T U M

I N S T I T U T F Ü R I N F O R M A T I K

Fundamentals of Ubiquitous Tracking for Augmented Reality

J. Newman, M. Wagner,
T. Pintaric, A. MacWilliams, M. Bauer,
G. Klinker, D. Schmalstieg

TUM-I0323
Dezember 03

T E C H N I S C H E U N I V E R S I T Ä T M Ü N C H E N

TUM-INFO-12-I0323-0/1.-FI

Druck: Institut für Informatik der
Technischen Universität München

Fundamentals of Ubiquitous Tracking for Augmented Reality

Joseph Newman[2], Martin Wagner[1], Thomas Pintaric[2], Asa MacWilliams[1], Martin Bauer[1], Gudrun Klinker[1], and Dieter Schmlastieg[2]

[1] Institut für Informatik, Technische Universität München
{wagnerm, macwilli, bauerma, klinker}@in.tum.de
[2] Vienna University of Technology
{newman, pintaric, schmalstieg}@ims.tuwien.ac.at

Abstract. To enable rich and meaningful Augmented Reality (AR) experiences within a Ubiquitous Computing environment, a detailed, coherent and up-to-date spatial model of the world is essential. However, current tracking technologies are limited in their range and operating environments. This has, so far, restricted the development of wide-area AR applications. To extend the range of AR applications, it will be necessary to combine widely different tracking technologies dynamically, aggregating their data and balancing their trade-offs. In this paper, we propose a formal framework, called *Ubiquitous Tracking*, which uses a graph-based model of spatial relationships to build dynamically extendible networks of trackers suitable for the high-precision, low-latency requirements of Augmented Reality. The framework is powerful, allowing us to model existing complex tracking setups; extensible, accommodating new trackers, filtering schemes and optimisation criteria; and efficient, allowing an effective implementation within existing AR systems.

1 Introduction and Related Work

There are many parallels and intersections between the fields of Ubiquitous Computing [27] (UbiComp) and Augmented Reality (AR). Approaches such as Sentient Computing [1, 11, 13] maintain a detailed and up-to-date model of spatial relationships, which appear to reproduce the perceptions a user has of the world. Given a suitably large model it is possible to create AR scenarios that allow users to roam through a wide area interacting with a rich and responsive environment.

In contrast, the majority of AR applications developed so far has been constrained, by the working volumes of existing tracking technologies, to carefully arranged spaces of a few square meters. Examples include the Boeing wire assembly feasibility study [7] and classroom-based geometrical construction [15]. Systems that have aimed at true mobility such as the Touring machine [9], Sentient AR [21], and Tinmith [23] have typically relied on a ubiquitous tracking infrastructure, such as GPS. Wide-area trackers generally provide only modest levels of accuracy at a low update rate, and cannot be used for tasks requiring greater precision. Furthermore, these systems generally assume that sensors are

deployed homogeneously throughout the entire area of interest, which results in much tedious off-line calibration and registration.

In this paper we propose a new approach, *Ubiquitous Tracking*, which formalises a mathematical framework in which dynamic qualities of spatial relationships between objects can be modelled as a graph. The mathematical specification has been designed, such that a software implementation of the framework maps intuitively to the component-based paradigm used in AR frameworks such as DWARF [3] and Studierstube [25].

In real large-scale AR environments it is probable that the quality of tracking will vary significantly. There will be a few, small high precision spots where expensive trackers such as those used to track medical instruments in an operating theatre would support a particular task, e.g. surgery. Conversely, in other areas, like hallways, cheap cell-based tracking may be sufficient. An optimal trade-off between precision, hardware cost and complexity must be struck in order to make such a system both effective and affordable and will be the ultimate criterion on which the strength of the framework will be judged.

2 Spatial Relationship Graphs

In order to provide a rich and meaningful Augmented Reality experience, it is important that we have a consistent, up-to-date model of the world which contains relevant environmental state, especially the spatial properties of objects.

The goal of the theoretical framework discussed in this paper is to provide a query mechanism that returns an optimal estimate of the geometric relationships between arbitrary objects, according to user-defined criteria.

For this purpose, we use a a graph-based model of spatial relationships. We first discuss properties of real-world relationships, then consider properties of measurements made in the real world and finally present a general concept of how knowledge can be inferred from these measurements, resulting in the construction and maintenance of a model of the real world.

2.1 Fundamentals

We visualise the spatial relationships between objects in a graph structure [5] in which nodes represent objects and edges represent spatial relationships between the objects.

A tracking device is simply a sensor that makes measurements of the spatial properties of objects within range. Initially we will consider object pose before turning to other properties such as velocity and acceleration. In order to to interpret data appropriately other attributes such as a timestamp and the uncertainty associated with a measurement must also be considered. We distinguish between three different sorts of graph: an idealised view of the world from the point of view of an omniscient observer, directly measured relationships corrupted by noise, and finally a graph derived from inferred relationships.

2.2 Real Relationships

In the real world, each pair of objects has, at every point in time, a geometric relationship that can, for example, be expressed using the standard computer graphics notation of a 4×4 homogeneous matrix representing arbitrary transformations between coordinate systems. We define a binary relation Ω on our object space $N = \{A, B, C, \ldots\}$. We then map every element (X, Y) of Ω onto a function w_{XY} describing the spatial relationship between the objects X and Y over time. This attribution scheme is called $\mathbf{W}$.

$$\mathbf{W} : (\Omega = N \times N) \rightarrow w, \text{ where } w : D_t \rightarrow \mathbb{R}^{4\times4} \tag{1}$$

D_t is the source time domain, mapped by w onto the target spatial relationship domain. This definition matches the output of common tracking devices, yielding spatial relationships for different points in time.

