Active Interaction: Live Remote Interaction through Video Feeds

Jeffrey Naisbitt, Jalal Al-Muhtadi, Roy Campbell
{ naisbitt@uiuc.edu, almuhtad@cs.uiuc.edu, rhc@cs.uiuc.edu }

Department of Computer Science
University of Illinois at Urbana-Champaign
Champaign, Illinois, USA

Abstract. Ubiquitous computing environments and the plethora of mobile devices that populate them have led to a global trend for remote collaboration and interaction. This trend demands seamless interfaces for remote interaction with these ubiquitous environments and their inhabitants. Existing applications for remote interactions are limited and typically require the user to mentally translate virtual avatars, tags, or names into real objects. Location information from accurate location sensing technologies, such as Ubisense, and other location sensing devices, facilitate enhanced interactions. Fusing the location information with live video feeds from multiple standard pan-tilt cameras, we provide a seamless, easily reconfigurable, real-time interface for remote interactions within distributed ubiquitous environments. Users interact with objects directly through the video feed itself. In effect, our framework dynamically discovers all resources and programmable objects in the vicinity and allows remote users to interact with the environment and all of its resources and objects as if they were physically present in the environment.

1 Introduction

Ubiquitous computing allows the coupling of the physical world to the information world. Active spaces organize networked computer devices into a distributed system that coordinates its activities with its mobile users. The human endeavors of creativity, learning, and collaboration in areas from science and engineering to art, music and the humanities become a crucible for new perspectives on how ubiquitous computing can impact society, facilitate collaboration, and serve users. Our research on Active Spaces examines how ubiquitous computing can support various physical human activities by providing a middleware infrastructure that supports the construction and management of truly immersive ubiquitous computing environments. We refer to our infrastructure as Gaia [2]. In this paper, we present a framework that extends our infrastructure and enriches it with remote control, management, and programmability. The goal of the framework is to offer remote users the ability to interact with active spaces and their contents, even if the users are located in an environment where rich device connectivity is not available.

The framework integrates dynamic discovery, accurate location sensing, and live cameras to create a real-time, seamless, dynamic interface that allows remote users to interact directly with the active space, access services, utilize resources, and collaborate with other users.

The majority of previous work focuses on local interaction within ubiquitous computing environments. Lights, machines, and other devices and services turn on or perform certain actions when a user is close. Specific device functionality can be restored or disabled depending on the proximity of specific users. However, the plethora of mobile devices has led to a global trend for supporting remote collaboration and remote interactions with ubiquitous computing environments and the devices and services contained within these environments. This trend demands seamless interfaces for real-time, remote interaction with these ubiquitous environments.

The existing applications for such interaction are limited. Most primarily use statically drawn three-dimensional models that allow the user some control, but they do not really show what is going on in the room. They do not allow the user to actually "see" people and objects; they just see tags, names, or virtual representations. Actually seeing the people and dynamic objects in the room provides a much richer environment for interaction. Additionally, the user is able to detect and recognize more objects, thus facilitating interac-

tion with a larger set of dynamic and mobile objects. Using the video feed itself, we also eliminate the initial overhead required for generating the three dimensional models. Finally, current work is limited to a single ubiquitous environment, whereas we extend this interaction to a conglomeration of these environments.

Location information from accurate location sensing technologies, such as Ubisense [12], and other location sensing devices, facilitates this enhanced interaction. Fusing the location information, provided from the MiddleWhere infrastructure [1], with live video feeds from standard pan-tilt cameras, we provide a novel real-time interface for remote interaction within ubiquitous computing environments. In fact, we extend this interface to aggregations of multiple ubiquitous computing environments, using multiple cameras, a central dynamic location database, and the Gaia middleware infrastructure. We provide a very simple interface using direct interaction with the video from the live camera feeds.

