Intuitive Appliance Identification using Image Matching in Smart Buildings

Kaifei Chen
John Kolb
Jonathan Fürst
Dezhi Hong
David E. Culler
Randy H. Katz

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2015-200
http://www.eecs.berkeley.edu/Pubs/TechRpts/2015/EECS-2015-200.html

September 15, 2015

Acknowledgement

This work is supported in part by the National Science Foundation under grant CPS-1239552 (SDB).

Intuitive Appliance Identification using Image Matching in Smart Buildings

Kaifei Chen
UC Berkeley
kaifei@berkeley.edu

John Kolb
UC Berkeley
jkolb@berkeley.edu

Jonathan Fürst
IT University of Copenhagen
jonf@itu.dk

Dezhi Hong
University of Virginia
hong@virginia.edu

David E. Culler
UC Berkeley
culler@berkeley.edu

Randy H. Katz
UC Berkeley
randykatz@berkeley.edu

ABSTRACT

The number of smart appliances is rapidly increasing to foster the Internet of Things. However, identifying an appliance for interaction in a building is also becoming more confusing. Most work on simplifying contextual appliance identification and selection either requires extra effort from users or infrastructure deployments. In this paper, we present an intuitive system for users to "look up" appliances in a smart building using an image. It constructs an annotated 3D visual model of a building interior using RGB-D cameras and matches a user-provided image on the model to determine the appliances in the image. Our system matched 98% images on a public robot-collected dataset and achieved 100% recall and precision among them. We also deployed the system in our lab with human captured RGB-D videos and images, which have more degrees of freedom and noise than robots. We matched 71% of the images. Of the matched images, 63% of them achieved 80% recall, and 78% achieved 80% precision.

1. INTRODUCTION

There are many smart home appliances emerging today, such as programmable thermostats, light bulbs, and fridges. The intent is to make everyday things software-controllable and connected to the Internet, namely the Internet of Things. Plenty of work has been done towards connecting and managing appliances to provide a more descriptive and programmable building [5, 6]. Even though it is desirable to delegate more monitoring and actuation to applications and services, people still need to interact directly with smart appliances. However, as the number of smart appliances grows in the environment, identifying which appliances to interact with through software becomes harder and more tedious for users.

Although researchers have explored different approaches to identify and interact with appliances, their approaches are not intuitive because of two major problems. First, some approaches require users to describe the appliance in cumbersome ways. For example, they build appliance directories. Users can query appliances with a formal query statement [5], which enforces strict syntax. Or, they can use natural language sentences with Amazon Echo, Google Now, or Microsoft Cortana. However, it is generally difficult to have a unique human-friendly description of a device, especially when multiple identical instances are at the same location. Second, many approaches require the deployment of additional infrastructure. Some works require setting up laser [29] or infrared [35] signal receivers on appliances, and a user uses a special signal transmitter for interaction. Other work depends on indoor localization [23], which generally produces tens of centimeters error and does not work for interactions at a distance. Therefore, given these drawbacks, we employ a vision-based approach, which is both intuitive for users and implies minimum deployment overhead.

In this paper, we explore an intuitive way of identifying ubiquitous appliances: *what you see is what you interact with.* We argue that with the advance in computer vision and image processing techniques, we should be able to identify the objects more easily and interact with them in a straightforward manner. To achieve this goal, we develop a system comprised of two major components: a modeling phase and a matching phase. In the modeling phase, the building manager needs to collect and mark a visual 3D building model. In the matching phase, users can identify appliances on the precomputed model using a smart phone to interact with these appliances with minimum effort. We aim to build a system that is intuitive in both phases. First, it should be intuitive and simple for a building manager to build a visual model. Second, it should be intuitive for users to identify and interact with appliances, which is not only supported by our image matching system, but also existing building management systems that provide clear and intuitive interfaces to appliances as well as responsive interactions.

To achieve our goal, we adopted a suite of computer vision techniques based on the specific requirements from our system. For example, we need to label appliances, but also require a fast response which contains point visibility analysis. We propose to avoid all accurate contour or box annotation in computer vision and only label the center points of appliances. Then we can project the annotated points to the camera plane and determine which are in the image

view, therefore avoiding all heavy computation in conventional visibility analysis, such as surface construction or hidden point removal [17]. We implemented our system using Real-Time Appearance-Based Mapping (RTAB-Map) [20], an open-source project for Simultaneous Localization And Mapping (SLAM). We tested our system on the RTAB-Map multi-session data set [21] collected by a 4-wheel robot, and successfully matched 98% of test images. We achieved 100% recall and precision on all appliances among the matched images. As an in-situ study, we also constructed a 3D model for our lab with 1075 images. The 3D model is captured with a human taking an RGB-D video, which generates more noise and has more degrees of freedom than a video captured by a robot. We collected images of appliances for identification testing after 4 days of normal use of the lab. We matched 71% of the images onto the 3D model. Of the matched images, 63% of them achieved 80% recall, and 78% achieved 80% precision.

We summarize our contributions as follows:

- We propose an intuitive way to identify smart appliances for interactions using a smart phone camera.
- We provide in-depth discussions on design choices among computer vision and image processing techniques for our particular purpose in smart buildings.
- We develope an end-to-end system to prove the efficiency of our approach.[1]
- We conduct a comprehensive evaluation of our system on a public dataset and a lab deployment.

The rest of this paper is organized as follows: Section 2 describes related work from object identification and computer vision. Section 3 elaborates our technical requirements and design choices. Section 4 discusses details of the system architecture and algorithms. Section 5 presents the evaluation results on several aspects of our system. Section 6 and 7 discuss and conclude the paper.

2. RELATED WORK AND BACKGROUND

In this section, we first talk about works that aim to simplify human-object interactions. We discuss their approaches and drawbacks compared to vision based approaches. After that, we summarize the work in computer vision that enables the components of our system.

