Structured Visual Markers for Indoor Pathfinding

Michael Kalkusch, Thomas Lidy, Michael Knapp, Gerhard Reitmayr, Hannes Kaufmann, Dieter Schmalstieg
Vienna University of Technology, Vienna, Austria
{kalkusch|lidy|knapp}@cg.tuwien.ac.at, {reitmayr|kaufmann|schmalstieg}@ims.tuwien.ac.at

Abstract

We present a mobile augmented reality (AR) system to guide a user through an unfamiliar building to a destination room. The system presents a world-registered wire frame model of the building labeled with directional information in a see-through heads-up display, and a three-dimensional world-in-miniature (WIM) map on a wrist-worn pad that also acts as an input device. Tracking is done using a combination of wall-mounted ARToolkit markers observed by a head-mounted camera, and an inertial tracker. To allow coverage of arbitrarily large areas with a limited set of markers, a structured marker re-use scheme based on graph coloring has been developed.

1. Motivation

Many visitors of our institute have a hard time finding their way through the complicated and poorly labeled corridors of our office building. We considered this situation a useful test case for a location-based service provided through our mobile augmented reality system. The system should guide a user on the way through the building by showing him directions and a floor map, based on context information (see Figure 1).

A constraint in the design of our solution was that, unlike previous AR outdoor navigation aids, GPS is not usable indoors, and we also did not have access to a proprietary building-wide positioning infrastructure (such as AT&T Cambridge's BAT system [1]). Instead, we choose to rely on a hybrid solution consisting of ARToolkit [2] for optical tracking and inertial tracking combined with room geometry information. This approach proved to be very flexible in terms of development and of positioning the infrastructure, but also pushes the limits of what ARToolkit tracking can provide.

2. Related Work

Location tracking is a prerequisite for any location aware application. In order to be able to provide the user with services and information related to her location, a system needs to sense or otherwise be told the current location of the user. Augmented Reality (AR) applications require a very accurate position tracking to register visual information accurately with the user's environment. Otherwise the augmented information may be positioned incorrectly resulting in a confusing or unusable user interface.

There is a wealth of work related to position tracking ranging from indoor systems covering a room size area to outdoor systems supporting the entire planet. This diversity is also present in the accuracy of tracking systems ranging from millimeters to several meters. Outdoors, GPS provides global position with accuracy between several meters and centimeters depending on additional techniques such as broadcasting correction signals or using the phase information of the received signals to improve the accuracy. However, GPS requires a direct line of sight to several satellites and therefore is not working properly inside buildings or in areas covered by trees or tall buildings (appropriately termed 'city canyons').

Indoors, tethered tracking systems using magnetic [3], ultrasonic [4] and optical technologies [5] achieve high accuracy in the millimeter to centimeter range. These systems are typically able to cover a room and require installations of large devices or dense arrays of beacons or sensors mounted in the covered area. Another research

Figure 1. The user is guided through the building by the *SignPost* navigation system.

system [1] can cover a whole building but is not available to the public.
Other approaches try to use local sensors and dead reckoning techniques to compute a user's location [6]. However these are prone to accumulation of subsequent errors in their computations unless they are synchronized with absolute positioning systems. Including knowledge about the environment in the form of geometric models and accessibility graphs [7] allows increasing the accuracy of such approaches significantly.
A large class of tracking solutions uses computer vision to track the movement of a camera. Some solutions place fiducial markers [8] in the environment to achieve good accuracy. There is also experimental work that uses marker free vision based tracking by selecting salient natural features in the environment [9] or comparing the camera's view with prerecorded imagery [10].
Our solution is relying on using a set of trained optical markers together with additional knowledge about the environment. This allowed us to improve the performance of the optical tracking component by reducing the number of markers required for a stable operation of the system.

3. Mobile Setup

Our wearable AR navigation system [14] is composed of a notebook computer with a 1 GHz processor and an NVidia GeForce2Go chip for accelerated 3D graphics. A Glasstron optical stereo see-through head mounted display is used to overlay information while providing an unobstructed view of the environment when roaming the building. In addition to that, a FireWire camera and InterSense InterTrax2 inertial tracker for tracking input are mounted to the helmet worn by the user. A wrist-mounted tracked augmented touch pad is deployed as the user interface for application control and monitoring.
The system runs the latest version of the *Studierstube* [12] software framework. *Studierstube* is a user interface management system for collaborative augmented reality, which addresses the question of how to use three-dimensional interaction and new media in a general work environment, where a variety of tasks are carried out simultaneously. It supports a variety of output devices such as head mounted displays, projection setups and virtual tables. It allows multi application, multi user setups running on a distributed system. Several applications have already been developed within *Studierstube*, all of them establishing a 3D interaction metaphor similar powerful as the desktop metaphor for 2D.

