

# Automatic Configuration of Pervasive Sensor Networks for Augmented Reality

*The ubiquitous tracking (Ubitrack) approach uses spatial relationship graphs and patterns to support a distributed software architecture for augmented reality (AR) systems in which clients can produce, transform, transmit, and consume tracking data.*

Most virtual reality (VR) and augmented reality (AR) systems are constrained to laboratories by the limited range of a single, expensive six degrees of freedom (6DOF) tracking system. Diverse low-cost electronics and sensors deployed in a ubiquitous manner could allow AR applications to cast off their shackles and roam throughout much larger environments. Ultimately, we can best realize this goal, not by extending the range of a single tracker, but by dynamically and automatically incorporating the heterogeneous sensors we anticipate will pervade the environment in the future.

AR and VR applications require a high update rate and low latency, so communicating tracking information requires a fast, highly optimized system with a direct network connection from sources of tracking data (sensors) to sinks (applications). In contrast, information involving the availability

of the user is or which source should be connected to which sink) occurs at much lower frequencies but requires the oversight of a complex problem domain potentially involving numerous entities.

In general terms, a middleware architecture for location sensing in distributed and heterogeneous environments should support several key requirements:

- device abstraction enabling portability of applications between families of similar devices;
- network transparency allowing access to device data independent of the device's physical connection;
- flexible processing of data from multiple devices delivered in a form that the application can use (for example by filtering, fusion, calibration, or registration);
- architectural support for established transformation, sensor fusion, and calibration algorithms;
- efficient processing and transport of tracking data for time-critical AR applications; and
- dynamic reconfiguration of the sensor data flow.

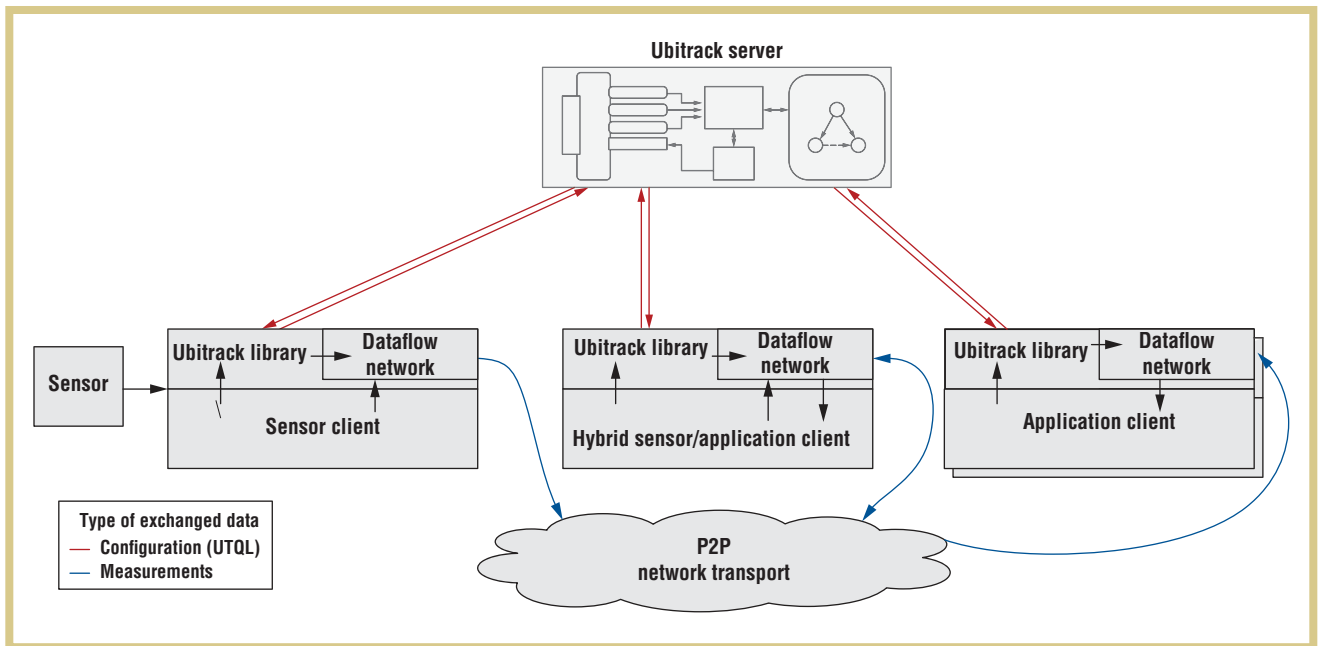
In AR and VR applications, dataflow networks have proven useful for efficient device abstraction

Daniel Pustka, Manuel Huber,  
Christian Waechter,  
Florian Echtler, Peter Keitler,  
and Gudrun Klinker  
*Technical University of Munich*

Joseph Newman  
*University of Cambridge*

Dieter Schmalstieg  
*Graz University of Technology*

of sensors, objects, and users, and the associated topological considerations (for example, where



**Figure 1. Ubitrack architecture overview.** The Ubitrack server coordinates clients that directly exchange sensor data with low latency and apply transformation, calibration, and sensor fusion algorithms as instructed by the server.

and network transparency (see the “Related Work in Ubiquitous Tracking” sidebar). However, to deliver a continuous stream of tracking data, these networks must be manually configured for every tracking situation.

The ubiquitous computing (ubicom) community has developed several approaches that incorporate dynamic and heterogeneous wide-area sensor networks (see the sidebar). However, these approaches primarily consider low-frequency data. Our Ubiquitous Tracking (Ubitrack) system combines the performance advantage of AR and VR dataflow networks with the flexibility and more sophisticated reasoning typically found in ubicom systems to support distributed access to an array of sensing technologies.

## System Overview

We propose a two-layered approach to distributed tracking for AR/VR applications: a *configuration layer*, responsible for low-frequency events, sets up the *runtime layer*, which handles the high-frequency events. The configuration layer stores structural information

concerning the physical environment and technical infrastructure in a spatial relationship graph (SRG) and mines it for the configuration information the runtime layer needs. The runtime layer polls the sensors and transforms, fuses, and delivers the sensor data through a distributed dataflow network.

Figure 1 shows the architecture’s basic components and the most relevant communication channels. The design’s core idea is to separate the configuration of data producers and consumers from the actual production and consumption of sensor data. This separation permits the relatively resource-intensive coordination process to run on the server, while clients communicate directly with one another according to the low-latency requirements of AR applications. For performance reasons, this approach uses a centrally coordinated peer-to-peer architecture rather than a pure client-server solution.

