Mobile Collaborative Augmented Reality

Gerhard Reitmayr and Dieter Schmalstieg
Vienna University of Technology
{reitmayr|schmalstieg}@ims.tuwien.ac.at

Abstract

The combination of mobile computing and collaborative Augmented Reality into a single system makes the power of computer enhanced interaction and communication in the real world accessible anytime and everywhere. This paper describes our work to build a mobile collaborative Augmented Reality system that supports true stereoscopic 3D graphics, a pen and pad interface and direct interaction with virtual objects. The system is assembled from off-the-shelf hardware components and serves as a basic test bed for user interface experiments related to computer supported collaborative work in Augmented Reality. A mobile platform implementing the described features and collaboration between mobile and stationary users are demonstrated.

Keywords: Augmented Reality, Mobile Computing, Wearable Computing, Computer Supported Collaborative Work, 3D Interaction, Hybrid Tracking.

1. Introduction and related work

This work explores the combination of three exciting emerging technologies:

- Augmented Reality (AR), enhancing a user's perception of the real world with computer generated entities,
- Mobile computing, allowing users to access and manipulate information anytime and independent of location, and
- Computer supported collaborative work (CSCW), allowing the computer to be used as a medium for human communication.

The combination of all three areas promises exciting new applications of mobile collaboration. Users of mobile AR can engage in spontaneous collaboration involving manipulation - possibly construction - of complex 3D models. In this paper, we present some technological advances in this direction. We begin by reviewing some fundamental issues and related work on the combination of AR with mobile computing and with CSCW.

Figure 1. A user wearing the mobile Augmented Reality kit.

1.1. Mobile Augmented Reality

Augmented Reality and mobile computing are often mentioned together, as many mobile computing platforms rely on some kind of head-up or head-mounted display to provide continuous access to information, often coupled with hands-free operation. The ultimate goal is to make the mobile computer a part of the user's normal daily life [23]. Augmented Reality as a user interface for mobile computing is particularly powerful when the computer has access to information on location and situation, so it can provide contextual information. A prominent example that served as an inspiration to our approach is Columbia's "Touring Machine", which is used as a mobile multimedia information system [6] and journalistic tool [10].

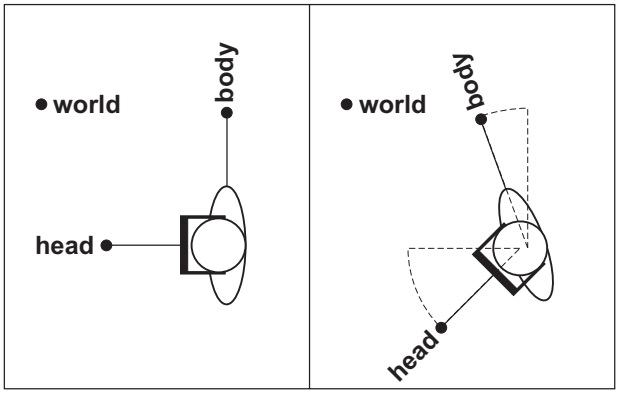

Figure 2. Influence of head and body rotation on the location of head-, body- and world-stabilized displays.

In mobile AR, three different presentation mechanisms have been identified [2]: *Head-stabilized*, where information is fixed to the users viewpoint, *Body-stabilized*, where information is fixed relative to the users body position, and *World-stabilized*, where information is registered with real world locations (see figure 2).

Applications of AR in mobile computing often follow a distinction between augmentation in a far field that is world-stabilized and a near field that is typically either head-stabilized or - less often - body-stabilized (see figure 3). There are also applications using a mixture of both approaches. In the far field both 2D and 3D annotations are common [1, 12]. In the near field there is a preference of 2D or simple 3D information that can only be browsed or navigated but not directly be manipulated [6, 14, 25]. There are also applications mixing and connecting annotations in both fields [10].

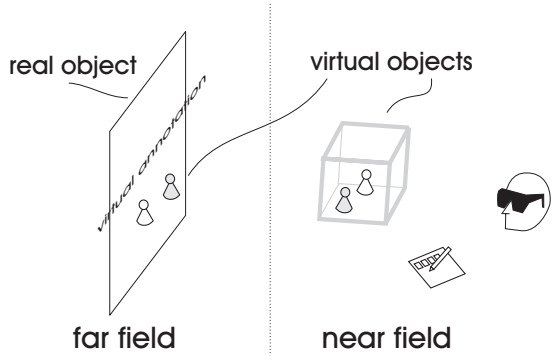

Figure 3. Far field objects may be world-stabilized and rendered monoscopic. Near field objects can be directly manipulated.

Lack of accurate world referenced tracking systems for the users location make direct interaction with virtual objects hard or impossible. Therefore world-stabilized annotation is easier to achieve in the far field because the tracking of the users position and orientation need not be as accurate as in the near field. Also the performance of mobile computers did not allow reasonable stereoscopic rendering with satisfying performance until recently. Either 3D objects that are positioned further away from the user were used or the user was limited to indirect interaction, as both strategies do not require stereoscopic rendering. With the recent advances in portable computer systems, we will show that applications with rich direct manipulation of 3D objects in the near field are becoming feasible.

