SnapLink: Fast and Accurate Vision-Based Appliance Control in Large Commercial Buildings

KAIFEI CHEN, University of California Berkeley, USA
JONATHAN FÜRST, IT University of Copenhagen, Denmark
JOHN KOLB and HYUNG-SIN KIM, University of California Berkeley, USA
XIN JIN, Johns Hopkins University, USA
DAVID E. CULLER and RANDY H. KATZ, University of California Berkeley, USA

As the number and heterogeneity of appliances in smart buildings increases, identifying and controlling them becomes challenging. Existing methods face various challenges when deployed in large commercial buildings. For example, voice command assistants require users to memorize many control commands. Attaching Bluetooth dongles or QR codes to appliances introduces considerable deployment overhead. In comparison, identifying an appliance by simply pointing a smartphone camera at it and controlling the appliance using a graphical overlay interface is more intuitive. We introduce SnapLink, a responsive and accurate vision-based system for mobile appliance identification and interaction using image localization. Compared to the image retrieval approaches used in previous vision-based appliance control systems, SnapLink exploits 3D models to improve identification accuracy and reduce deployment overhead via quick video captures and a simplified labeling process. We also introduce a feature sub-sampling mechanism to achieve low latency at the scale of a commercial building. To evaluate SnapLink, we collected training videos from 39 rooms to represent the scale of a modern commercial building. It achieves a 94% successful appliance identification rate among 1526 test images of 179 appliances within 120 ms average server processing time. Furthermore, we show that SnapLink is robust to viewing angle and distance differences, illumination changes, as well as daily changes in the environment. We believe the SnapLink use case is not limited to appliance control: it has the potential to enable various new smart building applications.

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

Additional Key Words and Phrases: Human-Building Interaction, Image Localization

ACM Reference Format:
Kaifei Chen, Jonathan Fürst, John Kolb, Hyung-Sin Kim, Xin Jin, David E. Culler, and Randy H. Katz. 2017. SnapLink: Fast and Accurate Vision-Based Appliance Control in Large Commercial Buildings. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 4, Article 129 (December 2017), 27 pages. https://doi.org/10.1145/3161173

1 INTRODUCTION

New generations of smart Building Management Systems (BMSs) are transforming how buildings are managed. They integrate and abstract building systems and appliances to a unified, high-level programming interface [3, 8],

Authors' addresses: Kaifei Chen, University of California Berkeley, Computer Science Division, Berkeley, CA, 94720, USA; Jonathan Fürst, IT University of Copenhagen, Computer Science Division, Copenhagen, 2300, Denmark; John Kolb; Hyung-Sin Kim, University of California Berkeley, Computer Science Division, Berkeley, CA, 94720, USA; Xin Jin, Johns Hopkins University, Department of Computer Science, Baltimore, MD, 21218, USA; David E. Culler; Randy H. Katz, University of California Berkeley, Computer Science Division, Berkeley, CA, 94720, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2017 Association for Computing Machinery.
2474-9567/2017/12-ART129 $15.00
https://doi.org/10.1145/3161173

Fig. 1. Application scenario for our vision-based appliance identification and control system. Users can get a control interface of an appliance by capturing an image from an arbitrary angle and distance, even when other similar appliances exist in the same or different rooms.

enabling numerous smart building applications, such as occupancy-based HVAC operation [4], fault detection [14], and demand response [13]. While many building management tasks can be automated, people still need to interact directly with many appliances like lights, projectors, and printers. Additionally, because comfort preferences vary from one person to another, direct human control of appliance settings is still necessary [15]. However, human-building interaction (i.e., direct control of appliances) is not convenient today, and the growing number and heterogeneity of digitally controllable smart appliances through a BMS in large commercial buildings further exacerbates this situation.

To make human-building interactions more convenient, we start with the intuition that it is natural for humans to use their visual senses when interacting with appliances: *"What you see is what you control."* This motivates us to design a *vision*-based appliance identification and control system that allows users to interact with any appliance they can see. Such a system needs to include a simple and intuitive user interface that converts human vision (an appliance image) to a control interface. We use *smartphones*, one of the most ubiquitous devices in modern life, as the vision sensor (i.e., capture an image of an appliance within view) and appliance controller (i.e., control the appliance in the captured image). Specifically, we aim to enable users to control an appliance through a smartphone application by simply pointing at it, as depicted in Figure 1. Users of this application should not be concerned with taking pictures from a specific angle or distance from the appliance. Whatever image is taken, users should get a proper control interface of what they are pointing at (e.g., on/off button for projector 2 in room 465) on the smartphone screen. Furthermore, no matter how many similar appliances exist (e.g., the same projector in various meeting rooms), users should still enjoy convenient appliance control by receiving an accurate and fast identification response when they point their smartphone at an appliance.

Even though a number of studies have investigated the problem [43, 49, 54], their proposals cannot scale to commercial buildings containing hundreds to thousands of appliances. Some approaches require additional infrastructure for each appliance, such as laser or infrared signal receivers [40, 54], introducing a large deployment overhead. Some are not applicable when controlling appliances from an arbitrarily large distance and angle, such as QR codes [9, 50]. Finally, some approaches are inconvenient for users, especially in a large building, such as

typing textual queries [7], memorizing vocal or gesture commands [43, 49], or manually browsing and searching on a 2D/3D map [18].

In this paper, we design and implement **SnapLink**, a prototype system that can quickly and accurately identify an appliance among hundreds to thousands of (possibly similar) appliances based on a query image from an arbitrary location and orientation. SnapLink uses *image localization* [46] to identify an appliance by finding where a query image is located in a pre-constructed *3D space* in the database. This design choice comes from two motivations: (1) From the user perspective, image localization provides high identification accuracy with arbitrary query images. (2) From a deployment perspective, 3D-model construction (the database for image localization) requires only capturing a video with a commercial off-the-shelf RGB-Depth camera while walking through a building. With a 3D model, appliance labels only need to be applied once per appliance, whereas previous vision-based systems [9, 22, 28] require many images for every appliance and every image to be labeled. In addition, we propose two ways to further improve the accuracy and latency of image localization-based appliance identification in commercial buildings: (1) a *Feature Sub-sampling* mechanism that eliminates redundant features from query images and (2) a *Geo-partitioned* 3D-model construction that does not build a monolithic 3D-model but a group of small 3D-models (of each room) to represent the entire building.

We have built a SnapLink prototype system including an Android smartphone application and integrated it with BOSS [8], a representative modern BMS. To evaluate it, we construct a database with 3D models by collecting 67 minutes of video footage from 39 different rooms from 5 different buildings, which is equal to the size of a small commercial building [11]. We also label an arbitrarily selected subset containing 179 appliances in these rooms. To our knowledge, this dataset size has one order of magnitude more modeling images and covers a much larger space than those featured in prior vision-based object identification work [9, 22, 28]. For 1526 test query images of these appliances from various angles and distances, our SnapLink system achieves **94%** identification accuracy and **120 ms** server processing time per query image, whereas our image retrieval-based baseline system takes 177 ms to achieve only 66% accuracy.

Our contributions are as follows:

- We introduce a novel and useful application, *arbitrary image*-based appliance identification and control with ubiquitous *smartphones*. We describe several technical requirements in designing a system to support this application.
- With the application requirements and technical challenges as the basis, we present a set of key approaches in order to maintain low latency and deployment overhead in large buildings: *image localization* in *3D models*, *Feature Sub-sampling*, and *Geo-partitioning*.
- We build an end-to-end system that integrates SnapLink with a modern BMS (i.e., BOSS [8]) and provides an Android smartphone application as a user interface.[1]
- We validate the performance of SnapLink at the scale of a commercial building that has 39 rooms and 179 appliances, which is an order of magnitude larger than in prior work [9, 22, 28]. In this real-world scenario, SnapLink achieves 94% identification accuracy and 120 ms server processing time.

While this paper focuses on appliance identification and control, we believe that it is only one example of a new generation of smart building applications enabled by SnapLink, including location-based authorization, augmented information display, indoor navigation, and building diagnosis and re-commissioning. The rest of this paper is organized as follows: In Section 2 we introduce our application scenario and its requirements. We also discuss candidate vision-based identification methods for our application. Next, we introduce details on SnapLink's design in Section 3 and its implementation in Section 4. We present deployment results of SnapLink in Section 5. We position SnapLink among prior work in Section 6, and conclude with a discussion and summary in Sections 7 and 8.

[1] https://github.com/SoftwareDefinedBuildings/SnapLink

2 TARGET APPLICATION: VISION-BASED APPLIANCE IDENTIFICATION AND CONTROL

Recent work on smart buildings has investigated how to unify various vertically integrated, isolated BMSs and Internet of Things (IoT) devices into a common set of abstractions. For example, BOSS (Building Operating System Services) develops a building operating system on which various applications can run to improve building performance and occupant comfort [8]. Applications identify the underlying heterogeneous appliances by standardized metadata and unique identifiers [3]. Such identification models work well for centralized building applications, but fail when occupants want to directly interact with appliances. Our hypothesis is that most humans want to control appliances within their sight. Thus, user-centric, cyber-physical applications should leverage visual identification.

2.1 Application Scenario and System Architecture

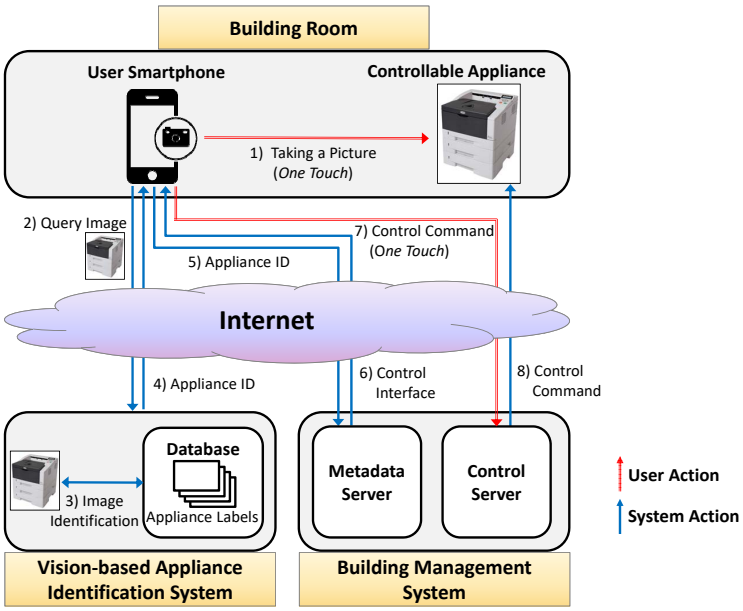


Fig. 2. Application scenario for our vision-based appliance identification and control system, consisting of controllable appliance, user smartphone, vision-based appliance identification system, and BMS.

Our work aims to design a vision-based appliance identification and control system that provides an intuitive mobile user interface and can be applied in an environment as complex as a commercial building. Consider the following application scenario. While walking through a building, Alice encounters a networked printer in a conference room and wishes to use it to print out a document from her smartphone. She turns on the SnapLink app and takes an *arbitrary* picture of the printer (as in Figure 1). The app then processes the picture and quickly recognizes it as the specific printer in the conference room. The app overlays the printer's control interface on the smartphone screen so she can interact with the printer. Alice clicks on the "Upload and Print" button to upload the document, which is then printed out by the printer.