### Ubitrack Server

The Ubitrack server is the system’s central component, maintaining a database of all coordinate frames, sensors,

and tracked objects as an SRG. The SRG, although initially empty, contains the aggregated spatial relationships specified by clients and thus contains complete knowledge of the tracking infrastructure’s topology. It doesn’t, however, contain actual sensor measurements.

Clients can query the server about parts of the SRG, including spatial relationships that can’t be measured directly but can only be inferred using spatial relationship patterns.<sup>1</sup> The server continuously matches the SRG against registered client queries. If it finds a match, it generates a data-flow network description, which it sends to the client. During this process, the server can also order a client to construct a dataflow network in support of other clients by processing tracking data and transmitting them over the network.

### Ubitrack Client

Ubitrack system clients can be sensors, output devices, and other human-computer interaction components as well as mixed forms, such as software agents representing virtual characters

## Related Work in Ubiquitous Tracking

The Ubitrack approach incorporates results from several research areas.

### Augmented and Virtual Reality

Distributed VR applications have traditionally relied on application-level rather than device-level distribution of events, for example, through distributed scene graphs.<sup>1,2</sup> This might have been a more straightforward solution for connecting distributed VR sites with static physical configuration, but it doesn't address the need for dynamic reconfiguration at runtime.

AR applications, especially those for mobile users, require coverage of larger physical areas as well as greater flexibility in the type of supported devices. Specifically, fusion of heterogeneous sensors is used both to improve tracking quality and cover larger areas.<sup>3</sup> Gudrun Klinker and her colleagues first postulated distributed tracking concepts necessary to propel AR beyond the confines of the laboratory into serious industrial settings.<sup>4</sup>

In AR and VR applications, dataflow networks processing high-frequency streams of sensor data have proven to be a useful approach for device abstraction. Two prominent examples are the Virtual Reality Peripheral Network (VRPN)<sup>5</sup> and OpenTracker.<sup>6</sup> The Ubitrack framework automatically computes suitable dataflow graphs given an abstract semantic representation of the world in form of an spatial relationship graph (SRG).

In recent years, researchers have made significant progress in the development of optical natural feature-based self-localization systems,<sup>7</sup> which don't rely on technical infrastructure. Therefore, the use of middleware might seem unnecessary for such systems. However, to realize a meaningful pervasive AR application, common coordinate frames must be established and positions of

objects and features exchanged. In addition, access to other sensors can improve robustness and accuracy.

### Ubiquitous Computing

The use of heterogeneous sensors and context-aware computing is much more common in the ubicomp area than in AR. Several ubicomp systems deal with wide-area sensor networks and federated sensor data models, such as Nexus,<sup>8</sup> QoSDream,<sup>9</sup> and the context toolkit.<sup>10</sup> However, these approaches primarily consider low-bandwidth communication. Therefore, they aren't directly applicable to AR's real-time and low-latency requirements.

Location models for ubicomp are frequently described in terms of Hightower's location stack.<sup>11</sup> The Ubitrack system mostly covers the functionality of the measurement, fusion, and arrangement layers. However, because we separate configuration and runtime layers, a straightforward mapping of individual Ubitrack components onto the location stack layers is impossible.

The EasyLiving Geometric Model used graph models to represent different coordinate systems.<sup>12</sup> However, because it used undirected (rather than directed) graph edges and a simple graph search (rather than spatial relationship patterns), this approach is less powerful than the Ubitrack framework.

### Computer Vision and Robotics

The SRG concept was heavily inspired by the informal coordinate frame drawings frequently found in robotics papers.<sup>13</sup> Furthermore, robotics has developed a rich collection of algorithms for alignment and fusion in multisensor systems. Our development of the spatial relationship pattern framework was driven by the desire to make these algorithms available to mobile AR and VR setups, which our first Ubitrack system didn't support.

that both react to the environment and provide information about their location in space.

To keep the interface for application programmers minimal, the client-side network communication is encapsulated in the Ubitrack client library. It supports interaction with the Ubitrack system at various degrees of complexity, ranging from simple "Where is object A?" queries to persistent requests for tracking data of all objects matching a given predicate with the additional specification of desired tracking quality and base coordinate frames.

### Application

A developer writes an application conforming to interfaces in the Ubitrack client library. Using this interface, the application becomes part of the dataflow network. The application retrieves information using callback nodes in the data flow and can pass information into the data flow using call-forward nodes.

### Dataflow Network

The dataflow network is formed by jointly executed processes run by the clients. It retrieves, processes, transmits, and delivers the actual sensor data. The data flow's reconfiguration is

triggered by the Ubitrack server, but it's interpreted and executed by the Ubitrack client library.

### Spatial Relationship Graphs

To enable the automated analysis and synthesis of complex tracking scenarios, a formal description of the situation is necessary. Inspired by the informal coordinate frame drawings frequently found in robotics papers (such as that by R.Y. Tsai and R.K. Lenz<sup>2</sup>), we proposed the use of SRGs.<sup>3</sup> The graph's nodes represent the coordinate frames of real or virtual objects, whereas the directed edges represent the spatial relationships

## Semantic Web and Graph Rewriting

An interesting approach to the inference of context from sensor data is the use of Semantic Web technologies (ontologies), such as in the Cobra project.<sup>14</sup> This kind of system description has some similarities to our SRG approach in that ontologies can evaluate complex relationships between entities.

In the model-driven engineering field, researchers have investigated the automatic analysis of model diagrams. This has resulted in graph transformation systems, such as Progres,<sup>15</sup> which are used to transform, simulate, and verify UML models and similar diagrams. Although the graph transformation rules used there are more complex than spatial relationship patterns, our system implementation profits from the results of this research.

