Dynamically Shared Optical Tracking

Florian Ledermann, Gerhard Reitmayr and Dieter Schmalstieg
Vienna University of Technology
{ledermann | reitmayr | schmalstieg}@ims.tuwien.ac.at

Abstract

Optical Tracking using ARToolkit provides us with the base technology for a wealth of Augmented Reality applications. However, marker-based optical tracking using a single camera has some drawbacks. Markers must be fully visible to the camera all the time to produce tracking output – occlusion by other objects and limited camera field of view constrain the area that can be effectively tracked by ARToolkit. In our approach to improve tracking availability, tracking data from multiple hosts is shared across the network. Pairwise camera-to-camera relationships are established automatically, as soon as any marker is seen by two cameras, independent of the cameras' placement (e.g. cameras worn by a user or mounted at a "hot spot" location to improve tracking in that area). The setup is completely dynamic: both cameras can be continuously moving, and there is no "special marker" that must be seen by both cameras – as soon as any one marker in the system is visible to both cameras, all missing tracking information can be calculated from the data sent over the network. In this paper, we describe such a configuration for multiple hosts in detail, as well as special aspects such as automatically selecting the best marker to use as a reference point, a description of the system's data flow, scalability and accuracy issues and future work such as automatic configuration of the system.

1. Introduction

ARToolkit [1] is enabling its users to add three-dimensional tracking capabilities to their applications. ARToolkit uses square markers as shown in Fig. 1, carrying a unique pattern to distinguish markers from each other. These markers are observed by a single camera, and the tracking software uses computer vision techniques to calculate the markers position and orientation from the captured image.

This kind of optical tracking provides us with the base technology for a wealth of augmented reality applications, without the need for buying expensive or complicated tracking hardware. Markers can be used as a tangible interface to handle virtual artifacts or as user interface elements.

We are using ARToolkit to build a multi-user mobile augmented reality system [4], providing tracking for individual users by a head mounted camera. Although we are using a wide-angle camera to perform the tracking, it is sometimes impeded by the camera's limited field of view or obstacles that cover parts of the markers. These problems limit the area that can be efficiently tracked and constrain the user's freedom of movement and the ability to use markers as artifacts to handle virtual content.

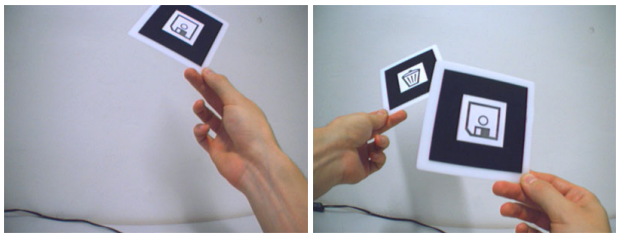

Figure 1. Common occlusion situations.

To improve tracking while preserving the flexibility of our system, which allows users to move around freely (even outdoors) and spontanously join collaborative sessions with other users, we developed an approach that uses, but is not dependent on additional resources. Additional cameras can cover a bigger part of the scene and contribute to the overall tracking information. We did not want to rely on statically mounted cameras but also use, for example, the camera of another user who is passing by or talking to us. In other words, the goal was to use a best-effort approach using all tracking information that is available somewhere in the system to complete missing local tracking information.

2. Occlusion

Tracking in ARToolkit is impeded whenever the marker to be tracked is not fully and clearly visible within the camera image. Fig. 1 shows some captured camera images

with common occlusion situations. Note that in some situations the user might not even realize that the marker is not tracked, because human vision is much more powerful in completing missing information than computers currently are. Especially if no feedback of the tracking success is given to the user, this can lead to annoying situations or data loss (If the user thinks her actions are recorded, but tracking is defunct).

