Crowd-AI Camera Sensing in the Real World

ANHONG GUO, Human-Computer Interaction Institute, Carnegie Mellon University, USA
ANURAAG JAIN, Human-Computer Interaction Institute, Carnegie Mellon University, USA
SHOMIRON GHOSE, Human-Computer Interaction Institute, Carnegie Mellon University, USA
GIERAD LAPUT, Human-Computer Interaction Institute, Carnegie Mellon University, USA
CHRIS HARRISON, Human-Computer Interaction Institute, Carnegie Mellon University, USA
JEFFREY P. BIGHAM, Human-Computer Interaction Institute, Carnegie Mellon University, USA

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 crowd- 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 crowd-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 crowd-AI vision systems.

CCS Concepts: • **Human-centered computing → Ubiquitous and mobile computing systems and tools**;

Additional Key Words and Phrases: Smart environments; Internet of things; deployment; camera; sensing; crowdsourcing; human computation; computer vision; machine learning

ACM Reference Format:
Anhong Guo, Anuraag Jain, Shomiron Ghose, Gierad Laput, Chris Harrison, and Jeffrey P. Bigham. 2018. Crowd-AI Camera Sensing in the Real World. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 2, 3, Article 111 (September 2018), 20 pages.
https://doi.org/10.1145/3264921

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.

Authors' addresses: Anhong Guo, anhongg@cs.cmu.edu; Anuraag Jain, anuraagjain@gmail.com; Shomiron Ghose, ronnie.ghose@gmail.com; Gierad Laput, gierad.laput@cs.cmu.edu; Chris Harrison, chris.harrison@cs.cmu.edu; Jeffrey P. Bigham, jbigham@cs.cmu.edu. Human-Computer Interaction Institute, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, PA, USA 15213.