4. Tracking

In our work we used the optical tracking available from a single camera mounted on helmet worn by the user and prior knowledge on the structure of the building in form of a geometric model. The camera provides input for the ARToolkit to detect and track well-known markers that were distributed in the environment. By combining the relative marker locations computed by the ARToolkit and the model of the building we can than track the user's movements within the environment. In addition to that we deployed the inertial tracker to provide frequent orientation updates whenever the optical tracking failed.
The geometric model of the building is structured into individual rooms and the connections between these rooms called portals. A portal can either be a real door or an artificial separation of larger rooms into small areas in the final model. The location of a room within a world coordinate system completes the model. It also stores additional information such as room names and occupation.
In each room optical markers were mounted on the walls. Their location was measured and added to the model (see Figure 2). We tried to place enough markers such that at any moment a user would perceive at least one marker. The marker's size determines the appropriate distance between the user and the marker. We chose a size of 20 by 20 centimeters to allow an average distance of about 2.5 meters between the user and the walls. Such a size worked also well if the marker was only seen at grazing angles.

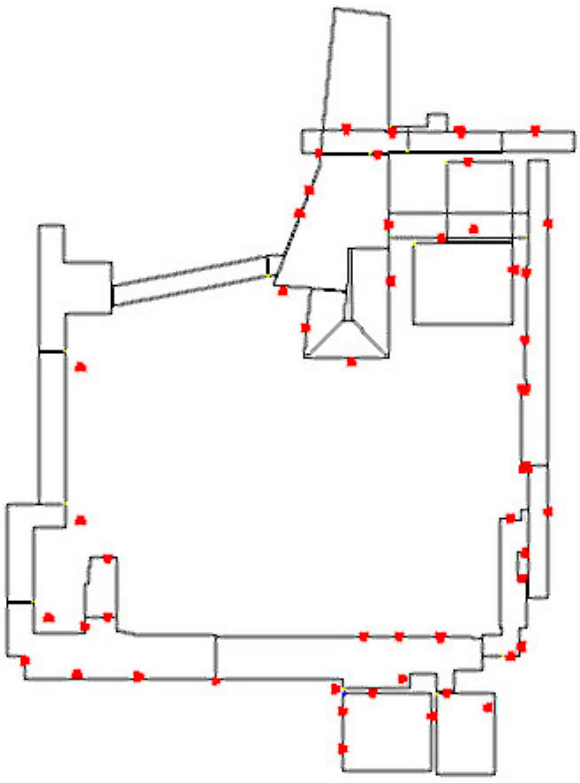

Figure 2. This is the geometric model of the 4th and 5th floor of the building. The marker locations are indicated by the dots. The whole setup includes 25 rooms on two floors with 25 unique markers and 40 reusable markers.

4.1. Computing the user's location within a room

For a user standing in a room, the system can track one or more optical markers. The information received by the tracking library gives the markers location with respect to the camera. This information needs to be transformed to get the local position of the user within the room (see Figure 3). To do so we need to perform the following calculations.

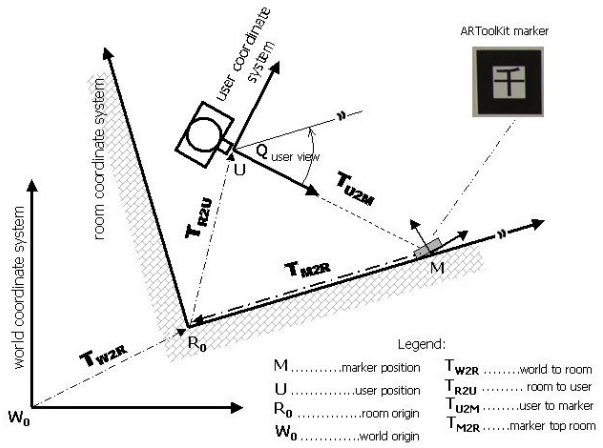

Figure 3. Four coordinate systems are involved to compute the user's position. *U* is the user's position and orientation, R_0 the room coordinate system. *M* is a marker's location within that room. In order to handle several rooms in several floors a world or model coordinate system W_0 is used.

