Noname manuscript No.
(will be inserted by the editor)

A Location-Sensitive Visual Interface on the Palm: Interacting with Common Objects in an Augmented Space

Seokhwan Kim · Shin Takahashi · Jiro Tanaka

Received: date / Accepted: date

Abstract We have created a visual interface using the human palm that is location-sensitive and always available. To accomplish this, we constructed an augmented space in an actual workspace by installing several depth cameras. To manage and connect the multiple depth cameras, we constructed a distributed system based on scalable client-server architecture. By merging depth images from different cameras, the distributed system can track the locations of users within their area of coverage. The system also has a convenient feature that allows users to collect the locations of objects while visualizing the objects via images from the depth cameras. Consequently, the locations of both users and objects are available to the system, thus providing a location-based context for determining which user is close to which object. As a result, the visual interface on the palm becomes location-sensitive, which could lead to various applications in daily life. In this paper, we describe the implementation of the aforementioned system, and demonstrate its potential applicability.

Keywords Location-Awareness · Interface on Body · Augmented Reality · Ubiquitous · System · Interaction

S. Kim
Department of Computer Science
University of Tsukuba
Tsukuba, Ibaraki, 305-0006, Japan
E-mail: skim@iplab.cs.tsukuba.ac.jp

S. Takahashi
E-mail: shin@cs.tsukuba.ac.jp

J. Tanaka
E-mail: jiro@cs.tsukuba.ac.jp

1 Introduction

The advancement of sensing and display technologies has made it possible to construct augmented spaces. Several systems have already been developed for augmenting all objects in a space, and enabling users to interact with those objects. Such systems include Microsofts EasyLiving [1], the Everywhere Display Projector [18], and LightSpace [22]. This study shares the concept of augmented space in such studies: it is assumed that all objects in an augmented space are interactive, and users can potentially interact with all of them.

LightSpace, in particular, enables interactions between augmented objects and a visual interface on the body [22]. Users are able to store digital content on their bodies and drop the content onto other physical objects (cf. the pick-and-drop metaphor of Rekimoto's early work [19]).

The present study employs a similar interactive concept (i.e., combining an interface on the body with augmented objects), but applies it to an actual workspace. Our space was constructed by installing depth cameras in a laboratory that is used by students. We installed several depth cameras to track users in the space. Accordingly, we implemented a distributed system that can seamlessly connect the depth cameras and merge their images, and thereby track the locations of users in the space. The installation of the depth cameras and the implementation of the distributed system are described in subsequent sections.

A key feature of our system is that it facilitates the process of registering or deleting the locations of objects. Because the system can track user locations, it provides a location-based context for determining which user is close to which object. A necessary preliminary step to enable this location-awareness is the collection

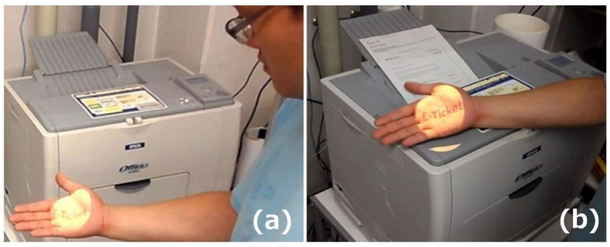


Fig. 1 An application of smart notification: When the user is close to the printer, a message is displayed on his palm regarding a scheduled printing task (a). The user can complete the printing task by dropping the message onto the printer (b).

(i.e., registration or deletion) of object locations, which generally requires considerable time and effort. To facilitate this process, we developed a support tool that enables such registration or deletion while visualizing objects via the images from the depth cameras. Thus, the system is able to determine the distance between users and registered objects, which establishes the location-based context.

Another key feature is that it provides a visual interface on the human palm. We installed two projected displays in the space. These devices display visual content on the users palm, and the user can interact with this interface via hand gestures or physical metaphors. Because the projected displays are installed in the space, the user is provided with a simple graphical user interface (GUI) on his/her palm, without temporal or spatial limitations (i.e., it is always available). Hence, we have created a visual interface on the palm that is location-sensitive and always available, and could lead to various applications in daily life.

Figure 1 shows an example. Because the system records the locations of objects and users (i.e., user skeletons), it can continuously track the distance between a user body part (e.g., a hand or a head) and an object. Employing this location-based context (i.e., the distance), the system can recognize which user is close to which device. Therefore, the system can intelligently display an appropriate message on a user's palm.

For example, if a user needs to print a document, the system displays a notification of the printing task on the user's palm when the user is close to a printer (i.e., the physical distance is sufficiently small). The user can then complete the printing task by dropping the notification onto the printer. This application allows users to efficiently carry out a task at a physical location. The notification is displayed when the necessary device (the printer in this example) is close enough, and the user can complete the task without carrying any additional devices (i.e., the notification can be displayed on a bare hand).

Our system includes the following features, which are described in detail below: 1) A distributed system to manage the depth cameras and track the locations of users, 2) a support tool to help collect object locations, and 3) projected displays to create a visual interface on the palm.

2 Related Work

The contributions of this study can be divided into three related domains: 1) constructing an augmented space, 2) registration of object locations, and 3) creating interfaces on the body. Below, we summarize each of these and clarify the motivations of this study.

2.1 Constructing an Augmented Space

The augmented space in this study allows interactions among all objects in the space. Here, we summarize some typical techniques for constructing an augmented space.

Computer vision is a popular approach for constructing an augmented space. Computer vision can track or identify both humans and objects. Microsoft's EasyLiving project [1] and the work of Lee *et al.* [12] are based on this approach. Studies from that era (i.e., about a decade ago) mainly relied on RGB cameras, which were adequate for envisioning futuristic environments in the early stages. However, because RGB image processing is sensitive to different lighting conditions, these techniques may present some practical drawbacks.

Another approach is to use highly capable sensors. The Gator Tech Smart House enables augmentation with various sensors, such as ultrasonic trackers [8]. Such sensors can provide high fidelity, but cost-effectiveness is an issue, because they are still expensive and not readily available.

An approach that has recently attracted attention involves the use of depth cameras. Depth cameras can provide features similar to those of RGB image processing (i.e., they can gather a lot of content), and are much more robust under different lighting conditions. Also, they are now affordably priced. LightSpace was the first system in which an augmented space was constructed using depth cameras [22]. The authors augmented all common surfaces (originally non-interactive) in their experimental space, and they successfully demonstrated the plausibility of using depth cameras via promising applications and interactions.