2.1 Appliance Identification and Interaction

Much work has been done to enable easier interaction between humans and appliances. Many of them require extra infrastructure deployment on either the appliance or the user. As examples, Kemp et al. [18] make users illuminate an appliance with a laser pointer and use a camera to capture the selection. HOBS [35] uses an infrared transmitter with a receiver installed on each appliance. Goggles [4] is a face-mounted device that tracks eye movements for appliance selection.

[1]The code and documentation are available at `https://github.com/SoftwareDefinedBuildings/SDB3D`

To avoid heavy deployment overhead, other works make use of sensors on commodity smart phones. Rekimoto et al. [31] attach 2D markers on appliances and use a smart phone camera to scan and identify them, but 2D markers suffer from inefficiency at distance and with poor illumination [16]. Tricorder [23] uses signal strength-based indoor localization to display nearby devices on a map. However, state-of-the-art indoor localization systems can only achieve tens of centimeters accuracy [24], and do not work when a user wants to control appliances from a distance.

Other approaches integrate natural language processing into their systems to provide a more intuitive way for users to identify and interact with appliances. For example, Amazon Echo is a recent product that parses spoken natural language, potentially for the control of appliances in a smart home. However, uniquely describing an appliance is difficult especially when there are duplicate appliances in the vicinity such as lights.

In comparison to audio, visual information is more intuitive and straightforward. Augmented Reality (AR) is the concept of overlaying information about or interface of objects in an image. Heun et al. [14] built an AR interface for smart devices, but do not discuss how to recognize the objects. Mayer et al. [25] and Jain et al. [15] collected a set of images in the building and manually labeled the appliances on the images. When a user's camera view can match any of the appliance images in terms of local features, such as Speeded Up Robust Features (SURF) [3], they display relevant AR information. However, because local feature-based image matching is not robust enough from different view angles [28], one object needs multiple images from different angles and distances, which significantly increases the labeling overhead for users.

2.2 Computer Vision

As we are leveraging image matching for appliance identification, we provide descriptions of some computer vision terms used across the paper. Then we describe topics in computer vision that are related to our work.

2.2.1 Terminology

Point Cloud: A set of 3D points. They can have extra information including RGB values. A point cloud can be visualized as a 3D world.

Image Registration or Image Matching: Computing the transformation (location and orientation) of an image as to another 3D coordinate system.

RGB-D Camera: A camera that captures both RGB images and their depth values at most pixels.

3D Model: A point cloud with useful extra information. In this paper, the extra information contains: (1) raw images and depth values that are used to construct the point cloud, along with their transformations, and (2) labels annotated on some points in the point cloud.

2.2.2 Computer Vision Topics

Object Recognition: Object recognition localizes and identifies objects in a 2D image or 3D point cloud. Ideally,

we could recognize the appliances in an image and interact with them. However, object recognition usually requires heavy training with a huge amount of human-labeled data [7]. Furthermore, state-of-the-art algorithms report a mean average precision of 53.1% for 2D images [10] and 57.6% for 3D point clouds [11]. These are not adequate for our purpose, let alone when there are many identical devices in a building such as monitors and lights.

Structure from Motion: A 3D model is important to localize an image to get context information for AR. Plenty of work has been done to construct a point cloud from images taken from different angles and locations, dubbed Structure from Motion (SfM). The primitive operation of SfM is to perform image registration of one image to another image. Image registration is a well-studied topic, such as the multi-view geometry approaches [12]. SfM based on only RGB images was used to build a 3D model of Rome [1]. However, this approach usually generates low-density models in indoor environments because they usually have low levels of illumination and are texture-less. Fortunately, thanks to recent commercial RGB-Depth cameras, such as the Microsoft Kinect and ASUS Xtion Live Pro, high density 3D models are feasible [13, 8].

Visibility Analysis: Visibility analysis computes whether a point is visible given the location, rotation, and view size of a camera. Alsadik et al. [2] summarizes two groups of state-of-the-art visibility analysis approaches. The first group builds surfaces in a point cloud and determine if the ray between the camera and the target point intersects with any surface. The second group consists of Hidden Point Removal (HPR) algorithms [17] and its extensions [27]. They calculate the convex hull of the point cloud's spherical flipping projection centered at the camera. The points on the convex hull are proved to be visible. However, both approaches involve iterating on all points, which is slow given the number of points in a point cloud.

3. SYSTEM DESIGN

One of our main contributions is analyzing the characteristics of different computer vision approaches for our particular purpose. In this section, we first describe a use case to show what is desired in the concept of "What you see is what you interact with." Based on the use case, we discuss the design requirements and details of how we decide to use each particular technique in our system.

3.1 Use Case

Imagine Alice is about to give a presentation in a meeting room. She wants to change several settings quickly in the room before her presentation: close the shutters, pull down the electronic projector screen, dim the lights near the screen, turn on the projector, and connect her laptop to the projector.

Today, she would walk to every window and manually close the shutters. Then she finds the control panel of the projector screen and finds the button to pull it down. Then she needs to walk to the light switches, and might have to try several switches to get to the correct one. Finally, she finds the remote controller of the projector and figures out how to turn it on and connect a laptop to it, either through

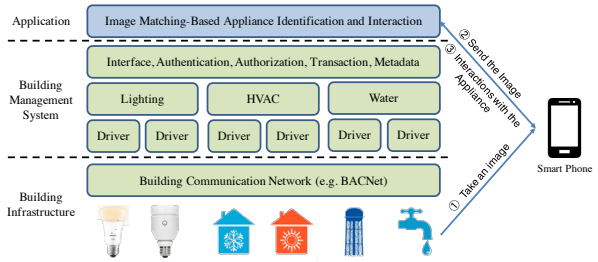


Figure 1: Using Building Management System.

a cable somewhere in the room or over local network at an IP address and port. All these efforts are tedious, and this happens frequently in daily life.

Instead, Alice wants to interact in a more intuitive way. She does not want to provide any description for the appliance that she intends to control, including the IP address, serial number, location, or voice dictation. Specifically, she wants to point her smart phone camera to the appliance, and instantly all controllable appliances are highlighted in the view. She taps on the appliance of her choice, and all its virtual UI components will be overlaid on the screen. Each type of appliance should have its particular UI components. For example, a light should have at least a toggle button and a brightness slider. Then she can naturally interact with the appliance and see the changes in the real world with imperceptible delay.

