Exploring the Design of Transitional Hybrid User Interfaces

Daniela G. Trevisan[a], Felipe Carvalho[b], Alberto Raposo[b], Carla M. D. S. Freitas[c], Luciana Nedel[c]

[a]*Computing Institute, UFF, Niteroi, RJ, Brazil*
[b]*Tecgraf, Department of Informatics, PUC-Rio, Rio de Janeiro, RJ, Brazil*
[c]*Institute of Informatics - UFRGS, Porto Alegre, RS, Brazil*

Abstract

The idea of hybrid user interfaces (HUI) does not rely only on the use of different devices but also on different interactive environments with the goal of bringing together the advantages of each environment. The main challenge regarding the development of such systems is to know which are the design aspects that should be taken into account in order to promote smooth and continuous interactions. In this way our work reinforces the importance of interactions continuity and dimensional task congruence as design principles to guide the development and interaction analysis within HUI. An example scenario was conceived from splitting a previous single desktop application for 3D volume sculpture into three different interactive environments (Wimp, Augmented Reality and Head-Mounted Immersive Virtual Reality). To achieve such goal we employ the OpenInterface platform to allow the management of several modalities for user interaction within and along the three environments. Finally, we discuss the outcomes of the analysis of interactions within our HUI according to the design principles proposed.

Keywords:
fast prototyping, hybrid user interfaces, transitional interfaces, OpenInterface

1. Introduction

Recent studies have demonstrated that in certain situations a mix of 3D and 2D interaction is preferred in relation to the exclusive use of one or the other [8]. Keeping many interaction environments in the same workplace would be of great advantage for tasks composed by many different types of sub-tasks. That difference would rely on the spatial demands of those sub-tasks; for example, some tasks would require a 2D interaction with mouse (e.g. WIMP interfaces - Windows, Icons, Menus and a Pointing Device) while others, a 3D interaction with a tangible interface (e.g. volume manipulation). It would be of great advantage to explore this heterogeneity in the sense of giving more interaction environments in the same physical workplace.

Trying to address such issue, the idea of hybrid user interfaces [10] does not rely only on the use of different devices but also on different interaction environments. The use of a hybrid interface composed by different interaction environments also enhances the exploration of the mixed reality continuum [18] because these interaction environments can complement each other, bringing particular advantages. For instance, in a single workplace, interaction environments with 2D and 3D features could be used for experimentation and execution of different interaction techniques.

To move along the continuum of mixed reality, a hybrid user interface could use the concept of transitional interfaces. The concept of transitional interfaces was first introduced by the MagicBook project [3] through an application that moved the user seamlessly along the mixed reality continuum [18]. In the MagicBook, the user walks from the real world to virtual reality passing through augmented reality using a hand-held device. The transition between these environments consisted of automated virtual camera movements and image effects, such as fade in and out.

A few years later, the concept of transitional interface was formalized aiming at a generalization of this term to multiple contexts [12]. The meaning of context somehow encompassed the idea of 3D spaces with properties such as scale (macro, micro, nano), representation (pho-

Email addresses: daniela@ic.uff.br (Daniela G. Trevisan), kamel@tecgraf.puc-rio.br (Felipe Carvalho), abraposo@tecgraf.puc-rio.br (Alberto Raposo), carla@inf.ufrgs.br (Carla M. D. S. Freitas), nedel@inf.ufrgs.br (Luciana Nedel)

torealistic, cartoon, etc.) and others. For each context, there is a clear definition of a motion function related to the properties of a virtual camera, and, for each transition between these contexts, there is a transition function related to visual factors in order to continuously provide smooth transitional motion rather than teleportation. The hardware and software setups used by authors during the development of these ideas were based on the contexts of Virtual Reality and Augmented Reality, and a constant set of input devices and displays during the studies, such as a hand-held device [18] or a Head Mounted Display combined with a 3D tracking input device [11]. We noticed that there was no change of interface paradigm during transitions since the contexts were restricted to 3D environments and interactions of transitions were conceived in an intuitive way.

This paper focus on a subset of the mixed reality continuum between WIMP and virtual reality interfaces passing through augmented reality. User interaction transitions between these three environments are provided through multimodal interactions, while the analysis of the interaction technique is performed in terms of two conceptual criteria: spatial task congruence, and interaction continuity. In order to illustrate the proposed approach, a hybrid volume sculptor application was developed using the OpenInterface platform, which is an open-source platform-independent software solution designed to support fast prototyping and implementation of interactive multimodal systems.

The remainder of the paper is structured as follows. In Section 2 we review relevant work in the area of hybrid user interfaces and transitional interactions. Design issues present in the development of hybrid user interfaces including the design of transitional interactions are discussed in Section 3. An overview of the multimodal software platform OpenInterface is presented in Section 4, while the hybrid 3D sculptor application as well as its interaction components are detailed in Section 5. Section 6 presents a usage scenario and Section 7 discusses the resulting user interaction in the hybrid interface according to the design issues introduced in Section 3. Finally, Section 8 presents conclusions and draws future work.

2. Related work on hybrid user interfaces

Interface systems that attempt to combine hardware and software components of different natures are sometimes called Hybrid User Interfaces, Mixed Reality Systems, or, until recently, Hybrid Display Systems. They are characterized by the use of multiple elements, which can be multiple devices or multiple user interfaces. The term HUI (Hybrid User Interfaces) was first proposed by Feiner and Shamash [10], referring to a heterogeneous environment, rich in interaction techniques, and with different kinds of devices used in a complementary way.

A pioneer work in this area was the Office of the Future [20], which combined several computer vision and computer graphics techniques to analyze surfaces of the real world and then add virtual information that was projected on them.

Rekimoto and Saitoh [22] explored the HUI heterogeneity using several computers and displays (projections and notebooks) in the same work environment. They follow the idea of ubiquitous computing using different devices. Some intuitive operations (pick-and-drop and hyperdragging) were created in order to transfer data among the devices. A similar project called EMMIE [6] employed different displays and devices to support co-located and remote collaboration. The proposed system used augmented reality with a see-through HMD (Head Mounted Display) to place information in the space, also providing some private visualization of information.

Another interesting work is the MagicMeeting [21], that proposed a collaborative environment using augmented reality based on tangible interfaces. By attaching 3D objects and WIMP interfaces to these tangible interfaces, the system provided interaction with 3D and 2D data.

Nakashima et al. [19] presented a prototype of a collaborative work environment with 2D and 3D environments for graphical modeling tasks. WIMP interfaces were projected on a table and 3D objects were displayed on a display called IllusionHole, which allows for 3D interactions. The Studierstube [24] also presented a collaborative system using augmented reality for co-located and remote users.

A HUI for the manipulation of medical data, also using 2D and 3D interactions, was developed by Bornik et al. [4]: a single 3D pointer is used as an interaction tool, and two visualization modalities were available, one on a tablet and another on a projection device.

