Realistic low-latency mobile AR rendering

W. Pasman, S. Persa and F. W. Jansen
Delft University of Technology, UbiCom project
Mekelweg 5, Delft, Netherlands
corresponding author: W.Pasman@twi.tudelft.nl

Abstract. When designing a system for mobile augmented reality, problems to be tackled concern tracking, rendering performance, end-to-end latency, battery usage, and communication bandwidth of the mobile platform. We developed an integral solution covering all these aspects, while still being manageable from the application's point of view. In this paper we outline the global layout of our system, and discuss a demo application projecting a statue on the campus.

Introduction

Today, various implementations for mobile AR exist. However, they are mainly focused on wayfinding and tourist guiding [1]. They often rely on simple line drawings and text, instead of full 3D rendering. Although such simple pictures can be used for architecture [2], the requirements are more severe than for tourist guiding: the virtual objects are larger in number and in size, the accuracy needs to be higher (depending on what the user is trying to do), and texture-mapped polygons instead of line drawings would be helpful in a large number of applications.

Optical AR, where the virtual objects are merged optically into the real world via a half-transparent mirror, is preferred for mobile outdoor AR applications. Video AR configurations, where the real world is recorded with a camera, and both virtual and real objects are shown on a monitor before the user's eyes, will degrade the user's view of the real world and give large latencies between the user's movement and the corresponding update of the display (**end-to-end latency**). This will decrease the quality of his view on the real world, which may decrease his performance, cause simulator sickness and even might be dangerous in case of a system failure.

A number of challenges rise when aiming for such a high-performance mobile AR system.

• Objects have to be positioned with absolute accuracy, because the virtual markers probably refer to something in the real world. In indoor immersive VR, for which most current tracking systems have been designed, absolute errors cause no immediate problems because all objects will be displaced the same amount. Highly accurate outdoor trackers do not exist yet.

• When using optical AR, the accuracy requirements directly imply requirements for the maximum end-to-end latency. However, even high-end graphics workstations today are optimized for maximum throughput and not for low latency.

• Virtual objects cannot just be rendered independent of the environment. If virtual objects have to be occluded properly by real-world objects, the rendering system needs to know about objects in the real world. Also, the system may need to be aware of other aspects of the environment, for instance for proper lighting of the virtual objects.

• The batteries are a point of concern for mobile AR applications. To save batteries, it is essential to carefully trade off the battery usage against wireless network load and final image quality. A first approach could be to move computations to the fixed backbone systems. However, the results of the computations still have to be brought back to the user, and if these results are large or latency-sensitive, the costs of the wireless link may be prohibitive.

We [3] have developed an overall approach to mobile AR. In this paper we outline parts in our system that are crucial for the accuracy of the final image: the tracking system, the renderer, the dynamic simplification system and the supporting database. Finally we present a sample architectural application where a virtual statue is merged into real world.

Previous work

We cannot discuss all relevant work here. Instead, we discuss existing work on latency, because of its far-reaching consequences for the system setup.
Latency is the most important source of misplacement of virtual objects in the real world [4]. Especially rotation of the head can be very fast, in the order of 300 degrees per second and faster if necessary, and in such a situation even a latency of 10 ms will result in a misplacement of 3 degrees (Figure 1). Every component in the chain between the moment the observer moves his head and the moment the new image is visible in the display has effect, so we have to consider all parts in this critical path. It has been shown experimentally that humans can detect latencies as low as 8 ms and probably less [5]. Fortunately, many tasks can be accomplished reasonably effective in presence of delays. For instance for immersive VR, latencies in the order of 40 ms are acceptable work with even relatively latency sensitive tasks as car driving [6].

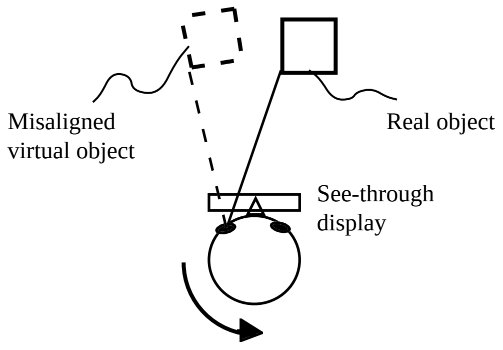

Figure 1: Head rotation is the most critical factor in the latency-accuracy relation. Here, the observer rotates his head counterclockwise while we try to align the virtual and real objects. The virtual object has a latency and will be displayed in a direction it should have been a short while ago. Therefore, the virtual object appears misplaced.