Failed tracking can generally be explained with one of three common reasons. For all of them, some improvements have been proposed:

1. Bad lighting conditions

   Optimal results require diffuse, bright white light with constant intensity over the whole area where tracking will be used. If there is not enough light, markers will not be recognized by ARToolkit. If the light is too bright or shines directly onto the marker, the black parts of the marker will reflect light into the camera, also preventing it from being recognized in the captured image.

   Well-known solutions for this problem are to modify the lighting in the room, or to use non-reflecting materials for the black parts of the markers to reduce reflection.

2. The marker is not fully visible in the camera image

   The marker must face the camera, be fully within the cameras field of view and not occluded by obstacles or other markers. If only one corner ot the marker is occluded or outside the FOV of the camera, it will not be recognized at all.

   Chances of full visibility can be improved by using several markers fixed to a rigid object. The offsets between the markers must be well-known, and there must be some component in the application that calculates the final position of the object from the last valid tracking input(s). In our application, we use this approach to track objects that can be rotated freely by the user, for example a pen (Fig. 2)

3. The marker is too small or too far away

   If a markers image does not cover enough pixels in the camera image, the tracking results will be inaccurate or the marker will not be recognized at all.

   A solution is to use markers of appropriate size. In our setup, we achieved acceptable results with three different marker sizes (measure given is the side of the black square):

   - 50mm Personal space (0.2m - 0.6m) Markers that are only used near the camera of one person.
   - 90mm Artifact space (0.4m - 2m) Markers that are used to handle virtual artifacts and to pass them to other users.
   - 180mm Environment space (1m - 5m) Markers that are part of the environment (attached to walls, viewed from a distance)

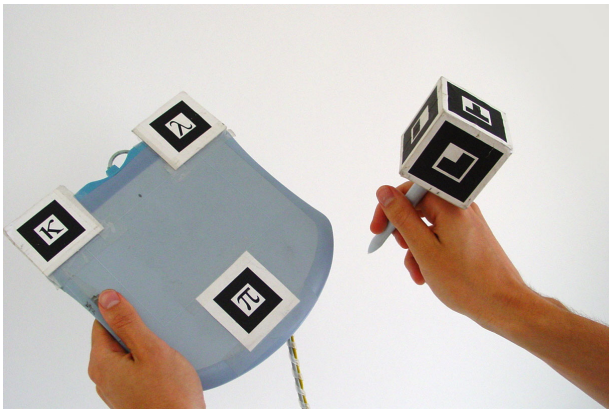

Figure 2. Multiple markers attached to a tablet and a pen, which can be rotated freely.

While tracking can be further improved in static setups by mounting the camera in an optimal position (e.g. overhead), this approach was not feasible for building our mobile augmented reality kit – the kit worn by the user is self-contained and cannot rely on laboratory conditions with good lighting and a fixed camera. However, *additional* cameras can be used to complete missing tracking information.

3. Static multi-camera tracking

Using more than one camera to contribute to the tracking information on a single host requires a far more complex infrastructure than the single-camera solution. Either multiple cameras have to be attached to a single computer, requiring non-standard driver software and modifications to the tracking software, or each camera is connected to a seperate host, which shares the tracking data over a network with other hosts participating in the system.

In both cases, the tracking information has to be transformed from each cameras local coordinate system to a common reference coordinate system. If the cameras are mounted in fixed, well-known positions, it is easy to add a fixed offset (which can be measured by hand) to the tracking results of one camera to transform the results into the coordinate system of the second camera, or use offsets for both cameras to transform the data into an arbitrary world coordinate system.

If the offset between the two cameras is not known, or cannot be measured accurately (for example, because one camera moves with the user), one could use an additional ”reference marker” to track the positions of both cameras in relation to the reference point, and then calculate the relation of the cameras to each other out of that information. A further improvement would be to use several such markers mounted in well-known positions (for example, on the walls of a room), to allow the positions of the cameras to be tracked [7].

The benefit of multi-camera tracking is that different regions can be covered by the tracking system (which increases the effective field of view), or the same region can be covered from different angles, which helps in case of occlusion and reflection problems.