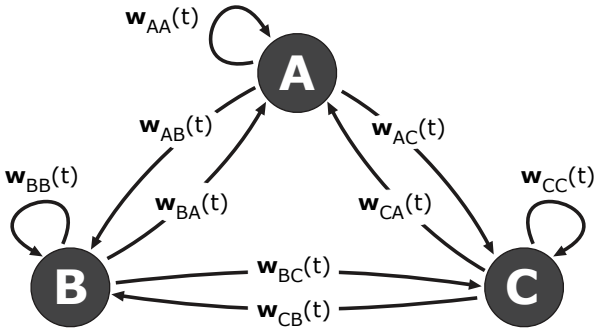

Fig. 1. Complete graph representing geometric relationships between three objects as perceived by an omniscient observer

For clarity of notation, we define the directed graph $G(\Omega)$ that represents the relation Ω. Thus, whenever two objects $A, B \in N$ are in a geometrical relationship with one another, the nodes representing these objects are connected by an edge that is annotated with the function w_{AB}. It can be seen in figure 1, that Ω is *transitive, reflexive* and *symmetric*, yielding a complete graph $G(\Omega)$. An omniscient observer would be able to perceive all geometrical relationships represented in this graph and the Ubiquitous Tracking problem would be solved.

2.3 Measured Relationships

Unfortunately, we can only make estimates of geometric relationships between real objects by performing *measurements*. Each measurement is made at a discrete point in time, yielding a geometric relationship that is equivalent to the real relationship, but corrupted by noise. In order to generalise the framework

to accommodate all possible measurements, we do not restrict ourselves to a fixed noise model, but rather describe the quality of measurements using a set of *attributes*, $\mathcal{A}$, which includes properties such as the latency between the actual measurement and the time at which the result is delivered, or a standard deviation in meters. A more detailed discussion of possible attributes is given in section 3.

By analogy to the relation Ω, we now define a relation Φ and an attribution $\mathbf{P}$ describing the measurements:

$$\mathbf{P} : (\Phi \subseteq N \times N) \to p, \text{ where } p : D_t \to \mathbb{R}^{4\times4} \times \mathcal{A} \tag{2}$$

The directed graph in figure 2, $G(\Phi)$, provides a visual representation of the relation Φ. An edge between two nodes exists only if measurements have been made. In the example, we assume that the geometric relationship between objects A and B has been measured twice, at times t_1 and t_2, yielding homogeneous matrices H_1 and H_2 and attribute sets $\mathcal{A}_1$ and $\mathcal{A}_2$ describing the quality of the measurements. The relationships between A and C and between B and C were only measured once at times t_3 and t_4 respectively, yielding $H_3, \mathcal{A}_3$ and $H_4, \mathcal{A}_4$ respectively. The relation Φ represented by the graph $G(\Phi)$ can be seen to be neither symmetric, transitive nor reflexive.

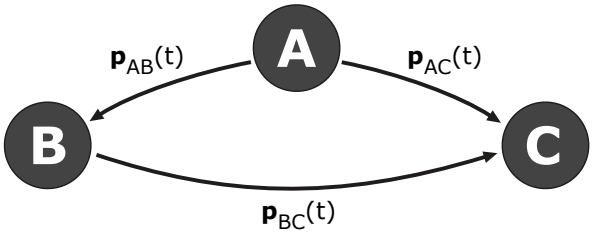

Fig. 2. Graph $G(\Phi)$ describing the measurements we made.

Note that the source time domain D_t of p is not continuous, instead consisting of a discrete set of points in time at which the measurements were taken. In our example, we derive the following functions p:

$$\begin{aligned} p_{AB} : \{t_1, t_2\} &\to \mathbb{R}^{4\times4} \times \mathcal{A} \\ t_1 &\mapsto (H_1, \mathcal{A}_1) \\ t_2 &\mapsto (H_2, \mathcal{A}_2) \end{aligned} \qquad \begin{aligned} p_{AC} : \{t_3\} &\to \mathbb{R}^{4\times4} \times \mathcal{A} \\ t_3 &\mapsto (H_3, \mathcal{A}_3) \\ p_{BC} : \{t_4\} &\to \mathbb{R}^{4\times4} \times \mathcal{A} \\ t_4 &\mapsto (H_4, \mathcal{A}_4) \end{aligned} \tag{3}$$

2.4 Inferred Relationships

The measurement relation Φ and its associated graph $G(\Phi)$ are of little use in providing a general query mechanism. At any point in time $t = T$, we can usually

expect to get no result from a query for the relationship between two objects, unless there has been a measurement at exactly time T, which is improbable.

In consequence, we have to *infer* knowledge about spatial relationships between objects. Our final goal is the binary relation Ψ that approximates the world relation Ω:

$$\mathbf{Q} : (\Psi \subseteq N \times N) \rightarrow q, \text{ where } q : D_t \rightarrow \mathbb{R}^{4\times 4} \times \mathcal{A} \tag{4}$$

Every q is a multivalued function that can be seen as the union of basis functions q', q'', ... Thus, the corresponding graph $G(\Psi)$ is modelled best as a multigraph, having an edge between two nodes representing objects A and B for every basis function that was inferred.