1.2. Collaborative Augmented Reality

One of the main disadvantages of desktop CSCW applications are functional and cognitive seams that separate users from each other and from their tools [11]. AR removes these seams by embedding computer-based tools in the users' real environment [4]. In collaborative AR, co-located users can experience a shared space that is filled with both real and virtual objects. Moreover, three-dimensional interaction opens up new application areas such as collaborative computer aided design.

Different display platforms can be used to establish a collaborative AR, e.g. optical see-through HMDs in our own *Studierstube* [21, 22] system, video see-through HMDs in the Shared Space work [3], or hand-held displays in Transvision [19]. Other approaches constrain collaborative AR to special active surfaces [20] or used it in a heterogeneous environment, e.g. EMMIE [5] or metaDesk [26]. Finally, collaborative AR can also enhance remote presence, e.g. through video conferencing [2] or mixed indoor/outdoor user interfaces [9].

However, with the exception of the mobile portion of [9] all these systems work in tethered stationary environments, and only allow a limited degree of spontaneous collaboration, a restriction that we aim to overcome.

1.3. Contribution

In this paper we explore the possibilities of a mobile 3D workspace for collaborative augmented reality. Our system is a descendant of the *Studierstube* AR platform and is characterized by the following properties:

- A mobile platform allows full stereoscopic real-time display. A freely configurable tracking system allows fusion of arbitrary sensors that complement each other to provide 6DOF manipulation of 3D objects in the near field, i.e. within arm's reach. With our 3D user interface management system, users can arrange multiple arbitrary 3D applications around their body and carry them along.

- Multiple users can naturally collaborate in an augmented shared space. Instantaneous collaboration can be established as users wearing the system meet. We demonstrate collaboration between a mobile and a stationary user that is established as the mobile user walks into the *Studierstube* lab. The underlying distributed system transparently synchronizes graphical and application data and even streams running 3D applications on the fly to newly joining members of a workgroup.

Several challenges need to be addressed to realize the described situations. Distributed applications need real time synchronization of shared data to present the same state to all users. Stationary and mobile tracking systems vary considerably, still the delivered data has to be exchanged between the different platforms and integrated. Limited capabilities of mobile platforms may require applications to provide degraded versions of their content. All of these parameters need to be negotiated instantly when the mobile user joins a shared space.

While the current system is definitely a prototype and can be improved in many ways, it demonstrates the potential of merging AR, mobile computing, and CSCW, yielding mobile collaborative augmented reality.

2. Application scenarios

A mobile collaborative AR system has as many applications as there are reasons to have a conversation. Some of the actual scenarios we are investigating are given here. As can be expected, they are typically concerned with 3D CSCW applications.

Meeting in indoor environment. Multiple users meet in a lab or conference room to discuss and work together with the aid of augmented reality. Some users may work with local stationary devices including projection walls and workbenches while others bring their own mobile equipment along. Depending on the users' preferences, all devices may share content and reference coordinates, or they may be separated.

An important feature is that the number of participating components (and users) may change at runtime. Thus, late-comers may enter the shared information space, bringing with them their own devices, which automatically log into the distributed system and synchronize with the ongoing session.

Meeting in outdoor environment. Imagine two architects meeting outdoors to discuss a building project at location. Both users are equipped with a mobile wearable workspace kit, providing them with access to their sketches, notes and modeling tools. As the two architects approach each other, their kits connect, setting up a virtual "place" in which collaboration is possible. The users can now show each other any data they have brought along, as well as modify existing designs. They can also populate the planned location with building "impostors", which are full 3D objects.

3. User interface

While the computational power for stereoscopic rendering and computer vision is becoming available in mobile computer systems, the size and weight of such systems is still not optimal. Nevertheless, our setup is solely build from off-the-shelf hardware components to avoid the effort and time required for building our own. On one hand this allows us to quickly upgrade old devices or add new ones and to change the configuration easily. On the other hand we do not obtain the smallest and lightest system possible.

3.1. Hardware

The most powerful portable graphics solution available currently is a PC notebook equipped with a *NVidia GeForce2Go* video chip. The device has a 1GHZ processor and runs under Windows 2000. We also added a wireless LAN network adapter to the notebook to enable communication with our stationary setup or a future second mobile unit. It is carried by the user in a backpack.

As an output device, we use an *i-glasses* see-through stereoscopic color HMD. The display is fixed to a helmet worn by the user. Moreover, an InterSense InterTrax^2 orientation sensor and a web camera for fiducial tracking of interaction props are mounted on the helmet.

