are in the room?
How many people are [...] on two sofa?
How many people [...] on the benches?
How many cars are parked here?

Figure 7.1: Example question sensors created by our participants, with regions of interest highlighted on the full camera image.

recognize faces [166] and monitor road surfaces [59]. In general, these computer-vision-powered systems require extensive training data and on-site tuning to work well. For example, FaceNet [152] achieved human-level face detection accuracy, but required a team of researchers to collect and prepare over 100 million images for training. This is obviously impractical for the long-tailed distribution of scenarios and the many bespoke questions users may wish to ask about their environments [22, 35, 85].

To flexibly adapt to new questions, researchers have created hybrid human- and artificial intelligence (AI)-powered computer vision systems [48, 76, 88, 116]. Rather than requiring an existing corpus of labeled training data, these systems build one on-the-fly, using crowd workers to label data until classifiers can take over. This hybrid approach is highly versatile, able to support a wide range of end user questions, and can start providing real-time answers within seconds [116]. However, prior work falls short of real-world deployment, leaving significant questions about the feasibility of such human-AI approaches, both in terms of robustness and cost. Moreover, it is unclear how users feel about such systems in practice, what questions they would formulate, as well as what errors and challenges emerge. We focus on four main research questions:

- RQ1 (System): What system components and architecture are needed to support human-AI camera sensing in real-time and at scale?
- RQ2 (Performance): What is the accuracy, latency, cost, and automation that can be achieved in real world deployments?
- RQ3 (Applications): How do end users apply human-AI camera sensing in their domestic and work lives?
- RQ4 (Qualitative): What are the perceived value and privacy trade-offs?

To investigate these questions, we iteratively built *Zensors++*, a full-stack human-AI camera-based sensing system with the requisite scalability and robustness to serve real-time answers to participants, in uncontrolled settings, over many months of continuous operation. With an early prototype of the system, we performed a discovery deployment with 13 users over 10 weeks to identify scalability problems and pinpoint design issues. Learning from successes and failures, we developed an improved system architecture and feature set, moving significantly beyond prior systems (including Zensors [116], VizWiz [26] and VizLens [76]). More specifically, Zensors++ makes the following technical advances: *(i)* multiple queues to support crowd voting and dynamic worker recruitment, *(ii)* a dynamic task pool that estimates the capacity of labelers for minimizing end-to-end latency, and *(iii)* a hybrid labeling workflow that uses crowd labels, perceptual hashing, and continuously-evolving machine learning models. With our final system, we conducted a second deployment with 17 participants, who created 63 question sensors of interest to them (24 of which are illustrated in Figure 7.1). This study ran for four weeks, resulting in 937,228 labeled sensor question instances (i.e., answers). We investigated the types and sources of errors from e.g., crowd labeling, user-defined questions, and machine learning classifiers. These errors were often interconnected, e.g., when users created questions that were difficult for crowd workers to answer, workers were more likely to answer incorrectly, which in turn provided poor training data for machine learning, ultimately leading to incorrect automated answers. Overall, this experience illuminated new challenges and opportunities in human-AI camera-based sensing. We synthesize our findings, which we hope will inform future work in this area.

7.2 Related Work

Our work is related to several areas in HCI, AI and crowdsourcing, including environmental sensing, computer vision and crowd-powered systems. We now review the most relevant work.

7.2.1 Environment Sensing

There is a significant literature in HCI and Ubicomp that has outfitted homes, offices, public environments and objects with sensors to detect various activities. Approaches for instrumentation vary both in density and the classes of activities being monitored.

Most straightforward is special-purpose sensing, wherein a single specialized sensor is used to monitor a single facet of an environment. For example, systems such as UpStream [112] and WaterBot [16] instrument a faucet with an acoustic sensor to monitor water consumption. Similar special-purpose approaches have been applied to HVAC systems using room-level temperature [108] and occupancy [153] sensors. Special-purpose sensors tend to be robust for well-defined, low-dimensional sensing problems, but are difficult to generalize.

Alternatively, a network of sensors (i.e., a distributed sensing system) can be used to offer added generality by enlarging the sensed area (e.g., occupancy sensing across a building) or by increasing fidelity through multiple reinforced readings (e.g., detecting earthquakes using an array of sensors). These systems can be homogeneous (e.g., many cameras) or heterogeneous (i.e., a mix of sensor types) [161, 177]. However, large numbers of sensors, needed to obtain good coverage and accuracy, can carry a substantial financial, social and aesthetic cost.

To reduce the deployment cost of distributed sensing systems, researchers have explored infrastructure-mediated sensing (e.g., powerlines [83, 180], plumbing [64], HVACs [147]), wherein a single sensor can detect an environmental facet across a large context. Although more "universal" than the aforementioned approaches, infrastructure-mediated sensing is still constrained by the class of infrastructure to which it is attached. Most closely related to our approach is the notion of general-purpose sensing [116, 117], where a single, highly-capable sensor can detect a wide range of events within a room.

7.2.2 Computer Vision and Crowd-Powered Systems

Computer vision has come closest to achieving general-purpose sensing, as cameras offer high-fidelity data that can be processed to yield sensor-like feeds. However, achieving human-level abstractions and accuracy is a persistent challenge, leading to the creation of computer vision and crowd-powered systems (e.g., [26, 76, 120]).

Crowdsourcing systems access "human intelligence" through online marketplaces such as Amazon Mechanical Turk [4]. Prior work in general-purpose, visual sensing has relied entirely on crowdsourced answers. For instance, VizWiz [26] had crowd workers answer visual questions from blind users using photos taken from a mobile phone. The same mechanism has been applied to automate critical tasks in other domains, including managing dialogue [94, 96, 121, 122] and promoting accessibility [76, 78, 88]. Unlike VizWiz and OMoby [129], we focus on stationary cameras that answer human-defined questions over time, providing a continuous sensor feed.

Researchers have also mixed computer vision and crowd-powered approaches to create systems that learn over time. For example, Legion:AR uses on-demand crowd labeling to train an HMM-based activity recognizer [120]. Likewise, Flock [48] trains hybrid crowd-machine learning classifiers to enable fast prototyping of machine learning models that can improve on both algorithmic performance and human judgment, accomplishing tasks where automated feature extraction is not yet feasible. VATIC [174] uses crowd workers to annotate video with labels and object bounding boxes, providing critical training data to bootstrap machine learning.

Finally, this work is most directly related to Zensors [116], our original system, which shares the same main concept of using cameras and crowds to power end-user-authorable sensor feeds. In this work, we move from proof of concept to large-scale deployment, coupled with comprehensive analyses to more deeply assess the feasibility of human-AI visual sensing systems.

7.3 Discovery Deployment

Building on prior work, we created an initial, minimally-viable system comprised of *(i)* a scalable backend, *(ii)* a web-based, user-facing, question authoring interface, and *(iii)* a labeling interface for crowd workers. The system also included *(iv)* an administrative interface for managing connected cameras and user accounts. Rather than describing the many intermediate versions of Zensors++ created over many months of development, we instead detail the final system in the next section. Here we briefly describe the Zensors++ interface as experienced by end users.

To initialize a “question sensor”, users must first place a networked camera (e.g., WiFi, PoE) in an environment of interest. Once the camera is in place and online, users can bind the camera to Zensors++. Through our web interface, a user can select the camera from a personal list, highlight a region of interest on the camera’s image, and ask a natural language question, e.g., “is the trashcan full?” The frequency at which the question sensor runs is also specified (e.g., every five minutes). This completes the question sensor creation process. At the specified interval, individual “question sensor instances” are processed by a backend. Initially “answers” are provided by crowd workers, using majority voting for basic quality control. Once answers are decided, they are forwarded to end-user applications. For example, users can setup notifications, e.g., send text message if “is the trashcan full?” equals yes). Answer streams are also viewable through a web-based visualizer (e.g., trashcan utilization over a month). Running exclusively on crowd power is not feasible long term, so Zensors++ caches all crowd labels to serve as a corpus for training computer-vision based machine learning classifiers, which take over when confident.

We used this initial system as a vehicle to run a “discovery deployment”, which ran for 10 weeks with 13 users, who created a total of 129 question sensors. Participants could select from cameras we set up at our institution, and were also offered cameras to set up themselves (e.g., at their home or office). Participants were given a short tutorial explaining how to use the system. During this pilot deployment, we prompted participants by email to log into the system every few days, iterate on their questions if need be, and report any bugs, issues or feature requests. In this discovery deployment, the primary goal was to understand the challenges in deploying such a system (RQ1) and assessing its performance (RQ2; e.g., accuracy, scalability, automation). We also gathered initial user feedback (RQ3; e.g., question types, quality of answers, error modes). In total, the system delivered 661,090 answers to users over the ten-week period.

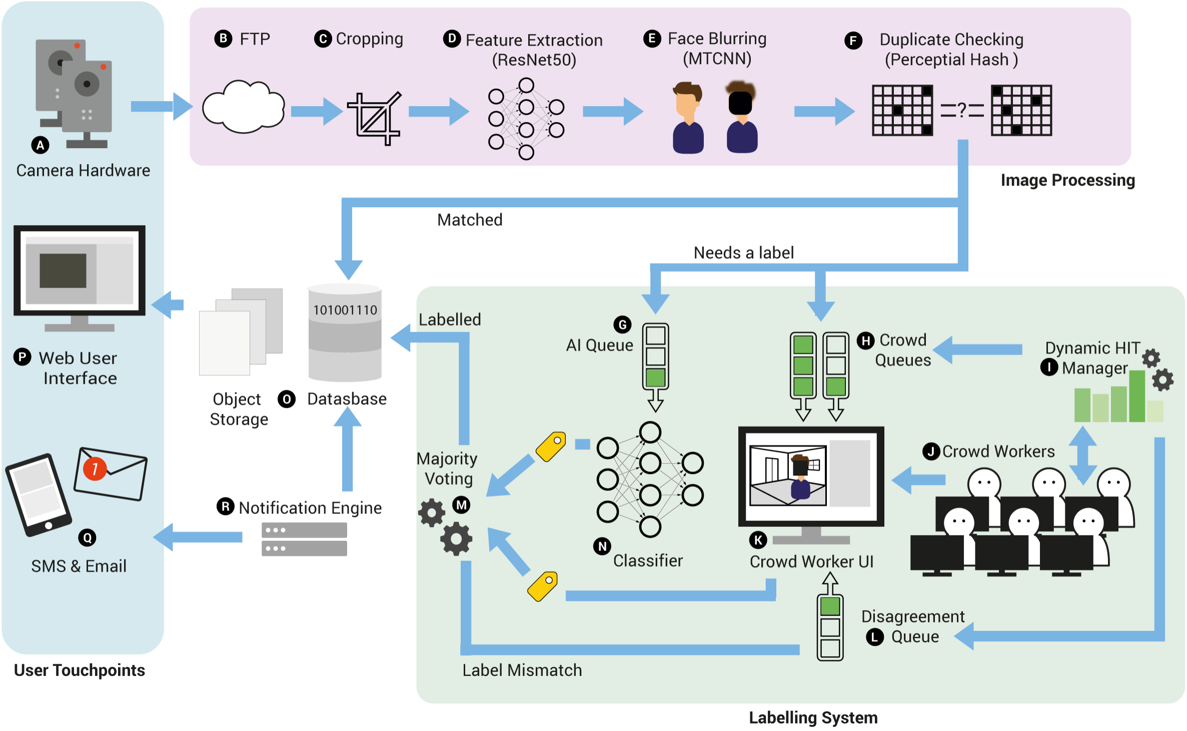


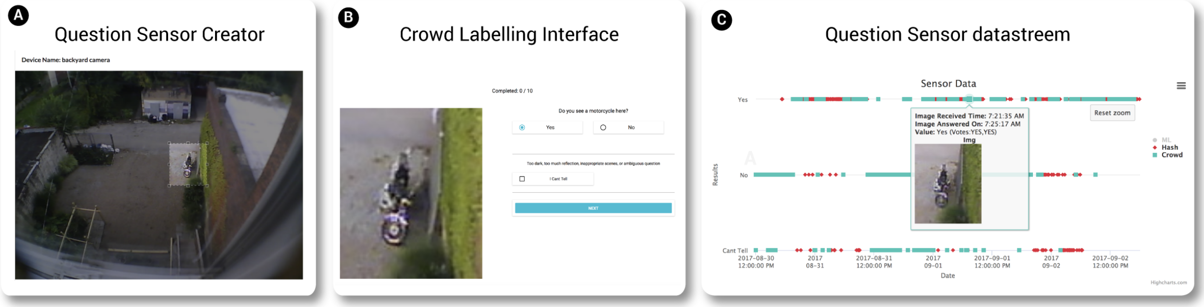
Figure 7.2: The system architecture of Zensors++ consists of four main components: user touchpoints (blue), image processing pipeline (pink), labeling system (green), and data storage (grey).

7.4 Zensors++

Our discovery deployment allowed us to iterate and strengthen our system design and implementation, which we now describe in greater detail. We focus on distinguishing features, rather than covering every technical facet.

7.4.1 Compute

We deployed Zensors++ on Amazon Web Services (AWS) [5], implemented as a stateless Django [57] application. Six vCPUs and 24 GiB of RAM were used for hosting the web user interface and crowd labeling system, which sat behind an Elastic Load Balancer (ELB) [7] accelerated by ElastiCache [6]. Another vCPU and two GiB of RAM was used for hosting an File Transfer Protocol (FTP) gateway. Forty vCPUs and 276 GiB of RAM were used for image processing and machine learning. Two vCPUs and 8 GiB of RAM were used for system administration and monitoring. In total, our processes consumed 49 vCPU cores and 310 GiBs of memory.



Device Name: backyard camera

Completed: 0 / 10

Do you see a motorcycle here?

● Yes ○ No

Too dark, too much reflection, inappropriate scenes, or ambiguous question

□ I Cant Tell

NEXT

Image Received Time: 7:21:35 AM
Image Answered On: 7:25:17 AM
Value: Yes (Votes:YES,YES)
Img

Figure 7.3: Screenshots of user interfaces. (A) End users highlight a region of interest in a camera image and add an associated natural language question. (B) Crowd workers provide answers for user-defined "question sensors" given an image region of interest. (C) End users can view real-time and historical data through visualizations.

7.4.2 Cameras

Zensors++ was designed to work with most off-the-shelf camera systems capable of posting data to a network service. For our study, we selected D-Link DCS 932L Wireless Day/Night cameras [53] (Figure 7.2A), available at many stores for about $30 USD, offering 640 × 480px video with WiFi connectivity. These cameras upload data via FTP to the Zensors++ backend, implemented in python using the pyftpdlib library (Figure 7.2B). To register a new camera, our web frontend generates unique FTP credentials that bind a camera to a user's account.

Although the D-Link cameras were versatile and inexpensive, we encountered some limitations. First, some users placed cameras facing out of windows, causing the camera's infrared (IR) LEDs to reflect off the glass and obscure images in dark contexts. As a simple solution, we disabled the IR LEDs by covering them with electrical tape. The cameras also had a limited field of view, which sometimes made it difficult to ask questions across an entire room. To emulate more expensive devices, we fitted our cameras with inexpensive, clip-on ($3) fisheye lenses (intended for smartphones).

7.4.3 Question Sensor Authoring

Our web interface (Figure 7.2P) allows users to manage question sensors and share camera feeds with other users. To create a question sensor, a user selects a camera from a personal list (e.g., backyard camera) and then drags a bounding box (Figure 7.3A) to select a region of interest. Next, the user specifies the question they want answered (e.g., "do you see a motorcycle here?"). They can also modify properties, such as the question type (e.g., Yes/No) and the sensing frequency (e.g., every 5 minutes). Once launched, the user can edit, pause, resume, or delete the question sensor at any time. Users can also view all of their question sensors' live data, and explore a visualization of historical data to perform retrospective analysis and sensemaking (Figure 7.3C).