At a higher level, the entire workflow of this application and the system architecture are depicted in Figure 2, which includes both user actions and the underlying system's behavior. It consists of a controllable appliance, the user's smartphone, a vision-based appliance identification system, and the BMS. After Alice takes a picture of the

target appliance, her smartphone sends the image to a vision-based appliance identification system. Using the query image, the identification system selects the most likely appliance from those in its database and sends the appliance ID to Alice's smartphone. The smartphone sends the appliance ID to the metadata server of the BMS and receives the control interface for the target appliance.[2] Finally, Alice can now see the control interface on her smartphone screen and control it as needed. When she touches a command button, the smartphone generates and delivers the control command to the BMS's control server, which finally controls the target appliance by sending it the proper actuation command. Note that, during the whole appliance identification process, all Alice needs to do is touch her smartphone screen once to take a picture before she can interact with the appliance.

2.2 Technical Requirements

Our system needs to meet the following requirements to enable the target application, which motivate our subsequent design of SnapLink detailed in Section 3.

Identification Accuracy. High appliance identification accuracy is our first goal. Our system needs to accurately identify a target appliance from an arbitrary image and distinguish it from other similar appliances. However, even a single incorrect result will be frustrating. Therefore, while achieving high identification accuracy, our system should also allow manual appliance selection as a fail-over. Identification accuracy should be high under changing environments. We assume most appliances are not moved frequently (e.g., printers, projectors, lights), but the environment they sit in can always have daily occupant activities and changes in lighting conditions.

Low Latency. As an interactive system, our system needs to be responsive so that users do not perceive a considerable delay between an action and its response. We aim to achieve no larger than 100 ms latency, which is the time requirement to create the impression of an instantaneous reaction [38]. However, since the most important performance metric is identification accuracy, minimizing latency should not sacrifice accuracy.

Scalability. Given that our system targets large commercial buildings with thousands of appliances, the database of our appliance identification system should hold a large number of appliance labels. Since finding the most relevant appliance among a large number of candidates creates a computational burden, our system should provide scalability in terms of database size to achieve both high accuracy and low latency when applied to large commercial buildings.

Low Deployment Overhead. To enable the application, system managers/staff need to construct a database with the information on all controllable appliances in a large commercial building. This involves nontrivial deployment overhead. Our system should be user-friendly but also deployment/management-friendly. To this end, we need to carefully consider what method to use for vision-based identification since it significantly impacts deployment overhead (e.g., image retrieval requires taking a large number of pictures in the deployment phase).

2.3 Design Options for Vision-based Appliance Identification

We now discuss how to enable vision-based appliance identification, the most important function of our system, and present our core design choices for SnapLink. A modern commercial building has hundreds to thousands of appliances, but most fall into a limited set of categories (e.g., thermostats, projectors, lights). This means that many appliance instances fall into the same category. To enable users to control a single target appliance, we need to recognize an appliance *instance* (i.e., a specific physical object) rather than its *category*.

[2]Alternatively, the same server could identify the appliance, use its ID to retrieve the proper handler from the BMS, and send a control interface back to Alice's phone. However, this is an implementation detail that does not affect the core functionality of the system, and the round trip time between the phone and a local BMS server is likely to be small.

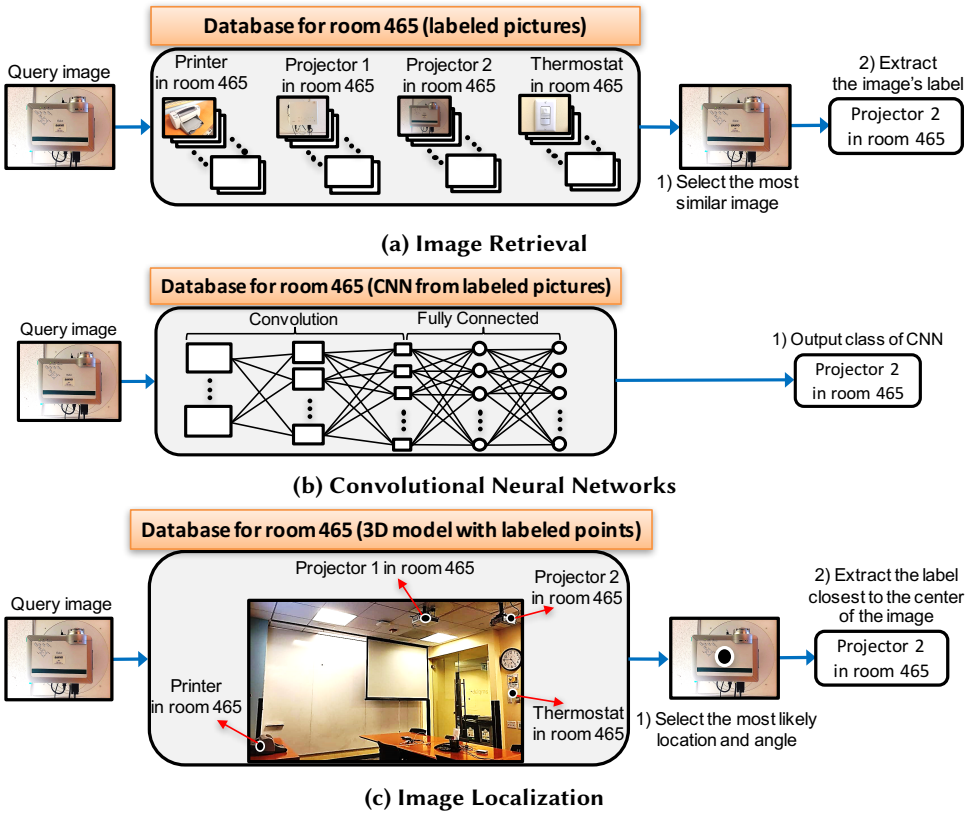


Fig. 3. Comparison between three design options for vision-based instance identification: image retrieval, Convolutional Neural Networks, and image localization.

There are several representative ways to perform instance recognition: *image retrieval*, *Convolutional Neural Networks (CNNs)*, and *image localization*, as depicted in Figure 3. An image retrieval-based instance identification system (Figure 3a) needs to construct a database of labeled appliance images. When receiving a query image as an input, it searches for the most similar labeled image among the database based on the number of common salient visual features (e.g., SURF (Speeded Up Robust Features) [5]). To our knowledge, image retrieval is used by all previous vision-based instance identification systems for appliance control, such as [9, 22, 28, 35]. However, it has several drawbacks that preclude its deployment at the scale of a large commercial building. First, existing techniques require every database image to contain only one appliance and each image must be individually labeled. Second, recognition succeeds when the query image is taken from a similar angle and distance as the corresponding labeled image in the database. In a commercial building with hundreds to thousands of appliances, this results in significant deployment overhead for building managers/staff[3] to take pictures of all appliances from various angles and distances and to label each of them (i.e., number of labels = appliances × angles × distances). The accuracy and deployment overhead would not be an issue when it comes to a well-constructed huge database,

[3]Given that each label of the vision-based identification system must be translated to its appliance ID in the BMS (metadata server), the labels must be added by professional building managers/staff who have knowledge of the BMS instead of crowdsourced from building occupants.

which already has an extremely large number of labeled images [41]. Unfortunately, this is not the case for our target application, which requires constructing a *local* database for each commercial building.

Convolutional Neural Networks (CNNs), which have been extensively used for category recognition [10, 31], have recently been applied to instance recognition as well (Figure 3b). A system trains labeled pictures into a CNN model in advance with the instance labels as classes. A query image is classified by the model and the output class is simply the recognized instance. However, as summarized in [55], CNNs only outperform image retrieval on instance recognition with scenery [24] and landmarks [41, 42] datasets, where training and testing images have a similar data distribution (e.g., number of images of an instance) and little occlusion of the instances is present. In building environments, the usage of appliances is unknown in advance, and occlusions are very common (especially when the viewing angle is large, such as in Figure 23b), therefore image retrieval is preferable to a CNN-based approach for our purposes. Moreover, an image classification process can take seconds to finish on a modern CPU [55], making CNNs prohibitively expensive without a GPU for our application, which requires low latency.

An image localization-based instance identification system (Figure 3c) involves constructing a database in the form of a 3D model of a building and one labeled point for each appliance in the 3D space [46]. To infer which appliances are visible within a query image, it localizes the query image in the 3D-space database using the most likely location and angle, captures the image of that specific location/angle (i.e., a small fraction) from the 3D space, and finds the appliance label point closest to the center of the captured image. Several aspects make image localization a better fit for our target application compared to image retrieval. First, the system deployment process is extremely simple because a 3D model that covers various angles can be built by capturing a video with a modern modeling tool, such as Project Tango [48], while walking through a building. Labeling on a 3D model is also simple because every appliance only needs to be labeled once as a single point in the 3D space, irrespective of the number of images containing that appliance in the database. Second, because a 3D model incorporates all image features into a 3D space, it can represent all angles and distances, resulting in a higher identification accuracy with an arbitrary query image compared to image retrieval. As an example, in a room that contains 100 appliances, each of which needs to be captured from 10 different viewing locations, image retrieval would require 1,000 reference images, whereas image localization only requires a video and 100 labeled points. Thus, our system uses image localization for vision-based appliance instance identification.

However, a naive approach to image localization is not suitable for our target application due to a number of problems. Constructing a properly functioning 3D model for a large commercial building involves many unsolved problems, such as failure to identify a previously visited place (loop closure detection [16]) caused by cumulative errors while constructing a 3D model. Furthermore, image localization imposes a significant computational burden, resulting in unsatisfactory latency according to our system requirements. To apply image localization to our target application, these issues need to be addressed, which is the focus of our SnapLink design in Section 3.

3 SNAPLINK SYSTEM DESIGN

In this section we introduce the details of the SnapLink design, which addresses three issues of image localization to support our target application: (1) how to construct a properly working 3D model in a large building (i.e., *geo-partitioning*), (2) how to identify appliances through image localization with a given 3D model, and (3) how to minimize the computation time without losing accuracy (i.e., *feature sub-sampling*). We first give an overview and then introduce the details of each design element.

3.1 SnapLink Overview

Figure 4 shows an overview of SnapLink, which comprises the *offline deployment phase* and the *online appliance identification phase*. In the deployment phase, a user or a building manager needs to capture videos inside the

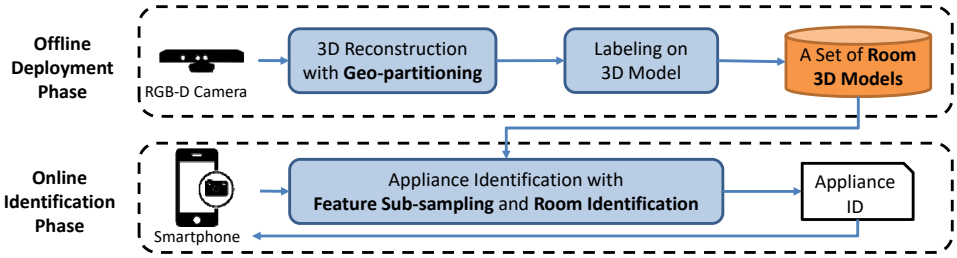


Fig. 4. SnapLink overview. The offline deployment phase builds a 3D point cloud with a set of partitioned 3D models and labeled appliances. The online identification phase recognizes an appliance instance through an image localization process including feature sub-sampling and room identification based on the set of 3D room models.

building using an RGB-Depth (RGB-D) camera. In doing this, SnapLink uses *geo-partitioning* to enable easy and scalable 3D model construction without cumulative errors [16]; it uses a set of room-based 3D models, not a single building 3D model, to represent the whole building. Each video for a room is fed into a standard 3D reconstruction tool, which generates a 3D point cloud for the room. Then, the manager labels each appliance point within each room's 3D model and stores these labeled 3D models in the database. We also build our own labeling tool to simplify the labeling process.