3.2 Using Building Management Systems

Even though we focus on intuitively identifying appliances, end-to-end human-object interactions also rely heavily on a good Building Management System (BMS) to handle all interactions. Based on the requirements, we expect the system to have a high-level architecture as shown in Figure 1. A mobile phone client captures an image of an appliance and transmits to our server, which runs as an application of the BMS. Our server finds the target appliance and sends its control information back. Then the user can interact with the appliance, and all traffic is relayed through our server to the BMS interface.

A BMS should be built on top of stable and fast building infrastructure, including physical appliance controllers and a communication network, such as BACNet. In the BMS, we need abstractions of different types of appliances, along with separate drivers for different models of hardware. To guarantee performance and safety, many other components are required, such as application authorization and authentication or interaction transaction management. As for our purpose, we also require the BMS to define all user interfaces that we can interpret and show on a smart phone. Such interfaces can be defined using specific interface deception languages [26]. Researchers have put great effort into building such a BMS [5, 6], and we expect to be able to take advantage of these in the near future.

3.3 System Requirements

The intuitive human-object interaction we propose requires minimum effort from users, but in turn imposes many design requirements on our system. Here we give a summary of

these requirements with explanations, then we talk about how we address each of them respectively in the remainder of this section.

- **A good visual model that is easy to build and manipulate.** A visual model of the building needs to be created for appliance identification. We require that (1) collecting data to construct the model is simple, such as taking a video of the building interior; (2) the visual model is accurate enough for visualization, and applying labels should be straightforward and effortless.
- **Fast, robust, and accurate RGB image matching onto the 3D model.** We require (1) fast matching because it is interactive; (2) robust matching from all image locations and orientations; (3) accurate matching, otherwise the visibility analysis will not be accurate; and (4) matching RGB image without depth values, because smart phone cameras do not provide depth information;
- **Fast, robust, and accurate visible point analysis.** The visible point analysis should also be fast, robust, and accurate for the same reasons as RGB image matching.

3.4 Object Recognition or Image Matching?

There are two ways to determine objects in an image: object recognition and image matching. Object recognition "understands" what objects are in an image based on models trained from human-labeled objects in images, such as a Convolutional Neural Network [22, 7]. Image matching searches for images similar to those that have already been labeled in terms of their local features.

We chose to use the image matching approach for two primary reasons. First, object recognition requires a large amount of manual labeling effort on the training data to account for the different forms of an object. For example, ImageNet [7] has more than 14 million images. In comparison, because an indoor environment has relatively limited and static space, we only need a limited number of images to capture local features of all appliances and their visual context, rather than "recognizing" the objects. Second, object recognition accuracy is not practical for our purpose. State-of-the-art algorithms can only achieve a mean average precision of 53.1% [10], whereas image matching can achieve more than 90% accuracy with 10-60% recall of all matches under different image transformations such as scale, rotation, blurring, illumination, and view point [28]. We argue that a small recall of matches is good enough for our purpose because we only need one correct match to register an image, even though it can be matched to many other images.

3.5 A List of Labeled Images or A 3D model?

Image matching based appliance identification involves two steps: (1) look for a similar image to the user-provided image among precollected images, and (2) determine which appliances are visible in the image. There are two ways to do the second step: (1) label all devices in all precollected images, which is used in [15], or (2) build a 3D model from all precollected images, and label on the point cloud.

In our system, we decided to create a 3D model for three reasons. First, because local feature-based image matching can only get up to 60% accuracy for different viewpoints and 80% accuracy for different scales (or distances) [19], we need many pictures of an appliance from different angles and distances to guarantee successful matches. However, this significantly increases the number of images we need t represent all appliances. For example, we use 1075 images to cover a 864 sq. ft. (80 sq. meters) lab in our evaluation. Therefore, labeling on every image is tedious and requires significant human effort. By contrast, as we will describe next, labeling an appliance in a 3D model involves simply clicking on a point in the model. Second, a 3D model can be used to create synthetic images using 3D-to-2D projection, which can be used to match user-provided images that are not similar to any precollected image. Third, 3D reconstruction is effortless and intuitive thanks to RGB-D cameras, because the depth information simplifies the matching algorithm, improves the accuracy, and produces a more dense point cloud. We can construct a point cloud by simply capturing a video with an RGB-D camera while walking through the building [34].

3.6 Labeling the Point Cloud

Object labeling is mostly used to create a ground truth database to train and evaluate computer vision models. For this purpose, all object labeling works require labeled object to be enclosed in a contour or a box in either 2D [32] or 3D [33] cases, so the model can capture all features of the objects. Because labeling is very strenuous work for a human, people usually use crowd-sourcing platforms, such as Amazon Mechanical Turk[2].

Conventional labeling does not meet our requirements, because we aim to make the system easy to use for the building manager. In addition, we need the visibility analysis to be fast, which is tightly coupled with how appliance labels are structured. Fortunately, instead of the visual features of an appliance, we only need its location to perform the visibility analysis later. Moreover, we assume that when a user wants to control an appliance, she will capture the main part of the appliance in the image intuitively. Therefore, we decided to only label a handful of center points (possibly as few as one point) on each appliance in the point cloud.

3.7 Visibility Analysis

After matching an image, we need to perform visibility analysis to determine which appliances are shown in it. There are two questions we need to answer: (1) Given the orientation and view angle of the image, which appliances are covered in the viewing cone? (2) Given the location of the image, which appliances in the view cone are occluded (hence the rest are visible)?