7.4.4 Notifications

We learned early in the deployment that users were interested in alerts, especially for infrequent events, e.g., "is there a package on my doorstep?" To enable this for our participants, we created a web-based query builder that allows question sensors to be combined into logic statements with thresholds to trigger notifications. Upon deployment, we quickly discovered that any incorrectly labelled question sensor instance (e.g., false positive) would trigger a false notification, endangering user trust in the system's accuracy. To mitigate this issue, we incorporated *hysteresis*; the system only triggers a notification if it finds three or more continuous identical labels for one state followed by three or more labels of another state. This approach protects against single errant answers and ensures notifications have a high probability of being correct. However, this mechanism fails for short-lived events, such as "is a person entering this door?"

Our notification system uses the Redis [9] in-memory data structure store on AWS to cache the recent data points for scalability. We use Simple Email Service (SES) [11] for sending email notifications, and Simple Notification Service (SNS) [12] for sending text messages (Figure 7.2Q).

7.4.5 Privacy Preservation

Since images of public and private places are shown to online crowd workers, it is important to de-identify personal information to protect privacy. We implemented several mechanisms to mitigate this significant issue. First, the cameras infrequently transmit low-quality images (640×480px) to our secure server. No video or audio is captured. Second, we apply a state-of-the-art face detection algorithm [181] to obscure faces with a black box (Figure 7.2E). Third, only small regions of interest are shown to crowd workers, allowing users to selectively reveal areas of their environment (Figure 7.2C).

7.4.6 Redundant Images

Using fixed-view cameras that sampled as frequently as every 10 seconds meant that many images transmitted to Zensors++ were redundant. This was especially true at night, when human activity decreases. To reduce labeling cost, we use a perceptual image hashing algorithm [179] (64-bit) to compare each incoming image to previously labeled images for that region of interest. If a match is found, we can simply co-opt an earlier human label for essentially zero cost (Figure 7.2F).

During our discovery deployment, we tuned two factors in our image comparison pipeline to achieve a balance between labeling cost and accuracy. The first parameter was the distance threshold between two image hashes. Through repeated testing of different bit distances, we ultimately selected a threshold of zero (i.e, the two perceptual hashes had to be identical bit-wise to be considered a match). The second parameter we optimized was the "look-back period". At first, we thought utilizing the entire history of a question sensor would be ideal, but we found that early answers tended to infill the hash space and prevent newer answers from being added to the corpus. Moreover, if an early answer happened to be wrong, it could be copied forward innumerable times with no way to recover. To counter these effects, we selected a look-back period of 48 hours. In the future, these parameters could be automatically adjusted using periodic validation from the crowd.

7.4.7 Crowd Interface

High quality crowd-generated labels are crucial to the end user experience and future machine learning automation. Our initial interface was specifically optimized for speed, heavily relying on keyboard shortcuts. However, from user testing, we found that workers often fell into a rhythm and pressed the wrong key when a different image appeared after a sequence of similar question sensors in a row, resulting in a non-trivial number of incorrect labels. In response, we redesigned the crowd interface to have large buttons and no keyboard shortcuts (Figure 7.3B), which forced a more deliberate selection.

Our crowd interface was implemented using the React JavaScript library [60] (Figure 7.2K) and we recruited workers from Amazon Mechanical Turk (Figure 7.2J), paying one cent for each answer in batches of ten. Labeling each question sensor instance took roughly three seconds, resulting in an hourly pay of approximately ~$10/hour.

7.4.8 Crowd Disagreement

Rather than relying on a single crowd worker to provide an answer for a question sensor instance, Zensors++ sources several answers which it fuses together to produce a more reliable and final "answer of record" (which manifests in e.g., end user visualizations and triggers notifications). We used a simple majority voting mechanism (Figure 7.2M). First we solicit two crowd answers; if they match (e.g., yes & yes) the final answer is recorded. If the answers differ (e.g., no & yes), we tie break by soliciting a third and final answer (e.g., no & yes & no = no). In the case of count questions, we take the median answer (e.g., 12 & 24 & 14 = 14).

To efficiently dispatch question sensor instances to crowd workers, Zensors++ uses message queues [8]. Question sensor instances are pulled in batches of ten to form a single Human Intelligence Task (HIT). In early versions of the system, it was possible for a single crowd worker to answer the same question sensor instance several times over multiple HITs, undermining multi-worker majority voting. To ensure that unique workers cast votes, we switched to a three-queue setup implemented using Simple Queue Service (SQS) [13]. A question sensor instance is first inserted into two "work queues" (Figure 7.2H) and then onto a third "disagreement queue" (Figure 7.2L) if the first two answers do not match. To ensure unique workers for majority voting and prevent queue starvation, the system pins a worker to a queue for a period of one hour. This practically eliminates the possibility of having multiple votes from the same worker for a question sensor instance.

7.4.9 HIT Queues

There is an inherent delay that exists between when a HIT is created and when it becomes available for crowd workers, as reported in [95]. Rather than waiting for images to be ready before firing HITs, we apply a dynamic task pool approach that estimates the capacity of labelers for minimizing latency (Figure 7.2I). We periodically query the system for the number of queued images, currently open HITs and newly generated question sensor instances, which are then all used as inputs to a weighted formula to determine the number of new HITs to spawn. Aside from the three worker queues, we also implemented an "expired queue" for images that have grown stale (and it is better to answer a more recent instance of a question sensor) and a "hash-hit queue" for images not needing human labels. These two queues are used as retainers [21, 26] in the event that the main queues run out of items or in cases of surplus labeler capacity.

7.4.10 Crowd Reliability

In our discovery deployment, we only recruited crowd workers with greater than 95% assignment approval rate. However, the repetitive nature of the work and constant availability of HITs lead to degraded worker attention. There were also several malicious workers who exploited our early system's limited safeguards during our discovery deployment. To partially mitigate this issue, we created a set of gold standard question sensor instances, which we randomly inserted into HITs. If a worker provided an incorrect answer, a warning pop-up appeared. As we will discuss later in Results, even this intervention was insufficient.

7.4.11 Machine Learning

Our machine learning pipeline (Figure 7.2G) was built to automatically create classifiers specifically tuned for each end user's question and region of interest. The process starts by taking region of interest images and computing a 2048-dimension embedding from ResNet-50 [89] (Figure 7.2D). This feature vector is stored for training and also used as input to a classifier (Figure 7.2N) that predicts an answer for a question sensor instance. These embeddings serve as input to a KNN [51] classifier for yes/no questions and a KNN regressor for count questions.

7.4.12 Promoting Good Question Sensors

In our discovery deployment, we found the single greatest source of sensing inaccuracy was not from crowd workers or machine learning, but rather end users themselves. We identified four distinct failure modes:

Poor image cropping (C): Creating a question sensor requires selection of a region of interest on a camera's feed. We found many cases where the selected region included too many distracting elements that were irrelevant to the question being asked, adding cognitive overhead for crowd workers and noise for machine learning. We also found instances where the cropped region was too tight, and only partially captured the question's subject.

Ambiguous language (L): Users enter questions as free-form, natural language text. As a result, the system encountered questions that were subjective and/or included ambiguous language difficult for crowd workers to interpret. For example, "is the table messy?" is a matter of personal threshold. Variability in answers from different crowd workers meant unpredictable data and difficulty in converging machine learning models.

Missing context (X): Users often have context about their environment and created questions that required additional knowledge. For example,"is the coffee machine in use?" assumes a crowd worker knows what a machine's screen looks like when its in use vs. idle. A context free framing could be: "Is there a person in front of the coffee machine?"

Poor image quality (Q): Finally, for some cases image quality was insufficient to enable reliable answers. Sometimes this was because the camera was too far away, reducing the effective resolution, or the change was too subtle. For example, “is it raining now?” was very hard to answer using our cameras.

We used these early findings to create an onboarding guide for our final system, which promotes the creation of more effective question sensors. We also integrated an “I can’t tell” button into the crowd interface as a mechanism to flag questions that were challenging to answer, prompting users to rephrase. We use the above categorizations (C, L, X and Q) in Figure 7.4.

7.5 Evaluation Deployment

Our discovery deployment illuminated many technical issues (RQ1), the most interesting of which we have described above. To more fully answer RQ2 (performance), RQ3 (applications) and RQ4 (qualitative feedback), we conducted a second, large-scale deployment with our final Zensors++ implementation. We recruited 17 new individuals (mean age 35, six female) through mailing lists and flyers. Their occupations included department and program directors, administrative coordinators, facility and lab managers, professors and students. Participants were given a tutorial on how to use the system and our onboarding guide on how to create good question sensors. Participants then created accounts on our web interface (Figure 7.3), and set up new or selected existing cameras that were of interest to them. After a brief brainstorming exercise with the experimenter serving as a facilitator, participants proceeded to create question sensors of their choosing.

During deployment, participants were encouraged to log in to their accounts, view real-time data, as well as explore historical data with the visualization tool. We also scheduled a midpoint check-in with our participants to gather initial feedback, and rectify any errors, perceived or actual. If participants were satisfied with their question sensors, we enabled the notification feature, which allowed them to receive emails or text messages based on simple triggers. At the conclusion of the deployment, we ran a formal exit interview, paying special attention to motivations around the different sensors they created and the value they derived.

The deployment ran for four weeks, powering 63 participant-created question sensors across a variety of locations, including homes, offices, labs, cafes, food courts, parking lots, classrooms, workshops, and shared kitchens. Figure 7.1 provides visual examples of 24 participant-defined question sensors; Figure 7.4 provides a compact, but full listing.

7.6 Results & Discussion

Across 63 question sensors powered for four weeks, Zensors++ achieved an average accuracy of ~80% for yes/no questions, and was within 0.2 units (e.g., people, cars) on average for count questions. 74.4% of images had hash hits, which means 25.6% of images went to the crowd for labeling. This resulted in an average crowd cost of 6/10ths of a cent per answer delivered. We now describe these results in greater detail, including implications for future systems.

Type	Question	Sampling Freq. (sec)	# Total Instances Captured	# Crowd Labels	Mean Labelling Time Sec (SD)	% Hash Rate	Cost / Day	% Crowd Labeled "Can't Tell"	ML Acc. (after 1 month)	Accuracy	Acc. (malicious 10% dropped)	Error Type	Proxy Q.
Yes/No	Do you see a motorcycle here?	300	8224	11030	4.84 (4.43)	45%	$3.94	16%	90%	96%	98%	–	P
Yes/No	Is the light on?	1200	2132	2960	5.21 (4.81)	45%	$1.06	15%	95%	91%	98%	–	P
Yes/No	Are there people in the classroom?	300	5100	3182	5.84 (5.07)	75%	$1.14	8%	78%	87%	97%	–	–
Yes/No	Is the door open?	60	37835	27418	5.00 (4.41)	73%	$9.79	9%	91%	92%	97%	–	–
Yes/No	This is a big classroom. Is there more than 3 people in this room?	300	8430	6315	6.42 (5.75)	72%	$2.26	9%	49%	85%	94%	–	–
Yes/No	Do you see any part of a car here?	1200	2134	4327	5.37 (4.73)	20%	$1.55	28%	90%	88%	93%	–	–
Yes/No	Is anything written on the whiteboard?	3600	736	1657	5.90 (5.05)	15%	$0.59	30%	84%	84%	93%	Q	–
Yes/No	Is a person entering this door?	10	147588	22472	5.33 (4.85)	96%	$8.03	10%	71%	84%	93%	L	P
Yes/No	is the the table occupied?	1200	2145	3168	5.31 (4.82)	42%	$1.13	14%	68%	85%	93%	–	–
Yes/No	Is the kitchen fridge open?	60	10523	3691	4.89 (4.23)	87%	$1.32	7%	81%	81%	93%	–	–
Yes/No	Is the door open?	10	34385	24888	4.77 (4.23)	75%	$8.89	18%	79%	89%	91%	–	–
Yes/No	Is someone sitting on any of this furniture?	60	10975	4693	5.77 (4.78)	86%	$1.68	9%	62%	89%	91%	–	–
Yes/No	Is anyone using the tools or equipment?	300	8205	10471	5.81 (4.88)	52%	$3.74	13%	74%	76%	91%	–	–
Yes/No	Is there a gathering of people in the lobby?	3600	743	1372	6.74 (6.17)	29%	$0.49	13%	73%	83%	91%	L	–
Yes/No	Is this seat occupied?	300	8228	10626	5.11 (4.49)	49%	$3.80	12%	77%	85%	90%	–	–
Yes/No	is the trash can overflowing?	3600	740	1280	5.63 (5.03)	35%	$0.46	22%	62%	74%	89%	–	–
Yes/No	Do you see any part of a car here?	300	8055	15495	5.10 (4.54)	26%	$5.53	36%	50%	80%	85%	–	–
Yes/No	The basement can flood , do you dry ground or water here ?	1200	2251	659	8.16 (7.65)	90%	$0.24	22%	63%	68%	84%	L	–
Yes/No	Is anyone at the table?	300	2502	1949	5.41 (4.86)	71%	$0.70	29%	54%	77%	84%	–	–
Yes/No	Are there cups or dishes in the sink?	1200	433	687	6.16 (5.81)	41%	$0.25	35%	71%	80%	84%	Q	–
Yes/No	Is there enough space to park a small car here?	300	8079	12194	5.42 (4.72)	42%	$4.36	36%	69%	69%	82%	–	–
Yes/No	Is the large wooden door open enough to see?	3600	748	827	7.58 (7.04)	59%	$0.30	33%	40%	62%	82%	X	–
Yes/No	Are the tables messy?	3600	736	1068	7.06 (6.81)	44%	$0.38	14%	74%	76%	81%	L	–
Yes/No	Do you see a garbage truck?	60	37735	30783	5.02 (4.60)	73%	$10.99	23%	61%	62%	81%	–	–
Yes/No	Are there more than 2 photos here?	3600	208	307	5.57 (5.09)	43%	$0.11	78%	33%	75%	79%	Q	–
Yes/No	This is the top of a trash can in our office. Is this trash can full?	1200	2140	4725	5.56 (5.17)	19%	$1.69	46%	67%	63%	79%	Q	–
Yes/No	Do you see the big metal shutter down ?	3600	776	215	7.22 (5.87)	90%	$0.08	19%	67%	79%	77%	–	–
Yes/No	Is the trashcan full or is there trash around the trashcan	300	8271	7693	5.63 (5.35)	66%	$2.75	18%	58%	63%	76%	–	–
Yes/No	Is there anyone in the glassed-in room?	300	2275	1447	5.67 (4.88)	75%	$0.52	10%	67%	73%	74%	Q	–
Yes/No	Are the lights in the building on?	60	14978	17995	5.01 (4.45)	56%	$6.43	18%	69%	60%	73%	C	P
Yes/No	Is the trashcan full or is there trash around the trashcan	300	8192	8146	5.44 (4.93)	63%	$2.91	21%	45%	62%	72%	–	–
Yes/No	Is someone standing at the printers?	10	150447	12797	5.18 (4.63)	97%	$4.57	11%	88%	64%	69%	–	P
Yes/No	Is there any paper or mail here?	3600	729	1408	6.46 (5.43)	26%	$0.50	18%	67%	70%	69%	–	–
Yes/No	This is the door of an office. Is anyone using this office?	300	4271	5996	5.41 (4.96)	47%	$2.14	19%	60%	56%	69%	CQ	P
Yes/No	This is a door of an office […] Is anyone using this office?	300	8235	9373	5.41 (5.45)	59%	$3.35	29%	49%	59%	69%	CQ	P
Yes/No	Is someone using a printer?	60	37893	4755	5.33 (4.53)	96%	$1.70	11%	73%	62%	68%	–	–
Yes/No	Are the tables setup in rows?	3600	210	325	6.46 (6.01)	41%	$0.12	18%	58%	70%	62%	–	–
Yes/No	Is the play room free now?	60	36667	5296	5.81 (5.19)	95%	$1.89	16%	73%	62%	59%	L	–
Yes/No	Are there papers on the printers?	3600	751	620	6.29 (5.52)	70%	$0.22	17%	45%	45%	58%	CQ	P
Yes/No	Is the light in the hallway to the right on?	60	17469	1152	5.70 (5.27)	98%	$0.41	39%	62%	55%	58%	X	P
Yes/No	Is the coffee machine in use?	3600	629	689	6.88 (6.28)	59%	$0.25	18%	71%	52%	57%	X	–
Yes/No	Is there food on the kitchen counter?	300	4273	2869	5.57 (5.09)	75%	$1.02	19%	74%	55%	57%	XQ	–
Yes/No	Is the meeting room free(Is the light on)?	300	3702	1083	6.22 (5.96)	89%	$0.39	28%	44%	47%	41%	L	P
	AVERAGE				**5.79**	**60%**	**$2.41**	**21%**	**67%**	**72%**	**80%**		
Count	How many people are in the room?	1200	2178	1226	6.49 (5.17)	78%	$0.44	10%	0.05	0.08	0.01	–	–
Count	How many cars do you see?	1200	2126	4196	5.75 (4.40)	21%	$1.50	20%	0.27	0.07	0.02	–	–
Count	How many people are in the room?	1200	2151	3493	6.65 (5.20)	36%	$1.25	12%	0.13	0.09	0.02	–	–
Count	How many people are sitting/using/sleeping on these two sofa?	300	5011	7143	5.95 (5.25)	45%	$2.55	17%	0.20	0.11	0.02	–	P
Count	How many people is sitting around this table?	300	8180	11712	5.91 (4.67)	44%	$4.18	11%	0.58	0.09	0.03	–	P
Count	How many people are sitting around this table?	300	5687	8661	5.96 (5.15)	41%	$3.09	12%	0.43	0.15	0.03	Q	P
Count	This is a small parking […] cars is parking here?	300	8135	17911	5.64 (5.07)	15%	$6.40	50%	0.26	0.16	0.06	X	P
Count	How many pedestrians do you see?	300	8106	14764	6.30 (5.35)	32%	$5.27	30%	0.45	0.32	0.09	–	–
Count	How many people are sitting on the benches out front?	3600	736	1411	6.38 (5.26)	26%	$0.50	23%	0.45	0.19	0.09	Q	P
Count	How many people are on the walkway?	60	5797	1925	6.11 (4.41)	88%	$0.69	6%	0.76	0.39	0.09	–	–
Count	How many cars are parked here ?	300	4382	5342	8.45 (6.30)	57%	$1.91	10%	2.00	0.34	0.10	–	–
Count	How many people used this door?	10	158122	113929	5.70 (4.75)	74%	$40.69	19%	0.39	0.29	0.12	L	–
Count	How many people are there?	300	890	762	6.51 (4.70)	67%	$0.27	19%	0.82	0.29	0.13	–	–
Count	How many people are in line at the cash register?	3600	480	756	8.29 (7.16)	40%	$0.27	15%		0.39	0.14	CQL	–
Count	How many people are sitting here ?	60	23893	26958	6.02 (4.83)	59%	$9.63	11%	1.01	0.34	0.14	–	–
Count	How many cars are waiting at the intersection?	300	8123	13317	6.19 (5.38)	39%	$4.76	33%	0.88	0.70	0.19	–	–
Count	How many non-parked cars do you see?	300	8082	15387	7.26 (5.81)	32%	$5.50	31%	0.78	0.51	0.20	X	–
Count	How many offices are occupied/in use right now?	300	1257	1644	6.51 (5.63)	51%	$0.59	16%	0.38	0.53	0.28	Q	–
Count	How many tables are not being used?	1200	575	610	7.88 (6.40)	61%	$0.22	5%	1.07	1.33	0.57	–	–
Count	How many cars are parked on the road?	3600	736	1867	8.46 (6.92)	10%	$0.67	29%	2.79	2.01	0.68	–	–
	AVERAGE				**6.62**	**45%**	**$4.50**	**19%**	**0.72**	**0.42**	**0.15**		