4. Dynamically shared multi-camera tracking

Although, in our mobile application, we cannot rely on additional tracking equipment to improve the accuracy and field of view of our ARToolkit tracking system, we still want to use the information provided by additional cameras, whereever available. To meet our need of users moving around freely, joining and leaving collaborative sessions with other users spontaneously, we needed a more flexible approach than described above.

In our approach, tracking data of each host is shared across the network with all hosts participating in the system. Pairwise camera-to-camera relationships are established automatically, as soon as any marker is seen by the two cameras in question – no matter if the second camera is carried by another user or just mounted at a ”hot spot” location to improve tracking in that area.

The dynamically calculated offset between the two cameras is then used to transform all tracking events coming from the network to the hosts own local coordinate system. Note that this is done on both hosts symmetrically, so as soon as there is one marker that can be seen from both cameras, both hosts can calculate tracking information for all markers that are known to them. Both hosts end up with positions for all markers (even those that they cannot see, but are tracked by the other host) in their local coordinate system.

In this section, for simplicity, we will illustrate the principles of our shared tracking system for a two host setup. Extending the system on multiple hosts will be discussed in section 4.5.

Fig. 3 gives a schematic overview of such a shared tracking session between two users. As soon as both users can see marker 1, both of them have local tracking data and remote tracking data for that marker available. Both hosts can then calculate the relative position of the other camera, and transform the remote tracking data accordingly. For example, host 1 has no local tracking data for marker 2 (because it is hidden behind another object), but it can still calculate its position from the relationship of the two cameras plus the tracking information sent from host 2.

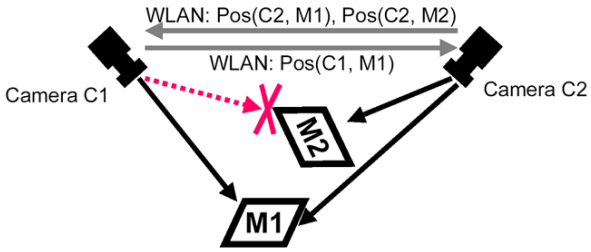

Figure 3. Schematic view of our shared tracking setup: The two hosts exchange their tracking data over the network.

4.1. Calculating the offset

Let us assume the situation in Fig. 3, with 2 hosts and two markers, one of which cannot be tracked directly by host 1. Let A_1 be the transformation matrix of marker A as tracked by host 1, and A_2 and B_2 the transformation matrices of the two markers as tracked by host 2. We want to find a transformation matrix X_{21} that transforms any event from a marker M in the coordinate system of host 2 (denoted as M_2) into the coordinate system of host 1 (denoted as M_1), so that

$$M_2 \cdot X_{21} = M_1$$

To get this transformation matrix, host 1 calculates

$$X_{12} = A_1^{-1} \cdot A_2$$

from his local information and the data received from host 2 and takes the inverse

$$X_{21} = X_{12}^{-1}$$

With this matrix, all events sent from host 2 can be transformed into the coordinate system of host 1. For example, the position of marker B can be calculated as

$$B_1 = B_2 \cdot X_{21}$$

4.2. OpenTracker

For processing tracking data, we use our software framework OpenTracker [2, 3]. OpenTracker processes data from

various tracking sources (each ARToolkit marker acts as a seperate tracking source) through a graph of filters, finally sending the processed tracking data to the application or over the network to other hosts. The whole processing graph can be described in an XML configuration file, allowing quick reconfiguration of the tracking sub-system. Tracking sources (i.e. ARToolkit markers) appear as leaf nodes in the processing hierarchy described by the configuration file.

Tracking data in OpenTracker is modeled as a stream of events, originating at the tracking source and running through the filter graph, finally being sent to one or more sinks. Filters can modify or block events according to their configuration. Each event has fields for position, orientation, timestamp and confidence value, among others. Table 1 gives an overview of OpenTracker filters that are used in our setup.