Our environment is implemented using an approach similar to that of LightSpace (i.e., depth cameras) [22]. However, we installed cameras in an actual workspace, whereas the LightSpace was based on an experimental space. Our real-world approach requires additional investigation. For example, it creates more concerns regarding the installation and synchronization of the cameras. (The details of the implementation are presented in Section 4.) Augmented spaces based on actual workspaces may also be necessary for specific research topics. For example, our system can be used to examine the effect of a users familiarity with a space [10], which is difficult to accomplish in an artificially constructed experimental space.

2.2 Registration of Object Locations

Location-based context means that the system is able to track the locations of users or objects to measure the distances between them. Consequently, the system can determine which users are most likely to manipulate which objects. This is the exact meaning of location-based context in this study. Location-based context can provide many usable features [16], and the registration of object locations is a necessary preliminary step to accomplishing this.

Generally speaking, there are two approaches to registration: automated and manual. The Gator Tech Smart House employs various sensors to track object locations automatically [8][11]. Devices store their profiles and register themselves automatically, and then the sensors are able to track the locations of the devices. We agree that this is an elegant approach, but the technology is not yet fully mature, and the cost remains relatively high.

Mobile augmented reality is a manual approach. In CAMAR [20] and iCam [17], the space is able to track the pose (i.e., location and heading) of a mobile device. With this pose data, it is possible to determine the location of an object using the mobile screen (users can collect the locations from the mobile screen). However, one drawback of this approach is that it can be time consuming, because the users must be close to the objects, which necessitates physical movement.

The Gator Tech Smart House demonstrated another promising technique [8]. They constructed a sophisticated virtual model that allows users to collect locations while navigating the virtual space via a simple GUI application. This technique is actually cheaper and more efficient than the aforementioned approaches (automatic registration and mobile AR), but considerable time and labor (e.g., 3D modeling) are involved in constructing the virtual model.

As noted above, we constructed an augmented environment using multiple depth cameras. By combining images from the depth cameras, it is possible to create a point cloud visualization, which can function as a virtual model without additional cost. Then users can collect the locations of objects from this point cloud visualization. The details are presented in Section 5.1.

2.3 Creating Interfaces on the Body

There are two different methods for creating interfaces on the body: wearable devices and the infrastructure-dependent approach. In this section, we describe both methods, and explain their benefits and shortcomings.

2.3.1 Wearable Devices

Skinput, by Harrison *et al.*, attaches analog sound sensors to a user's elbows [7]. When a point on the skin is touched, some vibration occurs, and this vibration is propagated along the skin. The sound sensors are able to detect such vibrations, and recent machine learning techniques allow the point of contact to be recognized. Omnitouch provides similar features, but relies on different sensors (i.e., depth cameras) [6]. The developers of this system tested a shoulder-worn device that included a micro projector and a depth camera. The micro projector displayed some images on a users palm, and the depth camera was able to track the movement of fingertips on the palm.

2.3.2 Infrastructure-Dependent Approach

Another technique for creating a visual interface on the body is to exploit the infrastructure of the space. The Palm Display of Kim *et al.* exploits a projector and camera installed in the space [9]. The projector displays some images on a users palm, and the user is able to interact with these images via fingertip gestures. LightSpace has a more sophisticated setup [22]. Multiple depth cameras and projectors are used so that every surface in the space becomes a touch-sensitive display, and a user is able to interact with them using the interface on his/her body.

The major advantage of this infrastructure dependent approach is obvious. It does not require the user to wear any devices. However, it is applicable only to indoor scenarios. On the other hand, wearable devices are applicable to both indoor and outdoor scenarios, but users must wear more than one device.

This study is concerned with interactions in an augmented space, which is an indoor scenario. Accordingly,

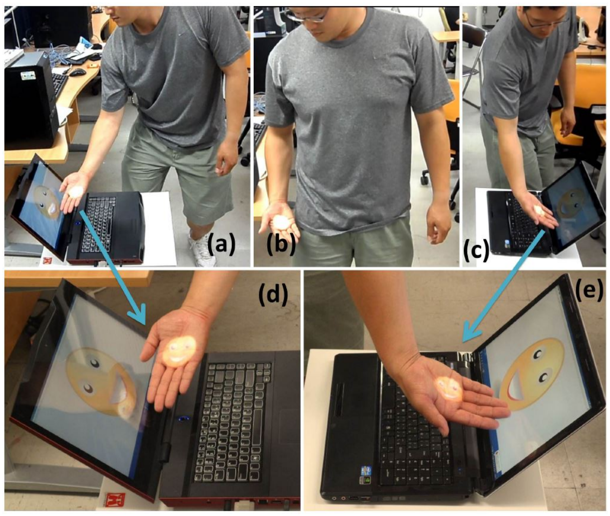


Fig. 2 An example of data stored on the body. The user stores a picture on his hand by touching the screen of a laptop (a, d), stores the picture on his pocket (b), and drops the picture onto another laptop (c, e).

we developed a visual interface using the infrastructure-dependent approach.

3 Applications

As previously noted, we have constructed a space that can record the locations of users and objects. Using this location-based context, the system can determine which user is manipulating which object. The space also creates a visual interface on the palm. In this section, we present some applications. The details of the system are discussed in Section 4.

3.1 Data Storage on the Body

The system can track users' bodies and the locations of devices. Thus, it can sense a touch event between a body part and a device by measuring the distance between them. A possible application arising from this capability is data storage on the body. Figure 2 illustrates this. A user touches a digital picture on a laptop display, and the picture is moved to his hand (Fig. 2a and 2d). Then he touches his right pocket (Fig. 2b), and the system stores the picture on his right pocket. Subsequently, the user again touches the right pocket, and the picture reappears on his hand (i.e., the user extracts the picture from the pocket). Then the user can drop the picture onto another computer (Fig. 2c and 2e).

When transferring data from one machine to another, temporary storage (e.g., flash drive) is sometimes necessary, and this can be rather cumbersome, because the device must be connected to and disconnected from each machine. The present application could be useful in such situations, allowing users to temporarily store data on their bodies and easily transfer the data to another machine.

