Architecture and Tracking Algorithms for a Distributed Mobile Industrial AR System

R. Koch[1], J.-F. Evers-Senne[1], I. Schiller[1], H. Wuest[2], and D. Stricker[2]

[1] Institute for Computer Science, Christian-Albrechts-University, Kiel, Germany
[2] FHG-IGD, Darmstadt, Germany

Abstract. In Augmented Reality applications, a 3D object is registered with a camera and visual augmentations of the object are rendered into the users field of view with a head mounted display. For correct rendering, the 3D pose of the users view w.r.t. the 3D object must be registered and tracked in real-time, which is a computational intensive task. This contribution describes a distributed system that allows to track the 3D camera pose and to render images on a light-weight mobile front end user interface system. The front end system is connected by WLAN to a backend server that takes over the computational burden for real-time tracking. We describe the system architecture and the tracking algorithms of our system.

1 Introduction

Augmented Reality (AR) aims at rendering 3D object information into the users field of view. The virtual elements are merged into the real view by a half-transparent display in front of the users eyes (See-Through-Augmentation) or by mixing the images of a camera with the rendering on an opaque standard display (Video-See-Through, Video-Augmentation). In industrial service augmentation scenarios, 3D information is presented to the service technician into the field of view for hands-free operation. Compositing of virtual objects with real views provides acceptable results only if the 3D position and orientation of the users head w.r.t. the service object is tracked during operation. The tracking allows to adjust the rendering of the virtual elements such that they appear to be fixed on the service object. Tracking is demanding as the technician may move rapidly and look around, loosing sight of the object to be augmented. Time delays in rendering and non-real-time augmentations must be avoided by all means. While real-time tracking is a computationally intensive task, it is also mandatory that mobile AR systems are light-weight, easy to carry, and have low power consumption. These conflicting requirements are solved by your distributed AR system. The system is split into a light-weight mobile front end and a backend computing server which are connected by wireless LAN. The scenario of such a system is sketched in figure 1.

This contribution is focused on the design architecture and the tracking algorithms for a mobile marker-less vision-based AR system. After a short review on related work we will describe the system architecture and the vision-based tracking algorithms. We close with results on the distributed system.

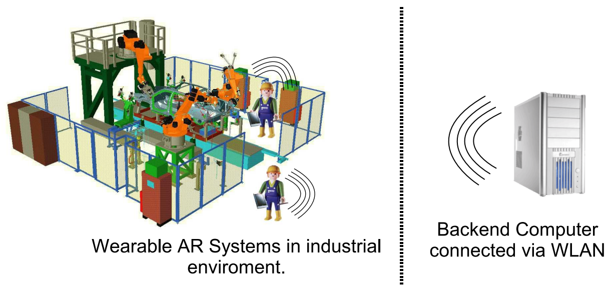

Wearable AR Systems in industrial enviroment.
Backend Computer connected via WLAN

Fig. 1. Augmented Reality scenario using a distributed tracking system.

2 Related work

Augmented Reality involves many different aspects like rendering, tracking, display hardware, computers, sensors, authoring, software architecture and user interface design. A classification of AR systems and applications is given by Azuma [1, 2]. He presented different fields of application, discussed the differences between See-Through- and Video-See-Through augmentation and analyzed several tracking approaches. He identified the "Registration Problem" as the most challenging problem in AR. Most of todays available AR systems are using fiducial markers [14, 15] or special devices to track the user and/or tools [12]. Medical AR systems to support surgery have been investigated and evolved quickly [8]. They are installed and used in well controlled small environments where elaborated marker tracking devices can be utilized.

Feiner [7] presented a mobile long range AR system to help people navigate through unknown terrain or supply them with location based information. But for our purpose this system lacks the accuracy and makes use of GPS which restricts them to out-door usage.

A marker-less vision-based approach is used in [5] where visitors of archaeological sites can use a Video-See-Trough display to view ancient buildings. The tracking in this case is reduced to 2D image registration, eventually guided by rotation data from the tilt and swivel base. In the ARVIKA project [4] an AR system for industrial service applications has been developed, but the main focus in that project was the user interface, the authoring of information and the animated visualization. The registration problem as well as object identification and tracking has been solved with fiducials. Recently, Reitmayr and Drummond[19] presented a robust system for Outdoor AR that registers buildings by edge and surface tracking, with the additional help of an inertial sensor. The system architecture is quite similar to the proposed system which was developed in the *ARTESAS* project.

