Preprint for Computers&Graphics 23 (6), 1999, 875-881
Refer to www.elsevier.com/locate/cag for final version

# Accurate overlaying for mobile augmented reality.

**W. Pasman* , A. van der Schaaf, R.L. Lagendijk, F.W. Jansen**

*UbiCom project, Faculty of Information Technology and Systems,Delft University of Technology, The Netherlands*

**Abstract**

Mobile augmented reality requires accurate alignment of virtual information with objects visible in the real world. We describe a system for mobile communications to be developed to meet these strict alignment criteria using a combination of computer vision, inertial tracking and low-latency rendering techniques. A prototype low-power and low-latency renderer using an off-the-shelf 3D card is discussed.

*Keywords: wearable, mobile, augmented reality, latency, alignment, information-on-the-spot*

## 1. Introduction

Mobile augmented reality [1,2] is a relatively new and intriguing concept. The ability of augmented reality [3] to present information superimposed on our view of the world opens up many interesting opportunities for graphical interaction with our direct environment. Combining this with mobility further increases the potential usage of this technology.

However, the technical problems with mobile augmented reality are just as great. First, it is necessary to acquire very accurate viewpoint measurements, because errors cause virtual objects to be merged at a wrong position in the real world, and a mismatch between virtual and real objects will be directly visible. Second, even more than other head-mounted display systems, augmented reality displays require an extremely low latency to keep the virtual objects at a stable position. Third, mobile augmented reality requires a low-power approach, limiting the amount of hardware that can be used to solve these problems.

### *1.1. Estimation of requirements*

An error in the measurement of the viewpoint of the user causes a deviation of the location of the displayed virtual objects from their expected position. The effect of a measurement error depends on its direction and the distance to the virtual and real objects. For example, positional errors will cause big changes in the virtual image of nearby objects, while it hardly affects objects that are far away.

Very little research has been done on the requirements for augmented reality. An accuracy corresponding to the visual acuity of the human, which is half an arc-minute and sub-millimeter, seems to be the ultimate goal. But such an accuracy is far more than required for most applications. In general the required accuracy depends on the consequences of an error, and the consequences of errors will depend on the task at hand. For example for medical purposes an misalignment of a millimeter may already be fatal [4], while for tourist information an error of a few meters and a few degrees will be acceptable.

Although literature [5,6,7,9,10] gives no conclusive numbers we think that many tasks can be supported with an AR system having a positional accuracy of a few millimeters within about 2 meters from the observer, and a rotational accuracy of a fraction of a degree. For comparison, the pixels of a 640x480 display projected in a 20°x 35° field of view at a distance of 1 meter have a size of about 0.8 millimeters. Here we think of applications such as remote maintenance and repair of complex machines,

* Corresponding author: dr. W. Pasman
Tel.: (+31)-15-2783107. Fax: (+31)-15-2787141.
email: W.Pasman@twi.tudelft.nl

assembly of complex systems, architectural support, tourist information systems, military applications and entertainment (see [5,6]). Only a few, especially medical AR applications, really require submillimeter accuracy [4].

### *1.2. Accuracy and latency requirements*

An important source of alignment errors is the time difference between the moment the observer moves and the moment the image corresponding to his new position is displayed. This time difference is called end-to-end latency. End-to-end latency is important because head rotations can be extremely fast and cause significant changes in the visible scene (Figure 1).

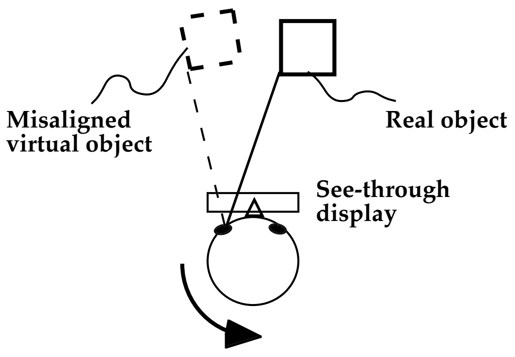

Fig. 1. A virtual object is displayed in overlay with a real object. When the user rotates his head to the left, the real object immediately moves to the right of his visual field. The virtual object, on the contrary, lags behind the first, and therefore stays in the old position relative to the display. Only after some time the virtual object is re-rendered in alignment with the real object.