Configurations using optical AR are more latency sensitive than immersive VR configurations, because real world objects have no latency and delays of virtual

objects can be seen directly. For applications such as AR architecture, latencies in the order of 10 ms seem acceptable [7].
Not so much has been done about low latency rendering. Typically, rendering is done into a back buffer, which is swapped into the display card in sync with the vertical refresh of the screen. This mechanism gives the typical latencies of about 40 ms for the rendering and displaying alone, assuming a 50 Hz refresh. Often, the rendering runs on a lower refresh rate, and latencies of 100 ms are common. The latency of the tracker still comes on top of this. The most serious attempt we know of was presented in [8]. Their rendering pipeline gave delays of 17 ms, but their end-to-end latency was still about 50 ms.

With a plethora of different graphics applications that depend on motion-tracking technology for their existence, a wide range of interesting motion-tracking solutions have been invented. Surveys of magnetic, optical, acoustic, and mechanical tracking systems are available in [9, 10]. Each tracking approach has limitations, most have a short range and are not suited for outdoor use. Most trackers have latency in order of 100 ms. Hybrid systems attempt to compensate for the shortcomings of each technology by integrating multiple sensors to produce robust results. Hybrid systems that combine inertial tracking with position tracking from camera images seem most promising, because the inertial data can increase the robustness and computing efficiency of a vision system by providing a frame to frame prediction of camera orientation, while the vision system can correct for the accumulated drift of the inertial system. A number of such systems exist [11, 12, 13]. The approach of Berger [13] and Simon [14] is suited for video mixing with synthetic objects, but they give no clue about the loop execution time. The system presented by Behringer [12] can achieve high output rate due to the simple image processing of matching silhouette, but it is not operable in an urban environment. The image processing system described by Harris [27] can track simple models like a model Hornet aircraft, cylindrical objects and patches on the ground, but will fail in an urban environment, where multiple objects with complex wire-frame models are present in the image. The problem of line matching is best presented by Schmid [28], but unfortunately their approach uses stereo vision, and is very computation intensive. Presented measurements [11] show maximum angular errors of about 10 degrees, and a latency of 92 ms, which is quite far from our requirements.

Remote computation is desirable to save batteries on the mobile system (the **headset**). An extreme approach was taken in the InfoPad project [15], where the mobile system is only an X server and every interface action, down to mouse drag operations, are handled in the backbone. However, remote computation gives latency. Typical latencies are in the order of tens of milliseconds including all protocol layers that exist in current computer architectures. Furthermore, large images will take a lot of time to transmit, also increasing the latency. Thus, latency sensitive rendering processes will have to run in the headset.

In the following sections, we discuss the structure of those critical components within the Ubicom mobile AR system.

UbiCom Mobile AR System

The prototype system as under development within the Ubiquitous Communications (**UbiCom**) project is too complex to describe here in detail. We will discuss the rendering system, the projected tracking system and the GIS database system in some detail. For more details on various parts the reader is asked to refer to the UbiCom home page [3].

As discussed in the previous section, we aim at a system with a total latency of at most 10 ms. We alotted 2 ms to the tracking system and 8 ms to the rendering system (including frame buffers and display delays).

Rendering system

To make rendering latencies as low at 8 ms feasible at all, a number of low-level mechanisms have been implemented. A basic source of latency is the display. Assuming a typical refresh rate of 50Hz, it takes 20 ms to display a single frame. The time to render the frame will come on top of that. It is clear that it is not possible to reach the required latency by sequentially rendering and displaying a frame. Instead, our system renders only a part of the frame at a time just ahead of the display's raster beam. Our prototype implementation renders the frame in 4 slices, and has a combined rendering and display latency of 8 ms [16].