Additionally, we provide multiple interfaces for defining objects within the room, both locally and remotely. This is useful if the objects are non-discoverable or simply user-defined hotspots. Using the location sensing devices themselves, we allow the user to simply outline the physical objects in the room. This eliminates the need to define precise room structure and object location beforehand. Additionally, no measuring or complex mapping techniques are necessary since anyone can outline a physical object.

We divide the remainder of this paper as follows. Section 2 presents some related work. Section 3 talks about our supporting infrastructure. Section 4 presents our architecture for the actual interface. Section 5 describes the control flow protocol. We conclude in Section 6 and present future work in Section 7.

2 Related Work

Currently, most related work focuses on different methods for mapping rooms and their objects. The actual interfaces focus primarily on virtual models, using virtual tags to represent objects and people. These methods require that tags or virtual objects be created for each object with which the user will interact, thus limiting the usability and increasing the configuration overhead. Current interaction projects primarily include simple application movers, 3D virtual modeling, augmented reality techniques, local location sensors, and virtual reality. Related modeling projects include three dimensional static modeling, ray tracing and image processing.

Biehl et al. [3] implemented a very basic interface for moving applications from one display/machine to another. This provides a useful interface, but it does not take into account the many other physical objects in the room – both static and dynamic. Additionally, this service only works with applications specifically designed using the application framework developed by Roman et al. [4]. Although the framework provides useful tools for application development within ubiquitous systems, it currently does not extend beyond the Gaia framework. Finally, this approach completely ignores the dynamic state of the current environment.

Glasberg et al. [5] developed a virtual interface for viewing and controlling PowerPoint presentations. He is currently working on a three dimensional virtual interface for more versatile interaction with a room. This provides a very nice interface with the ability to traverse the virtual room and view objects from any angle and zoom-factor. However, this requires objects be defined and stored statically, and dynamic objects are not included. Additionally, you do not get a full view of the actual current events in the room. Furthermore, this approach forces the user to see the avatars statically implemented within the model for representing people and various objects. Finally, the initial setup and configuration generally takes several days.

Hosoya et al. [7] present an interesting interface for interaction with both remote and local objects. Using RF sensors, they keep track of static and dynamic objects in a room. Using a video mirror image of the user's locality, they overlay virtual tags representing the objects with which the user can interact. This interface is interesting in that you can physically interact with the objects by obstructing their location in the mirrored video image. However, this interface is limited in that it requires a fairly life-sized image for accurate interaction with objects. Additionally, it lacks the ability to actually see and physically recognize the desired remote objects, as we provide through the

video feed; given that they must still create and configure avatars for interaction.

AT&T Labs in Cambridge [8] use small sensors, called *Bats* to track location and interact physically with the space. Although this facilitates object tracking and local interaction, they have not developed any interface for remotely interacting with these objects. Interaction takes place locally as the user manipulates these Bats.

Butz et al. [9] developed an interesting method for interacting with local rooms using virtual reality. Using special virtual reality goggles, they use augmented reality techniques of overlaying the avatars and tags over the actual room that the user sees. This is a great interface for local interaction, but it requires costly equipment, and it does not provide a method for remote interaction.

Harle et al. [10, 11] attempt to produce accurate dynamic models of objects in ubiquitous environments by using various image-processing techniques. This is a useful, although somewhat complicated method for environment mapping, but it does not provide any means for interaction. In the future, we could use mapping techniques such as these to enhance our application detection features.

3 Infrastructure

We extend the Gaia infrastructure to enable remote interaction and collaboration with objects, resources, and people in an active space. The remote interaction features live video and a seamless interface for interacting with objects and people. The interface can be accessed over the Internet, and it consists of a light client. We currently plan to implement this client for mobile devices such as PDAs or smart phones. We refer to the framework as "Active Interaction". Active Interaction utilizes Gaia services, including discovery, location, and communication primitives. In this section, we briefly talk about these aspects of our infrastructure.

