First Steps Towards Handheld Augmented Reality

Daniel Wagner Dieter Schmalstieg
Vienna University of Technology, Favoritenstr. 9-11/188/2, A1040 Vienna, Austria
{wagner|schmalstieg}@ ims.tuwien.ac.at

Abstract

In this paper we describe the first stand-alone Augmented Reality (AR) system with self-tracking running on an unmodified personal digital assistant (PDA) with a commercial camera. The project exploits the ready availability of consumer devices with a minimal need for infrastructure. The application provides the user with a three-dimensional augmented view of the environment. Our system achieves good overlay registration accuracy by using a popular marker-based tracking toolkit (ARToolKit), which runs directly on the PDA. We introduce an optional client/server architecture that is based on wireless networking and is able to dynamically and transparently offload the tracking task in order to provide better performance in select areas. The hardware/software framework is modular and can be easily combined with many elements of an existing AR framework. As a demonstration of the effectiveness, we present a 3D navigation application that guides a user through an unknown building to a chosen location.

Keywords: Augmented Reality, Mobile Computing, Tracking, PDA

1. Introduction

Augmented Reality (AR) is a natural complement to mobile computing, since a mobile AR system can assist the user directly in several situations. There has been a lot of work in creating AR mobile setups using mobile personal computer hardware. The advantage of those approaches is that hard- and software very similar to traditional non-mobile AR can be used. While there are working systems composed of a notebook and a head mounted display (HMD), most of these setups have been designed as a proof-of-concept and do not provide a usable form factor. Usually wearable prototypes have all their hardware mounted to a large and heavy backpack.

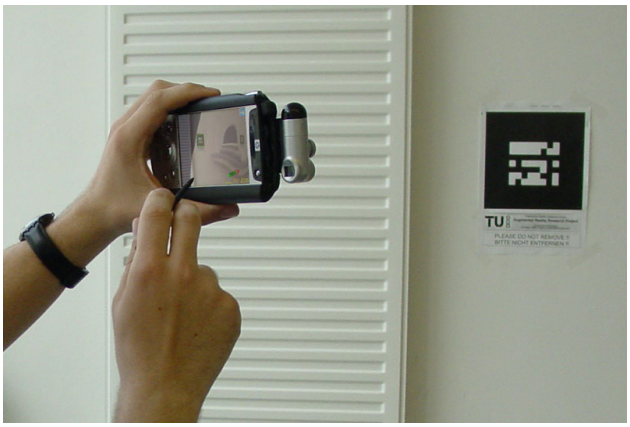

Figure 1. The self-contained handheld AR system can track optical markers in real time.

While such backpack/HMD setups combine superior performance with hands-free operation, they severely affect dexterity, prevent practical use and are socially unacceptable. At the same time, broad consumer interest in very small form factor devices and displays, such as cell phones and handheld computers, is dramatically accelerating the development in this area. We therefore consider it to be obvious that one of the next major steps in mobile AR development will be a shift to smaller and more ergonomic devices.

From a perspective of mobility, a personal digital assistant (PDA) has the perfect form factor for a self-contained interactive computing platform. In contrast to notebooks, PDAs can be carried around without ergonomic drawbacks since they were designed for exactly that purpose. They are sufficiently small and lightweight to be held in one hand.

The first wave of PDAs had weak processors and displays. However PDAs have recently advanced in speed and display quality such that they are now used for some tasks traditionally associated with desktop workstations. As a consequence, PDAs have become very successful commodity items, and a large user base exists. Communication providers are strongly backing the integration of inexpensive wireless networks and video cameras into PDA devices to build a market for new

communication services. These characteristics are also beginning to meet the requirements of mobile AR systems. The remaining disadvantage of PDAs - the lack of a standardized, well approved software interfaces - can be overcome with moderate engineering efforts.

In this paper we describe a modular framework for Augmented Reality applications on small handheld devices. We present the first implementation that works fully autonomously on a PDA (see Figure 1), including video see-through 3D rendering and optical tracking with optional back-end server support. The system features the following characteristics:

- an autonomous vision-based tracking system executing at interactive rates
- 3D scene rendering via a standardized graphics interface
- optional dynamic workload-sharing with a backend server, outsourcing the computationally expensive computer vision calculations to the server via a wireless network, if available, in order to speed up the overall process
- optional high-precision outside-in tracking of the PDA by a specialized tracker
- Integration of the handheld platform's software into our AR research framework *Studierstube* [25] for mutual re-use of resulting software components between workstation/notebook and PDA-based AR.

To demonstrate the power of our solution we have implemented an application that guides a user through an unfamiliar building to their destination. The PDA is used as a see-through video device as well as an input device. The application tracks fiducials attached to the building's walls using an optical tracking toolkit.

2. Related Work

The MARS (Mobile Augmented Reality Systems) project [9] presented in 1999 by Columbia University was one of the first truly mobile augmented reality setups which allowed the user to freely walk around while having all necessary equipment mounted onto his back. Several similar platforms such as BARS [12], Tinmith [17] and our own mobile *Studierstube* [13] examined various application areas.