In the ARTESAS project [9], one main focus is to develop vision based marker-less tracking for industrial service applications. The user wears a HMD with a small head-mounted camera. The head position of the user relative to the service object (e.g. an industrial part like a car engine) is registered by aligning the given 3D CAD model of the object with the view of the real object as seen by the head camera. From then on the head position and orientation is tracked relative to this initialization. Care is taken to handle fast head motions and to cope with tracking disruptions.

3 System Architecture

The AR-application is split into the AR-Browser which displays the output and interacts with the user, and the Tracking Framework which is responsible for pose computation. Fig. 2 shows the main components.

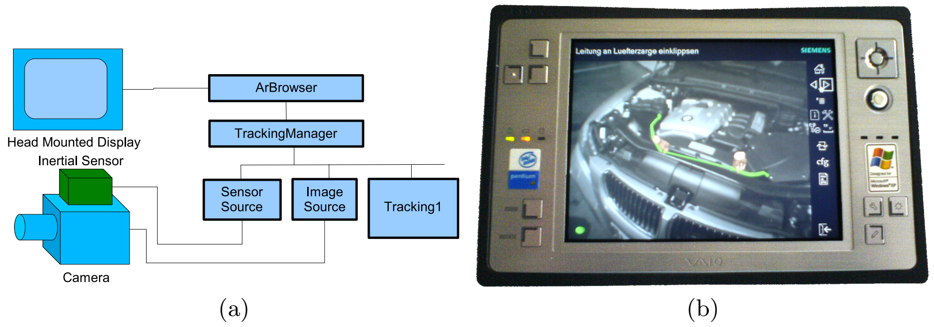
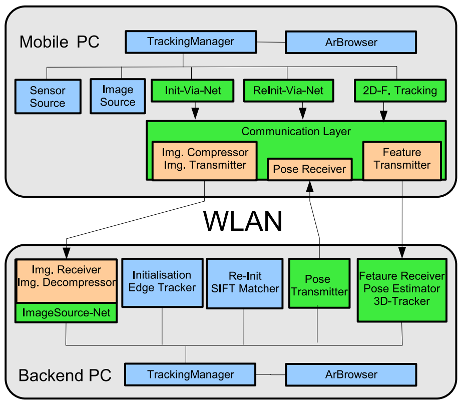


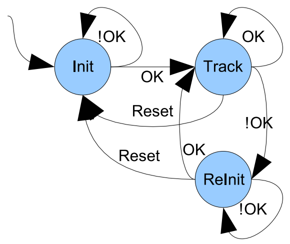
Fig. 2. (a) The basic architecture of the system. (b) Mobile subsystem Sony Vaio VGN-U70,Pentium M 1.0 GHz, Dimensions: 16.8 x 10.8 x 2.8 cm^3, Weight: 550g. It displays the tracking of a car engine. The augmentation shows where to mount a certain part.

The Tracking-Framework is designed to combine different tracking algorithms to fulfill different tasks like initialization, tracking, re-registration, latency compensation and inertial sensor fusion. Internally, the Tracking-Manager acts as a state machine with three different states: Init, ReInit, Track. Tracking always starts in the Init-state, which means that absolute position and orientation of the user with respect to the object has to be registered once. If this succeeds, the Tracking-Manager changes its state to Track, which means that the pose is tracked relative to a given start pose (the initialization). If this tracking fails (motion blur, corrupted images), the state is set to ReInit and a different algorithm has to re-register again to find an absolute pose and to switch back to tracking.

Each algorithm is implemented as a "component" with a well defined interface, such that it can be replaced by a different implementation easily. The Tracking-Manager itself consists of the state machine and several "hooks" (pointers) to attach components to. Depending on the current state and the last transition, one or more components are triggered to work. The image source is also realized as a component and can be exchanged by implementations for different cameras.



(a)



(b)

Fig. 3. (a) Modules of the Tracking-Framework in the distributed system. (b) States and transitions of the Tracking-Manager. The transitions correspond to the result of the previously executed algorithm.