2474-9567/2018/9-ART111 $15.00
https://doi.org/10.1145/3264921

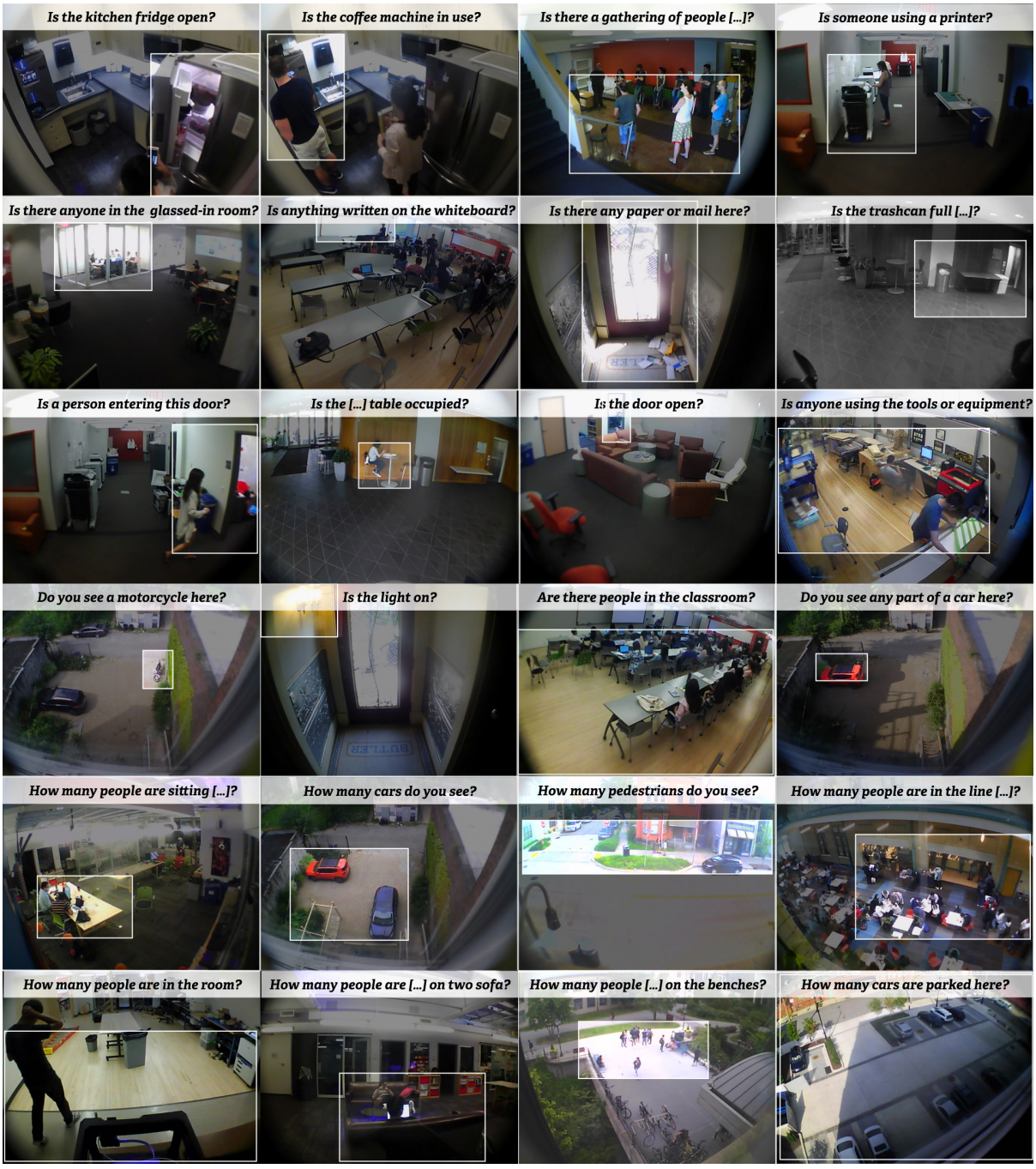

Fig. 1. Example question sensors created by our participants, with regions of interest highlighted on the full camera image.

This trend is slowly changing with consumer home cameras offering rudimentary computationally-enhanced functions, such as motion [24] and intruder detection [53]. Perhaps most sophisticated among these consumer offerings is the Amazon Echo Look, which can offer fashion advice [1]. In commercial and municipal camera systems [50], computer vision has been applied to *e.g.*, count cars and people [20, 52], read license plates [21], control quality [60], analyze sports [13], recognize faces [59] and monitor road surfaces [27]. In general, these computer-vision-powered systems require extensive training data and on-site tuning to work well. For example, FaceNet [56] 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 [15, 18, 35].

To flexibly adapt to new questions, researchers have created hybrid crowd- and artificial intelligence (AI)-powered computer vision systems [19, 31, 36, 45]. 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 [45]. However, prior work falls short of real-world deployment, leaving significant questions about the feasibility of such crowd-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 crowd-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 crowd-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 crowd-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 [45], VizWiz [16] and VizLens [31]). 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 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 crowd-AI camera-based sensing. We synthesize our findings, which we hope will inform future work in this area.

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.

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 [44] and WaterBot [12] 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 [43] and occupancy [57] 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) [58, 62]. 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 [33, 64], plumbing [29], HVACs [55]), 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 [45, 46], where a single, highly-capable sensor can detect a wide range of events within a room.

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.*, [16, 31, 47]).

Crowdsourcing systems access "human intelligence" through online marketplaces such as Amazon Mechanical Turk [2]. Prior work in general-purpose, visual sensing has relied entirely on crowdsourced answers. For instance, VizWiz [16] 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 [39, 41, 48, 49] and promoting accessibility [31, 32, 36]. Unlike VizWiz and OMoby [51], 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 [47]. Likewise, Flock [19] 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 [61] 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 [45], 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 crowd-AI visual sensing systems.

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.

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.

4.1 Compute

We deployed Zensors++ on Amazon Web Services (AWS) [8], implemented as a stateless Django [25] 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) [9] accelerated by ElastiCache [3]. 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.

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 [23] (Figure 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 2B). To register a new camera, our web frontend generates unique FTP credentials that bind a camera to a user's account.