So far work in mobile AR has focused on wide-area tracking. Most commercial solutions such as optical or infrared trackers cover only a limited work area, so researchers have aimed at using e. g., GPS [7], inertial sensors [3], and computer vision [24] for tracking. The Bat System [11] from AT&T provides building-wide accurate tracking of people and objects fitted with badges that are tracked by a 3D ultrasonic location system, but at the cost of deploying extensive electronic infrastructure. Recently, the freely available ARToolKit [4] has allowed a broad audience to perform optical tracking using inexpensive cameras.

Even before PDAs were mass-marketed, pioneering projects started using small displays as see-through devices. Among the early activities in this area was the work of Amselem [1] and Fitzmaurice [5], which used a small LCD to display location-based information.

Rekimoto used color-coded stickers to track objects in the environment with his NaviCam [23]. A workstation analyzed the gathered images and created an augmented view that displayed textual information. Due to the tethering the mobility still was much reduced.

Other early work was done in the Transvision [22] project by Sony CS Labs in 1996 which provided a shared augmented space for multiple users tracked by a Polhemus sensor [18]. The HITLab improved this concept [16] by providing an enhanced user interface and an optical tracking solution utilizing the camera image needed for the see-through effect. Other interesting server/client based work [19] was done at the Daimler-Chrysler research center, where researchers used analog video transmission from and to the handheld device.

Most Augmented Reality projects that are based on off-the-shelf PDAs still use these devices primarily as displays and outsource most challenging tasks such as rendering, tracking and application intelligence to stationary servers. In the Batportal project [11] AT&T Labs Cambridge developed a PocketPC-based sentient application that enriches the environment with context-sensitive information. The PDA worked as a thin client that displayed 3D graphics streamed from a server using X Server. In the AR-PDA project [8] a server/client architecture was developed for digital image streaming from and to an application server. The PDA acted as a thin client whereas the server was responsible for natural feature tracking and rendering tasks.

To our knowledge we present the first completely stand-alone Augmented Reality client on a PDA that can work as part of a passive, scalable wide area tracking infrastructure.

3. System overview

3.1. Design rationale

As indicated above, previous handheld AR devices were based on a thin client approach with a "video-in/video-out" communication mechanism for receiving assistance from a computing server, either via a tethered [5][16] or wireless network [19][8][11][1] (see Figure 2d). By contrast, the most versatile and desirable configuration is a handheld unit capable of being fully self-contained (Figure 2a).

However, these two solutions are just extreme examples of how work may be allocated among a server and handheld client. In the near future, solutions in between these extremes may be useful and necessary.

If we limit the discussion to a typical AR system using a single video source for both tracking and video see-through display, the processing pipeline is composed of the following main tasks: video acquisition, tracking, application computation, rendering, display. Offloading some of these tasks to a computing server is an instance of horizontally distributed simulation [15], and it is established knowledge that a scalable solution (many clients, many servers etc.) requires cautious use of the available network bandwidth [27].

Communication of raw video streams in both directions (Figure 2d) does not satisfy such bandwidth constraints. Therefore, in the following we are only considering solutions that have two important characteristics: (1) digital, low bandwidth and wireless (2) no round-trip communication of raw video streams. The video is fed directly to the handheld's display for augmentation.

The approach depicted in Figure 2b offloads the tracking task to a computing server and requires upstream communication of special pre-processed video for visual tracking purposes, that can be dramatically compressed (see section 3.5 for details), followed by downstream communication of pose information. The advantage of this approach is that a very concise, but generic and computationally expensive task is offloaded to the server, while all application details are handled exclusively by the client, thus dependencies between client and server are minimal.

Another possible solution includes the allocation of both tracking and application processing to the server (Figure 2c). In this case, the roles are reversed, and the handheld's role is that of a thin client. Rather than receiving 2D drawing instructions such as via the VNC protocol [11], a 3D command streaming protocol like Chromium [10] can be used. While this solution can be considered an improved version of the previous thin client approaches and has the advantage of allowing the re-use of existing workstation-based AR applications with only minor adjustments, it requires the permanent presence of a server and was therefore not considered further. Nevertheless, it may be a useful alternative in some situations, which we plan to fully examine in the future.

For the work described in this paper, a combination of Figure 2a and Figure 2b was adopted. We dynamically (at application runtime) and transparently (without requiring user intervention) switch between the two methods, which we consider a good compromise since it allows improved performance of the handheld client in selected "hotspot" locations without burdening the user. Alternatively, the PDA can also be tracked with a commercial outside-in tracking system to completely remove the computational cost from the PDA's execution pipeline as well as providing very high precision in selected areas. Switching between all three tracking modes occurs transparently to the client application and the user.