Padmos and Milders [7] indicate that for immersive reality (where the observer can not see the normal world), the end-to-end latency should be below 40 ms. For augmented reality the requirements will be even higher. They suggest that the displacement of objects between two frames should not exceed 15 arcmin (0.25°), which would require a maximal latency of 5 ms even when the observer rotates his head with a moderate speed of 50°/s. Several other authors use a similar approach [5,6,8,9,10] and come to similar maximal latency times. Actually, during typical head motions speeds of up to 370°/s may occur [11], and for fighter pilots speeds of up to 2000°/s have been reported [12]. But it is not likely that observers rotating their head that fast will notice slight object displacements. Many authors suggest that 10 ms will be acceptable for AR [8,13,14].

For indoor tracking, the required positional and rotational precision are realisable, as current trackers are capable of a rotational accuracy in the order of 0.25° and a positional accuracy of 0.25 inch [15]. For outdoors, systems are still under development [16]. The latency requirement of 10 ms is extreme: it is an order of magnitude smaller than the latency of current systems.

In this paper we describe how the requirements could be met with a combination of several levels of position and orientation tracking with different relative and absolute accuracies, and several levels of rendering to reduce the complex 3D data to simple scenes that can be rendered just-in-time. In section 1 we first describe the context of our research, the Ubicom project, a multi-disciplinary project carried out at Delft University of Technology, which aims at the development of a system for Ubiquitous Communication. In section 2 to 4 we focus on the problem of image stabilisation and discuss latency issues related to position tracking and display. We summarize our system set-up in section 5 and conclude with describing the current state in section 6.

## 2. Ubicom system

The Ubicom System [17] is an infrastructure for mobile multi-media communication. The system consists of a backbone compute server, several base stations, and a possibly large number of mobile units (figure 2).

The base stations maintain a wireless (radio or infrared) link to the mobile units. The radio transmission is scheduled in the 17 GHz range and will account for approximately 10 Mbit/s of data bandwidth per user, enough to transmit compressed video with high quality. The cell size (distance between the base stations) is in the order of 100 meter: typically the distance between lampposts to which the base stations may be attached.

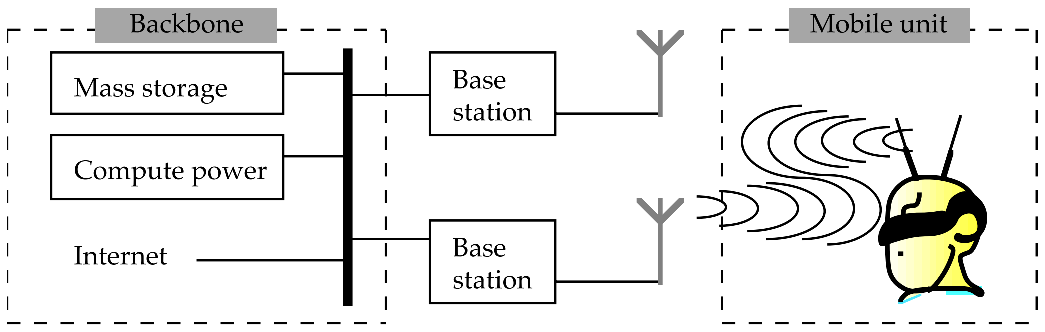

Fig. 2. Ubicom system setup. The mobile unit contains display, camera, and tracking devices, and is connected through a mobile link to one of several base stations. Memory and processing resources are limited in the mobile unit in order to reduce power consumption and extend battery life. Instead, the mobile connection is used to access resources like mass storage and compute power at the backbone.