In the appliance identification phase, SnapLink localizes a query image in a set of 3D models in the database to infer the labels that are visible in the image. On top of the standard image localization process, SnapLink adds two pre-processing stages before image localization: *feature sub-sampling* and *room identification*. The feature sub-sampling stage *randomly* selects features in the query image to reduce computation time while still providing adequate information for correct localization. The room identification stage is necessary to operate with the database partitioned by room.

3.2 Building a 3D Model Database with Geo-partitioning

In the offline deployment phase, we build a 3D model of the building and label appliances for the runtime appliance identification phase. To model a building, we collect a separate video for each room. This partitions our database into a collection of 3D room models, which is critical for a successful building model for several reasons. First, it simplifies the database construction process. Only a short video needs to be captured to model a room, which takes about 100 seconds on average in our case. When a room's environment is changed (e.g., furniture is rearranged or the room is remodeled), a building manager is required to recapture a video only for that room, not the whole building. Our 64-day evaluation described in Section 5.9 demonstrates that a room model is robust to everyday changes (e.g., content on whiteboard, items on table) as long as the appliance itself is still present. Second, geo-partitioning reduces cumulative errors in the 3D reconstruction process because it is applied for each room (small scale). Note that 3D reconstruction, which converts a video captured by an RGB-Depth camera to a 3D point cloud, is a necessary procedure for building an image localization database. Also note that it causes cumulative errors at a large scale due to drifting and loop closure detection failures [16]. Third, since geo-partitioning constructs the whole building dataset as a group of room datasets, the whole dataset can easily be hierarchically partitioned (e.g., to floor datasets). This enables the system to use the closest local server (e.g., a floor-specific server) for a partitioned dataset, instead of a single server for the whole dataset, which reduces network latency in a large scale building. The closest local server can be identified by using localization methods, such as WiFi fingerprints or GPS.

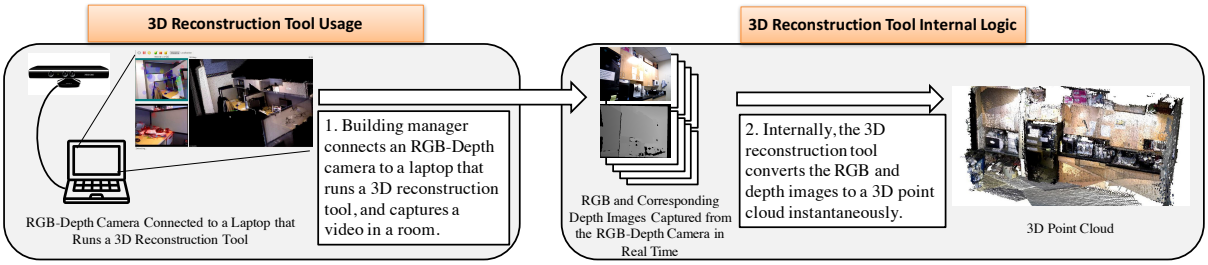


Fig. 5. An example of SnapLink's 3D modeling. Using an off-the-shelf RGB-Depth camera, a building manager captures a video of a room, which is automatically converted to a room 3D model by a 3D reconstruction program, such as RTABMap [30].

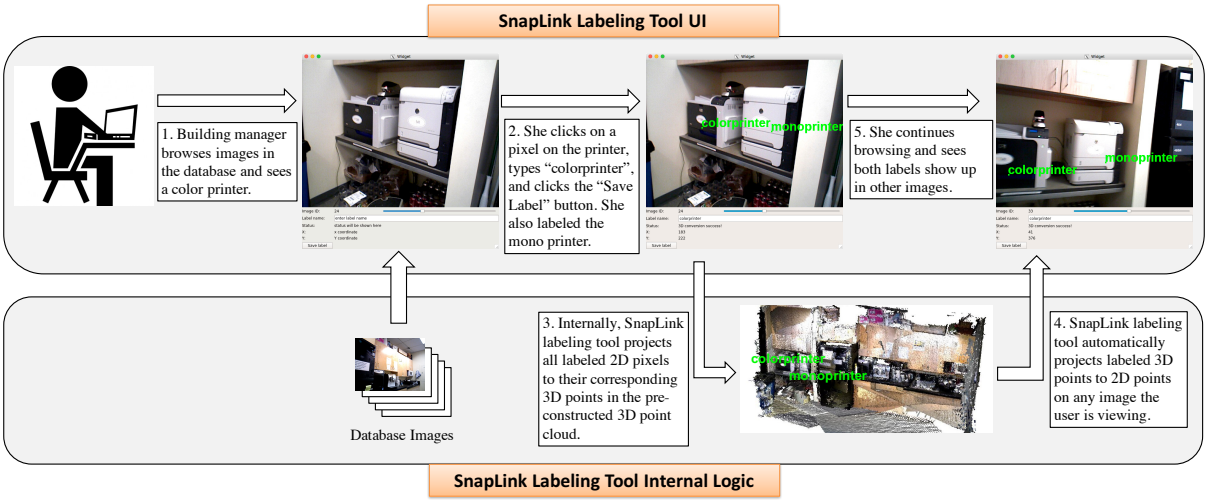


Fig. 6. An example of SnapLink's labeling process. A building manager can label an appliance by clicking on a pixel and typing a label while browsing captured images. Multiple appliances can be labeled in a single image. All labels are projected to the 3D space and therefore show up in all images containing it.

Figure 5 shows how a building manager can construct a 3D room model. She simply needs to connect a commercial off-the-shelf RGB-Depth camera to a laptop, run a 3D reconstruction program (e.g., Project Tango [48] or RTABMap [30]) on the laptop, and point the camera at appliances while walking through the room. With the captured room images, the 3D reconstruction program automatically computes the relative 3D transformations between similar image pairs by using the three-point-algorithm inside a RANSAC loop [52], projects the images onto a 3D space, and creates a list of 3D points, resulting in a 3D point cloud of the room.

Once a 3D point cloud is constructed for a room, a building manager needs to label each appliance in the room only one time on a 3D point in the 3D point cloud. To simplify this process, we built a labeling tool as depicted in Figure 6. A building manager browses room images captured by the camera. When she finds an appliance in an image, she clicks on any point on the appliance, types its label, and clicks on the "Save Label" button. Since our labeling tool can match each 2D point in a captured image to a 3D point in the 3D point cloud, it automatically labels the appliance on the corresponding 3D point. While she browses further, the tool displays the appliance

label if the browsed image includes the labeled point, which prevents her from relabeling the same appliance. Furthermore, she is allowed to label multiple appliances in a single image (e.g., "colorprinter" and "monoprinter" in Figure 6), which is not the case for image retrieval because each image is expected to contain only a single appliance instance. Undoubtedly, there are many possible ways to simplify the labeling process, such as using voice recognition to label appliances in a smartphone camera view, which is discussed further in Section 7.

3.3 Identifying Appliances in Partitioned 3D Models

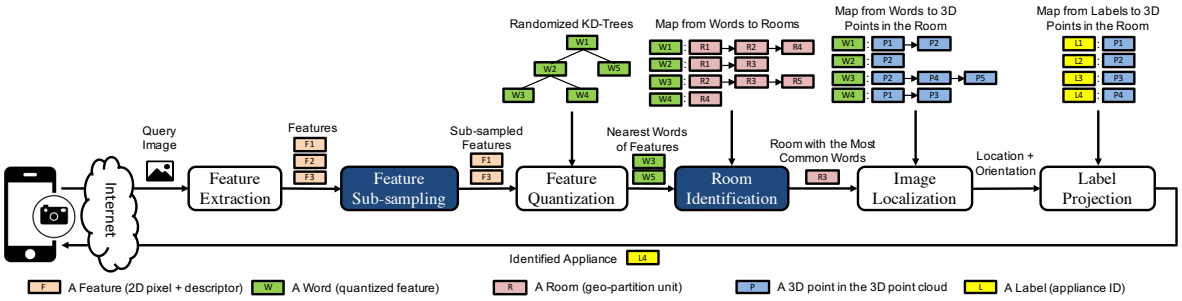


Fig. 7. SnapLink appliance identification pipeline. The extracted query image features are subsampled and quantized into words using KD-trees. Subsequently, room, image location and orientation are identified. Finally an appliance is identified by projecting labeled 3D points onto the query image.

Figure 7 depicts the appliance identification pipeline of SnapLink. As in the standard image localization approach, SnapLink first extracts salient features from the query images. We use SURF for its speed and accuracy [5, 22]. However, unlike previous work that uses all features, we only use randomly sub-sampled features to achieve lower latency without sacrificing accuracy, which we discuss in Section 3.4. We then quantize the sub-sampled features to their closest cluster among many clusters of features, whose centers are dubbed *words* [47]. These clusters are generated from all database features in advance using the distance ratio test [5]. We use randomized KD-Trees of these words to find the nearest cluster of a feature, because it outperforms other approaches in both speed and accuracy on visual feature descriptors [37]. The feature quantization greatly reduces the computation in later steps. We use an empirically optimal distance ratio threshold 0.7 and use the default of 4 randomized KD-Trees, as implemented in FLANN [37].

As we discussed in Section 3.2, to overcome problems in large scale 3D reconstructions, our database consists of 3D models of separate rooms. Consequently, we need to identify the room before we can localize the query image. We define the similarity between the query image q and room r as

$$s_{q,r} = \frac{\left|W_q \cap W_r\right|}{|W_r|} \tag{1}$$

where W_q is the set of words in image q, and W_r is the set of words in room r. To quickly compute $\left|W_q \cap W_r\right|$, we tally all words for room r contained in query image q. Then we compute $s_{q,r}$ for every room and find the one with the highest similarity. Note that this is done efficiently in one traversal of all words in image q, maintaining a running total for each room r and selecting the room with the highest total once all words have been processed. This approach is also described in [32].

With a room r identified, we can localize the query image q within it. We solve this as a standard Perspective-N-Point (PnP) [44] problem with a list of 2D points in image q and their corresponding 3D points in room r. To

get the list, for every 2D SURF point p in image q, we find its closest 3D point whose SURF descriptor is in the same cluster of p. Specifically, we compute

$$\underset{i, w_p = w_i^r}{\operatorname{argmin}} \|\mathbf{f}_p - \mathbf{f}_i^r\| \tag{2}$$

where w_p and $\mathbf{f}_p$ are the word and SURF descriptor of p, and w_i^r and $\mathbf{f}_i^r$ are the word and SURF descriptor of the i^{th} 3D point in r. In addition, we perform two optimizations when obtaining the list, inspired by [46]. First, we start our search with 2D points whose SURF descriptors are in smaller clusters to reduce noise. Second, we limit the size of the list to reduce redundant information. We use the empirically optimal value of 100, as in [46], for our list size limit.