3.2. Hardware setup

One primary goal of our development was to rely solely on standardized, widely available hardware to extend the range of practical applications. For our work we choose a PocketPC 2002 device (HP iPAQ 5450) with 400 MHz Intel XScale processor, 240x320 16-bits display, 64MB RAM and IEEE 802.11b wireless network interface. We extended it with an inexpensive camera with 320x240 color resolution, attached via an HP CompactFlash jacket (see Figure 3).

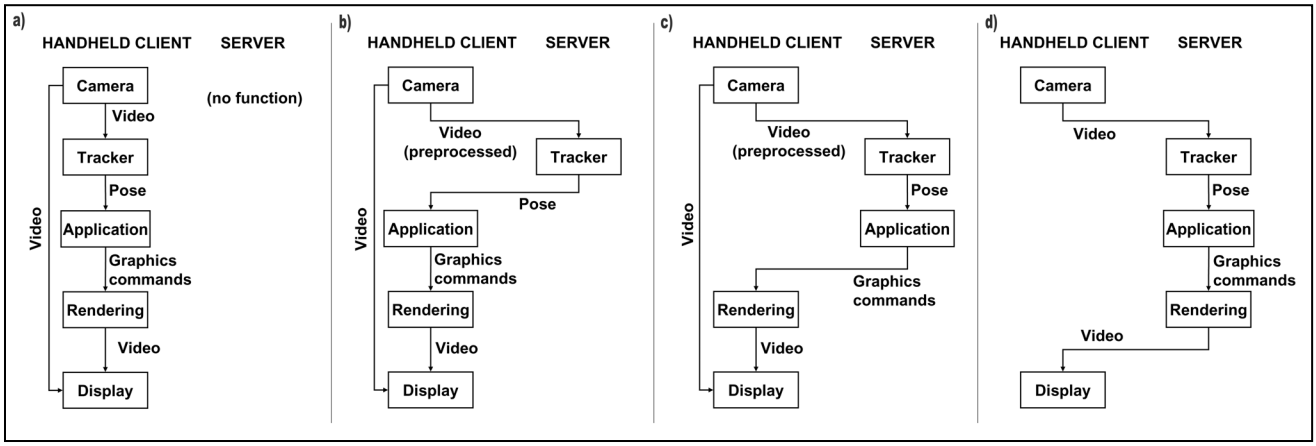

Figure 2. Client/server interaction types: (a) completely self-contained client, (b) tracking computation offl-loaded to server, (c) tracking and application executing on server, (d) thin client only resposible for input/output

Figure 3. The IPAQ's CompactFlash camera is used for both optical tracking and as a source for the video see-through display

The PocketPC is currently the most powerful commercial PDA. Nevertheless, its design was guided by issues such as portability and battery life, and a focus on business applications. To create a usable AR client in particular for the fully autonomous use-case, we had to take a number of characteristics into account:

- Advantages: The PocketPC was primarily chosen for its unrivaled ratio of performance to size and weight and its integrated I/O features, in particular the bright touch-sensitive display and the available camera add-on.
- Disadvantages: The PocketPC provides only a fraction of the processing power and memory bandwidth of a modern workstation CPU. In particular, the XScale CPU does not possess any floating point or signal processing capabilities, typically required by graphics and vision tasks. Additionally, the PocketPC has no 3D graphics acceleration, and even 2D graphics are not consistently accelerated. The CompactFlash camera delivers images with only 6.2 Hz, which compares unfavorably even to low cost USB webcams.

3.3. Software components

In addition to its limited hardware capabilities, the PocketPC platform currently lacks a well defined and matured set of development tools, that augmented reality developers are generally taking for granted. There is no industrial strength 3D graphics library such as OpenGL and no common video interface such as DirectShow. The available operating systems (Windows CE, Embedded Linux) are not trivially compatible to their workstation counterparts.

Consequently, a significant part of our work was directed at designing and implementing software components that make optimal use of the available restricted hardware capabilities of the handheld platform in order to achieve interactive performance.

However, our aim was not to create significant amounts of hand-optimized specialized code, but to organize the software into a framework of well-performing modules that are compatible to our existing AR software framework *Studierstube* [25] that has been used in previous research projects, and to allow for easy interoperability of these components as well as cross-platform development of applications. In this paper, the software reuse paid off primarily at the application level, but we expect additional synergy also on the system level (see future work).

In particular, we implemented two variants of optical tracking and a generic 3D graphics library as reusable software components, which are outlined in the following.

3.4. Optical tracking on PDA

ARToolKit [4] is a suitable tracking solution for our purposes, since it works with a single camera operating in the visible light range, and thus allows a handheld device to work autonomously by only relying on a passive infrastructure in form of fiducials attached to objects or, in our case, walls (see Figure 1).

To our surprise, porting (the tracking code of) ARToolKit to the PocketPC/Windows CE platform was relatively straight forward. After fixing a minor issue in memory management (Windows CE does not permit extremely large static variables, requiring dynamic allocation for such structures), the library could simply be cross-compiled using MS Embedded Visual C++. The library was straight forward to build but ran slowly, using the compiler's built in floating point emulation.