The mobile unit consists of a receiver unit and a head-set. The head-set contains a light-weight head-mounted display that offers the user a mix of real and virtual information. This may be realised either by superimposing the virtual information on the real world or by replacing parts of the real world with virtual information. In the latter case we need partially visual blocking of the view on the outside world. In addition to the display facilities, the mobile unit will also have a light-weight video camera that is used for position tracking and to record video data. In order to keep the power consumption low, the head-set and receiver unit will only have limited processing and memory capabilities. Figure 3 shows a functional blockdiagram of the head-set and receiver unit.

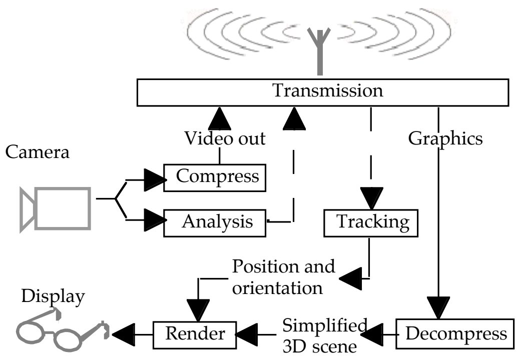

Fig.3. Diagram of the mobile unit. The camera at the mobile unit supports two main functions. First, the camera produces video, which is compressed and sent to the backbone for recording or distribution to other users. Second, the camera images are analysed to find landmarks that are used for position tracking. The actual matching of landmarks is computationally expensive and is done in the backbone. The backbone also supplies the mobile unit with simplified virtual scenes, which must be decompressed and processed before they can be displayed in overlay with the real world. Fast inertial tracking devices in the mobile unit measure the head-motion of the user and track the latest position and orientation of the mobile unit. This information is used for last-minute adjustments of the displayed graphics, such that these remain aligned with the real world.

## 3. Tracking

Central to the function of the mobile unit is the exact alignment of virtual information with the objects in the real world that the user is seeing. This requires that the exact viewing position and viewing direction of the user are known. Position as well as orientation tracking are therefore needed. Orientation tracking is much more critical than position tracking as a small rotation of the head will have a larger visual impact than a small movement to the left or right.

Tracking is done in three steps (Figure 4). A first position estimation is done using GPS or similar position detecting techniques. A possibility is to calculate the position relative to the base stations. A second level of position tracking is using object and scene recognition. Given a 3D description of the environment (e.g. a 3D GIS or CAD-model) and an initial position/orientation estimate, an accurate position and orientation can be calculated. However, the model data will only be available at the backbone and most of the calculations to derive the viewing position are likely to be performed at the backbone as well due to the amount of storage and processing power required for these calculations. Part of this computation could be offloaded to the active base station. The latency caused by first capturing a video image from the camera on the mobile unit, sending the image to the backbone, processing the image data at the backbone, and then transmitting the obtained viewing parameters back to the mobile unit will be substantial. This latency will be much larger than the latency requirement of the visual display. Therefore, a we have a third level of position tracking, anticipating on small position changes directly. In the third level, the movements are sensed with inertial trackers (accelerometers and gyros) and directly fed back to the display system. The high-latency but accurate measurements based on the camera measurements are used to compensate the drift in the inertial trackers.

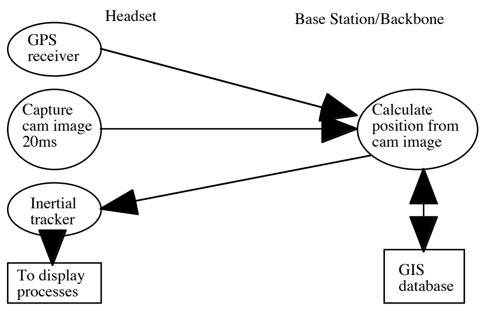

Fig. 4. Circles represent processes, boxes local memory and arrows the data flow. Position changes are directly determined with the inertial tracker. To compensate for drift, more accurate positions are calculated regularly in the backbone, based on GPS data and camera images.

