Building Wide-Area Applications with the AR Toolkit

Martin Wagner

Technische Universität München
Institut fuer Informatik
Boltzmannstrasse 3, 85748 Garching bei Muenchen, Germany
wagnerm@in.tum.de

Abstract— **We show how splitting the AR Toolkit in several functional components leads to an extension of its capabilities. We present how these AR Toolkit components can be used to extend the functionality of the Toolkit to wide area indoor tracking. The marker detection gets reconfigured whenever the system's user comes near so-called *transitional markers*, resulting in the possibility of integrating several AR Toolkit based applications within a single system.**

The cost of a wide area tracking infrastructure based on optical marker detection is very low, as only some markers have to be printed out and attached to known places. The AR Toolkit provides all features necessary for this applications, but has a limited number of markers that can be stored in and distinguished by the system at a given point in time. Wide area tracking usually requires many known features in the environment.

Splitting the AR Toolkit library in several components (image acquisition, marker detection and processing of the marker information) allows the realization of this wide area tracking scenario. Every component gets implemented in a separate process, communicating with each other using the DWARF system. As a result, the output of every component can be used by several others.

To solve the problem of a limited number of markers we attach *transitional markers* at prominent places like entrance doors to buildings. Once the system detects such a marker, it loads a set of normal markers. In addition, a new set of transitional markers is loaded. Whenever the system detects one of these, it unloads the current set of markers and switches to a new set of markers. This feature allows to create an "AR Toolkit enabled building", where every room contains a distinct AR Toolkit based application. The doors of these rooms carry huge transitional markers, allowing the system to perform coarse tracking and switch to adequate high-precision tracking applications.

Using the concept of transitional markers in combination with an encapsulation of AR Toolkit functionality in various components will allow a more seamless integration of wide and narrow area tracking within a single application.

I. Introduction

The AR Toolkit [9] provides a convenient way to implement new ideas in the field of Augmented Reality (AR). It is a software library that contains code for image acquisition, the detection of markers in this video stream, the calculation of the markers' three-dimensional position and orientation, identification of markers, the computation of virtual objects' position and orientation relative to the real world and the rendering of these virtual objects in the corresponding video frame. Most existing applications using the AR Toolkit (e.g. [8]) adhere to the given flow of data. A video stream is grabbed, analyzed according to a small set of marker information, the identity and position of these markers is calculated and finally some virtual objects get rendered. Usually all these steps are performed repeatedly in a single loop.

This approach is well suited for a small stationary setup. As the AR Toolkit's marker identification code relies on matching certain regions of interest in the video stream to a preloaded set of marker patterns, the maximum number of distinguishable patterns is clearly limited, resulting in a rather small working range of the applications implemented so far. In contrast, wide area tracking is usually less demanding in accuracy, however, the number of features to be distinguished by a tracking system that relies on peculiarities of the environment the tracked object is in grows linearly with the tracking range.

This paper deals with extending the AR Toolkits functionality to allow not only small stationary setups but wide-range tracking applications as well. The main advantage of this approach is that a single, reliable and well-tested environment can be used for both coarse and fine granular tracking tasks.

The paper is organized as follows. We start with an overview of related work in wide-area tracking and the combination of wide- and narrow-range tracking technologies. We then present the idea of so-called *transitional markers* that allow us to extend the AR Toolkit's functionality to wide-range environments. The implementation of this concept is based on finite state machines and discussed in detail in section V. To allow a full exploitation of AR Toolkit including its capability for real-time 6D tracking, we split up its functionality in two parts:

the first consists of image aquisition and marker detection, the second on pose estimation based on the detected markers. We propose that each part runs as a separate component, communicating to each other using our DWARF framework [6], [3], [4]. The details of this communication are discussed in section IV. We sum up the paper with a conclusion and some ideas about dividing the Toolkit in even more components.

II. Related Work

The problem we address in this paper has been treated extensively in the tracking community. Much work has been put in the development and evaluation of wide-range tracking methods. In addition, there exist a few papers discussing ideas about a combination of wide- and narrow-range tracking technologies, some ideas from these publications gave valuable input to the design of our approach.

A. Wide-Range Tracking

Let us start with an overview of existing wide-range systems. Perhaps the most often used existing system is the *Global Positioning System* GPS [7]. Its major advantage of working everywhere on the planet in combination with low cost is opposed by the requirement to have a clear look at the open sky. As such, it does not work indoors, a problem easily solved using the AR Toolkit.

In contrast, the AT&T Laboratories Cambridge *Bat System* [1] is based on ultrasonic sensors and allows building-wide tracking with an accuracy of 3cm. In addition, there has been some interesting work on extending the Bat system's functionality to real Augmented Reality applications [11]. Obviously, the installation of such an infrastructure yields a high cost penalty and is therefore not applicable in general.