The next step consisted in analyzing the performance to locate bottlenecks. Unfortunately, there are currently no usable profiling tools for such work. Therefore, the code was instrumented by hand to gather the necessary statistics. To achieve this aim, the basic float type was replaced by a class type that can gather dynamic access statistics and track the used numeric range. From this information, a small number of heavily used computation functions were identified together with the information relevant for fixed-point optimization (minimum and maximum value used etc.). These functions were responsible for the majority of the slow emulated floating point computations. After manually converting the floating point use in these functions to efficient fixed-

point computation based on the Intel fixed-point library, we achieved a threefold speedup, yielding 5-8 pose estimates per second when the XScale CPU can be utilized exclusively (i. e., no other tasks executed concurrently). The speed of the pose estimation is highly dependent on the number of processed pixels and thus of the marker size in the camera image.

Tests show that the tracking precision is only minimally reduced due to the usage of fixed point arithmetic: The tracked position shifts around 10 cm compared to the value calculated with floating point at a distance of 200 cm from a 15x15 cm^2 marker.

3.5. Server-assisted optical tracking

ARToolKit is intrinsically CPU intensive which generally is a serious problem for a low-power platform such as the PocketPC and can not be completely solved by applying optimizations. We implemented a tracking server that runs on a workstation (a 2.4GHz PC). The mobile client uses WLAN to send frame analysis requests to the server.

A 320x240x24bit RGB image has a size of 225Kb which too large for a fast service request. However, for tracking-only purposes, a full color video is not necessary. After applying a black/white thresholding followed by a 1-bit RLE compression the image size is reduced to 2-4 Kbytes, which easily fits the available WLAN bandwidth (theoretically up to 300 fps can be transmitted over 802.11b). Performing this simple compression has negligible overhead on the PDA (see the results section for detailed performance analysis). We transmit the compressed video using a low latency UDP based protocol, because the frequent updates make reliable transmission unnecessary. The server decompresses the image and passes it on to ARToolKit for analysis. The resulting transformation matrix and the index of the best recognized marker are sent back to the client.

Outsourcing the ARToolKit analysis task to a server significantly increases the overall performance. By using a parallel approach we can reduce the server's response time to virtually zero. In fact there is still a little overhead in the network interface for sending and receiving the network packets. While waiting for the server's reply the PDA is free to execute further necessary tasks which do not depend on that reply. We choose to read the next image which is by far the most time consuming task during a single frame generation in server assisted mode. Thereby the overhead for server-assisted tracking is reduced to virtually zero, at the expense of some additional latency introduced by waiting for the camera image read operation to complete (see Figure 4 for a timing diagram).

It is not necessary for the client to know the server's network address at start-up time. The client periodically sends out multicast messages in a separate thread to detect available servers. These messages are also used to detect areas of weak WLAN coverage. If no server is accessible, the client automatically falls back to stand-alone mode. As an intended side-effect, multiple tracking servers in the environment can be assigned to clients in a spontaneous way based on availability and accessibility.

3.6. Outside-in Tracking

Commercial outside-in tracking solutions such as the ARTTrack [2] yield superior performance and offload all tracking computations from the client, while requiring only untethered passive targets (retroreflective spheres) mounted to the tracked object. While these advantages are significant, the high cost and limited working area make the use of such commercial trackers only feasible in a limited area intended for high-precision work. To make good use of such resources, we adopted the strategy of transparently switching from ARToolKit to ARTTrack when the PDA client is in the working area of the outside-in tracking system.

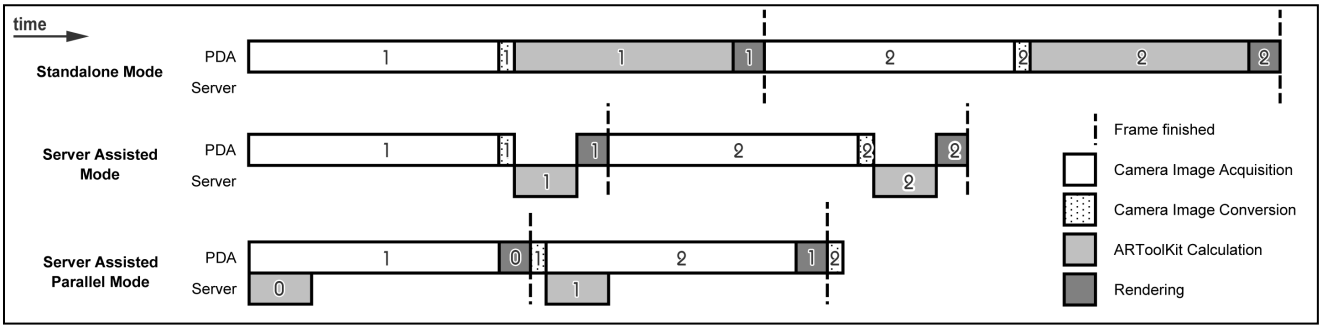

Figure 4. Timing diagram for PDA/server communication. Numbers denote the frame for which the action is executed