Figure 7.4: Detailed statistics on all 63 question sensors from our evaluation deployment.

7.6.1 Scale

Over 4 weeks, Zensors++ answered 937,228 sensor question instances, powered by both crowd workers and our automatic processes — an average throughput of 23 answers per minute. 606,903 individual labels were generated by 231 crowd workers, resulting in a throughput of 15 crowd labels per minute. In total, the deployment generated 43 GB of images (stored on Amazon S3 [10]).

7.6.2 Raw Crowd Accuracy

We conducted a post-hoc evaluation to quantify the accuracy of crowd answers. To investigate this, we randomly selected a subset of crowd-labeled question sensor instances (14,028 images). Three researchers then manually provided high-quality, ground-truth labels for this dataset. These expert labelers had the advantage of superior contextual understanding (e.g., able to see the whole image, not just the region of interest), as well as the ability to view datasets over time (e.g., knowing min and max of answer distributions).

For count questions, we found a mean absolute error of 0.47 (SD=0.56) units, which is reasonably accurate. However, for yes/no questions, we found a mean crowd accuracy of 62.8% (SD=26.0%). This low accuracy prompted us to investigate if there was a subset of workers bringing down the average. However, we did not find any obvious step function reduction in accuracy, and instead saw a continuous spectrum of crowd quality.

We noticed two types of malicious crowd behaviour. First were a subset of crowd workers who continuously selecting the “I can’t tell” option when questions were clearly answerable. Second, we found a set of workers who ignored our quality-control gold standard questions and warning pop-ups. Since we did not ban workers identified as malicious, they were able to repeatedly abuse the system. In order to ensure crowd quality, we recommend future systems employ continuous monitoring of crowd worker performance. To compensate for this errorful data, and estimate performance with effective safeguards in place, we decided to drop labels from the worst-performing 23 crowd workers (representing ~10% of our crowd workforce), which improved crowd accuracy by 8%. Accuracies with and without crowd worker data dropped is reported in Figure 7.4.

7.6.3 Question Sensor Accuracy

Sitting on top of crowd accuracy is the effective question sensor accuracy as seen by our participants. As previously discussed, we used a majority voting mechanism (that seeks extra answers as needed) to ascertain the best answer for a question sensor instance. This was intended to buffer against malicious crowd workers, which we confirmed in the previous section, and seek consensus for difficult cases.

Using our ground truth data set as a benchmark, we found the average accuracy for yes/no questions was 79.5% (max 98%, min 41%); 35% of the question sensors had an accuracy of over 90%, and 56% were over 80% accurate. For count questions, we found an absolute mean error of 0.2 (min error 0.01, max error 0.68). The latter result is particularly strong, as most count questions centered around people and car counting, with our results showing we were off by less

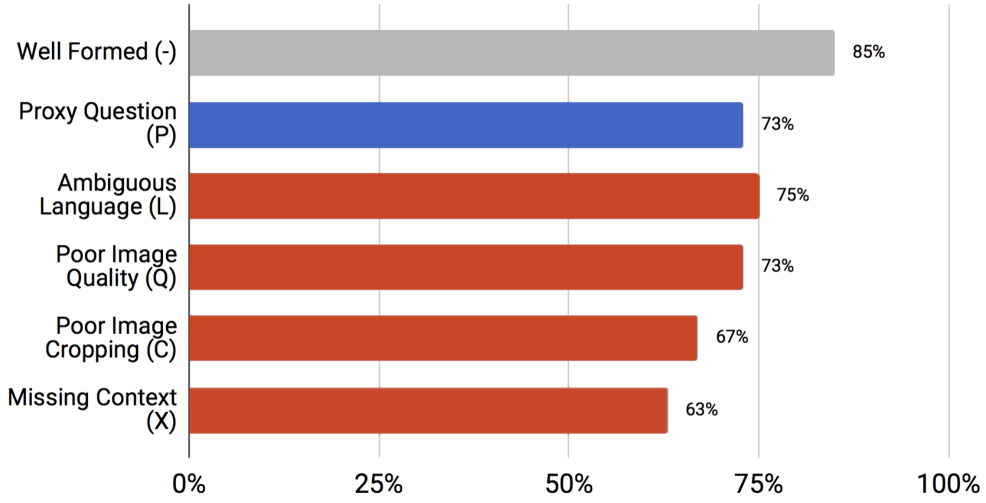


Figure 7.5: Yes/No question sensor accuracy when well formed (gray bar), formulated as a proxy question (blue bar), and when exhibiting four different error types (red bars).

than one unit on average. Five of the six most accurate count questions involved people counting, which is a relatively straightforward task where crowd workers are adept. This also highlights an opportunity to guide user-defined question sensors toward archetypes known to be accurate. Techniques such as CrowdVerge [84] could also be applied to provide automatic feedback on whether the crowd will agree on answers. A full table of question sensor accuracies can be found in Figure 7.4.

7.6.4 User-Defined Question Errors

Although we included an onboarding guide to help participants formulate effective question sensors, we encountered similar quality issues as found in our discovery deployment. In response, we systematically studied all 63 question sensor feeds, qualitatively identifying problems. We labelled error type, if any, in Figure 7.4. Overall, yes/no questions with no discernible problems had a mean accuracy of 85%, shown as the gray bar in Figure 7.5. On the other hand, question sensors marked with any error type had a mean accuracy of 71% (red bars in Figure 7.5).

The most frequent problem (13 of 63 sensors) was poor image quality (error label Q in Figure 7.4). We found users often asked questions about specific items in the field of view that were nearly (or totally) impossible to discern from the commodity cameras we used. For example, “are there papers on the printers” failed because of insufficient contrast between paper and printers. In these scenarios, higher quality cameras would undoubtedly boost accuracy.

Question sensors with insufficient context (error label X in Figure 7.4) had the lowest mean accuracy of 63%. This may stem from users having too much implicit knowledge about their local environments, which is hard to encapsulate in a short question or tightly cropped image region. For instance, in a tightly cropped view of a road, it can be difficult to tell if a partially visible car is parked or driving. An example of an ambiguous language type error (label L in Figure 7.4) was “is the meeting room free (is the light on)?”, which was confusing to both our crowd and

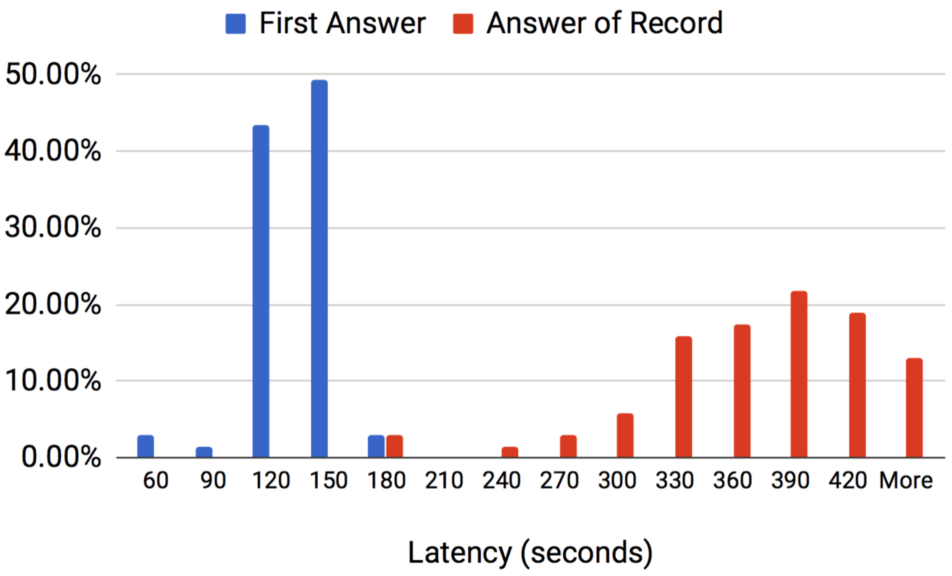


Figure 7.6: Histogram of end-to-end crowd latency. Median time to first answer was 120 seconds, while median time to answer of record was 355 seconds.

expert labelers as generally when rooms were free, lights automatically turn off. To make these question sensors more accurate in the future, we suggest asking users to provide examples to crowd workers, shepherd the crowd to yield better work quality [58], suggest superior phrasing, or show crowd workers the full camera image with region of interest highlighted.

7.6.5 Crowd Latency

The main source of latency in Zensors++ is the time required to gather answers from the crowd. We recorded several metrics: *(i)* time taken for a crowd worker to provide an answer for an individual question sensor instance (labeling duration), *(ii)* time taken from the moment an question sensor instance is captured to the moment the first crowd answer is received (first answer end-to-end latency), and *(iii)* time taken from the moment an question sensor instance is captured to the moment an answer of record is decided (answer of record end-to-end latency). Note that these metrics do not include the near-zero latency when an answer can be automated through perceptual image hashing and machine learning.

As shown in Figure 7.4, average labeling duration was 5.8 seconds for yes/no questions, and 6.6 seconds for count questions. Median end-to-end latency to get the first crowd answer was 120 seconds (SD=19.6). The end-to-end latency for answers of record was longer (median time of 355 seconds) and more distributed (SD=59.1), as two (and sometimes three) crowd labels were needed by our majority voting scheme. Although our disagreement queue had high priority, it could still take on the order of minutes to recruit a new worker who had not previously seen the image. Figure 7.6 provides the histogram of these latencies. Future systems might consider revealing answers as they come in, for example, revealing the first label to users, but marking it as tentative. As later answers arrive, the confidence can be updated, offering a balance between

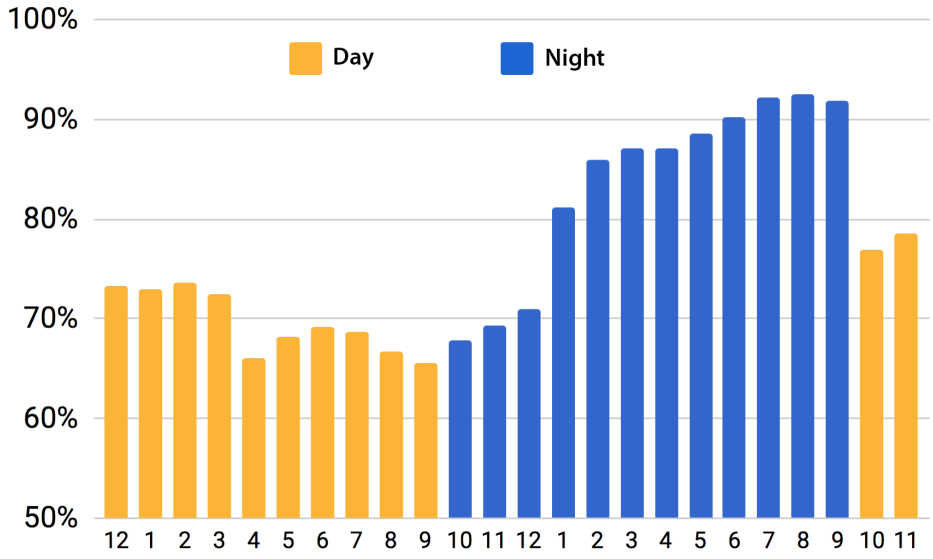


Figure 7.7: Mean perceptual hash hit rate over 24 hours, starting at noon, far left.

latency and accuracy (similar to the public information display in [71]).

7.6.6 Hashing Performance

We evaluated the effectiveness of perceptual image hashing in detecting and reducing redundant images. Out of the 937,228 question sensor instances Zensors++ received, 74.4% (697,345) had a hash hit, and did not have to go to the crowd for labeling, providing an answer at near-zero latency and cost (saving us approximately $17,500 over the course of 4 weeks). As one might expect, hash hit rate varied over the course of the day (Figure 7.7), and is most successful between 1am to 9am, when there is little human activity.

To more closely evaluate how well our perceptual image hashing scheme performed in identifying similar images, we randomly selected 200 images that were marked as "hashed" and retrieved the original crowd-labeled image from which the answer was inherited. A researcher then determined whether the new image was sufficiently different from the original to warrant a change in the labeled answer. Differences were found in only two cases out of the 200 image pairs, resulting in a hashing accuracy of 99.0%. In both failure cases, the selected region was large, but the question asked about relatively small changes — so small as to not alter the perceptual hash of the image, causing old (and incorrect) labels to be copied forward to hash hits. To mitigate this, we propose using longer length hashes for larger images, or applying more advanced hashing algorithms, such as those combining deep learning [141].