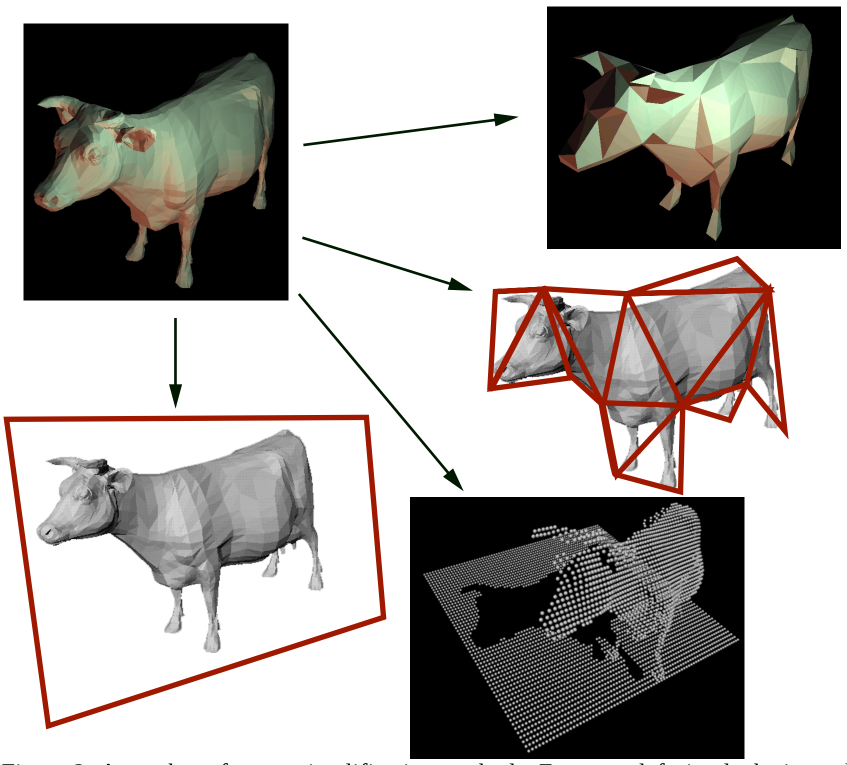

Figure 2. A number of scene simplification methods. From top left, in clockwise order: the original object, simplified polygon object, meshed imposter, IwD and simple imposter.

The number of polygons in the scene has to be reduced drastically in order to be manageable by the frontend and wireless link. This is done using conventional level of detail methods, such as simple imposters [18], meshed imposters [17] and simplified polygon objects [19] (Figure 2).
It is difficult to schedule simplification methods such that the required quality is reached while minimising the resource load. Ultimately, only the application can decide which quality and resource load is acceptable. We developed a two-phase quality of service mechanism that allows the application to steer the scheduling, quality and resource load. Before any rendering takes place, the application gets the opportunity to pick the proper point on a curve representing the current quality/load tradeoff. This mechanism fits with the project-wide QoS management philosopy [20].

Tracking system

In order to reach the requirements of centimeter-accuracy at a latency of only 2 ms, we propose a tracking system fusing several sensors: accelerometers, gyroscopes, a compass, a video camera and differential GPS (DGPS). To minimise power usage, the systems and calculations are distributed carefully between the headset and the backbone (Figure 3).

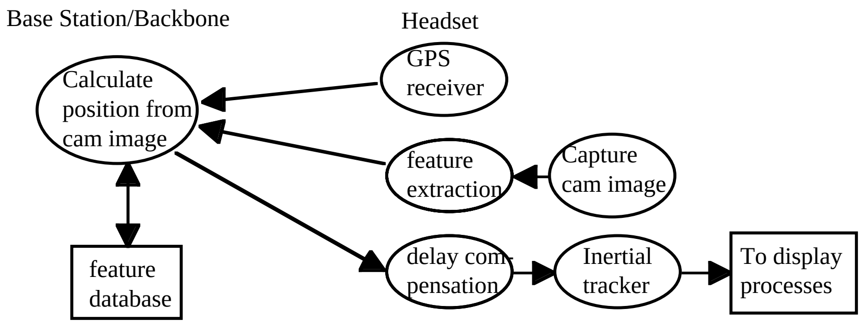

Figure 3. Integration of sensor data for position tracking. Part of the calculations is done in the backbone, to save power usage in the headset.

The DGPS system uses relatively much power, is not very accurate (2-10 m typical error) and very slow (1Hz refresh). Therefore, it is used only for initialising the tracking system.
The position of the camera can be determined by finding lines in the camera image and matching them with lines with known positions in the real world. The accuracy is rougly proportional to the pixel size of the camera, and can be of centimeter precision if there are matched lines at a meter distance.
The matching process is relatively expensive (see 'line matching' below), and it is done in the backbone. Transmitting camera images to the backbone is expensive as well, and compression is not a good idea because the edges in the image will be blurred, hampering the line extraction. Instead we extract the lines from the images within the headset, and transmit only the lines found. Available quick and low power time line extraction algorithms mainly rely on Hough transforms [21], and give lines without end points.