Adding new inferences. The relation Ψ is assembled by making new inferences of geometrical relationships, and for each new inference adding a new edge to the graph $G(\Psi)$. The process of assembling Ψ will be illustrated by using examples of increasing complexity starting with the simplest possible setup consisting of just two objects; $N = \{A, B\}$. Initially, given the measurements from the relation Φ:

$$q^m_{AB}(t) = p_{AB}(t) = \begin{cases} (H_1, \mathcal{A}_1) & \text{if } t = t_1 \\ (H_2, \mathcal{A}_2) & \text{if } t = t_2 \\ \text{undefined} & \text{otherwise} \end{cases} \tag{5}$$

The definition range of q^m_{AB} is $D_{q^m_{A,B}} = \{t_1, t_2\}$. By merely extending the definition range we obtain a new straightforward inference and consequently add a new edge to the graph $G(\Psi)$ between nodes A and B:

$$q^e_{AB}(t) = \begin{cases} (H_1, \mathcal{A}_1) & \text{if } |t - t_1| < |t - t_2| \\ (H_2, \mathcal{A}_2) & \text{otherwise} \end{cases} \tag{6}$$

Obviously, there are many other inference schemes that can be applied in parallel. The properties of a particular inference can be expressed through the attributes, which are likely to different from that of the raw measurement.

Given additional knowledge concerning the relation between objects A and B, further inferences can be made, yielding a general filter function

$$q^f_{AB}(t) = f((H_1, \mathcal{A}_1), (H_2, \mathcal{A}_2), \ldots, t) \tag{7}$$

For example, if we assume that A moves smoothly with respect to B, we could peform interpolation $q^i_{AB}(t)$, using knowledge of motion properties could yield a Kalman filter $q^k_{AB}(t)$ or particle filter $q^p_{AB}(t)$.

Modelling inferences as edges leads to a natural mapping to software components; every new inference is conceptually a new software component added to the overall tracking setup. This is described in detail in section 4.1.

Choosing an optimal inference. Figure 3 shows a graph with the inferences made thus far. The query mechanism proposed in the framework must now choose between these inferences, i.e. one edge from node A to node B. Obviously, the

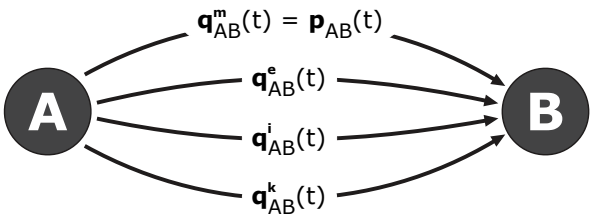

Fig. 3. A graph describing multiple inferences about the spatial relationship between two objects.

inquirer desires an "optimal" result (according to certain criteria). However, we cannot know *a priori* the optimal result for a particular application, as some applications have strict real-time requirements whilst others might prioritise high precision.

To address this problem, we define an error function operating on the attributes:

$$\begin{aligned} e : \mathcal{A} &\to \mathbb{R} \\ \mathcal{A}_i &\mapsto e(\mathcal{A}_i) \end{aligned} \tag{8}$$

The query mechanism now evaluates the error function for all inferred functions and returns the one corresponding to the lowest value. The general definition of an error function provides a plug-in mechanism for designing arbitrary optimisation criteria; it is the responsibility of the designer of the error function, for example, to favour low-latency inferences over those with low absolute spatial errors. An in-depth discussion of the error function's role can be found in section 3.

Making optimal inferences given complex setups. The simple two node example is now extended to n nodes. Given a query "Optimal spatial relationship at time T between objects X and Y with error function e", the framework performs the following steps:

1. Find all paths from X to Y in graph $G(\Psi)$, with the additional constraint that an edge corresponding to an inference $q(t)$ is only considered if $q(t)$ is defined at $t = T$.
2. Evaluate the given error function e over all the detected paths. The definition of the error function must be extended, essentially assigning the error function an ordered set of attribute sets to evaluate.

$$\begin{aligned} e : \mathcal{A}^* &\to \mathbb{R} \\ \mathcal{A}^*_i &\mapsto e(\mathcal{A}^*_i) \end{aligned} \tag{9}$$

3. Use the path corresponding to the lowest value of e to calculate the spatial relationship between X and Y by multiplying all the homogeneous matrices along the path together.

We cannot guarantee that a well-defined and correct set of attributes describing the properties of the query result is returned. The effect of a path evaluation on the resulting attributes is highly application-dependent and cannot be solved generally.

Every query adds an additional entry to $\mathbf{Q}$ and an extra edge to $G(\Psi)$. Section 4 contains details as to how the large amount of data resulting from queries can be dealt with.

Example. The measurements depicted in figure 2, have an attribute set consisting of a single value, the latency of the measurement. Assuming $\mathcal{A}_1 = \{2\text{ms}\}$, $\mathcal{A}_2 = \{2\text{ms}\}$, $\mathcal{A}_3 = \{10\text{ms}\}$ and $\mathcal{A}_4 = \{5\text{ms}\}$, we can initialize relation Ψ with the measured functions in equation (2), setting $q^m_{XY}(t) = p_{XY}(t)$. For the sake of simplicity, we add the simple inference q^e, defined analogously to equation (6), with no modification of the attribute set. The resulting graph is shown in figure 4.

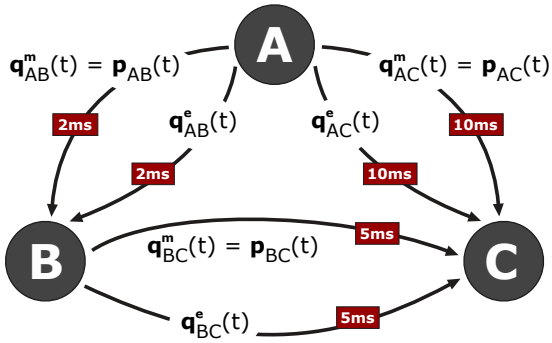

Fig. 4. A graph describing multiple inferences about the relationship between three objects. The attributes of the relation functions are shown in the boxes on the edges.