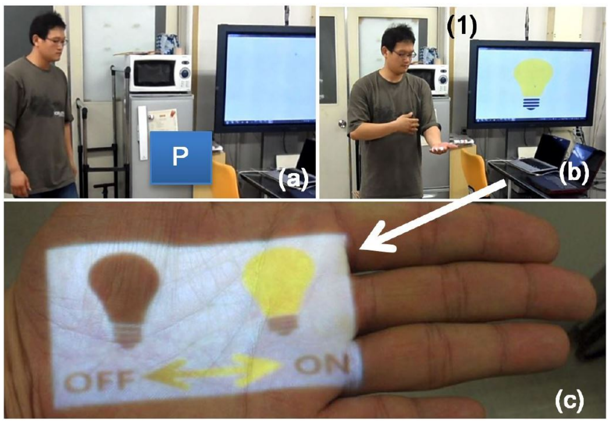


Fig. 3 Imaginary input in the air. Users can secure a location in the air for an interface. For example, the user swings his arm around at (P) in (a). Then, it shows an imaginary interface (c) for a light emulator in a large display, i.e., (1) in (b).

3.2 Virtual Midair Input

In the above example, the system recognized a touch event between a body part and a device by measuring the distance between them. Utilizing the same principle, the system can also recognize a touch event between a midair location and a body part. For example, if a location is associated with a specific interface, and a users hand is close to that location, the system can automatically display the interface on the hand.

Figure 3 illustrates this. The location indicated by the blue rectangle P in Figure 3a is reserved as the location of an imaginary light switch. The user memorizes this location, and when he wants to control the light, he simply places his hand near the reserved location, and the switch automatically appears, as shown in Figure 3b and c. The application displays a bidirectional arrow, the two directions corresponding to the on and off commands. The user can control the light by making the appropriate directional hand gestures.

This application implies that we can exploit any location in a space as a meaningful location associated with a specific interface. Generally speaking, controllers for electrical devices (e.g., lights or air conditioners) are placed at certain locations, and users must move to those locations. When this type of virtual input is

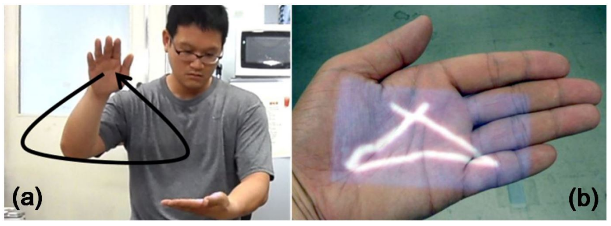


Fig. 4 Visual feedback viewer for a hand gesture. When the user makes a gesture (a), the shape of the gesture is traced on the other hand (b).

available, users can reserve a convenient location, and can access the controllers with less physical movement.

3.3 Complementary Interface

Because most interaction schemes are less than perfect on their own, additional modalities generally help to improve their capabilities. In this sense, a visual interface on the palm can be used to provide a visual modality to other interfaces. To illustrate this point, we present a visual feedback viewer for hand gestures.

An advantage of hand gestures is that they can be made without temporal or spatial constraints. Sometimes there may be no means of providing feedback (i.e., devices without visual or auditory components). In this case, an interface on the palm can serve as a visual feedback viewer, as shown in Figure 4. When a user makes a triangular gesture with one hand (Fig. 4a), the system traces the recognized shape on the other hand (Fig. 4b).

This type of viewer could be particularly useful in certain scenarios where users are manipulating devices without visual components (e.g., printers or microwaves), and for some interaction schemes that have no visual modality (e.g., speech or haptic interfaces).

4 Construction of the Augmented Space with Depth Cameras

Thus far, we have presented some applications utilizing augmented space. In this section, we describe the implementation of augmented space in detail.

4.1 Installation of the Depth Cameras

The augmented space in this study was constructed using depth cameras. As mentioned in Section 2, depth cameras make it possible to collect a great deal of information, are robust under different lighting conditions,

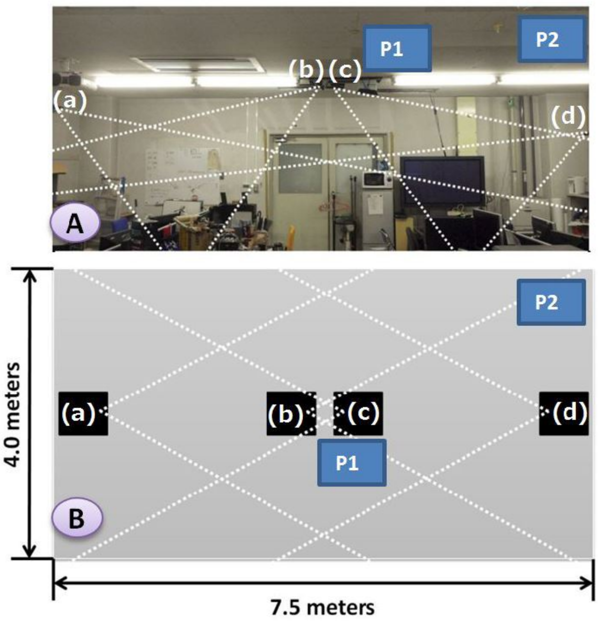


Fig. 5 Layout for the depth camera installation. Side view (A) and top view (B). (a)-(d) indicate the locations of depth cameras, and P1 and P2 indicate the locations of the projected displays. The white dotted lines indicate the fields of view for the depth cameras.

