# Configuration Strategies of an AR Toolkit-based Wide Area Tracker

Martin Wagner, Felix Loew

Technische Universität München
Institut für Informatik
Boltzmannstrasse 3, 85748 Garching bei München, Germany
(wagnerm,loew)@in.tum.de

## Abstract

*This paper deals with extending the AR Toolkit's functionality to allow not only small stationary setups but wide range tracking applications as well. Based on a visionary scenario of a mobile user walking around in an intelligent building, we split the AR Toolkit's functionality in several components based on our DWARF system. These components work together to allow a spatially distributed configuration of AR Toolkit's marker detection code. The results of the marker detection are used to conclude changes in the mobile user's coarse location. We present different strategies that could be employed for marker-based wide area tracking and discuss environmental factors influencing a good choice of strategy. We implemented the ideas discussed in this paper as part of a larger application and present results from this demonstration setup.*

## 1. Introduction

This paper deals with extending the AR Toolkit's [9] functionality to allow not only small stationary setups but wide range tracking applications as well. One of the core problems with marker-based wide area tracking is that the number of necessary marker descriptions quickly grows beyond an easily manageable size. To overcome this problem, we propose to split this data according to its spatial information. In short, we divide all marker information in small chunks that are stored exactly at the location where they are relevant. Once this data is divided in several parts, it gets easy to add additional contextual information allowing to adapt the wide area tracking strategy to the user's current environment and situation.

The AR Toolkit is a software library that provides image acquisition, marker detection and pose estimation facilities. Most applications using it (e.g. [8]) are implemented as monolithic systems, an approach well suited for a small stationary setup. In our research, we aim at combining ideas from ubiquitous computing [19] with augmented reality. In ubiquitous computing, applications are dynamically built out of small services provided by computing systems in the environment and by systems worn by the users. Thus monolithic systems have to be broken up in component systems to be useful in this context. In earlier work, we split AR Toolkit in several components [18].

To integrate augmented reality in ubiquitous computing environments, we have to provide some means to allow wide area tracking. In this paper, we propose to use AR Toolkit for this purpose as well and not only for local setups. As with all vision based trackers, AR Toolkit has to be configured with features that it should match to the images of the incoming video stream. The number of distinguishable features is clearly limited, as such, we present architectural concepts for dynamic reconfiguration of the AR Toolkit.

The paper is organized as follows. We start with an overview of related work in wide-area tracking, the combination of wide and short range tracking technologies and context aware data management. We discuss how AR Toolkit has to be split up in several components in order to enable the dynamic and distributed configuration of a wide area tracking system. We then present some strategies that may be used for marker-based wide area tracking and give some details on our setup and results we obtained with these strategies. We conclude the paper with some ideas on future extensions.

## 2. Related work

The problem of wide area tracking has been the topic of extensive research, much less work has been put in a combination of wide and narrow range systems to enable ubiquitous tracking with high precision hot spots suitable for AR applications. In this section, we describe related work in the area of wide and narrow range tracking as well as of context aware computing and mobile data management.

The most often used existing wide-range tracking system is the *Global Positioning System* GPS [6]. However, with its requirement to have a line of sight to the open sky, it is useless indoors. 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 [12]. However, the high cost of such a system leads to the investigation of vision based systems that can be set up much more easily.

There exists some work on vision-based methods for outdoor registration. Satoh et al. [15] have proposed a combination of gyroscope-based tracking with vision methods to compensate for drift effects. Simon et al.[16] propose using planar structures in the scene to get outdoor tracking right. Stricker and Kettenbach [17] propose to use prerecorded reference images to provide outdoor markerless tracking. All vision-based systems rely on a set of known features in the environment that obviously has to be small enough to create a significant discriminator. Our approach allows a dynamic reconfiguration of these systems and in particular of the AR Toolkit and extends their working range by allowing a spatial partitioning of feature knowledge.

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 [13]. Based on this architecture, Kalkusch et al. [7] propose to reuse marker information based on knwoledge of the spatial relationships between different rooms. Klinker et al. [10] propose a more general framework to seamlessly incorporate sensors in intelligent environments into user-space applications. All 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 trackers should be chosen at a given point in space and time.

Our approach relies on tracking AR Toolkit markers in the environment, as such, the exact position and the properties of these markers have to be known. In earlier work [18], we presented a system that configured the marker detection according to the right context, however, all information had to be known at once. But as this information can be seen as context, the user needs only the data related to his current context. Hess and Campbell [5] propose a context aware file system to facilitate access to relevant information, whereas Riché and Brebner [14] discuss strategies for minimizing conflicts when multiple users access contextual information simultaneously. Our system puts the focus on how to select spatially distributed information automatically and may use ideas from this work fro large scale deployment.