## 4. Low-latency rendering

Given an accurate viewing position, a new virtual image has to be generated. Similar to the position and orientation and orientation calculation, the choice here is also whether to calculate each new image at the backbone with a powerful render engine and to transmit the image to the mobile unit over the wireless link, or to render the image directly on the mobile unit, avoiding the latency of the wireless link. Even for the second option, direct rendering at the mobile unit with standard rendering hardware, there will be a latency in the order of 50-100 ms, which is unacceptable.

We chose to use a similar mechanism as used for the tracking: a direct, but imperfect feedback mechanism within the mobile unit that is kept up-to-date and corrected with a slower system running in the backbone. There are three approaches to approximate the required feedback:

A first approach is to apply viewport re-mapping techniques [18,19] to the image data available in the mobile unit. With viewport remapping, the available image data is larger than the actually viewed image, and the new viewport is translated and/or scaled over this image to adapt quickly to the new viewing position and direction. Compensation for position changes is very limited.

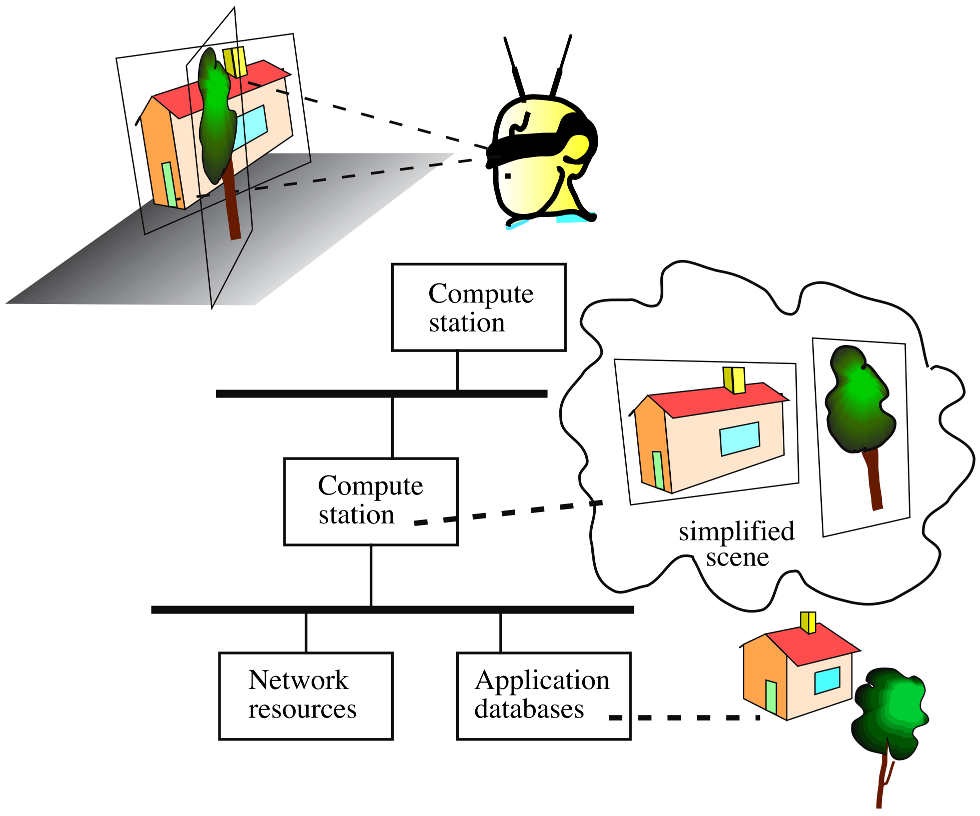

Fig. 5. The virtual scene is simplified in the backbone into a relatively small number of textures, depending on a recent viewpoint of the user. The mobile unit compensates for movements by deforming ("warping") and combining the textures.

A second approach, based on image warping techniques [20], allows further correction for parallax changes. It assumes that the available virtual and real-world data is segmented in layers of images, or "sprites". The viewed image can then be calculated by warping (stretching and shearing) and then merging the sprites (Figure 5). A problem with this technique is that there is insufficient info in the sprites to know exactly how they have to be warped.