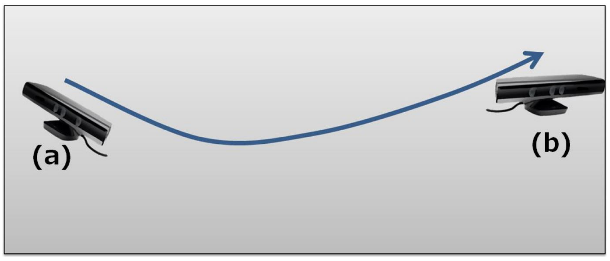
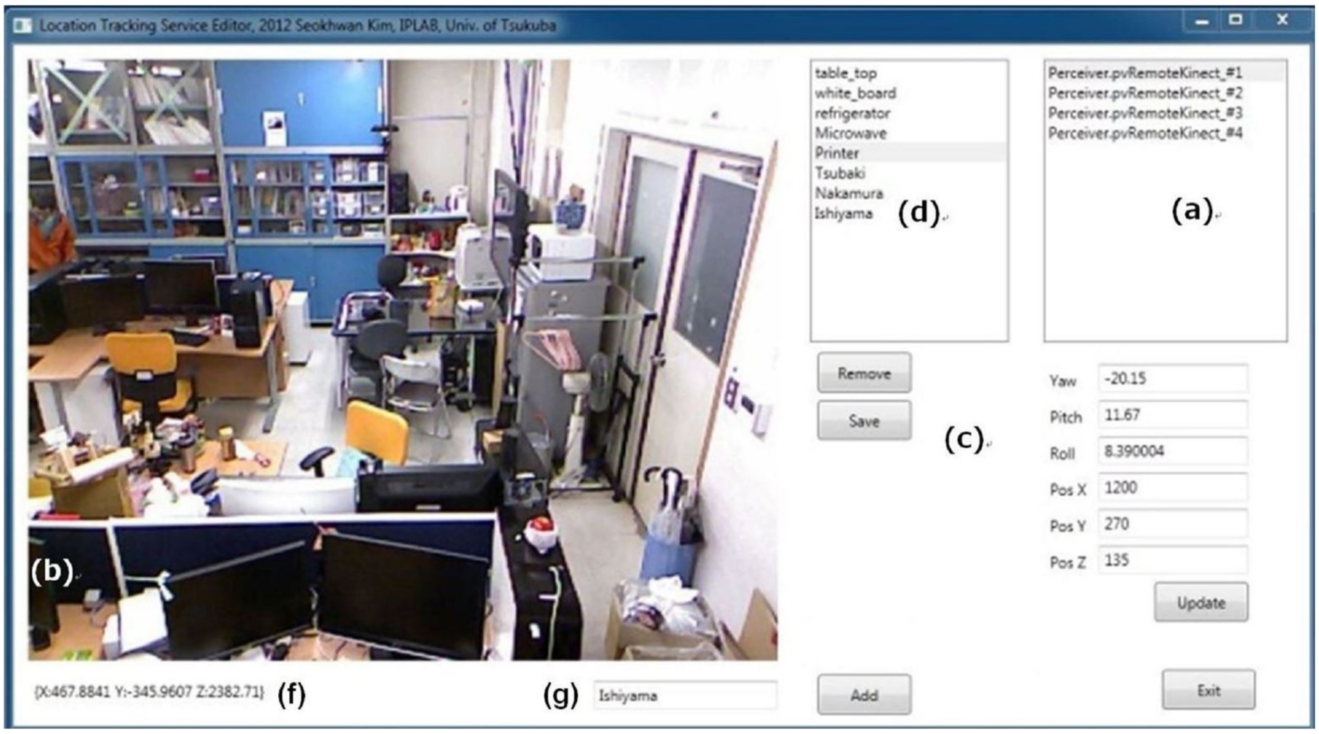


Fig. 6 Example of synchronization. If two cameras are placed at (a) and (b), they can be synchronized by moving one camera along the indicated path.

and are cost effective. To employ depth cameras, the first task is obviously their installation.

There are two primary considerations regarding installation: avoiding blind spots and ensuring that each camera covers as much area as possible. With these two concerns in mind, we installed four Microsoft Kinect cameras at points (a)-(d) in Figure 5. We took into account the cameras horizontal and vertical fields of view (FOVs), which are 43° and 57°, respectively. The white dotted lines in Figure 5 indicate the FOVs. (a)-(d) denote the locations of the depth cameras.

Another consideration is efficient tracking range. According to the official specifications, the efficient tracking range for the Kinect sensor is about 0.7-6m. However, in an informal preliminary test, the device was only able to effectively track users from 1 m to 4 m,


Location Tracking Service Editor, 2012 Seokhwan Kim, IPLAB, Univ. of Tsukuba

table_top
white_board
refrigerator
Microwave
Printer
Tsubaki
Nakamura
Ishiyama

Perceiver.pvRemoteKinect_#1
Perceiver.pvRemoteKinect_#2
Perceiver.pvRemoteKinect_#3
Perceiver.pvRemoteKinect_#4

Yaw	-20.15
Pitch	11.67
Roll	8.390004
Pos X	1200
Pos Y	270
Pos Z	135

{X:467.8841 Y:-345.9607 Z:2382.71}

Ishiyama

Fig. 7 Software used for manual adjustment and managing objects in the space. Users can add or delete object locations, and adjust a camera's pose (e.g., for fine synchronization).

which is similar to the range supported by Microsoft's official software development kit (SDK). Accordingly, we installed the cameras to cover only this range. As Figure 5 shows, there were four cameras in a distance of 7.5 m. Hence, these four cameras should be able to cover the entire area if each camera covers a range of about 1 m to 4 m.

4.1.1 Interference between Infrared Lights

A concern that arises when using multiple Kinect cameras is interference between the infrared (IR) lights. According to an informal experiment reported on a wesite, this interference can be especially severe when two depth cameras are facing each other directly [4]. However, this did not occur in our installation, because all cameras were pointed toward the floor (see Fig. 5). Moreover, we observed no interference effects in preliminary demonstrations.

4.2 Merging Multiple Depth Cameras

To combine 3D data from multiple depth cameras, synchronization (i.e., aligning the coordinates of each camera) is a necessary step. Here we describe this process.

4.2.1 Synchronization via Image Processing

Synchronization (i.e., merging) of multiple depth cameras was successfully demonstrated previously [22]. Because they set up their system in an experimental space, they were able to install retro reflective dots, which emitted relatively bright light that was easily extracted via simple image processing. However, such retro reflective dots are generally not available in actual workspaces.

Therefore, we had to develop a new synchronization procedure. Du *et al.* demonstrated a promising technique [2]. When two images are slightly different, the SIFT algorithm makes it possible to find many shared feature points between them [21]. These shared feature points play the same role as the retro-reflective dots employed in LightSpace [22]. After eliminating outliers using the RANSAC [3] algorithm, a synchronization matrix M can be found by solving Equation 1:

$$\begin{aligned}&M(X_1, X_2, ..., X_n) = (Y_1, Y_2, ..., Y_n)\\&\text{where } X_1, X_2, ..., X_n \text{ are inliers in Frame 1}\\&\text{and } Y_1, Y_2, ..., Y_n \text{ are inliers in Frame 2}\end{aligned} \tag{1}$$

This approach is based on the assumption that there are two slightly different images. Thus we can only use this method when two cameras are very close to each other, so that their images will differ only slightly. How-