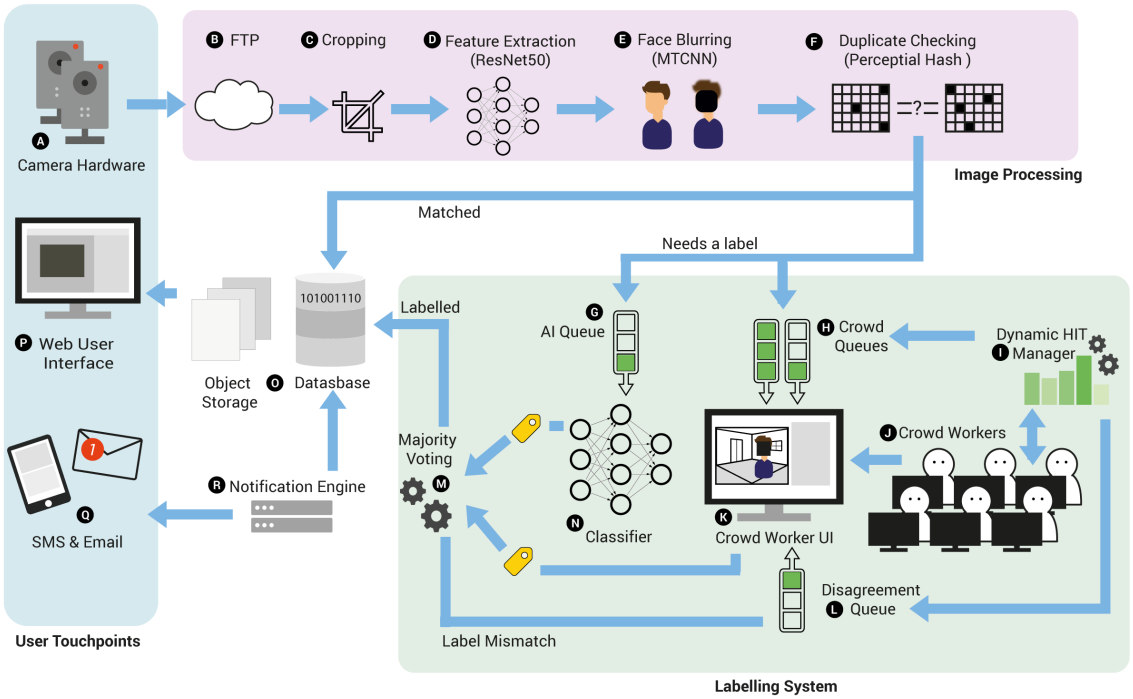

Fig. 2. The system architecture of Zensors++ consists of four main components: user touchpoints (blue), image processing pipeline (pink), labeling system (green), and data storage (grey).

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).

4.3 Question Sensor Authoring

Our web interface (Figure 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 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 3C).

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

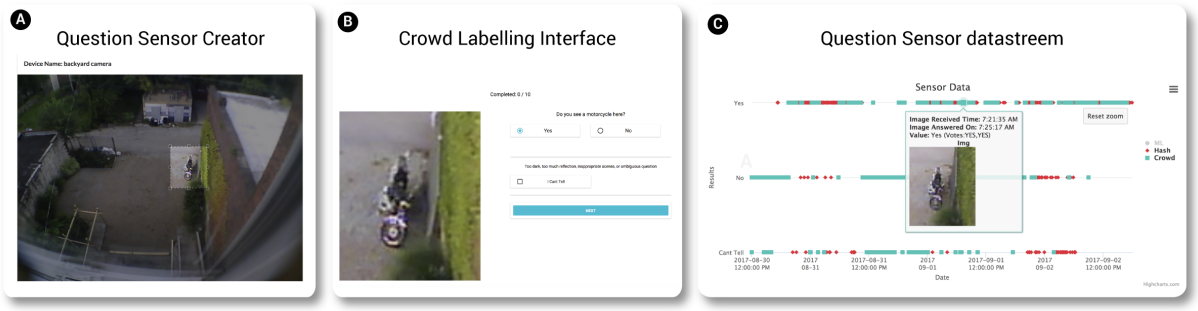

Fig. 3. Screenshots of user interfaces. (A) End users highlight a region of interest in a camera image and add an associated natural language question. (B) Crowd workers provide answers for user-defined "question sensors" given an image region of interest. (C) End users can view real-time and historical data through visualizations.

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 [11] in-memory data structure store on AWS to cache the recent data points for scalability. We use Simple Email Service (SES) [5] for sending email notifications, and Simple Notification Service (SNS) [6] for sending text messages (Figure 2Q).

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 [65] to obscure faces with a black box (Figure 2E). Third, only small regions of interest are shown to crowd workers, allowing users to selectively reveal areas of their environment (Figure 2C).