## REFERENCES

1. E. Frecon and M. Stenius, "Dive: A Scaleable Network Architecture for Distributed Virtual Environments," *Distributed Systems Eng. J.*, vol. 5, no. 3, 1998, pp. 91–100.
2. B. MacIntyre and S. Feiner, "A Distributed 3D Graphics Library," *Proc. ACM Siggraph*, ACM Press, 1998, pp. 361–370.
3. D. Hallaway, T. Hoellerer, and S. Feiner, "Bridging the Gaps: Hybrid Tracking for Adaptive Mobile Augmented Reality," *Applied Artificial Intelligence*, vol. 25, no. 5, 2004, pp. 477–500.
4. G. Klinker, T. Reicher, and B. Bruegge, "Distributed User Tracking Concepts for Augmented Reality Applications," *Proc. IEEE Int'l Symp. Augmented Reality (ISAR)*, IEEE Press, 2000, pp. 37–44.
5. R.M. Taylor et al., "VRPN: A Device-Independent, Network-Transparent VR Peripheral System," *Proc. ACM Symp. Virtual Reality Software and Technology*, ACM Press, 2001, pp. 55–61.
6. G. Reitmayr and D. Schmalstieg, "Opentracker: A Flexible Software Design for Three-Dimensional Interaction," *Virtual Reality*, vol. 9, no. 1, 2005, pp. 79–92.
7. G. Klein and D. Murray, "Parallel Tracking and Mapping for Small AR Workspaces," *Proc. 6th Int'l Symp. Mixed and Augmented Reality (ISMAR)*, IEEE CS Press, 2007, pp. 1–10.
8. M. Bauer and K. Rothermel, "Towards the Observation of Spatial Events in Distributed Location-Aware Systems," *Proc. 22nd Int'l Conf. Distributed Computing Systems Workshops*, IEEE CS Press, 2002, pp. 581–582.
9. G. Coulouris, *The QOSDream Project*, tech. report, Laboratory for Comm. Eng., Univ. of Cambridge, 2002.
10. A.K. Dey, "Providing Architectural Support for Building Context-Aware Applications," doctoral thesis, College of Computing, Georgia Inst. of Technology, 2000.
11. J. Hightower, B. Brumitt, and G. Borriello, "The Location Stack: A Layered Model for Location in Ubiquitous Computing," *Proc. 4th IEEE Workshop Mobile Computing Systems and Applications (WMCSA)*, IEEE CS Press, 2002, pp. 22–28.
12. B. Brumitt and S. Shafer, "Better Living Through Geometry," *Personal Ubiquitous Computing*, vol. 5, no. 1, 2001, pp. 42–45.
13. R.Y. Tsai and R.K. Lenz, "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration," *IEEE Trans. Robotics and Automation*, vol. 5, no. 3, 1989, pp. 345–358.
14. H. Chen, T. Finin, and A. Joshi, "An Ontology for Context-Aware Pervasive Computing Environments," *The Knowledge Eng. Rev.*, vol. 18, no. 3, 2003, pp. 197–207.
15. A. Schürr, A.J. Winter, and A. Zündorf, "The PROGRES Approach: Language and Environment," *Handbook of Graph Grammars and Computing by Graph Transformation: Vol. 2: Applications, Languages, and Tools*, G. Rozenberg, ed., World Scientific Publishing, 1999, pp. 487–550.

between these objects. Thus, in the case of two 3D coordinate frames, an edge might describe a transformation with 6DOF, three for position and three for orientation. This is the most common case for AR tracking setups, although other edge types are equally important. The graph contains a separate edge for each relationship. It can thus contain multiple edges between the same pair of nodes—for example, if multiple independent trackers are tracking the same objects. Furthermore, both nodes and edges have a set of attributes describing important characteristics, such as an object's name or the measurement data

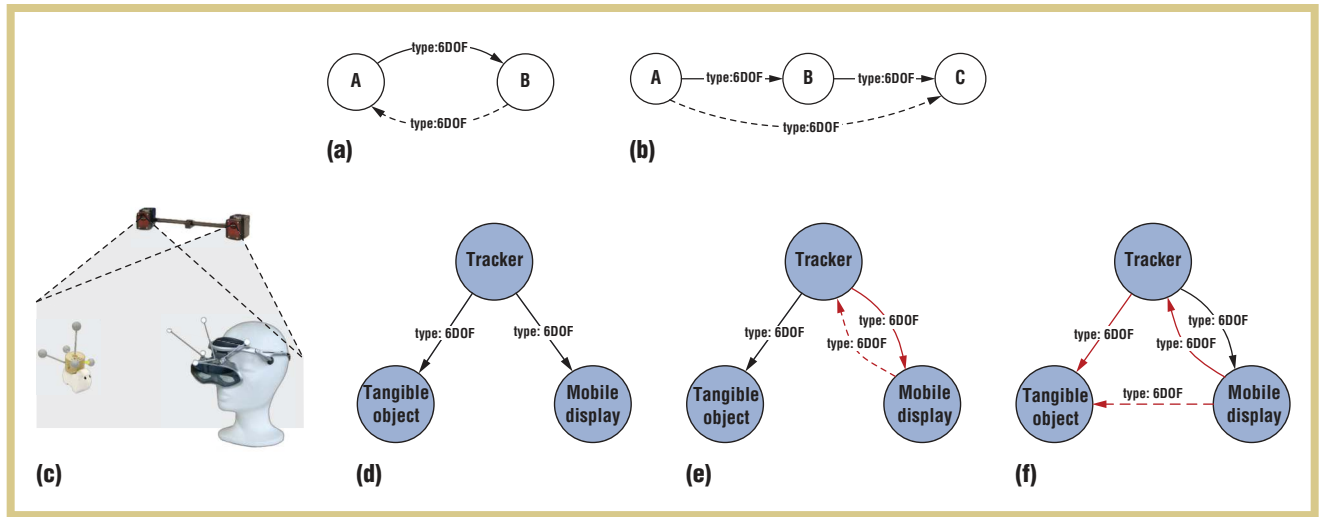
type of an edge (6DOF pose, 3D position, and so on).

Although the SRG describes the layout of a tracking environment in detail, an application doesn't immediately benefit from access to the SRG. The application needs the relevant data from the sensors rather than data describing the sensors. To obtain this data, the Ubitrack server provides the application with an abstract dataflow graph (DFG), which the application instantiates into a concrete dataflow network. The DFG is a directed graph with nodes representing computational units (such as matrix inversion or

multiplication) and edges representing the flow of tracking data through this graph. Sources in a DFG generally represent sources of tracking data, whereas sinks mostly correspond to interfaces to other parts of the application.

## Spatial Relationship Patterns

We use spatial relationship patterns to infer suitable DFGs from an SRG representation. A spatial relationship pattern identifies parts of the overall SRG for which a known algorithm exists. The corresponding dataflow component executes the algorithm and provides the result, again as an edge in the SRG.