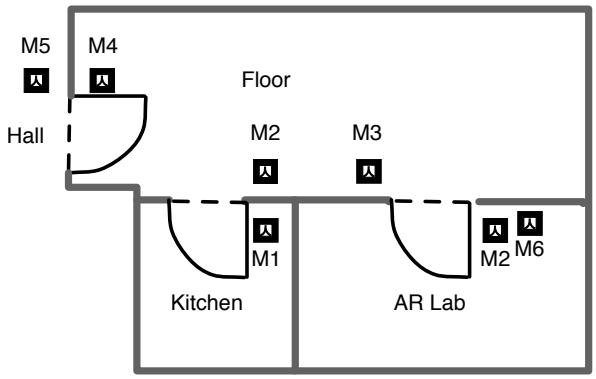


**Figure 1. Example scenario. Note that marker M2 is reused twice in the Floor and the AR Lab.**

## 3. System overview

The ideas presented in this paper have been developed with a larger visionary scenario in mind that illustrates our idea of combining ubiquitous computing with augmented reality. In this chapter, we will first present this scenario and then give a short overview of the capabilities of our system.

In our visionary scenario, a user with a mobile Augmented Reality system walks around an intelligent building (see figure 1). Many applications are available in this building, and most are tied to specific rooms. For example, the fridge in the kitchen has a virtually transparent door, the table in the Augmented Reality Lab uses AR technologies to facilitate desktop computing and the floor provides short speech output about what can be seen behind all the doors. To execute the matching applications at every point in time, the mobile system has to know which room the user is currently in. For this purpose, the user wears a small camera that detects AR Toolkit markers attached to certain specifically suited objects such as doors. For example, the inner side of the kitchen door carries a marker with the semantics "User left kitchen". We do not assume that the mobile system has any knowledge about our building, as we would like to incorporate arbitrary new users. As such, our system provides the semantical information that correlated with certain markers automatically to the mobile system. When the user leaves a room, his mobile system gets reconfigured such that he can use the applications in the new room and that the system correctly detects when he leaves the new room again.

In earlier work [18], we showed how AR Toolkit has to be split up in components to handle the multitude of markers in an efficient manner. The new contribution of our

system is twofold. First, we allow a distributed storage of marker descriptions that facilitates updates of small parts belonging to single rooms. Second, we present some strategies for detecting room changes and discuss their performance depending on environmental conditions.

## 4. Distributed configuration of AR Toolkit

Let us now discuss how we realized the distributed storage of AR Toolkit configuration data. Based on ideas from our DWARF system [3, 11], we split AR Toolkit in several components. By separating out all semantical and configuration information in a *Context* and *MarkerLoader* component, we facilitated the distributed configuration.

### 4.1. Splitting the AR Toolkit in components

Most existing AR Toolkit applications adhere to a 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

To validate our ideas, we used only the first three steps of the AR Toolkit processing pipeline and split them in two components, namely a *VideoGrabber* using platform-dependent code to acquire a video image and put it in a shared memory segment and the *MarkerDetection* incorporating the AR Toolkit library. Note that the latter component sends `ARMarkerInfo` structs using the CORBA Notification Service [4] to all other components interested in, as such, room based applications can use this information to perform the last two steps of the AR Toolkit pipeline.

In addition, we implemented some more components enabling the contributions described in this paper:

**MarkerLoader:** This component configures the *MarkerDetection* component with a set of markers depending on the current context, i.e. the room the user is currently in. Note that conceptually every spatial entity has its own *MarkerLoader* component running in it. Hence, the configuration data is stored according to its spatial organization.

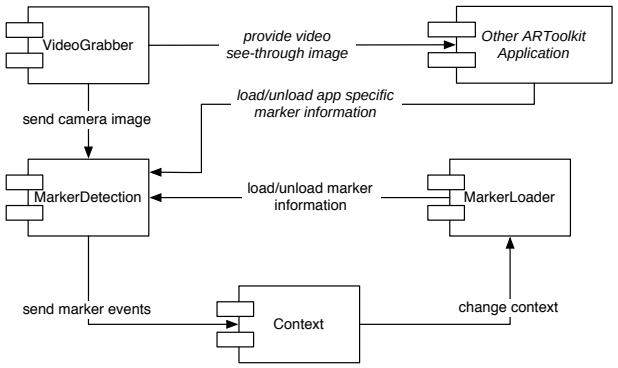


**Figure 2. Overview of all components. Note that the *Other AR Toolkit Application* component has not been implemented in our setup.**

**Context:** This component converts marker events received by *MarkerDetection* and converts them into high level context information. Based on this information, the room context gets changed and the system gets reconfigured automatically using the DWARF middleware [11].

