A High Performance AR System for Medical Applications

Sebastian Vogt[1], Ali Khamene[1], Frank Sauer[1], Andreas Keil[1], and Heinrich Niemann[2]
[1] *Siemens Corporate Research, Imaging & Visualization Department, Princeton, NJ, USA*
[2] *Chair for Pattern Recognition, Universität Erlangen-Nürnberg, Erlangen, Germany*
sebastian.vogt@scr.siemens.com

Abstract

We report on a new single PC based stereoscopic video-see-through AR system which we developed for medical applications. Recent advances in graphics hardware, memory bandwidth, and computing power of standard PCs made it possible that this system outperforms an earlier version which included 3 networked SGI workstations. We designed and implemented a new AR software platform. It is component based and—in conjunction with XML configuration files—provides efficiency, modularity, and extensibility for fast and robust prototyping of AR applications. The system has a compelling real-time performance with 30 frames/second, displaying stereoscopic augmented video views with XGA resolution.

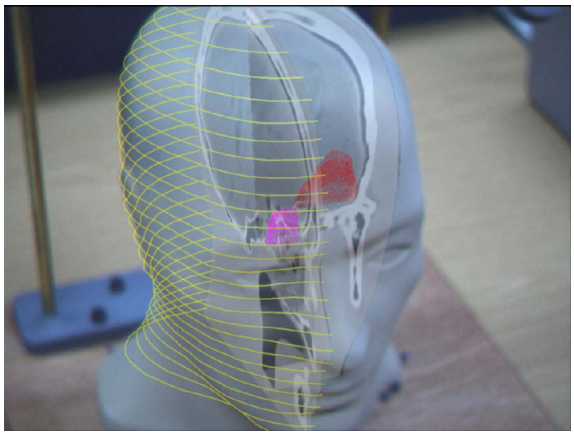

Figure 1. Phantom head augmented with data extracted from a CT scan.

1. Introduction

AR visualization in the medical field has first been suggested and investigated at UNC for ultrasound-guided procedures. Our AR system [3, 2] is based on a stereoscopic head-mounted display (HMD) of the "video-see-through" variety as pioneered at UNC [4]. A stereo pair of video cameras serves as artificial eyes, and the live stereoscopic video view of the real scene is overlaid with computer graphics in real-time. Figure 1 shows an example of a phantom object which is augmented with data extracted from a CT scan. Our AR system's special feature is the use of single camera tracking of retroreflective markers with a head-mounted tracking camera, which is rigidly attached to the two cameras that capture the stereo view of the scene [5]. The system has been tested in a variety of medical scenarios.

The centerpiece of our AR system is still the video-see-through HMD as originally described in [3]. However, we completely redeveloped the underlying AR system to make it more practical, efficient, modular, and extensible. This comprised major changes in our computer hardware as well as the AR software structure.

2. Single PC Based AR System Architecture

Processing the three incoming video streams with 30 Hz frame rate and realizing a correspondingly fast stereoscopic graphics output in XGA resolution puts high demands on the computational bandwidth and architecture of the AR system. We built our first AR system by using three networked SGI visual workstations [3]. Recent advances in graphics hardware, memory bandwidth, and computing power of standard PCs made it possible for us to transfer the technology of our AR system from 3 SGI PCs to a single PC with dual monitor graphics support. Figure 2 shows the structure of the new system on the left. The graphics system (Nvidia Quadro4 900 XGL) is capable of generating a single monoscopic AR image in 5–15 milliseconds, dependent on the complexity of graphics to be rendered on top of the video image, which itself is uploaded into the graphics memory and scaled from VGA to XGA format as part of the rendering process. This speed is necessary to achieve real-time performance of 30 Hz, where each stereo image has to be processed and visualized within 33 milliseconds.

The PC (Dell workstation with dual Xeon 1.7 GHz processor) has 2 PCI buses, which—equipped with 3 frame grabber cards—provide enough bandwidth to transfer all three video streams to the main memory in real-time. The three cameras are genlocked so that the video streams arrive at the computer in a synchronized way. Since we preserve the synchronization of the incoming frames, there is no time lag between video and augmenting graphics, which contributes to a very believable AR perception.