The first question can be solved by projecting all labeled 3D points onto the 2D image plane and determining which fall into the image pixel range. Because we choose to label each appliance with a small number of points, there could be only tens of points in a room. Furthermore, the 3D-to-2D projection algorithm has time complexity $O(n)$, where n is

[2] `https://www.mturk.com/mturk/welcome`

Figure 2: Modeling Phase Overview

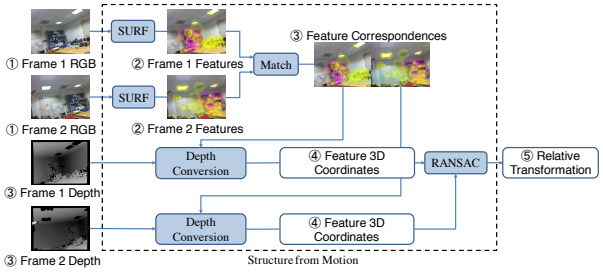


Figure 3: Image registration algorithm in the Structure from Motion (SfM) component.

the number of labeled points, so we expect this step to be very fast.

The second question is the conventional visibility analysis problem. As we described in Section 2.2.2, visibility analysis considers surfaces formed by points to determine occlusions, either by reconstructing the surfaces from points or projecting them spherically to calculate the convex hull [2]. However, all current approaches need to iterate all points in a point cloud, which can easily be several million for a room. Furthermore, these algorithms have higher complexity. For example, HPR has time complexity $O(n \log n)$ [17]. Therefore, we decided not to use conventional visibility analysis algorithms, and address the second question by listing multiple appliances sorted by distance from the camera.

4. SYSTEM ARCHITECTURE

Our system consists of two phases: (1) a modeling phase for the building manager to create a 3D model, and (2) a matching phase for users to identify and interact with appliances. In this section, we describe how each phase works.

4.1 Modeling

The modeling phase is designed for the building manager to construct a 3D model. Figure 2 shows the overview of the modeling process. The building manager uses an RGB-D camera to take a video walking through the building, and the Structure from Motion module outputs a point cloud with all registered images. Then the building manager labels all appliances to generate the 3D model that we need in the matching phase.

4.1.1 Structure from Motion

Our Structure from Motion (SfM) component converts an RGB-D video to a point cloud. It runs on a laptop connected to an RGB-D camera. To perform the reconstruction, the building manager carries them and takes a video walking through the building.

Given an RGB-D video, SfM assumes every two consecutive frames have a small difference in location and orientation, meaning they have enough common local features to be registered. This image registration step is the primitive of SfM. Figure 3 shows the details of the image registration algorithm with example pictures generated using RTAB-Map [20]. When doing image registration, SfM first computes all SURF features on both frames. For each feature in one image, SfM searches for the closest feature in the other image, therefore generating a list of pairs of matched points between the two images. As implemented in RTAB-Map, two features are matched using the Nearest Neighbor Distance Ratio (NNDR) approach. Particularly, feature $\mathcal{A}$ in image 1 matches feature $\mathcal{B}$ in image 2 if and only if

$$\text{distance}(\mathcal{A}, \mathcal{B}) < \phi \cdot \text{distance}(\mathcal{B}, \mathcal{C}) \quad (1)$$

for any feature $\mathcal{C}$ in image 2, where ϕ is a predefined ratio. A pair of matched points are supposed to be the same physical location in a 3D world, but could be at different positions on the two images. With the list of matched points, SfM computes their corresponding 3D point coordinates in their own image coordinate system based on the depth values. SfM then uses the RAndom SAmple Consensus (RANSAC) algorithm [9] to calculate the relative transformation between the two frames. After all consecutive frame pairs are registered, their 3D points will be transformed and added to a global point cloud.

RTAB-Map [20] fits our design for SfM well, so we decided to use it as our SfM implementation. After the point cloud of a building is generated, we save it in a PLY (Polygon File Format)[3] file. Along with the point cloud, all images and their locations and orientations are automatically saved in an RTAB-Map-defined SQLite database.

4.1.2 Labeling

We employ two approaches to label appliances. The first is CloudCompare[4], which is an open source project for point cloud manipulations. The building manager can inspect the point cloud and select a point. The coordinates of a list of selected points can be saved as a text file. We use one text file for every appliance and use the appliance name as the file name. Because CloudCompare implements the hidden point removal algorithm [17], we can directly label points on point clouds without constructing surfaces, which would be computationally intensive.

We have also built our own 2D-to-3D labeling tool to label 3D point from 2D images that are registered on the point cloud. Our tool allows the building manager to specify a 2D coordinate on an image for the calculation. It first calculates the 3D coordinate of the point in the camera's coordinate system using the depth values of the image saved in the 3D model. Then we convert it to a global 3D coordinate using the transformation of the image.

After the list of label files is created, we have a complete 3D model. In summary, the 3D model contains: (1) A point cloud PLY file, (2) an RTAB-Map-defined database containing registered images and their parameters, and (3) a list of label files containing the coordinates of labeled points.

4.2 Matching

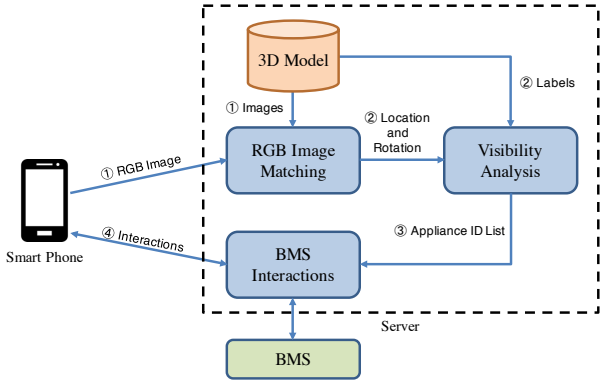


Figure 4: Matching Phase Overview

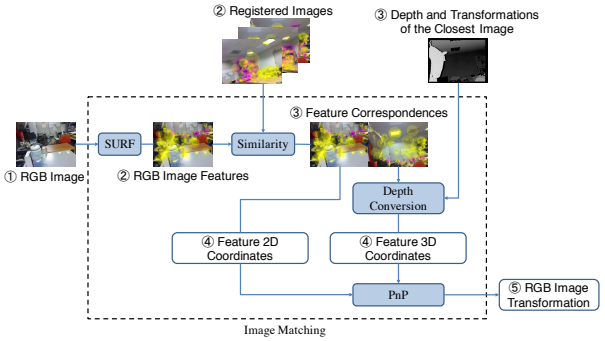


Figure 5: RGB Image Matching Algorithm

After the 3D model for a building is created, we put it on our server for the matching phase. Figure 4 gives an overview of the matching phase. A user takes an RGB image of the appliance she wants to interact with using our mobile application, which is immediately transmitted to our server. Then the server computes the location and rotation of the RGB image by matching it to the most similar image in the 3D model. It then performs visibility analysis for all labeled appliances to check whether they show in the RGB image. After the appliances are determined, our server talks to the BMS for interactions as described in Section 3.2.

