Supporting Interoperability and Presence Awareness in Collaborative Mixed Reality Environments

Oyewole Oyekoya[*] Ran Stone[†] William Steptoe[*] Laith Alkurdi[‡] Stefan Klare[‡]
Angelika Peer[‡] Tim Weyrich[*] Benjamin Cohen[†] Franco Tecchia[§] Anthony Steed[*]

Abstract

In the BEAMING project we have been extending the scope of collaborative mixed reality to include the representation of users in multiple modalities, including augmented reality, situated displays and robots. A single user (a visitor) uses a high-end virtual reality system (the transporter) to be virtually teleported to a real remote location (the destination). The visitor may be tracked in several ways including emotion and motion capture. We reconstruct the destination and the people within it (the locals). In achieving this scenario, BEAMING has integrated many heterogeneous systems. In this paper, we describe the design and key implementation choices in the Beaming Scene Service (BSS), which allows the various processes to coordinate their behaviour. The core of the system is a light-weight shared object repository that allows loose coupling between processes with very different requirements (e.g. embedded control systems through to mobile apps). The system was also extended to support the notion of presence awareness. We demonstrate two complex applications built with the BSS.

CR Categories: C.2.4 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities C.2.4 [Computer Communication Networks]: Distributed systems—Client/server D.2.12 [Software Engineering]: Interoperability—Distributed objects

Keywords: telepresence, interoperability, presence, mixed reality, telerobotics, virtual reality

1 Introduction

Telepresence is the experience involved in the displacement of a remote operator's perception into a virtual environment, or the illusion of being in a real environment other than true location. This means of shifting perception is achieved using various technologies that need to perform joint execution of tasks simultaneously. The availability of low-cost sensors, tracking technologies and robotic devices for achieving telepresence has increased in recent years. Providing interoperability allows these heterogeneous systems from multiple vendors to readily work together and exchange data.

The BEAMING (Being in Augmented Multimodal Naturally Networked Gatherings; www.beaming-eu.org) project aims to transport people (visitors) from one physical place in the world to a destination to enable interaction with local people and objects there [Steed et al. 2012]. This is achieved by capturing the visitor's actions, physiological and emotional state using a *transporter* to capture data that is transmitted across the Internet to the destination. Simultaneously, the destination is captured and streamed to the visitor's location (transporter) as a reconstructed virtual environment in real time (Figure 1). A transporter is a high-end virtual reality (VR) system that is used to capture the visitor's actions and display the remote space while the destination is a real space populated with *local* people and a robot or VR display. Thus, the actions and state of the visitor at the destination site is visualised by the local people as a physical robot and/or avatar on a VR display. The project combines technologies from networking, computer vision, computer graphics, virtual reality, audio, haptics, robotics and user interface together.

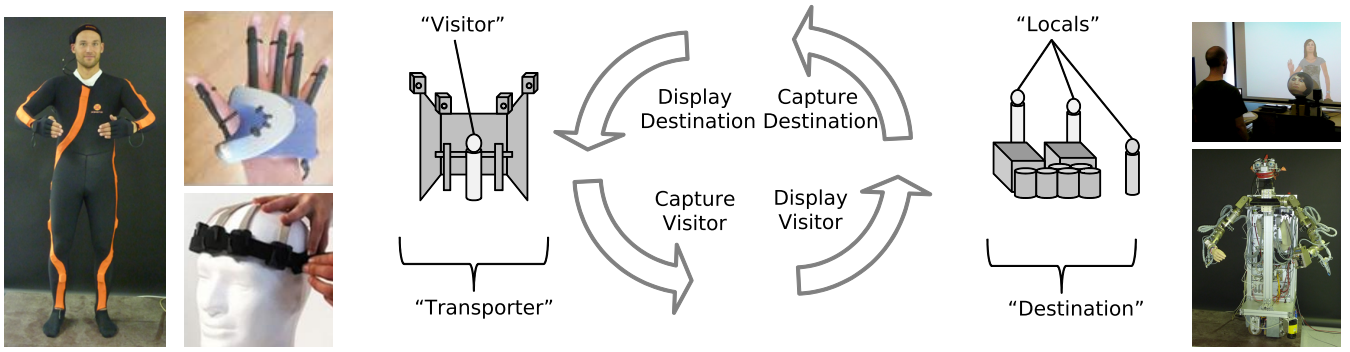
We aim to give a sense of spatial presence within the destination for visitors, hence the visual display at the transporter must be an immersive display [Steptoe et al. 2012; Normand et al. 2012] such as a head-mounted display (HMD) or a immersive projection system (e.g. CAVE displays [Cruz-Neira et al. 1992]). As it strives for social symmetry, the system must provide similar sensory experiences, particularly the dominant visual mode, to the visitor. To fully capture a visitor, we use multiple devices such as motion capture suit, hand tracker and EEG (Figures 1a and b). Visitor tracking is not limited to the body posture and motions but includes measuring neuro-physiological signals and mapping emotions onto remote avatars and robots.

We aim to give a sense of copresence to the locals so visitors are represented with a virtual or physical embodiment at the destination. Locals need no visual mediation to perceive the destination as being realistic and spatial because they perceive the actual physical location. However, stimuli representing the destination must be captured and transmitted in real time to the transporter. These can include static room models [Pece et al. 2013], point cloud scans [Pece et al. 2011] or audio [Olesen et al.]. Physical embodiments can be in the form of robots [Buss et al. 2009] or mobile telepresence systems [Tsui et al. 2011] while virtual embodiments are presented on displays at the destination such as desktop monitors, projections and spherical displays [Oyekoya et al. 2012], as depicted in Figure 1d.

Data exchange and communication between the visitor transporter and the destination is a critical aspect in the project. Treatment of all these data is not a trivial matter since different kinds of data need to be captured, streamed from the transporter and reproduced at the destination synchronously. The complexity is characterised by the diversity of devices and software interfaces which includes variations in networking, input and display capabilities. The complexity of data communication technologies behind networked games and virtual environments has been explored extensively in [Steed and Oliveira 2009].

To address this problem, this paper presents the design and implementation of the Beaming Scene Service (BSS), which provided interoperability for heterogeneous systems in the BEAMING project. The system extends previous work by integrating the concept of presence awareness in mixed reality applications. We also present the deployment of the proposed architecture to support mixed reality systems in two application scenarios. We focus on data trans-

[*]University College London. o.oyekoya@ucl.ac.uk
[†]IBM Research, Haifa Israel
[‡]Technische Universitat Munchen, Germany
[§]Scuola Superiore Sant'Anna, Pisa Italy



(a) *Body* **(b)** *Hand, EEG* **(c)** *BEAMING recreates a real environment (the destination) populated with people (locals) while a remote person (visitor) visits the the recreated model via a transporter.* **(d)** *Displays and Robot*

Figure 1: *High level data flow of the BSS between the transporter and destination sites.*

fer, session management, data representation and client applications supported in the project.