After we know the location and orientation of the query image, we can compute where every appliance appears in the image by projecting their labeled 3D points onto the query image plane. Our server returns the appliance ID whose label is closest to the center of the query image. As described in Section 2, the mobile client can then query for a control interface from the BMS server using the identified appliance ID to enable user interactions with that appliance.

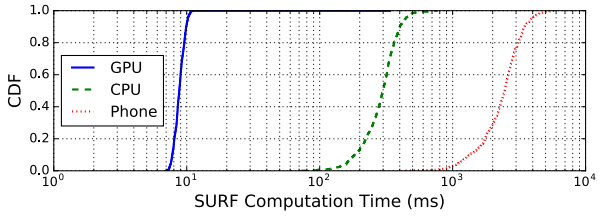


Fig. 8. CDF of SURF computation time of a 640×480 image on a server GPU (NVIDIA GeForce GTX 970), a server CPU (Intel i7-4790), and a smartphone CPU (Qualcomm Snapdragon 400).

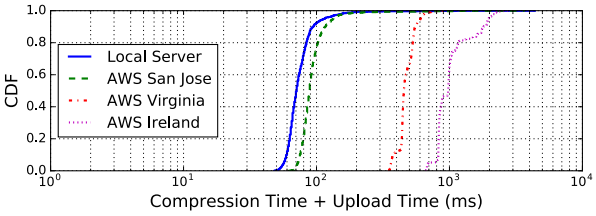


Fig. 9. CDF of JPEG compression time + upload time of a 640×480 image to different servers: on campus, in San Jose (43 miles away), in Virginia (2350 miles away), and in Ireland (5050 miles away).

To examine the placement of computation for our identification pipeline, we conducted two similar experiments as in [22]. The first experiment measures the computation time of SURF [5] using OpenCV [6] on a server GPU, server CPU, and smartphone CPU. The second experiment measures the transmission time of an image from a smartphone to four servers at different locations: on campus, San Jose, Virginia, and Ireland. Figure 8 and Figure 9 show the respective results. As we can see, it takes 70 ms on average to transmit an image and compute SURF on a server GPU, but more than 1000 ms to compute SURF on the phone. Figure 9 also substantiates the necessity of geographically partitioning the database in order to run computation nearby and reduce end-to-end latency. In SnapLink, we therefore offload all appliance identification computation to a server that is physically close.

3.4 Feature Sub-sampling

Since appliance identification is an interactive process, we aim to reduce the end-to-end latency to an imperceptible range. Even though it is not practical to achieve our 100 ms response time goal [36] while network transmission takes 80 ms on average (as shown in Figure 9), we try to minimize server computation without sacrificing accuracy.

Table 1 shows the time complexity of every step in the standard image localization process. Feature extraction complexity depends on the content of the query image, but can usually finish in 10 ms when parallelized on a GPU, as shown in Figure 8. The average KD-Tree search time in the feature quantization stage is $O(\log F)$, where F is the number of features. However, because of the recursive nature of KD-Tree search, it cannot benefit

Stage	Average Time Complexity	Note
Feature Extraction	Depends on image contents	Can run on GPU.
Feature Quantization	$O(F \log W)$	F is the № of features, W is the № of words.
Room Identification	$O(F + R)$	F is the № of features, R is the № of rooms.
Image Localization	$O(P)$	P is the (capped) 2D-3D point pair list size.
Label Projection	$O(L)$	L is the № of labels.

Table 1. Time complexity of every stage in the standard image localization pipeline. Note that F, R, P, and L are usually several orders of magnitude smaller than W.

from GPU parallelization, especially when data dimensionality is high [20], such as SURF descriptors. Since feature quantization reduces image features to a much smaller set of words, it makes room identification, image localization, and label projection consume little time. Therefore, feature quantization is the bottleneck of the pipeline.

KD-Tree search time increases linearly with the number of features. Therefore we try to use a subset of the image features. A 640×480 image can generate hundreds to thousands of features, but not all of them are useful. Conceptually, the features that are both unique and robust are the most useful because they help match 2D points to 3D points with less ambiguity. Jain et. al. [23] use a counting bloom filter to get a uniqueness score for every word in the query image. However, this approach only works after features are quantized to words.

Instead of trying to find unique and robust features, we propose a simple random feature sub-sampling scheme. We argue that enough randomly sub-sampled features can provide the requisite information for image localization, while keeping the KD-Tree search time low. Furthermore, by introducing a tunable number of sub-sampled features, the system can adjust the trade-off between accuracy and latency based on the average KD-Tree search time, which depends on the number of words in the database. Similarly, we can use this mechanism to adjust end-to-end latency based on current network transmission time. Our evaluation in Section 5.4 shows the effectiveness of this approach.

4 SYSTEM IMPLEMENTATION

In this section, we describe our implementation of the labeling tool, appliance identification runtime, and Android mobile client.

4.1 SnapLink Server

For the deployment phase, we choose to use RTABMap because it is open source and supports various types of hardware (e.g., stereo camera, RGB-Depth camera). We also implemented a SnapLink labeling tool in 400+ lines of C++ code, using Qt as the GUI library. It currently reads RTABMap [30] databases and presents images to users for labeling. Labels are saved to the original database file. In the future, we plan to add adapters to other popular 3D reconstruction tools, such as Google Tango [48] and Bundler [1].

We implemented our SnapLink runtime in 3200+ lines of C++ code. It receives queries in a RESTful interface using HTTP. We use the Qt event system to construct the image localization pipeline. In the pipeline, we use OpenCV [6] to extract SURF, solve perspective in point (PnP) problems, and project points between 2D and 3D. We use the randomized KD-Tree implementation in FLANN [37] and transformation utilities from PCL [45].

4.2 Android Mobile Application

We developed an Android client using 1700+ lines of Java code. Figure 10 shows an example screenshot. It has a full screen see-through camera view with control widgets overlaid on top. A capture button is at the bottom right.

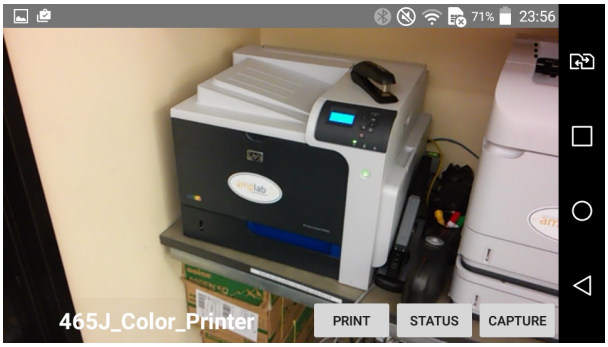


Fig. 10. SnapLink android application. The capture button sends a new image to a server. When an appliance is identified, it displays its name and control widgets retrieved from a BMS server.

The bottom left shows the currently identified appliance. Appliance control buttons are shown at the bottom center.

To minimize the identification latency, we disabled auto focus, auto exposure, and auto white balance, which can take up to 1 second to complete if enabled. However, if a user clicks on the capture button while moving the smartphone, it will create a blurry image that deteriorates salient feature extraction performance [26], and the user can choose to capture another still image if she sees an identification failure. Several approaches have been proposed to detect blurry images, such as using a Canny edge detector [22] or gyroscope readings [21]. We are currently implementing blurry image detection in the app, which will either ask the user to capture another image or automatically find a clear image in the image view stream. After the client sends a clear image and receives the identified appliance ID from a SnapLink server, it retrieves its control interface and metadata from BOSS [8] and updates the UI accordingly. In addition, it uses the secure publish/subscribe overlay network [2] designed for BOSS to send control commands.

5 EVALUATION

In this section, we first compare the accuracy and latency of image retrieval, image localization, and image localization with feature subsampling. Then we examine the scalability of SnapLink when database size increases as well as its robustness for identifying appliance instances in the same category, viewing angles, distances, illumination conditions, and changes in the environment. We also conduct a micro-benchmark and a field study to show how QR codes and SnapLink can compensate each other. Finally, we examine failure cases as well as the energy consumption of our Android mobile client.

5.1 Experimental Setup

We use an RGB-Depth camera (an ASUS Xtion PRO LIVE in some rooms, and a Kinect in others) to capture videos and use RTABMap [30] to create 3D models. Our dataset contains videos of 39 rooms across 5 buildings in two university campuses, and 4 of the rooms are captured by a first-time user. The rooms include conference rooms, office cubicles, lounges, kitchens, hallways, etc. We estimate the average room size to be 150 sq. ft., meaning our deployment covers about 5,850 sq. ft. of total space, which is about the size of a typical small office building [11]. To our knowledge, our dataset covers a much larger space than those in previous vision-based object identification systems [9, 22, 28]. Our videos are sampled at 1 frame per second, and our entire dataset contains 4034 images, which means we can capture a room in 100 seconds and an entire small office building in about 67 minutes.

After recording a video, we use Android phones (an LG G2 Mini in some rooms, and a Nexus 5x in others) to capture 480×640 (or 640×480) test pictures of appliances from different angles and distances. The appliances are arbitrarily selected from all appliances and objects in the rooms, including both controllable objects (e.g., lights) and non-controllable objects (e.g., bookshelves) in our evaluation. Note that if we want to include more appliances from the captured rooms later, there is no need to go back and capture extra data (as required in image retrieval), since they are already included in our 3D models. In total, we added 263 labels for 179 appliances (some large appliances are labeled on multiple points, but will work with only one) using our labeling tool, and collected 1526 images for testing.

Our setup includes a SnapLink server running in a docker container on an Ubuntu machine, which has an Intel Xeon E5-2670 CPU and 256 GB of memory. We run a Python script as a client in the same docker container to send test images to the server over HTTP. To compare SnapLink with other vision-based systems ([9, 22, 28]), we also implemented an image retrieval based system, which simply finds the image with the most salient features in common with the query image. Specifically, we find the image p such that

$$p = \underset{i}{\operatorname{argmax}} \left| Words_i \cap Words_q \right| \tag{3}$$

where $Words_i$ and $Words_q$ are feature words of image i and the query image, respectively. To achieve efficient search, we build a word-image map, then vote and tally all database images while iterating through all words in the query image as described in [32]. It uses the same database we collected, but with every image labeled by the appliance closest to the image center.

5.2 Metrics

Our evaluation focuses on two figures of merit: accuracy and latency. Since users capture a query image of an appliance to initialize interactions, we argue that our system performance is more important for test images containing an appliance, as opposed to those containing no appliance (e.g., a blurry image captured by accident, or a white wall). Therefore, we define accuracy as *recall*:

$$\text{Accuracy} = \frac{\text{Number of Correctly Identified Images}}{\text{Number of Images Containing an Appliance}} \tag{4}$$

When a test image contains multiple appliances, we regard the appliance closest to the center of the image as the ground truth, which is also how the SnapLink server identifies the target appliance when multiple appliance labels are visible in the query image (Section 3.3).

Given that we have already showed the average local network latency to upload an 640 image is 70ms in Figure 9, we only focus on server side latency in our evaluation.