There exists some work on vision-based methods for outdoor registration. Satoh et al. [13] have proposed a combination of gyroscope-based tracking with vision methods to compensate for drift effects. Simon et al.[14] propose using planar structures in the scene to get outdoor tracking right. Although both systems just mentioned provide outdoor tracking capabilities, they rely on a set of known features in the environment that obviously has to be small enough to create a significant discriminator. Our approach of transitional markers overcomes this restriction and can in principle be combined with these methods as well.

B. Combining Multiple Tracking Methods

Every single tracking technology has its particular strengths and limitations. To overcome these, many research groups have put work in combining a variety of tracking methods in a single application.

Auer and Pinz [2] have integrated optical and magnetic tracking within the Studierstube environment, later Reitmayr and Schmalstieg extended this work to a general XML-based architecture [12].

Klinker et al. [10] propose a more general framework to seamlessly incorporate sensors in intelligent environments into user-space applications.

Both systems just discussed provide the capability to integrate many diverse tracking technologies in a single application. However, they do not explicitly propose a concept how the current set of tracking devices should be chosen at a given point in space and time.

III. Transitional Markers

We think that a wide-area tracking problem can be solved best using a finite state machine. The inherent proximity metrics of three-dimensional space allow us to separate the working space into relatively small regions with a clearly limited size of markers that can be tracked. In addition, tracking technologies available in this region can be chosen accordingly.

A. An Example Scenario

To give an example, consider a user walking from his home to an office building. The user is equipped with a wearable computer having a camera attached to it and running the AR Toolkit software. In addition, for navigation in unknown open-sky environment, a GPS receiver is connected to the system. The whole system is shown in figure 1.

Fig. 1
The Wearable System

The system detects that the user passed by his front door by getting an event from the AR Toolkit that he came by a huge known marker attached to the door (see figure 2). We call such a marker a *transitional marker*, as it triggers a state transition in the finite state machine controlling our application. Once the user steps on the street, the system gets into a new state and starts the GPS tracker to get hold of a rough approximation of the user's position on the way to his office.

Fig. 2
The transitional marker attached to the user's front door

After a while, the user reaches his office building and the AR Toolkit subsystem detects another transitional marker at the building's front door. As such, it can trigger a new state transition. In the user's company's entrance hall, an AR Toolkit based application may show him today's work. For this application, a set of markers has to be loaded in the system. Based upon these markers, the AR Toolkit is used as intended and performs regular 3D overlay in a HMD the user just put on.

After a while, the user walks into his floor, the system detects this as it sees another transitional marker. The entrance hall's Toolkit markers are unloaded and a new set of transitional markers, one for each door in the floor, is loaded. Once the user enters his room, the system detects another transitional marker and loads a final set of markers suitable to the user's room.

B. Design Considerations for Transitional Markers

Note that the concept of transitional markers triggering a finite state machine is applicable in general and not limited to AR Toolkit based applications. In our work, however, we chose the AR Toolkit as the single tracking engine doing both wide-range and fine granular tracking. AR Toolkit has been designed for small applications with a clearly limited range of operation, in consequence, we had to get new experience for using the AR Toolkit in wide-range applications.

We found out that some aspects have to be kept in mind when choosing a suitable set of transitional markers.

- The detection of transitional markers is crucial to the reliability of the whole system. As such, they should be rather huge and attached to places the user must pass by. Examples for these places include doors and checkpoints.
- Misclassification of transitional markers should be avoided under all circumstances. We found out that a marker should only be classified as "seen" by the system if the AR Toolkit detection routine reported it in several subsequent calls. In addition, the marker's *confidence value* should be rather high. In our experiments, a threshold of $cf = 0.7$ seamed reasonable.
- Clearly, the transitional markers are loaded at the same time as the markers of the current AR Toolkit application. Therefore they must be different from every such marker. In addition, they should be absolutely unique and detectable reliably by the AR Toolkit, as a spuriously detected marker may yield the system in an inconsistent state.
- In contrast to existing AR Toolkit applications, the user should not attach the camera to his head. Instead, if the camera gets mounted in front of his chest, it moves more slowly and gets more reliable results for transitional markers. Of course the transitional markers should be mounted in approximately the same height.