To illustrate the operation of OpenTracker, Fig. 4 shows a simple configuration: Two ARToolkit Markers are defined to act as event sources. The tracked positions of the two markers are routed directly to the application (Symbolized by the "ApplicationSink" nodes at the bottom). In addition, a DynamicTransform node calculates the difference between the two positions, and sends the result to the application via a third sink.

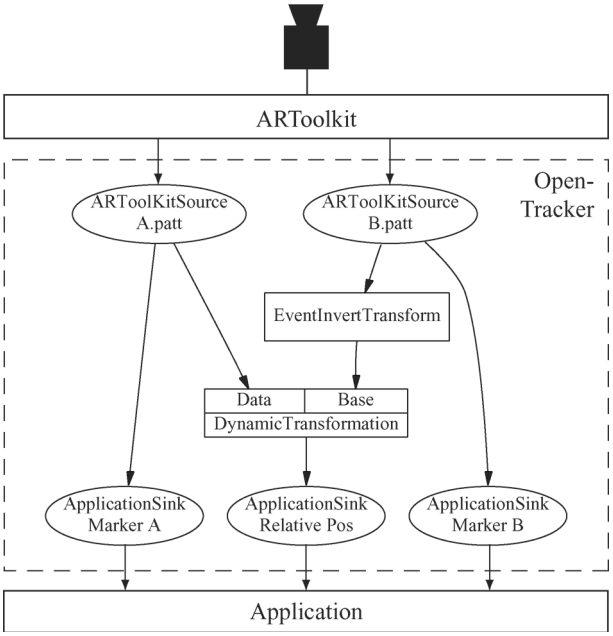

Figure 4. A Simple OpenTracker configuration: Two ARToolkit markers act as event sources, they are transformed and routed to the application.

4.3. Sharing tracking data

For sharing tracking data over the local network, we use UDP multicast, broadcasting the events on a port that is well-known to other hosts participating in the setup. This functionality is provided by OpenTrackers NetworkSink and NetworkSource nodes. Each host sends out the raw (untransformed) tracking data of all markers that it recognizes in its camera image. In addition, it listens to incoming tracking data of all other hosts on the network.

If a host has local tracking data for a marker available, and receives remote tracking data for the same marker, it can calculate the camera-to-camera relationship to the foreign host. We use a timeout of 200 milliseconds (roughly half the frame-rate of our ARToolkit setup) to detect if at any given moment there is local and remote tracking data available.

Fig. 5 shows the data-flow diagram for a shared tracking session between two hosts, using one marker, which acts as reference marker.[1] As can be seen clearly, the setup on both hosts is exactly symmetrical. Each host has an ARToolkit source, that fires events as soon as the marker can be seen by its camera. This event is sent to the other host via the network.

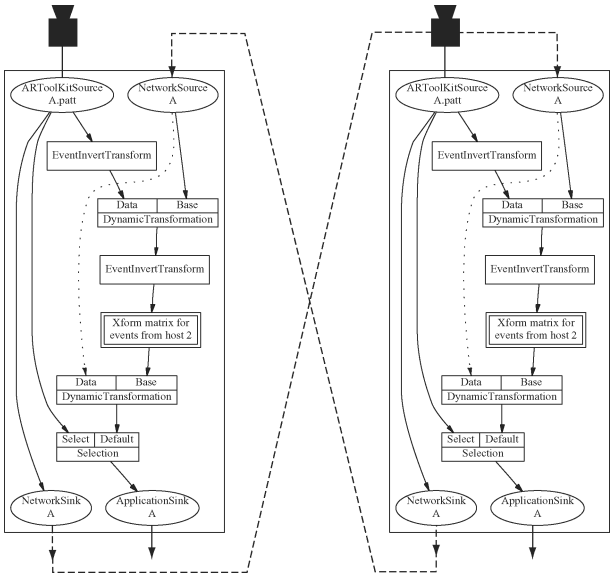

Figure 5. Shared tracking for one marker and two users.