A third approach is to have an extremely simplified 3D scene in the mobile unit. The scene should be so simple that it can be rendered with minimal latency. For distant objects the approach could effectively be the same as the second approach, but for nearby objects more geometry need be used. Much effort and finetuning will be required to avoid all latencies inherent to the heavy pipelining common to polygon rendering systems.

Analysis of these three possibilities shows that the lookup of image pixels uses by far most processing power. Even in the third case, if the scene contains a few hundred vertices less than 10% of CPU power is required for projecting the vertices into screen space, and this number can be reduced even further with dedicated hardware. Thus, the third approach requires only slightly more processing power than the first approach, but it is far more flexible gives better approximations of the new view. Therefore we chose to use the third approach.

## 5. First result

A first result concerns the low-latency rendering of a simplified 3D scene in the mobile unit. To get a low-latency rendering system, we split the display frame in four partitions and recalculate the position and orientation of the camera for each of these parts (Figure 6).

This approach allowed us to use relatively lightweight rendering hardware: a commercially available Voodoo2 3D accelerator card from Creative Labs [21], driven with MESA openGL [22] on top of Glide [23]. KURT Realtime Linux [24] was used to schedule the rendering process in sync with the displaying of the 3D card. We need realtime Linux because openGL calls can be done only from user-space processes under Linux, and because the Voodoo2 card can not be programmed to give interrupts at selected scanlines.

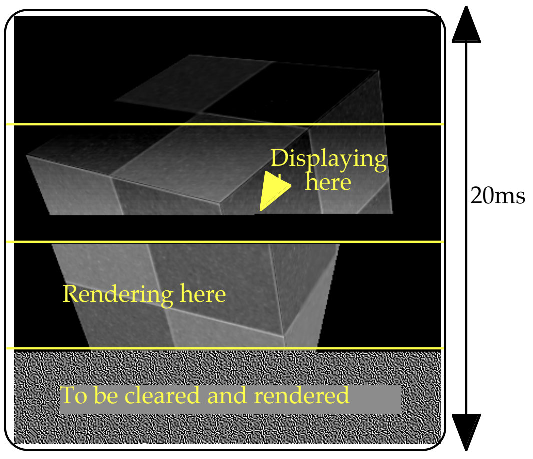

Fig. 6. Display time for the whole image is 20 ms (assuming a 50Hz display), excluding rendering. Dividing the image into partitions, and rendering just ahead of the partition being displayed, reduces the latency to 10ms (5 ms rendering and 5 ms display).

Currently we start rendering a new part at the moment the 3D card starts displaying the just-rendered part, but of course we could start later if we were able to predict how much time it will take to render the new part. This would allow for even lower latencies.

We currently have a frame rate of 60Hz, thus we have approximately 1/(4*60) = 4.1 ms to render the next frame part. The hardware we used is fast enough to clear the new part and draw a few hundred texture-mapped polygons in about 3 ms (depending on the size of the projected polygons), leaving some time for the other components of the system.

Our current setup implies that the latency is not constant for all displayed pixels. Instead, the first pixels in a part will have just over 4.1 ms latency, while the last pixels in the part will have just over 8.2 ms latency. The topmost frame part will have additional latency, because the vertical sync takes some extra time between displaying the bottom and top part. An artefact of the varying latency is that there will occur visual tearing at the boundary of two parts. This tearing is usually not disturbing, as the latency difference is small and only visible with objects moving extremely fast and in a horizontal direction relative to the display.

For actual latency measurements on our prototype, we used an oscilloscope (similar to [25]) to check the time between readout of the tracker and the actual output of the corresponding pixels to the VGA output of the 3D card. These measurements showed that the maximum latency is 8.8 ms and the average latency is about 6.35 ms, which matches our expectations.

This rendering method can be used in our wireless mobile unit. The mobile unit will have small batteries and therefore the power-hungry Voodoo2 cards may be not the best choice, but our low-latency rendering method can be implemented on virtually any 3D acceleration hardware. For example, 3D labs [26] sells the Permedia II 3D hardware using only 1.5W, and ATI [27] has similar hardware.