Accelerometers and gyroscopes are very fast and accurate, but due to their drift in these sensors, they have to be reset regularly, in the order of once per second depending on the drift of the gyroscopes and accelerometers. The sensors we use now [22] have very low drift and theoretically can do with much lower reset rates, but this sensor is large and heavy. In the future we plan to use smaller and lighter sensors, but these have higher drift [23].

GIS database

We need a GIS database in the backbone both for the line matching of the tracking system, and for the proper occlusion of virtual objects in the rendering system. We discuss how the GIS database is constructed, and which 3D data structure has use in order to allow quick response to latency-sensitive queries from both the rendering and tracking system.

The first, rough version of the GIS database has been constructed semi-automatically, using photos from the air and ground, DGPS measurements, 2D digital maps from the cadaster and manual steering of the processes (Figure 4). In contrast with common 2.5D GIS databases, our database has a full 3D structure.

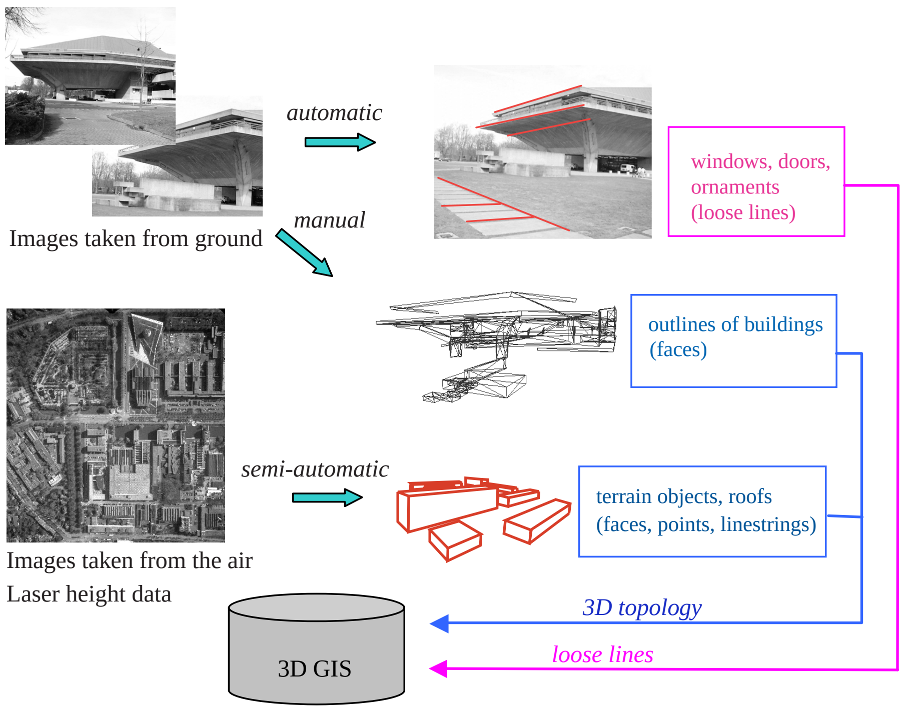
automatic
manual
Images taken from ground
windows, doors, ornaments (loose lines)
outlines of buildings (faces)
semi-automatic
terrain objects, roofs (faces, points, linestrings)
Images taken from the air
Laser height data
3D topology
loose lines
3D GIS

Figure 4. semi-automatic construction of the database. Photos taken from the ground are used to reconstruct the rough outlines of the buildings. These outlines are improved automatically with additional lines. Placement, alignment and height of buildings on a larger scale is semi-automatically based on images and laser height data taken from the air.

The rough outlines in the database are enriched with more lines, which are extracted from camera images. This enrichment can be done before the database is used for tracking, but also during use, by processing images from the camera on the headset.
The database is stored within Oracle 8. Both line matching and rendering are optimally served if only data especially relevant for the current position of the user is extracted from the database. Therefore, quick response to database queries (in the order of 1 second) is essential. To speed up especially response time, we decided to use our own functionality and 3D data structure, instead of oracle's spatial data engine (Figure 5) [24].

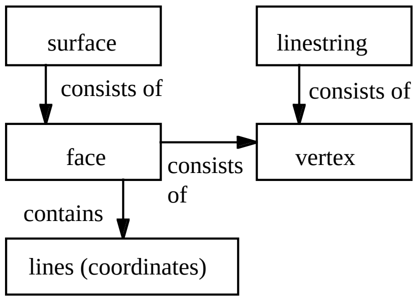

Figure 5. Proposed 3D data structure, optimised for supporting our rendering and tracking processers.