If one host has local tracking information and remote tracking information available, the transformation matrix

[1] Of course, shared tracking only makes sense if at least two markers are used. For illustration purposes we show only the data flow of one marker in the figure, a complete configuration for three markers is shown in figure 6.

Node name	Description	Transformation
Transform	Transfroms the input with a given transformation matrix.	$O = I \cdot T$
EventInvertTransform	Calculates the inverse of the input.	$O = I^{-1}$
EventDynamicTransform	Transforms the first input with the second input.	$O = I_1 \cdot I_2$
Selection	Forwards the preferred input, if available within a given timespan. Otherwise, the default input is forwarded.	
ConfidenceSelect	Forwards the event with the highest confidence value within a given timespan.	
Filter	Performs linear averaging of a number of (sequential) events	

Table 1. Common OpenTracker nodes.

for remote events can be calculated as explained in Section 4.1. Therefore, the inverse of the local event is calculated by an EventInvertTransform node, and the result is multiplied with the remote event by a DynamicTransformation node. Calculating the inverse of the result gives us the desired transfromation matrix X_{21}.

Tracking data for other markers that is received via the network is then transformed with this matrix and fed into a Selection filter. The selection filter prefers local tracking information, if available within a given timeout. If no local tracking data for the given marker is available, the transformed remote data is taken and sent to the application.

4.4. Confidence selection

If several markers are seen by both hosts, we want to use the marker that is tracked most accurately by both hosts as reference marker. Confidence Values are provided by ARToolkit as an estimate how accurate the tracking is. Confidence values range from 0.0 to 1.0, with higher values representing more accurate tracking. OpenTracker provides a ConfidenceFilter node that take several inputs and only passes the one with the highest confidence value. Normally, if there are several markers tracking a single object, one would use the tracking data with the highest confidence value as input for the application.

Since in our approach, the reference marker is seen by two cameras, the two confidence values must be combined to give a correct decision parameter for selecting the ”best” marker. We interpret the confidence value as a probability measure for the correctness of the sampled data, and therefore multiply the two confidence values to get a combined value. If more than one marker is seen by both cameras, the marker with the highest (combined) confidence value is taken to calculate the offset between the two cameras.

4.5. Multiple hosts

Since the configuration on each host is completely symmetric, it is very easy to extend the system from two to multiple hosts sharing their tracking data. Although every host communicates with every other host in the system, network traffic increases only linearly, because UDP multicast on a single channel is used for transmission of tracking data. Since it can be assumed that co-located cameras are connected via a single LAN segment, this is an efficient way to solve the distribution of the event data.

Each host has to maintain a transformation matrix for every host that is known to it, and the subgraph for calculating the transformation matrix and transforming remote events must be defined in the configuration file. This results in fairly large processing graphs with modular structure that can be easily extended to include more hosts or markers. Although, at the moment, this is done manually, this process can easily be automated (see Section 6).

4.6. Putting it all together

Fig. 6 shows the data flow on one host in a 2 host setup, using 3 markers, confidence selection and filtering for the events sent over the network. Fig. 7 shows video-overlay images of successful shared tracking.

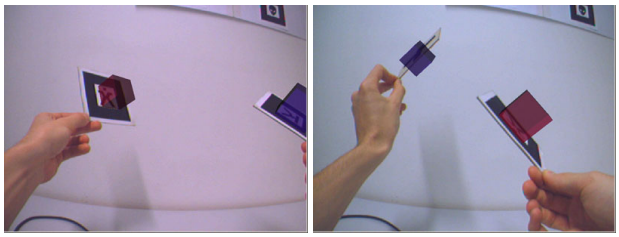

Figure 7. Successful shared tracking.