3.1 Gaia

In previous research, we developed Gaia. Gaia is a distributed meta-operating system designed to facilitate ubiquitous computing. Gaia provides the necessary infrastructure for a heterogeneous collection of devices to intercommunicate and coordinate themselves seamlessly into an active space. We also introduced the notion of a *super space*. We define a super space as a collection of reflective and recursive active spaces that allow the management, operations and maintenance of large-scale ubiquitous computing environments. Active Interaction utilizes Gaia communication layer and kernel services to enable seamless interaction interfaces.

3.2 Discovery

Active Interaction utilizes some of the features of the Olympus Discovery Service in Gaia [13] to discover all entities that exist in a given spatial region, their properties and exact coordinates. The Olympus Discovery Service supports semantic discovery (using ontologies), as well as spatial queries.

3.3 Ubisense and MiddleWhere

A key issue in Active Interaction is to provide a method to obtain accurate locations of objects and people, so that a remote interface can track and depict these entities accurately. We currently employ Ubisense location technology to obtain accurate 3D location information. Ubisense is a unidirectional UWB (Ultra Wide Band) location platform that uses a bidirectional TDMA (Time Division Multiple Access) control channel. RFID tags transmit UWB signals to networked readers and are located using "angle of arrival" and "difference of arrival" techniques. Ubisense has an accuracy of 6 inches with 95% confidence. The tags can be attached to various objects and people throughout the spaces, thus providing very useful information about any mobile object's location.

Even though Ubisense provides very useful and accurate location information, it is not currently available in all rooms or active spaces. Moreover, it is possible that other accurate location technologies are deployed in the future. Therefore, to be independent of the location technology being used, we rely on the MiddleWhere location infrastructure [1]. MiddleWhere provides a middleware layer that stores and updates location information in real-time. It aggregates location information from various location technologies and provides a

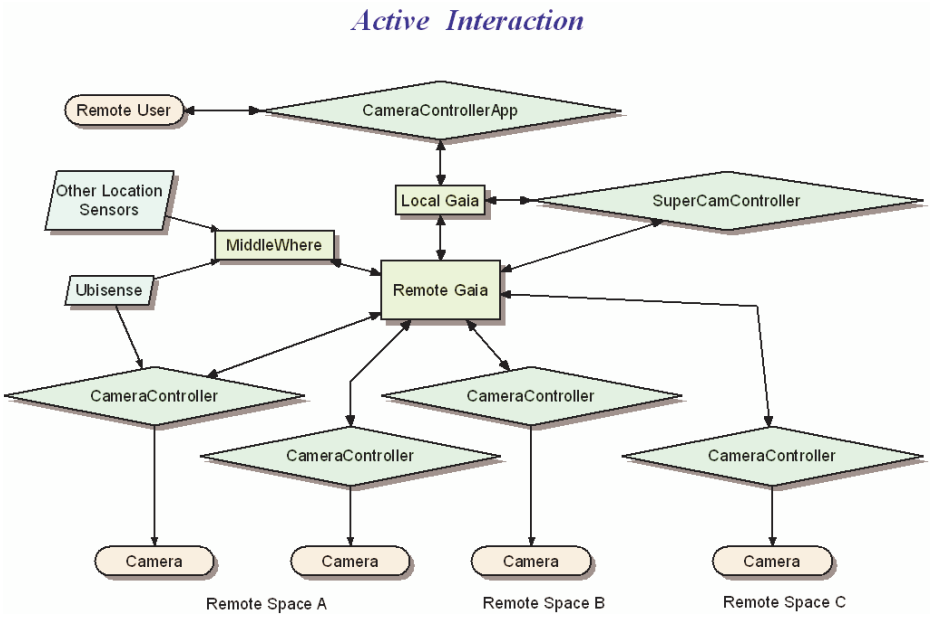

Figure 1. Interaction between the various application components and the middleware infrastructures.

uniform interface for obtaining location information. Additionally, it includes information about the accuracy of the location sensing devices. Therefore, given the location of an object, we can determine how accurately to rely on that data.