Assume further that the measurements were taken at different times, such that $t_1 < t_2 < t_3 < t_4$ and that we query for the relationship at time $T = t_3$. Thus, step 1 of the algorithm yields the following paths:

$$\mathbf{P1} : A \xrightarrow{q^e_{AB}} B \xrightarrow{q^e_{BC}} C, \qquad \mathbf{P2} : A \xrightarrow{q^m_{AC}} C, \qquad \mathbf{P3} : A \xrightarrow{q^e_{AC}} C$$

The error function e is defined such that it returns the sum of latency values of all edges along a given path: $e(\mathbf{P1}) = 2\text{ms}+5\text{ms} = 7\text{ms}$, $e(\mathbf{P2}) = 10\text{ms}$ and $e(\mathbf{P3}) = 10\text{ms}$. In consequence, the algorithm chooses $\mathbf{P1}$ and returns $(H_2 \cdot H_4, \mathcal{A}_{new})$. $\mathcal{A}_{new}$ can be calulated by summing the latencies along all the involved edges, giving $\mathcal{A}_{new} = \{7\text{ms}\}$.

Modelling sensor fusion and filter schemes. Hightower [12] describes sensor fusion as the "use of multiple location systems simultaneously to form hierarchical

and overlapping levels of sensing to increase accuracy beyond what is possible using any individual system." Multiple inferences of the type described earlier correspond to Hightower's "levels of sensing". Fusion can be performed by taking several measurements and then inferring a new estimate for a given time. If we fuse multiple information sources of object X's pose relative to object Y, we add a new inference q^f_{XY} to $\mathbf{Q}$, leading to a new edge AC in $G(\Psi)$.

Other filter schemes can be modelled in the same way as sensor fusion. Given knowledge of the environment such as the locations of walls and other obstacles, it is possible to create new inferences that exploit the constraint that objects cannot generally pass through walls. Hightower et al. [10] employ Bayesian filters, particle filters and Voronoi graphs [17], while Höllerer et al. [14] use a combination of spatial maps and accessibility graphs. New inferences can be made on the basis of this environmental knowledge.

3 Model Parameters

Now that we have described the theoretical framework behind Ubiquitous Tracking graphs, we will discuss in more detail how the parameters of our model have to be adjusted to make it useful for particular applications. Hitherto, we defined the relationships between different objects to be functions that take time as an input, and output both a spatial relationship and a set of *attributes* describing the "quality" of the given spatial relationship. These attributes acted as inputs to an *error function* that assisted in obtaining an inferred spatial relationship that is optimal in some sense. The spatial relationship itself, was simply modelled as a transformation matrix, although this representation is not generally sufficient to model all sensor measurements or spatial state.

3.1 Spatial Relationship Modelling

Representing geometrical relationships by a 4×4 homogeneous matrix has useful properties. However, there are other ways of expressing spatial relationships, for example a 3-vector and a quaternion can also represent the full range of affine transformations.

Many tracking sensors measure properties other than pose, such as 3D position (x, y, z), pixel location (x, y), or scalar distance ranges, d, which can only be related to pose indirectly when combining many such measurements. Other sensors measure different properties such as linear and angular velocities and accelerations. It is necessary to accommodate diverse representations of relationship state between objects, as well as the way in which different measurements affect the estimate of that state.

3.2 Attributes

Common attributes used to describe tracking quality are:

Latency describes the time (in seconds) between the actual measurement and the availability of its result to the rest of the system.

Update frequency measured in 1/s, indicates the rate at which trackers make their measurements.

Confidence value attributes are important for optical trackers that may misclassify video images and detect non-existent features. A well suited range would be $[0; 1]$, indicating the probability that the identification is correct.

Pose accuracy determination is the major problem when developing a tracker abstraction to integrate multiple trackers [6]. Even when simple Gaussian noise models are assumed, the evaluation of accuracy over multiple paths is non-trivial.

Monetary cost per measurement may be significant in the design of tracking setups. The tracking graph can incorporate all the trackers available on the market and then a suitable error function can be evaluated for the desired relationships between objects. The resulting optimal path consists of the trackers that should be installed.

Time to live is primarily relevant for temporarily static targets, indicating how long (in seconds) a relationship's value is likely to be valid.

Note that once a set of attributes has been defined, there may still be problems in obtaining actual values for given measurements or inferences. Firstly, the manufacturers of commercial trackers are often reluctant for their devices to output accuracy metrics. Secondly it may be difficult to calculate the attributes resulting from a hand-crafted inference. Although the cases of latency and update frequency are trivial, it becomes awkward to determine accuracy unless a very simple model is used. Even assuming Gaussian noise, the evaluation of absolute levels of accuracy, along a path incorporating multiple tracking devices, is non-trivial [6].

3.3 Error Functions

The error function is the core of the framework's path-finding algorithm. A whole path is taken as input, and the attributes along the path's edges' are extracted and an error value is computed, that is defined to be lower if the desired quality of the overall measurement is better. One simple example of an error function, e^t, that aims to tradeoff latency against update rate could be:

$$e^t := \sum_{q \in path} \text{lag}(q) + \frac{\lambda}{\text{rate}(q)} \tag{10}$$

The weighting of lag versus rate is determined by the variable λ.

As can be seen in this example, error functions exist that allow edgewise evaluation of paths. If such an error function is used, we can gain a speedup in the optimal path algorithm a well-known shortest path algorithms such as Dijkstra's algorithm [8]. The evaluated error functions can be used as edge weightings.

This strategy will not suceed for more complex attribute evaluations, as the error function cannot be expressed as the sum of the errors associated with each

edge along a path. For example, if the unscented transformation [19] is used to compute the covariance of tracking paths attributed with Gaussian covariance matrices, we must first compute the covariance of the whole path before the error function can be evaluated. Obviously, this leads to higher computational complexity, and other steps have to be taken to ensure that an unacceptable latency is not introduced. Some proposals are described in section 4.