**Figure 2.** The basic spatial relationship patterns (a) inversion and (b) concatenation form the reflective and transitive closure of a spatial relationship graph, as shown in the examples: (c) example scenario, (d) spatial relationship graph (SRG), (e) inversion application, and (f) concatenation application.

We apply this technique recursively to further identify solvable subproblems until a solution for all required data-flow elements is found.

Such patterns define the signature of an algorithm for solving a specific problem in a tracking setup. These algorithms have as input a set of measurement edges, defined by the problem, and return a set of measurement edges that is part of the solution. Spatial relationship patterns don't merely define the type of arguments and return values; they also impose restrictions on the geometric relationship between them. Hence, a pattern has two types of edges: input edges required in the SRG before a pattern can be applied (drawn as solid lines) and output edges added by a pattern (drawn as dashed lines). Inputs edges can be further restricted by specifying predicates over the attributes of the SRG edges.

Figure 2 shows the two basic spatial relationship patterns, which sufficiently describe many runtime setups (after calibration). The basic patterns create an SRG's transitive reflexive closure:

- **Inversion.** Edges in an SRG are directed and can be inverted, depending on their type. If the transformation is

represented by a matrix, this simply gives the inverse matrix.

- **Concatenation.** The most common way to compute the transformation on an unknown edge is by concatenation of subsequent edges, that is, by multiplying the transformations.

Consider a simple lab-based AR application where a single high-precision tracker tracks both a mobile display and a tangible object to be augmented (Figures 2c and 2d). In this case, an inversion pattern (Figure 2e) would be used to change the reference coordinate frame from the tracker to the display, followed by a concatenation (Figure 2f) that computes the object's location relative to the display. (You can find more complex patterns that describe sensor fusion and calibration methods from robotics and computer vision in earlier research.<sup>1)</sup>)

### Sensor Synchronization

Most algorithms with multiple input edges, such as concatenation, require that measurements on all their inputs are synchronized to produce correct results. Our system ensures this by choosing one sensor as the time reference. The system synchronizes the

measurements of other sensors by inserting interpolation components that compute results for the reference sensor's time stamps.

In the pattern framework, we attribute edges with the type of interface that's used for communication. Push edges provide measurements whenever they're available, whereas pull edges can be queried for a given time stamp. For correct synchronization, a pattern can have at most one push input that provides the reference timing. When combining multiple push sensors, the other inputs must be converted from push to pull by inserting an interpolation pattern.

Figure 3 shows the different variations of the concatenation pattern resulting from this push-pull expansion, together with the interpolation pattern.

### Using Patterns to Create Dataflow Networks

The procedure that results in a dataflow network suitable for an application client's needs can roughly be divided into several steps.

### Client Request

The protocol we used to exchange information between servers and clients is



an XML dialect called Ubitrack Query Language (UTQL). The client starts the communication by sending a request to the server, which contains descriptions of sensors, processing capabilities, and queries for particular spatial relationships. All this is expressed in the form of spatial relationship patterns, which have only outputs (sensors), only inputs (queries), or both (processing patterns).

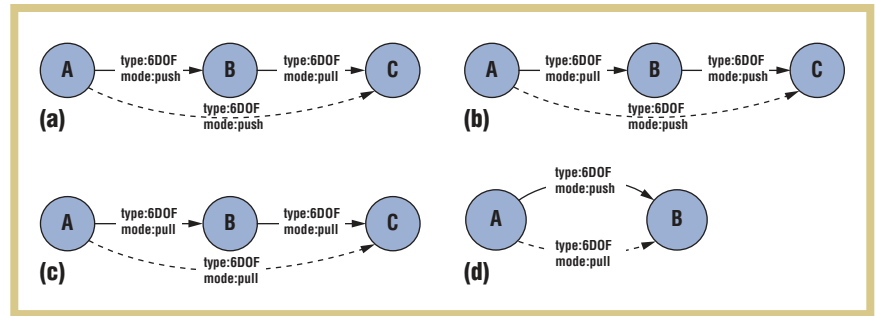
Depending on the kind of patterns transmitted, we can distinguish between sensor, application, and processing clients. However, mixed forms are also frequent, such as the mobile setup presented later. Because each client can transmit a different set of patterns, the system supports clients with different processing capabilities. Clients can add or remove patterns from the server at any time, which triggers a recomputation of the DFG.

### Pattern Application

The server first creates an initial SRG by adding all the clients' sensor patterns. On this SRG, it systematically applies processing patterns until solutions to all requests are found (or the search fails). This search for patterns can be split into a detection and control problem. Detection of all instances of a given pattern (subgraph isomorphism) is NP-complete in the general case, but we can apply some simple heuristics and cut-off criteria to quickly yield the relevant results. For example, new edges are only added if they don't provide the same information as existing edges, only derived in a different order. The control mechanism is bootstrapped by the detection and aims to apply patterns systematically toward a given goal (a client's request). The currently implemented strategies are simple and apply a combination of known relevant patterns.

### Dataflow Construction

Constructing the DFG from a sequence of located pattern instances is straightforward. All edges in the initial SRG are associated with a tracking component or some other service (such as a



**Figure 3. Pattern extensions for sensor synchronization: (a) push-pull, (b) pull-push, (c) pull, and (d) interpolation. Push inputs provide the time reference to which the pull inputs are synchronized. Push edges can be converted to pull by instantiating an interpolation pattern.**

database) in the DFG that provides the initially unprocessed measurements. Whenever a pattern is applied to the SRG, it creates a new algorithmic component in the DFG. This component provides new outputs, which are associated with the new SRG edges, while the component's inputs are connected to the components associated with the input edges of the patterns. This is repeated for all pattern applications until the desired measurement is available at the end of the DFG.

To support network transport of measurements with minimal latency, the tracking data must be sent directly from one client to another. Therefore, the server splits a global dataflow description into client-specific parts and integrates network sources and sinks. This results in distributed DFGs where several clients collaborate to satisfy a specific client's request.

### Dataflow Instantiation

When the server has computed a suitable DFG in response to a client's query, it sends UTQL messages to one or more clients, requesting that they instantiate and connect the required dataflow components corresponding to the applied patterns.