Line matching

Matching the lines from the database with the lines extracted from the camera images is essential to find the position and orientation of the camera (Figure 6)

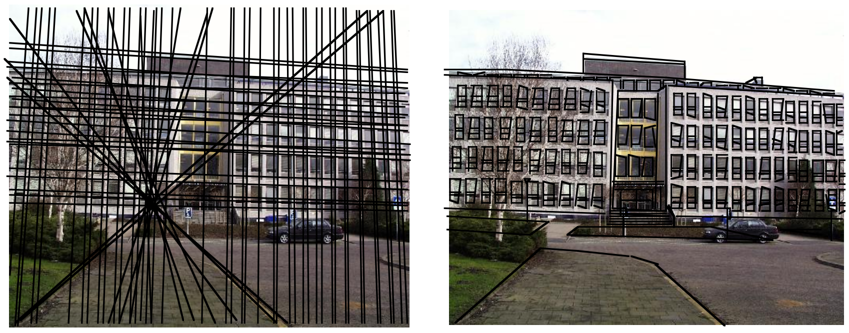

Figure 6. Lines extracted from the camera images (left) and from the GIS database (right). Matching those lines gives an accurate estimation of the camera position and orientation. Note in the left image that several of the long horizontal lines are doubled, because the image of the building is slightly curved. In practical situations, many of these spurious lines are found.

This matching is difficult:
• Many lines in the database will certainly be invisible, for instance lines at the back of a building. These lines have to be removed to improve the matching process. These hidden lines can be found with existing hidden surface algorithms.

However, the camera viewpoint is only approximately known, and therefore we cannot be completely sure which lines are visible. Depending on the stability of the matching process, lines of which it is not certain whether they are visible may be kept or removed. If kept, a false match may cause a precise but incorrect position estimation. If removed, the result will be some missed matches and a degraded position estimation.
• A related problem is that there will be lines invisible that were expected to be visible, for instance because there is a car in front of it, or because there is a tree in front of it that is not in the database.
• The database should provide lines of the appropriate level of detail: for nearby objects detail lines might be visible, while for distant objects only outlines are visible.
• Lack of resolution in the database. For instance, the exact locations of nearby tiles may not be stored, missing opportunities to accurately estimate the position.
• Repetitive patterns, such as tiles and windows may cause matching to be stuck in a local 'best match' instead of the correct match.
• Shadows and lens distortion may cause multiple lines to be extracted from the camera images instead of a single line (Figure 6), causing matching problems and extra link load and latency.
• Colors are ignored completely, reducing the possibilities for matching.
• If the headset provides lines without end points, special algorithms have to be developed to match them with the lines with end points from the database.

A number of these problems still have to be resolved. Consequently, the matching process still has to be guided by hand.

Statue application

To test the functionality of our system, we placed a virtual statue on the campus. The statue was scanned in from a real statue, and consists of 343k polygons. Our current prototype can not handle this model in full detail in realtime, because all VRML files are currently transmitted to and parsed uncompressed in the simplification machines. Instead, we use a 10 k polygons model internally.
At this time, our realtime tracking system is not yet running. To get an impression of the behaviour of our real-time rendering system, we replaced the real world and the tracking system with a video file annotated with off-line calculated position and orientation data. The video was played with the output routed to our rendering system, to generate AR overlays in realtime. The resulting images were mixed electronically with the video (Figure 7).
This environment video recorded by moving around with a video camera in front of the virtual statue, in a way similar to the movements that a person with our AR headset would make. The video was corrected offline for barrel distortion. The video file was also corrected for the time difference between the time the upper and lower line of the image will be displayed (using the rotational speed of the camera at that time as was calculated as discussed below).