2 Related Work

Several research efforts have been done on surveying middleware systems for various usage such as robotics [Mohamed et al. 2009], sensor networks [Henricksen and Robinson 2006; Molla and Ahamed 2006; Wang et al. 2008] and mobile computing [Gaddah and Kunz 2003]. Papagiannakis et al. [Papagiannakis et al. 2008] provide a survey of network technologies for mobile augmented reality systems. Mixed reality systems are a subset of middleware systems focussing on real-time synchronous communication across a range of devices. Mixed-reality systems include 2D or 3D models of space and tend to focus on visual elements of the scene.

One common abstraction for mixed-reality and distributed multimedia systems is an abstract data service such as a tuple-space [Davies et al. 1997; Gelernter 1985; Wyckoff et al. 1998]. This is manifest in the EQUIP system [Greenhalgh et al. 2001; Greenhalgh 2002], which was developed to support the merging of physical and virtual environments as part of the Equator project. A number of input and output modules were created within the EQUIP framework, which included publishing input from various devices, such as GPS receiver, tilt sensor, mouse/keyboard, PC joystick, 2D GUI input simulator (e.g. to simulate a joystick on a handheld device), RF-ID tag reader and video-based light tracking. The system also interfaced to the MASSIVE-3 collaborative virtual environment [Greenhalgh et al. 2000]. The system incorporated type definition modules for renderable objects, audio and video session descriptions. 3D rendering objects included simple solid geometries, abstract data visualisations, embedded views of MASSIVE-3 worlds, and georeferenced objects and positions (e.g. globally located using GPS). With an abstract data service approach there is a tension between flexibility of representation and the speed of updates. Specifically when dealing with mixed-reality systems, if we have descriptions of scenes as geometry, point clouds or video, these representations are likely to be too bulky to store in the tuple-space and are specific to only some of the collaborating processes.

Tracking is an important part of any Virtual or Augmented Reality (AR) application. Multimodal interaction requires multiple tracking systems to fully capture and replicate a visitor's actions at a destination. Reitmayr and Schmalstieg [Reitmayr and Schmalstieg 2005] presented the OpenTracker framework to deal with the problem of integrating tracking data from various devices for AR systems. The authors presented an implementation where the data type processed is a fixed structure tailored towards specific application and hope to extend it to different data structures in future work. Our system is not just concerned with tracking data but also the management of high-bandwidth reconstruction data. However, we deal with similar issues identified in the design requirements such as device abstraction, support for complex configurations and simplicity of integration.

Chao and Wang presented a framework for exchanging Computer Aided Drawing/Manufacturing data between heterogeneous systems [Chao and Wang 2001]. The framework focussed on the provision of data exchange mechanisms that would support collaborative design and manufacturing at geographically dispersed sites. The framework utilised a client-server architecture where clients request for product data (AutoCAD files) while the server searches for and translates the data into a standard format. The framework supported data browsing, data search, multi-database connections, automatic format recognition and translation. The work highlighted the importance of standardisation for the exchange of product data. Adopting a standard format is crucial for interoperability, as we highlighted in our work. Managing interoperability among heterogeneous systems has also been explored in smart home environments [Perumal et al. 2008]. Smart spaces can potentially be extended with avatar (through Augmented Reality, AR on smartphones) and robotic embodiments. Mobile devices such as robots and smartphones add a mobility feature that can sense the world around it and map the space. As such, they are prone to intermittent network connectivity, which can have an effect on the availability. Thus, we proposed managing the availability with the concept of presence awareness, as discussed below.

Most of the systems we identified tended to share location and event data but do not address the issue of presence awareness management. The concept of presence indication is seen in instant messaging applications where users' availability is displayed to friends e.g. "available", "unavailable", "busy", "away", "do not disturb", etc. The use of presence indicators for monitoring a visitor or robot's availability opens up a range of presence applications. Boyer et al. [Boyer et al. 1998] investigated tools to allow a distributed team to work better based on presence awareness of each other. Boyer et al. identified presence cues that can be tracked in determining presence status such as computer and telephony activities. In our paper, we apply this concept to handheld mobile devices. Recent advances have seen an increase in the use of mobile handheld devices with wireless networking capabilities. Client mobility and intermittent connectivity are inherent with such devices, as noted

by Kulkarni et al. [Kulkarni et al. 2003]. Mobile clients can move from one location to another and as such are prone to frequent disconnections and reconnections. Disconnections can occur due to poor network coverage at a particular location or due to the mobile device powering down to save energy. In the Equip system, Greenhalgh et al [Greenhalgh et al. 2001] reported that they handle intermittently connected wireless devices through a proxy service to send an receive EQUIP data items. They developed an additional lightweight ASCII protocol to communicate between the proxy and low resource devices as these devices were incapable of handling numerous events at a reasonable rate. We use XMPP (Extensible Messaging and Presence Protocol)[1] to deal with this issue, as well as awareness management, which is lacking in related works. For example, we handle intermittent connections by using 'unavailable' or 'sleep' mode to inform the service that the mobile client has not been disconnected.

3 Requirements

A key step towards software and hardware integration in BEAMING is the data communication and exchange between processes. One of the requirements is the interfacing of heterogeneous systems i.e. sensors on Linux, iOS, Windows, OSX, etc. Interfaces range from external services (e.g. Second Life) to specific hardware (e.g. robots and handheld mobile devices). Based on discussions amongst project developers, we identified several requirements:

Heterogeneity: This is a key requirement, as capture processes and rendering processes may have different operating systems and programming languages.

Scalability: There may be several visitors, avatars/robots, hence arbitrary numbers of rendering and capture processes. Most of the data transmission is from capture processes to rendering processes, hence data capture is a pre-requisite for rendering processes. At the visitor's site (such as the CAVE), which renders the destination, multiple rendering processes are required due to the multiple output modes, including visual, aural, and haptic. This is similar to the destination (i.e. the rendering of the visitor), which also requires multiple rendering processes to cope with the visual and aural modes.

Bandwidth Efficiency: There are three different types of data flow that were identified.

- High-bandwidth & large-size data: These are point-clouds, depth map, video, audio and haptic streams. These data require a separate pipeline from the destination capture process to rendering process using peer to peer protocols.
- Low-bandwidth & small-size data: These are motion and physiological tracking data requiring dynamic (fast and frequent) updates during sessions. Configuration information about large-size data are required to be published (slow and infrequent updates) e.g. current sending rates, available bandwidth, number of triangles/points, etc.
- Zero-bandwidth & large-size data: These are data that are registered at session initiation. For example, a model of a destination could be pre-computed (static 3D scan) or streamed in real-time (dynamic capture). The 3D scan may be hosted on a HTTP URL while the real-time capture is hosted on an IP address and port. The destination capture process publishes the URIs for the HTTP and custom protocols used, so that capture processes would subscribe to that IP/URL to load the static/dynamic data.

[1] http://www.xmpp.org/

Sending of high bandwidth data poses a difficult challenge, as it could slow down a session. Hence we proposed that any capture process that wants to send high-bandwidth data registers its configurations for other clients on the main server and additionally has a direct pipeline (point to point connections) that does not clog up the lower-bandwidth data.