4.2.1 *RGB Image Matching*

The image matching component takes a new RGB image from a smart phone and registers it onto the point cloud in the 3D model. Although image matching is already implemented in the SfM component, we cannot simply reuse it to match new images for two reasons. First, an image has to be registered with another image that shares enough common SURF features. However, the new image is from an arbitrary location and orientation, so we do not know which image it should be registered with. Second, the RGB image does not have depth information, so we cannot use the RANSAC approach to compute the relative transformation. We therefore adopted another approach to perform the RGB image matching.

Figure 5 shows how our RGB image matching works. We first calculate SURF features for the new RGB image. Then we find the closest image that has the highest similarity with the RGB image. We adopted the similarity definition used by RTAB-Map, which is:

$$\text{Similarity} = \frac{\text{Number of matched feature pairs}}{\text{Max number of features of two images}} \quad (2)$$

With the depth values of the closest image and its registration information, we calculate the global 3D coordinates of its matched features. Then the image matching problem becomes a Perspective-N-Point (PnP) problem [30]. A PnP problem is the determination of image location and orientation given the 3D coordinate of a point and the 2D coordinate of its corresponding point in an image. For noisy data, RANSAC can be used to solve PnP problem, which is what we adopted.

Many pieces are implemented in RTAB-Map, such as a wrapper of the PnP solver in OpenCV. We extended RTAB-Map to perform our RGB image matching.

4.2.2 *Visibility Analysis*

After we obtain the camera location and orientation, we need to perform visibility analysis to determine which appliances are visible in the RGB image. As we discussed in Section 3.7, we only check whether the labeled 3D points are visible in the image. We first check which points are behind the camera plane. The camera plane is an infinite 2D plane defined in the pinhole camera model [12], and all points with negative Z axis values are behind the plane. By transforming the points to the camera's 3D coordinate system, we remove all points with negative Z axis values. Then all the labeled points in front of the camera are projected onto the camera plane. We then pick the 2D pixels on the image plane that are within range of the image.

4.2.3 *Appliance Interactions*

When we have determined the appliance that a user intends to control, we talk to the building management system for specifications of UI and interactions. Because we assume a well designed BMS, this should be straightforward. We leave this to future work.

5. EVALUATION

To investigate the usefulness and effectiveness of our system, we evaluate it on the RTAB-Map multi-session dataset [21] and a deployment in our lab.

5.1 Experiment Setup

5.1.1 *RTAB-Map Multi-Session Dataset*

The RTAB-Map multi-session dataset was collected by a four-wheeled robot attached to a Microsoft Kinect [21], which produces 640 × 480 RGB images and corresponding depth values on most pixels. It contains five videos of a building floor, with a total path length of 750 meters. The center part of the floor is captured by all videos to perform global loop closure detection. We picked one of the videos to construct a point cloud with 304 images registered and labeled 10 objects in the 3D model, which are shown as Y axis labels in Figure 6a. We only label 1-3 points on each object. Although these objects are not smart now, we expect all of

[3] `http://paulbourke.net/dataformats/ply/`
[4] `http://www.danielgm.net/cc/`

them to be connected to the Internet in the foreseeable future. We manually subsampled 46 RGB images taken from the common center area from the other 4 videos that contain labeled objects. Because the 5 videos are taken at different times, the images are always from different angles and distances.

5.1.2 Lab Deployment

We deployed our system in our lab in Soda Hall in UC Berkeley. The size of the room is approximately 864 sq. ft. (80 sq. meters). We hand-hold a Microsoft Kinect to capture and reconstruct the interior of the lab in approximately 18 minutes. Our video was captured at 1 Hz and contains 1075 RGB-D images. We labeled 15 appliances in the lab with 1 point on each. The 15 appliances are shown as Y axis labels in Figure 6b.

To take into account the daily changes, we did the matching experiment dour days after the construction of the 3D model. For each of the 15 labeled appliances, we took 5-6 pictures of each appliance from different angles using an LG G2 Mini Android phone, and got 76 pictures with 2560×1920 pixels in total.

Compared to the RTAB-Map multi-session deployment, the lab deployment is more challenging for several reasons. First, the robot has fewer degrees of freedom in its motion than a human. It captures all images from a fixed height above the floor. This makes it easy to match images taken from a similar height and angle, but humans are likely to take images from a variety of heights and angles. Also, robots move more smoothly than human, which leads to fewer blurry images.

5.2 Image Matching

We regard an image to be matched (or registered) if it finds an image from the 3D model that is most similar to it, and they have enough common features to compute a relative transformation. In addition, as adopted in RTAB-Map in SfM, if the transformation between these two images, as computed by the PnP algorithm, has a rotation exceeding 90°, we regard the registration as a failure. 45 out of 46 (98%) images in the RTAB-Map multi-session dataset are matched. In our laboratory deployment, 54 out of 76 (71%) images are matched. This is reasonable given the degrees of freedom of human motion and changes in the environment in the four days.

5.3 Appliance Identification

Because it is hard to obtain the ground truth location and orientation of images, we study how well our system identifies objects, which in turn relies on image registration. Every appliance in every test RGB image, which we call an instance, is marked as one of four types based on human observation:

1. Not Visible: None of the object lies in the viewing cone of the image.
2. Occluded: The object lies in the viewing cone of the image but is occluded by other objects.
3. Partially Visible: Part of the object can be seen in the image.
4. Visible: The whole object can be seen in the image.