7.6.7 Cost

Next, we evaluated the cost of the question sensors created by our participants. In total, Zensors++ provided answers to 937,228 question sensor instances over four weeks, costing $6,069 in crowd

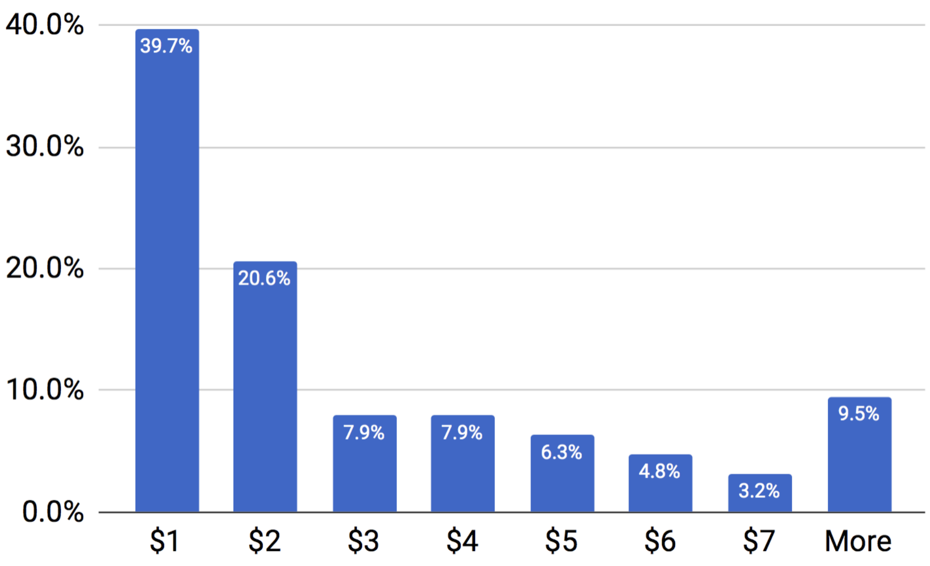


Figure 7.8: Distribution of daily cost across all question sensors without machine learning. More than 60% of question sensors cost $2/day or less to operate.

labor, for an average cost per answer of $0.006. As discussed previously, we utilized a majority voting scheme that started with two crowd answers, and sought a third answer only when necessary. Thus, instead of soliciting three labels per question sensor instance, Zensors++ required 2.53 labels on average, which saved $1,127 in crowd costs.

The cost of question sensors is directly correlated with their sampling frequency (see Figure 7.4). The average per-day cost of yes/no questions was $2.41 (SD=2.79) and $4.50 (SD=8.90) for count questions. The higher cost of count sensors was partly due to a lower hash hit rate of just 45% (vs. 60% for yes/no questions). We suspect this is because questions needing count answers are inherently more dynamic and complex, whereas yes/no questions are fundamentally testing the presence or absence of a visual feature.

The most expensive count sensor cost $40.69 per day, whereas the most expensive yes/no question cost $10.99 per day. However, from Figure 7.8, we see that 60% of our question sensors cost less than $2 per day. This result underscores that many question sensors are already within the bounds of our participants' perceived value (discussed later). Over time, cost could be further reduced by relying increasingly on hashed answers and machine learning. The daily cost distribution (Figure 7.8) shows that it is possible to run periodic crowd validation after a machine learning hand-off on a permanent basis. This suggests long term viability of such hybrid human-AI systems for many use cases.

7.6.8 Machine Learning Accuracy

To test the feasibility of machine learning for answering question sensor instances, we ingested data at the end of each day, and trained classifiers for each question sensor based on all available crowd-powered answers of record. We then benchmarked these classifiers against next-day crowd performance.

Figure 7.4 offers the machine learning accuracy for every question sensor on the last day of the deployment (i.e., trained on data from days 1 through 27, tested on day 28). The average accuracy for machine-learning-powered yes/no questions was 67% (SD=14%), which is close to average crowd accuracy of 72% (SD=13%). Of note, 43% of our yes/no questions exceeded crowd accuracy. Machine-learning-powered count questions had an average error of 0.72 (vs. 0.42 for the crowd). We saw that our classifier accuracy climbed quickly, often reaching its best performance within 2-4 days. However, after this point, accuracy would fluctuate, owing to sporadic changes in the environment. It may be that with more time and data, all environment "corner cases" could be captured and integrated into the model.

As for failure cases, we found that the two worst performing, machine-learning-powered yes/no questions (33% and 40% accuracy) had very little training data (208 and 748 labeled instances respectively), owing to their one image per hour sampling frequency. This may mean that higher rate question sensors, though more costly upfront, might enable a faster handoff. Additionally, we found that some sensors tasked with capturing infrequent events like "Do you see a garbage truck?" (frequency 60 seconds), which might only capture four positive labels in a month, introduced a significant skew in class labels. These may be addressable in the future by, e.g., oversampling positive examples.

Results suggest that some questions might never reach sufficient machine learning accuracy to be handed-off, and would always require crowd power. However, poor classifier performance can be easily detected by periodically bench-marking against the crowd. When handoff is determined to be infeasible, question sensors could be run indefinitely on crowd power if infrequent (e.g., every hour) with acceptable cost (e.g., $0.60 a day, or under $0.25 a day with average hashing performance), or could be disabled and prompt their creators to rephrase or otherwise improve the question sensor.

7.6.9 Participant Use of Data

Participants' usage of the system ranged from logging in once per month to multiple times per day. We found that participants in commercial settings (lab managers, administrators, facility operators) were more interested in investigating the longitudinal data feed to identify trends and changes. However, users in a personal capacity cared more about what was currently happening when using the system. Often times, for a single sensor, multiple stakeholders were interested in using the data feed differently. For example, for the question sensor "How many people are in line at the cash register?", a restaurant manager was interested in using the longitudinal data to identify consumption patterns to better plan and prepare food, while our student participants were more interested in knowing whether they can go and grab lunch without standing in a long line. Overall, participants were excited about the system's potential and wanted to incorporate it into their daily routines.

7.6.10 Types of Question Sensors

Participant question sensors (listed in Figure 7.4, examples shown in Figure 7.1) fell into one of two main categories: event/state or capacity/availability. Event/State question sensors focused on changes in the state of the environment, including: “Is there any paper or mail here?”, “Is anything written on the whiteboard?”, “Do you see the big metal shutter down?”, and “How many people are in the room?” On the other hand, capacity/availability question sensors tracked resource utilization, including: “Is the coffee machine in use?”, “Is the trash can full or is there trash around the trash can?”, “Is this seat occupied?”, and “How many tables are not being used?” We noticed that many questions centered around cars, people and lights. In future systems, it may be advantageous to use machine learning models tailored for the detection of these entities to facilitate early handoff.

7.6.11 Proxy Questions

We found that some questions are difficult (or impossible) for crowd workers to answer because they lack necessary context. For example, in brainstorming during onboarding, P15 wished to ask “Is my husband home?” using an outside camera view of their home’s parking. Of course, crowd workers do not know P15’s husband, and therefore were likely to provide inaccurate answers. Instead, P15 formulated an alternative question that was context free — “Do you see a motorcycle here?” — selecting a region where her husband parks. This question thus served as a “proxy” to know whether her husband is at home. As another example, P7 asked “Are the tables set up in rows?” as a proxy question for whether he needed to go to the classroom early to arrange the room before lecture. Lastly, P1 asked “Is someone standing at the printers?” as a proxy for knowing whether the printer queue was busy. The right-most column in Figure 7.4 denotes what we believe to be proxy questions, based on entry and exit interviews with participants.

Despite these examples of proxy questions, it remains unclear how often participants did not ask a question of interest because it could not be directly answered (even though a proxy question might have worked). Future work might consider techniques to help users define and formulate proxy questions to sense questions they care about when context is complex, e.g., through having conversations with the crowd to define rules [93].

7.6.12 Perceived Value

We asked participants how much they would be willing to pay for the question sensors they created in the context of a commercial product. For personal use question sensors (e.g., “Is there any paper or mail here?”), participants said they would pay on the order of $1-10 per month. For question sensors about line length and wait time, participants reported they would not pay for them personally, but would appreciate and prefer businesses that made such information available. Participants also called out question sensors that had poor accuracy during deployment, and said they would not pay for such data.

Participants who were acting in a professional role, such as a facilities or lab manager, were willing to pay the most (hundreds to thousands of dollars annually) to monitor longitudinal usage and important events (e.g., “Is the trashcan full or is there trash around the trashcan”, “Is anyone using the tools or equipment?”, and “The basement can flood, do you see water on the ground?”). This was especially true for question sensors that directly complemented or augmented existing work practices.

For example, a building manager's (P17) typical practice was to dispatch workers to inspect trashcans every few hours, emptying them when full. With Zensors++, he received text notifications when trashcans were full, and began to dynamically dispatch his staff. P17 also created question sensors to detect basement floods, which would usually require someone reporting an incident, or for one of his staff to discover flooding. Either way, flooding out of work hours resulted in a delayed response, that could now be proactively handled with Zensors++. Indeed P1, P2, P14, P15 and P17 all noted in interviews the importance of pushed data in forms of digests, reports and notifications, which they could more easily integrate into their work flows. In another case, a program director (P1) was using Zensors++ to ask "Is someone sitting on any of this furniture?" in order to test different furniture arrangements, akin to physical A/B testing.

7.6.13 Privacy and Sharing

Several participants (P2, P6, P14, P17) noted they were initially worried about privacy invasion, but once we explained our privacy preservation measures, they indicated feeling more favorable about using cameras for visual sensing. This behavior is consistent with the economic model of privacy, where users often make trade-offs between privacy and utility. P15 mentioned she forgot about the cameras after setting them up, chiefly looking at the data. P1 said it felt odd to have the ability to see what everyone is doing, and suggested to abstract images to a drawing or data visualization [32].

Most participants agreed that for cameras in common spaces, everyone residing or working in that space should assent, as well as have access to the camera and question sensors. For this reason, we used signs at our institution to inform people of camera-based sensing, and provided contact details to add them to the deployment if desired. All participants rejected the idea of sharing cameras and question sensors related to their homes, but did not seem to mind that images were being labeled by crowd workers. To further improve privacy in the future, systems could integrate human-in-the-loop progressive filtering [105], hire private professional crowds, or allow users to label their own data until machine learning is able to take over.

Of note, throughout deployment at our institution, signs were used to inform people that camera-based sensors were running. We did not formally study their effect, however, in one instance a camera was set up and accidentally omitted a printed sign. By the end of the day, multiple facility managers had received emails about the camera. However, in all other instances with properly placed signs, no concerns were raised during the several months of the study.

7.7 Future Work

While we concentrated on personal and commercial uses in this work, there are no doubt interesting industrial, civic and health application domains that could be well suited to Zensors++. Similarly. there are options to move beyond recruiting crowd workers solely from Amazon Mechanical Turk, and explore how different crowd groups are able to work within Zensors++. For example, a private crowd could produce higher quality results, potentially alleviating the need for e.g., majority voting. It may also be possible for long-term crowd workers to engage with users to refine question sensors, and even curate the machine learning handoff.

Our deployment, which continues at the time of writing, is gathering substantial data of a kind that is different from other visual questioning answering tasks. Foremost, it is real data from real questions that people have. Each question also contains many sequential examples, as opposed to one-off question instances. In the future, we hope to release a dataset for computer vision researchers. We would like to study how models learned from one question sensor can transfer to other question sensors of a similar type.

Finally, it would be interesting to more deeply explore how users are willing to share access to their cameras, question sensors and data. For example, some of our participants pointed cameras out of windows to monitor outside scenes. It would be interesting if e.g., municipalities could utilize these cameras as an ad hoc distributed sensing system for "is their litter on the sidewalk?" and "is snow accumulating on the road?" A business across the street might ask e.g., "is there available parking in front?" and "did a customer enter?" The ubiquity of cameras offers a unique opportunity for a public and highly distributed sensing fabric offering unmatched generality.

7.8 Conclusion

We have described our work on Zensors++, a human-AI sensing system designed to answer natural language user questions based on camera streams. We started with a discovery deployment, which helped to distill key insights that lead to the development of an improved system architecture and feature set. We then conducted a second deployment for four weeks, with 17 participants, resulting in nearly a million answers for 63 participant-defined question sensors. We demonstrated that crowd workers were able to provide labels quickly and at scale, and that the system could hand-off to machine learning classifiers in many cases. We also demonstrated the importance of image hashing for dramatically reducing the number of questions that needed to go to the crowd for labeling. We further discussed important system-related characteristics, such as scalability, accuracy, error types, latency, cost, and effectiveness of automation. We also synthesized feedback from our study participants, revealing user motivation, perceived value, privacy, and overall utility. Overall, we believe our deployments demonstrate the feasibility of human-AI, camera-based sensing at scale, and offer a number of insights for those wishing to deploy robust and responsive systems in the future.

Chapter 8

Conclusion and Future Directions

8.1 Thesis Contributions

In this dissertation, I have described a suite of human-AI systems I developed and deployed to enable visual information access in the real world. By combining the advantages of humans and AI, these systems can be nearly as robust and flexible as humans, and nearly as quick and low-cost as automated AI, enabling us to solve problems that are currently impossible with either alone. More broadly, this dissertation has made contributions in the domains of accessibility, intelligent user interfaces, ubiquitous computing, and generally, in human-computer interaction research.

Starting with the ones to make physical interfaces accessible for blind people, I introduced three systems to interpret static and dynamic interfaces, enabling blind people to independently access them through audio feedback or tactile overlays. I first created VizLens in Chapter 3, a robust and interactive screen reader for real-world static interfaces. VizLens trades off the advantages of humans and computer vision to be nearly as robust as a person in interpreting the interface and nearly as quick and low-cost as a computer vision system to re-identify the interface and provide real-time feedback. I further explored cursor-based interactions to support non-visual explorations by blind users in Chapter 4, integrating VizLens's real-world scene reader interaction as a type of finger cursor. I then described Facade in Chapter 5, a crowdsourced fabrication pipeline that enables blind people to independently create 3D-printed tactile overlays for inaccessible appliances. Facade makes end-user fabrication accessible to blind people, by shifting the sighted assistance to a virtual crowd working with computer vision. Facade combines a human-AI interpretation pipeline with an accessible 3D printing application.

VizLens and Facade enable blind users to access many static interfaces. To make dynamic touchscreens such as public kiosks and payment terminals accessible, I next introduced StateLens in Chapter 6 that addresses the very hard case in which blind users encounter a touchscreen in the real world that is inaccessible, which they cannot modify the hardware or software, and whose screen updates dynamically to show new information and interface components. Furthermore, StateLens takes advantage of different kinds of human intelligence: humans who provide access and collect videos at the interface to build up the training data, and online crowds who provide necessary labels to bootstrap automation.

Furthermore, for environmental sensing, I described the development and deployment of

Zensors++ in Chapter 7, a human-AI camera sensing system to answer natural language user questions based on camera streams. Zensors++ relies on end users to define questions of interest and specify image region, as well as online crowd workers to provide labels when necessary. Then it relies on machine intelligence to automate over time to reduce the cost and latency.

8.2 Research Vision and Approach

From an industry standpoint, these human-AI systems could accelerate the development lifecycle of building AI products. A simplified version of the Agile development lifecycle for an AI product may include (left side of Figure 8.1): first identify a problem or need, then find or collect the data for solving the problem, then build and test the model, and finally release and deploy it, then ideally from the usage data, repeat this process.

For the example of a conversational agent product like Alexa, users requests that could not be fulfilled are looked at, then prioritized in terms of importance. Then resources are devoted to integrate these new features before finally releasing them. It typically takes weeks for these new features to be available. For computer vision products such as the Nest Cam, this cycle is even longer. One problem with this long cycle is re-discoverability. Once users attempted a request but the system was not able to deliver, users will have the mental model that the system will not work and they would not try again. A bigger problem is the lack of opportunities to collect real-world data to make AI work. Without a working system, people simply would not opt in for such data collection.