4 Active Interaction Architecture

Active Interaction consists of the following primary components. A *CameraController* runs for each camera located within the spaces. The *CameraControllerApp* provides the graphical user interface. The *SuperCamController* provides the necessary interfaces for extending this service to super spaces. Finally, we describe our simple interface for defining non-discoverable objects and hotspots in the rooms. See Figure 1 for the overall component infrastructure for the application.

4.1 CameraController

The *CameraController* implements the various device drivers necessary for controlling the different camera models, including the different interfaces for interacting with them. The interface for these camera controls is made public using CORBA. Any user or application, wishing to remotely control or access the cameras, simply makes the appropriate CORBA function call (handled transparently by Active Interaction). The camera controllers implement the necessary transformations for converting between the camera coordinate system and the coordinate system used by the location infrastructure. We discuss the process for the actual object detection and distinction in the Object Selection section below. Additionally, the camera controllers do the primary location queries for tracking individuals and objects. The tracking does not require any image processing; it relies on the conglomeration of location sensing devices provided by the MiddleWhere infrastructure.

4.2 SuperCamController

The *SuperCamController* provides a single interface for interacting with and viewing objects through multiple active spaces. In order to facilitate mobile object tracking, we provide a single web, HTTP interface for monitoring these video

streams. The controller selects and broadcasts the appropriate video stream depending on which camera most appropriately displays the desired objects or people. Users view this video feed using this web interface, or any graphical user interface capable of viewing Motion JPEG video streams. When an object moves between spaces, or between camera regions, the super controller determines the best *active space* and camera for tracking or interacting with the objects, and then updates the video stream automatically, and transparently, changing it to reflect the new camera's view. The CameraControllerApp is notified of this change as well, thus allowing it to seamlessly communicate with the camera whose video the user is viewing.

4.3 CameraControllerApp

The CameraControllerApp provides the primary graphical user interface for the remote interaction with the active spaces. The actual video feed is the primary component of the interface. Through this video window, the remote user can interact with objects in the spaces. Several tabbed dialogs provide methods for manually controlling the cameras and location-related functionality. Additionally, this application provides the necessary interface to the camera controllers for selecting objects located within the real-time video display. We present the control flow for these interactions in section 5 below.

The video display primarily supports Motion JPEG streams, but we have also implemented support for standard DV inputs as well. Currently, we are trying to implement this display using OpenGL to provide support for the augmented reality and video overlays – discussed in the related and future work sections.

The CameraControllerApp is a thin client, providing a basic interface for interaction. The computations and any intensive network communications all take place remotely, within the active space itself. Therefore, we can implement this interface on smaller, more mobile devices such as smart phones or PDAs.

4.4 Manually Defining Objects and Hotspots

We developed and are currently implementing the infrastructure for defining objects and hotspots that are not automatically discovered by the location infrastructure. We define hotspots as objects or locations in the active spaces, with which the user can interact. Any object or spot in a room can have any functionality associated with it. This enhances our framework by allowing further interaction. First, we can define objects and hotspots by physically manipulating the Ubisense tags. Second, we are currently working on an interface for defining these hotspots through the video feed itself.

The Ubisense tags have buttons on them that trigger events. The user can define the start and stop points for defining object or hotspot dimensions using these simple buttons. In order to define a new object within an active space, the user simply holds the tag in a corner of the object and presses a button to signal the start of the object detection method. The user then traces the object while holding the button and releases the button when the object has been defined. The information from the Ubisense tag generates a polyhedron, which we associate with a name and interaction interface. This information is stored in the location database within the MiddleWhere infrastructure, thus seamlessly integrating it with the dynamically discovered objects.

Additionally, we are currently developing an interface for outlining hotspots using the video feed interface itself. In order to define a new hotspot, the user enters a "hotspot definition mode" so that the application does not attempt to recognize the region as a currently defined object. Similar to selecting an object, the user can outline a region within the video feed. A new polyhedron is created, and the user supplies a name and interface for the object.