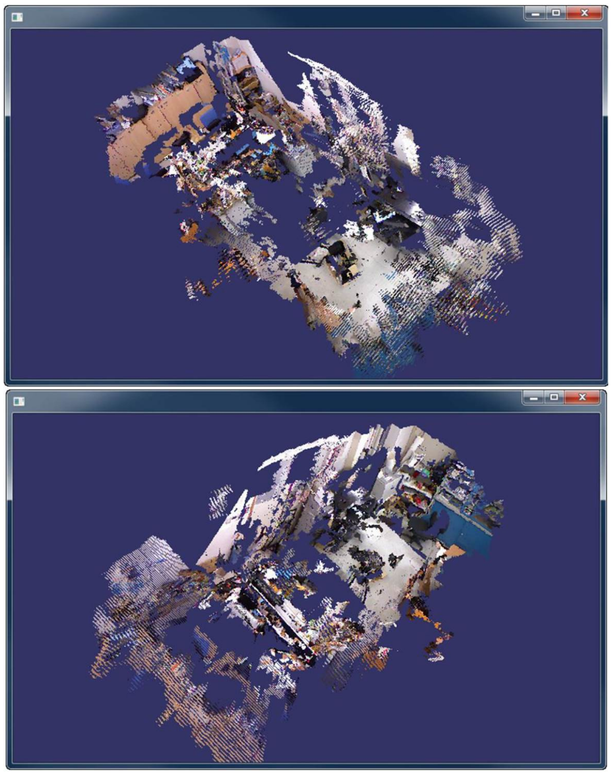


Fig. 8 Visualizations of a merged point cloud from four depth cameras, showing the space from different perspectives.

ever, as Figure 5 shows, the cameras are installed at separate locations.

To address this issue, we slowly move a camera from one location to another, capturing many frames in between, and find synchronization matrices between every consecutive frame. Figure 6 illustrates this process. Assume that two cameras are installed at points (a) and (b). The camera at point (a) is manually moved along the indicated path toward the camera at point(b), and synchronization matrices are found between all consecutive frames along the path. In this way, we can find the desired synchronization matrix by accumulating all intermediate matrices.

Using this technique, we were able to find matrices that synchronized the four cameras in Figure 5. In preliminary tests, it took about 30 min to complete this procedure.

4.2.2 Manual Adjustment

Although we were able develop an automated process, it is very difficult to eliminate all imperfections, because errors can occur when extracting matrices between frames. To deal with such errors, we developed a piece of software that allows the cameras pose (i.e., yaw, pitch, roll, and position) to be adjusted manually.

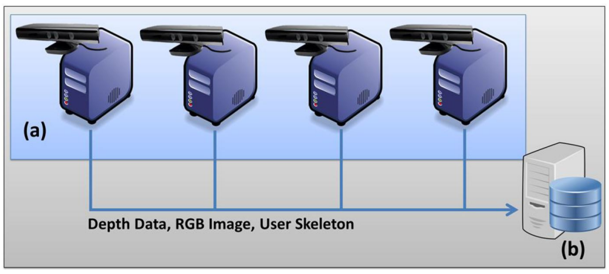


Fig. 9 Distributed system. Each of the four computers (a) hosts a depth camera, and all data are sent to the server (b).

Figure 7 shows a screenshot of the software. A list (a) of connected depth cameras is displayed. After selecting one of these cameras, the image panel (b) shows an RGB image from the selected camera. The camera's pose data (c) are also displayed. Users can edit these values, and the software shown in Figure 8 reflects the changes in real time. This software enabled us to complete the process of finding synchronization matrices for all cameras. Figure 8 shows a visualization of a point cloud from the four depth cameras shown in Figure 5.

4.3 Distributed System

After synchronizing the four cameras, we developed a distributed system to manage them. Here we describe the composition of this system.

Figure 9 shows the system composition. Each of the four computers (a) hosts one of the four depth cameras in Figure 5. The computer (b) functions as a server. Each of the four host computers sends depth data, RGB images, and user skeletons to the server. Then the server can visualize the environment, manage object locations, and track user locations, because it has access to all data from the four depth cameras. The applications shown in Figures 7 and 8 run on the server.

We expect that this scalable client-server architecture will be useful when additional cameras are necessary.

4.4 Tracking User Skeletons

As Figure 9 shows, four computers host the depth cameras and send user skeletons to the server. A problem may occur if a user is captured by more than one camera, because more than one set of skeleton data will be sent to the server. The server must accept only one set of skeleton data in such a case.

For this purpose, the host computers send not only skeleton data, but also the absolute distance between the user and the camera. Because the closest camera

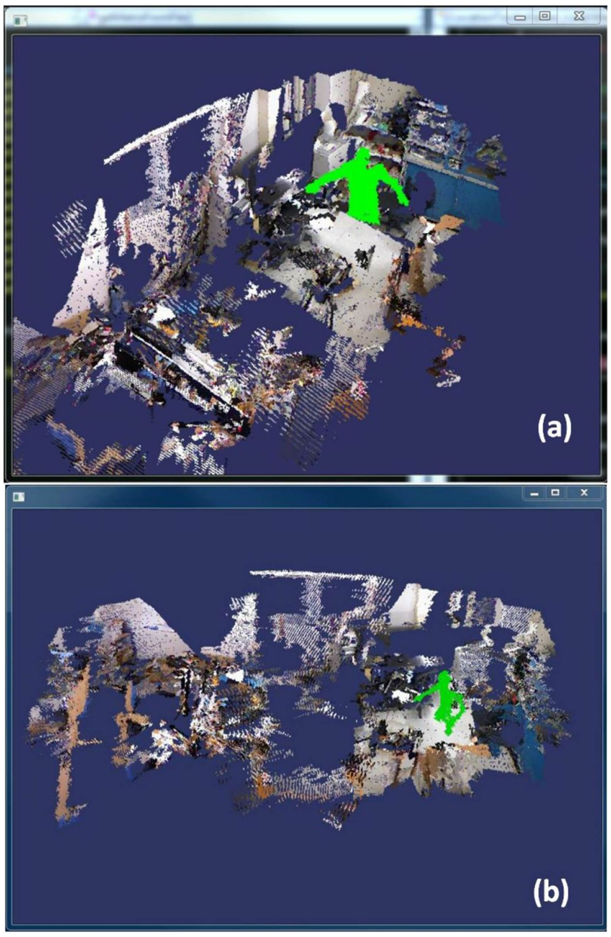


Fig. 10 A user in the space is tracked from different perspectives (a, b).

is clearly more reliable in most instances, the system accepts the skeleton data with the shortest distance, and ignores data from more distant cameras. Figure 10 shows visualizations of a user (colored green) from different perspectives.

4.5 Projected Displays for Creating an Interface on the Body

In this section, we describe the projected displays, which are located at P1 and P2 in Figure 5. The purpose of the projected displays is to display appropriate images on the palms of users.

Figure 11 shows the projected displays. As the figure indicates, each projected display is composed of a projector and a depth camera. The depth camera tracks the location of a users hand, and provides the location to the projector. Then the projector displays an appropriate image on the hand (see Figs. 2-4).