4 Implementation and Optimisation

The formal framework described in the previous sections can easily be implemented within a software framework, as we plan to do within Studierstube [26] and DWARF [3]. Two possible optimisations can reduce the computational complexity within such an implementation: precomputing data flow graphs and encapsulating spatial hierarchies within supernodes.

4.1 Precomputing Data Flow Graphs

An implementation of Ubiquitous Tracking within Studierstube or DWARF can precompute a data flow graph of components that calculate the quantity required by an application. The construction of this *data flow graph* depends only on the infrequently changing structure Φ of the measured *spatial relationship graph* and the associated attributes from $\mathcal{A}$, not on the pose measurements themselves. Thus, the implementation can precompute the data flow graph at lengthy intervals, and then leave the real-time computation of pose data to the correctly configured components.

Components. Both Studierstube and DWARF are based on extensible, communicating components that can be easily arranged into data flow graphs [4]. The formalisms described previously can be mapped onto software components. Although the granularity and distribution of such components and the tightness of their coupling will be different in different implementations, their basic functionality remains the same.

The relation Ψ and attribution $\mathbf{Q}$ of all measured and inferred spatial relationships is constructed by performing several different kinds of operations: making measurements with trackers, querying a database, applying filters, and multiplying matrices along paths within the graph.

We can thus identify several different types of *components* that perform transformations on Ψ and its attribution $\mathbf{Q}$. Generally, these have the effect of adding new edges to the graph $G(\Psi)$ and new attributions to $\mathbf{Q}$.

Tracking components provide a stream of measurements, consisting of pose information and the quality of the measurement. This corresponds to q^m_{AB} from equation 5, where A is the tracker, and B is the tracked object.
Note that a single tracking component may well track several objects $B_{1..n}$; it then adds several functions $q^m_{AB_{1..n}}$ to $\mathbf{Q}$, extending Ψ.

Static components provide static measurements, e.g. from a database, consisting of pose information and the quality of the measurement. This also corresponds to q^m_{AB} from equation 5.
Again, a single static component may know of several objects $B_{1..n}$.

Extrapolation, interpolation and filtering components for a pair of objects A and B take some existing q^{in}_{AB} and calculate an output of q^{out}_{AB}.
Examples of q^{out} are q^e (extrapolation), q^i (interpolation), q^k (Kalman filter), given in equations 6 and 7.

Fusion components are more complex; at the very least, they take two different inputs $q^{in,1}_{AB}$ and $q^{in,2}_{AB}$ for a relationship between the same two objects A and B, and calculate an output of q^{out}_{AB}.
However, a more complex fusion component may take several different existing measurements into account.

Inference components perform inferences upon an existing subset of $\mathbf{Q}$, effectively performing transitive closure on Ψ. The simplest possible inference component computes q^{out}_{AC} from q^{in}_{AB} and q^{in}_{BC}.
In general, an inference component takes several different inputs $q^{in}_{A_1..A_mB_1..B_m}$ from $\mathbf{Q}$ and calculate one or more outputs $q^{out}_{A_1..A_nB_1..B_n}$, following the algorithm described in section 2.4.
Inference components use an error function e, to choose between alternative paths through the graph. An inference component may also calculate spatial relationships for more than one pair of objects.

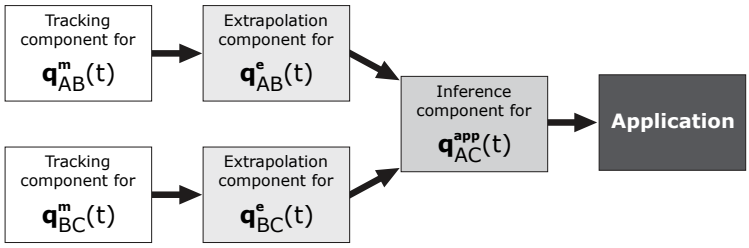

Fig. 5. Data flow graph for inference example of figure 4

Data Flow Graphs. To compute a desired function q^{app}_{AB}, an implementation constructs a tree of components, with data flowing from the leaves to the root. The leaves are trackers and databases encapsulating static measurements, the interior nodes are filters, interpolaters, extrapolators, inference components, and the output of the root node is the desired q^{app}_{AB}. An example data flow graph is shown in figure 5; this corresponds to path **P1** in the inference example shown in figure 4.

More generally, to compute several values simultaneously, the implementation will construct a directed graph, computing $q^{out}_{A_1..A_nB_1..B_n}$ at the sinks.

We now assume that the *attributes* $\mathcal{A}$ of a given ouput function q^{out} of a component change more slowly than the corresponding pose data (or do not change at all). Since the error function e depends only on the attributes, and not on the pose data itself (equation 8), a recomputation of an optimal data flow graph will not be necessary every time a position update must be delivered to the application. Consequently, the implementation may perform a computationally complex graph search in the background, while the components simultaneously compute the pose estimation in real time.

The way in which changes to the structure of the data flow graph is problematic. Whenever an attribute of any function within Ψ changes, the graph might need to be recomputed. An efficient algorithm for this is highly dependent on the types of attribute changes and the error function e.

4.2 Grouping Nodes

In many applications precise tracking is not important when the objects being queried are far away from one another. For example, when standing in front of a building it may be interesting to see which people are inside (and maybe where in the building they are located) but their exact position is not important. This is similar to the rendering of complex 3D scenes using *level of detail* [18] hierarchies. In the Ubiquitous Tracking scenario the same technique to reduce complexity can be applied by grouping of nodes in the tracking graph.

In a large and complex graph there will be two circumstances in which groups of nodes can be represented by a single *supernode*. Firstly, groups of nodes which are only statically related to one another need only be represented as a single node with respect to the rest of the graph. Secondly, it is reasonable to suppose that graph searches will tend to concentrate on special highly connected clusters of nodes known as *cliques* (if fully connected) or *near-cliques* (when highly connected but not fully connected). These clusters correspond to areas with particular spatial significance such as a room, or mobile body-centred space. These spaces we call *locales* [2].