5. Applications and results

In our application, the camera capturing the image is mounted onto a helmet worn by the user (see Fig. 8). See-through augmented reality is provided by a Sony glasstron Head-Mounted display. Since there is a fixed offset between camera and display, we can use the camera and ARToolkit

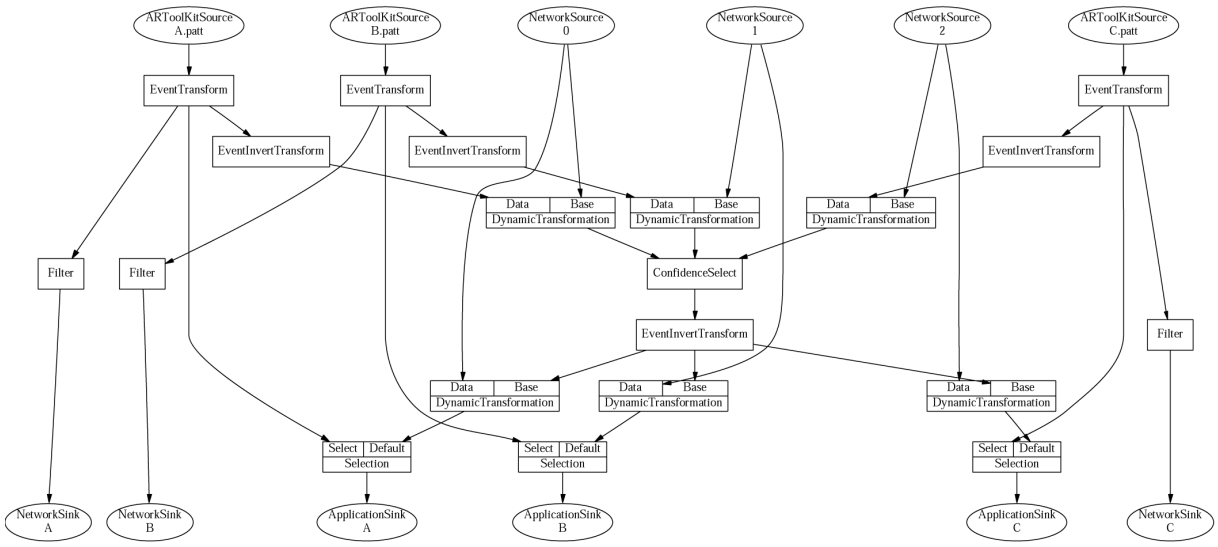

Figure 6. Data Flow Diagram for setup on one host, using 3 markers A, B and C.

to get the position of markers relative to the users viewpoint (adding the fixed offset to the tracking data).

The helmet described above is part of our mobile AR kit [4] – the user is wearing the helmet with camera and display, and a backpack with all the necessary hardware and batteries. As an interaction device, gloves [6] are provided that are also tracked by ARToolkit, allowing the user to ”pinch” or ”grab” virtual objects and still keep his or her hands free for interaction with real objects.

Figure 8. The mobile AR Kit.

Dynamically shared tracking has greatly improved the tracking availability, espescially in collaborative sessions with two or more users – ”split reality” situations, in which one user sees tracked content that is invisible to the other are greatly reduced. Although the calculated results are not as accurate as local tracking, they are sufficient to create the impression of continuity of the AR environment.

As a meeting place for collaborative sessions we have built a small table with an integrated overhead camera that performs ”hot spot” tracking of the table surface. The table is integrated like every other host into the tracking system, and provides reliable tracking of markers lying on its surface. Users can concentrate on each other or their interaction devices and still see the virtual objects placed on the table, even if the markers are not visible to the cameras mounted on their heads.

5.1 Accuracy

The accuracy of shared tracking depends on many parameters in the processing chain: The quality of the camera images, calibration of ARToolkit, size and visibility of the reference marker, angle and distance between the two cameras, the size of the marker to be tracked and network latency, to name only the most important ones. If only one of these factors is not optimally set, the results of shared tracking may be inaccurate or even unusable.