Proceedings of the Second IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR '03)
0-7695-2006-5/03 $17.00 © 2003 **IEEE**

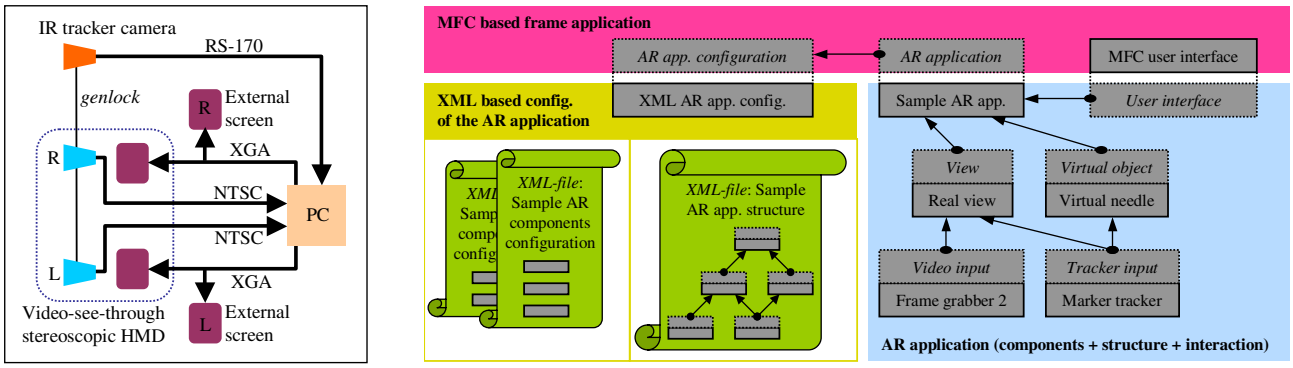

Figure 2. *Left:* **Scheme of the new single PC based AR system with 3 NTSC camera inputs and 2 XGA outputs.** *Right:* **Software component structure of the extensible AR system.**

3. Component AR Software Architecture

A powerful and flexible AR system requires a good abstraction hierarchy during the development of reusable AR software components. Basic concepts and program structures are topics of ongoing research, e.g. [1]. We designed a modular, extensible, and very efficient component based AR software library. The major issues were:

- Easy integration of new components into the system,
- Ability to configure each component independently,
- Defining AR application structures on a higher level.

By *component* we generally mean some piece of code that fulfills a specific task, e.g. providing an interface to a certain hardware device like a camera, a video file on the hard drive, or an external tracker, executing a computer vision algorithm, rendering some part of a virtual 3D scene, or dealing with user interactivity. The definition of an *AR application structure* should be based on a certain abstraction level of the incorporated components.

The principal idea behind this programming environment is depicted in Figure 2 on the right. The right side of the chart illustrates how the components of an AR application interact with each other by using *signals* and *slots*, which permits real information encapsulation, a fundamental concept of component programming. For example, if a component changes states it can emit according signals without knowing which other components are receiving them and thus act on them. In Figure 2 a signal/slot connection is represented by an arrow from the component emitting the signal to the component receiving the signal (with its slot). The abstract interface to the signals of a component is indicated by a broken framed box, e.g. 'VideoInput'. The actual implementation of the component is specific and symbolized by a closed framed box, e.g. 'Frame grabber 2'. On the one hand, this level of abstraction allows an independent generation and configuration of each specific component, and on the other hand allows the definition of the AR application structure on an abstract level.

XML is utilized to configure the component attributes and the application structure. One or more XML files define the properties of the particular components. The structure of the AR application is specified in an independent XML file by using the abstract interfaces of the components. This separation allows use of the same AR application structure on different systems or algorithms. For instance, one might want to use a certain AR application on a laptop for demonstration by using video files and on a real-time system by using live connected cameras. In this case, one loads a different XML component configuration file—the XML application structure file is the same in both cases.

Using XML has the advantage that there are many external tools available to create and edit those files. Thus, XML serves as a meta-language on the one hand to easily create and maintain AR applications and on the other hand to configure each single component. The actual components and the frame application are efficiently implemented with C++.

References

[1] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner. Design of a component–based augmented reality framework. In *Proc. ISAR*, pages 45–54, Oct. 2001.

[2] F. Sauer, A. Khamene, and S. Vogt. An augmented reality navigation system with a single-camera tracker: System design and needle biopsy phantom trial. In *Proc. MICCAI*, pages 116–124, Sept. 2002.

[3] F. Sauer, F. Wenzel, S. Vogt, Y. Tao, Y. Genc, and A. Bani-Hashemi. Augmented workspace: Designing an AR testbed. In *Proc. ISAR*, pages 47–53, Oct. 2000.

[4] A. State, M. A. Livingston, G. Hirota, W. F. Garrett, M. C. Whitton, H. Fuchs, and E. D. Pisano. Technologies for augmented-reality systems: realizing ultrasound-guided needle biopsies. In *SIGGRAPH 96*, pages 439–446, Aug. 1996.

[5] S. Vogt, A. Khamene, F. Sauer, and H. Niemann. Single camera tracking of marker clusters: Multiparameter cluster optimization and experimental verification. In *Proc. ISMAR*, pages 127–136, Sept. 2002.