In the top half of figure 6 two people can be seen with a variety of tracking hardware, while in the lower half graphs form an abstract representation of the spatial relationships. The shaded regions encompass objects with common features to their spatial relationships. The person on the left hand side is equipped with a camera, B, and a fiducial marker, A, rigidly attached to the head and hence statically related both to one another and also to the eyes, C and D. Consequently, these objects can be represented by a single supernode S_1. Similarly, the right hand person has the rigidly attached objects G, H, I and J represented as the supernode S_2; while in the rest of the room the fixed fiducial marker E and the InterSense[1] transmitter, F, form the supernode S_3. Fiducial marker, K, can be considered part of a mobile body-centred locale, S_4, centred on the right hand person. The whole room can be considered to be yet another locale, S_5.

[1] http://www.isense.com/

It is highly likely that a query for the spatial relationship between the objects G and K (camera and marker) will find K within the same locality as G, and consequently either within the same locale or a nearby one. The query can be optimised by directing the search locally before looking further afield. Queries on objects which are not local to one another can exploit the locality of supernodes to distribute the search more effectively.

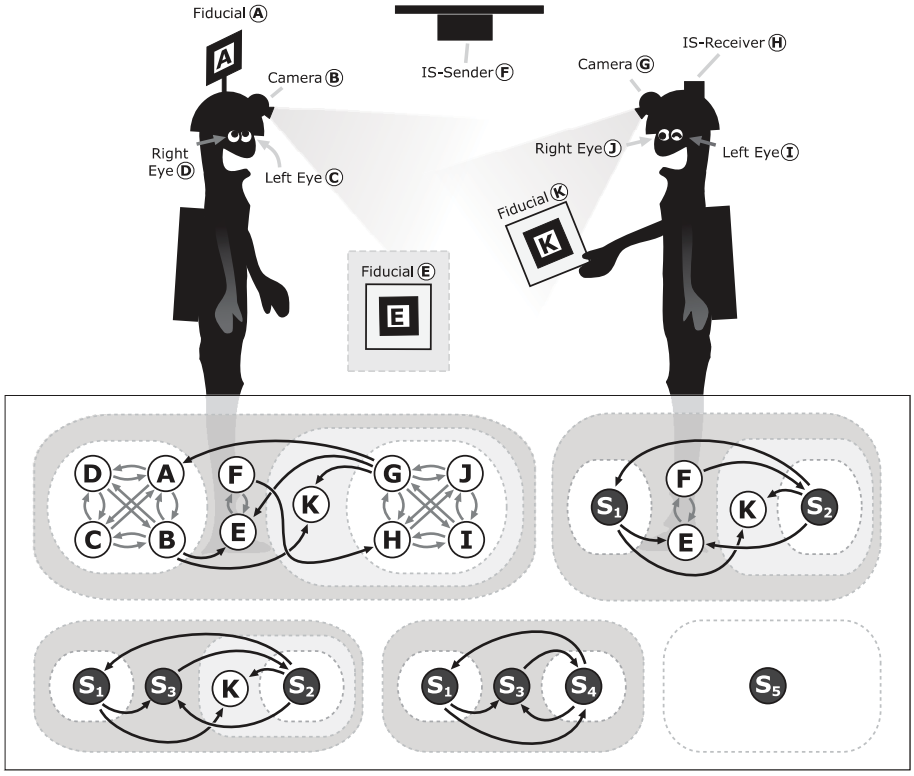

Fig. 6. People and objects in environment partitioned to form supernodes

5 Examples of the Framework

The decomposition of existing tracking setups using the framework, as it has been described thus far, is a useful exercise in assessing the power and generality of this approach.

Dynamically shared optical tracking as implemented by Ledermann et al. [16]. can be reproduced using the cameras and markers in figure 6. Figure 7 shows

the pose of mobile cameras, B and G, determined relative to the fixed marker E. Marker K need only be visible to camera G in order to calculate its position relative to the camera B. The rest of the figure shows how this setup maps into a graph, in which the inferred spatial relationship $q^{app}_{S_1K}(t)$ can be determined by evaluating and concatenating the relationships (including an inversion of the inference $q^{e}_{S_2E}$ we call $q^{inv}_{ES_2}$) along the path:

$$S_1 \xrightarrow{q^{e}_{S_1E}} E \xrightarrow{q^{inv}_{ES_2}} S_2 \xrightarrow{q^{e}_{S_2K}} K$$

Fig. 7. Shared tracking setup

The tracking setup can be extended to include the InterSense tracker which should provide further tracking redundancy, increasing the range of possible inferences.

Multi Camera System. The Advanced Real Time (A.R.T.)[2] optical tracker (ART Track) in figure 8 uses three cameras (A, B, C) to observe a target (supernode, S_{marker}) built from retroreflective markers (D, E, F, G). The relationships between camera A and each individual marker is illustrated in more detail to the right of the main graph. The markers are attached to a rigid body forming a single composite target. The measurement made by the camera is not in the form of a pose or even a 3D point, but a pixel location (x, y), which means that for each camera the markers are constrained to lie on rays passing through the focal points of the cameras and the real 3D positions of the markers. There is enough information based on the geometry of the cameras and the relative positions of the markers on the target to make an overall estimate of the target pose.

A complete treatment of this setup is beyond the scope of the paper, however, it should be possible to model this aggregation as a series of filters. First the raw pixel measurements are used to generate inferences of the locations of each marker defined as rays in the coordinate frame of the camera, $(\lambda x, \lambda y, \lambda z)$. A further filter or "inference aggregator" takes all these inferences as well as knowledge of the static relationships between the cameras and markers respectively to generate an estimate of the pose of the target S_t as well as appropriate attributes based on the noise in the pixel measurements, and timing errors.