Combining user with marker coordinate system

At each time step the optical tracking returns the position and orientation of a marker as a transform matrix T_{U2M}. This describes the transformation between the local user coordinate system and the marker's coordinate system. We need the user's movement with respect to the fixed marker, therefore we compute the inverse matrix T_{M2U} of the above transformation:

$$T_{M2U} = (T_{U2M})^{-1}$$

Computing user location within a room

For rendering the wire frame model and providing the user with directional hints we need to compute the user's location in respect to the room's coordinate system. The desired matrix T_{U2R} is calculated with the following equation:

$$T_{R2U} = (T_{U2M})^{-1} \circ T_{M2R}$$

The transformation T_{M2R} is provided by the geometry file and contains the marker position and orientation in respect to the room origin R_0. Combining the current user's field of vision and the knowledge from the navigation system, which door or portal has to be passed next, we can supply the user with directional hints (see Figure 8).

Transforming to world coordinate system

To be able to provide a consistent WIM we also have to take the global rooms position into account with respect to the world coordinate system W_0. By doing so we can easily render the whole floor directly from the geometry file to the WIM.

One important detail during rendering is, that rotation of the room should always be done at the current marker position. This improves rendering because the angular precision of the optical tracking sometimes suffers from jitter.

The information on the user's position and viewing direction is then used to compute the direction of the indication arrow and to render the augmentation of portal's and the room's geometry.

4.2. Reuse of markers

To use the approach described in the former section, we need to place a marker about every two meters and to cover each wall of a single room with at least one marker. Deploying markers on our floor, which covers about 25 rooms and long hallways, would require several hundred different markers to be created and trained for the ARToolkit recognition process. However, this is not feasible for two reasons:

- The more markers, the higher the degree of similarity of markers will be. Almost rotational symmetry can become a major problem when a large amount of markers is used. Additionally, lighting conditions may vary often between one room and another, or even within the same room. All this leads to inferior recognition accuracy. For a larger set of markers this implies a higher number of false recognitions.
- A large set of markers increases the search space that has to be compared by ARToolkit, which leads to a decrease in performance. As a consequence, it is not possible to scale the use of ARToolkit to arbitrary large marker assemblies.

To overcome this restriction, we developed a spatial marker partitioning-scheme that allows reusing sets of markers within the given environment. The idea behind this approach is that, if the tracking system knows the user's location, it can rule out large parts of the building because they will not be visible to the camera. Therefore, for two areas, which are not visible to each other, it becomes possible to use the same set of markers. This problem is equivalent to approaches for indoor visibility computation based on potentially visible sets.

We use the room definition in the geometric model as the basic element of this approach. Then we can build an adjacent graph for all rooms using rooms as nodes and portals between rooms as edges. In addition to the portals

representing logical connections we also add further edges that represent a line-of-sight between to rooms (see Figure 4). We call this second graph the extended adjacent graph.

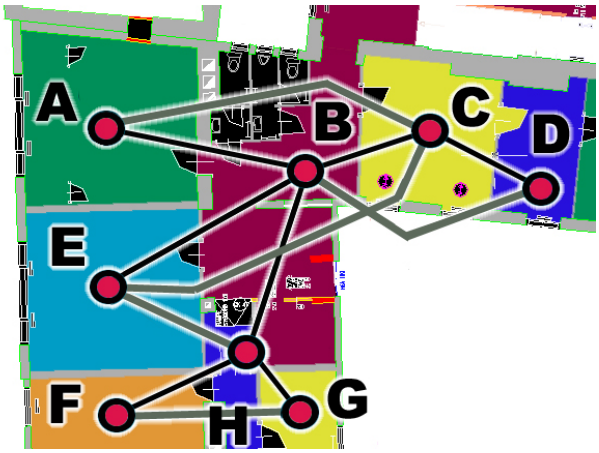

Figure 4. The adjacent graph and the extended adjacent graph of a part of the building. Different rooms are indicated by different floor colors. Each room is represented as a node. Black edges indicate a door between two rooms, gray lines indicate a line of sight between two rooms. In both cases disjunctive marker sets must be used.