## 6. System overview

If we analyze the latency of the inertial tracking and corresponding image rendering, we come to the system shown in Figure 7.

In global, we have three paths to refresh the image in the head-set with increasing latency times and increasing accuracy: a path local to the mobile unit, a path from head-set to base station and back, and a path from mobile unit via base station to the backbone and back.

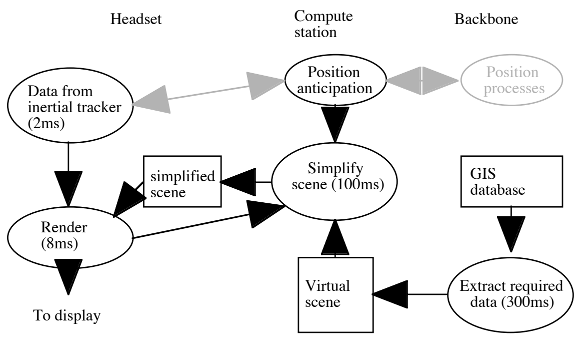

Fig. 7. Both the rendering and the tracking are distributed over mobile unit, compute station and backbone, giving different latencies for these parts of the rendering process.

In the mobile unit we minimise latency by using an inertial tracker (2 ms latency) and image rendering (at most 8 ms latency). The rendering is done just ahead of the display scanning, to avoid latency that might be caused by the refresh rate of the display.

In the base station, the virtual world information is simplified to get a scene that can be used in the mobile unit. Either the mobile unit itself requests for these images or the base station anticipates the need for new images and polygons from recent movement data passing through the base station. These new images and polygons will have a lag of about 200 ms when arriving at the mobile unit.

In the backbone there are two processes. The first calculates the viewpoint of the observer given camera images from the mobile unit and a GIS database. This process may be supported by GPS data acquired in the mobile unit, and may take up to 500 ms including all transmissions back to the mobile unit. The second process is the generation of a new simplified virtual world. Images and polygons generated from the new simplified virtual world model rendered at the base station will arrive at the mobile unit with a latency of about 1000 ms, one second.

## 7. Conclusions

We have proposed a multi-phase position and rendering approach to meet the severe alignment constraints associated with augmented reality. The current state of the research is that a hardware prototype system has been built from off-the-shelf components. We use a standard see-through head-mounted display and a standard inertial tracker. The system is not yet portable and wireless - we still need a power cable and a wire to a ceiling-mounted position tracker.

A low-latency rendering system has been implemented and is operational. The system is potentially lightweight and low-power. The average latency has been measured at about 6.35 ms, and we expect that this is low enough to reach our goal of 10 ms end-to-end latency.

We will use the prototype system to investigate the performance and whether the latency goals have been met, to do psychophysical measurements concerning latency, and to check where bottlenecks occur and whether the cpu load is acceptable. Thereafter we will continue to work on the scene simplification research and implementation, and on the vision-based position tracking.

## References

[1] S. Feiner, *KARMA*. Available Internet: www.cs. columbia.edu/graphics/projects/karma (1995).

[2] S. Feiner, B. MacIntyre, T. Höllerer, and A. Webster, A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. *Proceedings ISWC'97* (International Symposium on wearable computing, Cambridge, MA, October 13-14) (1997). Available Internet: www.cs.columbia.edu/ graphics/publications/ISWC97.ps.gz.

[3] P. Milgram, *Augmented Reality* (1995). Available Internet: http://vered.rose.utoronto.ca/people/anu_ dir/papers/atc/atcDND.html.

[4] M. Bajura, H. Fuchs and R. Ohbuchi, Merging virtual objects with the real world: Seeing ultrasound imagery within the patient. *Computer Graphics - Proceedings of the SIGGRAPH'92*, 26 (2), 203-210 (1992).

[5] R. T. Azuma, A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments,* 6, 4, 355 - 385 (1997). Earlier version appeared in Course Notes #9: Developing Advanced Virtual Reality Applications, ACM SIGGRAPH (Los Angeles, CA, 6-11 August 1995), 20-1 to 20-38.