### Dynamic Reconfiguration at Runtime

The methods we've described provide the necessary formal framework to

derive optimal dataflow configurations for AR applications, given a particular sensor configuration. For configuring static sensor arrangements, this can help in setting up an AR system because specifying an SRG is easier and less error-prone than manually configuring a dataflow configuration. Using a graphical SRG editor,<sup>4</sup> members of our group implemented complex multi-sensor calibration tasks within a few minutes.

In a ubicomp scenario where mobile systems move between tracking areas covered by different sensors, mechanisms are necessary to reconfigure the SRG based on sensor availability or region containment. These reconfigurations are generally triggered by analysis of the actual sensor data. As in our framework, the server coordinates the clients, but it doesn't receive actual tracking data. Such functionality must be implemented by special clients.

### Transition Between Locales

To ensure system scalability, we limit queries for available objects, such as persons, sensors, or fiducials, to a reasonable area of interest, a *locale*. In the SRG formalism, we treat locales as nodes. Containment in a locale is expressed as an edge of type `inLocale`, which is drawn from the locale node to the object in question. This lets us model queries for objects as spatial relationship

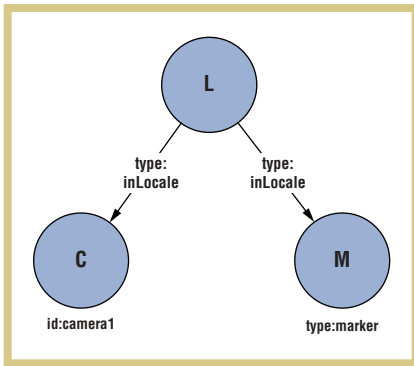


Figure 4. Query pattern resulting in all objects of type marker that are contained in the same locale as the camera1 node.

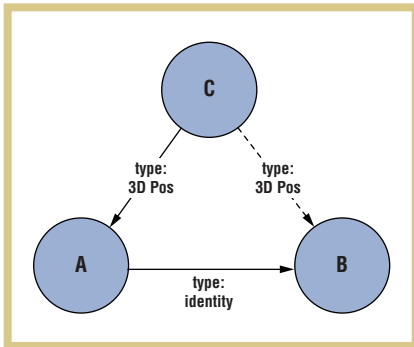


Figure 5. The Measurement identity pattern for 3D Pos measurements. It can be read as, “measurements of type 3D Pos valid for any object A are also valid for any other object B connected to A via an identity edge.”

patterns. In the following example, we look for objects of type **marker** that are contained in the same locale as the **camera1** node, shown in Figure 4.

Using this formalism, an object can be in multiple locales simultaneously, and there’s no need for a hierarchical relationship between locales. In addition, locales need not necessarily correspond to physical entities, such as rooms, but they could also express abstract concepts such as a social network.

A dedicated locale manager adds the locale edges at runtime. In our system, we implemented a general-purpose locale manager that can be configured using XML to listen for events from a particular sensor and insert or remove locale edges when an object enters or

leaves a particular locale, or when a sensor starts to deliver data about an object at all. Locales are defined by the convex hull of a set of points. Each locale has a separate manager to enhance scalability.

Unlike the sensors used for AR visualization and interaction, sensors that are monitored to detect locale containment don’t need to be extremely precise. Therefore, we can use relatively cheap technology, such as Wi-Fi or RFID positioning, for this purpose.

### Identity Pattern for Assigning Anonymous Sensors

Most sensors typically used in AR setups can uniquely identify the objects they’re tracking. Other (cheap) sensors don’t offer this feature, such as a surveillance camera combined with a simple blob-detection algorithm. In terms of the SRG, we model these objects as nodes with a special anonymous attribute and a temporary ID, assigned by the anonymous sensor client. In a situation where an ID-sensing tracker detects one of the objects with sufficient accuracy, the system can conclude that the two objects actually are the same by comparing the positions of both sensors. In this situation, the ID sensor doesn’t need to provide continuous measurements, but a single event, such as the read-out of an RFID tag or the detection of a marker in a single camera image, is sufficient. In the Ubitrack formalism, we model this *object merging* by inserting symmetric identity edges between the two nodes in the SRG. The meaning of an *identity* edge is that each measurement of one object is valid for the other object as well. The measurement identity pattern, shown in Figure 5, expresses this in the Ubitrack formalism.

The actual insertion of identity edges is done by a general-purpose identity manager, which is configured and instantiated for each anonymous sensor. This manager queries the Ubitrack system for all anonymous objects of the given tracker and a list of candidate

objects, usually limited to a particular locale or type (for example, person). By comparing the positions of both anonymous and candidate objects, the manager can detect matching objects and insert identity edges, which lets the system use the anonymous sensor until it loses track of the object.

### Dynamic Sensor Fusion Example

To illustrate Ubitrack’s capabilities, we describe an example involving dynamic fusion of multiple sensors for presenting AR objects to a mobile user. The user is guided through the hallway to a lab, where a virtual sheep is pastured. The mobile AR system collaborates with several stationary tracking devices installed in the environment. Figure 6a shows a map of the environment.

The user carries a tablet PC with integrated camera and an inertial measurement unit (IMU) with compass (Inter-sense InertiaCube), shown in Figure 6b. The camera image provides a video see-through AR view on the screen. The camera also detects black-and-white fiducial markers and determines their pose relative to the camera. For higher robustness and accuracy, the results are fused with the IMU using a Kalman filter. The client PC retrieves set of available markers from the Ubitrack server, depending on the current locale.

The mobile user enters the hallway in front of the lab. Using the camera, the user can track a fiducial marker at the hallway entrance. As the user looks at the marker, the system registers the user’s position.

On the hallway ceiling, an overhead tracker with a wide-angle camera detects and tracks people. This overhead tracker adds newly detected people as anonymous objects to the SRG. However, the identity manager can identify one of them as the user by comparing positions reported by the overhead camera and the marker tracker.

