Dynamic Gyroscope Fusion in Ubiquitous Tracking Environments

Daniel Pustka* Gudrun Klinker†

Institut für Informatik, Technische Universität München

ABSTRACT

Ubiquitous Tracking (Ubitrack) setups, consisting of many previously unknown sensors, offer many possibilities to perform sensor fusion in order to increase robustness and accuracy. In particular, the dynamic combination of mobile and stationary trackers enables the creation of new wide-area tracking concepts.

In this work, we present a setup in which a gyroscope is dynamically fused with three different mobile and stationary sensors, based on the concepts of Spatial Relationship Graphs (SRGs) and Patterns. For this, we contribute new patterns that, based on well-known algorithms, enable the transformation of rotation velocity and the fusion with different absolute trackers. The usefulness of the approach is shown in a system that automatically reconfigures the SRG based on course tracking data, and, depending on the structure of this SRG, automatically selects a suitable fusion algorithm.

Keywords: Augmented Reality, Tracking, Calibration, Sensor Fusion, Gyroscopes, Inertial Sensors

Index Terms: H.5.1 [Information Interfaces and Presentation]: Multimedia Information Systems—Artificial, augmented, and virtual realities; I.3.1 [Computer Graphics]: Hardware Architecture—Input devices

1 INTRODUCTION

In optical see-through augmented reality, accurate tracking, especially of the user's head orientation, is important to provide a good 3D registration of virtual objects with the world. Starting with the work of Azuma [3], many publications in the past have shown that inertial sensors, in particular gyroscopes, can be used to improve the latency and accuracy of the orientation tracking. By fusing the inertial data with measurements from other sensors, drift can be eliminated, which is the dominant source of error in inertial sensors.

In our previous work, we introduced the concept of spatial relationship graphs (SRGs) [13], which forms the basis of the ubiquitous tracking (Ubitrack) approach. SRGs allow the formal specification of tracking setups by describing the relationships between different coordinate frames and their properties. Using spatial relationship patterns [16], new geometric relationships can be derived from the SRG automatically, using well-known algorithms for tracking, calibration and sensor fusion. Based on this formal framework, a Ubitrack system can dynamically combine a variety of available, but previously unknown sensors in order to deliver to applications the tracking performance they require. This ideally leads to large heterogeneous systems where users can seamlessly move between imprecise wide-area tracking systems and local high-precision trackers. Accuracy and availability can be improved by combining user-worn and world-fixed sensors.

The goal of this paper is to use the Ubitrack tools to analyze typical tracking situations that can be improved with gyroscope fusion. This paper further extends the catalog of spatial relationship patterns given in [16] with new patterns that deal with these situations. We will distinguish particularly between the fusion with inside-out and outside-in trackers. We then derive the necessary formulas for transformation of incremental orientation data and provide a solution for gyroscope-to-tracker calibration. To show the usefulness of our approach, we first perform an evaluation of the sensor fusion quality. Then, a real ubiquitous tracking system is described, where, based on the derived spatial relationship patterns, the gyroscope worn by a mobile user is automatically fused with an inside-out marker tracking and two stationary tracking systems, depending on availability.

Related Work Hybrid tracking setups, consisting of inertial sensors combined with other tracking methods, are a well-studied field of research. Azuma [3] has introduced gyroscope sensing to AR, and the topic has been further deepened by [6, 21, 18, 17, 10, 2] and others.

We complement this work by adding a new on-line calibration algorithm and a spatial relationship graph formulation of the involved algorithms for measurement transformation, calibration and fusion. This is necessary to automatically integrate gyroscopes into ubiquitous tracking systems in a meaningful way.

In our previous work, the concepts of spatial relationship graphs (SRGs) [13] and spatial relationship patterns [16] were introduced, which allow to formally model relationships between the different coordinate frames in a tracking setup and for describing the operations performed by a tracking/calibration algorithm. These concepts will be described in more detail in the next section, and then extended by the formalism necessary to integrate gyroscopes into the framework.