Benko et al. [2] created a hybrid environment composed of a LCD (Liquid Cristal Display) placed vertically, a touchable display placed horizontally, and a see-through HMD for AR. This environment is used for the manipulation of archaeological objects. The application provided cross-dimensional gestural techniques, enabling interactions using 2D (via desktop) and 3D (via AR) visualizations, as well as the transfer of objects between the different environments.

Baumgartner et al. [1] presented a HUI to explore

the organization of a desktop, disposing documents spatially. The organization of documents was accomplished through gestures using a glove. The documents in the 3D space were visualized in an auto-stereoscopic display, and edited with the keyboard and a pen on a tablet below the display.

In general, the above-mentioned works sought to blend different technologies in order to execute certain tasks. However, there is a need for a criterion, or at least a reference to any methodology or concept, driving this "technological mixture". Moreover, there seems to be no explicit concern about possible transitions among the integrated technologies.

Conversely, although the concept of transitional interfaces was coined a decade ago in the MagicBook project [3], until now few works have been published in this topic. Grasset et al. [12] proposed an initial attempt to formalize this concept, and also published the first work [11] reporting some evaluation studies in this field. Usability, performance, presence, and awareness were assessed in that work. The evaluation scenario was composed by a series of navigation tasks that forced the subjects to transit between two interactive environments: virtual and augmented reality. However, most of the results from this work reported failures on navigation techniques used in each environment, as well as problems of disorientation on the entry and exit points from the environments after and before the transitions. The authors said that such problems would be fixed by using stereo vision and some sort of visual feedbacks.

The design of transitional interfaces is still an open problem. There are lots of issues to be addressed, for instance, the problem of transition between interactive environments with different hardware and software technologies. While the work of Grasset reported results on a scenario with a homogeneous hardware setup (HMD and a 3D input device) along the two environments, our work deals with explicit hardware and software transition, taking into account some properties to guide the design and analysis of such case.

3. Interaction design properties

In this section we describe interaction design aspects that have guided our design choices while developing the hybrid user interface. We used a user-task oriented approach based on the spatial task congruence and interaction continuity described as follows.

3.1. Dimensional task congruence

Many interactive environments force the user to perform 2D tasks using 3D techniques or vice-versa. This mismatch is problematic since there is no direct mapping between the user actions and effects on the environment or object in question.

In [8] the task congruence was supported by providing modalities for user interaction according to the task dimension. If a specific task is better performed in a n dimensional space, the user interaction modality must match the spatial demands of that task. Besides that, more problematic are the mixed-dimensional tasks. Here, the task is neither 2D nor 3D, but has characteristics of both. This is the case of the application scenario described in Section 5.

A well-known classification of 3D tasks divided interaction techniques into three main groups: selection, manipulation and navigation [5]. Our study relates task congruence with this classification scheme in the sense of providing to the user some interactive environments that aim at supporting tasks with techniques that have some spatial match with interaction devices available in the workplace. For example, in the manipulation environment, we adopted the use of the 3D information of position and orientation of some objects – used as hand-tools by the user – to enable more natural movements of hands.

Summarizing, dimensional task congruence property is the condition whereby the spatial demands of a task are matched directly by the interaction technique that is used to execute it.

3.2. Transitional Interaction

An interactive environment is assumed to be the complete presentation environment required for carrying out a particular interactive task. The interactive environment contains representations of the visual, haptic and auditory elements that a user interface offers to its users, as well as their relationships. In more complex environments, the user's task can be distributed along various interactive environments and discontinuity during transitions can become a problem. The idea of continuity is related to the process of avoiding breaks during interaction, which could result in disorientation and failure.

The importance of continuity in the development of real-time collaboration systems was mentioned by Ishii et al. [15]. They define a seam as a spatial, temporal or functional constraint that forces the user to shift among a variety of spaces or modes of operation. For example, the seam between word processing using a computer and traditional pen and paper makes it difficult to produce digital copies of handwritten documents without a translation step. All authors have agreed that systems asking users to abandon their acquired skills and

to learn a new protocol are likely to encounter strong resistance.

Dubois et al. [9] consider continuity at the perceptual and cognitive levels. Perceptual continuity is present if the user perceives the different representations of a given entity directly and smoothly. Cognitive continuity is present if the cognitive processes that are involved in the interpretation of the different perceived representations are similar.

In this work we consider continuity as the capability of the system to promote a smooth user interaction during task accomplishment, considering perceptual, cognitive and functional aspects [25, 26]. Perceptual continuity is defined as the ability of the system to provide information to the user in one perceptual environment (e.g. when the user is wearing a see-through head-mounted display (HMD)). Cognitive continuity is defined as the ability of the system to ensure that the user will interpret the perceived information correctly and that the perceived information is correct regarding to the internal state of the system (e.g. by using similar representations of the real and virtual objects). Functional continuity is defined as the adaptability of the user to change or learn new modes of interaction. Consequently, the functional property is related to the interaction technique.

All these definitions pointed out the need for smooth transitions between different operation modes to avoid frustration during the accomplishment of complex tasks.

4. Multimodal platform description

In this section we describe some technical aspects of the OpenInterface multimodal platform, which were necessary to understand the development of our case study application.

In OpenInterface[1], each component interface is described and registered into the repository using the XML-based Component Interface Description Language (CIDL). The C++ kernel then automatically generates proxies to perform the actual integration. Using a graphical front-end or the kernel API – that allows embedding the platform within the final application – users can configure components and compose complex execution pipelines for implementing multimodal applications.

[1] www.openinterface.org

4.1. Component description

An OpenInterface component consists of a computing unit, an algorithm, a bundled library, etc. To be integrated into OpenInterface, a component has:

- an id: an unique name for identifying a component among all other OpenInterface components. The structure of the id follows a hierarchical structure.
- a name: a simple human readable word identifying (non uniquely) the component.
- a language: the programming language used to implement the component.
- a container: the description of the delivery format. Basically, it aims at defining the way by which a component will be made available. These ways include jar file (for Java), shared object library (for C/C++), archive (C/C++), and directory (Matlab, Java).
- a description of the I/O pins: this is where one describes the interfaces of a component. Basically, it describes the logical units and their input and output functions.

An IO element encompasses the declaration of a component's interface. It contains declarations of the different facets of a component along with their respective sink or source pin. A facet is typically a logical unit inside a component, while a pin is a function/method of a given unit.

A *sink pin* represents a way for a component to receive data. Conversely, a source pin is the way for a component to send data. Both sink and source pins have mandatory properties:

- id: each pin has a unique id inside a given facet.
- interface: which it is a description of a function's signature. Currently, there are some restrictions on the kind of functions that can be described; the types of the functions parameters are limited to primitive types such as boolean, byte, char, int, long, float, double, string, and up to 3-dimensional array of primitive types.