For the human-AI systems I have developed and will create in the future, this process could be accelerated from weeks down to minutes or seconds (right side of Figure 8.1). These systems collect data for users' immediate needs, in order to build a model to work in the moment. To the end users, these systems are always intelligent and smart. But under the hood, large-scale data can be collected, and automation can be achieved overtime to support these user needs.

VizLens illustrates the approach I take in my research: I start by identifying a real problem (physical interfaces are not accessible), next understand where humans and machines work best

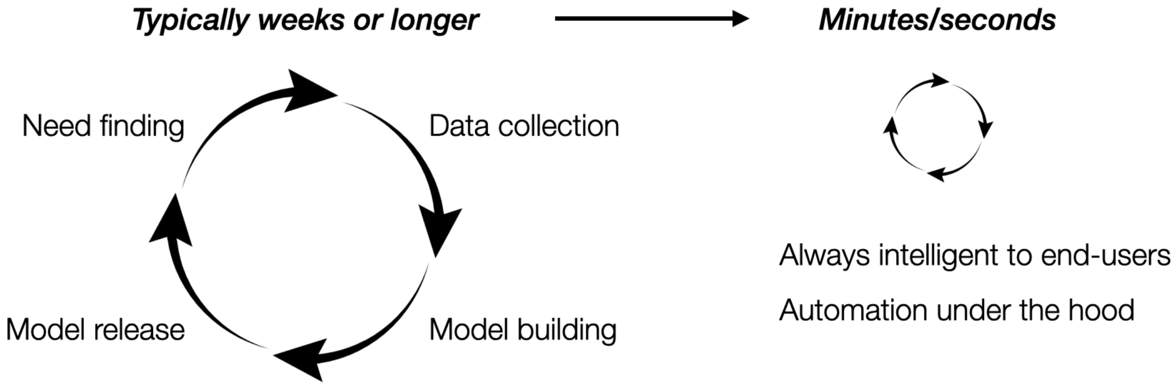

Typically weeks or longer
Minutes/seconds
Need finding
Data collection
Model release
Model building
Always intelligent to end-users
Automation under the hood

Figure 8.1: Human-AI systems could accelerate the development lifecycle of building AI products.

for solving this problem (human for interpreting arbitrary interface, machine for remembering patterns and re-identifying later), then design human-AI systems as technology enablers, and finally deploy them in the wild to collect big data for understanding their limits and contributing back to the AI community for approaching automation via datasets (e.g., VizWiz dataset [85], VizWiz-Priv dataset [86]). Next, I outline opportunities that I am excited to pursue in the future.

8.3 Future Directions

8.3.1 Accessibility as a Driving Force of AI

Accessibility is a unique problem domain because of its challenging constraints, but also the high potential value of technology enablers for end users [25]. In my research, I have pushed boundaries in access technologies for enabling visual access in both the real world and the digital world. In my dissertation, I focused on accessibility of visual information in the real world [76, 77, 80, 82]. With collaborators, I have also explored making digital content more accessible such as screenshots [146], and mobile augmented reality apps [90]. Moving forward, I am excited to continue working with people with disabilities to solve real challenging problems people face, and in turn study their use to inform the directions of building better and beneficial AI.

More specifically, continuing my work on making physical interfaces accessible for blind people, I am eager to bring the human-AI systems I have developed into the real world to help blind people in their everyday lives, to learn about their naturalistic usage, and to gradually construct an Internet of Interfaces. I plan to harden and deploy the VizLens and StateLens systems to better understand how blind users of diverse vision capabilities use physical interfaces in the wild. For example, I am interested in investigating whether the usage, interactions, errors, and challenges emerged differ for people who are blind or low vision, for young or old, for those who were blind at birth vs. those who lost sight later, etc. Furthermore, supporting such deployments will require substantial engineering in order to achieve running high-performance and low-latency computer vision on-device. So far, we have ported a few core pieces of the computer vision pipeline to iOS and Android, and demonstrated feasibility.

Deploying VizLens and related systems will also allow the collection of a unique dataset of real-world interfaces and interface interactions from blind users in the real world, in order to build a general model that learns from the individual cases. We hope to release this dataset to inspire computer vision researchers to work on this problem (similar to the VizWiz [85] and VizWiz-Priv [86] datasets). Yet, successful application of computer vision often requires massive amounts of data. As a way to validate StateLens at a larger scale and also as a complement to the VizLens dataset, we can collect videos of interface use from YouTube and other online repositories. People often share instructional videos about how to use kiosks they encounter, and we can process this data into a dataset using the same structure as the VizLens dataset, and release it as an additional challenge to the computer vision community, structured so that it might also benefit blind users.

Through the deployments, a queryable map of state diagrams for many of the devices in the world can be built using point-of-view videos that are collected by users and existing ones shared online. As users start to use a device, it can be geolocated, automatically recognized, or added into the queryable map. Additional states can be added to the existing diagram as users interact with the device. Changes to the devices can be automatically detected over time to update the interface state diagram. Usage data can also be collected as users operate the interface, and used to guide novice users about common functionalities (one example is the natural language summary in StateLens). Over time, the system can adapt its support to the users' expertise and usage over time. Furthermore, similar but slightly different models of a device may reuse another state diagram and enable transfer learning. Eventually, inaccessible and legacy devices can be transformed into smart IoT devices through an external "brain" powered by humans and AI.

8.3.2 Enabling Access in More Contexts

The systems that I create could find broader applications in other domains. For example, StateLens could augment how people generally interact with touchscreen interfaces in the real world as cognitive assistance. When configuring complicated medical devices, or when interacting with machines in different languages, StateLens could provide guidance through visual overlays in augmented reality [110]. These overlays can be in the forms of visual indicators, animated instructions, or simplifications of existing interfaces (Figure 8.2). We are also exploring authoring tools to help experienced users create interactive AR tutorials through demonstration, in order to then help novice users operate a device such as through replaying the hand movement trace, or movement of 3D objects [111].

Furthermore, Zensors++ could assist blind users in sensing visual changes in their environments. However, this will introduce new questions about users' trust with the system and dealing with the uncertainty and errors. In future work, I am excited to explore multi-modal approaches

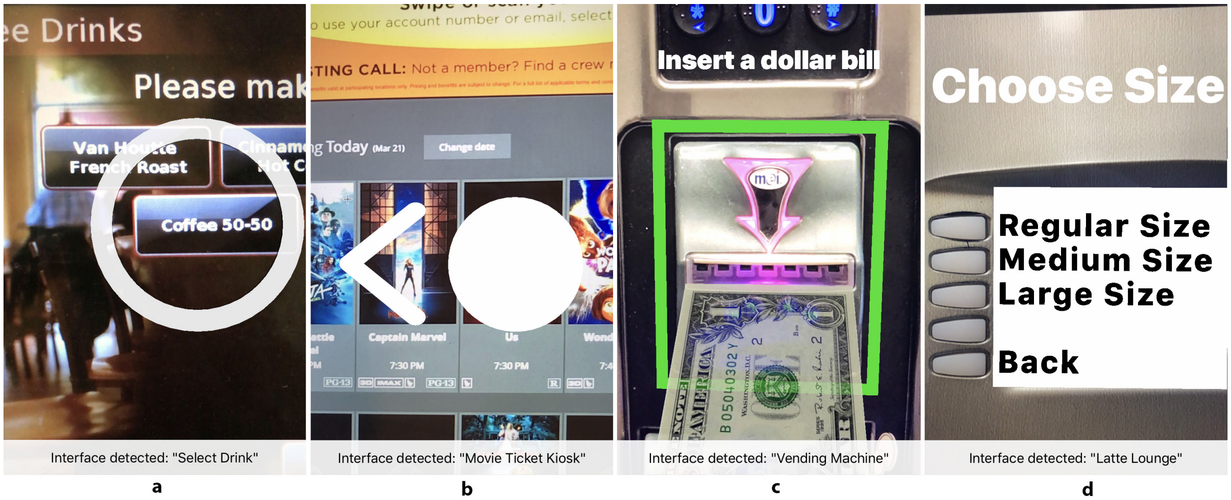

Please mak
Van Houtte French Roast
Coffee 50-50
Interface detected: "Select Drink"
a
Captain Marvel
Us
Interface detected: "Movie Ticket Kiosk"
b
Insert a dollar bill
Interface detected: "Vending Machine"
c
Choose Size
Regular Size
Medium Size
Large Size
Back
Interface detected: "Latte Lounge"
d

Figure 8.2: Design of AR-based visual guidance for different types of interactions: *(a)* On a coffee machine interface, a floating circle is displayed around the target button "coffee 50-50"; *(b)* On a movie ticket kiosk, an animated circle moving towards the target swipe direction is displayed to guide the user to swipe to the next page of movie list; *(c)* On a snack vending machine interface, an animation of inserting bills is displayed with the area highlighted, in order to guide the user to complete the payment; *(d)* On a text-heavy coffee machine, an overlay with bigger fonts and higher contrast is displayed on top of the original interface to make it more readable.

for end users to explore AI models' output to better establish their confidence for decision making. For example, with techniques such as VizLens and Touch Cursor, blind users could use the touchscreen to explore an image overlaid with many segmented regions for text, objects, faces, areas, colors, etc., which may be helpful for them to interpret an end-to-end generated image caption with the appropriate level of confidence.

My work on enabling no-touch, wrist-only interactions on smartwatches [72] has broader impact for not only people with situational impairments, but also for people with limb differences. Techniques for identifying user handprints on capacitive touchscreens [75] and presenting picking orders on head-up displays [73] could inform how assistive technologies be developed with limited hardware capabilities and with users' limited attention. In my future research, I will continue thriving to develop solutions that are generalizable across domains and contexts.

8.3.3 AI Datasets and Fairness

In addition to developing system, pushing the boundaries of HCI and AI also requires better datasets rooted in human problems. I have collaborated with AI researchers in developing datasets for visual question answering [85] and privacy [86], and I plan to continue this direction, as discussed in Section 8.3.1 to collect a dataset of interfaces and interface interactions from deploying VizLens and StateLens.

Relatedly, huge challenges exist in ensuring that the systems we are developing are fair for everyone, regardless of their gender, race, and disabilities. I have started to explore this in several directions. First, considerations regarding fairness in AI for people with disabilities have thus far received little attention, including issues threaten to lock people with disabilities out of access to key technologies (e.g., if voice-activated smart speakers do not recognize input from people with speech disabilities), inadvertently amplify existing stereotypes against them (e.g., if a chatbot learns to mimic someone with a disability), or even actively endanger their safety (e.g., if self-driving cars are not trained to recognize pedestrians using wheelchairs). To address these problems, we proposed a roadmap for identifying and addressing fairness issues of AI systems for people with disabilities [81], including *(i)* identify ways in which inclusion issues for people with disabilities may impact AI systems; *(ii)* test inclusion hypotheses to understand failure scenarios and the extent to which existing bias mitigation techniques work; *(iii)* create benchmark datasets to support replication and inclusion (and handle the complex ethical issues that creating such datasets for vulnerable groups might involve); and *(iv)* innovate new modeling, bias mitigation, and error measurement techniques in order to address any shortcomings of status quo methods with respect to people with disabilities. In our position paper [81], we focused on the first step, in which we performed risk assessments of existing AI systems with respect to different classes of disability.

Next, we have conducted a study to understand people with physical disabilities' experiences with sensing systems [104], focusing on the many challenges status quo sensing systems present for people with physical disabilities, as well as the ways they mitigate, react, and adapt to these challenges. Our findings point the way toward future opportunities to design and deploy more inclusive sensing systems. Furthermore, we are investigating the challenges and tradeoffs in fairness evaluations themselves, and contributing design considerations when making choices for such evaluations. In future work, I am also excited to design and study techniques for harvesting collective intelligence to uncover “unknown unknowns” in AI models, such as through error-based prompts, concept-based prompts, and real-time feedback.

8.3.4 Privacy Implications

In order to provide access to visual information, the human-AI systems described in this dissertation primarily use cameras as the visual sensor, which leads to privacy implications. In my future research, I am excited to design and study techniques for mitigating privacy concerns and balancing the trade-offs between privacy and utility.

For example in VizLens and Facade, before the initial reference images are sent to online crowd workers for labeling, blind users could use the smartphone touchscreen to explore the image overlaid with preliminary object recognition and OCR results, which may be helpful for them to discover potential privacy disclosures and make their own decisions of whether to submit that image. Additionally, using the VizWiz-Priv dataset [86], we could design algorithms to notify users with potential private information and its category, as well as automatically apply inpainting on private regions of the images.

For StateLens, users might not want to specify sensitive information by talking to the conversational agent with voice, e.g., passwords, payment information, and medical records. In those cases, sensitive information could be pre-stored on-device and automatically applied to preserve users’ privacy. The smart assistive hardware proxy discussed in Section 6.8.3 could also alleviate privacy concerns of shoulder surfing because the hardware itself is covering the screen while providing input actuation.

In Zensors++, to preserve the users’ privacy, end users can selectively reveal small regions of interest of their environment to crowd workers. Furthermore, we implemented several mechanisms to de-identify personal information, as detailed in Section 7.4.5. To further improve privacy in the future, systems could integrate human-in-the-loop progressive filtering, hire private professional crowds, or allow users to label their own data until machine learning is able to take over. We could also explore pushing the automation components to the edge, which the users have more control of.

To mitigate the privacy concerns of using cameras for sensing, I am also excited to explore sensor fusion and transfer learning techniques to use high-level events labeled using privacy-obtrusive sensors (e.g., cameras, microphones) to gradually train and transition into using privacy-unobtrusive sensors (e.g., pressure and motion sensors). Additionally, systems could notify users of nearby camera sensors before they enter the environments, and incorporate their privacy preferences.

8.4 Conclusion

In this dissertation, I have described a suite of human-AI intelligent interactive systems to enable visual information access in the real world. By combining the advantages of humans and AI, these systems can be nearly as robust and flexible as humans, and nearly as quick and low-cost as automated AI, enabling us to solve problems that are currently impossible with either alone. These human-AI systems focused on two application domains: accessibility and environmental sensing. To make physical interfaces accessible for blind people, I developed systems to interpret static and dynamic interfaces, enabling blind people to independently access them through audio feedback or tactile overlays. For environmental sensing, I developed and deployed a camera sensing system that collects human labels to bootstrap automatic processes to answer real-world visual questions, allowing end users to actionalize AI in their everyday lives. AI systems often require a huge amount of up front training data to get started, but targeted human intelligence can bootstrap the systems with relatively little data. Although humans may be slower initially, quickly bootstrapping to automated approaches provides a good balance, enabling human-AI systems to be scalable and rapidly deployable. These human-AI systems bring us closer to the vision of building a layer of hybrid intelligence, integrating humans and AI and tightly weaving the physical and digital worlds.

Bibliography

[1] 3D Hubs. 3D Hubs. https://www.3dhubs.com, 2019. 5.1

[2] Aira. Aira. https://aira.io, 2019. 2.2

[3] Amazon. Echo look | hands-free camera and style assistant with alexa–includes style check to get a second opinion on your outfit, 2018. URL https://www.amazon.com/Amazon-Echo-Look-Camera-Style-Assistant/dp/B0186JAEWK. 7.1

[4] Amazon Mechanical Turk. Amazon Mechanical Turk. http://www.mturk.com/, 2019. 2.2, 7.2.2

[5] Amazon Web Services. Amazon web services (aws) – cloud computing services, 2018. URL https://aws.amazon.com. 7.4.1

[6] Amazon Web Services. Amazon elasticache, 2018. URL https://aws.amazon.com/elasticache/. 7.4.1

[7] Amazon Web Services. Elastic load balancing, 2018. URL https://aws.amazon.com/elasticloadbalancing/. 7.4.1

[8] Amazon Web Services. Message queues, 2018. URL https://aws.amazon.com/message-queue/. 7.4.8

[9] Amazon Web Services. Redis, 2018. URL https://aws.amazon.com/redis/. 7.4.4

[10] Amazon Web Services. Amazon s3, 2018. URL https://aws.amazon.com/s3/. 7.6.1

[11] Amazon Web Services. Amazon simple email service, 2018. URL https://aws.amazon.com/ses/. 7.4.4

[12] Amazon Web Services. Amazon simple notification service (sns), 2018. URL https://aws.amazon.com/sns/. 7.4.4

[13] Amazon Web Services. Amazon simple queue service, 2018. URL https://aws.amazon.com/sqs/. 7.4.8

[14] Amazon Web Services, Inc. Amazon rekognition, 2019. URL https://aws.amazon.com/rekognition/. 6.5.1

[15] American Foundation for Blind. Braille: Deciphering the code. http://braillebug.afb.org/braille_deciphering.asp, 2016. 5.4.3

[16] Ernesto Arroyo, Leonardo Bonanni, and Ted Selker. Waterbot: Exploring feedback and persuasive techniques at the sink. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pages 631–639, New York, NY, USA, 2005. ACM.