Support the Beam-enabling process: A destination needs to be prepared for capture and reconstruction (Beam-enabling). A main goal of BEAMING is Beam-enabling a destination within a maximum duration of 30 minutes. This requires support for self-configuration and semi-automatic calibration. Hence, destination capture processes require data from the scene service to support Beam-enabling.

Session Management: A Beaming session takes place at the destination. Session management deals with session initiation, management and termination.

- Processes need to be aware of a Beaming session and the Beam-enabled location (destination).
- Locals should be aware when the visitor representation is active.
- Users should be able to initiate and terminate a Beaming session.
- Users should be able to record and replay a session.

Quality of Service: The system should enable different granularity of the Quality of Service (QoS) control. Processes should be able to set packet priority, reliability and frequency of data serialization.

Scene Graph: It must be able to support a hierarchical data structure of nodes that is commonly used in games and 3D applications to represent entities or objects in a scene. The parent-child relationship of all created nodes must be maintained in a graph/tree structure across all client processes. The concept of implementing scene graphs across networked virtual environments has been employed in previous works [Tramberend 1999; Naef et al. 2003; Lake et al. 2010; Hesina et al. 1999].

Presence Awareness: Most clients are assumed to be processes that are permanently connected to a Beaming session. However, intermittent connectivity and mobility characterises handheld mobile devices. As such, mobile devices were connected to the system using XMPP presence management. The integration of XMPP also demonstrates the extensibility of the system.

4 Distribution Architecture

The architecture of the BSS is shown in Figure 2. A client-server replicated shared object approach is adopted. BSS exchanges data over the Internet using IP protocols through a centralized server, rather than a dedicated network. The Beaming Scene Server is responsible for coordinating the flow of data and world state between clients. The central server acts as the point of rendezvous and timing, easy to set up between sites (no peer networks or multicast). Every process in a Beaming session (clients) receives a copy of those shared objects when they connect to the server. Every process can edit a shared object, changes are transparently copied to all other processes. Each process can publish information simply by creating a shared object. All slow or limited volume data is published on the BSS providing single transport and easy logging while all time-critical and high volume data use native peer to peer streams but configuration of those streams is stored on the BSS. Hence, one single point of contact contains all information about the scene.

BSS provides a cross-platform C library that can be integrated into the client process driving the software or hardware system, to facilitate communication with the server. A client process is also able to connect directly with any UDP client to exchange data. New client processes can be added dynamically at run-time, which makes the framework very adaptable and flexible. Clients add nodes to the world via the Beaming Scene Server which ensures that all the other processes are informed. A logger captures and stores all events being distributed for later analysis. A Beaming Scene Session would support one destination and a set of visitors.

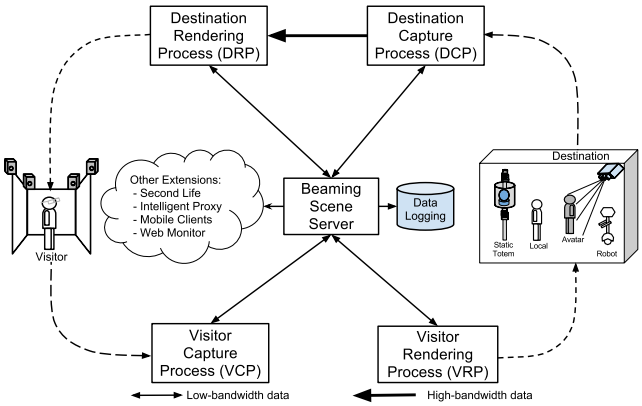


Figure 2: *Architecture of the Beaming Scene Service*

4.1 Data Transfer

Data transmission is managed by RakNet[2] middleware, a cross-platform C++ game network engine that provides UDP transport. A lower-level framework powers higher level features such as game object replication. A process can connect to the library and publish information by creating an object/node which is shared and replicated amongst client processes. Each process will own all of the objects it creates, and will perform local edits/updates on the properties. These properties are serialised to keep the databases of all processes consistent.

UDP transport was chosen to enable more control over the reliability of low-bandwidth and small-size data. For example, a resend is inappropriate for avatar/robot data.

4.2 Data Representation

Data types were created to support the creation of objects/nodes. These replicated objects include avatars, emotion recognition, facial expressions, video, 3D objects, point clouds, etc. The data contains representations of the shared mixed-reality as a user space (i.e. what it looks like) and details of the processes and data streams that can be used by other processes to render the environment. The idea is that any client process should be able to connect onto the BSS and extract information about an entity (e.g. avatar, robot, room or cameras).

There are two obvious classes of data: (i) tracked human, typically represented by an avatar or robot; and (ii) reconstructed environment typically represented by video, audio, point cloud or objects.

[2]http://www.jenkinssoftware.com/

4.2.1 Tracked Human

These data types typically represent the user being captured. Users' activities are not only captured at the transporter (as a visitor) but also at the destination in less detail (as a local). Tracked humans can include locals if a capture process separates the locals from the background and tracks the locals (e.g. using the Kinect). Users are typically represented by an avatar in the recreated virtual environment in order to send skeleton data, however, higher level data also need to be captured and streamed separately such as tactile, emotion and facial gestures. Therefore, we created five data types to represent a tracked human.

Avatar Skeleton: In order to animate an avatar, we use a skeleton hierarchy made up of 123 joints (or bones) per avatar. Each joint has an associated position and orientation (in quaternion angles). An avatar can represent a visitor or a local. The pelvis joint is relative to world coordinates while the others are relative to the pelvis/parent joint. The avatar's 3D positions and orientations are represented by seven floats per node (three translations and 4 quaternions) and stored internally as 123 objects arranged in a scene graph structure. The parent-child relationship is maintained, thus allowing processes to poll for data in local or world co-ordinates. Thus, avatar joints (e.g. head, hand, torso) can be extracted in world-coordinates. By adopting this as a standard, developers are able to transform data (Figure 3) from one format (e.g. Microsoft Kinect® or Xsens [Roetenberg et al. 2009]) to the avatar skeleton hierarchy.

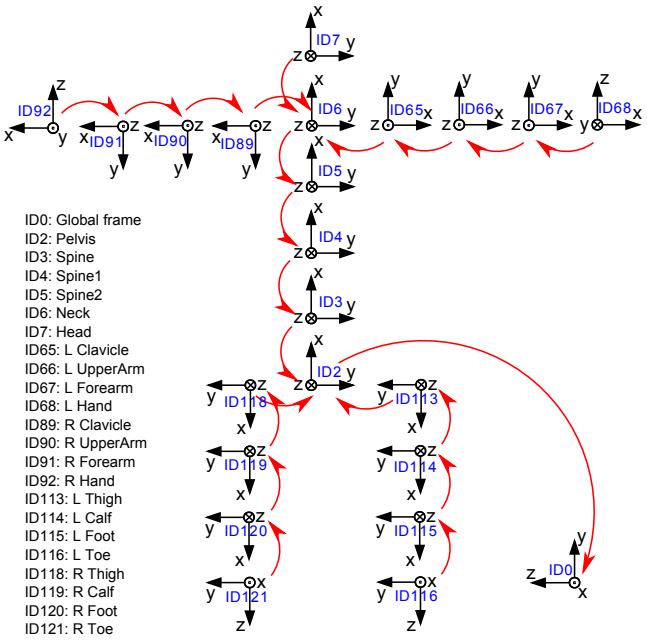


Figure 3: *The drawing shows the bone-fixed skeleton frames. The visitor is looking at the viewer and is performing a T-pose with the palm facing to the ground. Positions and orientations are given in relative coordinates, whereas relations are marked by red arrows.*

Facial Expression: Facial expressions require different data depending on avatar or robot representation, therefore this data type needs higher-level abstraction than the joint rotations used for driving the avatar skeleton. Thus, we created float/boolean attributes, each corresponding to a particular expression or mouth shape.