The *Source pin* has an additional property: *Callback*. A *callback* is the way for a component to call a function it does not know about at implementation time. For instance, a mouse driver component would expose a callback for notifying about the current mouse position and buttons' states. A callback setter will then be called to

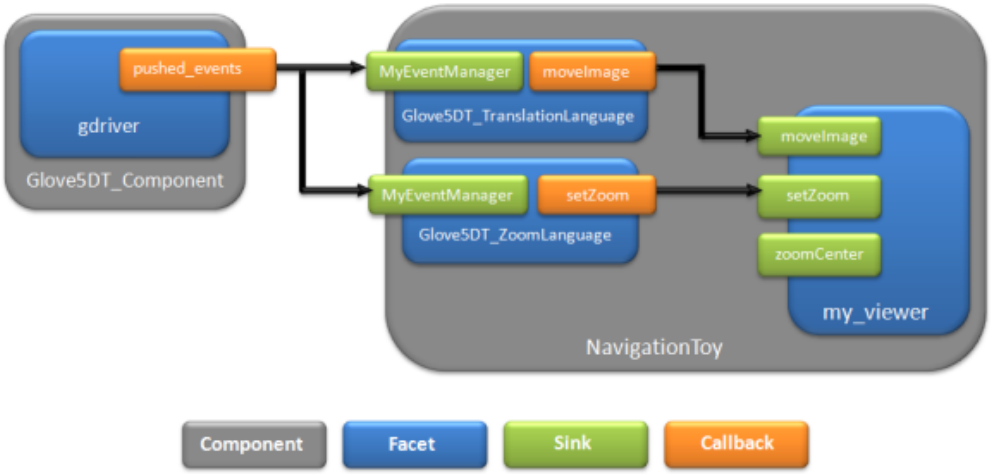

Figure 1: OpenInterface pipeline.

register the external function. So, instead of polling the mouse component for its state, the user will be notified by the registered function only when the state changes. For a visual representation of the component and its pins, see Figure 2, bottom view.

In order to assist the creation of a component description, the Component Builder plug-in provides the following features: the CIDL generation from source code, and the component packaging and deployment. The user opens the desired source file – which represents the interface of the component – within the editor, and can interactively modify the generated corresponding XML description. The user can directly edit code within the editor by either modifying non-compliant interface or removing undesired functions. Currently, only C/C++ and Java source code parsing are supported.

4.2. Pipe description

As illustrated in Figure 1, to build a running application, OpenInterface uses the concept of pipeline to interconnect and configure components. It allows controlling the components' life cycle and execution site (remote, local), and provides low level (threshold, filter, etc.) and high level (multicast, synchronization, etc.) data flow control for building up complex systems. A pipeline also supports dynamic reconfiguration of connections at runtime.

A pipeline thus defines and configures connections between components using the PDCL (Pipeline Description and Configuration Language; see [16], for more details). It provides simple embedded data flow controls, such as direct function and asynchronous calls, as well as simple mechanisms for extending the expressiveness of the pipeline, in order to simplify intuitive interaction implementation.

The pipeline also allows isolating a component in a separate process or distributing the execution of an assembly across a set of computers running an instance of the OpenInterface kernel. The distribution can either be performed seamlessly using the PDCL syntax or with connectors implementing a well-known protocol.

In order to support this stage, OpenInterface provides the design-time visual editor (Figure 2). The figure shows a typical interactive session within the SKEMMI editor. While the left pane of the visual editor contains a hierarchical view of the project being built, the right pane contains the integrated components, the adapters, and the annotation elements, in different tabs respectively. Using drag and drop, users can initiate the conceptual assembly of the desired components (Figure 2, top) and further refine the pipeline to actually implement the desired behavior (Figure 2, middle). The SKEMMI editor is further explained in the next section.

4.3. Graphical Editor

This section describes the Eclipse-based end-user interface of the OpenInterface kernel. The SKEMMI editor provides data flow editing features while also putting more emphasis on informal prototyping, through machine learning components, and on the collaboration of different actors, designers and programmers, within a complex iterative design process [17]. To help users to abstract from implementation details or to dive into them at their best convenience, the SKEMMI editor provides the ability to seamlessly navigate through three