The next step is to generate a minimal number of disjoint sets of markers, such that each room is assigned to a set of markers and two criteria described next are met. This is similar to graph coloring problems where a minimal set of colors is used to color nodes in a graph with certain restrictions. The following constraint must be fulfilled to yield a useful distribution of marker sets.

- Two nodes connected in the extended adjacent graph must have disjoint sets of markers. Otherwise the camera tracking system cannot decide which room the marker belongs to, if both instances are visible from one room.
- Looking at one node all adjacent nodes in the extended adjacent graph must have disjoint marker sets. This constraint has to be fulfilled by all nodes in the extended adjacent graph. Otherwise the common node provides a point of view that allows sighting of a marker in two different places.

Starting with a given correct room position, the system tracks the user within the current room and into neighboring rooms. It can detect a room change by comparing tracked markers with the sets of markers assigned to the current room and its neighbors. If it recognizes a marker from a set of a neighbor room it assumes that the user entered this room, which then becomes the current room. These reusable marker sets are managed by the PartitionKit, which is described in section 5.2.
The major task of the PartitionKit is to toggle the nodes status between active and inactive whenever a room change happens (see Figure 5).

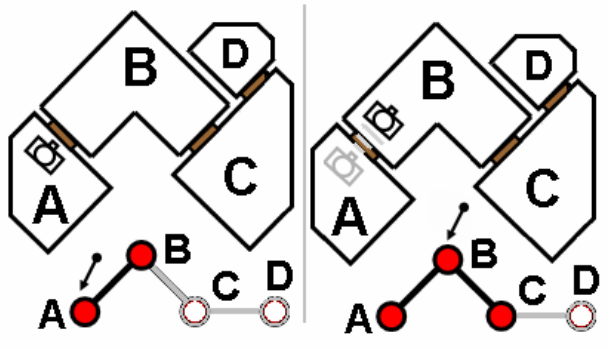

Figure 5. User walks from room A (shown on the left) to room B (shown on the right). The corresponding adjacent graph is show, below each floor map. Current node marked with an arrow. Inactive nodes are drawn in gray.

The left image shows that the current node A and its adjacent node B are marked active, when the user is in room A, while node C and D are inactive (gray). As soon as the user moves to room B and a room change occurs, the PartitionKit updates the whole adjacent graph (shown on the right side in Figure 5). Now the current room is room B and the status of both adjacent rooms A and C are switched to active. Note that room D is still inactive, because there is no line of sight between room B and room D.
We extended the scheme by placing a single globally unique marker in every room, which can be used to re-initialize the system in case of an error. It is also suitable for the determination of the current room at start up. The user just has to turn around until the system detects the unique marker and switches into “interactive mode”. This unique marker set is disjoint from any reusable marker set.

4.3. Fusion

Our mobile Augmented Reality kit is also equipped with an inertial orientation tracker providing low latency updates on the user's head orientation. The frequency of updates is about 100 Hz, which exceeds the update rate of the optical tracking by an order of magnitude. Hence we incorporate this information into the computation of the user's viewing direction in-between measurements from the optical tracking system. However the tracking information is subject to drift that can lead to large errors after a short period of time.

Therefore we need to correct the measurements of the inertial tracker (see Figure 6). This is implemented similar to the description given by Newman et al. [13].

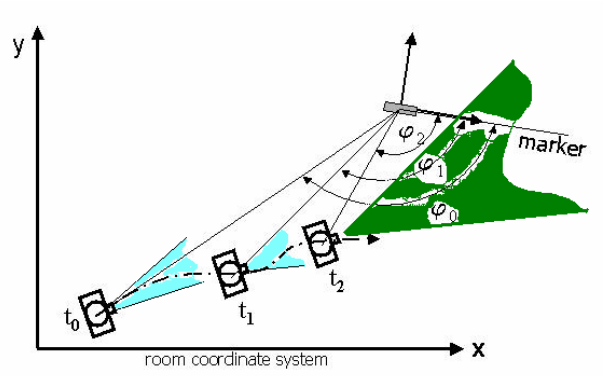

Figure 6. User walks towards marker shown in three different time snap-shots (t_0, t_1,t2) with corresponding angles. In time step t2, no marker is tracked and the user's viewing direction is updated from the inertial tracker.