The main user interface is a pen and pad setup using a *Wacom* graphics tablet and its pen. Both devices are optically tracked by the camera using markers. The 2D position of the pen (provided by the Wacom tablet) is incorporated into the processing to provide more accurate tracking on the pad itself. Figure 4 gives an overview of the setup. See section 5 for details on tracking.

3.2. User interface management software

As our software platform we use *Studierstube 2.1* [21], a user interface management system for AR based on but not limited to stereoscopic 3D graphics. It provides a multi-user, multi-application environment, and supports a variety of display devices including stereoscopic HMDs (for an example, see section 6.2). It also provides the means of 6DOF interaction, either with virtual objects or with user interface elements registered with and displayed on the pad.

Applications are implemented as runtime loadable objects executing in designated volumetric containers, a kind of 3D window equivalent. While the original stationary *Studierstube* environment allowed a user to arrange multiple application in a stationary workspace, our mobile setup

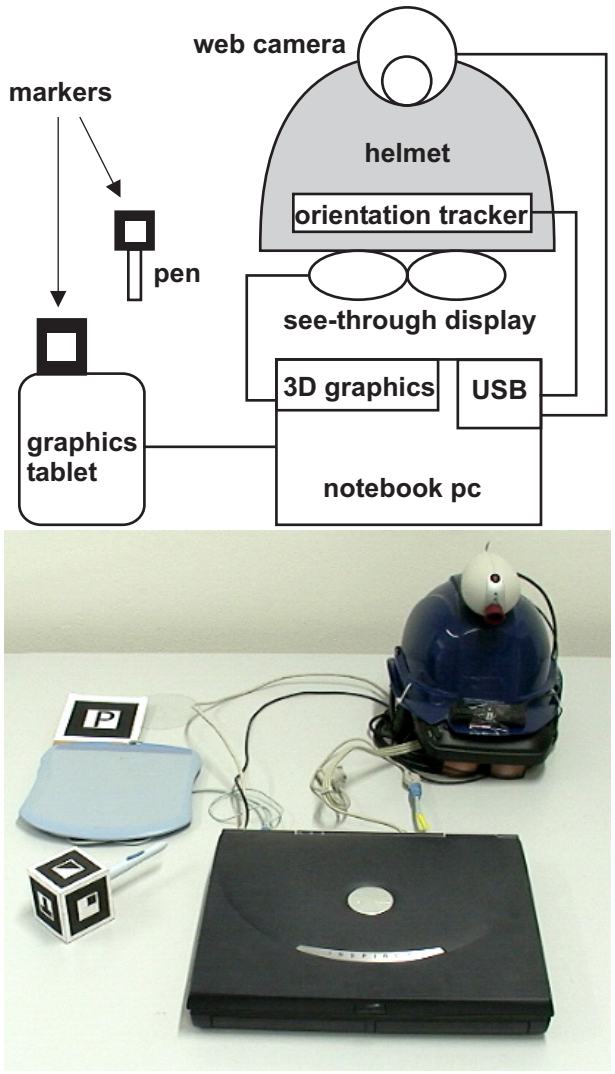

Figure 4. All components of the mobile setup are commercial items: (top) hardware design diagram, (bottom) picture of the components

with body-stabilized display allows to arrange 3D information in a *wearable* workspace that travels along with a user. Applications stay where they are put relative to the user, and can easily be accessed anytime, aided by proprioception and spatial memory [16]. This idea has been tried with 2D interfaces [7], but our implementation has more resemblance to the ToolSpaces of Pierce et al. [17].

Our user interface management system is also capable of managing multiple locales, which can contain any number of graphical objects. Locales are important for multi-user or multi-display operation. For example, each mobile user will require a separate wearable workspace that defines a distinct locale (coordinate system). As one user moves about, a second user's locale will be unaffected, but the second user will be able to see movement of the graphical objects contained in the first user's locale. For effective collaboration, it will in most cases be necessary to add a third stationary locale, that contains graphical applications that both users should work with (see also section 6.2).

Support for multiple users requires our 3D user interface management system to build upon a distributed graphics platform, which is described in the following.

4. Distribution

4.1. Distribution runtime software

Our software development environment is realized as a collection of C++ classes built on top of the Open Inventor (OIV) toolkit [24]. At the core of OIV is an object-oriented scene graph storing both geometric information and active interaction objects.

Like *Studierstube*, most DVE systems use a scene graph for representing the graphical objects in the application, but many systems separate application state from the graphical objects. To avoid this "dual database" problem [15], we introduce a distributed shared scene graph using the semantics of distributed shared memory. Distribution is performed implicitly through a mechanism that keeps multiple local replicas of a scene graph synchronized without exposing this process to the application programmer or user. Our OIV extension – Distributed Open Inventor (DIV) [8] – uses OIV's notification mechanism to distribute changes.