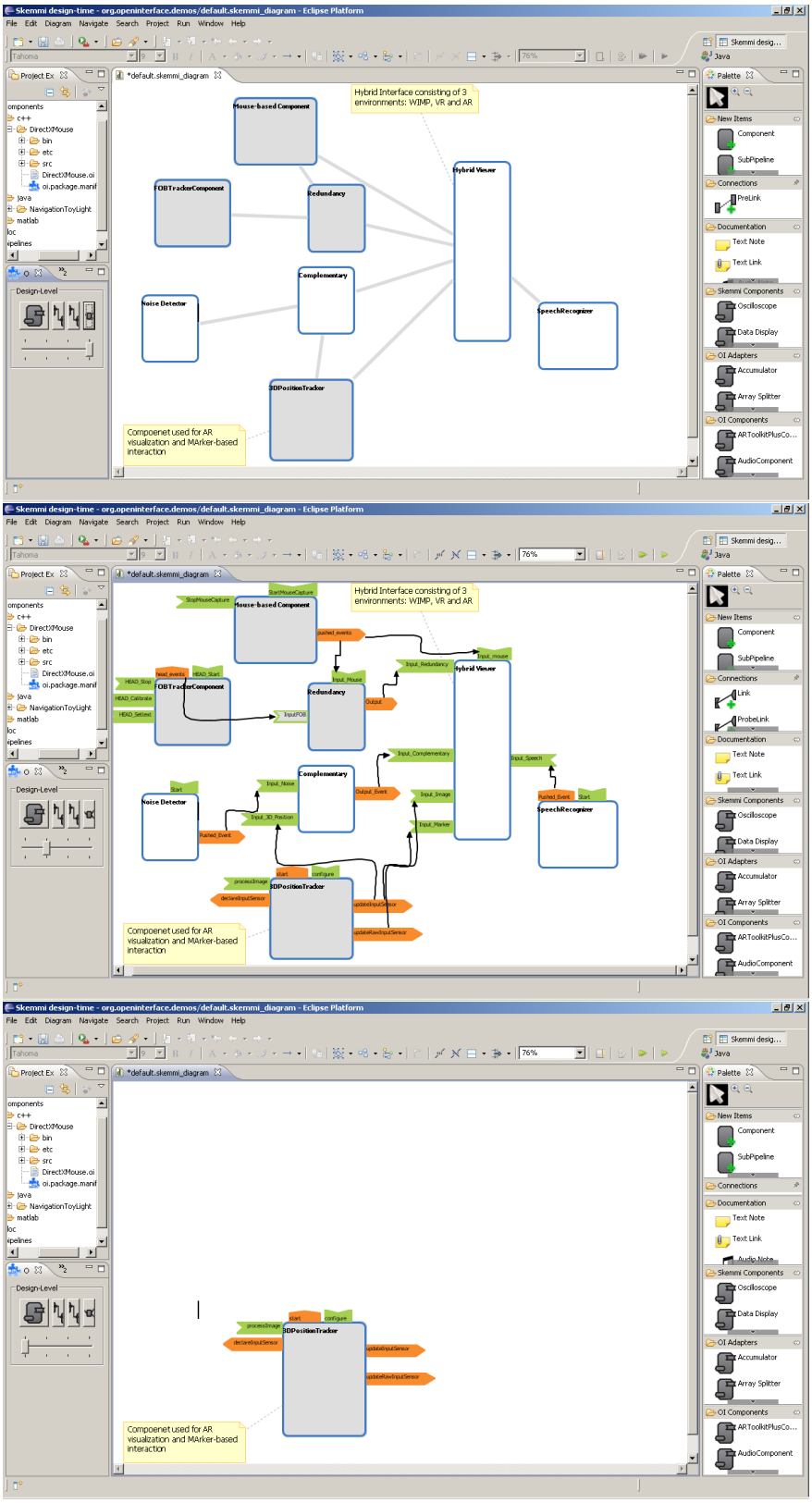

Figure 2: Three-level view of the 3D hybrid sculpture application within SKEMMI. Top: work flow; Middle: data flow; Bottom: component.

design-levels by the use of a three-level scale. The common representation for components is to depict them, their ports, and each link between input and output ports. However, this information may be superfluous in a first overall design sketch. For example, when initially designing interactions, less emphasis is put on ports and types. Basically, the required components are elicited, logical links are drawn between them, and notes and documentations are added to describe the overall concept.

To support this "brainstorming" design phase, the graphical editor provides a "work flow" level prospect that shows only components, conceptual links between them, and annotations, as illustrated by Figure 2, top view. This level can be further refined to an actual implementation level (see Figure 2, center view). The same work flow is augmented with technical details such as ports, data types, etc. At this stage, conceptual links can be instantiated to apply the mapping between design concept, notes, requirements and available components. The third level gets focuses on a single component design (see Figure 2, bottom view), also allowing the redefinition of a component interface, if it does not suit the requirements of the designers.

5. Demonstration example

In this section, we demonstrate how to transform an already-implemented single-desktop 3D application into three complementary interactive environments with support to multimodal interactions.

5.1. Example application overview

We start by describing the interaction scenario we have chosen for demonstrating the HUI usage. Volume sculpting is needed in the context of rendering volumetric datasets in order to provide an intuitive way to examine and explore inner parts of datasets such as those obtained from medical imaging devices or physical phenomena simulations.

Huff et al. [13, 14] reported the development of interactive, intuitive and easy-to-use sculpting tools, which specify regions within the volume to be discarded from rendering, thus allowing the inspection of the inner volume. Interactive rates were obtained for these sculpting tools by running special fragment programs on the graphics hardware. These tools were implemented using two interaction metaphors (virtual pointer and virtual hand) and following different approaches in terms of devices and single versus two-handed interaction.

We focused on the tools that used three-dimensional geometries associated to the virtual hand metaphor [5].

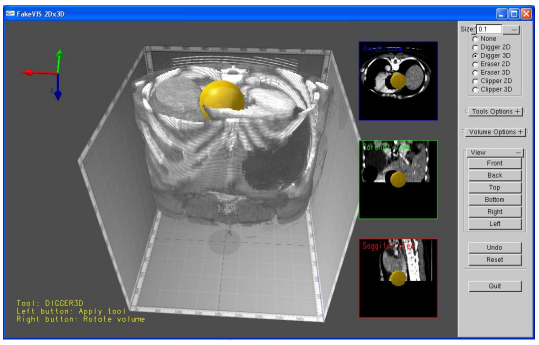

Figure 3: First version of the viewer with 2D and 3D visualizations of an interactive sculpting session where the volume of interest is a torso dataset.

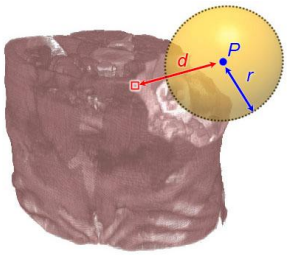

Figure 4: 3D Digger tool: eliminating voxels from the volume.

The virtual hand is represented as different cursors with shapes that reproduce with 3D geometries the sculpting tools the user can apply to the 3D volume: 3D Rubber (which allows erasing parts of the volume); 3D Digger (that behaves as a carving tool); and 3D Clipper (represented as a cutting plane).

The system showed in Figure 3 was initially developed to be used in a desktop workstation using 2D and 3D mouse-based interactions. In order to address the design issue of keeping dimensional congruence [8], which says the interaction technique used to perform a task must match the spatial demands of that task, we propose here to split user's interactions along a hybrid user interface.

In this work, as a proof-of-concept, we adopted the task provided by the 3D Digger tool as shown in Figure 4. This tool eliminates voxels in the interior of a virtual sphere, i.e., the region being defined by a 3D point P and a ray r interactively specified by the user. The selection of a voxel for removal is done calculating the Euclidean 3D distance d of its center to P. A voxel is removed if $d < r$.

In order to allow the user to sculpt the volume with a digger tool within and through the three environments (WIMP, AR – augmented reality and HIVR – Head-

Mounted Immersive VR), we have designed interactions for each environment, as well as transitions between them.

We developed and integrated into the OpenInterface platform eight components: the viewer itself (1); five interaction components, which are image capture (2), 3D tracking movements to support volume and tool manipulation (3), noise detection (4), user head tracking (5), speech recognition (6); and two components for fusion, complementarity fusion (7) and redundancy fusion (8). All these components, together with a mouse interaction component, were integrated into the OpenInterface platform allowing the rapid prototyping of our hybrid environment with multimodal user interactions.

Figure 5 presents the first conception of our case study application, which could be implemented in a very similar way using SKEMMI Graphical Editor, as illustrated in Figure 2.

Details about each component are given in the next subsections.

5.2. Hybrid viewer

As mentioned before, the hybrid viewer application was developed by splitting a previous single desktop viewer (see Figure 3) to allow several modalities for user interaction within and through the three environments (WIMP, AR, and HIVR). In this way, the hybrid viewer component consists of the final user interface, where the combined or assigned interaction events from the other components are continuously arriving.

Figure 5 shows an overview of the required interaction devices and components of the hybrid viewer user interface, while Figure 6 presents the interactions to allow seamless transitions between the three environments. All components involved in such interactions are described in the next subsections.

When the application is launched by the OpenInterface execution pipeline, all interaction components, as well as the hybrid viewer component, are launched simultaneously. The hybrid viewer starts the WIMP user interface with mouse-based and speech interaction ready for use. In order to dispatch the interaction event to the corresponding interactive environment, we developed a dialog control. It is always checking the state of the speech port, and when a speech event arrives in the hybrid viewer, the related environment is activated as well as all the user interaction modalities that are available for that specific environment. The interaction modalities available in each environment were previously defined in design time by specifying the execution pipeline of the application, as we describe hereafter.