[6] R. T. Azuma, *Registration errors in augmented reality* (1997). Available Internet: http://epsilon.cs.unc.edu/~azuma/ azuma_AR.html.

[7] P. Padmos and M. V. Milders, Quality criteria for simulator images: A literature review. *Human Factors*, 34 (6), 727-748 (1992).

[8] R. Azuma and G. Bishop, Improving static and dynamic registration in an optical see-through hmd. *Proceedings of the SIGGRAPH '94*, 197-204 (1994).

[9] M. Olano, J. Cohen, M. Mine and G. Bishop, Combatting rendering latency. *Proceedings of the 1995 symposium on interactive 3D graphics* (Monterey, CA, April 9-12), 19-24 and 204 (1995).

[10] R. L. Holloway, Registration error analysis for augmented reality. *Presence*, 6 (4), 413-432 (1997).

[11] U. List, Nonlinear prediction of head movements for helmet-mounted displays. *U.S. Air force Human Resources Laboratory*, Technical paper AFHRL-TP-83-45, December (1983).

[12] D. G. Aliaga, Virtual objects in the real world. *Commun. ACM* , 40, 3 (Mar), 49-54 (1997).

[13] S. R. Ellis and B. D. Adelstein, *Visual performance and fatigue in see-through head-mounted displays* (1997). Available Internet: http: //duchamp.arc.nasa.gov/research/ seethru_summary.html.

[14] H. J. G. de Poot, *Monocular perception of motion in depth*. Unpublished doctoral dissertation, Faculty of Biology, University of Utrecht, Utrecht, The Netherlands (1995).

[15] InterSense, *IS-600 Mark 2 precision motion tracker: Robust 6 degree-of-freedom motion tracking for simulation and training* (1999). Available Internet: http://www.isense.com/is600.html.

[16] E. Foxlin, M. Harrington and G. Pfeiffer, Constellation: a wide-range wireless motion-tracking system for augmented reality and virtual set applications. *Proceedings of the 25th annual conference on Computer Graphics*, 371-378 (1998).

[17] Ubicom,*Introduction to UbiCom* (1997). Available Internet: www.ubicom.tudelft.nl/program/default.asp.

[18] M. Regan and R. Pose, Priority rendering with a virtual address recalculation pipeline. *Proceedings of the SIGGRAPH'94* (Orlando, FL, 24-29 July). In Computer Graphics, Annual conference series, 155-162 (1994).

[19] W. R. Mark, L. McMillan and G. Bishop. Post-rendering 3D warping. *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Providence, RI, April 27-30), 7-16 (1997). Available Internet: www.cs.unc.edu/~billmark/i3dwww/i3dpaper-web.pdf.

[20] J. Lengyel and J. Snyder, Rendering with coherent layers. *Proceedings of the SIGGRAPH'97*, 233-242 (1997).

[21] Creative Labs, *3DBlaster Voodoo2* (1999). Available Internet: www.creative-labs.co.uk/techknow/graphic/voodoo2/ voodoo2_index.htm.

[22] Mesa, *The Mesa 3D graphics library* (1999). Available Internet: www.mesa3d.org.

[23] 3dfx Interactive, *Glide SDK V2.x for Linux* (1999). Available Internet: www.europe.3dfx.com/view.asp?IOID=109 .

[24] KURT, *KURT: The KU Real-Time Linux* (1999). Available Internet: http://hegel.ittc.ukans.edu/projects/kurt/ .

[25] M. C. Jacobs, A. Livingston and A. State, Managing latency in complex augmented reality systems. *Proceedings of the 1997 symposium on Interactive 3D graphics*, 49-54 (1997).

[26] 3Dlabs, *Permedia 2* (1999). Available Internet: www.3Dlabs.com/products/ p2.html.[27] ATI technologies, *ATI Rage LT pro* (1999). Available Internet: www.atitech.ca/ca_us/technology/hardware/rageltpro.html.