Every time we receive a measurement by the optical tracking system, we compute the user's true head orientation Q_{UserView} (see Figure 3) as described in section 4.1. As long as at least one marker is in the user's field of view, like in Figure 6 time steps t_0 and t_1, the $Q_{\text{CurrentARview}}$ is equal to the Q_{UserView}.

$$Q_{\text{CurrentARview}} := Q_{\text{Marker}} = Q_{\text{correction}} \text{ o } Q_{\text{Inertial}}$$

Then we compute a correction orientation for the measured inertial orientation:

$$Q_{\text{correction}} = Q_{\text{Marker}} \text{ o } (Q_{\text{Inertial}})^{-1}$$

When no marker is visible (see Figure 6 at time step t_2), we have to rely on the inertial tracking information, which is pure angular information. In order to improve the tracking we use the correction term calculated, while optical tracking worked. The correction is then applied to any subsequent measurement of the inertial tracker to provide a correct user orientation:

$$Q_{\text{CurrentARview}} := Q_{\text{correction}} \text{ o } Q_{\text{Inertial}}$$

As soon as the optical tracking starts to send new marker data, the correction terms are recalculated and the user position is updated.

5. Example Application

As an example application for the described tracking system we developed *SignPost* - a system, which guides the user through a building from his current position to a selected destination room by providing the user with appropriate information through augmented reality.

SignPost is built on the Studierstube framework and implements the described tracking solution to get information on the current position of the user within the building. The OpenTracker [15] library is used to integrate the ARToolkit tracking library, to transform the data gained from it (and other trackers) and to forward it over a defined interface to the Studierstube. For graphical output via the head-mounted display OpenInventor is used, which is part of the Studierstube framework.

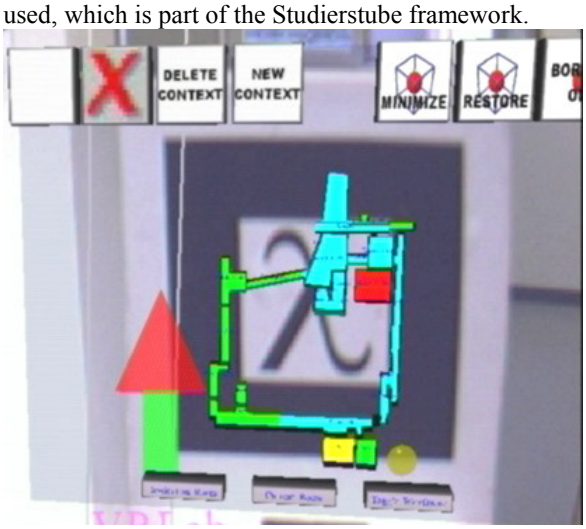

Figure 7. Word-in-Miniature (WIM) model displayed on the augmented wrist pad. The current room is highlighted in yellow, the destination room in red and the path in between in cyan. Three buttons below the WIM allow the user to set a destination room, toggle the wire frame and set the current room.

5.1. Interaction

The system continuously provides the user with two kinds of visual feedback:

- Directional hints: Via the HMD a wire frame model of the current room is superimposed on top of the real scene. The application uses a shortest path search on an adjacency graph of the building to determine the next door/portal on the way to the destination. The doors/portals are always highlighted in white. In addition, a direction arrow shows the direction to the next door or portal (see Figure 8), indicating either to move forward, turn left/right or to make a U-turn. The wire frame overlay can be turned on and off optionally by the user.
- WIM model on wrist pad: The WIM ("world in miniature") model always shows a full miniature view of all rooms on the floor in order to allow the user to determine his current position in the building, which is always highlighted in yellow (see Figure 7). Additionally the path to the selected destination room is highlighted in cyan, while the destination room itself is shown in red. An ARToolkit marker is placed onto the wrist pad allowing the application to track the position of the wrist pad. Thus, the user can bring the WIM model into his view by moving his arm with the wrist pad in front of his eyes. As a consequence, the WIM model does not

permanently cover the screen, and the user can decide when he wants the floor plan to be displayed. Moreover, he can adjust the size by moving his arm nearer to his eyes or farther away. The wrist pad enables the user to select a destination room as well as to toggle the wire frame mode mentioned above.

The *SignPost* application uses a scene graph based on a floor plan loaded from an OpenInventor file. In the application actually two scene graphs are used. The first scene graph is used for the WIM model on the wrist pad. One whole floor is shown at a time.