We still need to complete the implementation of the interface for defining what the object is, and what forms of interaction are available for the object. However, the infrastructure is in place, and object recognition is available. In the future, we plan to use existing image processing techniques, as described in the Related Work section, to provide more accurate methods for dynamic object definition and tracking.

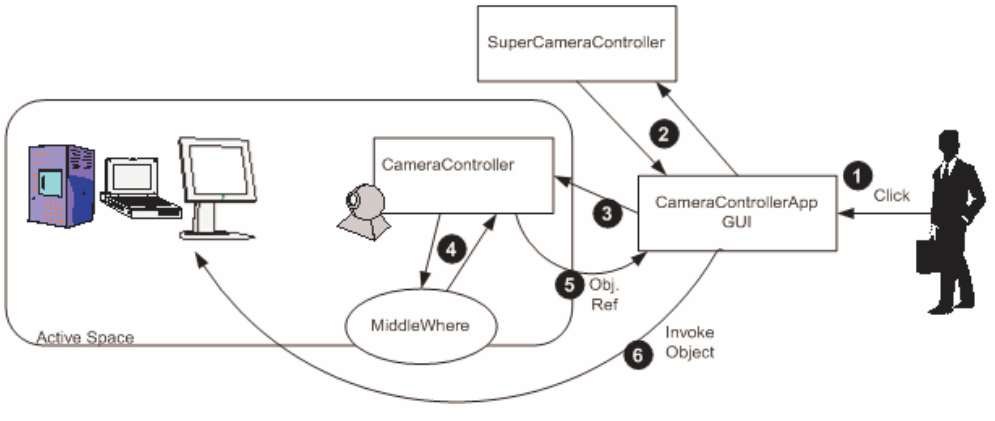

Figure 2. Control flow protocol for remote interaction. Describes the steps involved in the user interaction with a remote object.

5 Control Flow Protocol

The control flow for the remote interaction consists of object selection, lookup of object definition and interface, and finally remote invocation or interaction. See Figure 2 for the control flow graph and Figure 3 for a screenshot of the graphical user interface.

5.1 Object Selection

The primary method for interaction is through basic mouse-clicks. Using a mouse, the user clicks on one of the various objects in the CameraControllerApp's video window. Whether the user left-clicks or right-clicks, results in different actions as in the subsection below. The CameraControllerApp then communicates the window's x, y information to the corresponding camera controller.

5.2 Lookup of Object Definition & Interface

The CameraController uses the point received from the CameraControllerApp to calculate the corresponding camera coordinates in terms of its internal pan and tilt degrees. The controller then converts this information to a directional vector, in terms of the real-world Cartesian coordinates of the location infrastructure. This vector represents the direction of the selected object from the camera's location. We only compute the directional vector since the actual magnitude of the vector, corresponding to the depth of distance from the camera, is impossible to determine without more advanced image processing techniques, triangulating with multiple cameras, or using expensive stereo cameras. We find that the directional vector is sufficient anyway, given the accuracy with which the user can select objects within the real-time video window.

Using this vector, the CameraController then constructs a spatial query, which it sends to the location infrastructure. It can optimize the query by limiting it to the region surrounding the vector. Depending on the accuracy of the location information, the region can be enlarged to ensure more accurate object selection as well. MiddleWhere returns all the objects within this specified region. Note that MiddleWhere keeps an updated list of all objects in the space using the discovery framework described previously.

Next, the controller intersects this list of objects with the directional vector. Generally, this returns a single object, however objects can appear in front of each other from the camera's perspective. Therefore, in order to determine which object we should select, the controller simply compares the distances of these objects from the camera itself. The object with the shortest distance from the camera is the one selected, and this object is returned to the CameraControllerApp along with various properties and its available IDL interactions and corresponding LUA scripts (described below).