5.3 Accuracy and Latency

We first look at the accuracy and latency of image retrieval, image localization, and image localization with feature subsampling, by running and testing these three configurations on all our data. We choose to subsample 300 features, which is empirically optimal as shown in Section 5.4. Figure 11 shows the average accuracy as well as the average server processing times with standard deviations for each technique. Image retrieval only yields 66% accuracy with 177 ms average computation time, whereas image localization yields 94% accuracy with 198 ms average computation time. After applying our feature subsampling mechanism, SnapLink maintains 94% accuracy while reducing the computation time by 39% to 120 ms with a smaller standard deviation. This time reduction significantly helps to push the end-to-end latency down to an imperceptible value.

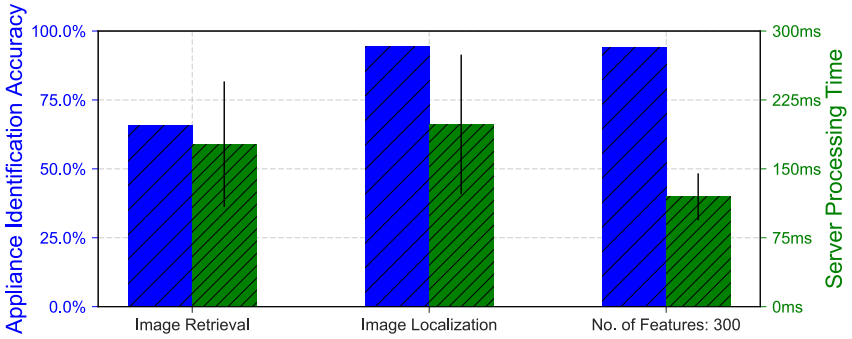


Fig. 11. Accuracy and identification time for image retrieval, image localization, and image localization with feature subsampling.

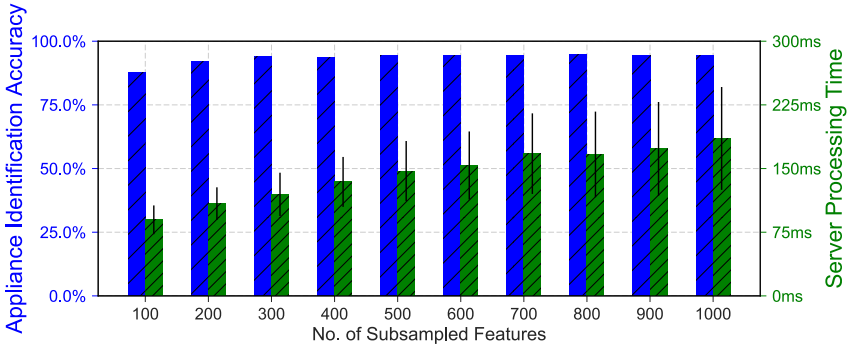


Fig. 12. Accuracy and identification time for different numbers of subsampled features. Accuracy does not further improve when the number of features exceeds 300.

5.4 Feature Subsampling

To further study how feature subsampling influences the accuracy and latency, we run image localization using our entire database while varying the number of subsampled features per image. Figure 12 shows that the accuracy saturates at 94% with 300 features subsampled for the kind of appliances we need to recognize in building environments, and adding more features only increases the server computation time. This shows that we do not need all features from the query image. Therefore, we choose to subsample 300 features in our system, as well as in the following evaluations.

5.5 Scalability

One of our design goals is to make SnapLink scale to the size of a commercial building. To further understand how feature subsampling reduces our server processing time, we run image localization with and without feature subsampling on different numbers of rooms and test with images from the respectively used rooms. Figure 13 shows a breakdown of the server processing time when performing image localization without subsampling for different sizes of 3D point clouds. The point clouds are created by selecting different combinations of rooms from the overall database. SURF takes about 50 ms, but can easily be reduced to less than 10 ms by using a GPU. Because we perform feature quantization, room identification and image localization both take a small amount

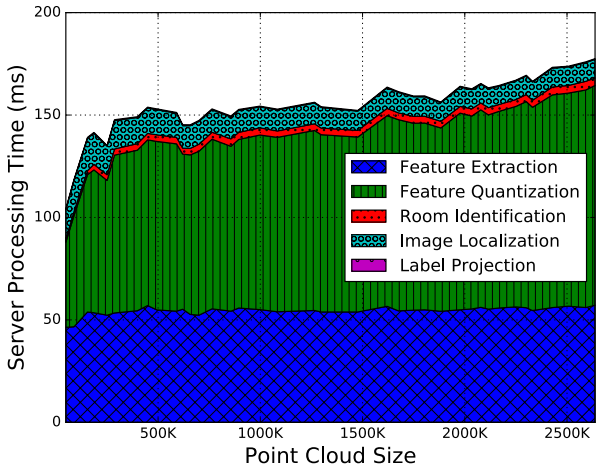


Fig. 13. Time breakdown vs. the size of point cloud using image localization without subsampling

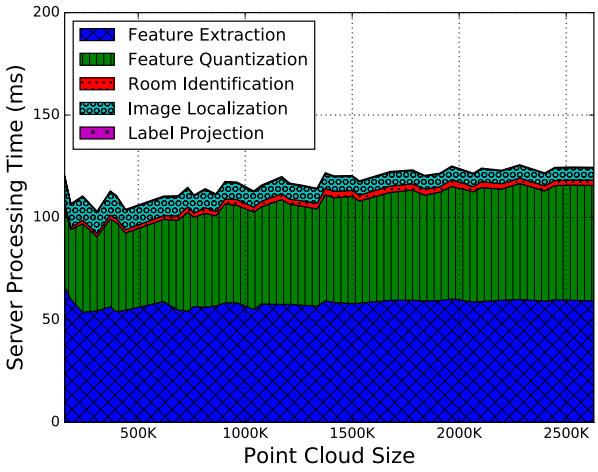


Fig. 14. Time breakdown vs. the size of point cloud using image localization with 300 features subsampled

of time. Label projection also takes a negligible amount time because of the small number of labels. However, because of the nature of KD-Trees, the feature quantization time increases logarithmically with the number of words. This generally increases as the size of the point cloud increases. This results in noticeable latency in the appliance identification process, which gets worse when deployed with more data in a larger building.

Figure 14 shows that feature subsampling effectively reduces the feature quantization time when the data size increases. It provides a tunable parameter to choose between accuracy and latency. When the data size gets larger, we can reduce the number of subsampled features to further reduce the latency, while sacrificing a small amount of accuracy. In future work, we plan to explore different subsampling strategies to select "better" features (rather than random ones) before the quantization.

5.6 Instance Recognition In Categories

Since a building contains many instances of appliances from the same category, it's important that SnapLink can distinguish instances among the same category with image localization. To study how SnapLink performs, we select the six most common categories among all 179 labeled appliances and summarize their identification accuracy among all instances in the category in Table 2. SnapLink achieves more than 95% accuracy for four out of the six categories and 89.2% for projector. However, it can only get 77.1% accuracy when identifying lights. We investigated this in our dataset, and found that many lights were turned on when we built the 3D model, which caused the image to underexpose, such as in Figure 15. This likely degrades the feature extraction performance of the image. In addition, many lights are located on the ceiling, where few distinct visual features can be found. In future work, we plan to add continuous trajectory estimation between query images using smartphone motion sensors to mitigate this problem.

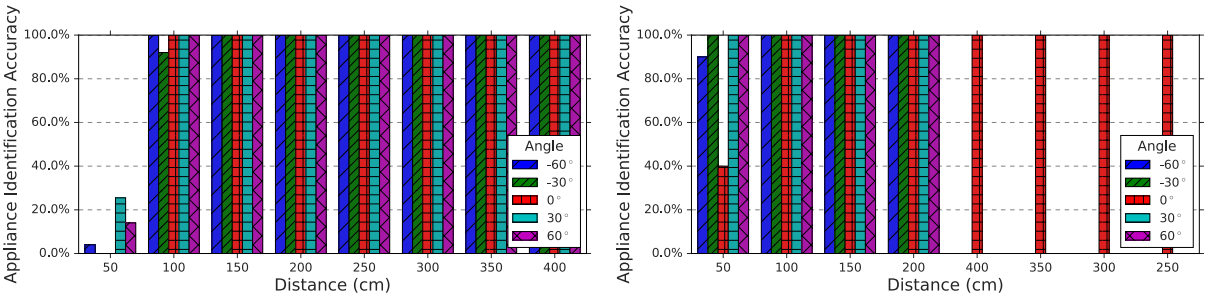
5.7 Angle and Distance

Since users can control appliances from arbitrary locations, SnapLink should be robust to different viewing angles and distances. To evaluate this, we select two sufficiently large lounges (i.e., Lounge 1 and Lounge 2, both in the 39 rooms in our dataset) and create a 3D model of each. Each lounge has three appliances labeled, one of which is our target appliance in this evaluation. We capture test images (50 per location in Lounge 2, and 10 in Lounge

Appliance Category	Number of Instances	Accuracy Among the Category
Printer	25	99.1%
Monitor	15	98.3%
Projector	12	89.2%
Light	12	77.1%
Wall Clock	10	98.7%
Microphone	9	95.8%

Table 2. Summary of instance identification accuracy among the six most common categories. SnapLink achieves high accuracy among each category except lights, which suffer from underexposure and a lack of visual features present on ceilings.

Fig. 15. Example underexposed image caused by a light, which can degrade feature extraction.



(a) Lounge 1

(b) Lounge 2 (no data from oblique angles when distances > 200 cm due to limited physical space)

Fig. 16. Identification accuracy at different angles and distances with Image Localization and 300 Features Subsampled. Query images close to the appliance tend to fail because they contain less visual context.

2) of the target appliance from 5 different angles with 30° intervals, each with different distances with 50 cm intervals (Because Lounge 1 is not spacious enough, we cannot take images from more than 200 cm except when the viewing angle is 0°). All test images are sent to a SnapLink server to be identified among the three appliances in the lounge along with all appliances in the other 38 rooms in our dataset. All tests are done with 300 features subsampled.

Figure 16 shows the identification accuracy at all locations. SnapLink only experiences significant failures when the viewing distance is small (i.e., ≤ 50 cm). The main reason is that when the query is taken from a close distance, not enough features may be captured, especially those providing distinctive contextual information. We argue that this will not happen often in practice, because most times users try to control appliances outside their arm range, such as a ceiling light or a projector. Another option to solve this problem is the attachment of a visual marker (e.g., QR code) to appliances that are usually within an occupant's arm range (see section 5.10).

5.8 Illumination Conditions

The illumination conditions in a building can be affected by various factors, such as lights turning on and off and window shades being opened and closed. SnapLink needs to be robust to common illumination changes when deployed. We conduct an experiment in an office room with 5 appliances (2 monitors, 2 speakers, and 1

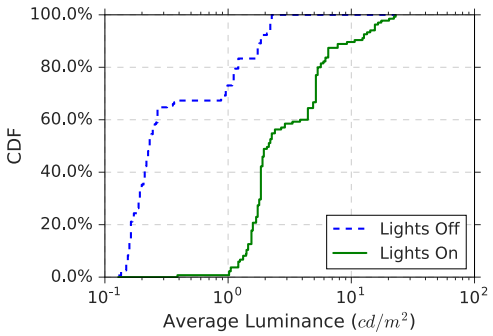


Fig. 17. CDF of average scene luminance (cd/m^2) ($C = 1$) when lights are on and off.