Affective State (Emotion): Real time affective state recognition from EEG (Electroencephalography), GSR (Galvanic Skin Response) & HRV (Heart Rate Variability) requires three double vari-

ables for valence, arousal and a miscellaneous variable for one other affect state.

Tactile Feedback: The use of a touch device for hand requires variables to exchange duration of vibration, intensity of vibration and temperature.

Generic Data: As the needs of different systems might change over time, these data type allows sending of generic data (e.g. pointer to a structure in memory), sent as a void pointer with a maximum size of 1024 Bytes.

4.2.2 Reconstructed Environment

These data types represent reconstructed data from the destination, typically an indoor area e.g. room. All replicated objects in this category have a host and port of the streaming points for dynamic scenes and a http URL for static scenes, in addition to the data types specified below.

3D Objects: This data type represents dynamic objects in the scenes such as meshes, point cloud data, etc. The object's 3D position and orientation are specified by the creating process.

Video Source: Although video data is sent through separate network channels, information about the video source is published on the BSS. This includes camera calibration matrices and dimensions of the video being streamed.

Audio Source: Although audio data is also sent through separate network channels, information about the audio server is published on the BSS.

Point Cloud: Although point cloud data is sent through separate network channels, information about the source of the point cloud is published on the BSS. This includes the level of quality of the point cloud and the number of points/triangles.

4.3 Session Management

1. **Joining a session**: All client processes are expected to load the library that allows connection as a client in the client-server architecture. A function, *startclient* allows the client process to establish connection with the server. Processes can specify a name, type (e.g. VISITOR, LOCAL, SPECTATOR, ROBOT, HAPTIC_DEVICE, etc), configuration file, a flag to denote whether it is a unidirectional viewer (read access) or wants to create and update objects (read\write access). Processes are also able to change the packet priority, reliability and serialization frequency.
2. **Monitoring a session**: A function, *check* continuously sends outgoing packets, receives incoming packets and checks for new connections to the server. This includes regular ping requests by the server to monitor lost connections and attempting reconnections.
3. **Creating shared data**: Client processes with write access are able to create shared objects using any data type, e.g. avatar data type relating to avatar joints in the XML avatar format. Processes can add multiple objects and set the properties of each object.
4. **Writing data to a session**: Client processes are able to update each object that it has created.
5. **Reading data from a session**: Client processes are able to read data from other remotely connected client processes.
6. **Leaving a Session**: A client process disconnects by removing all owned objects and publishing the event to the server. Other client processes are notified of this disconnection, and destroy any locally-stored data structures pertaining to the disconnected client. Additionally, if a client process does not respond in a timely manner, eventually the server process will delete the objects that the client created and informs others to do the same.

4.4 Clients

4.4.1 Client Manager

The Client Manager (Figure 4) is a platform-independent application built with C++ and Qt[3] that can run as a stand-alone executable, or as a dynamically linked library. It provides a comprehensive solution for client-side functionality, including connection and initialisation to the BSS, synchronous use of multiple tracking devices, logging and replay of prior sessions. It follows a modular design and is easy to extend. It encapsulates many aspects of client-side functionality, including connection management, tracking, logging, and replay. In spectator mode, it may be used in a basic way, by connecting to a Beaming Scene Server, and simply observing the unfolding activity of other clients. However, it may also be used to achieve all aspects of visitor tracking and data transmission in visitor mode. It supports a number of tracking devices used for capture of elements of a visitor's activity during a Beaming session, which range from motion capture systems to trackers, from eye trackers to physiological state monitors, etc. It is able to interoperate with renderers such as an OpenGL interface, eXtreme Virtual Reality (XVR) software [Tecchia et al. 2010] or the Unity game engine[4], to visualise avatar animations (based on tracking data) and virtual environments.

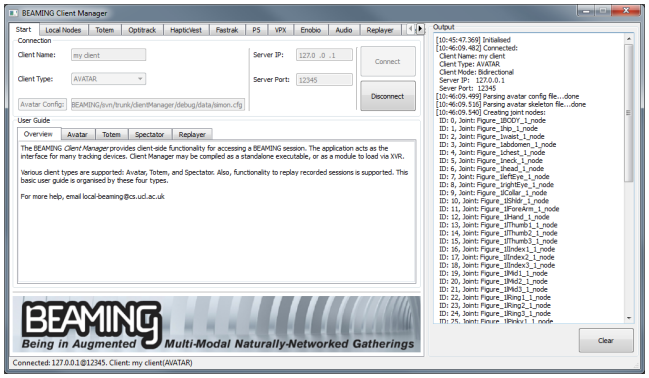


Figure 4: *Client Manager GUI*

4.4.2 XVR

The XVR framework [Tecchia et al. 2010] is a complete development environment explicitly designed to support the typical concepts of VR programming. XVR allows for the development and graphical rendering of CAVE-based or HMD applications [Mortensen et al. 2008; Carrozzino et al. 2008]. It is organized around a virtual machine and applications are developed using a dedicated scripting language (S3D), whose constructs and commands are targeted to VR, giving developers the means to deal with 3D animation, positional sound effects, audio and video streaming, and user interaction. The behaviour of the application is specified using a custom scripting language, providing the basic features and exposing VR-related methods available as functions or classes. The

[3] http://www.qt-project.org/
[4] http://www.unity3d.com/

XVR framework has a number of built-in features, often sufficient to develop the most common types of application. Like in any typical scripting language, basic functions offer simple control over variables, files, strings, and math manipulation. Beyond these basic features, the framework offers lots of predefined classes and functions, as well as the ability to introduce user defined data structures and classes. Extensibility via plug-ins is also available to accommodate for more specific and advanced functionalities implemented via external libraries. It also adopts a shared memory mechanism in order to pass data between programs e.g. multiple instances of the client manager.

4.4.3 Web Monitoring

The web monitor was built using PHP and acts as a client which monitors the state of the system. It displays all connected clients and accompanying objects on the server using a web browser.

4.4.4 Third Party Client

A visitor, not on the main Beaming network, is able to view the session through a third party client, Second Life (SL)[5], which has a fairly comprehensive set of meta tools built in e.g. Flash and HTML5 support (i.e. including video). SL visitors can be represented in the destination and on the visitors' displays.