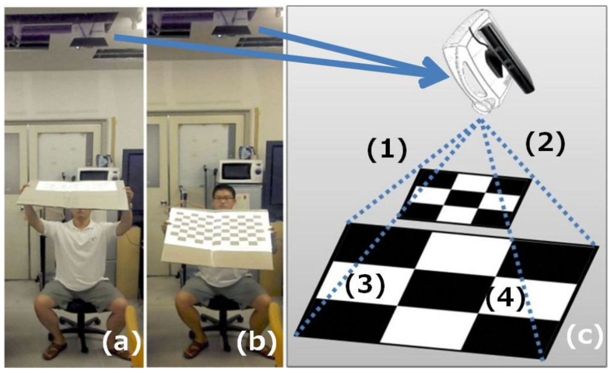


Fig. 11 Calibration of a projector and a depth camera. The camera captures images at two different heights (a, b). Then the projector's pose is determined by connecting the corners of the two images (c).

Obviously, it is necessary to calibrate the projector and the depth camera. For this purpose, the projector first displays a chessboard pattern, and two sample images are captured at different heights, as shown in Figure 11a and b. Then we can find the directional vectors that connect the corners of the two images, represented by dotted lines (1)-(4) in Figure 11c. Using these vectors, we can estimate the pose of the projector. This calibration process is required only once [6].

Some related studies used the POSIT algorithm [15] for the calibration [6][22]. However, this algorithm requires several intrinsic parameters of the camera and projector, and these parameters were not available to us. Therefore, we developed this method.

5 The Support Tool

To enable the applications mentioned in Section 3, registration of objects is also necessary. The software shown in Figure 7 can be used for this purpose. We present the details below.

5.1 Pick What You See: Collecting the Locations of Objects

Although the system provides visualizations of point clouds, the quality is usually not very high. Our support tool is capable of displaying RGB images from each camera. When the depth cameras are connected to the server via the host computers, they appear in list box (see Fig. 7a). Users can select any camera from the list, and the corresponding RGB image will be displayed (Fig. 7b).

This RGB image can be used to collect object locations. When a user clicks on a 2D point in the RGB

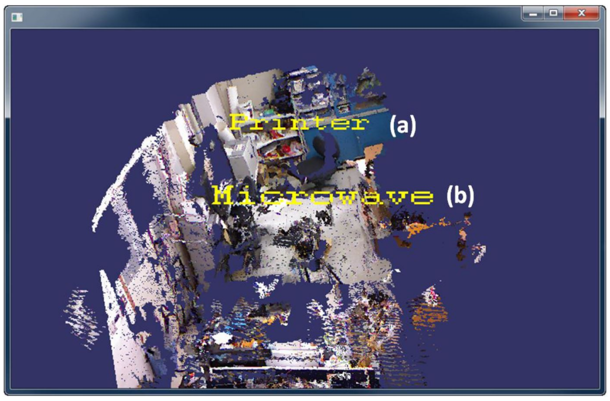
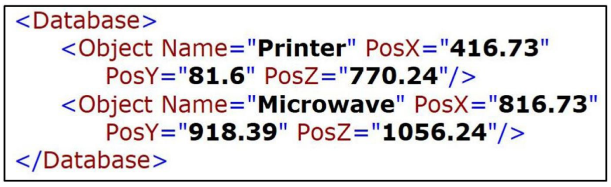


Fig. 12 The visualization of registered objects in the space. Text labels (a,b) are placed near the objects.



```
<Database>
    <Object Name="Printer" PosX="416.73"
        PosY="81.6" PosZ="770.24"/>
    <Object Name="Microwave" PosX="816.73"
        PosY="918.39" PosZ="1056.24"/>
</Database>
```

Fig. 13 An example of the XML database. This example shows the positions and names of a printer and a microwave.

image, it is converted into a 3D point by adding the relevant depth data, and the results (Fig. 7f) are displayed in the bottom left corner of the window. The user can input the name of an object in a text box (Fig. 7g), and the object will be added to the list (Fig. 7d). Below this list box there are Remove and Save buttons. When the user selects an item in the list box and clicks the Remove button, the corresponding object is removed. When the user clicks the Save button, the current locations of registered items are stored in an extensible markup language (XML) file for later use. Figure 13 shows an example of the database file containing the names and locations of objects.

All of these changes (i.e., registration and deletion) are reflected in real time, as shown in Figure 12. In the example illustrated, text labels for the two items (printer (a) and microwave (b)) are added.

Because the desktop environment is 2D, we expect that the 2D graphical panel (the RGB image) is easier to use than the 3D visualization (the point cloud). Indeed, the difficulty created by mismatched input and output dimensions has been an issue of note for a long time.

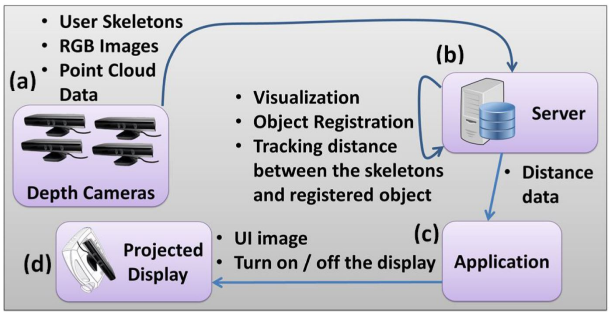


Fig. 14 Overall composition of the system. Depth cameras (a) send their data to the server (b). The server stores the necessary context (i.e., locations) and provides that context to the software (c) when necessary. The software can turn the projected display (i.e., the visual interface on the palm) on and off.

5.2 Visualizing Only Necessary Data to Support Real-Time Updating

It is difficult to support real-time updating (i.e., to guarantee more than 15 frames per second [FPS]). When point clouds from four cameras are merged, the server must process about 17 MB of data per frame, assuming that the cameras have VGA resolution (640 X 480), including three floating point numbers per 3D point (12 bytes) and RGB data (3 bytes). Even though the amount of data varies (e.g., missing depth data), it is almost impossible to process such a tremendous amount of data in real time. In preliminary tests, we were only able to obtain 2-3 FPS, and our server has an Intel Core i7 2.8 GHz processor, 8 GB RAM, and an nVidia GTS250 chipset.

To address this, the system has a feature whereby only necessary data are visualized. In this mode, the host computers send only point cloud data corresponding to users, and more than 20 FPS was obtained.

5.3 Summary

Figure 14 shows the overall composition of the system. In this figure, machines (a) and (b) correspond to machines (a) and (b) in Figure 9. The depth cameras (a) send user skeletons, RGB images, and point cloud data to the server (b).