5.2.1. Wimp interaction

The WIMP interface was used for mouse-based interaction on menus and application control settings. That decision was taken as an advantage for the hybrid environment due to the use of an established technology. Menus and other items could be created to be used inside AR and HIVR but that would be a bad design choice since most of the settings remain static during the interaction within these environments. Also, menus inside these environments would visually disturb the attention of the users, breaking down the sense of presence and occupying screen space. Moreover, the use of other interaction modalities such as vocal commands for menus selection might also be considered a bad design choice, since vocal user interaction can take too long time and demand a great effort for commands memorization.

5.2.2. Immersive virtual reality interaction

In navigation tasks, exploration and search are frequent tasks for the creation of a spatial knowledge. For the present work, exploration objectives were taken as a closer visual evaluation of the sculpted areas. This closer visualization would be accomplished in the AR environment by bringing the marker that holds the volume near the camera. Since computer vision operations are used to detect the marker, it is necessary some distance between the marker and the camera, and that distance would not allow a closer evaluation. In order to provide that closer evaluation, a dedicated and immersive virtual reality environment was created.

5.2.3. Augmented reality interaction

The AR environment was used for manipulation tasks using spatial interaction techniques. Bimanual and multimodal interactions on tangible interfaces could be considered advantageous for that. Manipulation tasks can be executed using egocentric or exocentric views. For precise manipulations the egocentric view has been verified as an adequate alternative. An egocentric view could be used in the HIVR environment. However, in the context of such hybrid environment, AR seems more appropriate because the visibility of the real environment and real hands provides a more natural bimanual interaction using tangible interfaces holding virtual information.

Tangible interfaces emphasize the use of physical objects as interaction tools. If such kind of interaction were carried out in the HIVR, additional information would be necessary to represent the real hands and to handle tracking precision problems. In this way, to interact with the augmented reality scenario, the user

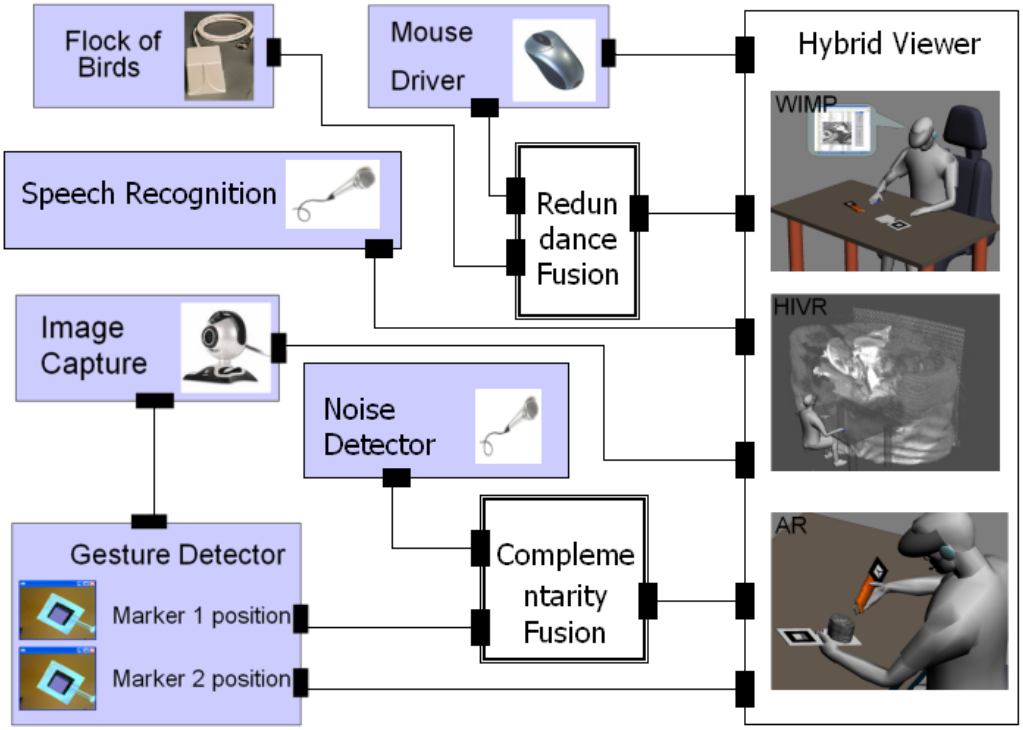

Figure 5: Initial conception of interactions within the example application scenario.

should move the cursor position by gesturing in his/her physical workspace, while the volume is being visualized at the captured and processed marker position. Volume manipulations such as rotations and translations are allowed by moving the marker – with one hand – in front of the camera, while the other user's hand is used for tool manipulation. The user blows in the microphone to apply the selected tool (e.g. to remove the volume region indicated by the tool position).

5.2.4. Transitional scenario interaction

The transitional interfaces used to move across the environments were made using a multimodal approach combining voice commands (see Section 5.8) or spatial interactions commands (see Section 5.5). For instance, to trigger the HIVR environment from the AR environment, the interaction condition is to use a voice command (the word "VR", from virtual reality) and, simultaneously, to keep the tool position inside the bounding box of the volume. The system will select the region of interest indicated by the 3D tool position in the AR as being the starting viewpoint in the HIVR environment. The only condition to go from any environment to the WIMP environment is to use the voice command saying the word "WIMP" (meaning the conventional graphical interface). The same rule is applied to go to the AR environment saying the word "AR" (meaning augmented reality environment). Once the user is inside the WIMP environment, a small window showing the AR environment is kept visible to allow the visualization of some application changes during mouse-based interactions. If the user brings the two markers (volume's marker and tool's marker) close enough inside the image, the system will automatically start the AR environment too.

5.3. Fusion components

The CARE properties [7] have been shown to be useful concepts for the design and evaluation of multimodal interaction. We have decided to make these concepts explicit during the application development.

While *equivalence* and *assignment* express the availability and respective absence of choice between multiple modalities for performing a given task, *complementarity* and *redundancy* describe relationships between devices, languages or, more generally, between modalities for performing a given task.