Our software system uses object-oriented runtime extension through subclassing. New node classes for OIV are loaded and registered with the system on the fly. Using this mechanism, we can take the scene-graph based approach to its logical consequence by embedding applications as nodes in the scene graph. Applications in *Studierstube* are not written as monoliths linked with a runtime library, but as new application classes that derive from a base application node. Application classes are loaded as binary objects on the fly during system execution, and instances of application objects are embedded into the scene graph. Naturally, multiple application nodes can be present in the scene graph simultaneously, which allows convenient multitasking.

Application classes are derived from an application foundation class that extends the basic scene graph node interface of OIV with a fairly capable application programmer's interface (API). This API allows convenient management of 3D user interface elements and events, and also supports a multiple-document interface – each document gets its own 3D window. Multiple documents are implemented through application instances embedded as separate nodes in the scene graph. However, they share a common application code segment, which is loaded on demand.

As the scene graph is distributed using DIV, so are the applications embedded in it. A newly created application instance will be added to all replicas of a scene graph, and will therefore be distributed. The programming model of making application instances nodes in the scene graph also implies that all application specific data – i. e., data members of the application instance – are part of the scene graph, and thus are implicitly distributed by DIV.

4.2. Application migration

Migrating applications between different hosts is an important feature of a distributed virtual environment. Building on *Studierstube's* distributed architecture and applications embedded as nodes in the scene graph, application migration is straight forward: All application state is encoded in the scene graph through the application node and its contained sub graph. Marshaling an arbitrary scene graph into a memory buffer is a standard operation of OIV (SoWriteAction). The application is marshaled, so its complete live state – both graphical and internal – is captured in a buffer, and can be transmitted over the network to the target host (using a reliable TCP connection), where it is unmarshaled (SoDB::readAll) and added to the local scene graph, so it can resume operation.

If the source copy is removed, the application will completely migrate to the destination host. For simple replication at the destination host, this step can be omitted. In both cases the destination host must load the application's binary object module if not already present in memory (the binary must either be available at the destination host, e. g., via a shared file system, or must be sent along with the marshaled application). Using migration, several scenarios can be supported which are interesting for CSCW applications.

Late joining. When hosts are added to a *Studierstube* session after the distributed system is already executing, it is necessary to build a copy of the replicated application instances at the new host. This is easily achieved through the application migration mechanism described above using the variant that does not delete the source application instance.

Early exit. The opposite operation to late joining of a host is early exit, where one host stops operation of the distributed system while the remaining hosts continue to execute. In this case, no application migration is necessary, the exiting host simply deletes its application instances.

5. Tracking

5.1. Tracking software

Mobile AR requires significantly more complex tracking than a traditional VR application. Unfortunately, we have not found an existing tracking software package that allows for the high degree of customization we need, yet is easy to use and extend. To overcome this situation,we have developed *OpenTracker* [18], an open software architecture for the different tasks involved in tracking input devices and processing multimodal input data.

In a typical VR or AR application tracking data passes through a series of steps. It is generated by tracking hardware, read by device drivers, transformed to fit the requirements of the application and send over network connections to other hosts. Different setups and applications may require different subsets and combinations of the steps described but the individual steps are common among a wide range of applications. Examples of such invariant steps are geometric transformations, Kalman filters and data fusion of two data sources. While experimenting, this sequence may frequently be adjusted.

The main concept behind *OpenTracker* is to break up the whole data manipulation into these individual steps and build a data flow network of the transformations. Building on this idea, *OpenTracker* offers the following:

- An object-oriented approach to an extensive set of sensor access, filtering, fusion, and state transformation operations
- Behavior specification by constructing graphs of tracking objects (similar in spirit to scene graphs or event cascades) from user defined tracker configuration files
- Distributed simulation by network transfer of events at any point in the graph structure
- Decoupled simulation by transparent multi-threading and networking
- both a time-based and event execution based model
- application independence

The framework's design is based on XML, taking full advantage of this new technology by allowing to use standard XML tools for development, configuration and documentation. For example, figure 5 was automatically generated from the XML based tracker configuration file used for the experiments described in section 6.1.

Through its scripting capability (XML tracker configuration files) as well as easy integration of new tracking features, *OpenTracker* encourages exploratory construction of complex tracking setups, as will be shown in the following.

5.2. Tracking configuration

The tracking of the user and the interaction props is achieved by combining data from various sources. The *OpenTracker* component receives data about the users head