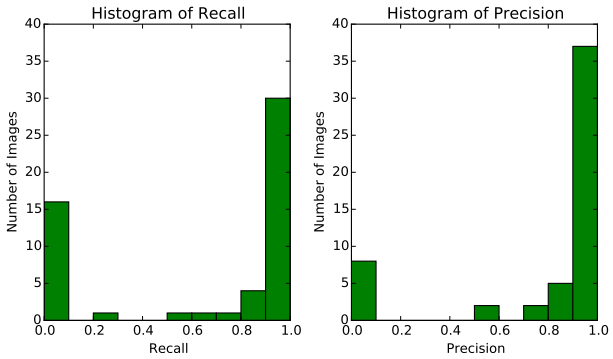


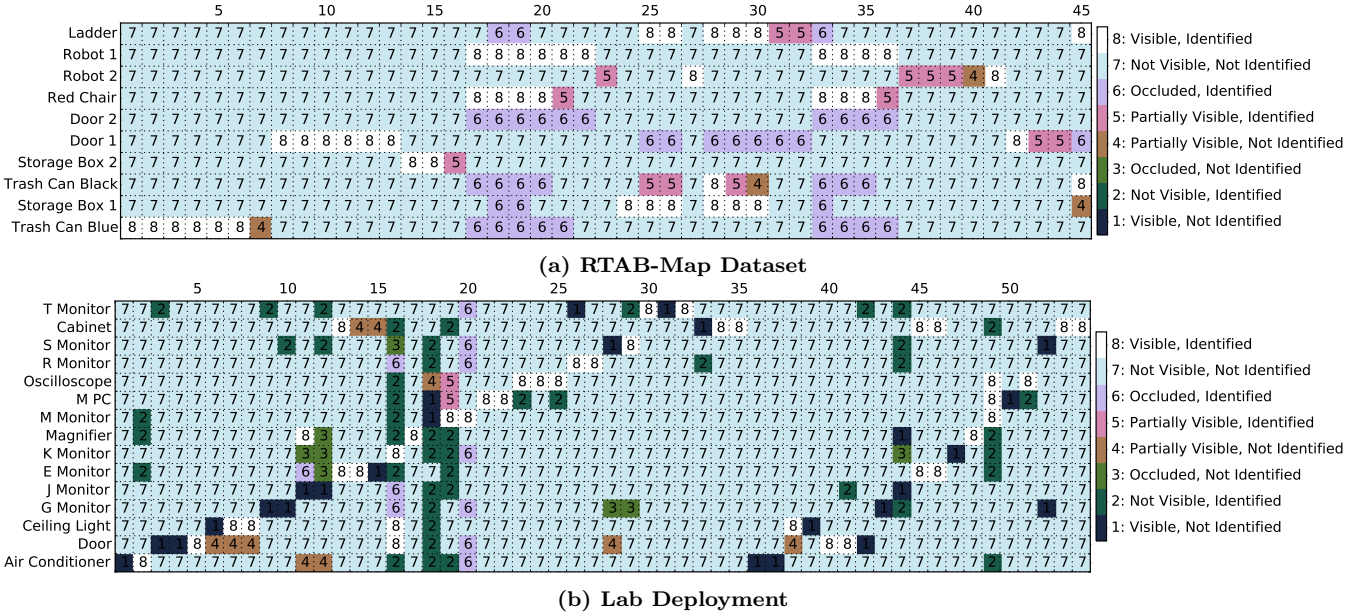
Figure 7: Lab deployment recalls *(left)* and precisions *(right)*

Our system reports whether an appliance is identified for every appliance and every successfully matched image. Therefore, every instance falls into one of 8 categories given both its ground truth and its identification result. The 8 categories are shown in the result Figure 6.

Figure 6a shows the results of the 45 matched images in the RTAB-Map multi-session dataset, along with the 10 labels and 8 categories. There are in total 450 instances, and 392 (87.1%) of them are type 7 and type 8, which means our system reports correct results. Another 40 (8.9%) instances are type 6, which means the appliances are occluded but located in the viewing cone of the image. Because we do not perform any occlusion analysis, which is computationally intensive, these results are expected. The remaining 18 (4.0%) instances, which are type 4 or type 5, are partially visible in some images and are not always identified. Because we only label central points of an appliance, if a labeled point falls out of the image viewing cone, our system does not identify the appliance. This will not impact our user experience because users will intuitively try to put the entire target appliance in the image. In our system, we regard all instances in categories 4-8 to be successful identifications. Therefore, we achieved 100% success on the RTAB-Map multi-session dataset.

The lab deployment results are shown in Figure 6b. Among all 810 instances across 54 images and 15 appliances, 707 (87.3%) are type 7 and 8, 11 (1.4%) are type 6, and 12 (1.5%) are type 4 and 5. Given the noise arising from human video capture and environmental changes, there are more erroneous instances than in the RTAB-Map multi-session dataset experiment. To further evaluate how the system performs, we define the recall of each image as the number of appliances correctly identified by the system divided by the total number of appliances actually appearing in the image. We also define the precision of each image as the number of appliances correctly identified by the system divided by the total number of appliances identified by the system.

Figure 7 shows the histogram of recall and precision of all 54 images. We can see that both the recall and precision distributions are bimodal, which means some images give completely wrong results, whereas the rest perform fairly



(a) RTAB-Map Dataset

(b) Lab Deployment

Figure 6: Image matching results. Every column is a matched image, and every row is an appliance.