3.1 Distributed System Architecture

The system may be designed as single processor system or in a distributed processing environment. In a distributed environment, a light-weight portable subsystem serves to acquire image and sensor data and to render the augmentations into the users view. Typically this subsystem should be easy to carry, with wireless access and low power consumption. Unfortunately, such a subsystem does not have sufficient processing power for real-time tracking. Therefore, a backend subsystem is connected with the portable system with sufficient processing capacity. Most important, the distributed system was designed to optimize user comfort (small size, long battery life etc) and at the same time to minimize the necessary transmission bandwidth between portable subsystem and backend server. Such a distributed system was realized as an extension to the ARTESAS system. The computationally intensive tracking part was separated into fast 2D feature matching on the mobile subsystem and robust 3D tracking estimation on the backend server. The optimization strategy how to minimize transmission bandwidth and maintain a lightweight mobile system is described in detail in [23]. The components of the distributed system and the corresponding state machine is shown in fig. 3.

4 Tracking Algorithms

The core of each tracking framework are the vision-based algorithms which are separated into the Init, Reinit and Tracking state. In the ARTESAS project three different algorithms are used in the three different states. Each is selected for its special abilities.

4.1 Initialization and Registration with 3D model

To initialize the first camera pose we use a semi-automatic model-based approach, which tries to fit a given 3D line model to edges in an image. As only a local search is performed, the user has to move either the real or the virtual camera until the 3D line model roughly overlays the corresponding object in the image. Our implementation is based on the work of [20] and [18]. More details can be found in [10].

The camera pose estimation is based on the minimization of the distance between a projected 3D line and a 2D line in the image. A very efficient technique to create correspondences between 2D image points and 3D model points is to create control points on the projected lines and to perform a one-dimensional search for gradient maxima along the orthogonal direction of the regarded line. As it is difficult to decide which of the gradient maxima really correspond to the control point on the projected line, more than one point is considered as a possible candidate. In [18] it was shown that using multiple hypotheses of gradient maxima can prevent the tracking of being perturbed by misleading strong contours. When more than one gradient maximum exists for a control point, then during the minimization always that hypothesis is used which has the closest distance to the projected control point. If p_i is the i^{th} control point and $q_{i,j}$ is the j^{th} hypothesis of the i^{th} control point, then the error to be minimized can be expressed as

$$err = \sum_i \min_j \mathit{\Delta}(p_i, q_{i,j}) \tag{1}$$

where $\mathit{\Delta}$ is the distance function between the projected control point p_i and the edge feature point q_i in the image. The distance function can be written as:

$$\mathit{\Delta}(p_i, q_i) = |(q_i - p_i) \cdot (n_i)| \tag{2}$$

where n_i indicates the normal of the projected line.

To make the pose estimation robust against outliers, an estimator function can be applied to the projection error. As recommended in [18] we use the Tukey estimator ρ_{Tuk} in our implementations as robust estimator. Together with the estimator function we can write the error as follows:

$$err = \sum_i \rho_{Tuk}(\min_j \mathit{\Delta}(p_i, q_{i,j})) \tag{3}$$

For the camera pose estimation this error function is numerically minimized by using Levenberg-Marquardt.

To decide if a tracking step was successful the line model is projected into the image with the computed camera pose. For every control point the average deviation of the direction of the projected lines to the direction of the image gradient is calculated and weighted with the absolute value of the image gradient. If the average of these values is smaller than a given threshold, the line model

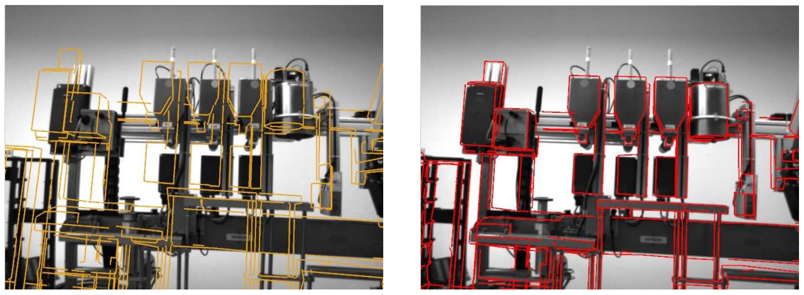

(a: Starting pose) (b: registered pose)

Fig. 4. Initialization with a line model. In (a) the initial starting position of the line model is illustrated, (b) shows the line model after the automatic registration.

is well aligned to edges in the image and the initialization of the camera pose is regarded as successful.