4.4.5 Intelligent Proxy

Intelligent proxy [Hasler et al. 2013] is an automated software agent on the BSS that can intercept and learn from AVATAR data streams. As such, it's able to represent the visitor when she or he is away or busy with other tasks.

4.4.6 Mobile Clients

The increasing use of mobile devices could potentially enable users to participate in outdoor Beaming sessions. However, client mobility and intermittent connectivity are inherent issues with mobile clients. As such, an extension to the BSS (Figure 5) was proposed to enable mobile clients to set presence information based on XMPP (Extensible Messaging and Presence Protocol). Presence information is a status indicator that conveys ability and willingness of a user to communicate. XMPP provides us with the ability to set presence information (such as mood and away states) for a visitor or local. This system allows communication with mobile clients (handheld devices) using a portal/bridge. It uses the XMPP protocol to exchange messages and presence information in close to real time. Based on XML, it enables the near-real-time exchange of structured yet extensible data between any two or more network entities. The XMPP control layer is independent of the middleware that manages the data types. An XMPP server is used as a provider of the XMPP authentication, presence and group-chat services. The control messages are sent via the XMPP server (Openfire[6]) to the portal where the data is sent to other client processes on the BSS.

5 Applications

BSS has been used in three main applications: acting rehearsal, outdoor collaboration and teaching musical instrument. The acting rehearsal [Steptoe et al. 2012] describes the experience of setting up a distributed multimodal mixed reality system to support remote acting rehearsals between two remote actors and a director. We used a range of capture and display devices to connect the people

[5]http://www.secondlife.com/

[6]http://www.igniterealtime.org/projects/openfire/

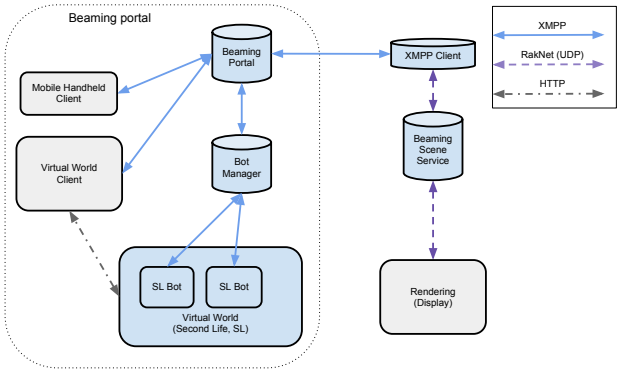


Figure 5: *Architecture of the Beaming Portal (XMPP-Based)*

located in three technologically-distinct locations through the BSS. We describe the teaching and outdoor applications in more detail below.

5.1 Teaching Application

This application is concerned with teaching a musical instrument, where the teacher (visitor) visits the destination where the students (locals) are located. The teacher located in one room is physically embodied within a robot avatar located in another room to interact directly with the student and the teaching equipment (Figure 6). The teaching application follows a multimodal teaching scheme in which the teacher delivers knowledge of a musical instrument and how to use it. While this application involves one student and one teacher, it is in principle scalable to multiple teachers and a classroom.

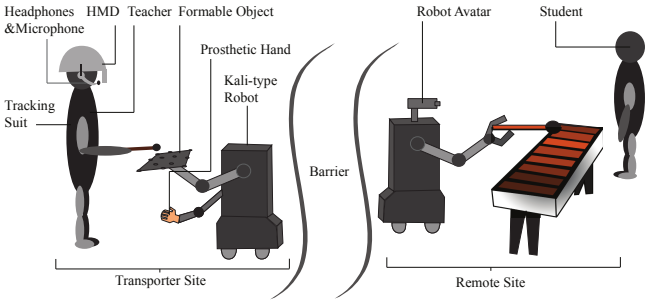


Figure 6: *Overview on teaching application using robot avatar and Kali-type robot*

The teacher is beamed to the remote site in order to interact with the student and the teaching equipment. This remote site can be the student's house, a teaching facility at a university or something similar. The scene starts with the teacher knocking on the door of the room. The student opens the door, greets the teacher with a handshake and invites him/her to step into the room with the help of his/her robot avatar. The teacher starts the lesson by giving historical information about the musical instrument, how it has been used and the physics that generates the sounds. The teacher is able to play notes on the instrument while being embodied in the robot. The teacher plays a note and the student repeats it. After a while the teacher and student are able to play together on the xylophone.

Another mode of operation, the proxy mode is activated if the teacher has to stop teaching and step out for some time. Within this

mode, a proxy software takes over the control of the robot avatar and continues the discussion and demonstration. Questions asked by the student are memorized and provided to the teacher upon return. The proxy mode can also be important to perform an introduction that is usually static and repetitive in its nature. The proxy can also demonstrate predefined actions, for example a preselected melody to be played on the xylophone.

5.1.1 Description of the hardware and software

Figure 7 provides an overview of the data flow as well as interconnected software and hardware components in the realized application. The visitor is tracked using a tracking suit and controls the robot avatar. The visitor interacts with the Kali-type robot to get haptic feedback when interacting with different kind of objects that are located at the remote site. A Kinect array is mounted on the robot avatar to capture an online model of the remote environment. An offline background model is fused with the online data of moving persons and objects in order to display visual information to the visitor. Three dimensional audio from the xylophone as well as locals is transmitted to the visitor site. The voice of the visitor is played through a loudspeaker mounted on the robot avatar. The robot avatar has a second operating mode when the visitor is unreachable. In this case, the proxy takes over and a software program is in charge of the operation of the robot avatar.

The tracking suit is essential for the mapping of the visitor's full-body motion to robot avatar motions. Full body motion is tracked via a tracking suit (Xsens) and mapped to the adopted avatar hierarchy. To share this visitor motion data with other processes, it is written to the BSS. This data is read at the remote site, mapped from the avatar hierarchy to the robot kinematics and used to control robot motions. Human finger motions are captured by datagloves (Cyberglove) and are mapped to the robotic prosthetic hand to allow grasping of objects. The mapping of the locomotion is enhanced by model-based prediction as the current usage of a non-holonomic platform is problematic otherwise.

In order to provide haptic feedback to the visitor we are using the Kali-type robot concept. This implies that we can estimate the location of objects and locals beforehand and that we position the Kali-type robot at these positions in space to provide haptic feedback. In order to do so, the Kali-type robot is equipped with a formable object end-effector [Klare et al. 2013]. This formable object can simulate different surfaces corresponding to the objects being manipulated at the remote site. The formable object deforms its shape to simulate the region around the contact area on the objects being manipulated. A dummy hand is used to imitate the local while performing a handshake. In the demo we consider interactions with a door, the hand of a local as well as the xylophone. Currently, we are using a simplified intention recognition algorithm that activates an containment area around the object that is closest to the right human hand. More intelligent algorithms are planned for the future.

For the visual feedback, we fuse a static background of the room (generated from scans taken from the room beforehand) with live data captured by the Kinect array mounted on the robot avatar. These two streams are fused into one stream that is displayed on the head-mounted display. The hardware setup includes three Kinects distributed around the robot avatar neck to provide a wide viewing angle.