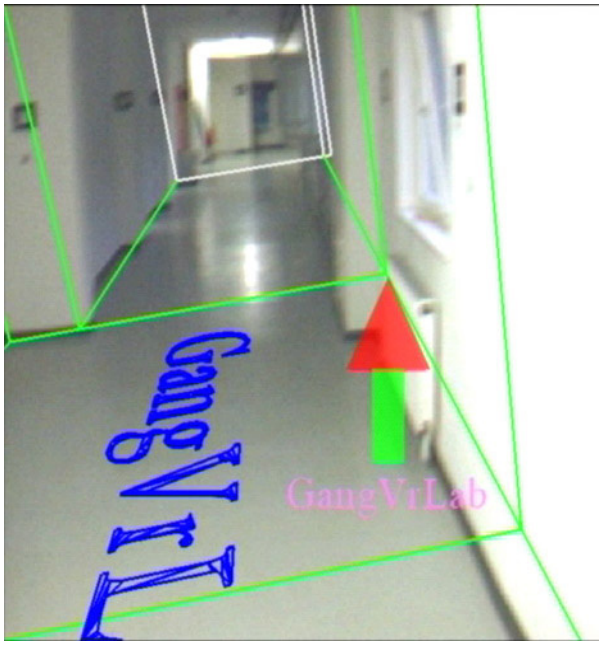

Figure 8. Information overlay in the head mounted display. Red arrow shows the direction. The name of the current room is overlaid below the arrow. The blue writing is part of the room geometry and coupled with the green wire frame model of the room.

The second scene graph is used to display the wire frame model of the current room in the head-mounted display. Only the current room is shown and all other rooms are disabled and not rendered. The overlaid wire frame geometry is aligned with the real room geometry and continuously updated, using data from optical and inertial tracking.

Once the marker of the wrist pad gets into the user's field of vision the WIM is displayed and the user may interact via the wrist pad.

5.2. Partition Kit

We extended the OpenInventor scene graph with several nodes, which are used by our application and can also be used by other applications. Two of these nodes are the PartitionKit and the StationKit. Each room in the OpenInventor scene graph is attached to a StationKit, which contains information on the room and the references to the marker set of that room. The PartitionKit manages the disjunctive sets of reusable markers. Based on the extended adjacent graph described in section 4.2 we improve the performance of the marker detection, because only markers in the current room and the adjacent rooms are used for detection. The major contribution of the PartitionKit is, that it forwards tracking events only to the nodes of the current room and its adjacent rooms. For this part of the application we only need the adjacent graph, but we do not need the extended adjacent graph (see Figure 4). Without the PartitionKit all existing markers would be compared with the current one, which could lead to a spontaneous room change to a different floor. This improves the fault tolerance of the whole application.

5.3. PathFinder Algorithm

Assuming you have a number of rooms connected by doors, it is common to handle them by means of a directed adjacent graph. Each room is represented as a node. An edge is inserted between two nodes, if two rooms are connected via a door or a portal. Using a directed graph, it is possible to map one-way doors, like emergency exits, in the adjacent graph.

The SignPost application uses its so-called PathFinder-component for searching the shortest path from the user's current position to the desired destination. After loading the floor-plan file, the PathFinder builds an adjacent graph based on the coherency information provided in the file. The PathFinder determines the coordinates of the room center and the portals. Then for each room it calculates the distances from the room center to all the portals of the and further from the portals to the room centers of the subsequent rooms.

Using this adjacent graph, the PathFinder is able to indicate whether there is a coherent path from a certain room to another one, and how long the way is. It is most important for our application that the PathFinder returns a list of the rooms and portals, which have to be passed along the path. This is done by utilizing an algorithm based on the shortest path algorithm, which was introduced by Dijkstra [11].

Although the PathFinder could recalculate and update the shortest path several times per second, in the *SignPost* application a new path is only calculated when a room-change occurs. A room-change happens, whenever ARToolkit recognizes a marker of an adjacent room. The PathFinder always adapts the shortest path starting from the current room. So even if the user walks the wrong way, a new shortest path based on the user's current position is proposed.

6. Results

The current model used for development and testing purposes covers parts of two adjacent floors in our office building. Data acquisition was very work intensive since available digital maps proved insufficiently accurate for AR applications. Initial experiences show that the scheme works quite well, but is heavily relying on sufficient marker density. Acceptable results require a marker every 2-3m in a hallway and at least 4 markers per room. As can be expected, the more markers are used the less jitter of the registered wire frame is perceived. The gaps between the markers are avoided by using the inertial tracker. This means that the user can continue on his way without viewing a marker for some meters. Unfortunately the inertial tracker sometimes starts to drift in one direction leading the user to bear away from his path. Heading to a wall and viewing a marker from a close position adjusts the path shown on the users' view to the correct path again.