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 [63] (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 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.

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 3B), which forced a more deliberate selection.

Our crowd interface was implemented using the React JavaScript library [28] (Figure 2K) and we recruited workers from Amazon Mechanical Turk (Figure 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.

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 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 [10]. 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) [7]. A question sensor instance is first inserted into two "work queues" (Figure 2H) and then onto a third "disagreement queue" (Figure 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.

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 [40]. 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 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 [14, 16] in the event that the main queues run out of items or in cases of surplus labeler capacity.

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.

4.11 Machine Learning

Our machine learning pipeline (Figure 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 [37] (Figure 2D). This feature vector is stored for training and also used as input to a classifier (Figure 2N) that predicts an answer for a question sensor instance. These embeddings serve as input to a KNN [22] classifier for yes/no questions and a KNN regressor for count questions.

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 4.

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 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 1 provides visual examples of 24 participant-defined question sensors; Figure 4 provides a compact, but full listing.

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.

6.1 Scale (RQ2)

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 [4]).

6.2 Raw Crowd Accuracy (RQ2)

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 4.

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**		

Fig. 4. Detailed statistics on all 63 question sensors from our evaluation deployment.

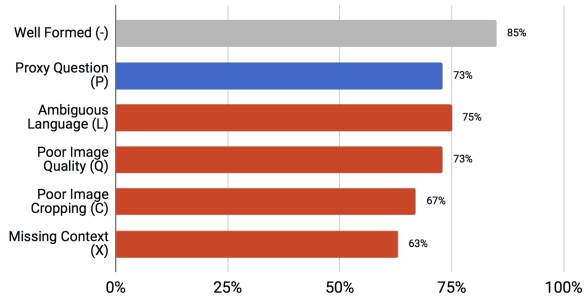

Fig. 5. Yes/No question sensor accuracy when well formed (gray bar), formulated as a proxy question (blue bar), and when exhibiting four different error types (red bars).

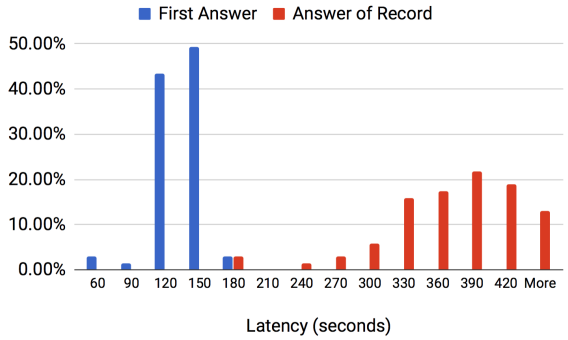

Fig. 6. Histogram of end-to-end crowd latency. Median time to first answer was 120 seconds, while median time to answer of record was 355 seconds.

6.3 Question Sensor Accuracy (RQ2)

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 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 [34] 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 4.

6.4 User-Defined Question Errors (RQ2)

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 4. Overall, yes/no questions with no discernible problems had a mean accuracy of 85%, shown as the gray bar in Figure 5. On the other hand, question sensors marked with any error type had a mean accuracy of 71% (red bars in Figure 5).

The most frequent problem (13 of 63 sensors) was poor image quality (error label Q in Figure 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 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 4) was "is the meeting room free (is the light on)?", which was confusing to both our crowd and 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 [26], suggest superior phrasing, or show crowd workers the full camera image with region of interest highlighted.

6.5 Crowd Latency (RQ2)

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 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 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 latency and accuracy (similar to the public information display in [30]).

6.6 Hashing Performance (RQ2)

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), 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 [54].

6.7 Cost (RQ2)

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 labor, for an average cost per answer of $0.006. As discussed previously, we utilized a majority voting scheme that started with two crowd

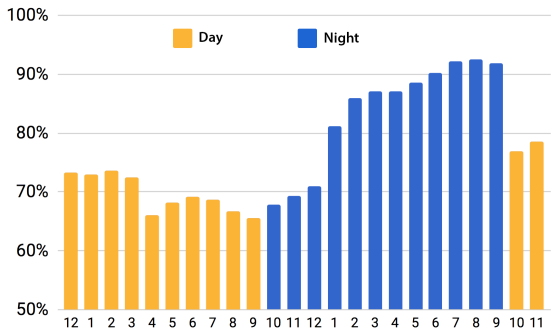

Fig. 7. Mean perceptual hash hit rate over 24 hours, starting at noon, far left.