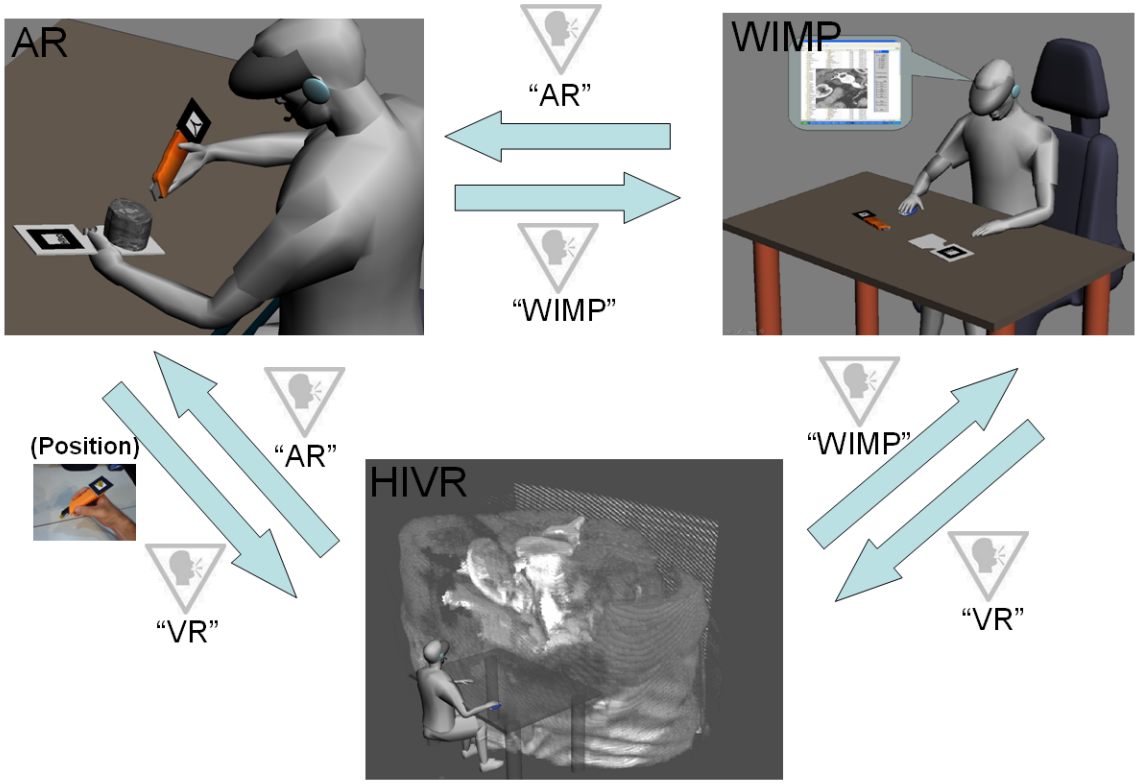

Figure 6: User's interaction along the three environments: WIMP, HIVR and AR.

These composition components describe the fusion mechanism of data provided by 2 to n interactions components. The criteria for triggering fusion are twofold: the complementarity/redundancy of data, and the time. For the management of time, a temporal window is associated with each piece of data handled by a composition component. A temporal window defines the temporal proximity of two pieces of data ($\pm\Delta t$), and is used to trigger fusion. Two pieces of data $d1$ and $d2$ are combined if their time-stamps, $t1$ and $t2$, respectively, are temporally close. $Close(t1, t2)$ is satisfied if $t2 \in [t1 - \Delta t, t1 + \Delta t]$. For the management of data, the same interactive environment in use is responsible for validating the data that will be available for the user's interaction at this time.

As can be observed in Figure 5, the user can interact with the volume through gestures captured by a camera, using a conventional mouse, a position capture device, or a noise detector, and these modalities are fused according to the CARE properties. In the following subsections we illustrate how such mechanisms of fusion were implemented in our application.

5.4. Image capture

The Image Capture component used within OpenInterface uses the ARToolkit[2] library to capture the webcam image. Then, this component sends the captured image to the 3D position detector component to be processed, and to the hybrid viewer component to be visualized. However, the image visualization is activated only when the speech event "AR" is detected by the dialog control in the Hybrid Viewer component, because this image corresponds to markers used in the augmented reality environment.

5.5. 3D position detector

The 3D Position Detector component uses the ARToolkit library to capture the positions of the 3D markers in the physical space. The webcam is active while the user moves two printed markers in the physical space to interact within the augmented viewer. One of the markers is used to hold and manipulate the volumetric dataset visual representation (see Figure 6, top left)

[2]http://sourceforge.net/projects/artoolkit

on the user hand. The other one is used to identify the tracked tool and sculpt the volume (see Figure 7).

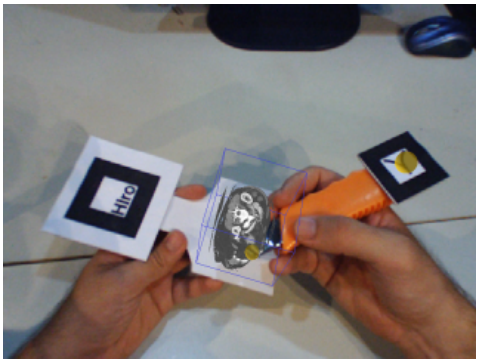

Figure 7: Bimanual, multimodal user interaction in the AR environment.

5.6. Noise detector

The NoiseDetector component provides a single command that can be activated using a microphone. The technique is simple and does not require much processing. The component takes small sound buffers from the microphone and when it detects a sufficient noise, it triggers a callback function passing true (otherwise, it passes false). In our implementation, it passes true when more than 40% of the samples (absolute value) is greater than 0.8 (their maximum value is 1.0). This component was implemented in C++ using the Portable Real-Time Audio Library [3] to capture sound and to process the signal.

The best way to generate such noise is by blowing directly into the microphone. Each time the blowing noise is detected, a single event is sent to the complementary component to be merged with the 3D location data sent by the 3D position detector component. At this point, we are combining two complementary signals (e.g. noise + 3D position) to perform a single interaction in the hybrid viewer, which is erasing the voxels underneath the position marked by the tracked tool. In the case of signals detection failure, or if the time stamp between the two detected events is not satisfying the *Close* function (previously described in Section 5.3), none event is sent to the Hybrid Viewer component. Consequently, none action is performed. More details about this interactive component can be found in [23].

[3] http://www.portaudio.com/

5.7. User head tracking

When the user is inside the HIVR environment, the mouse-based interactions become available. However, when it is not being used, the system enables the head tracking, and the virtual camera follows the user head movement (redundancy mechanism of fusion). We have implemented this technique using a Flock of Birds[4] motion tracker as a component into the OpenInterface platform. This component continuously sends the user head orientation to the Hybrid Viewer component, but it is activated only when the speech event "VR" (meaning virtual reality environment) is interpreted by the dialog control.

5.8. Speech recognition

As mentioned before, speaking "AR", "VR" and "WIMP" trigger the transitions between the environments as illustrated in Figure 6. To implement such vocal commands, we have used Sphinx-4[5], a state-of-the-art speech recognition system entirely written in Java. It is now an interaction component available in the OpenInterface platform.