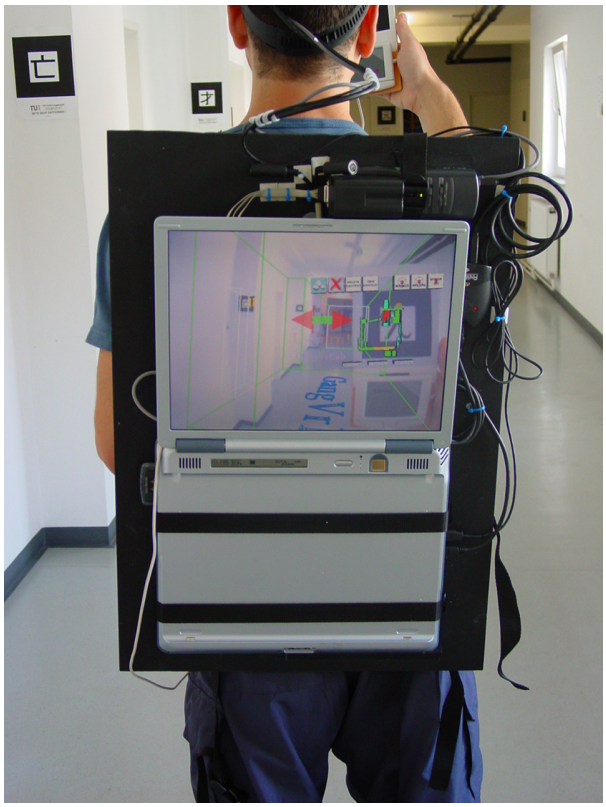

Figure 9. Mobile Setup running the *SignPost* application. Augmentation of the wire frame model and directional hints guide the user to the desired location. User holds PIP in right hand and the WIM is overlaid on top of it.

We found, that the degree of correct marker recognition relies heavily on the light conditions in the rooms. ARToolkit provides a very useful feature to address this problem: One can assign a threshold to marker recognition, which influences the lightness at which a marker is recognized properly. However, in our application we are confronted with varying light conditions from one room to another or even in a room. Some rooms are filled with sunlight from several windows, other rooms are illuminated from neon lamps and some hallways are very gloomy, especially when a few lamps are out of order.

Consequently our application makes high demands on the marker recognition pushing ARToolkit to its limits. We propose an idea to address this problem in section 7.

Sometimes, markers still are confused by ARToolkit and erroneously recognized as other markers. A solution to this problem is provided by our marker-structuring scheme more or less as a side-effect: Whenever ARToolkit cannot clearly distinguish a marker from one or more other markers, it happens that multiple marker IDs are proposed by the system. Due to marker-structuring (and the PartitionKit), most of the marker IDs can be dropped – because it is sure that they are not in the line of sight – leaving (in the majority of cases) one marker which can be clearly identified. Thus, the PartitionKit also helps with uniquely identifying markers, even when a larger amount of markers is used.

7. Conclusion and Future Work

We described an autonomous Augmented Reality indoor navigation system (see Figure 9), which makes use of visual ARToolkit markers, thus keeping the costs of the system rather inexpensive. The system incorporates an elaborate marker-structuring scheme, which enables the re-use of ARToolkit markers in rooms that are not in line of sight. The algorithm is based on adjacent graphs, which are also used for finding the shortest path to a desired destination. The big advantage of our approach is, that the total number of markers incorporated in the recognition process is reduced to a minimum. The re-using marker set concept in combination with the extended adjacent graph (see section 4.2) grants a very small total amount of markers. Thus, it is possible to provide a whole building with only a few marker sets.

We plan to extend our digitized map to all floors of our building as more comprehensive location based services can be integrated into the guide with a more complete model. For this amount of digitalization work, professional grade surveying equipment will be necessary, but we think the result will justify the effort. The setup is suitable for outdoor scenes as well, and we intend to try this soon.

As far as the lighting condition problem mentioned in section 6 is concerned we plan to implement adaptable thresholds, which allow assigning each room its own threshold for marker recognition according to its prior-known lighting.