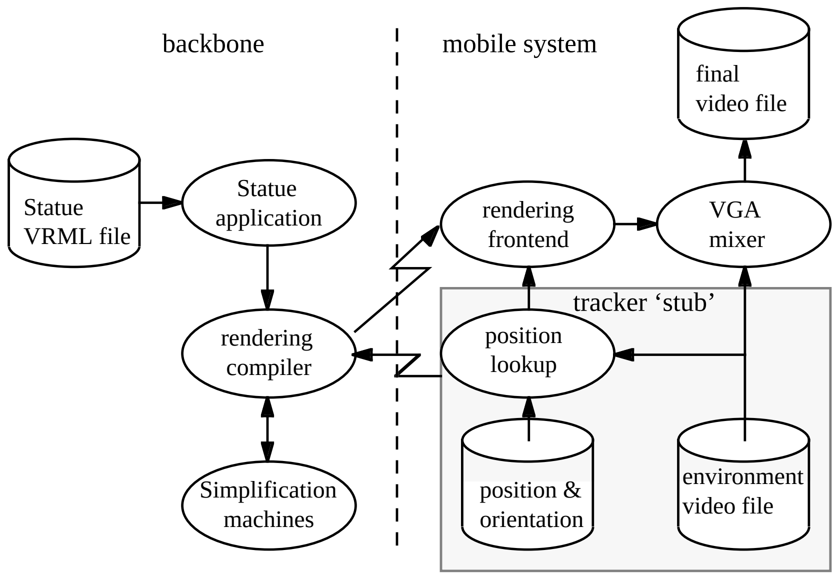

Figure 7. Setup to test our real-time low power rendering system.

The position and orientation data was recovered from the camera images alone, because we don't yet have the sensor fusion processes running. The position was recovered for each camera halfframe, by matching a few recognised points (typically, 3) with the real-world positions stored in the GIS database. A simple symmetric 5-tap low pass filter was applied to remove some of the noise.
The video was annotated with audio codes, in order to keep track of the frame number, so that the position and orientation corresponding to the current video frame could be picked from the positions file.
Figure 8 and 9 show preliminary snapshots from the final video file. If the observer would come much closer, it becomes necessary to represented the object with a polygon model, because the bandwidth to the headset does not allow us to update images quick enough. The result can look quite rude because of the maximum number of polygons that the rendering frontend can handle. The geometric distortions are acceptable here, but to keep the statue visually appealing as well a texture should be mapped on the polygon model. The simplification software can handle such textures.

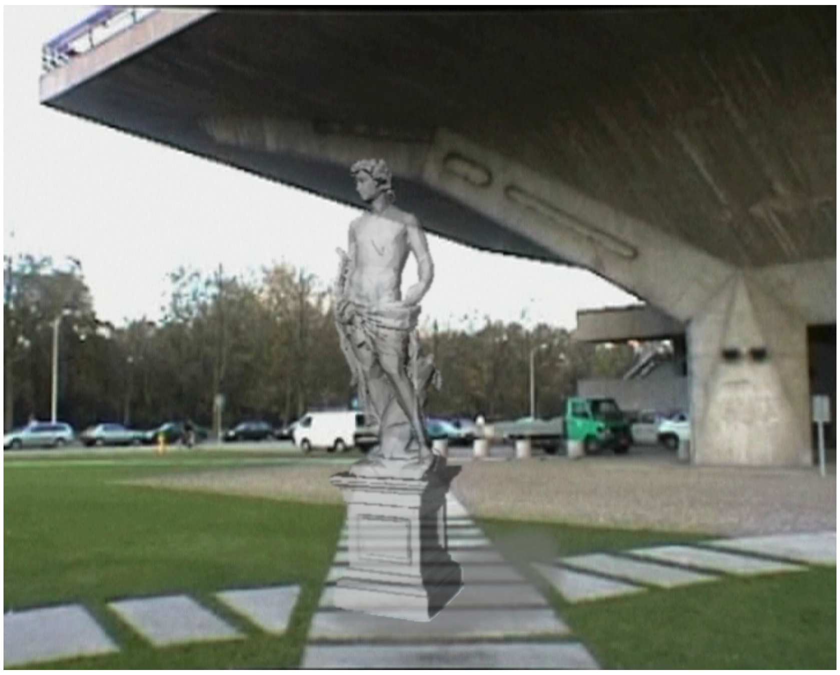
Figure 8. Preliminary nearby look at the statue. The statue is slightly transparent, due to the optical mixing of our AR system.