An overview of the dependencies between all these components is shown in figure 2

### 4.2. Distributed configuration

Our basic idea is to distribute the whole information about the environment in the environment itself. As such, conceptually every room has its own *Context* and *MarkerLoader* component running on a dedicated network node in the room. If the user enters a new room, the connections to the old components are interrupted and new connections to the new room's components get established. As a result, the mobile system has only the relevant contextual information at all times. Of course, multiple components representing many rooms can run on a single computer for efficiency reasons.

The advantage of this approach is that all necessary configuration information is split up into small chunks, each only containing information about a single spatial entity. To give an example, consider our setup shown in figure 1 containing four rooms. The *MarkerLoader* in the floor has to configure the AR Toolkit *MarkerDetection* with information about markers M2, M3 and M4, the one in the hall loads information about M5, the kitchen's loader stores M1 and finally the AR Lab loads information about M2 and M6. Note that M2 can get reused in the Floor and the AR Lab without any problems.

The *Context* component carries all semantic information to trigger changes in the current location context. In our example, it changes the context from *Room=Kitchen* to *Room=Floor*, if the marker M1 is detected in a suitable way to conclude that the user has left the Kitchen.

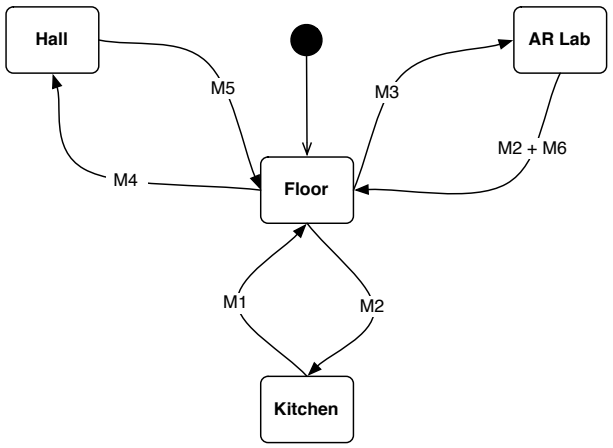


**Figure 3. Example scenario represented as state machine.**

Note that the sum of all *MarkerLoader* and *Context* components can be seen as a state machine having rooms as states and markers as transitions. The resulting state machine for our example scenario is shown in figure 3.

## 5. Strategies for detecting room changes

The *Context* component is responsible for the conversion of marker information to room context. It holds the information about the current environment. As such, we can taylor this component specifically for the current room. In this section, we first present some factors influencing the choice of strategies for detecting room changes based on detected marker and then present strategies we implemented.

### 5.1. Factors influencing the choice of strategy

Choosing a good strategy for detecting that a user changed rooms is not easy. Sometimes the image analysis code wrongly detects markers although we have not seen them. If a change of room context is then triggered because of this, the system gets in a wrong state and additional mechanisms have to be used to undo the change of context. To avoid this it is necessary to find a good strategy for the *Context* component. There are many factors influencing the behavior of the marker detection. For each of these factors certain strategies will be more favorable that others. Factors we found are:

**Light conditions:** Depending on the light conditions markers will be detected easier or not. Usually in full daylight the marker detection is able to detect markers quite robustly. If conditions get darker, the AR Toolkit code tends to classify markers correctly only occasionally.

**Movement of the user:** The marker detection is also influenced by the users movements. If a user moves quite fast a marker might not be detected due to image blur.

**Marker position:** If markers are attached to locations which are not in the view of the user markers might also not be detected.

**Camera:** Depending on the camera properties and the camera calibration the quality of the marker detection varies.

**Room properties:** If a room is a long hall you could assume the direction of the user and the sequence of markers that will be detected. On the other hand a large room with many possiblities to walk to is more difficult to handle.

### 5.2. Basic strategies

Starting from these factors, we designed and implemented several strategies to trigger context changes. Note that these strategies and their exact configuration can vary from room to room, depending on each room's properties.

**Minimum number of frames with marker:** A very basic but often sufficient strategy is to require that a marker is detected several times within a given time frame or number of subsequent image frames. The larger the number of frames gets, the more robust this strategy is. The disadvantage of this strategy is that we have to ensure that the user's camera has the marker to be detected in its field of view for a probably long time. In consequence, the user is restricted in his movements if context changes should be detected correctly.