[2] http://www.ar-tracking.de/

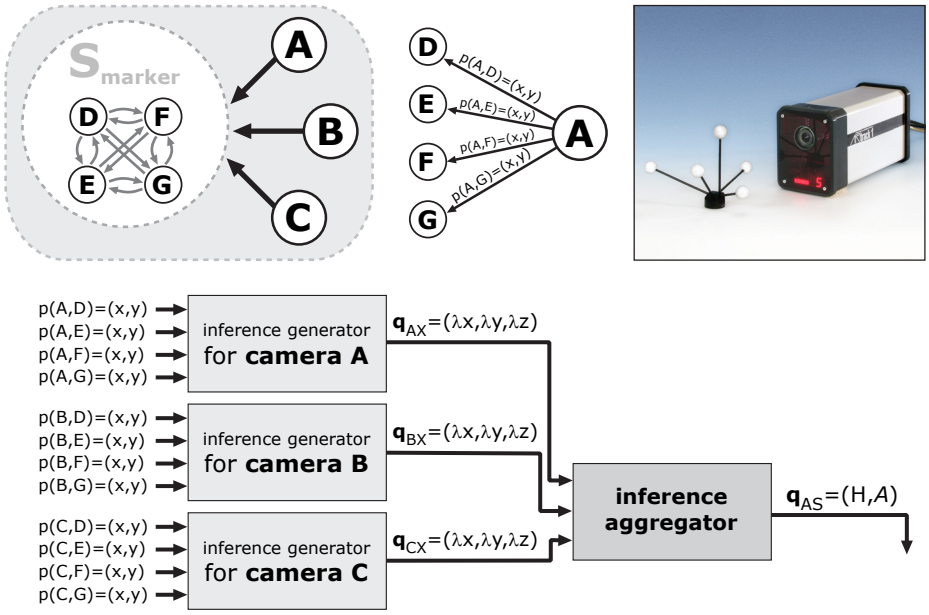

Fig. 8. ART optical tracker decomposition

Occluded Markers. One common medical Augmented Reality scenario involves a patient being tracked optically using retroreflective markers registered preoperatively with three-dimensional images [22]. During system setup the patient is visible to the tracking system, but during the operation the markers attached to the patient are occluded by sterile tissue. However, markers fixed to the operating table are visible at all times. If the patient is assumed not to move relative to the table, they can be tracked continuously.

In order to augment the surgeon's view through a display D of structures within the patient's body B a query $q^{app}_{DB}(t)$ is made. A suitable error function should always select a direct measurement when available and fall back on indirect measurements otherwise.

6 Future Work

Simulation. To prove the applicability of the framework in a large building-wide Ubiquitous Tracking setup, we will have to design and implement a simulation environment that allows us to generate synthetic data of virtual trackers observing objects roaming through the environment. The data can serve as input to a reference implementation of the framework. We will then conduct run-time measurements to evaluate the efficiency of the framework, and accuracy analysis to refine the design of attributes, error functions and the state space of objects.

Implementation. As described in section 4, both Studierstube and DWARF provide a suitable basis for the implementation of the ubiquitous tracking concepts. We plan to pursue both implementation tracks in parallel, while maintaining interoperability between DWARF and Studierstube.

An implementation in Studierstube would generate the data flow graphs in the form of OpenTracker networks [24], which are then evaluated within the Studierstube run-time environment.

An implementation in DWARF would use DWARF services as the components in the data flow graph, and combine them dynamically, based on their attributes, using the DWARF middleware [20].

Auto-calibration. In real environments, any form of infrastructure is vulnerable — subject to vandalism, repair and manipulation by persons unknown. It is not practical to regularly recalibrate every device, and consequently the parts of our model which are generally static will become increasingly inaccurate unless it is adaptive. If the attributes of the static spatial relationships include a level of uncertainty that increases slowly with time, then probabilistic inferences will take into account the possibility that the model has changed. Evidence from sensors that are considered reliable can be used to effectively recalibrate the model and reduce the level of uncertainty.

Evaluation. Once the framework has been used to implement real scenarios, we hope to gather information and expertise in order to propose a suitable set of standard attributes. These can then be used to describe new tracking technologies and filtering schemes.

7 Conclusion

In this paper, we presented a theoretical framework to describe all current tracking setups in a standard way. The framework consists of relations modelled as graphs, which allow application developers to satisfy tracking demands by defining: *state spaces*, *attributes* and *error functions*. The design of state spaces encapsulates the way in which sensor measurements are related to the representation of spatial representationships. Attributes characterise sensor measurements, and are operated on by error functions that are designed to discriminate between different solutions to a spatial query on a per-application basis.

The framework is a first step towards a systematic implementation of Ubiquitous Tracking concepts and hopefully serves as a basis for partial standardisation of complex tracker setups. Modelling complex setups in a unified mathematical framework brings up new issues commonly overlooked in real-world applications, such as the role of time in processing sensor information. We hope to have started a more formal approach towards the ubiquity of Augmented Reality.

Acknowledgements

The authors would like thank Gerhard Reitmayr for his valuable contributions, and inspiring work on OpenTracker. This work was partly sponsored by the *High-Tech-Offensive Bayern* of the Bayerische Staatskanzlei and by the Austrian Science Foundation *FWF* under contracts no. P14470 and Y193, and Vienna University of Technology by *Forschungsinfrastrukturvorhaben TUWP16/2002*.