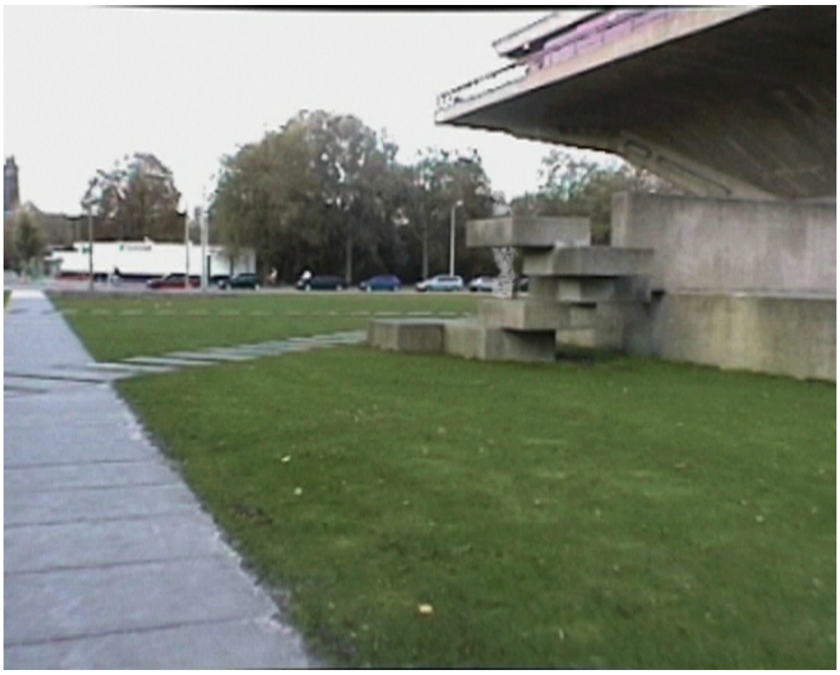
Figure 9. The virtual statue partially occluded by real-world artwork.

Conclusions, future work

We showed the system concept of the UbiCom mobile AR system, and we showed the capability of the rendering system to render complex virtual objects in a stable and robust way, while at the same time using only little power.

Our prototype system is not really low power, as it is based on off-the-shelf PC hardware. However, low-power alternatives with comparible performance exist, for instance ATI and 3dLabs have single-chip accelerators using approximately 1.5W [25] that can replace our Voodoo2 graphics card. We are working on hardware for a wireless low-power platform [26].

The most important component still to be finished is the tracking system. Differences in the lines extracted from the camera images and between the lines in the GIS database have to be coped with, and in such a way that the matching can be done in realtime while using as little power as possible in the headset.

A number of optimizations still have to be made in the rendering system. We plan to investigate how to update imposter images incrementally, using a robust version of MPEG video coding.

References

1. Feiner S, MacIntyre B, Höllerer T, Webster A. A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment. Proc. ISWC'97 (International Symposium on wearable computing (Cambridge, MA, October 13-14), 1997, 74-81. Available FTP: www.cs.columbia.edu/graphics/publications/ ISWC97.ps.gz.

2. Thomas B, Piekarski W, Gunther B. Using augmented reality to visualise architecture designs in an outdoor environment. Proc. DCNet'99 (Design computing on the Net, nov 30 - dec 3, University of Sydney, Australia), 1999. Available Internet: http://www.arch.usyd.EDU.AU/kcdc/journal/vol2/dcnet/sub8/.

3. UbiCom. Ubiquitous Communications: Aiming at a new generation systems and applications for personal communication. DIOC program at Delft University of Technology. 2000. Available Internet: http://www.ubicom.tudelft.nl.

4. Holloway RL. Registration error analysis for augmented reality. Presence, 1997, 6 (4), 413-432.

5. Poot HJG de. Monocular perception of motion in depth. Doctoral dissertation, Faculty of Biology, University of Utrecht, Utrecht, The Netherlands, 1995. ISBN 90-393-0820-9.

6. Padmos P, Milders MV. Quality criteria for simulator images: A literature review. Human Factors, 1992, 34 (6), 727-748.

7. Pasman W, Schaaf A van der, Lagendijk RL, Jansen FW. Accurate overlaying for mobile augmented reality. Computers & Graphics, 1999, 23 (6), 875-881.

8. Olano M, Cohen J, Mine M, Bishop G. Combatting rendering latency. Proceedings of the 1995 symposium on interactive 3D graphics (Monterey, CA, April 9-12), 1995, 19-24 and 204. Available Internet: www.cs.unc.edu/~olano/ papers/latency.

9. Youngblut C, Johnson RE, Nash SH, Wienclaw RA, Will CA. Review of Virtual Environment Interface Technology. Internal report P-3186, Institute for Defense Analyses (IDA), Alexandria, VA, 1996. Available Internet: http://www.hitl.washington.edu/scivw/IDA/.