To measure the accuracy of our setup, we recorded the tracking of an artifact-space marker that was directly visible to both cameras during a collaborative session of two users. As well as the local output of ARToolkit, we recorded the calculated output of shared tracking via a second marker. With this information, we could plot the deviance between our shared tracking system and the direct output of ARToolkit-based tracking. The plot is shown in fig. 9, together with a plot of the movement speed of the marker. Deviance ranges from 2 to 10 cm, with higher

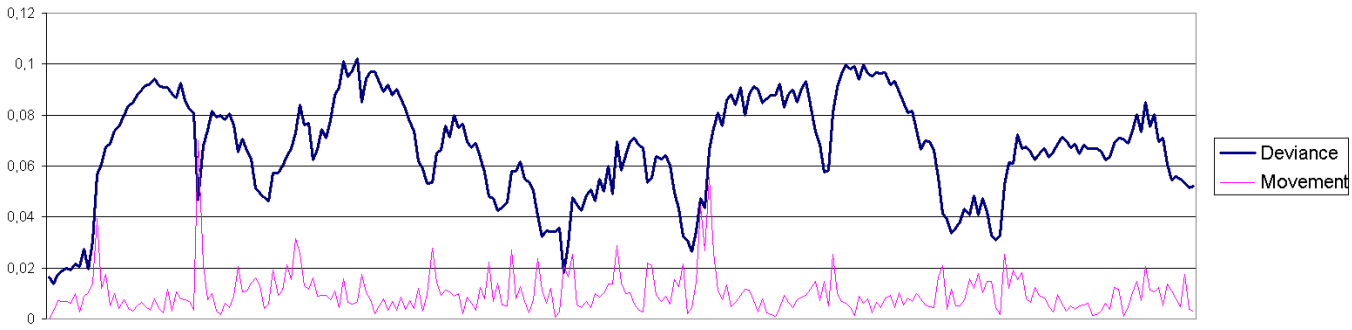

Figure 9. Plot of the deviance (in meters) between direct ARToolkit tracking and shared tracking during a collaborative session. The thin line shows the movement speed of the marker the measure was taken from.

deviations during and after fast movements of the tracked marker, as can be seen in the figure.

The subjectively perceived error is not as big as these numbers suggest, because the direction of the deviance vector tends to stay reasonably constant over the whole session. Users may notice a constant offset between marker and virtual content during shared tracking, but can handle virtual artifacts nearly as good as if directly tracked. For input devices and applications that require precise positioning, extra measures like using cameras with higher resolution and precise calibration of each individual camera have to be taken to improve the accuracy of the underlying ARToolkit tracking.

Another method of improving the accuracy of ARToolkit's output is to simply apply a linear averaging filter to the data. OpenTracker provides such a filter with its *Filter* node, and we use it to filter the last 3 events sent over the network for shared tracking. Local events are passed to the application without filtering, to keep response times low for local changes. Events sent over the network are filtered, which adds a delay of approximately 300ms but reduces noise and errors in the data used for shared tracking.

5.2 Additional improvements

A strength of OpenTracker is its ability to integrate various tracking technologies into a common, homogenous framework. The idea of a shared tracking system is therefore not limited to ARToolkit as its data source, but can include different tracking systems, translating between them as necessary and using the best available data as input for the application. For example, if a mobile user enters a room equipped with a magnetic tracking system, all the data provided by magnetic tracking is immediately available if there is one marker that is fixed or tracked by the magnetic system and ARToolkit. The mobile system doesn't even have to know that there is a magnetic tracking system, all it sees is another host providing tracking data and the necessary information to translate between the two coordinate systems.

Knowledge about the environment can also be used to further extend the range of operation of shared tracking. If there are multiple markers attached to the walls of a room or a building, the knowledge of the geometric relationships between these markers can be used to translate between two cameras coordinate systems, if each camera sees any of these markers – even if no single marker is seen by both cameras at the same time. A similar improvement is to attach markers to the cameras themselves (or, in our mobile system, to the helmet carrying the camera), which enables translation of tracking data if one camera sees the other camera (as it is the case if two users are talking to each other face-to-face).