Then the server can visualize a point cloud and display an RGB image (see Fig. 7, 8, 10, 12). As mentioned above, users can register objects, and the server can calculate the distances between the objects and user skeletons. These distance data are transferred to the software (c), and the software, in turn, can display an image on a users palm via the projected display (d).

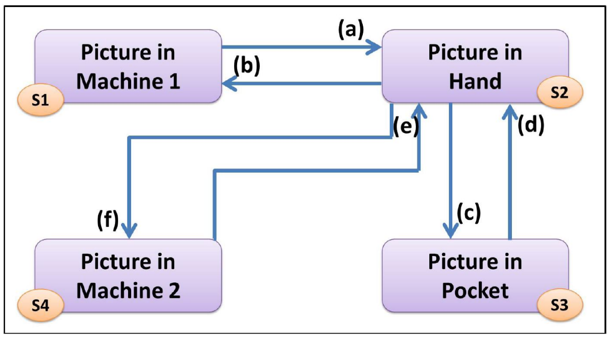


Fig. 15 State diagram for the application that presents data storage on the body.

6 Implementation of the Applications

In this section, we explain the implementation of the applications described in Sections 1 and 3.

6.1 Smart Notification of a Printing Task

In Section 1, we introduced an application that notifies a user of a printing task when the user is close to a printer. To implement this application, we must first input the location of a printer, and this is done using the software shown in Figure 7. Then the server (Fig. 14b) examines the distance between the printer and the user skeleton. The distance data are shared with the software (Fig. 14c). If the distance is relatively small, the software turns on the projected display and causes the notification to be displayed on the users palm (see Fig. 1).

After an image is displayed on the users palm, the distance between the printer and the users palm is continuously tracked. If this distance is relatively small, the software prints a sample page. Communications among the server, software, and projected display are manually established via a network. The system is connected to a printer, and a routine for printing a sample page is included in the software.

6.2 Data Storage on the Body

In Section 3.1, we introduced an application that enables the storage of digital content on some part of the body and the transfer of the content to machines. To implement this application, we must first input the locations of two computers, which can be registered using the software shown in Figure 7. Then the distance between the user's right hand and the computers is continuously examined.

When a touch event occurs (i.e., between the user's right hand and a machine or between the hand and the pocket), the system takes a different action depending on the state of the application. Figure 15 shows the state transition diagram. There are four states, depending on the location (the two machines, the hand, and the pocket) of the data (the image on the computer shown in Fig. 2).

The application starts with the state ”picture in machine 1” (state S1 in Fig. 15), and the picture is first shown at that location. When the user touches machine 1, the state changes to picture in hand (state S2, transition (a) in Fig. 15). When the hand is close to the pocket, the state changes to ”picture in pocket” (state S3, transition (c) in Fig. 15). When the user touches the pocket again, the state reverts to ”picture in hand” (state S2, transition (d) in Fig. 15). Employing the same principle, the user can move the picture to machine 1 (state S1, transition (b) in Fig. 15), to machine 2 (state S4, transition (f) in Fig. 15), or back to his pocket (state S3, transition (c) in Fig. 15).

6.3 Virtual Midair Input

The implementation of this application is almost the same as that of the smart notification example (see Section 6.1). To reserve a midair location, the user places his hand at the location, and the location is secured via the software shown in Figure 7.

When the user's left hand is near the location, the software turns on the projected display and displays an image. A gesture is recognized by tracking the movement of the right hand. In tests, we constructed a simple light emulator (shown in the display (1) of Fig. 3b) that was manually connected to the system via the network. Recognition of hand gestures is explained in the next section.

6.4 Complementary Interface

For the application in this example, we arbitrarily specified the right hand for making gestures and the left hand for the feedback viewer. Using the skeleton data from the depth cameras, the left and right hands are easily distinguished. For gesture recognition, we employed the one dollar gesture recognizer [23]. This created a problem, because the depth camera provides 3D hand movement data, whereas the one dollar recognizer accepts only 2D data. To address this issue, we projected the 3D data onto an appropriate 2D plane, as in a previous study [5]. The system continuously provided the right hand movements to the recognizer, and a visualization was displayed when a gesture was recognized.

7 Limitations and Future Work

Here we describe the limitations of the current implementation, and suggest some promising future directions.

7.1 Temporary Disappearance of a User

One concern with the current system is that a user's location can temporarily be lost. In particular, this occurs when users cross the borders between coverage areas of different cameras (i.e., intersections between the dotted lines in Fig. 5). Currently, the OpenNI implementation takes 1-2 s to recognize a skeleton, depending on the image processing capabilities or sensors. When more sophisticated algorithms and devices are developed, this issue will be addressed.

7.2 Lack of Accuracy

The accuracy of the depth camera installation is not high. We are currently using four Kinect cameras to cover a 4 m - 7.5 m space. This setup could be used to cover a broader area, but high accuracy is not easily obtained. Because we employ a distributed system architecture, we could add more depth cameras when higher accuracy is required. However, this issue can be better addressed when more advanced (i.e., higher resolution) depth cameras become available.

7.3 Supporting Complex Shapes

The current system treats object locations as points. This is because the system is not accurate enough to support complex shapes (e.g., spheres or cubes). Although this was sufficient to implement the applications demonstrated here, a greater variety of applications would be possible if the system supported complex shapes. It would be worthwhile to develop a more advanced version of the software shown in Figure 7, and resolve the issues related to it.

7.4 Continuous Tracking of Objects

In this work, we considered only interactions among static objects. Of course, some objects in a space (e.g., books or laptops) can be moved. To support interactions with such movable objects, continuous tracking is necessary, and should be investigated in future work.

There are various technologies that can track moving digital devices with good accuracy [14]; however, there are fewer solutions for non-digital devices (e.g., notes or books). We expect that our system may be extended to create an infrastructure for tracking non-digital devices. In our setup, an object is imaged using multiple cameras (see Fig. 5); therefore, when an object is registered using the software (see Fig. 7), the system can obtain multiple images from different angles. Such images are valuable in improving the accuracy of object detection. This is a possible area for future work, when more sophisticated image-processing algorithms and higher resolution cameras become available.

7.5 Identifying Users

We have demonstrated a system that can track users; however, we did not consider how the system might identify users. Although this is beyond the scope of this work, there is a promising scenario that may be enabled with the current implementation.

In many laboratories and offices, most users undergo some authentication procedure when they enter the workplace. If our system can communicate with the authentication system, it could continuously track a user with a registered profile. Realizing this scenario and addressing new issues related to it is a direction for future work.