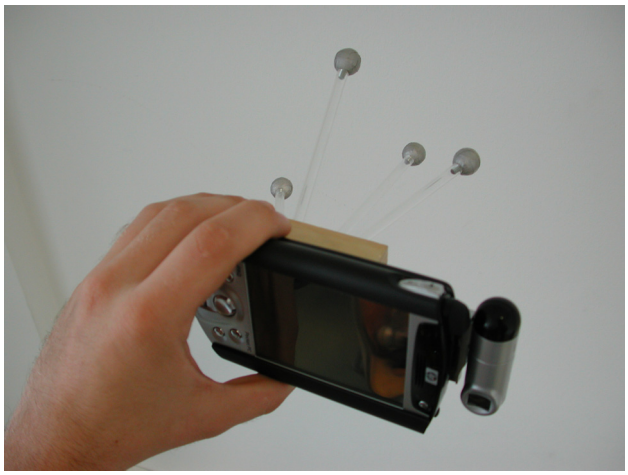

Figure 5. The ARTTrack target composed of an arrangement of retroflective balls provides passive high-precision tracking

We built a simple tracking target (see Figure 5) which can easily be applied to and removed from the read of the PDA's chassis by using Velcro. By using the same network transmission format for both ARToolKit and the ARTTrack communication to the client, support for ARTTrack is completely transparent. If the PDA's tracking target is recognized by the ARTTrack server, these pose values are preferred over the values computed by ARToolKit. The PDA notices no difference between tracking data analyzed by ARToolKit on the server and data coming from the ARTTrack system.

This extension demonstrates the simplicity of integrating arbitrary tracking systems without the need to customize the PDA's software. In particular it allows us to use high-performance and high-precision tracking systems in an area of the building where too few fiducial markers are present to provide a continuous tracking service.

3.7. 3D Graphics Library

As computer graphics researchers we are accustomed to working with standardized low level APIs such as OpenGL. Most augmented reality projects even use an object-oriented middleware product such as Open Inventor or Performer to provide further abstraction from the underlying hardware. On the PocketPC platform there is no built-in 3D graphics subsystem, and no widespread general purpose commercial solutions currently exist.

Although there have been prior attempts to create libraries similar to OpenGL such as PocketGL [28] and TinyGL [29], these projects are either restricted to games or incomplete and discontinued. We therefore chose to implement our own software renderer called SoftGL (see Figure 6), which uses a subset of OpenGL as the API, in order to ease the porting of existing AR applications and lay a foundation for future development. We implemented the most important OpenGL API methods for rendering primitives, performing transform & lighting computations and drawing a video background. Despite software-only rendering and a lack of native floating point computations, the lightweight graphics requirements of typical AR applications such as hidden line make this approach reasonably competitive. Our plan is to make this library compliant with the OpenGL|ES specification [30], which is currently in draft.

The renderer is partially based on existing rasterization code originally developed for PCs without 3D graphics acceleration, but with floating point support on the CPU. As can be expected, the geometry stage of the OpenGL pipeline makes heavy use of floating point operations, while the rasterization stage uses integer operations.

A straight port of this code using compiler-generated floating point emulation performed quite well without further manual optimizations. Preliminary investigation revealed that performance limitations are indeed due to (emulated) floating point operations for models with complex geometry and due to integer performance when rendering large low-detailed models with heavy overdraw

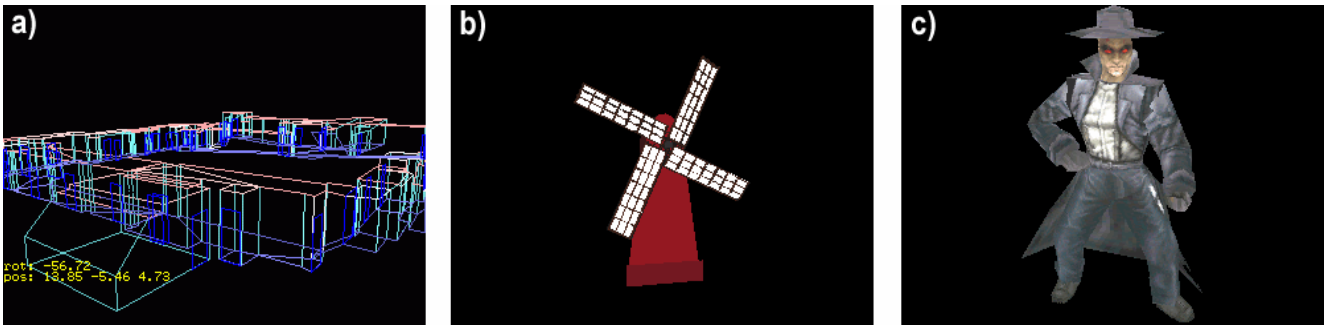

Figure 6: SoftGL renderings, a) the test environment for the indoor navigation application. b) a standard OpenInventor model rendered with Coin3D, which runs on top of SoftGL. c) a textured 3D character.

such as in a building. For complex geometric models, the floating point emulation becomes the bottleneck. Unfortunately, the transform and lighting computations do not easily lend themselves to fixed-point optimization, since the numeric range varies dramatically depending on the model data, and in some situations precision requirements mandate the use of floating point arithmetic.