References

1. M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, A. Ward, and A. Hopper, *Implementing a Sentient Computing System*, IEEE Computer, 34(8), 2001, pp. 50–56.
2. J. W. Barrus, R. C. Waters, and D. B. Anderson, *Locales: Supporting Large Multi-User Virtual Environments*, IEEE Computer Graphics and Applications, 16(6), November 1996, pp. 50–57.
3. M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Riss, C. Sandor, and M. Wagner, *Design of a Component-Based Augmented Reality Framework*, in Proceedings of the 2nd IEEE and ACM International Symposium on Augmented Reality (ISAR 2001), New York, NY, October 2001, IEEE Computer Society.
4. M. Bauer, O. Hilliges, A. MacWilliams, C. Sandor, M. Wagner, J. Newman, G. Reitmayr, T. Fahmy, G. Klinker, T. Pintaric, and D. Schmalstieg, *Integrating Studierstube and DWARF*, in Int. Workshop on Software Technology for Augmented Reality Systems (STARS 2003), 2003.
5. B. Brumitt, J. Krumm, B. Meyers, and S. Shafer, *Ubiquitous Computing and the Role of Geometry*, IEEE Personal Communications, October 2000, pp. 41–43.
6. E. M. Coelho and B. MacIntyre, *High-level Tracker Abstractions for Augmented Reality System Design*, in International Workshop on Software Technology for Augmented Reality Systems, October 2003, pp. 12–15.
7. D. Curtis, D. Mizell, P. Gruenbaum, and A. Janin, *Several Devils in the Details: Making an AR App Work in the Airplane Factory*, in First IEEE Workshop on Augmented Reality, November 1998.
8. E. W. Dijkstra, *A note on two problems in connexion with graphs*, Numerische Mathematik, 1, 1959, pp. 269–271.
9. S. Feiner, B. MacIntyre, T. Höllerer, and T. Webster, *A touring machine: Prototyping 3D mobile augmented reality systems for exploring the urban environment*, in Proc. of ISWC 1997, October 1997.
10. D. Fox, J. Hightower, H. Kautz, L. Liao, and D. Patterson, *Bayesian Techniques for Location Estimation*, in Proceedings of The 2003 Workshop on Location-Aware Computing, October 2003.
11. A. Harter, A. Hopper, P. Steggles, A. Ward, and P. Webster, *The Anatomy of a Context-Aware Application*, in Mobile Computing and Networking, 1999, pp. 59–68.
12. J. Hightower and G. Borriello, *Real-Time Error in Location Modeling for Ubiquitous Computing*, in Location Modeling for Ubiquitous Computing - Ubicomp 2001 Workshop Proceedings, Atlanta, GA, September 2001, pp. 21–27.

13. J. Hightower, B. Brumitt, and G. Borriello, *The Location Stack: A Layered Model for Location in Ubiquitous Computing*, in Proceedings of the 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA 2002), Callicoon, NY, June 2002, pp. 22–28.
14. T. Höllerer, D. Hallaway, N. Tinna, and S. Feiner, *Steps toward accommodating variable position tracking accuracy in a mobile augmented reality system*, in 2nd International Workshop on Artificial Intelligence in Mobile Systems (AIMS '01), 2001, pp. 31–37.
15. H. Kaufmann and D. Schmalstieg, *Mathematics And Geometry Education With Collaborative Augmented Reality*, Computers & Graphics, 27(3), 2003.
16. F. Ledermann, G. Reitmayr, and D. Schmalstieg, *Dynamically Shared Optical Tracking*, in 1st Int'l Augmented Reality Toolkit Workshop, 2002.
17. L. Liao, D. Fox, J. Hightower, H. Kautz, and D. Schulz, *Voronoi Tracking: Location Estimation using Sparse and Noisy Sensor Data*, in Proc. of Int. Conf. on Intelligent Robots and Systems (IROS), IEEE/RSJ, 2003.
18. D. Luebke, M. Reddy, Jonathan, D. Cohen, A. Varshney, B. Watson, and R. Huebner, *Level of Detail for 3D Graphics*, Morgan Kaufmann, July 2002.
19. B. MacIntyre, E. M. Coelho, and S. Julier, *Estimating and Adapting to Registration Errors in Augmented Reality Systems*, in IEEE Virtual Reality Conference 2002 (VR 2002), Orlando, Florida, March 2002.
20. A. MacWilliams, T. Reicher, and B. Brügge, *Decentralized Coordination of Distributed Interdependent Services*, in IEEE Distributed Systems Online – Middleware Work in Progress Papers, Rio de Janeiro, Brazil, June 2003.
21. J. Newman, D. Ingram, and A. Hopper, *Augmented Reality in a Wide Area Sentient Environment*, in Proc. of IEEE and ACM Int. Symp. on Augmented Reality (ISAR 2001), New York, NY, October 2001, pp. 77–86.
22. L.-P. Nolte and F. Langlotz, *Intraoperative Navigationssysteme*, Trauma und Berufskrankheit, 1(2), 1999, pp. 108–115.
23. W. Piekarski and B. H. Thomas, *Tinmith-evo5 A Software Architecture for Supporting Research Into Outdoor Augmented Reality Environments*, tech. rep., Wearable Computer Laboratory, University of South Australia, December 2001.
24. 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.
25. D. Schmalstieg, A. Fuhrmann, G. Hesina, Z. Szalavari, L. M. Encarnao, M. Gervautz, and W. Purgathofer, *The Studierstube Augmented Reality Project*, PRESENCE - Teleoperators and Virtual Environments, 11(1), pp. 32–45.
26. D. Schmalstieg, G. Reitmayr, and G. Hesina, *Distributed Applications for Collaborative Three-Dimensional Workspaces*, in IEEE Virtual Reality Conference 2002, Orlando, Florida, March 2002, IEEE Computer Society, pp. 59–66.
27. M. Weiser, *Hot Topics: Ubiquitous Computing*, IEEE Computer, October 1993.