Sphinx uses speech recognizers based on statistical Hidden Markov Models and the Java Speech API Grammar Format (JSGF) to perform speech recognition using a BNF-style grammar. Our grammar to allow user transitions from one environment to another is as follows:

```
#JSGF V1.0;
grammar navigation;
public <navigation> =|VR|AR|Wimp;
```

Commands recognition results provided by this component seem to be satisfactory in terms of user interaction performance. In a very preliminar evaluation, approximately 9 in 10 commands were successfully recognized after a short time spent for the user, in training mode. These results were obtained using a head set microphone and none interference with the noise detector component was observed.

6. Usage scenario

We now demonstrate how our hybrid viewer with support to multimodal interactions could be used for visually exploring a volume through sculpting. We will use a very simple scenario based on removing a predefined region from a volume. The operations involved

[4] http://www.ascension-tech.com/

[5] http://sourceforge.net/projects/cmusphinx

in this task are usual to 3D visual exploration performed by expert users.

We will follow Mark, a 3D visualization analyst who is trying to navigate along the three environments in order to perform adequate interactions to remove voxels from the volume. Mark has to remove the maximum number as possible of red voxels from the volume, while avoiding the elimination of the white ones voxels as well.

Mark uses the hardware set up as illustrated in Figure 6. He knows that two input devices allow the manipulation of two different objects in the application: the tool and the volume of interest (VOI). The tool can be translated along the x, y, and z axes, while the VOI can be rotated around itself but cannot be translated.

Mark has no previous knowledge about the volume dataset, and so his initial exploration goals are very fuzzy and he would like to let the visual navigation process itself guide his interaction.

When the hybrid viewer is launched, Mark is confronted with the initial WIMP interface of the application. From that, Mark is able to select in a menu which volume dataset is available to interact with as well as the desired tool. Mark selects the digger tool and the torso dataset. As a feedback, the volume and the tool representation are shown in the WIMP environment but Mark can not interact with that. In order to manipulate both volume and tool, Mark pronounces "AR", and by placing the markers in front of the camera he can see the volume on the marker and the virtual sphere at the end of the physical tool.

Now, in the AR environment, Mark can use the bimanual interaction to find the region to be removed into the volume. When Mark believes to be pointing at red voxels, he blows in the microphone, and the voxels inside the sphere are removed. Mark starts by removing voxels located in the middle of the red region. Before removing voxels located in the boundaries of the region he decides to explore the volume in depth. Then, while Mark is still pointing at the region of interest he pronounces "VR", and he is gracefully guided to the HIVR environment directly to the region of interest. There, Mark explores the volume by moving his head; he can visualize the red voxels still remaining. However, for the first time in that immersive environment, Mark feels a certain difficulty in moving his head to navigate into the volume. So he tries to navigate using the mouse-based interaction. Mark concludes that remaining voxels could be properly removed if the size of the virtual sphere is reduced. Then, he pronounces "WIMP", and is driven to the "WIMP" environment, where the menu is available for mouse-based interactions. After he changed the size parameter of the tool, Mark pronounces "AR" to go back to the AR environment, and to continue interacting with the volume and removing red voxels. Transitions between the environments are performed until the task is completed.

7. Interaction analysis

According to the design issues discussed in Section 3 and the proposed hybrid interaction scenario presented in Section 6, in this section we present the analysis of the user interaction for our case study application.

Regarding the hybrid viewer itself we can say that the supported interactive environments (e.g. WIMP, immersive virtual and augmented interactions) are complementar as fusion mechanism, once the specific task (e.g. volume sculpting) is performed partially in the WIMP, partially in the augmented, and partially in the immersive virtual environment. Then, the criteria used to split the task between these environments were based on the spatial task congruence and task continuity.

7.1. Matching the task congruence property

The task congruence is supported by providing different modalities for user interaction according to the task dimension. In this way, we have assigned the mouse-based user interaction modality to the WIMP environment, considering tasks of menu selection.

Once the user moves to the augmented reality environment in order to sculpt the volume, the user interaction modalities become bimanual and three-dimensional. The bimanual gestural modality for tool and volume manipulation allows moving the selected tool (that was the cursor associated to the virtual hand metaphor in the original application) and the volume in the augmented viewer simulating the real user interaction scenario.

Once the HIVR scenario is used for volume inspection, the dimensional congruence in the HIVR environment was partially matched. Not all navigation techniques can be used in this particular workplace (e.g. desktop workplace), because the user remains sit all the time. In this context, navigation techniques using physical locomotion are not feasible. The use of a tracked HMD brings a corrected match for the task of closer visual evaluation once it provides 6 DoF for head movements. The navigation task, responsible for the movements of the user along the scene, had a partial dimensional congruence matched because the user actually does not move in the real world. Mouse-based interactions were used for the virtual navigation using the user

head direction as start input or continuous input. That partial match becomes evident from the combination of an input coming from the tracked HMD (e.g. the 3D user head direction) and 2D mouse-based techniques.

7.2. Matching the task continuity property

In order to provide task continuity, mainly regarding the perceptual continuity property, it was decided to use a webcam with wide-angle lenses that provide wider field-of-view (FoV). This guarantees that a larger part of the physical space is visible, and less movement of the head is necessary to have an overview of that space during the manipulation task with two hands.

The task continuity across environments was achieved by using short vocal commands to change from one environment to another and the use of the same display (e.g. HMD) for visualization in all environments. The speech commands require a low cognitive effort for words memorization, keeping the user hands free for the ongoing task execution.

Task continuity was observed between AR and VR transitions by keeping the user interaction focused on the region of interest (inside the volume) indicated by the 3D tool position. Besides that, task continuity between augmented and virtual environments transitions is maintained because the entry point in the volume, in the virtual environment, is given by the tracked tool position in the augmented environment.

On the other hand, we detected a functional discontinuity between AR and WIMP transitions when the user changes of interactive device, from bimanual tracked tools in the augmented environment to mouse interactions in the WIMP environment. Such discontinuity can be justified if we consider that we provided task congruence using the adequate modalities for user interaction according to the task dimension, as explained before. The same reasoning can be used for the transition from VR to WIMP. In this case the user is changing from the tracked HMD interaction – for 3D navigation in the volume – to mouse-based – for 2D WIMP navigation –, and again task congruence is kept in order to maintain the functional continuity of the task.

8. Conclusions and future work

In this work we have identified some design issues that hybrid environments with support to transition between different interaction modalities should be able to address: dimensional task demand and task continuity. Obviously, there is no way to say definitely which interaction technique or set of interaction techniques best suit an entire category of interactive tasks. However, we believe that the design properties here discussed are becoming an important approach for bridging the gap between analysis and design in task-oriented design of HUIs.

We have also demonstrated how systems with multimodal interaction can be quickly designed and implemented using the OpenInterface platform. We have used such multimodal platform to assembly several interaction components in the hybrid environment presented using a volume sculpting tool as acase study application.

The main advantages in using such platform can be summarized as follows:

- Integrate new modalities, devices, and functional cores (i.e. components) into the platform. The software can be provided in any supported programming languages (C/C++, Java, Matlab and .NET; extensions can be easily added), and semi-automatic tools are provided to ease that process.