The audio feedback is set up as follows: The visitor as well as locals wear microphones. The visitor uses the headphones that are installed in the HMD. Audio feedback is rendered given the input of the tracking data. This tracking data provides the head position and orientation of the visitor as well as the position of the locals. This information is continuously updated on the BSS and read by different audio programs. The loudspeaker is mounted on the robot to provide audio feedback to the locals.

The proxy, as discussed earlier is able to take control of the robot avatar when the visitor is not available. It is able to process the audio stream, to talk and to change the posture of the robot. The posture of the visitor and locals is available to the proxy program, and can be accessed from the BSS to adapt the robot avatar's behaviour online.

5.2 Outdoor Application

This application is concerned with exploring user experience in an outdoor destination (quad). A quad (quadrangle) is an interesting destination because of the complexity of outdoor tracking in this area. The added mobility feature of handhelds allowed us to support a visitor or local in outdoor environments. For example, a visitor could use a smartphone to Beam-in to a session while being out of the office on a bus or a train. Another example would be to provide a user with a mobile device that will enable him to participate as a local in an outdoor destination. We demonstrate this by designing an outdoor task, where the local user explores an outdoor destination while collaborating with visitors in three different transporters.

Local: The outdoor user (local) uses a mobile handheld device to perform a scavenger hunt task within the quad area. The handheld device displays a 2D-map (top-down view, Figure 8c) of the quad to enable visualisation of the visitors' locations relative to the local's position and the relative direction they are facing.

Visitors: The three users are defined by their visitor environments (transporters) allowing the visitor to visualise the other participants' location and the direction that they are facing.

1. Visitor uses an immersive environment, CAVE and interacted with the reconstructed environment (Figure 8a) via a head and hand tracker with built-in joystick and buttons.

2. Visitor uses a desktop running the SL viewer (Figure 8b) and interacted via a mouse and keyboard.

3. Visitor uses a mobile hand-held device running a SL app (3D Lumiya viewer[7], as shown in Figure 8d) and interacted via the touchscreen functionality of the smartphone to control the camera and walking direction.

The three visitors are not present in the quad but they can see the locations of the boxes on their respective displays.

Task Objectives: Scavenger hunt is a game where participants seek to gather a list of items (boxes with QR codes) spread around the quad. Each task will be composed of 10 addition problems that the participants would need to solve i.e. 10 boxes. The boxes are located in the physical quad area, and are represented in the same exact locations as objects in the visitors' environments. Each box would be located in a different location within the quad. To ensure that participants would have to collaborate to solve the problems, the addition problem was split into four numbers e.g. 123+7+50+10 was split into 123, 7, 50 and 10. The participants would have to reach that location in order to activate their part of the problem and get the location of the next problem.

By gathering qualitative data (through questionnaires) and quantitative performance data on the overall use of the mixed-reality system, the application aims to measure the level of copresence from the perspective of the local i.e. the sense of being in the quad with the others. Also, as all transporters receive the same data about

[7] http://www.lumiyaviewer.com/

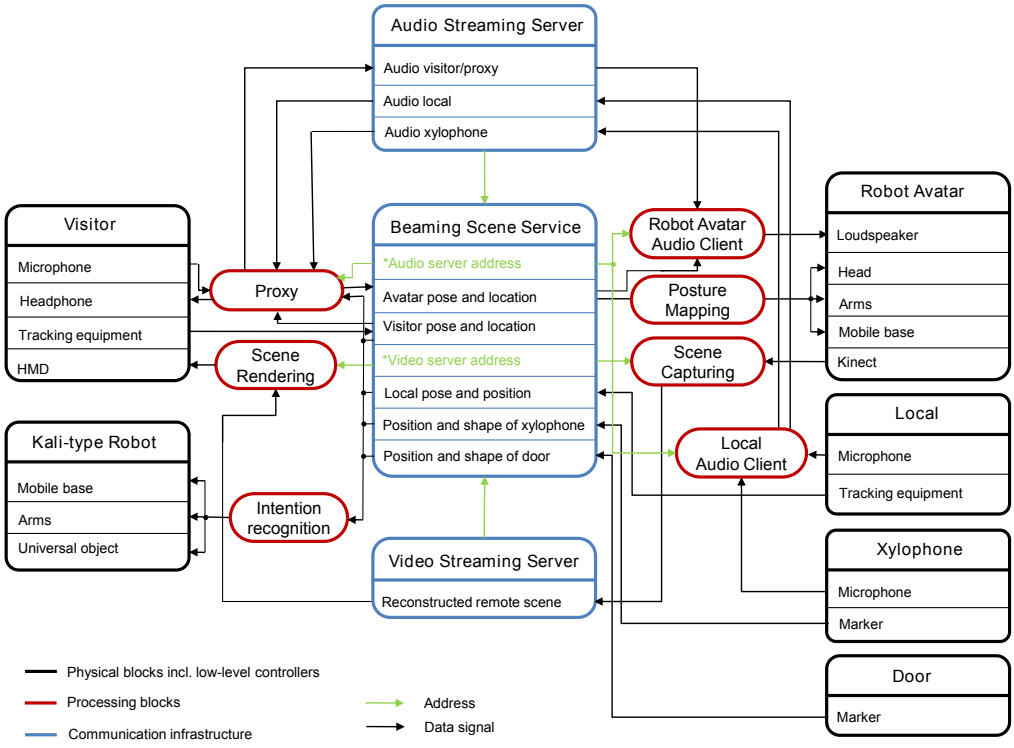


Figure 7: *Architecture of the Teaching Application*

the local and the destination but represent it in different ways, each visitor would have a different level of presence.

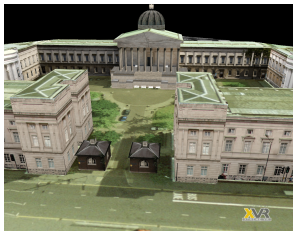
We also aimed to demonstrate how visitors who don't have access to immersive technologies (such as CAVE and head mounted displays) could use a free 3D virtual world client (e.g. SL) or mobile apps (e.g. 2D-map Client or 3D Lumiya viewer, as shown in Figures 8c and 8d respectively). The choice of visitor environment could be dictated by cost and/or space constraints or limited by availability of equipment at the transporter.

5.2.1 Description of the hardware and software

We implemented an Android app for the local that displayed a 2D aerial photograph of the destination (quad) while both locals and visitors were represented as facial icons (Figure 8c) that conveyed their location and orientation. The local's position and orientation are globally located within the quad using GPS, which is converted into the standard coordinate system and transmitted to the transporters through the portal (Figure 5). This setting enables the local to walk around the quad while viewing and communicating with the visitors via the mobile device. As the mobile client only captures the local's position and orientation it does not produce a full skeleton dataset. Instead it only updates the pelvis joint (ID2, Figure 3), hence the local's avatar representation in the visitor's environments had limited or no body language.