As the user turns (measured by the IMU) and looks down the hallway, he or she can see a sign in front of the lab

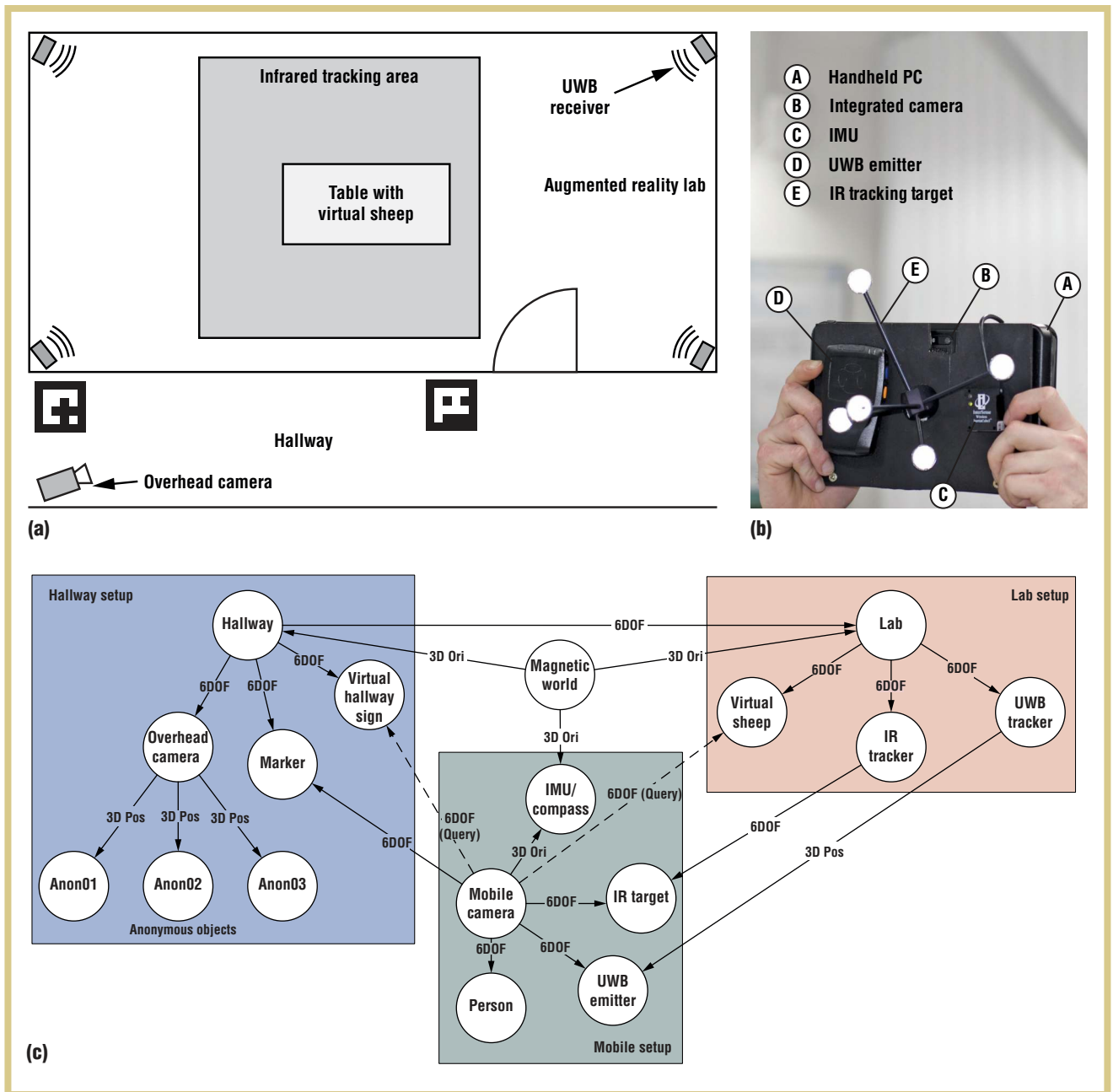


Figure 6. Illustration of the example scenario: (a) a map of the environment, showing the positions of markers, the overhead tracker and ultra-wideband and infrared-optical tracking systems, (b) the handheld PC with an integrated camera, inertial measurement unit (IMU), ultra-wideband emitter (UWB) and infrared-optical tracking target and (c) the spatial relationship graph (SRG) of the setup. (Note: For clarity, this SRG only contains sensor and query edges. Also, we show only the “type” attribute.)

door. Walking down the hallway, the user stays tracked by the overhead camera. In front of the lab door, another marker lets the user take a look at the sign with higher accuracy.

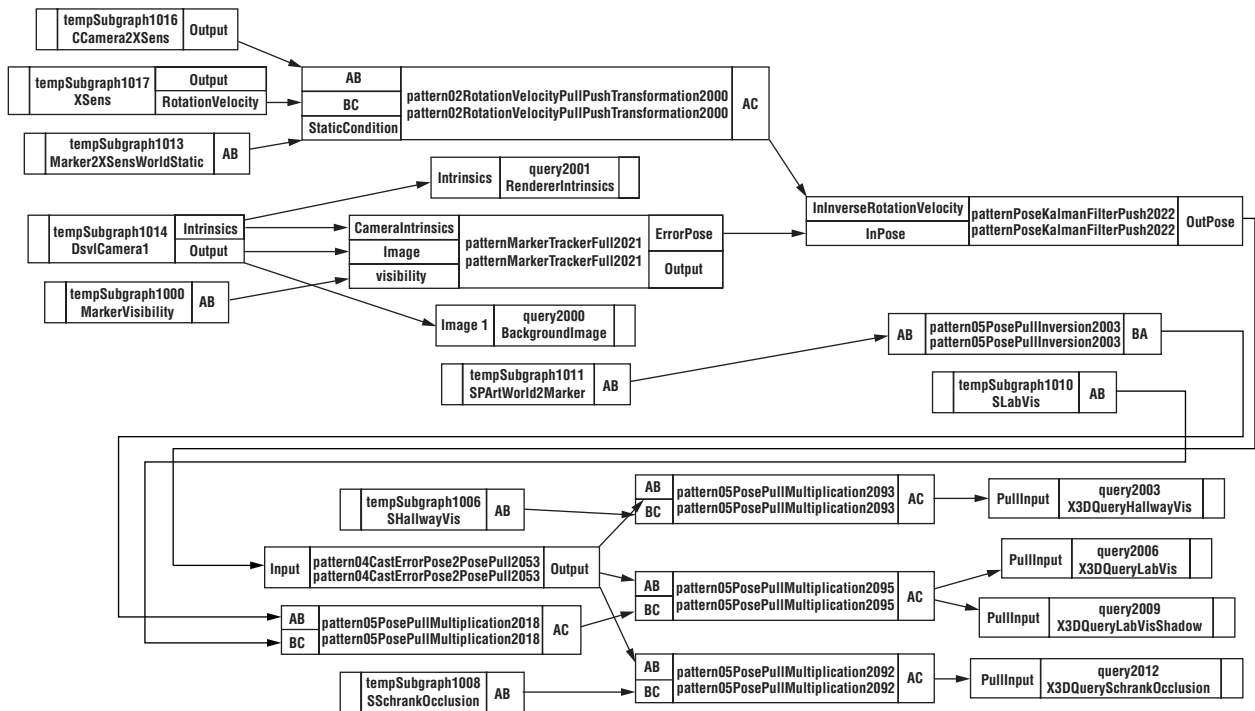
The lab is covered by a commercial Ubisense ultra-wideband (UWB)