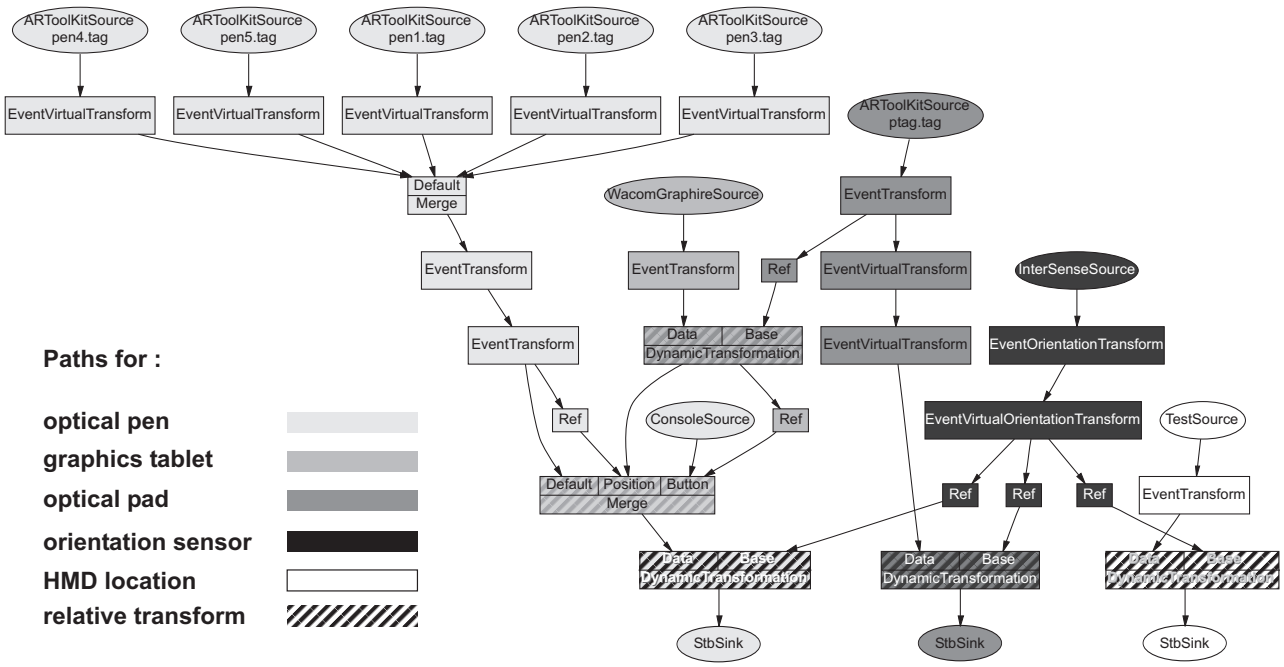

Figure 5. The data flow graph for the mobile tracking setup was automatically generated from the XML description

orientation from the InterTrax^2 sensor to provide a coordinate system with body stabilized position and world stabilized orientation.

Within this coordinate system the pen and pad are tracked using the video camera mounted on the helmet and ARToolKit [13] to process the video information. Because the video camera and the HMD are fixed to the helmet the transformation between the cameras and the users coordinate system is fixed and determined in a calibration step.

The pad is equipped with one marker. This is enough for standard operation, where the user holds it within her field of view to interact with 2D user interface elements displayed on the pad. The pen, however, is equipped with a cube featuring a marker on the five sides that are not occluded. This allows to track the pen in almost any position and orientation. Moreover whenever the user touches the pad with the pen the more accurate information provided by the graphics tablet is used to set the position of the pen with respect to the tablet.

The data flow graph describing the necessary data transformations is shown in figure 5. Round nodes at the top are source nodes that encapsulate device drivers. The round nodes at the bottom are sinks that copy the resulting data to the AR software. Intermediate nodes receive events containing tracking data, transform it and pass it on, downwards. An important type of transformation is the relative transformation that takes input from two different devices and interprets the location of one device relative to the location of the other (called the *base*).

Different colors denote paths through the graph that describe how the tracking data for different devices are processed. Relative transformations are marked by cross stripes in the color of the two paths connecting. For example, the *optical pen* path describes the five markers that are each transformed to relate the pen point location. This information is merged, then further transformed. After another merge with data from the graphics tablet, it is once more transformed to the reference system established by the orientation sensor.

Similarly, the *optical pad* path describes the computation to obtain the location of the pad. As a side effect, the *optical pad* information is used at one step to transform the 2D information from the *graphics tablet* path to the actual pen position which is subsequently merged with the pure optical information.

Finally the white *HMD location* path is used to provide information about the head location. The TestSource node's task is to provide a constant value which is then transformed by the orientation sensors.

We would like to note that using a visual XML editor, this complex configuration was created without writing a single line of code.