The helmets of our mobile AR kits are also equipped with inertial trackers, measuring head movements. Although, due to accumulating errors, they cannot be used to accurately track the head of the user over a long time, their output allows us to bridge the time gap if the user turns his head, providing a continuous stream of tracking data until the next marker is visible.

6. Conclusions and future work

We presented a setup for distributed tracking that can make use of multiple cameras without being dependent on them. The system, as described here, relies heavily on our tracking middleware OpenTracker, because it allows us to design complex dataflow graphs (including network transmission) by simply editing XML configuration files.

The resulting system is even easier to set up and maintain than the static approaches described in section 3, because no manual registration of cameras or environment is necessary for the system to work.

For the application, using dynamically shared tracking is completely transparent. All processing of local and remote tracking data is done by OpenTracker, the application just receives the events that are provided on a best-effort basis – if local information is available, it is used, otherwise the missing information is calculated (if possible) from data received from other hosts.

One possible future improvement would be the automatic detection and joining of additional hosts – currently this is done by editing the configuration files on all hosts participating in the system. Since the changes to the configuration files are already quite modular, it should be easy to automate this. However, this requires application support for performing the run-time changes to the undelying OpenTracker configuration, and an additional protocol for discovering new hosts.

Another idea proposed in [7] is the partitioning of the environment into zones, where the same marker has different meanings. Depending on "context markers", the tracking subsystem decides which zone is currently active and sends corresponding tracking output to the application and other hosts. This allows the coverage of large environments (outdoors, large buildings) with only a reasonable small set of markers which can be reused, depending on the location the user is currently in. This could also be combined with GPS positioning of the user to determine the currently active zone. Such a system could be implemented in OpenTracker alone, which sends the output of one marker to various "virtual" ApplicationSinks, depending on the context of operation.

7. 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 grant ("MARDIS").

OpenTracker software for shared and distributed tracking is freely available under LGPL at the project web site *http://www.studierstube.org/opentracker*.

References

[1] Kato H., Billinghurst, M. (1999). *Marker Tracking and HMD Calibration for a video-based Augmented Reality Conferencing System*, Proceedings International Workshop on Augmented Reality (IWAR99). October, San Francisco, USA.

[2] Reitmayr G., D. Schmalstieg (2001). *An Open Software Architecture for Virtual Reality Interaction*, ACM Symposium on Virtual Reality Software and Technology 2001 (VRST 2001), Banff, Alberta, Canada, Nov. 15-17, 2001.

[3] Opentracker Website, visited July 05, 2002. *http://www.studierstube.org/opentracker*.

[4] Reitmayr G., D. Schmalstieg (2001). *Mobile Collaborative Augmented Reality*, Proceedings International Symposium on Augmented Reality (ISAR'01), New York NY, Oct. 29-30, 2001.

[5] Schmalstieg D., A. Fuhrmann, G. Hesina, Zs. Szalavári, L. M. Encarnaçao, M. Gervautz, W. Purgathofer (2002). *The Studierstube Augmented Reality Project*, PRESENCE, 11(1), pp. 33-54.

[6] Veigl S., A. Kaltenbach, F. Ledermann, G. Reitmayr, D. Schmalstieg. *Two-Handed Direct Interaction with ARToolKit*, IEEE First International Augmented Reality Toolkit Workshop (ART02), Darmstadt Germany, Sept. 29, 2002, to appear.

[7] Kalkusch M., T. Lidy, M. Knapp, G. Reitmayr, H. Kaufmann, D. Schmalstieg *Structured Visual Markers for Indoor Pathfinding*, IEEE First International Augmented Reality Toolkit Workshop (ART02), Darmstadt Germany, Sept. 29, 2002, to appear.