The visitors see a reconstruction of the quad (Figure 8a) on their respective displays. Both locals and visitors were represented as native avatars in the CAVE and SL viewers. The CAVE visitor sees life-sized avatars of the other avatars whereas the desktop and mobile visitors sees SL avatars relative to the size of their respective display. Tracking data is collected from each interaction mode and sent to the BSS for each visitor.

We integrated the desktop virtual world (SL) and handheld clients



(a) *3D Model*

(b) *Second Life (Desktop)*

(c) *Mobile (2D Map)*

(d) *Mobile (Second Life)*

Figure 8: *Interfaces for the outdoor application.*

into the BSS using the XMPP architecture. The XMPP architecture also enabled us to extend the BSS in a variety of ways. Second Life supports a variety of animation states (such as 'laughing', 'typing' and 'away' states) which provides more information of the controlling user's presence states. Similarly, smartphones come with a growing set of embedded sensors [Lane et al. 2010], such as an accelerometer, digital compass, gyroscope, GPS, microphone, and camera, which can be indicative of a user's presence states. These states can be transmitted by the mobile and SL clients as XMPP messages to the BSS. More importantly, XMPP was used to handle intermittent connectivity of handheld mobiles on the BSS e.g. the

smartphone's sleep mode does not result in disconnections from the server and instead is set as 'unavailable' presence status.

The XMPP architecture also extended the content sharing and collaboration capabilities of the BSS. For example. the boxes placed in the quad are represented as objects on the BSS. The local uses the smartphone to scan the QR codes. Virtual replicas of the objects were placed in the visitors' virtual models. When all session participants, local and visitor, were within near proximity of the object, a shared image would overlay the objects on their respective screens. This enables users to collaborate in the mixed-reality task (scavenger hunt).

6 Conclusions and Future Work

In this paper, we have presented the BSS, which attempts to network and process very different systems together with loose coupling between devices. Key features of the BSS include support for expressive data types, scene graphs, heterogeneity, scalability and presence awareness. It mixes tuple-spaces for the main processes with a presence awareness mechanism for less-well connected processes. BSS also has the advantage of supporting diverse interaction and devices and is easy to integrate. Although the BSS doesn't have a lot of constraints, it is synchronous and timely whereas XMPP is near real-time, and asynchronous relative to the BSS. The applications demonstrate the system's flexibility of serving a wide range of devices that differ in their technological capabilities, from low-fidelity mobile clients and virtual worlds to robots and highly immersive systems. However, BSS has the disadvantage that it does not currently deal with adherence to common standards, as this requires closer collaboration between partners.

The applications allowed for an early recognition of integration issues and for an iterative refinement of the system architecture. There is no dedicated network connection between project partners, so all traffic flows over standard Internet connectivity, with overall latency between the nodes on the order of 15 20 milliseconds. The resulting network infrastructure allows us to stream audio, video and general data between processes, even if the end points are behind different private networks. Depending on the type of process connected to the BSS, we can choose to use all or only a subset of the streams. Hence syncronisation of data between the BSS and peer to peer streams needed to be catered for. Due to the general low latency of data transmission and despite the large amount of data exchanged, users in various locations apparently did not perceive latency effects during various demonstrations of the applications. We plan to conduct user studies to explore how users experience collaborative interactions and explore usability issues.

Furthermore, we are planning an experiment to explore presence and co-presence with the outdoor application and how presence awareness of mobile handheld users influences engagement. We also intend to extend the outdoor application so that locals with handheld mobiles can take pictures or stream videos on the move, which are then uploaded to (or streamed from) a http URL or online database. These details are stored on the BSS. Using the computed global position and orientation of the local, the picture (or frames) can be super-imposed on the reconstructed quad model in the visitor's environments, based on the GPS and gyroscope data from the local's smartphone. This concept is demonstrated in [Pece et al. 2013], where videos from a smartphone camera are inserted live into a panoramic image of a remote place. This will enable visitors to visualise and explore the destination from the local's perspective in real-time. We also intend to extend the collaboration capabilities of the BSS by integrating technologies that will enable users to annotate on and manipulate both physical and virtual objects at the destination.

Future implementations will extend the XMPP architecture so users could retrieve their ID from third party authentication services, such as their work place, Google or Facebook. The presence awareness concept could also apply to managing the availability of robots. We can envision a future where smart home environments include robot. Presence awareness could play a major part in managing the availability of these systems. At its simplest, a visitor's process would need to know if a robot is currently parked and available for use. However, this can get more complicated if one visitor needs to transfer teleoperation control from one robot to another, to temporarily leave a session (e.g. "away" mode) or to pass the control to another visitor.

Acknowledgements

The authors acknowledge the support of other project partners in the EU FP7 *BEAMING* project (contract no. 248620): Aalborg Universitet, Institut D'Investigacions Biomediques August Pi i Sunyer (IDIBAPS), Technion Israel Institute of Technology, Interdisciplinary Center Herzliya, Starlab Barcelona, Universitat de Barcelona and University of Sheffield. We also thank the associated research fellows and students.

References

BOYER, D. G., HANDEL, M. J., AND HERBSLEB, J. 1998. Virtual community prescence awareness. *ACM SIGGROUP Bulletin 19*, 3, 11–14.

BUSS, M., PEER, A., SCHAU, T., STEFANOV, N., UNTERHINNINGHOFEN, U., BEHRENDT, S., LEUPOLD, J., DURKOVIC, M., AND SARKIS, M. 2009. Development of a multi-modal multi-user telepresence and teleaction system. *The International Journal of Robotics Research*.

CARROZZINO, M., EVANGELISTA, C., SCUCCES, A., TECCHIA, F., TENNIRELLI, G., AND BERGAMASCO, M. 2008. The virtual museum of sculpture. In *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts*, ACM, 100–106.

CHAO, P., AND WANG, Y. 2001. A data exchange framework for networked cad/cam. *Computers in industry 44*, 2, 131–140.

CRUZ-NEIRA, C., SANDIN, D. J., DEFANTI, T. A., KENYON, R. V., AND HART, J. C. 1992. The cave: audio visual experience automatic virtual environment. *Commun. ACM 35*, 6 (June), 64–72.

DAVIES, N., WADE, S., FRIDAY, A., AND BLAIR, G. S. 1997. Limbo: A tuple space based platform for adaptive mobile applications.

GADDAH, A., AND KUNZ, T. 2003. A survey of middleware paradigms for mobile computing. *Techical Report, July*.

GELERNTER, D. 1985. Generative communication in linda. *ACM Transactions on Programming Languages and Systems (TOPLAS) 7*, 1, 80–112.

GREENHALGH, C., PURBRICK, J., AND SNOWDON, D. 2000. Inside massive-3: flexible support for data consistency and world structuring. In *Proceedings of the third international conference on Collaborative virtual environments*, ACM, 119–127.

GREENHALGH, C., IZADI, S., RODDEN, T., AND BENFORD, S. 2001. The EQUIP platform: Bringing together physical and virtual worlds. *Mixed Reality Laboratory-University of Nottingham-UK*.