8 Conclusions

We have demonstrated applications that can be achieved via a visual interface on the human palm, which is always available and is location-sensitive. We constructed an augmented space in a workspace by adding depth cameras. We employed a distributed system to manage the depth cameras, and developed a support tool to facilitate the collection of object locations. We have also demonstrated several potential applications.

In future work, we plan to focus on continuous tracking of objects and user identification (see Sections 7.4 and 7.5). Potentially, every surface on objects in a room can be augmented to be interactive. Moreover, it can be fully personalized. Lee *et al.* demonstrated augmenting various surfaces [13] and, with our system, it is possible to make them fully personalized. For example, when a user reads a newspaper, our system can project a related article onto it, selected based on a user's personal interests; it can also be automatically translated into the user's native language. Another example is that users can write a message using gestures on a book, and attach it on a book (or anywhere they wish). The attached message can then only be read by specific people.

We anticipate that this work will contribute to the development of an infrastructure for such services, and may help to create new interactions in such environments.

References

1. Brumitt, B., Meyers, B., Krumm, J., Kern, A., Shafer, S.A.: Easyliving: Technologies for intelligent environments. In: Proceedings of the 2Nd International Symposium on Handheld and Ubiquitous Computing, HUC '00, pp. 12–29. Springer-Verlag, London, UK, UK (2000)
2. Du, H., Henry, P., Ren, X., Cheng, M., Goldman, D.B., Seitz, S.M., Fox, D.: Interactive 3d modeling of indoor environments with a consumer depth camera. In: Proceedings of the 13th International Conference on Ubiquitous Computing, UbiComp '11, pp. 75–84. ACM, New York, NY, USA (2011)
3. Erdem, C.E., Bozkurt, E., Erzin, E., Erdem, A.T.: Ransac-based training data selection for emotion recognition from spontaneous speech. In: Proceedings of the 3rd International Workshop on Affective Interaction in Natural Environments, AFFINE '10, pp. 9–14. ACM, New York, NY, USA (2010)
4. Fukushima, N.: Interference of ir light from multiple kinects @ONLINE (2010). URL http://youtu.be/p2dokAm1uQg. [Online; accessed 30-January-2014]
5. Gustafson, S., Bierwirth, D., Baudisch, P.: Imaginary interfaces: Spatial interaction with empty hands and without visual feedback. In: Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology, UIST '10, pp. 3–12. ACM, New York, NY, USA (2010)
6. Harrison, C., Benko, H., Wilson, A.D.: Omnitouch: Wearable multitouch interaction everywhere. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, pp. 441–450. ACM, New York, NY, USA (2011)
7. Harrison, C., Tan, D., Morris, D.: Skinput: Appropriating the body as an input surface. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '10, pp. 453–462. ACM, New York, NY, USA (2010)
8. Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., Jansen, E.: The gator tech smart house: A programmable pervasive space. Computer **38**(3), 50–60 (2005)
9. Kim, S., Takahashi, S., Tanaka, J.: New interface using palm and fingertip without marker for ubiquitous environment. In: Proceedings of the 2010 IEEE/ACIS 9th International Conference on Computer and Information Science, ICIS '10, pp. 819–824. IEEE Computer Society, Washington, DC, USA (2010)
10. Kim, S., Takahashi, S., Tanaka, J.: Point-tap, tap-tap, and the effect of familiarity: to enhance the usability of see-and-select in smart space. Transaction of Human Interface Society **14**(4), 445–455 (2012)
11. Lee, C., Nordstedt, D., Helal, S.: Enabling smart spaces with osgi. IEEE Pervasive Computing **2**(3), 89–94 (2003)
12. Lee, J., Choi, J., Shin, D., Shin, D.: Multi-user human tracking agent for the smart home. In: Proceedings of the 9th Pacific Rim International Conference on Agent Computing and Multi-Agent Systems, PRIMA'06, pp. 502–507. Springer-Verlag, Berlin, Heidelberg (2006)
13. Lee, J.C., Hudson, S.E., Tse, E.: Foldable interactive displays. In: Proceedings of the 21st Annual ACM Symposium on User Interface Software and Technology, UIST '08, pp. 287–290. ACM, New York, NY, USA (2008)
14. Liu, H., Darabi, H., Banerjee, P., Liu, J.: Survey of wireless indoor positioning techniques and systems. Trans. Sys. Man Cyber Part C **37**(6), 1067–1080 (2007)
15. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. Int. J. Comput. Vision **60**(2), 91–110 (2004)
16. Marquardt, N., Diaz-Marino, R., Boring, S., Greenberg, S.: The proximity toolkit: Prototyping proxemic interactions in ubiquitous computing ecologies. In: Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology, UIST '11, pp. 315–326. ACM, New York, NY, USA (2011)
17. Patel, S.N., Rekimoto, J., Abowd, G.D.: icam: Precise at-a-distance interaction in the physical environment. In: Proceedings of the 4th International Conference on Pervasive Computing, PERVASIVE'06, pp. 272–287. Springer-Verlag, Berlin, Heidelberg (2006)
18. Pinhanez, C.: The everywhere displays projector: A device to create ubiquitous graphical interfaces. In: G. Abowd, B. Brumitt, S. Shafer (eds.) Ubicomp 2001: Ubiquitous Computing, *Lecture Notes in Computer Science*, vol. 2201, pp. 315–331. Springer Berlin Heidelberg (2001)
19. Rekimoto, J.: Pick-and-drop: A direct manipulation technique for multiple computer environments. In: Proceedings of the 10th Annual ACM Symposium on User Interface Software and Technology, UIST '97, pp. 31–39. ACM, New York, NY, USA (1997)
20. Shin, C., Lee, W., Suh, Y., Yoon, H., Lee, Y., Woo, W.: Camar 2.0: Future direction of context-aware mobile augmented reality. In: Ubiquitous Virtual Reality, 2009. ISUVR '09. International Symposium on, pp. 21–24 (2009)
21. Sinha, S.N., Frahm, J.M., Pollefeys, M., Genc, Y.: Feature tracking and matching in video using programmable graphics hardware. Mach. Vision Appl. **22**(1), 207–217 (2011)
22. Wilson, A.D., Benko, H.: Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In: Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology, UIST '10, pp. 273–282. ACM, New York, NY, USA (2010)
23. Wobbrock, J.O., Wilson, A.D., Li, Y.: Gestures without libraries, toolkits or training: A $1 recognizer for user interface prototypes. In: Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology, UIST '07, pp. 159–168. ACM, New York, NY, USA (2007)