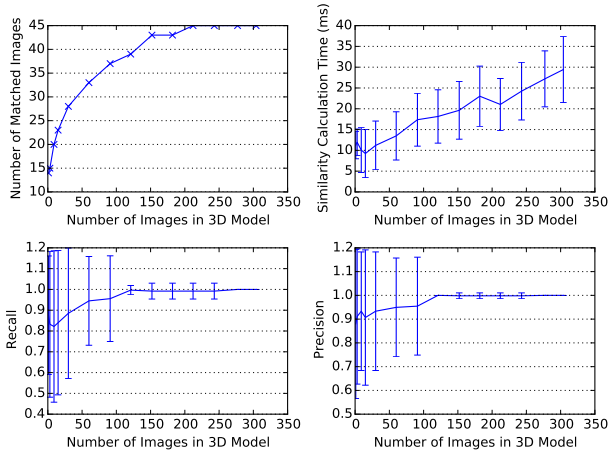


Figure 8: Number of matched images *(top left)*, similarity calculation time *(top right)*, recall *(bottom left)*, and precision *(bottom right)* with different numbers of images in the 3D model

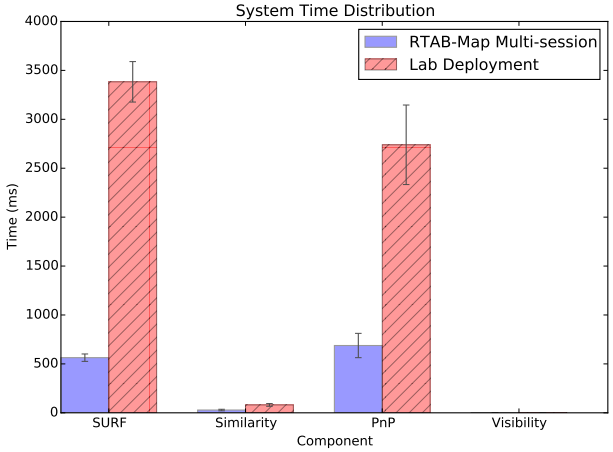


Figure 9: Time of system components in matching phase for RTAB-Map multi-session dataset and lab deployment

well. Among the 54 images, 34 (63%) have recall larger than 0.8, and 42 (78%) have precision larger than 0.8.

5.4 Amount of Images in 3D Model

We studied the number of images required to cover a certain space, such that enough RGB images can still be matched and the recall and precision remain stable and high. We chose to use the RTAB-Map multi-session dataset because it gives perfect results with 304 images in the 3D model. We uniformly and randomly subsampled the dataset with predefined proportions and reran the experiments. Figure 8 contains four subplots that show how the following values change as the number of images in the database increases: (1) the number of matched images, (2) the similarity calculation time, (3) recall, and (4) precision. As we can see, the number of images that we can locate saturates at around 200 images in the 3D model. Also, because we iterate through all images in the 3D model to find the most similar one, the matching time increases linearly as expected. Both recall and precision reach 100% with around 120 images in the 3D model. Thus, we conclude that for our RTAB-Map multi-session dataset experiment, we can achieve the same performance with only 200 images instead of 304. This also implies that we do not have to capture as many images as possible when building the 3D model. A more interesting question is what images we should keep in the 3D model, and we leave this to future work.

5.5 Execution Time Analysis

The time we spend on processing every RGB image is the key to our goal of achieving low latency. Although the image

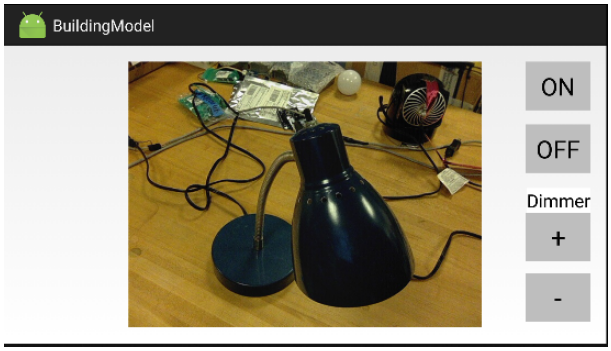


Figure 10: Android Application User Interface

transmission from smart phone to the server also contributes to latency, it depends largely on the network environment, which we do not get to control. We break down the time of the matching phase on our server based on the four main components: (1) SURF, (2) similarity calculation, (3) PnP, and (4) visibility analysis.

Figure 9 shows the time spent on each component for both experiments. It is interesting to see that SURF and PnP took most of the time. As we use 2560×1920 pixel images in the lab office deployment, its SURF and PnP calculations are several times longer in duration than the processing of the 640×480 Kinect images in the RTAB-Map multi-session dataset. This means that we should subsample the image before sending to the server.

5.6 Android Application

We built an Android application as part of our end-to-end system. Figure 10 shows an example of the user interface in the application. As we can see from the perspective of a user, the smart phone camera is pointing at a smart lamp. Our system matched the image and determined the appliance in the view of the camera, so it talks to the BMS to get a list of UI elements to show on the right side of the screen. In this case, The lamp has ON/OFF and dimmer buttons.

6. DISCUSSION

Bringing computer vision to practical smart building applications involves many challenges. With our system and evaluation, we identify several issues for future work. First, our system assumes the visual environment does not change over time, which is not true for mobile appliances. Users could submit images in a crowdsourcing strategy to update the 3D model as the building environment changes. This would make the system more robust in the case of dynamic environments and mobile appliances. Second, as we observed in the lab deployment, some images cannot be registered to the 3D model because there is no similar image. On the other hand, there are many "duplicated" images that are very similar to each other in the 3D model. It is useful to learn which images should be saved in the 3D model to provide a higher image matching rate. Third, as the majority of the image matching time is spent on SURF and PnP computations, we can further reduce the end-to-end delay by subsampling the image before transmitting to the server.

7. CONCLUSION

In this paper, we presented an intuitive appliance identification system using image matching. We break down the problem to several tractable components and present a comprehensive analysis on design choices of computer vision techniques. We tested our system on a public data set and deployed our system in our lab. The results demonstrate the viability of the system, and also raise interesting questions and challenges on incorporating computer vision technologies to augment smart buildings.

Acknowledgements

This work is supported in part by the National Science Foundation under grant CPS-1239552 (SDB).

8. REFERENCES

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. M. Seitz, and R. Szeliski. Building rome in a day. *Communications of the ACM*, 54(10):105–112, 2011.