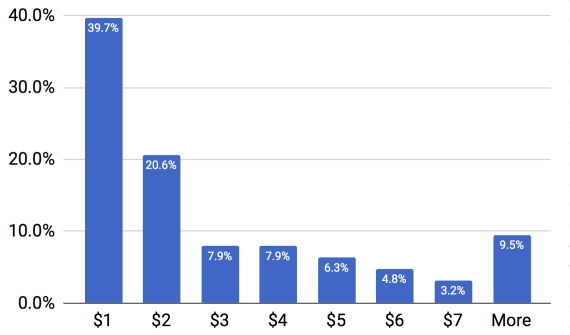

Fig. 8. Distribution of daily cost across all question sensors without machine learning. More than 60% of question sensors cost $2/day or less to operate.

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 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 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 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 crowd-AI systems for many use cases.

6.8 Machine Learning Accuracy (RQ2)

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 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.

6.9 Participant Use of Data (RQ3)

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.

6.10 Types of Question Sensors (RQ3)

Participant question sensors (listed in Figure 4, examples shown in Figure 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.

6.11 Proxy Questions (RQ3)

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 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 [38].

6.12 Perceived Value (RQ4)

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.

6.13 Privacy and Sharing (RQ4)

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 [17].

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 [42], 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 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.

8 CONCLUSION

We have described our work on Zensors++, a crowd-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 crowd-AI, camera-based sensing at scale, and offer a number of insights for those wishing to deploy robust and responsive systems in the future.

ACKNOWLEDGMENTS

This research was funded in part by gifts from Bosch and Google. We thank all our study participants and workers on Amazon Mechanical Turk who contributed to our studies. We thank the reviewers for their valuable feedback and suggestions. We also thank Xu Wang, and members of the Big Lab and FIGLAB for their help and support.

REFERENCES

[1] Amazon. 2018. Echo Look | Hands-Free Camera and Style Assistant with Alexa–includes Style Check to get a second opinion on your outfit. (2018). Retrieved August 8, 2018 from https://www.amazon.com/Amazon-Echo-Look-Camera-Style-Assistant/dp/B0186JAEWK

[2] Amazon Mechanical Turk. 2018. Human intelligence through an API. (2018). Retrieved August 8, 2018 from https://www.mturk.com