IV. Splitting the AR Toolkit in Components

Most existing AR Toolkit applications are monolothic. They use the library and adhere to its given flow of data, consisting of the following steps:

1. Image acquisition, using operating system specific code
2. Image analysis, detecting previously registered markers

3. Marker identification, based on the results of image analysis
4. Calculation of relevant positional and orientational data
5. Drawing of an OpenGL scene based on this data, either in optical or video see-through mode

This approach is well suited for demo applications evaluating new ideas in the field of Augmented Reality. However, we run into problems whenever code of such demos should be reused or even combined with code of other applications.

This is the case if we want to create an *AR Toolkit enabled* floor. The basic idea of such a system is to have a different AR Toolkit based application in every single room of a floor. The proposed wide-range tracking with transitional markers detects which room the user is in and triggers a room-specific application. This application uses the AR Toolkit's fine granular tracking features to perform realtime overlay of real and virtual objects.

Obviously, it would be comfortable if new applications in new rooms could be added easily to the system without the need to rewrite major parts of it. Analyzing existing AR Toolkit applications, we found out that the parts of code controlling the image acquisition and marker detection are nearly always the same, and that these applications differ mainly in the way they handle this information.

In consequence, we can split up the AR Toolkit's functionality in two components:
1. Image acquisition and Image analysis, yielding an output of `ARMarkerInfo` structures.
2. Marker identification based on `ARMarkerInfo` structures and further processing of this information.

To integrate existing applications in the concept we propose is easy. Component 1 has to be written and started once. All other components that need to know what the system currently knows about AR Toolkit markers can access this component simultaneously. To adopt them to the concept consists of replacing the existing image acquisition code by code that handles the communication with the new acquisition component.

The greatest advantage of componentizing the AR Toolkit is the possibility to use the same video camera setup for wide- and narrow-range tracking. The image is analyzed once and the results are processed both by wide- and narrow-range tracking code.

V. Implementation Details

This section discusses the technical details of how we propose to implement the combination of wide-range tracking with common AR technology offered by the AR Toolkit. First, we discuss how we split up the Toolkit's functionality in components. Afterwards, we show how this can be extended with a finite state automaton to get the full benefits of the transitional markers concept.

A. DWARF as Component Framework

In our research, we developed the DWARF framework that allows the creation of component-based augmented reality applications. As discussed in [4], [3], [6], the DWARF system allows the dynamic combination of components running on computers in the network at runtime. The matching of components is based on so-called *Needs* and *Abilities*.

Based on this framework, we split the usual workflow of the AR Toolkit in the two components discussed in section IV, image acquisition and marker detection on one side and processing of this information on the other side.

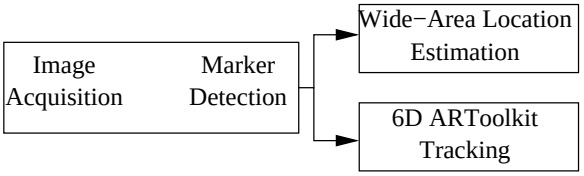

Fig. 3
AR Toolkit Components

Both components run in separate processes and may even be distributed across different computers for load balancing purposes. They communicate with each other using the CORBA Notification Service [5]. Events of type `CORBA::Any` are sent over a bus architecture to all other components that are interested in them. To ensure a correct data and language mapping on different architectures and programming languages, the `CORBA::Any` data must consist of a single structure adhering to the interface definition shown in figure 4.

This event bus architecture allows us to process the output of the AR Toolkit marker detection routines multiple times independently. In our application, we have one component that does the wide range tracking and another component that uses the remaining AR Toolkit library functions to perform realtime augmented reality in a Head Mounted Display. Figure 3 shows this general architecture.

B. The Finite State Machine

The second interesting problem in the implementation is how to control the behaviour of the application to incorporate the special role of transitional

```
module DWARF {
   struct Time {
     unsigned long seconds;
     unsigned long microseconds;
   };
   module Markers {
      /**
        * This struct is filled out when
        * an AR Toolkit marker is detected
        * and sent in the body of a
        * structured event. Only the "id"
        * attribute is meaningful to an
        * application that is only
        * interested in the visibility of
        * a marker. The rest is specific
        * to the AR Toolkit and only used
        * by explicit AR Toolkit
        * applications.
        */
      struct ARMarkerInfo {
         long   area;
         long   id;
         long   dir;
         double cf;
         double pos[ 2 ];
         double line[ 4 ][ 3 ];
         double vertex[ 4 ][ 2 ];
         /// The time the marker
         /// was detected
         Time timeStamp;
      };
   };
};
```

Fig. 4
IDL definition of AR Toolkit Marker Events

markers.

As mentioned above, these special markers trigger transitions in a finite state machine. Let us discuss the simple example setup depicted in figure 5. The setup consists of a floor and two rooms. Inside each room, an AR Toolkit based application is running. The system should detect which room the user is in, and if he is in room 1, it should start the first AR Toolkit application, in room 2 application 2 respectively.