**Weighting number of frames by confidence value:** It is also possible to take the confidence value of a marker detection into account. A very simple strategy is to only take markers with high confidence value into account. A more sophisticated strategy is to combine the number of frames and the confidence value. A context change will be triggered if the sum of all confidence values during the last time frame is beyond a given threshold.

**Detect several markers at the same time:** Another strategy is to attach two markers next to each other, initiating a context change only together. A transition is only triggered if both markers are detected simultaneously within a given time frame. Obviously, this method can be extended to an arbitrary large number of markers.

**Sequence of markers:** Finally, we can define a sequence of markers triggering the transition. In the most simple setup, a "leave room" marker is attached in the current room and a "confirm" marker is attached in the new room. Only if the *MarkerDection* detects the confirm marker after the leave room marker, a change of context is triggered. Again, this strategy can be extended to a larger sequence, which may be well suited if the user must follow a fixed path to get from one room to another.

All of these strategies can be combined with each other to yield even more robust strategies. In addition, all strategies have parameters (such as the minimum number of frames) that can be fine tuned. These parameters may even be adjusted automatically depending on other contextual information. For example, the current lighting conditions may influence the choice of confidence value, as the AR Toolkit code usually yields better results in brighter environments. If we have no means to estimate the current lighting conditions automatically, we may use the current time of day to get a rough estimate as shown in table 1.

| Light conditions | Number of frames |
|---|---|
| good: full daylight | high (e.g 20-30) |
| medium: normal daylight | middle (10-20) |
| bad: no daylight, halogen lamps | low (5-10) |

**Table 1. Light conditions and number of frames**

### 5.3. Deployment of strategies

Due to our distributed configuration, each room is equipped with its own context component. This component employs a particular fine-tuned strategy that matches the environmental conditions inside the specific room. To deploy a wide area tracking system as decribed in this paper, one has to keep in mind the peculiarities of the building. If for example a laboratory can only be entered by a long vestibule, it is a good idea to apply the "sequence of markers" strategy for the transition from outside to the lab via the vestibule. In contrast, the "multiple markers" strategy may be chosen if the user is expected to stay a rather long time in front of a rather large wall with enough space for multiple markers on it before leaving a room.

## 6. Implementation

This section describes our current implementation and results we obtained with it. We describe an evaluation setup that has been used to conduct some experiments with different strategies under varying conditions.

### 6.1. Implementation details

All components described in this paper were implemented as components of our AR framework DWARF [3]. A DWARF component is a process running on an arbitrary computer. An instance of the DWARF *Service Manager* is running on every computer in the local subnet and connects the individual components according to their *Needs* and *Abilities*. All components described here are implemented in C++, however, alternative implementations in Java or Python are possible due to our flexible CORBA based middleware.

**VideoGrabber** The *VideoGrabber* component uses the Linux *libdc*[1] library to control a camera conforming to the 1394-based Digital Camera Specification[2]. The parameters of the camera such as frame rate, resolution and color format can be described using an XML configuration file. The *VideoGrabber* gets the camera image and writes it to a shared memory segment. Then multiple other DWARF services running on the same computer can access the data using single writer multiple reader sempahores. A single instance of this component is running continuously on the user's mobile computer and connected to all remote components via wireless LAN.

**MarkerDetection** The *MarkerDetection* component incoroporates the AR Toolkit library. It gets the video image from the *VideoGrabber* using shared memory and gets configured via remote CORBA method calls from the *MarkerLoader*. Once a single or several patterns are loaded, the *MarkerDetection* calls AR Toolkit's `arDetectMarker` method and encapsulates the return value to events that are sent via a CORBA event bus to all other interested components. A single instance of this component is running continuously on the user's mobile computer and connected to all remote components via wireless LAN.

[1] `http://sourceforge.net/projects/libdc1394`
[2] `http://www.1394ta.org`

**MarkerLoader** The *MarkerLoader* component is responsible for the correct configuration of the *MarkerDetection*. There are several instances of this component, each corresponding to a single context, i.e. room in our scenario. Each *MarkerLoader* has a list of standard AR Toolkit pattern descriptions that it reads either from the file system or a MySQL database. These descriptions are then sent to the mobile user's *MarkerDetection* component. Using this strategy, the mobile setup does not have to have any local knowledge about the markers in its current room before entering it.

**Context** The *Context* component is responsible for evaluating the events sent from the *MarkerDetection* and creating high-level contextual information from it. This component is running in several instances, each employing a potentially different strategy adopted to its location. Whenever the *Context* component detects that the user has left the current room, it sends out a *Location Event* using the CORBA event bus. This event is then used by all other components interested in to reconfigure according to the new context.