Apart from our own work, there are relatively few efforts in building large heterogeneous tracking setups using different off-the-shelf sensors. One example is described in [7], however the authors are focusing on providing a nice hand-off between commercially available systems and their own sensors.

2 SPATIAL RELATIONSHIP GRAPHS AND PATTERNS

In [16], [13] and [15] the concept of Spatial Relationship Graphs (SRGs) and Spatial Relationship Pattern as a method to formally describe tracking environments and tracking algorithms respectively was introduced. Since our discussion makes heavy use of these notions we briefly outline the main concepts and their interaction here. Again, for more details see [16], [13] and [15].

2.1 Spatial Relationship Graphs

In our approach, a tracking setup is specified using a spatial relationship graph (SRG), which describes relevant coordinate frames and tracking devices. The nodes of a spatial relationship graph represent coordinate frames, e.g. that of a camera located at its camera center, that of a CAD-model augmented onto some object or that of a tracker target. If the transformation between two coordinate frames is known or measured at runtime, this is indicated by a directed edge between those coordinate frames. Note that edges do not represent the measurements themselves, but indicate availability of measurements, i.e. they usually contain a reference to some software component, e.g. a driver, that provides the actual measurements at runtime. Edges in the SRG also have attributes specifying

*e-mail:daniel.pustka@in.tum.de

†e-mail:gudrun.klinker@in.tum.de

relevant properties of the measurement, such as data type (3D Position, 3D Rotation, 6DoF Pose, etc.), quality or whether the relationship is known to be static. In the graphical representation, these attributes are drawn as annotations to the edge in question. An example of an SRG is shown in figure 10.

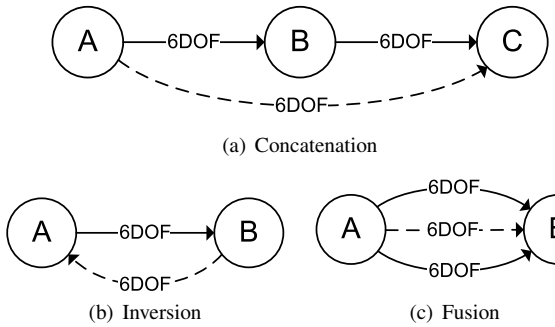
2.2 Construction of Data Flow Networks

The goal of our approach is to take the abstract spatial relationship graph declaration and construct at runtime a data flow network consisting of tracking and transformation components that provide an application with estimates of those spatial relationships that the application requires. Generally, the transformations needed by an application are not directly measured, but instead can be *inferred* from existing measurements. When a new transformation is inferred, this adds a new edge to the graph which connects different nodes than the existing edges or has different attributes.

To describe which measurements can be inferred from the given SRG description, we use spatial relationship patterns [16]. Spatial relationship patterns are subgraphs of SRGs, which have two different kinds of edges: input edges that have to be present before a pattern can be applied, and output edges, which are added to the SRG afterward. Input edges are denoted by solid lines and output edges are dashed.

The goal is to find a chain of pattern applications on the given SRG that allows us to infer the edge that corresponds to an application's request. Each pattern application corresponds to a component in a data flow network that performs the actual computation by taking the measurements of the input components and producing the inferred measurement of the output edge. Therefore, by finding the right chain of pattern applications, a data flow network can be constructed automatically at run time from a given SRG description.

Most tracking problems involving only 6D measurements can be solved using the three spatial relationship patterns depicted in figure 1:



(a) Concatenation

(b) Inversion (c) Fusion

Figure 1: Basic Spatial Relationship Patterns

The *inversion* pattern represents the most basic transformation of tracking data: Consider that a transformation from coordinate frame A to coordinate frame B is described by a 4×4 matrix M. Then the inverse transformation, going from B to A, can be computed as M^{-1}. Similar methods exist when the 6D transformation is described e.g. by a translation and a quaternion.

The *concatenation* pattern exploits the transitivity of spatial relationships: If the transformations from A to B and from B to C are given as 4×4 matrices M and N, the transformation from A to C can be computed as the product MN.

When two or more edges are available between two nodes, the measurements can be fused by statistical combination, using the accuracy information that accompanies the measurements. This usually results in estimates of higher accuracy.