ISBN 1-58113-998-5. doi: 10.1145/1054972.1055059. URL http://doi.acm.org/10.1145/1054972.1055059. 7.2.1

[17] Nikola Banovic, Tovi Grossman, Justin Matejka, and George Fitzmaurice. Waken: Reverse engineering usage information and interface structure from software videos. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 83–92, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7. doi: 10.1145/2380116.2380129. URL http://doi.acm.org/10.1145/2380116.2380129. 6.2.1, 6.8.2

[18] Sian Barris and Chris Button. A review of vision-based motion analysis in sport. *Sports Medicine*, 38(12):1025–1043, 2008. 2.1, 7.1

[19] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *Proceedings of the 9th European Conference on Computer Vision - Volume Part I*, ECCV'06, pages 404–417, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-33832-2, 978-3-540-33832-1. doi: 10.1007/11744023_32. URL http://dx.doi.org/10.1007/11744023_32. 3.4.3, 5.4.1, 6.5.1

[20] Be My Eyes. Be My Eyes. http://www.bemyeyes.org, 2019. 2.2, 3.2.2, 5.2, 6.2.2

[21] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 33–42, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047201. URL http://doi.acm.org/10.1145/2047196.2047201. 2.2, 3.2.2, 7.4.9

[22] Michael S. Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz. Direct answers for search queries in the long tail. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 237–246, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1015-4. doi: 10.1145/2207676.2207710. URL http://doi.acm.org/10.1145/2207676.2207710. 2.1, 7.1

[23] Michael S Bernstein, Greg Little, Robert C Miller, Björn Hartmann, Mark S Ackerman, David R Karger, David Crowell, and Katrina Panovich. Soylent: a word processor with a crowd inside. *Communications of the ACM*, 58(8):85–94, 2015. 2.2, 3.2.2

[24] Hugh Beyer and Karen Holtzblatt. *Contextual design: defining customer-centered systems*. Elsevier, 1997. 6.4.1

[25] Jeffrey P Bigham and Patrick Carrington. Learning from the front: People with disabilities as early adopters of ai, 2018. 8.3.1

[26] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. Vizwiz: Nearly real-time answers to visual questions. In *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 333–342, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0271-5. doi: 10.1145/1866029.1866080. URL http://doi.acm.org/10.1145/1866029.1866080. 2.2, 3.2.2, 4.4.3, 5.2, 6.2.2, 7.1, 7.2.2, 7.4.9

[27] Jeffrey P. Bigham, Chandrika Jayant, Andrew Miller, Brandyn White, and Tom Yeh. 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*, pages 65–72. IEEE, 2010. doi: 10.1109/CVPRW.2010.5543821. 2.3, 3.2.2, 4.1, 4.2.2, 5.2

[28] Jeffrey P. Bigham, Michael S. Bernstein, and Eytan Adar. Human-computer interaction and collective intelligence. *Handbook of Collective Intelligence*, 57, 2015. 5.2

[29] Paul E Black. Manhattan distance"" dictionary of algorithms and data structures. *http://xlinux. nist. gov/dads//*, 2006. 3.4.4

[30] Meera M Blattner, Denise A Sumikawa, and Robert M Greenberg. Earcons and icons: Their structure and common design principles. *Human–Computer Interaction*, 4(1):11–44, 1989. 3.4.4, 4.3.1, 6.5.2

[31] Glenn A Bowen. Naturalistic inquiry and the saturation concept: a research note. *Qualitative research*, 8(1):137–152, 2008. 4.4.4

[32] Michael Boyle, Christopher Edwards, and Saul Greenberg. The effects of filtered video on awareness and privacy. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work*, CSCW '00, pages 1–10, New York, NY, USA, 2000. ACM. ISBN 1-58113-222-0. doi: 10.1145/358916.358935. URL `http://doi.acm.org/10.1145/358916.358935`. 7.6.13

[33] Erin Brady and Jeffrey P. Bigham. Crowdsourcing accessibility: Human-powered access technologies. *Foundations and Trends in Human-Computer Interaction*, 8(4):273–372, 2015. doi: http://dx.doi.org/10.1561/1100000050. URL `http://dx.doi.org/10.1561/1100000050`. 5.2

[34] Erin Brady, Meredith Ringel Morris, Yu Zhong, Samuel White, and Jeffrey P. Bigham. Visual challenges in the everyday lives of blind people. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2117–2126, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2481291. URL `http://doi.acm.org/10.1145/2470654.2481291`. 5.2

[35] Erin Brady, Meredith Ringel Morris, Yu Zhong, Samuel White, and Jeffrey P. Bigham. Visual challenges in the everyday lives of blind people. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 2117–2126, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2481291. URL `http://doi.acm.org/10.1145/2470654.2481291`. 2.1, 2.2, 3.2.2, 6.2.2, 7.1

[36] Stacy M. Branham and Shaun K. Kane. Collaborative accessibility: How blind and sighted companions co-create accessible home spaces. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 2373–2382, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702511. URL `http://doi.acm.org/10.1145/2702123.2702511`. 5.8

[37] Craig Brown and Amy Hurst. Viztouch: Automatically generated tactile visualizations of coordinate spaces. In *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction*, TEI '12, pages 131–138, New York, NY, USA,

2012. ACM. ISBN 978-1-4503-1174-8. doi: 10.1145/2148131.2148160. URL http://doi.acm.org/10.1145/2148131.2148160. 5.2

[38] Emeline Brule, Gilles Bailly, Anke Brock, Frederic Valentin, Grégoire Denis, and Christophe Jouffrais. Mapsense: Multi-sensory interactive maps for children living with visual impairments. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 445–457, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858375. URL http://doi.acm.org/10.1145/2858036.2858375. 5.2

[39] Erin Buehler, Amy Hurst, and Megan Hofmann. Coming to grips: 3d printing for accessibility. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '14, pages 291–292, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2720-6. doi: 10.1145/2661334.2661345. URL http://doi.acm.org/10.1145/2661334.2661345. 5.2

[40] Erin Buehler, Shaun K. Kane, and Amy Hurst. Abc and 3d: Opportunities and obstacles to 3d printing in special education environments. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '14, pages 107–114, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2720-6. doi: 10.1145/2661334.2661365. URL http://doi.acm.org/10.1145/2661334.2661365. 5.2

[41] Erin Buehler, Stacy Branham, Abdullah Ali, Jeremy J. Chang, Megan Kelly Hofmann, Amy Hurst, and Shaun K. Kane. Sharing is caring: Assistive technology designs on thingiverse. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 525–534, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702525. URL http://doi.acm.org/10.1145/2702123.2702525. 5.2

[42] H. Burton, D.G. McLaren, and R.J. Sinclair. Reading embossed capital letters: An fmri study in blind and sighted individuals. *Human brain mapping*, 27(4):325–339, 2006. doi: 10.1002/hbm.20188. 5.4.3

[43] Joseph Chee Chang, Aniket Kittur, and Nathan Hahn. Alloy: Clustering with crowds and computation. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 3180–3191, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858411. URL https://doi.org/10.1145/2858036.2858411. 2.3

[44] Tsung-Hsiang Chang, Tom Yeh, and Rob Miller. Associating the visual representation of user interfaces with their internal structures and metadata. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 245–256, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047228. URL http://doi.acm.org/10.1145/2047196.2047228. 6.2.1

[45] Xiang 'Anthony' Chen, Stelian Coros, Jennifer Mankoff, and Scott E. Hudson. Encore: 3d printed augmentation of everyday objects with printed-over, affixed and interlocked attachments. In *Proceedings of the 28th Annual ACM Symposium on User Interface*

Software and Technology, UIST '15, pages 73–82, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3779-3. doi: 10.1145/2807442.2807498. URL `http://doi.acm.org/10.1145/2807442.2807498`. 5.2

[46] Xiang 'Anthony' Chen, Jeeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E. Hudson. Reprise: A design tool for specifying, generating, and customizing 3d printable adaptations on everyday objects. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 29–39, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4189-9. doi: 10.1145/2984511.2984512. URL `http://doi.acm.org/10.1145/2984511.2984512`. 5.2

[47] Xiang 'Anthony' Chen, Jeeeun Kim, Jennifer Mankoff, Tovi Grossman, Stelian Coros, and Scott E. Hudson. Reprise: A design tool for specifying, generating, and customizing 3d printable adaptations on everyday objects. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 29–39, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4189-9. doi: 10.1145/2984511.2984512. URL `http://doi.acm.org/10.1145/2984511.2984512`. 6.4.1

[48] Justin Cheng and Michael S. Bernstein. Flock: Hybrid crowd-machine learning classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing*, CSCW '15, pages 600–611, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-2922-4. doi: 10.1145/2675133.2675214. URL `http://doi.acm.org/10.1145/2675133.2675214`. 2.3, 7.1, 7.2.2

[49] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies*, 6(4):271–288, 1998. 2.1, 7.1

[50] Paolo Comelli, Paolo Ferragina, Mario Notturno Granieri, and Flavio Stabile. Optical recognition of motor vehicle license plates. *IEEE transactions on Vehicular Technology*, 44(4):790–799, 1995. 2.1, 7.1

[51] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Trans. Inf. Theor.*, 13 (1):21–27, September 2006. ISSN 0018-9448. doi: 10.1109/TIT.1967.1053964. URL `https://doi.org/10.1109/TIT.1967.1053964`. 7.4.11

[52] Antonio Criminisi, Ian Reid, and Andrew Zisserman. A plane measuring device. *Image and Vision Computing*, 17(8):625–634, 1999. 5.4.1, 6.5.1

[53] D-Link. Dcs 932l wireless day/night camera., 2018. URL `https://eu.dlink.com/uk/en/products/dcs-932l-day-night-cloud-camera`. 7.4.2

[54] D-Link. Sound & motion detection., 2018. URL `http://us.dlink.com/features/motion-detection-alerting/`. 2.1, 7.1

[55] Adrian A. de Freitas, Michael Nebeling, Xiang 'Anthony' Chen, Junrui Yang, Akshaye Shreenithi Kirupa Karthikeyan Ranithangam, and Anind K. Dey. 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, pages 5909–5920, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858177. URL `http://doi.acm.org/10.1145/2858036.2858177`. 3.1, 5.2

[56] Morgan Dixon and James Fogarty. Prefab: Implementing advanced behaviors using pixel-based reverse engineering of interface structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1525–1534, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753554. URL `http://doi.acm.org/10.1145/1753326.1753554`. 6.2.1, 6.8.4

[57] Django. The web framework for perfectionists with deadlines, 2018. URL `https://www.djangoproject.com`. 7.4.1

[58] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. Shepherding the crowd yields better work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work*, CSCW '12, pages 1013–1022, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1086-4. doi: 10.1145/2145204.2145355. URL `http://doi.acm.org/10.1145/2145204.2145355`. 7.6.4

[59] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. The pothole patrol: using a mobile sensor network for road surface monitoring. In *Proceedings of the 6th international conference on Mobile systems, applications, and services*, pages 29–39. ACM, 2008. 2.1, 7.1

[60] Facebook. React – a javascript library for building user interfaces, 2018. URL `https://reactjs.org`. 7.4.7

[61] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `http://doi.acm.org/10.1145/358669.358692`. 6.5.1

[62] Martin A. Fischler and Robert C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692. URL `http://doi.acm.org/10.1145/358669.358692`. 5.4.1

[63] Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. Crowddb: Answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference on Management of Data*, SIGMOD '11, page 61–72, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306614. doi: 10.1145/1989323.1989331. URL `https://doi.org/10.1145/1989323.1989331`. 2.3

[64] Jon E. Froehlich, Eric Larson, Tim Campbell, Conor Haggerty, James Fogarty, and Shwetak N. Patel. Hydrosense: Infrastructure-mediated single-point sensing of whole-home water activity. In *Proceedings of the 11th International Conference on Ubiquitous Computing*, UbiComp '09, pages 235–244, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-431-7. doi: 10.1145/1620545.1620581. URL `http://doi.acm.org/10.1145/1620545.1620581`. 7.2.1

[65] Giovanni Fusco, Ender Tekin, Richard E. Ladner, and James M. Coughlan. Using computer vision to access appliance displays. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers & Accessibility*, ASSETS '14, pages 281–282, New York,

NY, USA, 2014. ACM. ISBN 978-1-4503-2720-6. doi: 10.1145/2661334.2661404. URL http://doi.acm.org/10.1145/2661334.2661404. 2.1, 3.1, 3.2.1, 4.2.1, 6.1, 6.2.2

[66] S Garrido-Jurado, Rafael Muñoz-Salinas, Francisco José Madrid-Cuevas, and Manuel Jesús Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, 2014. doi: 10.1016/j.patcog.2014.01.005. URL https://doi.org/10.1016/j.patcog.2014.01.005. 3.6.2

[67] Google. Dialogflow, 2019. URL https://dialogflow.com. 6.5.2

[68] Google Cloud. Cloud vision, 2019. URL https://cloud.google.com/vision/. 6.5.1

[69] Timo Götzelmann. Lucentmaps: 3d printed audiovisual tactile maps for blind and visually impaired people. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '16, pages 81–90, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4124-0. doi: 10.1145/2982142.2982163. URL http://doi.acm.org/10.1145/2982142.2982163. 5.2

[70] Timo Götzelmann and Aleksander Pavkovic. *Towards automatically generated tactile detail maps by 3D printers for blind persons*, pages 1–7. Springer International Publishing, Cham, 2014. ISBN 978-3-319-08599-9. doi: 10.1007/978-3-319-08599-9_1. URL http://dx.doi.org/10.1007/978-3-319-08599-9_1. 5.2

[71] Miriam Greis, Florian Alt, Niels Henze, and Nemanja Memarovic. I can wait a minute: Uncovering the optimal delay time for pre-moderated user-generated content on public displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 1435–1438, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557186. URL http://doi.acm.org/10.1145/2556288.2557186. 7.6.5

[72] Anhong Guo and Tim Paek. Exploring tilt for no-touch, wrist-only interactions on smartwatches. In *Proceedings of the 18th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '16, page 17–28, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450344081. doi: 10.1145/2935334.2935345. URL https://doi.org/10.1145/2935334.2935345. 8.3.2

[73] Anhong Guo, Shashank Raghu, Xuwen Xie, Saad Ismail, Xiaohui Luo, Joseph Simoneau, Scott Gilliland, Hannes Baumann, Caleb Southern, and Thad Starner. A comparison of order picking assisted by head-up display (hud), cart-mounted display (cmd), light, and paper pick list. In *Proceedings of the 2014 ACM International Symposium on Wearable Computers*, ISWC '14, page 71–78, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450329699. doi: 10.1145/2634317.2634321. URL https://doi.org/10.1145/2634317.2634321. 8.3.2

[74] Anhong Guo, Xiang 'Anthony' Chen, and Jeffrey P. Bigham. 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*, CHI EA '15, pages 2043–2048, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3146-3. doi: 10.1145/2702613.2732755. URL http://doi.acm.org/10.1145/