RF location system, providing position updates with an accuracy of approximately 15 centimeters. A UWB-emitting tag is attached to the mobile setup, allowing the PC to be detected and uniquely identified. In the center of the lab, a working volume of

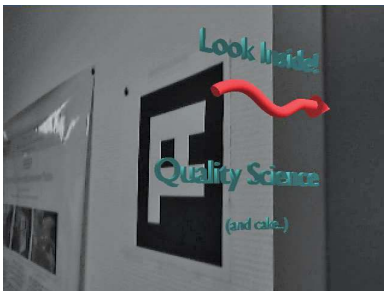
approximately  $4 \times 4$  meters is covered by a high-precision A.R.T. infrared-optical tracker. To be tracked by this system, the mobile setup includes a set of retro-reflective marker balls.

When the user enters the lab, the UWB-emitting tag is detected.





**(a)**



**(b)**

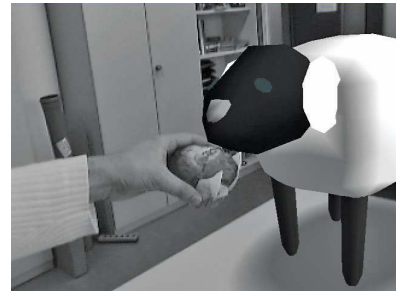
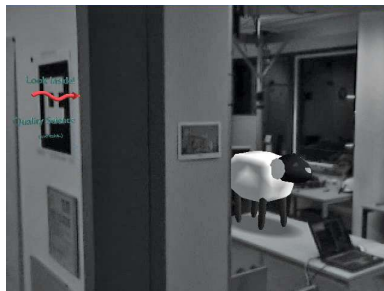


Figure 7. Results of the example scenario. (a) One of the generated dataflow graphs and (b) some captured video frames as seen by the mobile AR user.

Using the tag data and the compass, the system determines the user's pose, and the user can see the virtual sheep on the table. As the user approaches the table, he or she enters the tracking area of the high-performance infrared tracking system (detected by the UWB tracker), which automatically connects to the mobile setup.

Figure 6c shows the SRG of the setup. The edges connecting the mobile setup and the hallway or lab might be unavailable, depending on the user’s locale.

The software system used in the scenario consists of a Ubitrack server and a set of clients, which together provide the information for the SRG and dynamically add or remove information derived from the tracking context. For the static parts of the hallway and lab, two *world clients* add the relationships between the different stationary tracking systems and the rooms to the global SRG. These clients also add the virtual object nodes, which contain the URL of associated 3D models for rendering.

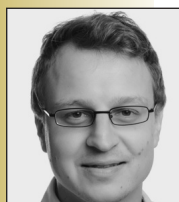
The mobile client sends the SRG describing the mobile setup to the server at startup. The setup includes a camera, IMU, UWB emitter, and infrared target as well as the calibrated relationships between them. Because the mobile client contains the rendering application, it also sends a query for the 6DOF pose of all renderable objects in the current room relative to the camera coordinate frame. Another query provides the square marker tracking with the list of available markers in the room.



**Daniel Pustka** is a research and development engineer at Advanced Realtime Tracking (A.R.T.). His research interests include sensor fusion, multisensor registration, and the convergence of augmented reality and ubiquitous computing. Pustka has a diploma in computer science from the Technical University of Munich, where he performed the research reported here. He is a member of IEEE. Contact him at [daniel.pustka@ar-tracking.de](mailto:daniel.pustka@ar-tracking.de).



**Manuel Huber** is a research assistant and PhD student in the Augmented Reality Group at the Technical University of Munich. His research interests include augmented reality, multisensor systems, and RFID-based localization. Huber has a diploma in computer science from the Technical University of Munich. He is a member of the ACM, IEEE, and the German Society for Computer Science (Gesellschaft für Informatik). Contact him at [huberma@in.tum.de](mailto:huberma@in.tum.de).



**Christian Waechter** is a research assistant in the Augmented Reality Group at the Technical University of Munich. His research interests include people tracking, multisensor environments, and ubiquitous tracking. Waechter has a diploma in computer science from the Technical University of Munich. He is a member of IEEE. Contact him at [waechter@in.tum.de](mailto:waechter@in.tum.de).



**Florian Echtler** is a postdoctoral researcher at the Munich University of Applied Sciences. His research interests include touch-based user interfaces and augmented reality. Echtler has a PhD in computer science from the Technical University of Munich, where he performed the research reported here. He is a member of the ACM and the German Society for Computer Science (Gesellschaft für Informatik). Contact him at [echtler@in.tum.de](mailto:echtler@in.tum.de).



**Peter Keitler** is co-founder of Extend3D ([www.extend3d.com](http://www.extend3d.com)), a company offering products and services in the field of industrial augmented reality. He was formerly a research assistant in the Augmented Reality Group at the Technical University of Munich. Keitler has a PhD in computer science from the Technical University of Munich. Contact him at [peter.keitler@extend3d.de](mailto:peter.keitler@extend3d.de).



**Gudrun Klinker** is a professor of computer science at the Technical University of Munich. Her research interests include approaches to ubiquitous augmented reality that lend themselves to realistic industrial applications. Klinker has a PhD in computer science from Carnegie Mellon University. Contact her at [klinker@in.tum.de](mailto:klinker@in.tum.de).



**Joseph Newman** is a senior software engineer at Ubisense. His research interest lies in the intersection of ubiquitous computing and mixed reality, especially with regard to mobility. Newman has a PhD in ubiquitous tracking for distributed mixed-reality environments from the Graz University of Technology, where he performed the research reported here. Contact him at [joe.newman@ubisense.net](mailto:joe.newman@ubisense.net).