For initialization, we perform the line model registration on several levels of the image pyramid. Thereby the size of the convergence region gets bigger and the final registration result is more accurate. The line model which is used as input for the initialization can be automatically generated out of a polygonal model. In [11] a possible method is proposed, which extracts a line model out of a rendered scene, by analyzing discontinuities of the z-buffer image. Thereby a line model is generated which consists of 3D lines only visible from the given view. Fig. 4 shows the result of line matching for an industrial object.

4.2 Real-time object tracking

Tracking is performed by finding 2D features (intensity corners) in the camera image, by establishing 2D-3D correspondences with the underlying 3D model, and by tracking the 2D feature from frame to frame. The correspondence between the 2D image and 3D model is known because the initialization registers the 3D camera pose w.r.t. the 3D model. The 3D surface model is rendered onto the registered view and the depth buffer is extracted from the renderer. For each 2D feature, the associated depth value can then be read out to form the 3D correspondence. This *model-based tracking* allows the efficient computation of the desired camera pose. Robustness is achieved by pyramid-based tracking and a RANSAC algorithm for pose estimation that can handle features on moving objects gracefully [22].

The 2D feature tracks can be computed efficiently on the mobile subsystem and sent to the backend system which computes the pose robustly, having sufficient computational resources. In addition to model-based tracking, 2D feature tracks without known 3D correspondences can be used to establish further 3D points by triangulating novel 3D correspondences from separate view points, even outside of the given 3D model. These correspondences stabilize the tracking if correspondences on the object are lost or if the user looks away from the object.

4.3 Reinitialization

A reinitialization is needed in case that a user looks away from the object for a short time and feature tracking is lost. The ReInit state is similar to the Init state but in contrast to the Init, no user interaction is required. The basic algorithm used for re-registration is a feature matcher using SIFT descriptors [21]. This ReInit module is triggered to "learn" a pose and features during tracking. Before initialization, the SIFT matcher is not able to compute a pose. Immediately after initialization, the absolute pose and the corresponding camera image is used to generate a first set of reference keys and model-based 2D-3D correspondences. If tracking is interrupted, the ReInit algorithm is called with the next image, trying to find 2D-2D correspondences to one of the reference views, and then calculates the absolute pose from the new 2D-3D correspondences.

A new reference image and a corresponding camera pose are stored only if no reference image with similar camera pose already exists. In every reference image point features are extracted and 2D/3D-correspondences are created by projecting the 2D points onto the 3D geometry model. Correspondences of 3D coordinates and and 2D points in the camera image are obtained by a wide baseline matching of SIFT features [21] between a reference image and the current camera images. By analyzing the histograms, the reference images are ordered in similarity, and only those which resemble the current camera image are used in decreasing order of similarity for the re-initialization of the camera pose.

4.4 Latency Compensation and Inertial data fusion

For Video-See-Through augmentation, the time delay between image capture and rendering determines the frames per second which can be processed. The accuracy of the augmentation is not affected by this latency because the computed pose belongs exactly to one image. This changes if real See-Through augmentation [3] is used. Assuming, the user moves his head, his view changes over time. After finishing the computation of the pose, the virtual 3D objects are rendered for this pose, which belongs to an image from some time in the past. But due to the users movements, the rendering does not match his current view. In fact, the lag or latency is determined by three times:

$$t_{lag} = t_{image} + t_{processing} + t_{rendering}.$$

t_{image} is the image read time from capture until access in memory. This amount of time is nearly constant over time and can be measured beforehand. $t_{processing}$ is the time required to compute the pose. It can vary from frame to frame but it can be determined exactly. $t_{rendering}$ is the time for rendering and displaying. It can vary depending on the content to be rendered, it can be measured coarsely but for typical AR $t_{rendering} << t_{image} + t_{processing}$ can be assumed.

To reduce the visible latency, the following assumptions and instrumentation can help. The quickest move to change the view a human can do is to turn his/her head. This can easily exceed a rate-of-turn up to $720°/s$. Compared to these dynamics, lateral movements are slow, the change in view is small and these movements can be predicted better. Adding an inertial sensor with a high measurement frequency and a short latency allows to update the rotational component of the computed pose right before rendering. The translational components of the computed pose can be updated by a motion model. We have therefore added an inertial sensor with a read lag time of only 8 ms that allows to compensate the delay of camera pose. Following the ideas of [13], all pose updates are realized using an Extended Kalman Filter and the implementation is encapsulated as a component `Fusion` for pose update and lag compensation.