[2] B. Alsadik, M. Gerke, and G. Vosselman. Visibility analysis of point cloud in close range photogrammetry. *Int. Ann. Photogramm. Remote Sens*, pages 9–16, 2014.

[3] H. Bay, T. Tuytelaars, and L. Van Gool. Surf: Speeded up robust features. In *Computer vision–ECCV 2006*, pages 404–417. Springer, 2006.

[4] A. Bulling, D. Roggen, G. Tröster, G. Tröster, and G. Tröster. *Wearable EOG goggles: Seamless sensing and context-awareness in everyday environments.* ETH, Eidgenössische Technische Hochschule Zürich, Wearable Computing Laboratory, 2009.

[5] S. Dawson-Haggerty, X. Jiang, G. Tolle, J. Ortiz, and D. Culler. smap: a simple measurement and actuation profile for physical information. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, pages 197–210. ACM, 2010.

[6] S. Dawson-Haggerty, A. Krioukov, J. Taneja, S. Karandikar, G. Fierro, N. Kitaev, and D. E. Culler. Boss: Building operating system services. In *NSDI*, volume 13, pages 443–458, 2013.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[8] H. Du, P. Henry, X. Ren, M. Cheng, D. B. Goldman, S. M. Seitz, and D. Fox. Interactive 3d modeling of indoor environments with a consumer depth camera. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 75–84. ACM, 2011.

[9] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 580–587. IEEE, 2014.

[11] S. Gupta, P. Arbeláez, R. Girshick, and J. Malik. Aligning 3d models to rgb-d images of cluttered

scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4731–4740, 2015.

[12] R. Hartley and A. Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[13] P. Henry, M. Krainin, E. Herbst, X. Ren, and D. Fox. Rgb-d mapping: Using depth cameras for dense 3d modeling of indoor environments. In *In the 12th International Symposium on Experimental Robotics (ISER*. Citeseer, 2010.

[14] V. Heun, S. Kasahara, and P. Maes. Smarter objects: using ar technology to program physical objects and their interactions. In *CHI'13 Extended Abstracts on Human Factors in Computing Systems*, pages 961–966. ACM, 2013.

[15] P. Jain, J. Manweiler, and R. Roy Choudhury. Overlay: Practical mobile augmented reality. In *Proceedings of the 13th Annual International Conference on Mobile Systems, Applications, and Services*, pages 331–344. ACM, 2015.

[16] H. Kato and K. Tan. 2d barcodes for mobile phones. In *Mobile Technology, Applications and Systems, 2005 2nd International Conference on*, pages 8–pp. IEEE, 2005.

[17] S. Katz, A. Tal, and R. Basri. Direct visibility of point sets. *ACM Transactions on Graphics (TOG)*, 26(3):24, 2007.

[18] C. C. Kemp, C. D. Anderson, H. Nguyen, A. J. Trevor, and Z. Xu. A point-and-click interface for the real world: laser designation of objects for mobile manipulation. In *Human-Robot Interaction (HRI), 2008 3rd ACM/IEEE International Conference on*, pages 241–248. IEEE, 2008.

[19] N. Y. Khan, B. McCane, and G. Wyvill. Sift and surf performance evaluation against various image deformations on benchmark dataset. In *Digital Image Computing Techniques and Applications (DICTA), 2011 International Conference on*, pages 501–506. IEEE, 2011.

[20] M. Labbé and F. Michaud. Memory management for real-time appearance-based loop closure detection. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 1271–1276. IEEE, 2011.

[21] M. Labbe and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 2661–2666. IEEE, 2014.

[22] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[23] J. Lifton, M. Mittal, M. Lapinski, and J. A. Paradiso. Tricorder: A mobile sensor network browser. In *Proceedings of the ACM CHI 2007 Conference-Mobile Spatial Interaction Workshop*, 2007.

[24] D. Lymberopoulos, J. Liu, X. Yang, R. R. Choudhury, V. Handziski, and S. Sen. 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*, pages 178–189. ACM, 2015.

[25] S. Mayer, M. Schalch, M. George, and G. Sörös. Device recognition for intuitive interaction with the web of things. In *Proceedings of the 2013 ACM conference on Pervasive and ubiquitous computing adjunct publication*, pages 239–242. ACM, 2013.

[26] S. Mayer, A. Tschofen, A. K. Dey, and F. Mattern. User interfaces for smart things–a generative approach with semantic interaction descriptions. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(2):12, 2014.

[27] R. Mehra, P. Tripathi, A. Sheffer, and N. J. Mitra. Visibility of noisy point cloud data. *Computers & Graphics*, 34(3):219–230, 2010.

[28] K. Mikolajczyk and C. Schmid. A performance evaluation of local descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(10):1615–1630, 2005.

[29] S. N. Patel and G. D. Abowd. A 2-way laser-assisted selection scheme for handhelds in a physical environment. In *UbiComp 2003: Ubiquitous Computing*, pages 200–207. Springer, 2003.

[30] L. Quan and Z. Lan. Linear n-point camera pose determination. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 21(8):774–780, 1999.

[31] J. Rekimoto and Y. Ayatsuka. Cybercode: designing augmented reality environments with visual tags. In *Proceedings of DARE 2000 on Designing augmented reality environments*, pages 1–10. ACM, 2000.

[32] B. C. Russell, A. Torralba, K. P. Murphy, and W. T. Freeman. Labelme: a database and web-based tool for image annotation. *International journal of computer vision*, 77(1-3):157–173, 2008.

[33] S. Song, S. P. Lichtenberg, and J. Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 567–576, 2015.

[34] J. Xiao, A. Owens, and A. Torralba. Sun3d: A database of big spaces reconstructed using sfm and object labels. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 1625–1632. IEEE, 2013.

[35] B. Zhang, Y.-H. Chen, C. Tuna, A. Dave, Y. Li, E. Lee, and B. Hartmann. Hobs: head orientation-based selection in physical spaces. In *Proceedings of the 2nd ACM symposium on Spatial user interaction*, pages 17–25. ACM, 2014.