**Dieter Schmalstieg** is a full professor of virtual reality and computer graphics at the Graz University of Technology, where he directs the Studierstube research project on augmented reality. His research interests include augmented reality, virtual reality, real-time graphics, 3D user interfaces, and ubiquitous computing. Schmalstieg has a PhD in computer science from Vienna University of Technology. Contact him at [schmalstieg@tugraz.at](mailto:schmalstieg@tugraz.at).

The overhead tracking system is represented by another client that runs on a stationary computer. It adds detected people as anonymous objects with temporary IDs to the server's SRG. The data association between the overhead tracking and the mobile setup is done by a separate identity management client, which receives the positions of anonymous and candidate objects in the hallway from the Ubitrack system.

Inside the lab, the UWB client adds detected UWB emitters to the SRG and provides their positions to the data-flow network. Because these emitters have unique IDs, they can be directly associated with other clients' objects. Locale containment for the lab and the infrared tracker is handled by a locale

manager running on a computer in the lab.


Figure 7 shows an example of the generated DFGs and some captured video frames from the view of the mobile AR user.

Our system architecture fulfills the exacting performance requirements necessary for immersion in an AR world, while simultaneously extending the bounds within which users can experience AR environments, thus bridging the divide between the pervasive computing and AR worlds.

Ubitrack differs in several aspects from current ubicomp approaches such

as the location stack.<sup>5</sup> In Ubitrack, all sensor measurements are modeled as (relative) spatial relationships between two arbitrary coordinate systems. Therefore, basic coordinate system transformations are handled at the same conceptual layer as more advanced sensor fusion algorithms. In fact, transformations are necessary to enable fusion and not just a convenience function for application programmers.

On the architectural side, Ubitrack maintains a strict separation between configuration and runtime layers. All processing that involves inspection of measurements (such as ID assignment) must be done by specialized clients at the runtime layer that in turn reconfigure the configuration layer's SRG.

UTQL provides a common standard that lets clients describe previously unknown sensors and environments and express the relationships in which they are interested. Sensor data is dynamically fused and communicated to clients in a peer-to-peer fashion via a dataflow network. The emergent structure of this network depends on the behavior of clients implemented as reusable extensible and largely decentralized components. 

## ACKNOWLEDGMENTS

This work was supported by the Bayerische Forschungsförderung (project TrackFrame, AZ-653-05) and the Presenccia Integrated Project funded

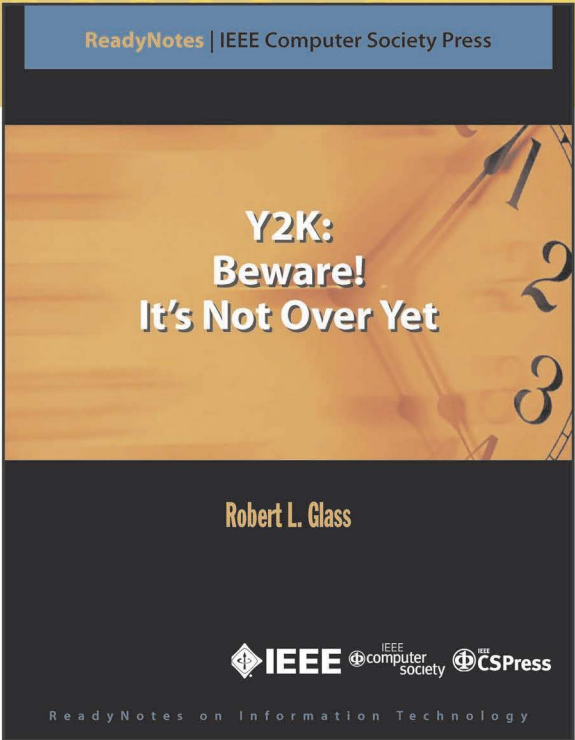
under the European Sixth Framework Program, Future and Emerging Technologies (FET) (contract number 27731).


## REFERENCES

1. D. Pustka et al., "Spatial Relationship Patterns: Elements of Reusable Tracking and Calibration Systems," *Proc. IEEE Int'l Symp. Mixed and Augmented Reality (ISMAR)*, IEEE CS Press, 2006, pp. 88–97.
2. R.Y. Tsai and R.K. Lenz, "A New Technique for Fully Autonomous and Efficient 3D Robotics Hand/Eye Calibration," *IEEE Trans. Robotics and Automation*, vol. 5, no. 3, 1989, pp. 345–358.
3. J. Newman, "Ubiquitous Tracking for Augmented Reality," *Proc. IEEE Int'l Symp. Mixed and Augmented Reality (ISMAR)*, IEEE CS Press, 2004, pp. 192–201.
4. J. Hightower, B. Brumitt, and G. Borriello, "The Location Stack: A Layered Model for Location in Ubiquitous Computing," *Proc. 4th IEEE Workshop Mobile Computing Systems and Applications (WMCSA)*, IEEE CS Press, 2002, pp. 22–28.
5. P. Keitler et al., "Management of Tracking for Mixed and Augmented Reality Systems," *The Engineering of Mixed Reality*, E. Dubois, P. Gray, and L. Nigay, eds., Springer, 2009, pp. 251–273.



Selected CS articles and columns are also available for free at <http://ComputingNow.computer.org>.



**NEW** from  **CS Press**

**Y2K: BEWARE!  
IT'S NOT OVER YET**

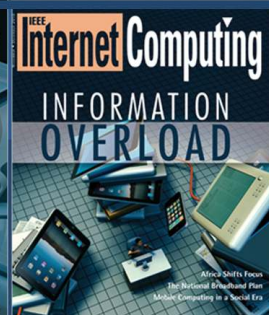
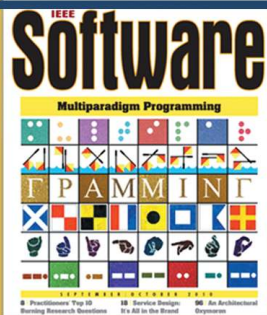
by Robert L. Glass

In these new and updated essays told around the proverbial IT campfire, Robert Glass looks back at the Y2K crisis to find lessons applicable to future crises coming all too soon.

PDF edition • \$12 list / \$9 members • 15 pp.

Order Online:  
[COMPUTER.ORG/STORE](http://COMPUTER.ORG/STORE)

**13**  
magazines



# Subscribe for HALF YEAR 2011 by 12 August



**IEEE**



IEEE  
**computer society**



**14**  
journals