3D Model \ Testing	Lights On	Lights Off
Lights On + 39 Rooms	90.4%	93.7%
Lights Off + 39 Rooms	91.9%	100%

Table 3. Identification accuracy when images are tested against the 3D model and the 39 rooms in our dataset. It shows SnapLink is robust to illumination changes.

fan) under two different illumination conditions: all lights on (bright) and all lights off (dark), while all window shades are put down. Under each illumination condition, we capture a video to construct a 3D model, and capture at least 25 test images of each appliance from different viewing angles and distances.

To quantify the illumination condition when lights are on and off, we compute the average scene luminance (in cd/m^2) from the EXIF data of the test images using the following equation (described in [19]):

$$L = C\frac{k^2}{S \cdot t} \tag{5}$$

where L is the average scene luminance (in cd/m^2), k denotes the f-number, S denotes the ISO speed setting value, t denotes the exposure time (in seconds), and C is the hardware-dependent reflected-light meter calibration constant. k, S, and t can be found in the EXIF data, and we arbitrarily set C to be 1. Since we are only comparing relative luminances captured using the same camera, the value of C can be an arbitrary constant. Figure 17 shows the CDF of the average luminance (when $C = 1$) with a log-scale x axis. We can see that the scene is about 10 times brighter when lights are on than off.

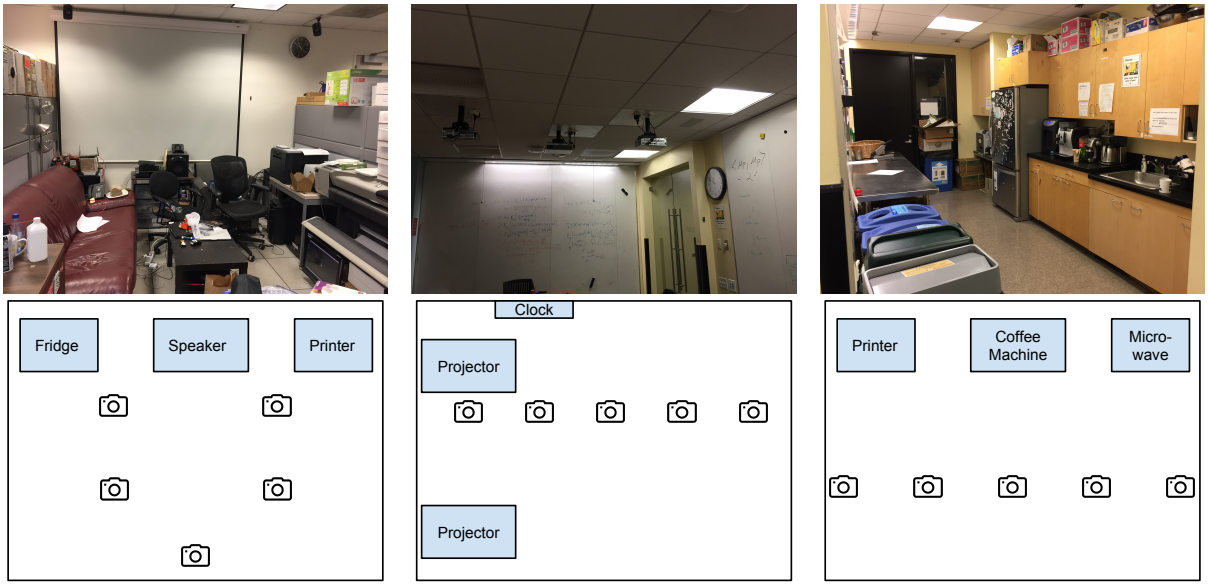
Table 3 summarizes the identification accuracy when test images are tested against a 3D model and the 39 rooms. Test images captured under different illumination conditions do not yield worse identification accuracy than those captured under the same illumination condition. This is because of the robust nature of salient features [5] as well as the auto exposure adjustment in modern smartphone cameras. Note that dark test images have higher identification accuracy than the bright test images in the bright 3D model, which is because the two test image sets are taken from different arbitrary angles and distances.

5.9 Changes in the Environment

Room	Appliances
Lounge	1 Fridge, 1 Speaker, and 1 Printer
Conference Room	2 Projectors, 1 Clock
Kitchen	1 Microwave, 1 Coffee Machine, and 1 Printer

Table 4. Summary of appliances in the changing environment experiment.

To show how SnapLink is robust to everyday changes after a 3D model is captured, we construct 3D models of 3 rooms (which are among the 39 rooms in our dataset) containing 9 appliances, and evaluate SnapLink



(a) Lounge **(b) Conference Room** **(c) Kitchen**

Fig. 18. Changing environment experiment setup. Each room contains 3 appliances and is shared by many people in a commercial building, where changes to the environment happen every day. We capture 20 test images of each appliance at each capture location every day (57,600 images in total).

every day thereafter for 64 consecutive days. Table 4 summarizes the 3 rooms and 9 appliances, and Figure 18 shows pictures of the 3 rooms, which are all shared by many occupants in a commercial building and naturally change all the time without our intervention. As examples, actively changed items include: objects on the coffee table and chairs in the lounge, content on the whiteboard and projector screen in the conference room, and items on the table and countertop in the kitchen. To ensure images captured every day have the same distribution in terms of viewing angles and distances, we capture 20 test images of every appliance from five locations in each room, with minor viewing point movements in the arm's range between images. The lower part of Figure 18 shows test image capture locations.[4] In total, this experiment generates 57600 test images (20 images $\times$ 5 locations $\times$ 3 appliances $\times$ 3 rooms $\times$ 64 days). To reduce the authors' bias, we also asked a volunteer to collect about half of all the test images, captured on dates when they were available. While identifying appliances in the test images, out search space also include all appliances from the remaining 36 rooms.

Figure 19 shows the identification accuracy for each of the three rooms as an increasing number of days have elapsed since the initial 3D models were captured. SnapLink's accuracy is mostly above 90% during the entire experiment, but varies day by day due to environmental changes and varying quality in the captured test images. For example, the dip in accuracy for the conference room from day 43 to day 45 is caused by blurry test images. Furthermore, our 64-day experiment shows no clear trend of degrading accuracy, which suggests that SnapLink can retain high accuracy throughout an even longer period. This robustness comes from two factors: (1) the RANSAC algorithm that we use while solving the PnP problem, and (2) the environments did not change dramatically, which we argue is generally true for most commercial buildings.

[4]The room dimension ratios are not to exact scale.

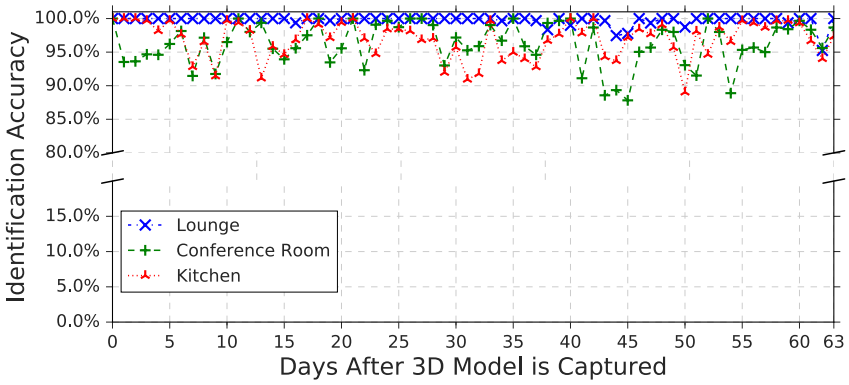


Fig. 19. Identification accuracy over time as the environment changes. The accuracy fluctuates caused by everyday changes in the environment and occasional blurry test images. We see no clear trend of degrading accuracy, meaning the database can be used for a longer period of time.

5.10 Comparison with QR Codes

QR Codes are two-dimensional machine-readable barcodes that are quick to decode with error correction. They form the basis of a very popular technique for object identification. We use a micro-benchmark to show the advantages and disadvantages of QR codes for our application. We also show how QR codes can help improve SnapLink performance with a field study involving 7 appliances.

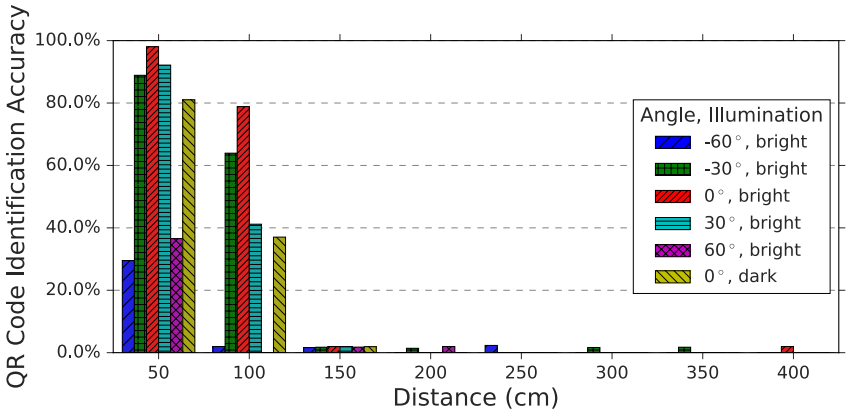


Fig. 20. QR code recognition rate at different viewing distances and angles, as well as under different illumination conditions.

5.10.1 Micro-benchmark. In the micro-benchmark experiment, a QR code encoding an 11-byte string with error correction level "L" [27] is printed at size 7.5 cm × 7.5 cm and attached to a wall in a well-lit conference room. From different viewing distances and angles, we capture 50 images of the QR code at each location using an LG G2 Mini Android phone. Since QR codes can be quickly decoded on phones with no need for offloading, the images are captured at resolution 2448 × 3264, which is much higher than the image resolution (i.e., 640 × 480) used in SnapLink. We also did the same experiment from a viewing angle of 0° in a darker environment, where

with all lights are turned off while there is still ambient light from the windows and adjacent rooms. We use ZBar[5], a popular open source barcode reader, to decode QR codes in the images on a Linux machine. Since QR codes yield few false positives with Reed-Solomon error correction [27], we consider a QR code to be correctly decoded if the encoded string is among the set of decoded content.

Figure 20 shows the QR code recognition accuracy at different locations and under different illumination conditions. It shows that accuracy drops sharply when the viewing distance is larger than 150 cm or when the viewing angle is larger than 60°. A darker environment also diminishes the recognition accuracy. As we show in Section 5.7, SnapLink performs poorly when viewing distance is low with insufficient salient features. Hence, QR codes perfectly compensate for this drawback of SnapLink.

Fig. 21. 4 of the 7 appliances in the field study setup, with 7.5 cm × 7.5 cm QR codes attached.

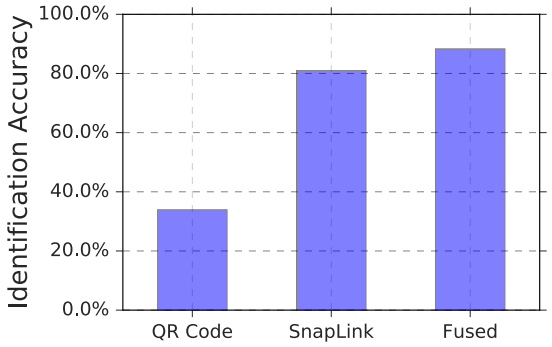


Fig. 22. Identification accuracy of QR codes, SnapLink, and a fusing system that combines both.