**Speaker** The *Speaker* component receives all location events from the *Context* component and plays prerecorded sound files that explain the user in which room he currently is in. This component is just an example application for the use of contextual information obtained form our wide-area tracking system.

### 6.2. Demonstration setup

The components for AR Toolkit-based wide area tracking have been evaluated as a part of a larger demo project. This project has been called ARCHIE (Augmented Reality Collaborative Home Improvement Environment) and tries to evaluate some new AR technologies for collaborative support of architectural planning. Further information can be found on our website[3].

The mobile setup consisted of two Pentium IV-based Laptops running Linux, a Sony Glasstron HMD and an iBot FireWire camera for marker detection. The setup is shown in figure 4.

We successfully demonstrated the AR Toolkit based wide area tracker incorporating a multitude of *Context* and *MarkerLoader* instances presented in this paper.

**Figure 4. The mobile setup**

## 7. Conclusion and Future Work

In this paper, we have shown how the AR Toolkit's functionality has to be split in several components in order to extend its functionality to wide area tracking. We proposed to store the Toolkit's configuration information distributed according to its spatial organization. As a byproduct of this distributed storage, our concept allows the context aware choice of strategies for marker-based wide area tracking. To enhance AR Toolkit's functionality, it would be nice to integrate a plugin mechanism for varying strategies of the multi-marker tracking feature. We discussed several factors influencing these strategies and presented four basic strategies that in combination fit most indoor situations.

Starting from this design, much more work could be done in the future to enhance the robustness of our approach. With a stringent evaluation of the ideas discussed here, valuable insight in a good choice of strategies could be obtained, finally leading to a semiautomatic configuration of the context estimation component.

Another area of major future work is to consider algorithms for detection and revertion of wrongly triggered transitions of context. Although the general concept of a state machine seems well suited for modelling indoor environments, it is not without problems. Especially if the system assumes to be in another room than it is in, the system state gets inconsistent. Work has to be put in finding robust reconfiguration strategies that allow the system to revert to a consistent state again.

[3] `http://www.augmentedreality.de`

## Acknowledgements

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

The authors would like to thank Gudrun Klinker and Thomas Reicher for helpful discussions and valuable input. Special thanks go to all people who participated in the ARCHIE project.

## References

[1] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, and A. Ward. Implementing a Sentient Computing System. *IEEE Computer*, August 2001.

[2] T. Auer and A. 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] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, C. Sandor, and M. Wagner. Design of a Component-Based Augmented Reality Framework. In *Proceedings of the International Symposium on Augmented Reality*, New York, NY, 2001.

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

[5] C. K. Hess and R. H. Campbell. A Context-Aware Data Management System for Ubiquitous Computing Applications. In *Proceedings of the 4th International Conference on Mobile Data Management*, Melbourne, Australia, 2003.

[6] B. Hofmann-Wellenhof, H. Lichtenegger, and J. Collins, editors. *GPS – Theory and Practice*. Springer, Vienna, New York, 4th edition, 1997.

[7] M. Kalkusch, T. Lidy, M. Knapp, G. Reitmayr, H. Kaufmann, and D. Schmalstieg. Structured Visual Markers for Indoor Pathfinding. In *The First IEEE International Augmented Reality Toolkit Workshop*, 2002.

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

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

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

[11] A. MacWilliams and T. Reicher. Decentralized Coordination of Distributed Interdependent Services. *IEEE Distributed Systems Online*, June 2003. Accepted for publication as Middleware Works in Progress Paper.

[12] J. Newman, D. Ingram, and A. Hopper. Augmented Reality in a Wide Area Sentient Environment. In *Proceedings of the International Symposium on Augmented Reality*, New York, NY, 2001.

[13] G. Reitmayr and D. 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.

[14] S. Riché and G. Brebner. Storing and Accessing User Context. In *Proceedings of the 4th International Conference on Mobile Data Management*, Melbourne, Australia, 2003.

[15] K. Satoh, M. Anabuki, H. Yamamoto, and H. Tamura. A Hybrid Registration Method for Outdoor Augmented Reality. In *Proceedings of the International Syposium on Augmented Reality*, New York, NY, 2001.

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

[17] D. Stricker and T. Kettenbach. Real-time and Markerless Vision-Based Tracking for Outdoor Augmented Reality Applications. In *Proceedings of the International Symposium on Augmented Reality*, New York, NY, 2001.

[18] M. Wagner. Building Wide-Area Applications with the AR Toolkit. In *The First IEEE International Augmented Reality Toolkit Workshop*, 2002.

[19] M. Weiser. The computer of the twenty-first century. *Scientific American*, pages 94–100, Sep. 1991.