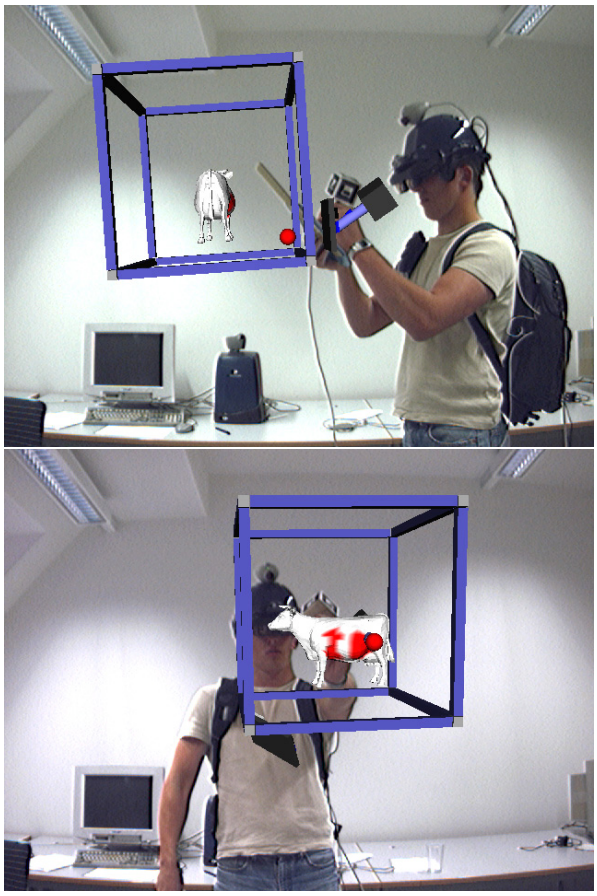

Figure 6. A user interacting with the paint application : (top) selecting the desired drawing color on the pad, (bottom) after moving the application window to another position.

6. Experiments

To illustrate the possibilities of the described system, we set up two different situations outlined in the following report. They use only simple demonstration applications but show the important properties such as direct manipulation and collaboration between users.

6.1. Mobile user

The mobile setup in the configuration described in section 3 is already capable of providing a workspace located around the user. This space moves along with the user, but its orientation is world stabilized, due to the use of the orientation tracker to establish head rotation (section 5). Within this workspace the full set of functions of the *Studierstube* framework is available. The user can place several applications and carry them around with himself. He can interact directly using the pen with displayed 3D objects or manipulate 2D user interface elements on the pad.

Figure 6 shows some pictures of a user interacting with a simple 3D painting application. Using 2D interfaces presented on the pad the user selects the painting color. Then he grabs the brush, shown as a red sphere, and paints the object in the selected color. He can also manipulate the 3D window itself, e.g. place it somewhere else in the personal workspace.

6.2. Mobile/stationary collaboration

This setup investigates the collaboration between a mobile and a stationary user. Both users share a locale for common collaborative applications.

The stationary user is tracked using a tethered magnetical tracking system driven by a dedicated tracking server that communicates the tracking data to the other hosts via a multicast group. The users display, a stereoscopic Sony Glasstron HMD, is driven by a desktop PC rendering host. The mobile user is equipped with the setup described in section 3. A third rendering host featuring a video camera was used to render a third person view over a video image to document the setup. Figure 7 displays the configuration.

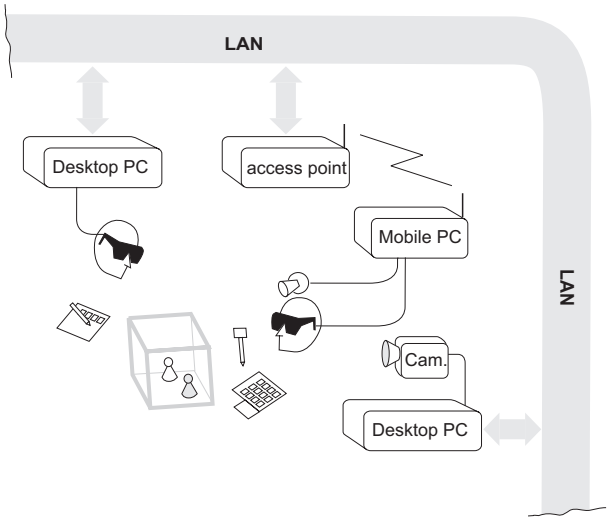

Figure 7. In the mobile/stationary interaction setup, three host computers connected to different input and output devices share a 3D scene via the network

While the stationary user's tracking is straightforward, the mobile user has to be correctly registered within the stationary users coordinate system to use the same locale as the stationary user. This is achieved by an additional optical marker that is placed within the environment. Its position within the reference frame of the magnetic tracker is registered. This establishes an anchor for transforming the loca-

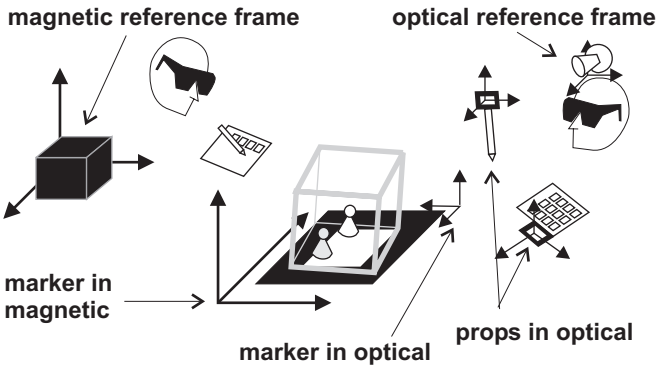

Figure 8. Several coordinate reference frames are used to register mobile user in the stationary setup

tions generated by the optical tracking of the mobile user to the magnetic reference frame. Figure 8 gives a schematic overview of the involved coordinate systems.