Building the floor model is currently no comfortable task, as one has to edit the coordinates in a text file. Right now,

we make this task more convenient by providing a Map Viewer tool, which shows the floor plan with all its marker positions, reference points, etc. Future extensions to make map editing more comfortable range from working with XML files to writing a complete graphical interaction-based Map Editor.
The quality of the tracking system could be improved by using Kalman filters [16] to predict the user's motion. Based on the prediction value it should be possible to discard erroneous measurements.
The tracking system itself should be made reusable as an optional standalone component in the *Studierstube* framework. Based on this we will be able to focus on the application behavior and design for future location based applications.

Acknowledgements

This work was sponsored by the Austrian Science Fund (FWF) under contracts no. P14470-INF and START Y193, and Vienna University of Technology by an infrastructure lab g rant ("MARDIS"). Special thanks to Oliver Mattausch for doing all the fantastic Japanese markers and translating them to English and implementing the XML to OpenInventor converter together with Tamer Fahmy, who also initially implemented the Pathfinding component.

Videos and pictures are available at
http://www.studierstube.org/mobile/projects/SignPost/.
For more information on the Studierstube framework see
http://www.studierstube.org/

References

[1] Addlesee, M., Curwen, R., Hodges, S., Hopper, A., Newman, J., Steggles, P. & Ward, A. (2001),'A sentient computing system', IEEE Computer: Location-Aware Computing .
[2] Billinghurst, M. & Kato, H. (1999), Collaborative mixed reality, in 'Proc. ISMR'99', Springer Verlag, Yokohama, Japan, pp. 261–284.
[3] F. Raab, E. Blood, T. Steiner and R. Jones. Magnetic position and orientation tracking system. *IEEE Trans. On Aerospace and Electronic Systems*, AES-15(5):709—718, September 1979
[4] InterSense. InterSense IS-900 Wide Area Precision Motion Tracker. *http://www.isense.com/*. 2002.
[5] G. Welch, G. Bishop, L. Vicci, S. Brumback, and D. Colucci. High performance wide-area optical tracking - the hiball tracking system. *Presence: Teleoperators and Virtual Environments 10:1*. 2001.
[6] Seon-Woo Lee and Kenji Mase. Incremental Motion-Based Location Recognition. *Proc. ISWC'01*. Zurich, Switzerland, October 2001.
[7] Tobias Höllerer Drexel Hallaway, Navdeep Tinna, Steven Feiner, „Steps Toward Accommodating Variable Position Tracking Accuracy in a Mobile Augmented Reality System", Columbia University , AIMS 2001
[8] Jun Rekimoto. Matrix: A Realtime Object Identification and Registration Method for Augmented Reality. *Proc. APCHI'98*. 1998
[9] Axel Pinz. Consistent Visual Information Processing Applied to Object Recognition, Landmark Definition, and Real-Time Tracking. *VMV'01*, Stuttgart, Germany, 2001.
[10] M. Kourogi, T. Kurata and K. Sakaue. A Panorama-based Method of Personal Positioning and Orientation and Its Real-time Applications for Wearable Computers. *Proc. ISWC'01*. Zurich, Switzerland, October 2001.
[11] Edsger W. Dijkstra, A note on two problems in connection with graphs. *Numerische Mathematik*. 1:269—271, 1959
[12] Schmalstieg, D., Fuhrmann, A., Hesina, G., Szalavari, Z., Encarnao, L. M., Gervautz, M. & Purgathofer, W. (2002), 'The Studierstube augmented reality project', PRESENCE - Teleoperators and Virtual Environments 11(1).
[13] Newman, J., Ingram, D. & Hopper, A. (2001), Augmented reality in a wide area sentient environment, in 'Proc. ISAR 2001', IEEE, New York, New York, USA.
[14] Reitmayr, G. & Schmalstieg, D. (2001a), Mobile collaborative augmented reality, in 'Proc. ISAR 2001', IEEE, New York, New York, USA, pp. 114–123.
[15] Reitmayr, G. & Schmalstieg, D. (2001b), An open software architecture for virtual reality interaction, in 'Proc. VRST 2001', ACM, Banff, Alberta, Canada, pp. 47–54.
[16] Rudolph E.Kalman (1960), "A New Approach to Linear Filtering and Prediction Problems", Baltimore, Md., in 'ASME–Journal of Basic Engineering', 82 (Series D): 35-45.