5.10.2 Field Study. To show how QR codes can be integrated into SnapLink and compensate for its drawbacks, we conduct a field study in a room (which is among the 39 rooms in our dataset) with 7 appliances (3 fridges, 3 printers, and 1 speaker). Each appliance is attached with a 7.5 cm × 7.5cm QR code encoding its unique ID. We capture more than 50 test images of each appliance from different angles and distances, including from low distances (e.g., < 30 cm) where SnapLink performs poorly. All images are tested using both ZBar [53] and Snaplink, as well as a fusing system, which first uses ZBar and subsequently uses SnapLink if no QR Code is decoded. Because QR code identification features a low rate of false positives, we consider the identification successful as long as the ground truth appliance ID is among all decoded QR codes from ZBar. When using Snaplink, we distinguish among the 7 appliances as well as all appliances in the other 38 rooms in our dataset.

Figure 22 shows the identification accuracy achieved by each of the three approaches. QR codes and SnapLink yield 34% and 81% accuracy respectively, and the fused result improves the accuracy to 88%. This shows that QR codes and SnapLink can complement each other very well for the purpose of appliance identification from an arbitrary viewing distance and angle.

5.11 Failure Analytics

To examine the causes of failures in our system, we summarize six categories of common failures that we have found over our deployment period. Figure 23 shows these failure cases with examples. In each example, the

[5] http://zbar.sourceforge.net/

(a) Labels Too Close (b) Label Occluded (c) Similar Scenes

(d) Not Covered in DB (e) Lack of Features (f) Incorrect Label

Fig. 23. Common failure cases (left is query image, right is the image in database found by image retrieval)

lefthand image is the query image, and the righthand image is its closest image using image retrieval, which helps demonstrate the problem. In the results, (a) contains a water machine and a coffee machine that are close to each other, and the same is true of their respective labels. This situation is sensitive to image localization accuracy. (b) shows that the user intends to identify the fridge, but a printer is occluded by the fridge and has a label that is closer to the image center. (c) shows two scenes that look similar, and our system cannot distinguish between them using their SURFs. In (d), the flowerpot is the target appliance, but it's barely captured by any image in the training video. (e) shows that the ceiling light does not have a sufficiently unique visual context, so our system mistakenly localizes it as another ceiling light. (f) shows the query image was identified as the light, but labeled incorrectly as the projector. These failures cases can be useful to guide our future work on improving the success rate and demonstrate that most failures are due to external factors rather than artifacts of the system itself.

5.12 App Energy Usage

Our design goal is to make the SnapLink mobile client power efficient. Because the LG G2 Mini uses the Qualcomm MSM8226 Snapdragon 400 SoC, we use the Qualcomm Trepn Power Profiler to measure accurate per-app power usage by leveraging specific Snapdragon features.[6] We adjust the screen brightness to 100% and measure three different power consumption traces for 60 seconds: overall power usage while showing the home screen (idle), SnapLink power usage while streaming a camera view to screen (preview), and SnapLink power usage while sequentially sending query images as fast as possible using WiFi. Figure 24 shows the power usage trends, where idling consumes on average 0.48 Watts, preview consumes on average 1.64 Watts, and preview with continuous queries consumes on average 2.78 Watts. With the 3.8V 2440 mAh battery in the test phone, the expected battery life when using SnapLink to preview and continuously identify appliances is 2.84 hours ($\frac{(2.44 \text{ Amp hour}) * (3.8 \text{ Volt})}{0.48+2.78 \text{ Watt}}$).

[6] https://developer.qualcomm.com/software/trepn-power-profiler

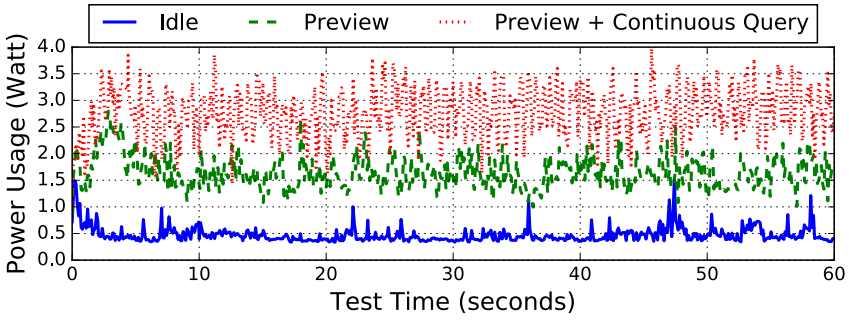


Fig. 24. Energy Usage over Time

6 RELATED WORK

In this section, we discuss existing appliance identification approaches and compare them with SnapLink.

6.1 Vision-based Object Interaction

With many recent improvements in computer vision (e.g., deep learning [10, 31] augmented reality [22]), vision-based human object interactions have been well studied. While general augmented reality systems focus on understanding a scene and how to naturally overlay virtual objects, we focus on quick and accurate image localization based appliance instance identification in a commercial building. Mayer et al. [35] built a mobile interface to display information and to control appliances, which internally uses image retrieval to find the appliance in the current view. Overlay [22] is an object tagging application that uses image retrieval to find the object in the current camera view. It relies on an always-on camera to provide a continuous video stream to narrow down the search space, whereas our application requires the user to be able to take her phone out to start controlling appliances at an arbitrary time. Kong et al. [28] used convolutional neural networks to recognize the category of appliances in the current camera view and identify the instances using unsupervised activity recognition and WiFi-based indoor localization systems. However, activity recognition and indoor localization cannot differentiate appliances of the same type in the same room, such as ceiling lights and printers in a copy room. Snap-To-It [9] allows users to take a picture to identify an appliance using image retrieval, using smartphone sensors for location validation. All of these works use image retrieval for instance recognition that requires many images to be captured for every appliance and each of the images to be labeled, which is a prohibitive effort for large scale deployment in a commercial building. Instead, SnapLink overcomes this problem by constructing a 3D model and allowing people to label 3D points with a labeling tool.

Fiducial markers (e.g., QR codes [50], ARToolKit [25], AprilTags [39]) are also used for vision-based object interactions. They encode appliance IDs and are attached to appliances for visual identification. However, as we discussed in Section 5.10, markers of practical sizes are hardly recognizable from a room size distance (e.g., 3 meters) or a large angle. Nevertheless, fiducial markers complement SnapLink very well when query images are taken from a small distance.

6.2 Other Object Identification Approaches

Besides vision-based appliance identification, people have explored many other approaches. However, they are not ideal to be applied to commercial buildings. First, some existing approaches do not provide an easy interface to users. Many BMS services allow users to type and send a query statement following a strict syntax [3, 7, 29], which requires manual typing as well as background knowledge of the building. Voice- and gesture-based appliance

identification systems [43, 49] allow users to speak a statement or do a gesture as a control command. They have been successfully applied in many residential buildings [12]. However, because they require users to memorize a unique voice command (or gesture) for every single appliance and control command, they cannot be deployed in a commercial building, which generally has hundreds to thousands of appliances and is shared by occupants who may not be familiar with the building.

Second, some approaches introduce significant deployment overhead. For instance, using traditional remote controllers or switches for all appliances will incur significant deployment overhead and cannot benefit from the flexibility and extensibility of modern BMSs. Some systems allow users to browse appliances in an inventory list or on a 2D/3D-graphic map of a building [18], which requires a user to be aware of her current location and orientation on a large building map. To solve this problem, other systems narrow down the list using indoor localization systems [17], which usually require extra deployment and do not work for interactions at a distance [33]. Some require installing extra infrastructure on every appliance, such as laser or infrared signal receivers [40, 54], Bluetooth dongles [17], and NFC or RFID tags [51], which introduces large deployment overhead in a commercial building.

In spite of these drawbacks, these approaches can be combined with vision-based approaches to enable better appliance interactions. For example, we can build a system that provides the voice commands for an appliance after it is identified by SnapLink and then receives voice-based commands from users.

7 DISCUSSION AND FUTURE WORK

As we deploy and evolve SnapLink, we see some potential improvements we can add to our system, as well as several interesting applications that can be enabled.

7.1 System Improvements

Integrate Category Recognition: Although we have shown that SnapLink is robust to daily changes in the environment for months after the initial 3D model is created, image localization may fail to provide robust instance recognition when appliances themselves are moved, or enough of the environment is changed. We plan to add category recognition to validate localization results and detect environmental changes. For example, if an appliance is removed, category recognition can detect that and issue an alert for label updates.

Crowdsource Data and Update 3D Model: A 3D model can also be built from crowdsourced images and videos from building occupants. We plan to add this feature to reduce deployment overhead for building manager. Moreover, crowdsourced data can be used to detect changes and update 3D models over time.

Improve Labeling Process: Besides data collection, labeling is the only aspect of SnapLink that requires human intervention. To minimize human effort, we have a labeling tool that enables one simple click-and-type interaction to label each appliance. We plan to push this further with more intuitive and automatic mechanisms. For example, we can allow users to point their smartphone camera at an appliance and label it by speaking its ID aloud. We can also use category recognition to provide an initial guess for appliance IDs.

Use Extra Features: Apart from SURF, many other features can be utilized to help identify instances. For example, Optical Character Recognition (OCR), which recognizes text from images, can be used for places where text is present, such as room number plates.

Integrate Visual Markers: Our experiments show that image localization accuracy can be low when the viewing distance is small because these kinds of images cannot capture sufficient unique visual features. We have shown how QR codes can be used to mitigate this problem, although with some deployment overhead. We plan to

integrate SnapLink with various visual markers, such as QR codes and Apriltags [39], and therefore improve its accuracy on appliances that are usually controlled within arm's range.

7.2 Applications Inspired by SnapLink

Location-based Authorization: SnapLink demonstrates the robustness and ease of use of image localization inside buildings. This inspires us to build a location-based authorization service by asking a user to capture and upload an arbitrary video of her current room. The video can be localized by our system to validate the location, and the randomness of the video can be utilized to avoid replay attacks. This is useful in scenarios where only occupants in a room are allowed to control appliances in that room.

Indoor Navigation: With SnapLink deployed, we can also use it to localize people and display navigation guidance. For example, we can build an app in a museum that shows information about nearby art and navigate people to certain locations.

Building Chronology: Insired by [34], we can collected users' query images over time and construct a visual chronology of a building model. This not only visualizes the history of building change, but also can potentially help find lost items.

8 CONCLUSION

In this paper, we presented SnapLink, an accurate and responsive vision-based appliance identification system that enables users to control building appliances in their sight by using ubiquitous smartphones. This work started with an intuition: "What we see is what we control." SnapLink leverages image localization for better appliance identification accuracy and much lower deployment overhead in commercial buildings. On top of the standard image localization process, we introduce geo-partitioning to enable easy and accurate 3D model construction, and provide a labeling tool to simplify appliance labeling process. In addition, SnapLink uses a feature sub-sampling mechanism which reduces computation time and scales well when database size increases, without losing identification accuracy. We built an end-to-end system including SnapLink, a BMS, and a smartphone application and test it at a building scale using 1526 test images among 39 rooms captured by 4034 images. Our results show that SnapLink achieves 94% identification accuracy with 120 ms of server processing time and is robust to different viewing distances and angles, as well as changes in the environment (e.g., illumination, daily occupant activities). We believe that SnapLink forms a significant step in enabling users to interact with their environments using computer vision technologies. As future work, we plan to move SnapLink towards a smart building service, where many more applications can benefit from image localization inside buildings.