2702613.2732755. 3.4.1, 5.2

[75] Anhong Guo, Robert Xiao, and Chris Harrison. Capauth: Identifying and differentiating user handprints on commodity capacitive touchscreens. In *Proceedings of the 2015 International Conference on Interactive Tabletops & Surfaces*, ITS '15, page 59–62, New York, NY, USA, 2015. Association for Computing Machinery. ISBN 9781450338998. doi: 10.1145/2817721.2817722. URL https://doi.org/10.1145/2817721.2817722. 8.3.2

[76] Anhong Guo, Xiang 'Anthony' Chen, Haoran Qi, Samuel White, Suman Ghosh, Chieko Asakawa, and Jeffrey P. Bigham. Vizlens: A robust and interactive screen reader for interfaces in the real world. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 651–664, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4189-9. doi: 10.1145/2984511.2984518. URL http://doi.acm.org/10.1145/2984511.2984518. 1, 1.1, 4.1, 4.2.3, 4.3.3, 4.4.3, 4.6.3, 5.2, 5.4.1, 5.4.2, 5.7.1, 6.1, 6.2.2, 6.5.1, 6.5.2, 6.6.3, 7.1, 7.2.2, 8.3.1

[77] Anhong Guo, Jeeeun Kim, Xiang 'Anthony' Chen, Tom Yeh, Scott E. Hudson, Jennifer Mankoff, and Jeffrey P. Bigham. Facade: Auto-generating tactile interfaces to appliances. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '16, pages 315–316, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4124-0. doi: 10.1145/2982142.2982187. URL http://doi.acm.org/10.1145/2982142.2982187. 5.2, 8.3.1

[78] Anhong Guo, Jeeeun Kim, Xiang 'Anthony' Chen, Tom Yeh, Scott E. Hudson, Jennifer Mankoff, and Jeffrey P. Bigham. Facade: Auto-generating tactile interfaces to appliances. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 5826–5838, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025845. URL http://doi.acm.org/10.1145/3025453.3025845. 1, 1.1, 6.2.2, 7.2.2

[79] Anhong Guo, Anuraag Jain, Shomiron Ghose, Gierad Laput, Chris Harrison, and Jeffrey P. Bigham. Crowd-ai camera sensing in the real world. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2(3):111:1–111:20, September 2018. ISSN 2474-9567. doi: 10.1145/3264921. URL http://doi.acm.org/10.1145/3264921. 1, 1.2

[80] Anhong Guo, Saige McVea, Xu Wang, Patrick Clary, Ken Goldman, Yang Li, Yu Zhong, and Jeffrey P. Bigham. Investigating cursor-based interactions to support non-visual exploration in the real world. In *Proceedings of the 20th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '18, pages 3–14, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5650-3. doi: 10.1145/3234695.3236339. URL http://doi.acm.org/10.1145/3234695.3236339. 1.1, 6.2.2, 8.3.1

[81] Anhong Guo, Ece Kamar, Jennifer Wortman Vaughan, Hanna Wallach, and Meredith Ringel Morris. Toward fairness in ai for people with disabilities: A research roadmap. *arXiv preprint arXiv:1907.02227*, 2019. 8.3.3

[82] Anhong Guo, Junhan Kong, Michael Rivera, Frank F. Xu, and Jeffrey P. Bigham. Statelens: A reverse engineering solution for making existing dynamic touchscreens acces-

sible. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*, UIST '19, page 371–385, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450368162. doi: 10.1145/3332165.3347873. URL `https://doi.org/10.1145/3332165.3347873`. 1, 1.1, 8.3.1

[83] Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. Electrisense: Single-point sensing using emi for electrical event detection and classification in the home. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, UbiComp '10, pages 139–148, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-843-8. doi: 10.1145/1864349.1864375. URL `http://doi.acm.org/10.1145/1864349.1864375`. 7.2.1

[84] Danna Gurari and Kristen Grauman. Crowdverge: Predicting if people will agree on the answer to a visual question. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3511–3522, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025781. URL `http://doi.acm.org/10.1145/3025453.3025781`. 7.6.3

[85] Danna Gurari, Qing Li, Abigale J. Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P. Bigham. Vizwiz grand challenge: Answering visual questions from blind people. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3608–3617, 2018. 1.1, 2.1, 2.2, 3.2.2, 6.2.2, 7.1, 8.2, 8.3.1, 8.3.3

[86] Danna Gurari, Qing Li, Chi Lin, Yinan Zhao, Anhong Guo, Abigale J. Stangl, and Jeffrey P. Bigham. Vizwiz-priv: A dataset for recognizing the presence and purpose of private visual information in images taken by blind people. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 8.2, 8.3.1, 8.3.3, 8.3.4

[87] Nathan Hahn, Joseph Chang, Ji Eun Kim, and Aniket Kittur. The knowledge accelerator: Big picture thinking in small pieces. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, page 2258–2270, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450333627. doi: 10.1145/2858036.2858364. URL `https://doi.org/10.1145/2858036.2858364`. 2.3

[88] Kotaro Hara, Jin Sun, Robert Moore, David Jacobs, and Jon Froehlich. Tohme: Detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, pages 189–204, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3069-5. doi: 10.1145/2642918.2647403. URL `http://doi.acm.org/10.1145/2642918.2647403`. 2.3, 3.2.2, 7.1, 7.2.2

[89] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 7.4.11

[90] Jaylin Herskovitz, Jason Wu, Samuel White, Amy Pavel, Gabriel Reyes, Anhong Guo, and Jeffrey P. Bigham. Making mobile augmented reality applications accessible. In *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, 2020. 8.3.1

[91] Megan Hofmann, Jeffrey Harris, Scott E. Hudson, and Jennifer Mankoff. Helping hands: Requirements for a prototyping methodology for upper-limb prosthetics users. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 1769–1780, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858340. URL `http://doi.acm.org/10.1145/2858036.2858340`. 5.2

[92] Megan Hofmann, Gabriella Hann, Scott E Hudson, and Jennifer Mankoff. Greater than the sum of its parts: expressing and reusing design intent in 3d models. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, page 301. ACM, 2018. 6.4.1

[93] Ting-Hao Kenneth Huang, Amos Azaria, and Jeffrey P. Bigham. Instructablecrowd: Creating if-then rules via conversations with the crowd. In *Proceedings of the 2016 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, CHI EA '16, pages 1555–1562, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4082-3. doi: 10.1145/2851581.2892502. URL `http://doi.acm.org/10.1145/2851581.2892502`. 7.6.11

[94] Ting-Hao Kenneth Huang, Walter S Lasecki, Amos Azaria, and Jeffrey P Bigham. “is there anything else i can help you with?” challenges in deploying an on-demand crowd-powered conversational agent. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '16. AAAI, 2016. 2.3, 7.2.2

[95] Ting-Hao Kenneth Huang, Yun-Nung Chen, and Jeffrey P. Bigham. Real-time On-Demand Crowd-powered Entity Extraction. *ArXiv e-prints*, April 2017. 7.4.9

[96] Ting-Hao Kenneth Huang, Joseph Chee Chang, and Jeffrey P. Bigham. Evorus: A crowd-powered conversational assistant built to automate itself over time. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, pages 295:1–295:13, New York, NY, USA, 2018. ACM. ISBN 978-1-4503-5620-6. doi: 10.1145/3173574.3173869. URL `http://doi.acm.org/10.1145/3173574.3173869`. 2.3, 7.2.2

[97] Amy Hurst, Scott E. Hudson, and Jennifer Mankoff. Automatically identifying targets users interact with during real world tasks. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10, pages 11–20, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-515-4. doi: 10.1145/1719970.1719973. URL `http://doi.acm.org/10.1145/1719970.1719973`. 6.2.1

[98] InVisionApp Inc. Invision, 2019. URL `https://www.invisionapp.com`. 6.7.1

[99] Chandrika Jayant, Hanjie Ji, Samuel White, and Jeffrey P. Bigham. Supporting blind photography. In *The Proceedings of the 13th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '11, pages 203–210, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0920-2. doi: 10.1145/2049536.2049573. URL `http://doi.acm.org/10.1145/2049536.2049573`. 2.1, 3.2.1, 4.2.2, 5.4.1

[100] Keechul Jung, Kwang In Kim, and Anil K Jain. Text information extraction in images and video: a survey. *Pattern recognition*, 37(5):977–997, 2004. 2.1, 3.2.1

[101] Shaun K. Kane and Jeffrey P. Bigham. Tracking @stemxcomet: Teaching programming to blind students via 3d printing, crisis management, and twitter. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education*, SIGCSE '14, pages 247–252, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2605-6. doi: 10.1145/2538862.2538975. URL `http://doi.acm.org/10.1145/2538862.2538975`. 5.2

[102] Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. Slide rule: Making mobile touch screens accessible to blind people using multi-touch interaction techniques. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '08, pages 73–80, New York, NY, USA, 2008. ACM. ISBN 978-1-59593-976-0. doi: 10.1145/1414471.1414487. URL `http://doi.acm.org/10.1145/1414471.1414487`. 4.2.4, 6.1, 6.5.2

[103] Shaun K. Kane, Brian Frey, and Jacob O. Wobbrock. Access lens: A gesture-based screen reader for real-world documents. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 347–350, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2470704. URL `http://doi.acm.org/10.1145/2470654.2470704`. 2.1, 3.2.1, 4.2.3, 4.3.3, 4.4.3

[104] Shaun K. Kane, Anhong Guo, and Meredith Ringel Morris. Sense and accessibility: Understanding people with physical disabilities' experiences with sensing systems. In *Proceedings of the 22nd International ACM SIGACCESS Conference on Computers and Accessibility*, 2020. 8.3.3

[105] Harmanpreet Kaur, Mitchell Gordon, Yiwei Yang, Jeffrey P Bigham, Jaime Teevan, Ece Kamar, and Walter S Lasecki. Crowdmask: Using crowds to preserve privacy in crowd-powered systems via progressive filtering. In *Proceedings of the AAAI Conference on Human Computation and Crowdsourcing*, HCOMP '17. AAAI, 2017. 7.6.13

[106] Jeeeun Kim and Tom Yeh. Toward 3d-printed movable tactile pictures for children with visual impairments. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 2815–2824, 2015. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702144. URL `http://doi.acm.org/10.1145/2702123.2702144`. 5.2

[107] Juho Kim, Phu Tran Nguyen, Sarah Weir, Philip J. Guo, Robert C. Miller, and Krzysztof Z. Gajos. Crowdsourcing step-by-step information extraction to enhance existing how-to videos. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pages 4017–4026, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2556986. URL `http://doi.acm.org/10.1145/2556288.2556986`. 6.2.1

[108] Neil Klingensmith, Joseph Bomber, and Suman Banerjee. Hot, cold and in between: Enabling fine-grained environmental control in homes for efficiency and comfort. In *Proceedings of the 5th International Conference on Future Energy Systems*, e-Energy '14, pages 123–132, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2819-7. doi: 10.1145/2602044.2602049. URL `http://doi.acm.org/10.1145/2602044.2602049`. 7.2.1

[109] KNFB Reader. KNFB Reader. http://www.knfbreader.com/, 2019. 2.1, 3.2.1, 4.2.1

[110] Junhan Kong, Anhong Guo, and Jeffrey P. Bigham. Supporting older adults in using complex user interfaces with augmented reality. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '19, page 661–663, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366762. doi: 10.1145/3308561.3354593. URL https://doi.org/10.1145/3308561.3354593. 6.8.2, 8.3.2

[111] Junhan Judy Kong. An authoring tool for creating interactive ar user tutorials by demonstration. Technical Report CMU-CS-20-116, School of Computer Science, Carnegie Mellon University, 2020. 8.3.2

[112] Stacey Kuznetsov and Eric Paulos. Upstream: Motivating water conservation with low-cost water flow sensing and persuasive displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 1851–1860, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi: 10.1145/1753326.1753604. URL http://doi.acm.org/10.1145/1753326.1753604. 7.2.1

[113] Richard E. Ladner, Melody Y. Ivory, Rajesh Rao, Sheryl Burgstahler, Dan Comden, Sangyun Hahn, Matthew Renzelmann, Satria Krisnandi, Mahalakshmi Ramasamy, Beverly Slabosky, Andrew Martin, Amelia Lacenski, Stuart Olsen, and Dmitri Groce. Automating tactile graphics translation. In *Proceedings of the 7th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '05, pages 150–157, New York, NY, USA, 2005. ACM. ISBN 1-59593-159-7. doi: 10.1145/1090785.1090814. URL http://doi.acm.org/10.1145/1090785.1090814. 3.8

[114] Benjamin Lafreniere, Tovi Grossman, and George Fitzmaurice. Community enhanced tutorials: Improving tutorials with multiple demonstrations. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pages 1779–1788, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1899-0. doi: 10.1145/2470654.2466235. URL http://doi.acm.org/10.1145/2470654.2466235. 6.8.2

[115] Benjamin Lafreniere, Tovi Grossman, Fraser Anderson, Justin Matejka, Heather Kerrick, Danil Nagy, Lauren Vasey, Evan Atherton, Nicholas Beirne, Marcelo H. Coelho, Nicholas Cote, Steven Li, Andy Nogueira, Long Nguyen, Tobias Schwinn, James Stoddart, David Thomasson, Ray Wang, Thomas White, David Benjamin, Maurice Conti, Achim Menges, and George Fitzmaurice. Crowdsourced fabrication. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 15–28, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4189-9. doi: 10.1145/2984511.2984553. URL http://doi.acm.org/10.1145/2984511.2984553. 5.2

[116] Gierad Laput, Walter S. Lasecki, Jason Wiese, Robert Xiao, Jeffrey P. Bigham, and Chris Harrison. Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pages 1935–1944, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702416. URL http://doi.acm.org/10.1145/2702123.2702416. 2.3, 3.2.2, 3.4.2, 7.1, 7.2.1, 7.2.2

[117] Gierad Laput, Yang Zhang, and Chris Harrison. Synthetic sensors: Towards general-purpose sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 3986–3999, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025773. URL `http://doi.acm.org/10.1145/3025453.3025773`. 7.2.1

[118] Walter Lasecki, Christopher Miller, Adam Sadilek, Andrew Abumoussa, Donato Borrello, Raja Kushalnagar, and Jeffrey Bigham. Real-time captioning by groups of non-experts. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology*, UIST '12, pages 23–34, New York, NY, USA, 2012. ACM. ISBN 978-1-4503-1580-7. doi: 10.1145/2380116.2380122. URL `http://doi.acm.org/10.1145/2380116.2380122`. 2.2, 3.2.2

[119] Walter S. Lasecki, Kyle I. Murray, Samuel White, Robert C. Miller, and Jeffrey P. Bigham. Real-time crowd control of existing interfaces. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pages 23–32, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0716-1. doi: 10.1145/2047196.2047200. URL `http://doi.acm.org/10.1145/2047196.2047200`. 2.2, 3.2.2

[120] Walter S. Lasecki, Young Chol Song, Henry Kautz, and Jeffrey P. Bigham. Real-time crowd labeling for deployable activity recognition. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work*, CSCW '13, pages 1203–1212, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1331-5. doi: 10.1145/2441776.2441912. URL `http://doi.acm.org/10.1145/2441776.2441912`. 2.3, 7.2.2

[121] Walter S. Lasecki, Phyo Thiha, Yu Zhong, Erin Brady, and Jeffrey P. Bigham. Answering visual questions with conversational crowd assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '13, pages 18:1–18:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2405-2. doi: 10.1145/2513383.2517033. URL `http://doi.acm.org/10.1145/2513383.2517033`. 2.2, 3.2.2, 5.2, 6.2.2, 7.2.2

[122] Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. Chorus: A crowd-powered conversational assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, pages 151–162, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2268-3. doi: 10.1145/2501988.2502057. URL `http://doi.acm.org/10.1145/2501988.2502057`. 2.2, 3.2.2, 7.2.2

[123] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. Sugilite: Creating multimodal smartphone automation by demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6038–6049, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025483. URL `http://doi.acm.org/10.1145/3025453.3025483`. 6.2.1

[124] Yang Li, Xiang Cao, Katherine Everitt, Morgan Dixon, and James A. Landay. Framewire: A tool for automatically extracting interaction logic from paper prototyping tests. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pages 503–512, New York, NY, USA, 2010. ACM. ISBN 978-1-60558-929-9. doi:

10.1145/1753326.1753401. URL http://doi.acm.org/10.1145/1753326.1753401. 6.2.1

[125] LookTel Money Reader. LookTel Money Reader. http://www.looktel.com/moneyreader, 2018. 2.1, 4.2.1

[126] LookTel Recognizer. LookTel Recognizer. http://www.looktel.com/recognizer, 2018. 2.1, 4.2.1

[127] David Lowe. The computer vision industry., 2015. URL https://www.cs.ubc.ca/~lowe/vision.html. 2.1, 7.1

[128] Roberto Manduchi and James M. Coughlan. The last meter: Blind visual guidance to a target. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, pages 3113–3122, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2473-1. doi: 10.1145/2556288.2557328. URL http://doi.acm.org/10.1145/2556288.2557328. 2.1, 3.2.1, 4.2.2, 4.4.3, 5.4.1

[129] Mashable. Omoby: Visual search for the iphone., 2010. URL https://mashable.com/2010/03/04/omoby-visual-search-iphone. 7.2.2

[130] Samantha McDonald, Joshua Dutterer, Ali Abdolrahmani, Shaun K. Kane, and Amy Hurst. Tactile aids for visually impaired graphical design education. In *Proceedings of the 16th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '14, pages 275–276, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2720-6. doi: 10.1145/2661334.2661392. URL http://doi.acm.org/10.1145/2661334.2661392. 5.2

[131] Samantha McDonald, Niara Comrie, Erin Buehler, Nicholas Carter, Braxton Dubin, Karen Gordes, Sandy McCombe-Waller, and Amy Hurst. Uncovering challenges and opportunities for 3d printing assistive technology with physical therapists. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '16, pages 131–139, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4124-0. doi: 10.1145/2982142.2982162. URL http://doi.acm.org/10.1145/2982142.2982162. 5.2

[132] Meridian Outpost. What are the limitations of OCR? http://www.meridianoutpost.com/resources/articles/ocr-limitations.php, 2019. 2.1, 3.2.1

[133] Microsoft. Seeing AI. https://www.microsoft.com/en-us/seeing-ai, 2019. 2.1, 3.2.1, 4.2.1

[134] George A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological review*, 63(2):81, 1956. doi: 10.1037/h0043158. 5.3.2

[135] Thomas B Moeslund and Erik Granum. A survey of computer vision-based human motion capture. *Computer vision and image understanding*, 81(3):231–268, 2001. 2.1, 7.1

[136] Shunji Mori, Hirobumi Nishida, and Hiromitsu Yamada. *Optical Character Recognition*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1999. ISBN 0471308196. 4.2.1

[137] T. Morris, P. Blenkhorn, L. Crossey, Q. Ngo, M. Ross, D. Werner, and C. Wong. Clear-

speech: A display reader for the visually handicapped. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 14(4):492–500, Dec 2006. ISSN 1534-4320. doi: 10.1109/TNSRE.2006.881538. 2.1, 3.1, 3.2.1, 4.2.1, 6.1, 6.2.2

[138] Marius Muja and David G. Lowe. Fast approximate nearest neighbors with automatic algorithm configuration. In *International Conference on Computer Vision Theory and Application VISSAPP'09)*, pages 331–340. INSTICC Press, 2009. 5.4.1

[139] Suranga Nanayakkara, Roy Shilkrot, Kian Peen Yeo, and Pattie Maes. Eyering: A finger-worn input device for seamless interactions with our surroundings. In *Proceedings of the 4th Augmented Human International Conference*, AH '13, pages 13–20, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-1904-1. doi: 10.1145/2459236.2459240. URL `http://doi.acm.org/10.1145/2459236.2459240`. 3.2.1, 4.2.3

[140] Nest. Nest aware., 2018. URL `https://nest.com/cameras/nest-aware/`. 2.1, 7.1

[141] Dat Tien Nguyen, Firoj Alam, Ferda Ofli, and Muhammad Imran. Automatic image filtering on social networks using deep learning and perceptual hashing during crises. *arXiv preprint arXiv:1704.02602*, 2017. 7.6.6

[142] Jeffrey Nichols, Brad A. Myers, Michael Higgins, Joseph Hughes, Thomas K. Harris, Roni Rosenfeld, and Mathilde Pignol. Generating remote control interfaces for complex appliances. In *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, UIST '02, pages 161–170, New York, NY, USA, 2002. ACM. ISBN 1-58113-488-6. doi: 10.1145/571985.572008. URL `http://doi.acm.org/10.1145/571985.572008`. 5.2

[143] NinjaTek. NinjaFlex. `https://ninjatek.com/ninjaflex/`, 2019. 5.5.2

[144] National Federation of the Blind. The braille literacy crisis in america: Facing the truth, reversing the trend, empowering the blind. National Federation of the Blind, Jernigan Institute, March 2009. 3.3

[145] OrCam. OrCam. `http://www.orcam.com`, 2019. 2.1, 3.2.1, 4.2.1, 4.2.3, 4.3.3, 4.6.4

[146] Sujeath Pareddy, Anhong Guo, and Jeffrey P. Bigham. X-ray: Screenshot accessibility via embedded metadata. In *The 21st International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '19, page 389–395, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366762. doi: 10.1145/3308561.3353808. URL `https://doi.org/10.1145/3308561.3353808`. 8.3.1

[147] Shwetak N. Patel, Matthew S. Reynolds, and Gregory D. Abowd. *Detecting Human Movement by Differential Air Pressure Sensing in HVAC System Ductwork: An Exploration in Infrastructure Mediated Sensing*, pages 1–18. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-79576-6. doi: 10.1007/978-3-540-79576-6_1. URL `https://doi.org/10.1007/978-3-540-79576-6_1`. 7.2.1

[148] Andrea Polesel, Giovanni Ramponi, and V John Mathews. Image enhancement via adaptive unsharp masking. *IEEE transactions on image processing*, 9(3):505–510, 2000. 3.7.2

[149] Raf Ramakers, Fraser Anderson, Tovi Grossman, and George Fitzmaurice. Retrofab: A design tool for retrofitting physical interfaces using actuators, sensors and 3d printing. In

Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems, CHI '16, pages 409–419, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858485. URL http://doi.acm.org/10.1145/2858036.2858485. 5.2, 6.2.2

[150] Daniela Retelny, Sébastien Robaszkiewicz, Alexandra To, Walter S. Lasecki, Jay Patel, Negar Rahmati, Tulsee Doshi, Melissa Valentine, and Michael S. Bernstein. Expert crowdsourcing with flash teams. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*, UIST '14, page 75–85, New York, NY, USA, 2014. Association for Computing Machinery. ISBN 9781450330695. doi: 10.1145/2642918.2647409. URL https://doi.org/10.1145/2642918.2647409. 2.3

[151] Akash Das Sarma, Ayush Jain, Arnab Nandi, Aditya Parameswaran, and Jennifer Widom. Surpassing humans and computers with jellybean: Crowd-vision-hybrid counting algorithms. In *Third AAAI Conference on Human Computation and Crowdsourcing*, 2015. 2.3

[152] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823, 2015. 2.1, 7.1

[153] James Scott, A.J. Bernheim Brush, John Krumm, Brian Meyers, Michael Hazas, Stephen Hodges, and Nicolas Villar. Preheat: Controlling home heating using occupancy prediction. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 281–290, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0630-0. doi: 10.1145/2030112.2030151. URL http://doi.acm.org/10.1145/2030112.2030151. 7.2.1

[154] Lei Shi, Idan Zelzer, Catherine Feng, and Shiri Azenkot. Tickers and talker: An accessible labeling toolkit for 3d printed models. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 4896–4907, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858507. URL http://doi.acm.org/10.1145/2858036.2858507. 5.2

[155] Lei Shi, Yuhang Zhao, and Shiri Azenkot. Markit and talkit: A low-barrier toolkit to augment 3d printed models with audio annotations. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*, UIST '17, pages 493–506, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4981-9. doi: 10.1145/3126594.3126650. URL http://doi.acm.org/10.1145/3126594.3126650. 4.2.3

[156] Roy Shilkrot, Jochen Huber, Connie Liu, Pattie Maes, and Suranga Chandima Nanayakkara. Fingerreader: A wearable device to support text reading on the go. In *CHI '14 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '14, pages 2359–2364, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2474-8. doi: 10.1145/2559206.2581220. URL http://doi.acm.org/10.1145/2559206.2581220. 3.2.1, 4.2.3

[157] Kristen Shinohara and Josh Tenenberg. Observing sara: A case study of a blind person's interactions with technology. In *Proceedings of the 9th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '07, pages 171–178, New York, NY,

USA, 2007. ACM. ISBN 978-1-59593-573-1. doi: 10.1145/1296843.1296873. URL http://doi.acm.org/10.1145/1296843.1296873. 4.1

[158] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings 1996 IEEE Symposium on Visual Languages*, pages 336–343, Sep 1996. doi: 10.1109/VL.1996.545307. 4.6.3

[159] Saiganesh Swaminathan, Thijs Roumen, Robert Kovacs, David Stangl, Stefanie Mueller, and Patrick Baudisch. Linespace: A sensemaking platform for the blind. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pages 2175–2185, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-3362-7. doi: 10.1145/2858036.2858245. URL http://doi.acm.org/10.1145/2858036.2858245. 5.2

[160] Amanda Swearngin and Yang Li. Modeling mobile interface tappability using crowdsourcing and deep learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, CHI '19, New York, NY, USA, 2019. ACM. ISBN 978-1-4503-5970-2. doi: 10.1145/3290605.3300305. URL https://doi.org/10.1145/3290605.3300305. 6.8.4

[161] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. *Activity Recognition in the Home Using Simple and Ubiquitous Sensors*, pages 158–175. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. ISBN 978-3-540-24646-6. doi: 10.1007/978-3-540-24646-6_10. URL https://doi.org/10.1007/978-3-540-24646-6_10. 7.2.1

[162] Brandon Taylor, Anind Dey, Dan Siewiorek, and Asim Smailagic. Customizable 3d printed tactile maps as interactive overlays. In *Proceedings of the 18th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '16, pages 71–79, New York, NY, USA, 2016. ACM. ISBN 978-1-4503-4124-0. doi: 10.1145/2982142.2982167. URL http://doi.acm.org/10.1145/2982142.2982167. 5.2

[163] Ender Tekin, James M. Coughlan, and Huiying Shen. Real-time detection and reading of led/lcd displays for visually impaired persons. In *Proceedings of the 2011 IEEE Workshop on Applications of Computer Vision (WACV)*, WACV '11, pages 491–496, Washington, DC, USA, 2011. IEEE Computer Society. ISBN 978-1-4244-9496-5. doi: 10.1109/WACV.2011.5711544. URL http://dx.doi.org/10.1109/WACV.2011.5711544. 2.1, 3.2.1, 4.2.1, 6.2.2

[164] Tesseract OCR. Tesseract OCR. https://github.com/tesseract-ocr/tesseract, 2019. 3.7.2

[165] J. Thatcher. 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, pages 39–46, New York, NY, USA, 1994. ACM. ISBN 0-89791-649-2. doi: 10.1145/191028.191039. URL http://doi.acm.org/10.1145/191028.191039. 3.1

[166] Matthew A Turk and Alex P Pentland. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*, pages 586–591. IEEE, 1991. 2.1, 7.1

[167] United States Access Board. Advancing full access and inclusion for all. https://www.access-board.gov/guidelines-and-standards, 2016. 5.4.3

[168] V7 Ltd. Aipoly. https://www.aipoly.com, 2019. 2.1, 3.2.1, 4.2.1

[169] Gregg Vanderheiden and Jutta Treviranus. Creating a global public inclusive infrastructure. In *Proceedings of the 6th International Conference on Universal Access in Human-computer Interaction: Design for All and eInclusion - Volume Part I*, UAHCI'11, pages 517–526, Berlin, Heidelberg, 2011. Springer-Verlag. ISBN 978-3-642-21671-8. URL http://dl.acm.org/citation.cfm?id=2022591.2022652. 3.1, 5.2, 6.1

[170] Gregg C. Vanderheiden. Flexible access system for touch screen devices, April 11 2000. US Patent 6,049,328. 6.1

[171] Marynel Vázquez and Aaron Steinfeld. An assisted photography framework to help visually impaired users properly aim a camera. *ACM Transactions on Computer-Human Interaction*, 21(5):25:1–25:29, November 2014. ISSN 1073-0516. doi: 10.1145/2651380. URL http://doi.acm.org/10.1145/2651380. 2.1, 3.2.1, 4.2.2, 5.4.1

[172] David Vernon. Machine vision-automated visual inspection and robot vision. *NASA STI/Recon Technical Report A*, 92, 1991. 2.1, 7.1

[173] Vladimir Vezhnevets, Vassili Sazonov, and Alla Andreeva. A survey on pixel-based skin color detection techniques. In *Proc. Graphicon*, volume 3, pages 85–92. Moscow, Russia, 2003. 3.4.3, 6.5.1

[174] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision*, 101(1):184–204, 2013. 2.3, 7.2.2

[175] Xu Wang, Benjamin Lafreniere, and Tovi Grossman. Leveraging community-generated videos and command logs to classify and recommend software workflows. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2018. 6.8.2

[176] Samuel White, Hanjie Ji, and Jeffrey P. Bigham. Easysnap: Real-time audio feedback for blind photography. In *Adjunct Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pages 409–410, New York, NY, USA, 2010. ACM. ISBN 978-1-4503-0462-7. doi: 10.1145/1866218.1866244. URL http://doi.acm.org/10.1145/1866218.1866244. 2.1, 3.2.1, 4.2.2, 5.4.1

[177] D. H. Wilson and C. Atkeson. *Simultaneous Tracking and Activity Recognition (STAR) Using Many Anonymous, Binary Sensors*, pages 62–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005. ISBN 978-3-540-32034-0. doi: 10.1007/11428572_5. URL https://doi.org/10.1007/11428572_5. 7.2.1

[178] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli: Using gui screenshots for search and automation. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pages 183–192, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-745-5. doi: 10.1145/1622176.1622213. URL http://doi.acm.org/10.1145/1622176.1622213. 6.2.1

[179] Christoph Zauner. Implementation and benchmarking of perceptual image hash functions. 2010. 7.4.6

[180] M. Zeifman and K. Roth. Nonintrusive appliance load monitoring: Review and outlook.

IEEE Transactions on Consumer Electronics, 57(1):76–84, February 2011. ISSN 0098-3063. doi: 10.1109/TCE.2011.5735484. 7.2.1

[181] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503, 2016. 7.4.5

[182] Xiaoyi Zhang, Anne Spencer Ross, Anat Caspi, James Fogarty, and Jacob O. Wobbrock. Interaction proxies for runtime repair and enhancement of mobile application accessibility. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, pages 6024–6037, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4655-9. doi: 10.1145/3025453.3025846. URL `http://doi.acm.org/10.1145/3025453.3025846`. 6.2.1

[183] Yuhang Zhao, Sarit Szpiro, and Shiri Azenkot. 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, pages 239–249, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3400-6. doi: 10.1145/2700648.2809865. URL `http://doi.acm.org/10.1145/2700648.2809865`. 3.2.1

[184] Yu Zhong, Pierre J. Garrigues, and Jeffrey P. Bigham. 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*, ASSETS '13, pages 20:1–20:8, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2405-2. doi: 10.1145/2513383.2513443. URL `http://doi.acm.org/10.1145/2513383.2513443`. 2.1, 3.2.1, 4.2.2, 5.4.1

[185] Yu Zhong, Walter S. Lasecki, Erin Brady, and Jeffrey P. Bigham. 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*, CHI '15, pages 2353–2362, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3145-6. doi: 10.1145/2702123.2702437. URL `http://doi.acm.org/10.1145/2702123.2702437`. 2.3, 3.2.2, 3.4.2, 3.5.4, 3.7, 3.8, 4.1, 4.2.4, 4.3.4, 4.4.3, 4.6.3, 5.2, 6.2.2