5.3 Remote Invocation or Interaction with Objects

Obviously, different objects facilitate or require different forms of interaction. The available interfaces and interactions for each object are stored within the object returned from the location infrastructure. These interfaces are defined as CORBA IDL objects. For our framework, we support a default action for any object discovered or defined. If the user left-clicks on the object, the default action is performed - if one has been defined. Otherwise, we treat it as a right-click. When the user right-clicks, a context-menu with the available options is displayed. This menu is primarily a list of all the defined interactions available through the object. Primarily, this list represents items corresponding to the IDL interface for the object. Additionally, more advanced interactions are available using LUA scripts [14]. LUA is a scripting language that, among other features, provides simple interfaces for interacting with IDL interfaces through CORBA and Gaia. Complicated interactions with objects that require invoking a series of different operations can be represented with LUA scripts. The CameraControllerApp itself uses the object reference to communicate and interact directly with the selected object. For example, we can turn lights on, control computers and displays, or perform any other user-defined or system-defined action. Given the applications running on a machine, we could interact directly with them as well, provided they implement an IDL interface.

The control flow for interacting with people is the same; however, the actual interaction with people is treated slightly different, given that more work may need to be done in order to determine the appropriate action. The list of available interactions depends on the resources available within the active space containing the user. The simplest action would be to have the camera start tracking that person through the active space, or active spaces as desired. As stated previously, we have already successfully implemented this feature. However, since we are trying to use the location information to facilitate remote interaction, we also provide several other options. We are currently working on their integration with Active Interaction. First, we automatically obtain general information about the person and his preferred methods for contact. If the person has defined

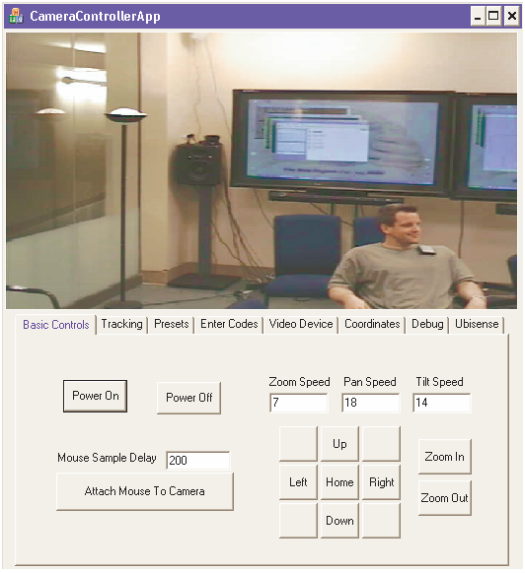

Figure 3. CameraControllerApp screenshot. Note, the user could interact with the light, displays, speakers, or the person in the figure.

preferences for forms of contact, we can automatically select the appropriate device for contacting and communicating with the person. If the person has not specified a preferred form of contact, we can manually select the contact method from a list of available methods, such as a cell phone, email, text message, or Voice over IP (VoIP). This list would be dynamically updated depending on the specific active space and its corresponding resources, including any resources the people in the room introduce.

Bresler [14] et al. is currently developing a framework for Voice over IP within the Gaia infrastructure. By integrating this service within our framework, we can automatically communicate with the remote person through whatever audio devices are available within the active space, including speakers, cell phones, or PDAs.

6 Conclusion

Integrating the video feeds from multiple cameras with accurate location sensing, we have developed Active Interaction, an innovative application for remote interaction with static and dynamic objects within active spaces. By interacting directly with

the objects as they are seen through the video feeds, and using the MiddleWhere infrastructure, we are able to seamlessly and dynamically add objects and immediately detect them without generating any avatars or visual representations of the objects. No additional work, such as avatar or model generation, is needed for the user to interact with new objects since they are automatically discovered by the infrastructure and added to the location database. Additionally, the interface is more natural since users do not need to mentally translate avatars and tags into objects – they see the objects themselves. This provides a useful, seamless interface for remote interaction with ubiquitous environments and their resources.

7 Future Work

Currently, a prototype of the Active Interaction framework is implemented. This prototype allows interaction with limited devices and services only, and serves as a proof of concept. It allows interaction with any objects discovered by Gaia's semantic discovery service. However, we still need to provide a mechanism for alerting the user as to which objects support interaction. Additionally, we need to improve the mechanism for determining appropriate forms of interaction with people located within the space.