To remedy this situation, we chose to extend the OpenGL API with a number of meta commands that control the configuration of the renderer. The decision concerning which numeric mode to use is placed on the programmer who can easily set the relevant parameters.

Technically, the library was implemented as a collection of C++ template classes which allow the configuring of the floating point type at compile-time, with a C wrapper for API compatibility. Developers can choose between using float, double, fixed-point emulation or any other custom data type (for example the "instrumented float" for profiling) that behaves as a basic float type to store vertex data. By providing multiple renderer instances internally, SoftGL can switch between numeric representations at runtime. This allows developers to choose the appropriate calculation mode where necessary.

However, this approach does not provide optimal efficiency, since the API always uses values passed as floats, which need to be converted to and from fixed-point on the fly. To further enhance the efficiency of the renderer we added selected higher-level retained mode functions on top of SoftGL, which avoid conversions and provide developers with indexed vertices and normals, so the number of vertex operations is drastically reduced for static models.

The compliance to the OpenGL API even allowed us to port Coin3D, a scene graph library compatible to Open Inventor, to the PocketPC platform (see Figure 6b).

Signpost uses only a subset of the functionality provided by SoftGL. In the future we plan to rely more on higher level APIs such as Coin3D which will fully exploit the capabilities of SoftGL.

4. Application: Signpost

To test our PDA-based augmented reality system with an actual application within the *Studierstube* framework, we choose to implement a client for the Signpost [13] project. The application guides a user through an unknown building by showing a variety of navigation hints, including a wireframe visualization of the building structure superimposed on the video image, labeling of relevant elements and highlighting of the next exit to take. In addition, an arrow indicates the direction (Figure 7), and a 2D overview map is available (see Figure 8).

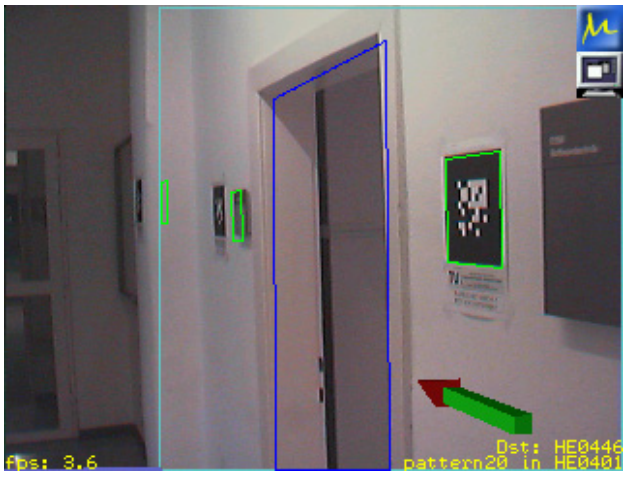

Figure 7. Signpost in stand-alone mode. Note the registered 3D overlays in the video image

Relying on the PDA's touch screen capabilities allowed us to create a graphical user interface (GUI), as in Figure 8, which we found easy and intuitive. The GUI primarily consists of a menu through which several settings, such as selecting the start and end points as well showing the current position on an overview map, are available.

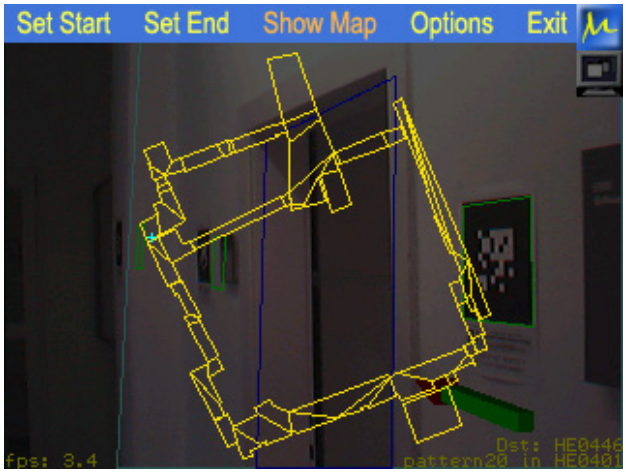

Figure 8. Signpost showing the overview map (current user position marked with small cross)

While the menu is active, the Signpost application continues working as normal. To improve the menu's readability we darken the rendered video output. This allows uninterrupted visibility of the current position on the overview map while being guided to the selected target via the registered 3D overlays.

The graphical user interface is built upon PocketFrog [26], an open source 2D graphics library for the PocketPC platform.

5. Results

Surprisingly, despite the low camera resolution of 320x240 pixels, the registration accuracy of the PDA Signpost client was very satisfying.