Obviously, for every door we have to attach two transitional markers, so we have six transitional markers altogether. In addition, every application needs its own set of markers. As the two applications are in distinct rooms, and the controlling application registers a change of room, these two sets

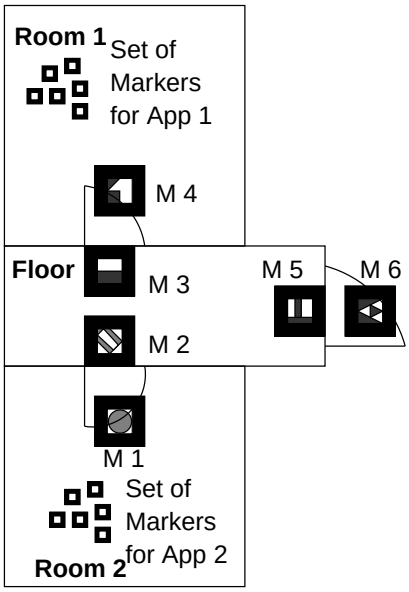

Fig. 5
Example transitional marker setup

can be overlapping or equal, leading to a reusability of markers.

The finite state automaton that controls the general application is depicted in figure 6 and works as follows.

Initially, we assume the user to be outside the whole setup. The system is in state `Init` and the AR Toolkit component is configures such that it only detects marker *M6*. Once it detects *M6*, it switches to state `Floor`. The AR Toolkit feature detection is now reconfigured in a way that it looks only for the transitional markers *M2, M3* and *M5*.

If the user enters a room, say Room 2, the state is switched again according to the state transition scheme in figure 6 and the current set of transitional markers is replaced by a new one consisting of a single marker *M1*. In addition, the set of markers for Application 2 is loaded. If the user leaves the room, this set has to be unloaded along with the current set of transitional markers.

C. Controlling the AR Toolkit from the Finite State Machine

The last problem we have to handle is how the transitions in the finite state automaton can change and control the image acquisition and marker detection component.

The detection component itself can handle the invariable settings such as camera parameters and access methods to the video stream. The variable and

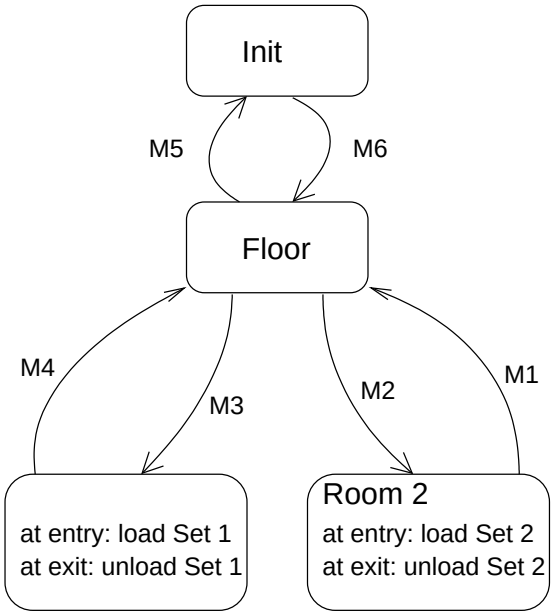

Fig. 6
The finite state automaton of the example setup

```
module DWARF {
   module Markers {
      interface MarkerDetector {
         /**
           * Registers a new Marker, the
           * standard description file can
           * be found in URL.
           * Returns a unique id needed for
           * removal
           */
         int registerMarker(in string URL);
         /**
           * Unregisters a previously
           * registered marker.
           */
         void unregisterMarker(int id);
       };
   };
};
```

Fig. 7
Control interface of detection component

application-dependent settings have to be changed using remote object calls. Figure 7 gives an example how an IDL definition of such a control facility could look like.

VI. Conclusion and Future Work

We have described a new concept of extending the AR Toolkit to allow wide-range tracking. Based on the idea of transitional markers, a finite state machine configures the AR Toolkit marker detection routines in a way that allows to solve the problem that the AR Toolkit can not handle too many markers at the same time.

The current implementation is based on the Dwarf system and splits the AR Toolkit library in two components, one handling the marker detection and the other doing the calculation of the marker's position and orientation relative to the camera and the video overlay. In addition, a prototypical finite state machine uses the information about the detected markers to perform coarse tracking and reconfigure the detection component.