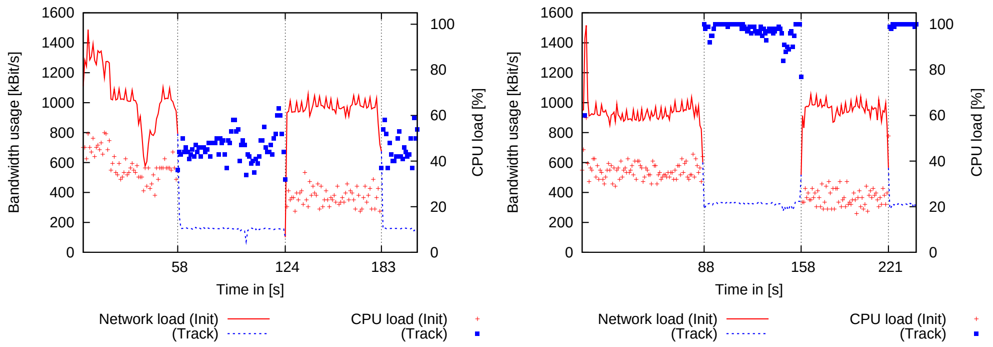
4.5 Distributed Tracking Framework

Computing the pose on a backend PC and rendering on the mobile computer means that one instance of the Tracking-Framework has to run on each system. By attaching specialized components, network connections between both instances can be established, as shown in fig 3.

The EdgeInit module should run on the backend machine, therefore it has to be replaced on the mobile system by a component which sends the current image over the network and waits for the answer to hand it back to its Tracking-Manager. On the backend side, the EdgeInit component expects to receive images from a component on the Image-Source hook. Implementing an Image-Source component which receives compressed images over the network solves this problem without modifying the EdgeInit component. Poses and result codes have to be sent from the backend to the mobile system in any state and for every frame. For this purpose, the Tracking-Manager is extended by a new pose process hook. A Pose-Sender component can than be attached to communicate tracking results to the client. The Tracking-Manager expects the tracking result directly from the components it called. Therefore these components have to receive the results themselves. Because this is the same for InitViaNet, ReInitViaNet and Tracking, a shared network communication layer is introduced.

5 Results

The system has been tested by tracking an industrial object (the motor of a car) for which the exact geometry is known by its 3D CAD surface model. Images of size 320x240 pixel, 8 bit monochrome, were used for testing. During initialization, about 10 fps are sent with JPEG compression, which reduces the bandwidth to 1 Mbit/s, with a computational load on the mobile system of only 35%. During tracking, a continuous rate of 20 to 40 fps can be achieved while reducing the requested bandwidth to 180 - 330 kbps. Therefore, in a single WLAN channel, many AR systems can run concurrently, allowing many service technicians to work in the same environment. In addition, the CPU load leaves room for additional tasks like speech analysis to control the interface hands-free. Fig. 5 shows the performance of the system for network and CPU load.



(10 fps Init, 20 fps Track) (10 fps Init, 35 fps Track)

Fig. 5. Network load (incl. TCP overhead) and CPU load of the mobile system.

6 Conclusions and Future Work

We have presented a distributed marker-less AR system that allows to register and track 3D objects in real-time and to simultaneously render augmentations into the users view with high quality. The wireless wearable front end system is lightweight and allows high mobility while the connected backend system allows high quality tracking. The tracking is distributed with rendering, 2D image compression and 2D feature tracking on the front end, and 3D processing on the backend system. The performance evaluation shows that the front end system may even run with lower performance to extend battery lifetime.

Acknowledgments

Parts of this work has been developed in the ARTESAS project, funded by the German government (BMBF). The development of the distributed extension to the ARTESAS system was funded by Siemens A&D, Nürnberg, Germany.