Since performance is a major issue on PDAs, we made an intensive analysis of where the system's bottlenecks are located. Table 1 shows how much time is spent in each subsystem for each type of client/server mode during application usage. By optimizing the PocketPC ARToolKit version using the Intel fixed point library we achieved a performance speedup of more than 200%. But as Table 1 shows, doing the ARToolKit calculations still costs almost half of the processing time in standalone mode. Another 41% of the overall frame time is spent in reading camera images.

In server-assisted (non-parallel) mode a PC takes over the tracking task. All ARToolKit related work is now done in approximately 50ms which is about 3.5 times faster than in standalone mode. For further optimization we added the server assisted parallel mode. While the PDA waits for the server's reply it reads in the next frame's camera image. This introduces additional lag, but enhances the overall frame rate. The tracking task then takes up only 8% of the overall performance due to network overhead. At this stage a single bottleneck dominates the overall performance: The application now spends 68% of the frame time for reading the camera image. This issue raises the need for faster cameras and/or faster extension buses, which is outside our control. However, we are confident that the demand for video broadcast on handheld devices will soon resolve these particular hardware issues.

We were able to achieve constant overall frame rates of approximately 5.0 fps in server assisted parallel mode. When performing the ARToolKit calculation on the PDA itself, the frame rate varies with the number and size of markers currently visible between 2.5 fps up and 3.5 fps.

6. Conclusions and future work

We consider the outcomes described in this paper significant because they represent the first mobile 3D Augmented Reality application that is immediately useful and runs on socially acceptable, off-the-shelf hardware. The business and games market pushing video conferencing and mobile recreational activities will fuel the demand for faster and more capable PDAs. We can reasonably expect devices with built-in 3D acceleration to become available soon. This trend is manifested in the appearance of products with built-in wireless connectivity and cameras which will allows us to use completely unmodified off-the-shelf devices for Augmented Reality applications. Therefore we think that handheld devices will be important for future AR applications.

One may consider turning the handheld PDA into a wearable by equipping the PDA with a graphics output port to drive a monocular HMD. The PDA itself may then be attached to the belt and be used as a touch pad only. Although this frees the second hand and make the system wearable in the classic sense, we do not believe that users prefer this setup to the usual handheld PDA working style. A wearable PDA setup consumes more power because of the HMD, is more expensive and more complicated to use.

Future work will focus on generalizing all the presented concepts into a more data-driven framework, which avoids the pitfalls of proprietary applications. Moreover, while the current work has used only widely available hardware, the next step will be to support more sophisticated sensors that can easily be added to commercial products, such as inertial sensors, GPS, or RFID readers. The recently presented IS-1200 hybrid tracker [6], which provides a combination of high accuracy and small form factors, is also a promising tracking device for a PDA. To manage such sensor extensions, we are working on porting our OpenTracker [20] software component to the PDA platform, which will provide easy access to a broad range of tracking solutions. In the near future we plan to release all parts of this work under an open source license.

	absolute time (ms)	% in stand alone version	% in server-assisted version
read camera image	161	41%	68%
24 to 32 bits conversion	8	3%	4%
artoolkit on iPAQ	138 (varies heavily)	44%	
artoolkit over PC	16 (41 non-parallel)		8%
draw video background	6	2%	3%
3d render tasks	15	5%	8%
application overhead	17	5%	9%

Table 1. Performance statistics in stand-alone mode and server-supported mode.

7. Acknowledgements

This project was sponsored by the Austrian Science Fund FWF under contracts no. P14470INF and Y193. We thank G. Reitmayr, M. Knapp, T. Pintaric and A. Walzer for their help.

References

http://www.ims.tuwien.ac.at/research/handheld_ar/

[1] Amselem, D. A window on shared virtual environments. *Presence, Vol. 4, No. 2*, pp. 130-145, 1995.

[2] Advanced Realtime Tracking GmbH, http://www.ar-tracking.de

[3] Bachmann, E., Duman I., Usta, U., McGhee R., Yun, X., and Zyda, M., Orientation tracking for Humans and Robots Using Inertial Sensors. *International Symposium on Computational Intelligence in Robotics & Automation (CIRA 99)*, Monterey, CA, pp.187-194, 1999

[4] Billinghurst, M., Kato, H., Weghorst, S. and Furness, T. A. A Mixed Reality 3D Conferencing Application. *Technical Report R-99-1 Seattle: Human Interface Technology Laboratory, University of Washington,* 1999

[5] Fitzmaurice, G. W. Situated Information Spaces and Spatially Aware Palmtop Computers. *Communications of the ACM*, Vol.36, Nr.7, pp 38-49, July 1993

[6] Foxlin, E., and Naimark L. VIS-Tracker: A Wearable Vision-Inertial Self-Tracker, *IEEE Virtual Reality 2003 (VR2003)*, March 22-26, 2003