[3] Amazon Web Services. 2018. Amazon ElastiCache. (2018). Retrieved August 8, 2018 from https://aws.amazon.com/elasticache/
[4] Amazon Web Services. 2018. Amazon S3. (2018). Retrieved August 8, 2018 from https://aws.amazon.com/s3/
[5] Amazon Web Services. 2018. Amazon Simple Email Service. (2018). Retrieved August 8, 2018 from https://aws.amazon.com/ses/
[6] Amazon Web Services. 2018. Amazon Simple Notification Service (SNS). (2018). Retrieved August 8, 2018 from https://aws.amazon.com/sns/
[7] Amazon Web Services. 2018. Amazon Simple Queue Service. (2018). Retrieved August 8, 2018 from https://aws.amazon.com/sqs/
[8] Amazon Web Services. 2018. Amazon Web Services (AWS) – Cloud Computing Services. (2018). Retrieved August 8, 2018 from https://aws.amazon.com
[9] Amazon Web Services. 2018. Elastic Load Balancing. (2018). Retrieved August 8, 2018 from https://aws.amazon.com/elasticloadbalancing/
[10] Amazon Web Services. 2018. Message Queues. (2018). Retrieved August 8, 2018 from https://aws.amazon.com/message-queue/
[11] Amazon Web Services. 2018. Redis. (2018). Retrieved August 8, 2018 from https://aws.amazon.com/redis/
[12] Ernesto Arroyo, Leonardo Bonanni, and Ted Selker. 2005. Waterbot: Exploring Feedback and Persuasive Techniques at the Sink. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '05)*. ACM, New York, NY, USA, 631–639. https://doi.org/10.1145/1054972.1055059
[13] Sian Barris and Chris Button. 2008. A review of vision-based motion analysis in sport. *Sports Medicine* 38, 12 (2008), 1025–1043.
[14] Michael S. Bernstein, Joel Brandt, Robert C. Miller, and David R. Karger. 2011. 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)*. ACM, New York, NY, USA, 33–42. https://doi.org/10.1145/2047196.2047201
[15] Michael S. Bernstein, Jaime Teevan, Susan Dumais, Daniel Liebling, and Eric Horvitz. 2012. Direct Answers for Search Queries in the Long Tail. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '12)*. ACM, New York, NY, USA, 237–246. https://doi.org/10.1145/2207676.2207710
[16] 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. 2010. VizWiz: Nearly Real-time Answers to Visual Questions. In *Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 333–342. https://doi.org/10.1145/1866029.1866080
[17] Michael Boyle, Christopher Edwards, and Saul Greenberg. 2000. The Effects of Filtered Video on Awareness and Privacy. In *Proceedings of the 2000 ACM Conference on Computer Supported Cooperative Work (CSCW '00)*. ACM, New York, NY, USA, 1–10. https://doi.org/10.1145/358916.358935
[18] Erin Brady, Meredith Ringel Morris, Yu Zhong, Samuel White, and Jeffrey P. Bigham. 2013. Visual Challenges in the Everyday Lives of Blind People. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2117–2126. https://doi.org/10.1145/2470654.2481291
[19] Justin Cheng and Michael S. Bernstein. 2015. Flock: Hybrid Crowd-Machine Learning Classifiers. In *Proceedings of the 18th ACM Conference on Computer Supported Cooperative Work & Social Computing (CSCW '15)*. ACM, New York, NY, USA, 600–611. https://doi.org/10.1145/2675133.2675214
[20] Benjamin Coifman, David Beymer, Philip McLauchlan, and Jitendra Malik. 1998. A real-time computer vision system for vehicle tracking and traffic surveillance. *Transportation Research Part C: Emerging Technologies* 6, 4 (1998), 271–288.
[21] Paolo Comelli, Paolo Ferragina, Mario Notturno Granieri, and Flavio Stabile. 1995. Optical recognition of motor vehicle license plates. *IEEE transactions on Vehicular Technology* 44, 4 (1995), 790–799.
[22] T. Cover and P. Hart. 2006. Nearest Neighbor Pattern Classification. *IEEE Trans. Inf. Theor.* 13, 1 (Sept. 2006), 21–27. https://doi.org/10.1109/TIT.1967.1053964
[23] D-Link. 2018. DCS 932L Wireless Day/Night Camera. (2018). Retrieved August 8, 2018 from https://eu.dlink.com/uk/en/products/dcs-932l-day-night-cloud-camera
[24] D-Link. 2018. Sound & Motion Detection. (2018). Retrieved August 8, 2018 from http://us.dlink.com/features/motion-detection-alerting/
[25] Django. 2018. The web framework for perfectionists with deadlines. (2018). Retrieved August 8, 2018 from https://www.djangoproject.com
[26] Steven Dow, Anand Kulkarni, Scott Klemmer, and Björn Hartmann. 2012. Shepherding the Crowd Yields Better Work. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work (CSCW '12)*. ACM, New York, NY, USA, 1013–1022. https://doi.org/10.1145/2145204.2145355
[27] Jakob Eriksson, Lewis Girod, Bret Hull, Ryan Newton, Samuel Madden, and Hari Balakrishnan. 2008. 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*. ACM, 29–39.
[28] Facebook. 2018. React – A JavaScript library for building user interfaces. (2018). Retrieved August 8, 2018 from https://reactjs.org
[29] Jon E. Froehlich, Eric Larson, Tim Campbell, Conor Haggerty, James Fogarty, and Shwetak N. Patel. 2009. HydroSense: Infrastructure-mediated Single-point Sensing of Whole-home Water Activity. In *Proceedings of the 11th International Conference on Ubiquitous Computing (UbiComp '09)*. ACM, New York, NY, USA, 235–244. https://doi.org/10.1145/1620545.1620581

[30] Miriam Greis, Florian Alt, Niels Henze, and Nemanja Memarovic. 2014. 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)*. ACM, New York, NY, USA, 1435–1438. https://doi.org/10.1145/2556288.2557186