A. Proposals for the AR Toolkit

To handle the dynamic loading and unloading of patterns, the AR Toolkit provides the procedures `arLoadPatt`, `arActivatePatt`, `arFreePatt` and `arDeactivatePatt`. These procedures are well suited for regular small applications with a clearly limited and fixed set of markers. However, for large-scale applications, an intelligent algorithm allowing the reuse of patterns that have been loaded already could be advantegeous.

In addition, the currently static memory mapping of the marker structs should be changed to dynamic allocation in order to allow a more flexible style of use.

B. Future Work

The current implementation is only a proof of concept and does not yet allow arbitrary room topologies for wide area tracking. To allow this, a finite state machine that can be configured using XML should be developed in the near future. Once this is done, the idea of transitional markers could be evaluated in a larger application involving multiple users.

To increase the robustness of the wide-range tracking, some studies on misdetection and similarity of AR Toolkit markers should be conducted. One way to go could be to use the new multi marker feature of the AR Toolkit. In every case, strategies for resolving inconsistent system states resulting from misclassification have to be developed.

Finally, the idea of splitting up the AR Toolkit's functionality in components could be driven even further. Every single piece of the Toolkit's functionality, from image acquisition to the OpenGL drawing routines, could be encapsulated in a single component.

Acknowledgements

The work described in this paper was partially supported by the High-Tech-Offensive of the Bayerische Staatskanzlei.

The author would like to thank Martin Bauer, Asa MacWilliams, Christian Sandor and Thomas Reicher for their work on DWARF. Special thanks to the students of the TRAMP project's Context Team for their support in testing some of the ideas described in this paper.

References

[1] Mike Addlesee, Rupert Curwen, Steve Hodges, Joe Newman, Pete Steggles, and Andy Ward. Implementing a sentient computing system. *IEEE Computer*, August 2001.

[2] Thomas Auer and Axel Pinz. Building a Hybrid Tracking System: Integration of Optical and Magnetic Tracking. In *Proceedings of the International Workshop on Augmented Reality*, San Francisco, CA, 1999.

[3] Martin Bauer, Bernd Bruegge, Gudrun Klinker, Asa MacWilliams, Thomas Reicher, Christian Sandor, and Martin Wagner. Design of a Component-Based Augmented Reality Framework. In *Proceedings of the International Symposium on Augmented Reality*, New York, NY, 2001.

[4] Martin Bauer, Bernd Bruegge, Gudrun Klinker, Asa MacWilliams, Thomas Reicher, Christian Sandor, and Martin Wagner. An Architecture Concept for Ubiquitous Computing Aware Wearable Computers. In *International Workshop on Smart Appliances and Wearable Computing*, Vienna, Austria, 2002.

[5] CORBA Notification Service Specification. Object Management Group, `http://www.omg.org/technology/documents/formal/notification_service.htm`, February 2001.

[6] DWARF Homepage. `http://www.augmentedreality.de`.

[7] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins, editors. *GPS – Theory and Practice*. Springer, Vienna, New York, 4th edition, 1997.

[8] Hirokazu Kato, Mark Billinghurst, Ivan Poupryev, K. Imamoto, and K. Tachibana. Virtual Object Manipulation on a Table-Top AR Environment. In *Proceedings of the International Syposium on Augmented Reality*, Munich, Germany, 2000.

[9] Hirokazu Kato, Mark Billinghurst, and Ivan Poupyrev. *ARToolKit version 2.33 Manual*, 2000. Available for download at `http://www.hitl.washington.edu/research/shared_space/download/`.

[10] Gudrun Klinker, Thomas Reicher, and Bernd Bruegge. Distributed User Tracking Concepts for Augmented Reality Applications. In *Proceedings of the International Symposium on Augmented Reality*, Munich, Germany, 2000.

[11] Joseph Newman, David Ingram, and Andy Hopper. Augmented Reality in a Wide Area Sentient Environment. In *Proceedings of the International Symposium on Augmented Reality*, New York, NY, 2001.

[12] Gerhard Reitmayr and Dieter Schmalstieg. OpenTracker – An Open Software Architecture for Reconfigurable Tracking based on XML. In *Proceedings of the ACM Symposium on Virtual Reality Software & Technology (VRST)*, Banff, Alberta, Canada, 2001.

[13] Kiyohide Satoh, Mahori Anabuki, Hiroyuki Yamamoto, and Hideyuki Tamura. A hybrid registration method for outdoor augmented reality. In *Proceedings of the International Syposium on Augmented Reality*, New York, NY, 2001.

[14] Gilles Simon, Adrew W. Fitzgibbon, and Andrew Zisserman. Markerless Tracking using Planar Structures in the Scene. In *Proceedings of the International Syposium on Augmented Reality*, Munich, Germany, 2000.