References

1. Azuma, R. *A Survey of Augmented Reality* Presence: Teleoperators and Virtual Environments, Vol. 6, pp. 355-385, 1997.
2. Azuma, R, Baillot, Y, Behringer, R and Feiner, S; Julier, S; MacIntyre, B, *Recent advances in augmented reality* IEEE Computer Graphics and Applications. Vol. 21, no. 6, pp. 34-47. 2001

3. Azuma, R. *Tracking Requirements for Augmented Reality*, http://www.cs.unc.edu/˜ azuma/cacm.html
4. Friedrich, W. Wohlgemuth, W., *ARVIKA Augmented Reality in Entwicklung, Produktion und Service*, ISMAR '03, Tokyo, Japan.
5. Vlahakis, V., Ioannidis, N., Karigiannis, J. *ARCHEOGUIDE: Challenges and Solutions of a Personalised Augmented Reality Guide for Archeological Sites*, IEEE Computer Graphics and Applications magazine, 22, Sept-Oct. 2002, pp. 52-60.
6. Thomas, B., *Challenges of Making Outdoor Augmented Reality Games Playable*, In Proc. 2nd CREST Workshop, May 23 - 24, 2003, Nara, Japan
7. Feiner, S., MacIntyre, B., Hllerer, T., and Webster, T. *A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment.* In Proc. ISWC '97, October 13-14, 1997, Cambridge, MA.
8. Heining, S. M., Euler, E., Sauer, F., Schneberger, M., Zrl, K., Mutschler, W.: *Augmented Reality and in-situ Visualization in Thoracoscopic Spine-Surgery.* In Proc. of CAOS-International, Chicago, Illinois, USA, 2004, 355-357.
9. http://www.artesas.de, http://www.ar-tracking.de/
10. Wuest, H., Vial, F., Stricker, D.: *Adaptive Line Tracking with Multiple Hypotheses for Augmented Reality*, In Proceedings ISMAR 2005, Vienna, Austria.
11. Wuest, H. and Stricker, D., *Tracking of industrial objects by using CAD models*, Virtuelle und Erweiterte Realitaet: 3. WS GI-Fachgruppe VR/AR,pp 155-163, 2006.
12. Foxlin, E., Harrington, M. and Pfeifer, G., *Constellation: a wide-range wireless motion-tracking system for augmented reality and virtual set applications*, In Proc. SIGGRAPH '98 ACM Press, 371-378
13. Foxlin, E., Naimark, L., *VIS-Tracker: A Wearable Vision-Inertial Self-Tracker*In Proc. Virtual Reality 2003 , March 22-26, 2003, Los Angeles, CA
14. Kato, H., Billinghurst, M., and Poupyrev, I. *ARToolKit version 2.33 Manual, 2000.* Available for download at http://www.hitl.washington.edu/ research/shared_space/download/.
15. Wagner, D., Pintaric, T., Ledermann, F., Schmalstieg, D., *Towards Massively Multi-User Augmented Reality on Handheld Devices*, Lecture Notes in Computer Science, Volume 3468, Jan 2005, Pages 208 - 219
16. Pasman, W., Woodward, C.,*Implementation of an Augmented Reality System on a PDA*, In Proc. ISMAR '03, Oct. 7-10, Tokyo, Japan
17. Beier, D., Billert, R., Brderlin, B., Stichling, D., Kleinjohann, B., *Marker-less Vision Based Tracking for Mobile Augmented Reality*, In Proc. ISMAR '03, Oct. 7-10, Tokyo, Japan
18. Vacchetti, L., Lepetit, V., Fua, P., *Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking*, Proceedings of International Symposium on Mixed and Augmented Reality, ISMAR 2004, Arlington, USA.
19. Reitmayr,G. and Drummond, T., *Going out: Robust Model-based Tracking for Outdoor Augmented Reality*, Proceedings ISMAR, San Diego, USA, 2006.
20. Drummond,T. and Cipolla, R., *Real-Time Visual Tracking of Complex Structures*, IEEE Trans. on Pattern Analysis and Machine Intelligence, 24(7), 2002.
21. Lowe, D., *Distinctive Image Features from Scale-Invariant Keypoints.* International Journal of CV, 60(2), 2004.
22. Streckel, B., Evers-Senne, J.-F., Koch, R, *Lens Model Selection for a Markerless AR Tracking System.* ISMAR 2005, Vienna, Austria, 2005.
23. Jan-Friso Evers-Senne, Arne Petersen, and Reinhard Koch, *A Mobile Augmented Reality System with Distributed Tracking* Proc. 3DPVT, June 2006, Chapel Hill, NC, USA.