ACKNOWLEDGMENTS

The authors would like to thank Albert Goto for help with collecting experimental data. This work is supported in part by the National Science Foundation grant CPS-1239552 (SDB) and California Energy Commission.

REFERENCES

[1] Sameer Agarwal, Yasutaka Furukawa, Noah Snavely, Ian Simon, Brian Curless, Steven M Seitz, and Richard Szeliski. 2011. Building rome in a day. *Commun. ACM* 54, 10 (2011), 105–112.

[2] Michael P Andersen, Gabe Fierro, and David E Culler. 2016. Enabling synergy in iot: Platform to service and beyond. *Journal of Network and Computer Applications* (2016).

[3] Bharathan Balaji, Arka Bhattacharya, Gabriel Fierro, Jingkun Gao, Joshua Gluck, Dezhi Hong, Aslak Johansen, Jason Koh, Joern Ploennigs, Yuvraj Agarwal, et al. 2016. Brick: Towards a Unified Metadata Schema For Buildings. In *Proceedings of the 2nd ACM International Conference on Embedded Systems for Energy-Efficient Built Environments*. ACM.

[4] Bharathan Balaji, Jian Xu, Anthony Nwokafor, Rajesh Gupta, and Yuvraj Agarwal. 2013. Sentinel: occupancy based HVAC actuation using existing WiFi infrastructure within commercial buildings. In *Proceedings of the 11th ACM Conference on Embedded Networked Sensor Systems*. ACM, 17.
[5] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. 2006. Surf: Speeded up robust features. In *European conference on computer vision*. Springer, 404–417.
[6] Gary Bradski and Adrian Kaehler. 2008. *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.".
[7] Stephen Dawson-Haggerty, Xiaofan Jiang, Gilman Tolle, Jorge Ortiz, and David Culler. 2010. sMAP: a simple measurement and actuation profile for physical information. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*. ACM, 197–210.
[8] Stephen Dawson-Haggerty, Andrew Krioukov, Jay Taneja, Sagar Karandikar, Gabe Fierro, Nikita Kitaev, and David Culler. 2013. BOSS: building operating system services. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*. 443–457.
[9] A de Freitas, Michael Nebeling, Xiang 'Anthony' Chen, Junrui Yang, ASKK Ranithangam, and Anind K Dey. 2016. Snap-to-it: a user-inspired platform for opportunistic device interactions. In *Proceedings of the 34th Annual ACM Conference on Human Factors in Computing Systems (CHI 2016)*.
[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 248–255.
[11] Michael Deru, Kristin Field, Daniel Studer, Kyle Benne, Brent Griffith, Paul Torcellini, Bing Liu, Mark Halverson, Dave Winiarski, Michael Rosenberg, et al. 2011. US Department of Energy commercial reference building models of the national building stock. (2011).
[12] echo 2016. Amazon Echo. (2016). https://www.amazon.com/Amazon-Echo-Bluetooth-Speaker-with-WiFi-Alexa/dp/B00X4WHP5E
[13] Varick L Erickson and Alberto E Cerpa. 2010. Occupancy based demand response HVAC control strategy. In *Proceedings of the 2nd ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building*. ACM, 7–12.
[14] Romain Fontugne, Jorge Ortiz, Nicolas Tremblay, Pierre Borgnat, Patrick Flandrin, Kensuke Fukuda, David Culler, and Hiroshi Esaki. 2013. Strip, bind, and search: a method for identifying abnormal energy consumption in buildings. In *Proceedings of the 12th international conference on Information processing in sensor networks*. ACM, 129–140.
[15] Marc Fountain, Gail Brager, Edward Arens, Fred Bauman, and Charles Benton. 1994. Comport control for short-term occupancy. *Energy and Buildings* 21, 1 (1994), 1–13.
[16] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. 2015. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review* 43, 1 (2015), 55–81.
[17] Jonathan Fürst, Kaifei Chen, Mohammed Aljarrah, and Philippe Bonnet. 2016. Leveraging physical locality to integrate smart appliances in non-residential buildings with ultrasound and Bluetooth low energy. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*. IEEE, 199–210.
[18] Jonathan Fürst, Gabe Fierro, Philippe Bonnet, and David E Culler. 2014. BUSICO 3D: building simulation and control in unity 3D. In *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*. ACM, 326–327.
[19] Helke Gabele and D Wüller. 2006. The usage of digital cameras as luminance meters. *Germany, Cologne: University of Applied Sciences Cologne, Diploma Thesis* (2006).
[20] Fabian Gieseke, Justin Heinermann, Cosmin E Oancea, and Christian Igel. 2014. Buffer kd Trees: Processing Massive Nearest Neighbor Queries on GPUs.. In *ICML*. 172–180.
[21] Wenlu Hu, Brandon Amos, Zhuo Chen, Kiryong Ha, Wolfgang Richter, Padmanabhan Pillai, Benjamin Gilbert, Jan Harkes, and Mahadev Satyanarayanan. 2015. The case for offload shaping. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 51–56.
[22] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2015. OverLay: Practical Mobile Augmented Reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*. ACM, 331–344.
[23] Puneet Jain, Justin Manweiler, and Romit Roy Choudhury. 2016. Low Bandwidth Offload for Mobile AR. In *Proceedings of the 12th International on Conference on emerging Networking EXperiments and Technologies*. ACM, 237–251.
[24] Herve Jegou, Matthijs Douze, and Cordelia Schmid. 2008. Hamming embedding and weak geometric consistency for large scale image search. *Computer Vision–ECCV 2008* (2008), 304–317.
[25] I Poupyrev H Kato, Mark Billinghurst, and Ivan Poupyrev. 2000. Artoolkit user manual, version 2.33. *Human Interface Technology Lab, University of Washington* 2 (2000).
[26] Nabeel Younus Khan, Brendan McCane, and Geoff Wyvill. 2011. SIFT and SURF performance evaluation against various image deformations on benchmark dataset. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*. IEEE, 501–506.
[27] Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Lindsay Munroe, Sebastian Schrittwieser, Mayank Sinha, and Edgar Weippl. 2010. QR code security. In *Proceedings of the 8th International Conference on Advances in Mobile Computing and Multimedia*. ACM, 430–435.
[28] Quan Kong, Takuya Maekawa, Taiki Miyanishi, and Takayuki Suyama. 2016. Selecting home appliances with smart glass based on contextual information. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. ACM,

97–108.
[29] Andrew Krioukov, Gabe Fierro, Nikita Kitaev, and David Culler. 2012. Building application stack (BAS). In *Proceedings of the Fourth ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. ACM, 72–79.
[30] Mathieu Labbe and Francois Michaud. 2013. Appearance-based loop closure detection for online large-scale and long-term operation. *Robotics, IEEE Transactions on* 29, 3 (2013), 734–745.
[31] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.
[32] Jason Zhi Liang, Nicholas Corso, Eric Turner, and Avideh Zakhor. 2015. Image-based positioning of mobile devices in indoor environments. In *Multimodal Location Estimation of Videos and Images*. Springer, 85–99.
[33] Dimitrios Lymberopoulos, Jie Liu, Xue Yang, Romit Roy Choudhury, Vlado Handziski, and Souvik Sen. 2015. A realistic evaluation and comparison of indoor location technologies: experiences and lessons learned. In *Proceedings of the 14th International Conference on Information Processing in Sensor Networks*. ACM, 178–189.
[34] Kevin Matzen and Noah Snavely. 2014. Scene Chronology. In *Proc. European Conf. on Computer Vision*.
[35] Simon Mayer, Markus Schalch, Marian George, and Gábor Sörös. 2013. Device recognition for intuitive interaction with the web of things. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*. ACM, 239–242.
[36] Robert B Miller. 1968. Response time in man-computer conversational transactions. In *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*. ACM, 267–277.
[37] Marius Muja and David G Lowe. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 11 (2014), 2227–2240.
[38] Jakob Nielsen. 1994. *Usability engineering*. Elsevier.
[39] Edwin Olson. 2011. AprilTag: A robust and flexible visual fiducial system. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 3400–3407.
[40] Shwetak N Patel and Gregory D Abowd. 2003. A 2-way laser-assisted selection scheme for handhelds in a physical environment. In *UbiComp 2003: Ubiquitous Computing*. Springer, 200–207.
[41] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2007. Object retrieval with large vocabularies and fast spatial matching. In *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 1–8.
[42] James Philbin, Ondrej Chum, Michael Isard, Josef Sivic, and Andrew Zisserman. 2008. Lost in quantization: Improving particular object retrieval in large scale image databases. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 1–8.
[43] Qifan Pu, Sidhant Gupta, Shyamnath Gollakota, and Shwetak Patel. 2013. Whole-home gesture recognition using wireless signals. In *Proceedings of the 19th annual international conference on Mobile computing & networking*. ACM, 27–38.
[44] Long Quan and Zhongdan Lan. 1999. Linear n-point camera pose determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 21, 8 (1999), 774–780.
[45] Radu Bogdan Rusu and Steve Cousins. 2011. 3d is here: Point cloud library (pcl). In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 1–4.
[46] Torsten Sattler, Bastian Leibe, and Leif Kobbelt. 2011. Fast image-based localization using direct 2d-to-3d matching. In *2011 International Conference on Computer Vision*. IEEE, 667–674.
[47] Josef Sivic and Andrew Zisserman. 2003. Video Google: A text retrieval approach to object matching in videos. In *null*. IEEE, 1470.
[48] tango 2016. Project Tango. (2016). https://get.google.com/tango/
[49] Michel Vacher, Dan Istrate, François Portet, Thierry Joubert, Thierry Chevalier, Serge Smidtas, Brigitte Meillon, Benjamin Lecouteux, Mohamed Sehili, Pedro Chahuara, et al. 2011. The sweet-home project: Audio technology in smart homes to improve well-being and reliance. In *2011 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. IEEE, 5291–5294.
[50] Jian-tung Wang, Chia-Nian Shyi, T-W Hou, and CP Fong. 2010. Design and implementation of augmented reality system collaborating with QR code. In *Computer Symposium (ICS), 2010 International*. IEEE, 414–418.
[51] Evan Welbourne, Leilani Battle, Garrett Cole, Kayla Gould, Kyle Rector, Samuel Raymer, Magdalena Balazinska, and Gaetano Borriello. 2009. Building the internet of things using RFID: the RFID ecosystem experience. *IEEE Internet computing* 13, 3 (2009).
[52] Jianxiong Xiao, Andrew Owens, and Antonio Torralba. 2013. SUN3D: A database of big spaces reconstructed using sfm and object labels. In *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE, 1625–1632.
[53] zbar 2017. ZBar bar code reader. (2017). http://zbar.sourceforge.net/
[54] Ben Zhang, Yu-Hsiang Chen, Claire Tuna, Achal Dave, Yang Li, Edward Lee, and Björn Hartmann. 2014. HOBS: head orientation-based selection in physical spaces. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*. ACM, 17–25.
[55] Liang Zheng, Yi Yang, and Qi Tian. 2017. SIFT meets CNN: A decade survey of instance retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

Received February 2017; revised August 2017; accepted October 2017