ARToolKit on the PocketPC Platform

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 port of ARToolKit onto the PocketPC platform including optimizations that led to a three-fold speedup over the native cross-compiled version. The ported ARToolKit module was successfully used in the Handheld AR project.

Keywords: ARToolKit, PocketPC, Augmented Reality

1. Introduction

In the last few years ARToolKit [1] has become the primary inexpensive optical tracking solution for many Augmented Reality developers. ARToolKit is a suitable tracking solution for many purposes, since it works with a single camera operating in the visible light range, and thus allows a handheld device to work autonomously relying only on passive infrastructure in the form of fiducials attached to objects. This makes ARToolKit a primary candidate for a stand-alone tracking solution on ultra mobile devices such as PDAs (see Figure 1).

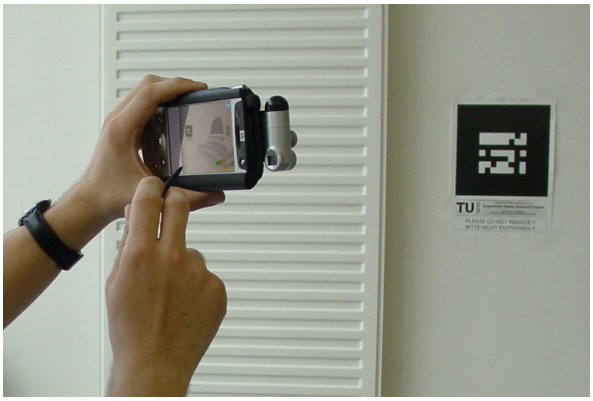

Figure 1. The handheld tracking a fiducial.

2. Related Work

Most previous projects used a client/server architecture for tracking tasks. Rekimoto used color-coded stickers to track objects in the environment with his NaviCam [4]. A workstation analyzed the gathered images and created an augmented view that displayed textual information. Mobility was limited due to the tethered nature of the setup. In the AR-PDA project natural feature tracking is used. The Computer Vision and rendering tasks are outsourced to a server by streaming digital images from and to an application server. The PDA acts only as a thin client whereas the server is responsible for computational intensive tasks. The Mobile Reality [4] project used a combination of infrared beacons and inertial tracking to track the current pose for a location-based service application.

3. Porting ARToolKit

ARToolKit contains several modules such as a VRML renderer, camera abstraction and the tracking code itself, but we were only interested in the tracking core of ARToolKit, which can easily be combined with existing solutions for graphics, device management etc., and should be of strongest interest for the majority of developers. To our surprise, porting this portion of ARToolKit to the PocketPC/Windows CE platform was relatively straight forward.

The original tracking code is implemented in pure C and therefore very portable. After fixing a minor issue in memory management (Windows CE does not permit extremely large static variables; instead, it requires dynamic allocation for such data 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 non-optimized version could analyze

between 1 and 3 frames per second depending on the size and number of visible fiducials.

As a consequence the next step consisted in analyzing the performance to locate bottlenecks. Unfortunately, there were no usable profiling tools for such work at that time. 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. For fixed-point code it is important to know the minimum and maximum values used to correctly choose the number of fraction bits. From this information, a small number of heavily used computation functions were identified together with the information relevant for fixed-point optimization.

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 [3], we achieved a threefold speedup, yielding 5-8 pose estimates per second using the Microsoft ARM compiler when the XScale CPU can be utilized exclusively (i.e., no other tasks executed concurrently). The occurring range of numbers (minimum and maximum values) varies strongly in different ARToolKit methods which made it necessary to adapt the number of fraction bits to each specific function that we converted to fixed-point notation. By using the Intel XScale compiler we were able to gain another speedup of roughly 100% which results in 10-15 pose estimations per second.

4. Applications

ARToolKit PDA version was used in the Handheld AR project for the "Signpost on PDA" application. The Signpost application [5] guides a user through an unfamiliar building to a destination room.

The AR Kanji Learning application [7] is a round base game that runs on PDAs. Two players are facing each other with a pile of cards between them. The user must search for the Kanji letter corresponding to the symbol that is shown on the PDA's screen. After turned around, the card is augmented with the corresponding object the PDA's screen.

5. Results

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. In our tests we were able to achieve 10-15 pose estimations per second with ARToolKit.

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 19x19 cm^2 marker.

The camera devices are the performance bottleneck on today's PDAs in AR applications. In our tests with current consumer Compact Flash cameras we were not able to read more than 7-8 camera images per second which limits the overall performance dramatically.

As PDAs become more popular, devices with integrated multimedia capabilities will soon reach the market. We envisage future devices having fast integrated cameras, faster CPUs with dedicated signal processing units such as Wireless MMX [4]. We expect that such hardware together with ARToolKit will provide all the necessary ingredients for truly mobile, compelling AR applications.

6. Acknowledments

This project was sponsored by the Austrian Science Fund FWF under contracts no. P14470INF and Y193, and the Austrian Ministry of Science *bm:bwk* as *Forschungsinfrastrukturvorhaben #TUW16*/2002.

References

[1] 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

[2] Goose, S., Wanning, H., Schneider, G., "Mobile Reality: A PDA-Based Multimodal Framework Synchronizing a Hybrid Tracking Solution with 3D Graphics and Location-Sensitive Speech Interaction", *Proceedings. UBICOMP 2002*, Sept. 29 – Oct. 1, Göteborg, Sweden

[3] Intel Corporation, "Intel Graphics Performance Primitives" http://www.intel.com/design/pca/ applications processors/swsup/gpp.htm

[4] Intel Corporation, "Intel Wireless MMX Technology" , http://www.intel.com/design/pca/prodbref/ 251669.htm

[5] Reitmayr G., Schmalstieg D.: Location Based Applications for Mobile Augmented Reality. *Proceedings of the 4th Australasian User Interface Conference*, pp. 65-73, Adelaide, Australia, Feb. 2003.

[6] 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

[7] Wagner, D., Barakonyi, I., "Augmented Reality Kanji Learning", *Demo to appear at ISMAR'2003*, Oct. 7-10 2003

[8] Wagner, D., Schmalstieg, D. "First Steps Towards Handheld Augmented Reality", To appear in: *Proceedings of the 7th International Conference on Wearable Computers*, White Plains, NY, USA, Oct. 21-23, 2003