[31] Anhong Guo, Xiang 'Anthony' Chen, Haoran Qi, Samuel White, Suman Ghosh, Chieko Asakawa, and Jeffrey P. Bigham. 2016. 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)*. ACM, New York, NY, USA, 651–664. https://doi.org/10.1145/2984511.2984518

[32] Anhong Guo, Jeeeun Kim, Xiang 'Anthony' Chen, Tom Yeh, Scott E. Hudson, Jennifer Mankoff, and Jeffrey P. Bigham. 2017. Facade: Auto-generating Tactile Interfaces to Appliances. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 5826–5838. https://doi.org/10.1145/3025453.3025845

[33] Sidhant Gupta, Matthew S. Reynolds, and Shwetak N. Patel. 2010. 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)*. ACM, New York, NY, USA, 139–148. https://doi.org/10.1145/1864349.1864375

[34] Danna Gurari and Kristen Grauman. 2017. 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)*. ACM, New York, NY, USA, 3511–3522. https://doi.org/10.1145/3025453.3025781

[35] Danna Gurari, Qing Li, Abigale J Stangl, Anhong Guo, Chi Lin, Kristen Grauman, Jiebo Luo, and Jeffrey P Bigham. 2018. VizWiz Grand Challenge: Answering Visual Questions from Blind People. *arXiv preprint arXiv:1802.08218* (2018).

[36] Kotaro Hara, Jin Sun, Robert Moore, David Jacobs, and Jon Froehlich. 2014. 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)*. ACM, New York, NY, USA, 189–204. https://doi.org/10.1145/2642918.2647403

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

[38] Ting-Hao Kenneth Huang, Amos Azaria, and Jeffrey P. Bigham. 2016. 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)*. ACM, New York, NY, USA, 1555–1562. https://doi.org/10.1145/2851581.2892502

[39] Ting-Hao Kenneth Huang, Joseph Chee Chang, and Jeffrey P. Bigham. 2018. 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)*. ACM, New York, NY, USA, Article 295, 13 pages. https://doi.org/10.1145/3173574.3173869

[40] Ting-Hao Kenneth Huang, Yun-Nung Chen, and Jeffrey P. Bigham. 2017. Real-time On-Demand Crowd-powered Entity Extraction. *ArXiv e-prints* (April 2017). arXiv:cs.HC/1704.03627

[41] Ting-Hao Kenneth Huang, Walter S Lasecki, Amos Azaria, and Jeffrey P Bigham. 2016. "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.

[42] Harmanpreet Kaur, Mitchell Gordon, Yiwei Yang, Jeffrey P Bigham, Jaime Teevan, Ece Kamar, and Walter S Lasecki. 2017. 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.

[43] Neil Klingensmith, Joseph Bomber, and Suman Banerjee. 2014. 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)*. ACM, New York, NY, USA, 123–132. https://doi.org/10.1145/2602044.2602049

[44] Stacey Kuznetsov and Eric Paulos. 2010. 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)*. ACM, New York, NY, USA, 1851–1860. https://doi.org/10.1145/1753326.1753604

[45] Gierad Laput, Walter S. Lasecki, Jason Wiese, Robert Xiao, Jeffrey P. Bigham, and Chris Harrison. 2015. Zensors: Adaptive, Rapidly Deployable, Human-Intelligent Sensor Feeds. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems (CHI '15)*. ACM, New York, NY, USA, 1935–1944. https://doi.org/10.1145/2702123.2702416

[46] Gierad Laput, Yang Zhang, and Chris Harrison. 2017. Synthetic Sensors: Towards General-Purpose Sensing. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems (CHI '17)*. ACM, New York, NY, USA, 3986–3999. https://doi.org/10.1145/3025453.3025773

[47] Walter S. Lasecki, Young Chol Song, Henry Kautz, and Jeffrey P. Bigham. 2013. Real-time Crowd Labeling for Deployable Activity Recognition. In *Proceedings of the 2013 Conference on Computer Supported Cooperative Work (CSCW '13)*. ACM, New York, NY, USA, 1203–1212. https://doi.org/10.1145/2441776.2441912