An *OpenTracker* standalone server running on the mobile unit tracks the anchor markers location relative to the camera. Inverting this location information yields the cameras location relative to the marker. Using this information and the registration data of the marker it computes the users location and subsequently that of the interaction props within the magnetic tracker reference frame and sends it to the other hosts via a second multicast group. Note that all three hosts receive tracker data from both multicast groups (optical/inertial and magnetic) and use OpenTracker components to combine the data for correct display and interaction of all tracked elements.

All three hosts use *Studierstube* to render the users view. They are configured individually for the different needs of the output devices (e.g. the Sony Glasstron uses a field-interleaved stereo video signal, whereas the Virtual IO-Glasses device uses a line-interleaved stereo signal). Each run an instance of the shared chess application and use the tracking data communicated via the two multicast groups to render the users actions properly.

The application migration features described in section 4.1 allow the mobile user to enter the workspace and late-join the running chess game. The current state of the application is streamed to the mobile kit upon connecting to the local network and the kit becomes another host participating the ongoing session.

Figure 9 shows the scene from a third person view. It was generated by the third host, rendering a view over a video camera background. On the table, behind the board one can see the Wacom pad and the notebook driving the mobile users view.

Figure 9. A game of AR Chess demonstrates collaboration of the stationary user (left) and the mobile user (right) as seen by the "documentation camera" user

6.3. Performance

During these experiments we achieved a display frame rate of 10 fps and a tracking frame rate of about 6 fps for the optical tracking. We found that the sum of the frame rates was limited by computational power to about 16 fps. Therefore we needed to fine tune the systems performance by limiting the optical tracking rate or the display rate. We choose to limit the optical tracking frame rate to enhance the quality of the displayed environment.

7. Conclusions and future work

We describe a mobile collaborative AR setup that features stereoscopic rendering and direct manipulation of the augmented objects. An advanced user interface management system allows the user to work with different applications presenting 2D and 3D user interfaces at the same time. A personal workspace surrounding the user allows her to organize the applications spatially and to directly interact with them in a natural way.

The distribution features allow collaboration between users on different hosts. Situations such as late-joining and migrating of applications between hosts are supported. These are important for naturally integrating mobile users into collaborative work situations.

A flexible tracking software component allows to build complex and hybrid tracking setups by the means of writing a configuration file. Thus other devices can be easily included to further enhance the quality of tracking.

The current systems hardware is very much a prototype

– many improvements are possible. One problem is that tracking of the interaction props is still too inaccurate and slow to provide unencumbered and natural use. While the monoscopic principle of ARToolKit used to track the props has the advantage that it is modest in the required resources (a single web cam is sufficient), a dedicated stereo tracking system, possibly with its own computing appliance, may greatly enhance tracking quality. We also incorporate more wireless devices as they become available (e. g., Bluetooth input devices).

Also the system software needs to address further challenges. Automatic adaption to the available tracking systems and negotiable platform capabilities are aspects that are not present yet.

Future applications for this system will include location aware computing to provide augmentation at world referenced locations. A mobile user may walk around a building and collect data from such "special" locations or change application parameters at location. This requires interaction with applications fixed to the location and data transfer between the mobile users workspace and the world referenced locations.

Collaboration including mobile platforms generates situations where users late-join a session and therefore are in the need of being updated to the current state of applications. A mobile user may also bring an instance of an already running application into the session, requiring to provide options to synchronize the state of different instances. When a user wants to leave a collaborative session, he should be provided with options to take a copy of the applications with him. Research into providing user interfaces for such situations is part of planned future work.

Acknowledgments

This project was sponsored by the Austrian Science Fund *FWF* under contract no. P14470-INF. Many thanks to Ivan Viola and Matej Mlejnek for their contribution to the implementation and work on the setup, Gerd Hesina for his work on the distributed system, and Hannes Kaufmann for helping out during production of this paper.

References

[1] R. Behringer, C. Tam, J. McGee, S. Sundareswaran, and M. Vassiliou. A wearable augmented reality testbed for navigation and control. In *Proc. ISAR 2000*, pages 12–19, Munich, Germany, October 5–6 2000. IEEE and ACM.

[2] M. Billinghurst, J. Bowskill, J. Morphett, and M. Jessop. A wearable spatial conferencing space. In *Proc. ISWC'98*, Pittsburgh, Penn., USA, October 19–20 1998.

[3] M. Billinghurst, S. Weghorst, and T. Furness. Shared space: An augmented reality interface for computer supported collaborative work. In *Proc. of Collaborative Virtual Environments Workshop '96*, Nottingham, Great Britain, September 19–20 1996.