GREENHALGH, C. 2002. Equip: a software platform for distributed interactive systems. *The Equator Project, University of Nottingham, Nottingham*.

HASLER, B. S., TUCHMAN, P., AND FRIEDMAN, D. 2013. Virtual research assistants: Replacing human interviewers by automated avatars in virtual worlds. *Computers in Human Behavior 29*, 4, 1608 – 1616.

HENRICKSEN, K., AND ROBINSON, R. 2006. A survey of middleware for sensor networks: state-of-the-art and future directions. In *Proceedings of the international workshop on Middleware for sensor networks*, ACM, 60–65.

HESINA, G., SCHMALSTIEG, D., FURHMANN, A., AND PURGATHOFER, W. 1999. Distributed open inventor: A practical approach to distributed 3d graphics. In *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, 74–81.

KLARE, S., FORSSILOW, D., AND PEER, A. 2013. Formable object-a new haptic interface for shape rendering. In *World Haptics Conference (WHC)*, IEEE.

KULKARNI, P., SHENOY, P. J., AND RAMAMRITHAM, K. 2003. Handling client mobility and intermittent connectivity in mobile web accesses. In *Proceedings of the 4th International Conference on Mobile Data Management*, Springer-Verlag, London, UK, UK, MDM '03, 401–407.

LAKE, D., BOWMAN, M., AND LIU, H. 2010. Distributed scene graph to enable thousands of interacting users in a virtual environment. In *Proceedings of the 9th Annual Workshop on Network and Systems Support for Games*, IEEE Press, 19.

LANE, N., MILUZZO, E., LU, H., PEEBLES, D., CHOUDHURY, T., AND CAMPBELL, A. 2010. A survey of mobile phone sensing. *Communications Magazine, IEEE 48*, 9, 140–150.

MOHAMED, N., AL-JAROODI, J., AND JAWHAR, I. 2009. A review of middleware for networked robots. *International Journal of Computer Science and Network Security 9*, 5, 139–148.

MOLLA, M. M., AND AHAMED, S. I. 2006. A survey of middleware for sensor network and challenges. In *Parallel Processing Workshops, 2006. ICPP 2006 Workshops. 2006 International Conference on*, IEEE, 6–pp.

MORTENSEN, J., YU, I., KHANNA, P., TECCHIA, F., SPANLANG, B., MARINO, G., AND SLATER, M. 2008. Real-time global illumination for vr applications. *Computer Graphics and Applications, IEEE 28*, 6, 56–64.

NAEF, M., LAMBORAY, E., STAADT, O., AND GROSS, M. 2003. The blue-c distributed scene graph. In *Proceedings of the workshop on Virtual environments 2003*, ACM, 125–133.

NORMAND, J.-M., SANCHEZ-VIVES, M. V., WAECHTER, C., GIANNOPOULOS, E., GROSSWINDHAGER, B., SPANLANG, B., GUGER, C., KLINKER, G., SRINIVASAN, M. A., AND SLATER, M. 2012. Beaming into the rat world: Enabling real-time interaction between rat and human each at their own scale. *PLoS ONE 7*, 10 (10), e48331.

OLESEN, S. K., MARKOVIĆ, M., MADSEN, E., HOFFMANN, P. F., AND HAMMERSHØI, D. 3d sound in the telepresence project beaming. In *Joint Baltic-Nordic Acoustics Meeting 2012*.

OYEKOYA, O., STEPTOE, W., AND STEED, A. 2012. Sphereavatar: a situated display to represent a remote collaborator. In *Proceedings of the 2012 ACM annual conference on Human Factors in Computing Systems*, ACM, 2551–2560.

PAPAGIANNAKIS, G., SINGH, G., AND MAGNENAT-THALMANN, N. 2008. A survey of mobile and wireless technologies for augmented reality systems. *Computer Animation and Virtual Worlds 19*, 1, 3–22.

PECE, F., KAUTZ, J., AND WEYRICH, T. 2011. Adapting standard video codecs for depth streaming. In *Proceedings of the 17th Eurographics conference on Virtual Environments & Third Joint Virtual Reality*, Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, EGVE - JVRC'11, 59–66.

PECE, F., STEPTOE, W., WANNER, F., JULIER, S., WEYRICH, T., KAUTZ, J., AND STEED, A. 2013. Panoinserts: mobile spatial teleconferencing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ACM, New York, NY, USA, CHI '13, 1319–1328.

PERUMAL, T., RAMLI, A. R., LEONG, C. Y., MANSOR, S., AND SAMSUDIN, K. 2008. Interoperability among heterogeneous systems in smart home environment. In *Signal Image Technology and Internet Based Systems, 2008. SITIS'08. IEEE International Conference on*, IEEE, 177–186.

REITMAYR, G., AND SCHMALSTIEG, D. 2005. OpenTracker — A Flexible Software Design for Three-Dimensional Interaction. *Virtual Reality 9*, 1 (December), 79–92.

ROETENBERG, D., LUINGE, H., AND SLYCKE, P. 2009. Xsens mvn: full 6dof human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep*.

STEED, A., AND OLIVEIRA, M. F. 2009. *Networked Graphics: Building Networked Games and Virtual Environments*. Morgan Kaufmann.

STEED, A., STEPTOE, W., OYEKOYA, W., PECE, F., WEYRICH, T., KAUTZ, J., FRIEDMAN, D., PEER, A., SOLAZZI, M., TECCHIA, F., BERGAMASCO, M., AND SLATER, M. 2012. Beaming: An asymmetric telepresence system. *IEEE Computer Graphics and Applications 32*, 6, 10–17.

STEPTOE, W., NORMAND, J.-M., OYEKOYA, O., PECE, F., GIANNOPOULOS, E., TECCHIA, F., STEED, A., WEYRICH, T., KAUTZ, J., AND SLATER, M. 2012. Acting rehearsal in collaborative multimodal mixed reality environments. *Presence: Teleoperators and Virtual Environments 21*, 4, 406–422.

TECCHIA, F., CARROZZINO, M., BACINELLI, S., ROSSI, F., VERCELLI, D., MARINO, G., GASPARELLO, P., AND BERGAMASCO, M. 2010. A flexible framework for wide-spectrum vr development. *PRESENCE: Teleoperators and Virtual Environments 19*, 4, 302–312.

TRAMBEREND, H. 1999. Avocado: A distributed virtual reality framework. In *Virtual Reality, 1999. Proceedings., IEEE*, IEEE, 14–21.

TSUI, K. M., DESAI, M., YANCO, H. A., AND UHLIK, C. 2011. Exploring use cases for telepresence robots. In *Proceedings of the 6th international conference on Human-robot interaction*, ACM, New York, NY, USA, HRI '11, 11–18.

WANG, M.-M., CAO, J.-N., LI, J., AND DASI, S. K. 2008. Middleware for wireless sensor networks: A survey. *Journal of computer science and technology 23*, 3, 305–326.

WYCKOFF, P., MCLAUGHRY, S. W., LEHMAN, T. J., AND FORD, D. A. 1998. T spaces. *IBM Systems Journal 37*, 3, 454–474.