[48] Walter S. Lasecki, Phyo Thiha, Yu Zhong, Erin Brady, and Jeffrey P. Bigham. 2013. Answering Visual Questions with Conversational Crowd Assistants. In *Proceedings of the 15th International ACM SIGACCESS Conference on Computers and Accessibility (ASSETS '13)*. ACM, New York, NY, USA, Article 18, 8 pages. https://doi.org/10.1145/2513383.2517033

[49] Walter S. Lasecki, Rachel Wesley, Jeffrey Nichols, Anand Kulkarni, James F. Allen, and Jeffrey P. Bigham. 2013. Chorus: A Crowd-powered Conversational Assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology (UIST '13)*. ACM, New York, NY, USA, 151–162. https://doi.org/10.1145/2501988.2502057
[50] David Lowe. 2015. The Computer Vision Industry. (2015). Retrieved August 8, 2018 from https://www.cs.ubc.ca/~lowe/vision.html
[51] Mashable. 2010. Omoby: Visual Search for the iPhone. (2010). Retrieved August 8, 2018 from https://mashable.com/2010/03/04/omoby-visual-search-iphone
[52] Thomas B Moeslund and Erik Granum. 2001. A survey of computer vision-based human motion capture. *Computer vision and image understanding* 81, 3 (2001), 231–268.
[53] Nest. 2018. Nest Aware. (2018). Retrieved August 8, 2018 from https://nest.com/cameras/nest-aware/
[54] Dat Tien Nguyen, Firoj Alam, Ferda Ofli, and Muhammad Imran. 2017. Automatic Image Filtering on Social Networks Using Deep Learning and Perceptual Hashing During Crises. *arXiv preprint arXiv:1704.02602* (2017).
[55] Shwetak N. Patel, Matthew S. Reynolds, and Gregory D. Abowd. 2008. *Detecting Human Movement by Differential Air Pressure Sensing in HVAC System Ductwork: An Exploration in Infrastructure Mediated Sensing*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1–18. https://doi.org/10.1007/978-3-540-79576-6_1
[56] Florian Schroff, Dmitry Kalenichenko, and James Philbin. 2015. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 815–823.
[57] James Scott, A.J. Bernheim Brush, John Krumm, Brian Meyers, Michael Hazas, Stephen Hodges, and Nicolas Villar. 2011. PreHeat: Controlling Home Heating Using Occupancy Prediction. In *Proceedings of the 13th International Conference on Ubiquitous Computing (UbiComp '11)*. ACM, New York, NY, USA, 281–290. https://doi.org/10.1145/2030112.2030151
[58] Emmanuel Munguia Tapia, Stephen S. Intille, and Kent Larson. 2004. *Activity Recognition in the Home Using Simple and Ubiquitous Sensors*. Springer Berlin Heidelberg, Berlin, Heidelberg, 158–175. https://doi.org/10.1007/978-3-540-24646-6_10
[59] Matthew A Turk and Alex P Pentland. 1991. Face recognition using eigenfaces. In *Computer Vision and Pattern Recognition, 1991. Proceedings CVPR'91., IEEE Computer Society Conference on*. IEEE, 586–591.
[60] David Vernon. 1991. Machine vision-Automated visual inspection and robot vision. *NASA STI/Recon Technical Report A* 92 (1991).
[61] Carl Vondrick, Donald Patterson, and Deva Ramanan. 2013. Efficiently scaling up crowdsourced video annotation. *International Journal of Computer Vision* 101, 1 (2013), 184–204.
[62] D. H. Wilson and C. Atkeson. 2005. *Simultaneous Tracking and Activity Recognition (STAR) Using Many Anonymous, Binary Sensors*. Springer Berlin Heidelberg, Berlin, Heidelberg, 62–79. https://doi.org/10.1007/11428572_5
[63] Christoph Zauner. 2010. Implementation and benchmarking of perceptual image hash functions. (2010).
[64] M. Zeifman and K. Roth. 2011. Nonintrusive appliance load monitoring: Review and outlook. *IEEE Transactions on Consumer Electronics* 57, 1 (February 2011), 76–84. https://doi.org/10.1109/TCE.2011.5735484
[65] Kaipeng Zhang, Zhanpeng Zhang, Zhifeng Li, and Yu Qiao. 2016. Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters* 23, 10 (2016), 1499–1503.

Received February 2018; revised May 2018; accepted September 2018