[4] M. Billinghurst, S. Weghorst, and T. F. III. Wearable computers for three dimensional CSCW. In *Proc. ISWC '97*, Cambridge, Massachusetts, USA, October 13-14 1997.

[5] A. Butz, T. Höllerer, S. Feiner, B. MacIntyre, and C. Beshers. Enveloping users and computers in a collaborative 3d augmented reality. In *Proc. IWAR'99*, pages 35–44, San Francisco, CA, USA, October 20–21 1999. IEEE.

[6] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban enviroment. In *Proc. ISWC'97*, pages 74–81, Cambridge, MA, USA, October 13–14 1997.

[7] S. Feiner and A. Shamash. Hybrid user interfaces: Breeding virtually bigger interfaces for physicall smaller computers. In *Proc. UIST'91*, pages 9–17, November 1991.

[8] G. Hesina, D. Schmalstieg, and W. Purgathofer. Distributed open inventor : A practical approach to distributed 3D graphics. In *Proc. ACM VRST'99*, pages 74–81, London, UK, December 1999.

[9] T. Höllerer, S. Feiner, T. Terauchi, G. Rashid, and D. Hallaway. Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system. *Computer & Graphics*, 23(6):779–785, 1999.

[10] T. Höllerer and J. Pavlik. Situated documentaries: Embedding multimedia presentations in the real world. In *Proc. ISWC'99*, pages 79–86, San Francisco, CA, USA, October 18–19 1999.

[11] H. Ishii, M. Kobayashi, and K. Arita. Iterative design of seamless collaboration media. *Comm. of the ACM*, 37(8):83–97, August 1994.

[12] S. Julier, M. Lanzagorta, Y. Baillot, L. Rosenblum, S. Feiner, and T. Höllerer. Information filtering for mobile augmented reality. In *Proc. ISAR 2000*, pages 3–11, Munich, Germany, October 5–6 2000. IEEE and ACM.

[13] H. Kato and M. Billinghurst. Marker tracking and HMD calibration for a video-based augmented reality conferencing system. In *Proc. IWAR 99*, San Francisco, USA, October 1999.

[14] S. Long, D. Aust, G. Abowd, and C. Atkeson. Cyberguide: Prototyping contextaware mobile applications. In *Proc. CHI'96*, 1996.

[15] B. MacIntyre and S. Feiner. A distributed 3D graphics library. In *Proc. ACM SIGGRAPH '98*, pages 361–370, Orlando, Florida, USA, July 19–24 1998.

[16] M. R. Mine, F. P. Brooks, Jr., and C. H. Sequin. Moving objects in space: Exploiting proprioception in virtual-environment interaction. In *Proc. ACM SIGGRAPH '97*, pages 19–26, 1997.

[17] J. S. Pierce, M. Conway, M. van Dantzich, and G. Robertson. Toolspaces and glances: Storing, accessing, and retrieving objects in 3D desktop applications. In *Proc. 1999 Symposium on Interactive 3D Graphics*, pages 163–168, 1999.

[18] G. Reitmayr and D. Schmalstieg. OpenTracker – an open software architecture for reconfigurable tracking based on XML. In *Proc. IEEE Virtual Reality 2001*, pages 285–286, Yokohama, Japan, March 13–17 2001.

[19] J. Rekimoto. Transvision: A hand-held augmented reality system for collaborative design. In *Proc. VSMM'96*, pages 18–20, Gifu, Japan, September 1996.

[20] J. Rekimoto and M. Saitoh. Augmented surfaces: A spatially continuous workspace for hybrid computing. In *Proc. CHI'99*. ACM, 1999.

[21] D. Schmalstieg, A. Fuhrmann, and G. Hesina. Bridging multiple user interface dimensions with augmented reality. In *Proc. ISAR 2000*, pages 20–29, Munich, Germany, October 5–6 2000. IEEE and ACM.

[22] D. Schmalstieg, A. Fuhrmann, Z. Szalavari, and M. Gervautz. Studierstube – an environment for collaboration in augmented reality. In *Proc. of Collaborative Virtual Environments Workshop '96*, Nottingham, UK, September 19–20 1996.

[23] T. Starner, S. Mann, B. Rhodes, J. Levine, J. Healey, D. Kirsch, R. Picard, and A. Pentland. Augmented reality through wearable computing. *Presence*, Special Issue on Augmented Reality(4):386–398, 1997.

[24] P. Strauss and R. Carey. An object oriented 3D graphics toolkit. In *Proc, ACM SIGGRAPH'92*. ACM, 1992.

[25] B. H. Thomas, V. Demczuk, W. Piekarski, D. H. epworth, and B. Gunther. A wearable computer system with augmented reality to support terrestrial navigation. In *Proc. ISWC'98*, pages 168–171, Pittsburgh, USA, 1998.

[26] B. Ullmer and H. Ishii. The metaDESK: Models and prototypes for tangible user interfaces. In *Proc. UIST '97*, pages 223–232, Banff, Alberta, Canada, October 1997. ACM Press.