[7] Gabber, E. and Wool, A. How to prove where you are: tracking the location of customer equipment, *Proceedings of the 5th ACM conference on Computer and communications security*, pp. 142-149, 1998

[8] Gausemeier, J., Fruend, J., Matysczok, C., Bruederlin, B., and Beier, D., Development of a real time image based object recognition method for mobile AR-devices. *Pro.c 2nd international conference on Computer graphics, Virtual Reality, visualisation and interaction in Africa*, pp. 133-139, 2003

[9] Höllerer, T., Feiner, S., Terauchi, T., Rashid, G., and Hallaway, D. Exploring MARS: Developing Indoor and Outdoor User Interfaces to a Mobile Augmented Reality System. *Computers and Graphics, 23(6), Elsevier Publishers, Dec. 1999*, pp. 779-785, 1999

[10] Humphreys, G., Houston, M., Yi-Ren Ng, Randall, F., Ahern, S., Kirchner, P., and Klosowski, J.T., Chromium: A Stream Processing Framework for Interactive Graphics on Clusters. *Proceedings of SIGGRAPH 2002*, pp. 693-702, 2002

[11] Ingram, D., Newman, J., Augmented Reality in a Wide Area Sentient Environment. *Proceedings of the 2nd IEEE and ACM International Symposium on Augmented Reality (ISAR 2001)*, October 2001, New York

[12] Julier, S.J., Baillot, Y., Lanzagorta, M., Brown, D., and Rosenblum, L., BARS: Battlefield Augmented Reality System, *NATO Information Systems Technology Panel Symposium on New Information Processing Techniques for Military Systems*, October 2000.

[13] Kalkusch, M., Lidy, T., Knapp, M., Reitmayr, G., Kaufmann, H., and Schmalstieg, D. Structured Visual Markers for Indoor Pathfinding. *Proceedings of the First IEEE International Workshop on ARToolKit (ART02)*, 2002

[14] Knapp, M., Signpost 2.0 Project Page, http://www.cg.tuwien.ac.at/~knapp/Signpost2/

[15] MacIntyre B. and Feiner S. A Distributed 3D Graphics Library. *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pp. 361-370, 1998.

[16] Mogilev, D., Kiyokawa, K., Billinghurst, M., Pair, J., AR Pad: an interface for face-to-face AR collaboration. *CHI '02 extended abstracts on Human factors in computer systems*, pp. 654-655, April 2002

[17] Piekarski, W. and Thomas, B. Tinmith evo5 - An Architecture for Supporting Mobile Augmented Reality Environments. *2nd Int'l Symposium on Augmented Reality*, pp 177-178, New York, NY, Oct 2001.

[18] Polhemus, http://www.polhemus.com

[19] Regenbrecht, H.T., and Specht, R., A Mobile Passive Augmented Reality Device – mPARD. *Proceedings o ISAR*, October 2000

[20] Reitmayr, G., and Schmalstieg, D., An Open Software Architecture for Virtual Reality Interaction. *Proceedings of VRST'01, Banff, Canada*, Nov. 15-17, 2001

[21] Reitmayr, G., Schmalstieg, D. "Location based Applications for Mobile Augmented Reality", *Proc. 4th Australasian User Interface Conference*, Adelaide, Australia, Feb. 2003

[22] Rekimoto, J. TransVision: A Hand-held Augmented Reality System for Collaborative Design. *Proceedings of Virtual Systems and Multi-Media (VSMM '96)*, Sept. 1996

[23] Rekimoto, J., and Nagao, K. The World through the Computer: Computer Augmented Interaction with Real World Environments, *User Interface Software and Technology (UIST '95)*, pp. 29-38, November 14-17, 1995

[24] Ribo, M., Lang, P., Ganster, H., Brandner, M., Stock, C., Pinz, A. Outdoor AR tracking. *CG&A 2002 Vol.22, No.6*, pp. 54-63, Nov. 2002

[25] Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavari, Z., Encamao, L. M., Gervautz, M., and Purgathofer, W. The Studierstube augmented reality project. *Presence - Teleoperators and Virtual Environments* 11(1), 2002

[26] Thierry Tremblay, PocketFrog, http://pocketfrog.droneship.com/

[27] Zyda, M., Gossweiler, R., Morrison, J., Singhal, S., and Macedonia, M. Panel: Networked Virtual Environments. *Proceedings of the Virtual Reality Annual International Symposium, VRAIS '95*, IEEE Computer Society Press, pp. 230-231, 1995

[28] SundialSoft, PocketGL, http://www.sundialsoft.freeserve.co.uk/pgl.htm

[29] Bellard, Fabrice, TinyGL, http://fabrice.bellard.free.fr/TinyGL

[30] Khronos Group, OpenGL|ES, http://www.khronos.org/

[31] Lewis, S.A., Havey, G.D., Hanzal, B., Handheld and Bodyworn Graphical Displays, *Proceedings of 2nd. International Symposium on Wearable Computers*, pp. 102-107, Oct. 19 - 20, 1998