The interaction with users in the space through VoIP is not fully complete at this time, as the VoIP component of Gaia is still under development.

Additionally, we need to provide a usable interface for the different interactive methods, such as writing text, speaking, broadcast video, or other user-defined interactions, and gauge the effectiveness and ease of these interfaces by conducting usability studies.

Finally, we plan to provide a remote interface that would enable keyboard and mouse redirection to the remote machines or services using Clicky [6]. This would allow live remote interaction with applications that do not implement IDL interfaces.

References

[1] Ranganathan, A., Al-Muhtadi, J., Chetan, S., Campbell, R., Mickunas, M.D.: MiddleWhere: A Middleware for Location Awareness in Ubiquitous Computing Applications. Presented at 5th International Middleware Conference (Middleware 2004) (2004)

[2] Roman, M., Hess, C.K., Cerqueira, R., Campbell, R.H., Nahrstedt, K., M.: Gaia: A Middleware Infrastructure to Enable Active Spaces. Pervasive Computing Magazine, vol. 1 (2002) 74-83

[3] Biehl, J.T., Bailey, B.P.: ARIS: An Interface for Application Relocation in an Interactive Space. Proceedings of Graphics Interface (2004) 107-116

[4] Roman, M., Ho, H., Campbell, R., "Application Mobility in Active spaces," presented at 1st International Conference on Mobile and Ubiquitous Multimedia, Oulu, Finland, 2002.

[5] Glasberg, M. S.: ActivePresentation: A Software Infrastructure for Presentation Control in ActiveSpaces. In Proceedings of the Joint WebMedia/LA-Web 2004 Conference, Ribeirão Preto, São Paulo, Brazil (2004), Portuguese only.

[6] Andrews, C., Sampemane, G., Weiler, A., Campbell, R.: "Clicky: User-centric input for Active Spaces". UIUC Technical Report: UIUCDCS-R-2004-2469, UILU-ENG-2004-1770.

[7] Hosoya, E., Kitabata, M., Sato, H., Harada, I, Nojima, H., Morisawa, F., Mutoh, S., Onozawa, A.: "A Mirror Metaphor Interaction System: Touching Remote Real Objects in an Augmented Reality Environment." Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2003).

[8] Addlesee, M., Curwen, R., Hodges, S., Newman, J., Steggles, P., Ward, A., Hopper, A.: "Implementing a sentient computing system." IEEE Computer 34 (2001) 50-56

[9] Butz, A., Hollerer, T., Feiner, S., MacIntyre, B., Beshers, C.: "Enveloping Users and Computers In a Collaborative 3D Augmented Reality." International Workshop on Augmented Reality (IWAR), (October, 1999).

[10] Harle, R., Hopper, A.: "Dynamic World Models from Ray-tracing." Proceedings of the Second International Conference on Pervasive Computing and Communications, IEEE, March 2004.

[11] Harle, R., Ward, A., Hopper, A.: "Single Reflection Spatial Voting: A Novel Method for Discovering Reflective Surfaces Using Indoor Positioning Systems." Proceedings of the First International Conference on Mobile Systems, Applications, and Services (MobiSys 2003), ACM, May 2003.

[12] UbiSense, "Local position system and sentient computing." http://www.ubisense.net/.

[13] Ranganathan, A., Chetan, S., Al-Muhtadi, J., Campbell, R.H., Mickunas, M.D.: "Olympus: A High-Level Programming Model for Pervasive Computing Environments," presented at IEEE International Conference on Pervasive Computing.

[14] Anwar, Z., Bresler, J., Chan, E., Campbell, R.H., "A VoIP Communication System for Siebel Center, UIUC Tech report, 2005.
[15] Ierusalimschy, R., Henrique de Figueiredo, L., Filho, W.C.: "Lua - an extensible extension language," Software Practice & Experience, volume 26, no. 6, 1996.