10. Borenstein J, Everett HR, Feng L. Where am I? Sensors and Methods for Mobile Robot Positioning. Technical Report, University of Michigan, 1996. Available Internet: http://www-personal.engin.umich.edu/~johannb/position.htm.

11. Azuma R, Hoff B, Neely H, Sarfaty R. A Motion-Stabilized Outdoor Augmented Reality System. Proceedings of IEEE VR '99 (Houston, TX, 13-17 March), 1999, 252-259. Available Internet: http://www.cs.unc.edu/~azuma/ azuma_publications.html.

12. Behringer R. Registration for outdoor augmented reality applications using computer vision techniques and hybrid sensors. Proc. IEEE Virtual Reality, 1999, 244 -251.
13. Berger M-O, Wrobel-Dautcourt B, Petitjean S, Simon G. Mixing synthetic and video images of an outdoor urban environment. Machine Vision and Applications, 1999, 11 (3), 145-159. Available Internet: http://www.loria.fr/equipes/isa/ pages/English/Publications.html.
14. Simon G, Lepetit V, Berger M-O. Registration methods for harmonious integration of real and computer generated objects. Computer Graphics Forum, Conference Issue Eurographics, 1999. Available Internet: www.loria.fr/~gsimon/ eg99.html.
15. Brewer E, Burd T, Burghardt F, et al. Design of wireless portable systems. Proc. IEEE Compcon '95 'Technologies for the Information Superhighway', 1995, 169 -176.
16. Pasman W, Schaaf A van der, Lagendijk RL, Jansen FW. Information display for mobile augmented reality: Merging the real and virtual world. Proceedings of the IMC'98 (Rostock, Germany, November 24-25, 1998), 1998. Available Internet: www.cg.its/tudelft.nl/~wouter.
17. Decoret X, Schaufler G, Sillion F, Dorsey J. Multi-layered impostors for accelerated rendering. Computer Graphics Forum (Proceedings of Eurographics '99), 1999, 18 (3), 61-73. Available Internet: http://www-imagis.imag.fr/~Jean-Marc.Hasenfratz/ EG99/paper-EG99.html.
18. Aliaga DG, Lastra AA. Smooth transitions in texture-based simplification. Computers Graphics, 1998, 22 (1), 71-81.
19. Garland M. Quadric-based polygonal surface simplification. Doctoral thesis, Carnegie Mellon University, 1999. Available Internet: http://www.cs.cmu.edu/ ~garland/thesis.
20. Dijk H, Langedoen K, Sips H. ARC: A bottom-up approach to negotiated QoS. Proc. WMCSA'2000 (7-8 December, Monterey, CA), 2000, 128-137.
21. Davies ER. Machine Vision: theory, algorithms, practicalities. San Diego: Academic press, 1997. ISBN 0-12-206092-X.
22. Crossbow. FOG-Auto IMU600AA. Crossbow Technology, Inc., 41 E. Daggett Dr., San Jose, CA 95134, 2000. Available Internet: http://www.xbow.com/pdf/ FOG-AUTO2.PDF.
23. Persa S, Jonker PP. On positioning for augmented reality systems. Proc. HUC'99 (Handheld and Ubiquitous Computing, Karlsruhe, Germany, September), 1999, 327-330. Springer Verlag, Berlin.
24. Zlatanova S, Verbree E. A 3D Topological model for augmented reality. proceedings of the MMSA'00 (2nd Int Symp on Mobile Multimedia Systems and Applications, Delft, Netherlands, Nov. 9-10), 2000, 19-26. Available Internet: http://www.geo.tudelft.nl/frs/staff/sisi/ubicom.htm.
25. 3Dlabs. Permedia 2. 3D Labs Inc., Sunnyvale, CA 94086, 1999. Available Internet: http://www.3Dlabs.com/products/p2.html.
26. Bakker JD, Mouw E, Joosen M, Pouwelse J. The LART Pages. Delft University of Technology, Faculty of Information Technology and Systems, 2000. Available Internet: http://www.lart.tudelft.nl.
27. Harris C. Tracking with Rigid Models. In A. Blake, Yuille (Eds), Active Vision, 1992, 59-74. MIT Press.
28. Schmid C, Zisserman A. The geometry and matching of lines and curves over multiple views. International Journal on Computer Vision, 2000, 40 (3), 199-234.