- Use a graphical interface to dynamically or statically combine components, and generate a running application. External applications can also control the pipeline by using the provided multi-language API.

It is worth noting that no significant delay was observed in events recognition after the integration of the several components into the multimodal platform, and all components used in this approach are open source and can be freely downloaded from Internet. With our demonstration we hope to motivate the development of several multimodal applications in many application fields.

Our purpose in future works is to identify tendencies and build a body of reusable knowledge leading towards better overall interface design. For that, we intend to develop a protocol for usability evaluation concerning to multimodal user interaction, in order to verify these design issues while the user is performing hybrid tasks, and compare the results with the previous work [13] obtained from user tests in a single desktop environment.

On the other hand, we have also observed that the concept of continuous interaction space became more evident in the context of hybrid user interfaces following the ideas of ubiquitous computing, which argues that interaction environments should not reside only in the user desktop, but also in other devices and in the surrounding environment. In this direction, in next versions of our system we intend to provide dynamic components for allowing interaction according to different

contexts of use (e.g. user, environment, task, or platform).

Acknowledgments

This work was developed in the scope of the projects CNPq/INRIA EDGE and CNPq/FNRS MIDAS. Special thanks to Lionel Lawson from *Universite catolique de Louvain*, Belgium, for the OpenInterface support. Alberto Raposo thanks FAPERJ for the individual support (#E-26/102.273/2009).

References

[1] S. Baumgartner, A. Ebert, M. Deller, Dimensional congruence for interactive visual data mining and knowledge discovery., in: EuroVis, Eurographics Association, 2007, pp. 99–106.

[2] H. Benko, E. Ishak, S. Feiner, Cross-dimensional gestural interaction techniques for hybrid immersive environments, in: Virtual Reality 2005, IEEE, 2005, pp. 209–216.

[3] M. Billinghurst, H. Kato, I. Poupyrev, The magicbook - moving seamlessly between reality and virtuality, in: IWAR 99, pp. 35–44.

[4] A. Bornik, R. Beichel, E. Kruijff, B. Reitinger, D. Schmalstieg, A hybrid user interface for manipulation of volumetric medical data, in: IEEE Symposium on 3D User Interfaces, IEEE, 2006.

[5] D. Bowman, E. Kruijff, J. LaViola, I. Poupyrev, 3D User Interfaces - Theory and Practice, Addison-Wesley, 2005.

[6] A. Butz, T. Hollerer, et al., Enveloping users and computers in a collaborative 3d augmented reality, in: IWAR 99, pp. 35–44.

[7] J. Coutaz, L. Nigay, D. Salber, A. Blandford, J. May, R.M. Young, Four easy pieces for assessing the usability of multimodal interaction: The CARE properties, in: INTERACT, pp. 115–120.

[8] R.P. Darken, R. Durost, Mixed-dimension interaction in virtual environments, in: VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology, ACM, New York, NY, USA, 2005, pp. 38–45.

[9] E. Dubois, L. Nigay, J. Troccaz, Assessing continuity and compatibility in augmented reality systems, Journal of Universal Access in the Information Society 1 (2002) 263–273.

[10] S. Feiner, A. Shamash, Hybrid user interfaces: breeding virtually bigger interfaces for physically smaller computers, in: UIST '91: Proceedings of the 4th annual ACM symposium on User interface software and technology, ACM, New York, NY, USA, 1991, pp. 9–17.

[11] R. Grasset, A. Duenser, M. Billinghurst, Moving between contexts - a user evaluation of a transitional interface, in: ICAT 2008: 18th International Conference on Artificial Reality and Telexistence, Keio University, Yokohama, Japan.

[12] R. Grasset, J. Looser, M. Billinghurst, Transitional interface: concept, issues and framework., in: ISMAR, IEEE, 2006, pp. 231–232.

[13] R. Huff, C. Dietrich, L. Nedel, C. Freitas, S. Olabarriaga, Erasing, digging and clipping in volumetric datasets with one or two hands, in: ACM International Conference on Virtual Reality Continuum and Its Applications, ACM, 2006, pp. 271–278.

[14] R. Huff, R.S. Rosa-Junior, L. Nedel, C. Freitas, Volume sculpting based on geometric tools, Journal of the Brazilian Computer Society 15 (2009) 3–18.

[15] H. Ishii, M. Kobayashi, K. Arita, Iterative design of seamless collaboration media, Commun. ACM 37 (1994) 83–97.

[16] J.Y. Lawson, Openinterface description languages specification, Technical report, available at http://www.openinterface.org/platform/documentation (2006).

[17] J.Y.L. Lawson, A.A. Al-Akkad, J. Vanderdonckt, B. Macq, An open source workbench for prototyping multimodal interactions based on off-the-shelf heterogeneous components, in: EICS '09: Proceedings of the 1st ACM SIGCHI symposium on Engineering interactive computing systems, ACM, New York, NY, USA, 2009, pp. 245–254.

[18] P. Milgram, H.J. Colquhoun, A taxonomy of real and virtual world display integration, Mixed Reality, Merging Real and Virtual Environments, Ohmshda and Springer-Verlag (1999) 5–30.

[19] K. Nakashima, T. Machida, K. Kiyokawa, H. Takamura, A 2d-3d integrated environment for cooperative work, in: Symposium on Virtual Reality Software and Technology, ACM, 2005, pp. 16–22.

[20] R. Rascar, G. Welch, M. Cutts, et al, The office of the future: A unified approach to image-based modeling and spatially immersive displays., in: SIGGRAPH, ACM, 1998, pp. 179–188.

[21] H. Regenbrecht, M. Wagner, G. Baratoff, Magicmeeting: A collaborative tangible augmented reality system, Virtual Reality - Systems, Development and Applications 6 (2002) 151–166.

[22] J. Rekimoto, M. Saitoh, Augmented surfaces: a spatially continuous work space for hybrid computing environments, in: Proceedings of CHI'99, ACM, 1999, pp. 378–385.

[23] R.S. da Rosa, Jr., M.A.C. Farias, D.G. Trevisan, L.P. Nedel, C.M.D.S. Freitas, A multimodal medical sculptor, in: INTERACT'07: Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 637–640.

[24] D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L.M. Encarnação, M. Gervautz, W. Purgathofer, The studierstube augmented reality project, Presence: Teleoperators and Virtual Environments 11 (2002) 33–54.

[25] D.G. Trevisan, Designing smooth connections between worlds, in: Extended abstracts of the 2004 conference on Human factors and computing systems CHI2004, Session Doctoral Consortium, ACM Press, 2004, pp. 1043–1044.

[26] D.G. Trevisan, J. Vanderdonckt, B. Macq, Conceptualizing mixed spaces of interaction for designing continuous interaction, Virtual Reality Journal 8 (2004) 83–95.