Note that the same patterns exist for 3D rotations. More complex patterns based on the concept of corresponding measurements are described in [16]. The goal of this paper is to extend this catalog of patterns with the algorithms necessary for the integration of gyroscopes into such Ubiquitous Tracking setups.

3 Incremental Rotation and Rotation Velocities

We start the discussion of gyroscope integration by deriving some basic rules for the treatment of incremental rotations. Given two sequential rotations r_{t_1} and r_{t_2} at times t_1 and t_2, we can express r_{t_2} as r_{t_1} multiplied by an incremental rotation Δr:

$$r_{t_2} = r_{t_1} \cdot \Delta r \quad where \quad \Delta r = r_{t_1}^{-1} \cdot r_{t_2}$$

In the spatial relationship graph we treat incremental rotations as separate edges. The following figure shows an SRG where both absolute orientation and incremental rotation is measured.

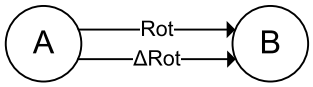


Figure 2: SRG with absolute and incremental rotation measurements

3.1 Basic Incremental Rotation Patterns

In order to combine gyroscopes with other sensors, we need concatenation and inversion patterns for incremental rotation, similar to the patterns described above, as the two sensors are unlikely to use the same coordinate frames. In realistic gyroscope scenarios, we can assume the following restrictions: First, only concatenation of incremental with absolute orientation is necessary, not incremental with incremental. Second, the absolute orientation part can be considered static, i.e. not changing over time.

Target Coordinate Change The first important transformation of relative orientation is the change of the target coordinate frame. For any given pair of rotations r and q let $r' = r \cdot q$ be the product of r and q, which effectively moves the target coordinate frame of the transformation r.

Now let t_1 and t_2 be two consecutive points in time and let r and q be two absolute rotations measured at t_1 and t_2 resulting in r_{t_1}, r_{t_2} respectively q_{t_1}, q_{t_2}. We can compute the resulting incremental rotation in the transformed coordinate frame as

$$\begin{aligned} \Delta r' &= r'^{-1}_{t_1} \cdot r'_{t_2} \\ &= (r_{t_1} \cdot q_{t_1})^{-1} \cdot (r_{t_2} \cdot q_{t_2}) \\ &= q_{t_1}^{-1} \cdot \Delta r \cdot q_{t_2}. \end{aligned}$$

Assuming that q is static, i.e. $q_{t_1} = q_{t_2} = q$, we can write

$$\Delta r' = q^{-1} \cdot \Delta r \cdot q.$$

The resulting spatial relationship pattern is displayed in figure 3:

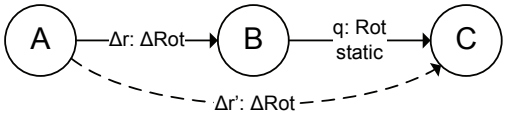


Figure 3: Incremental rotation target coordinate change pattern

Source Coordinate Change Similarly, if we let $r' = q \cdot r$ we can change the source coordinate frame of the rotation r. In this case we calculate the resulting incremental rotation in the transformed coordinate frame $\Delta r'$ as

$$\begin{aligned}\Delta r' &= {r'_{t_1}}^{-1} r'_{t_2} \\ &= (q_{t_1} \cdot r_{t_1})^{-1} \cdot (q_{t_2} \cdot r_{t_2}) \\ &= r_{t_1}^{-1} \cdot q_{t_1}^{-1} \cdot q_{t_2} \cdot r_{t_2}.\end{aligned}$$

Assuming again that q is static, this simplifies to

$$\Delta r' = \Delta r$$

This means that incremental rotations are valid for all source coordinate frames connected by static transformations. The resulting spatial relationship pattern is displayed in the figure 4:

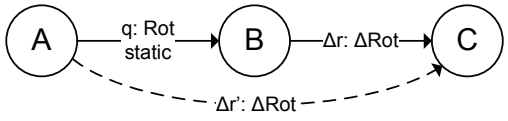


Figure 4: Incremental rotation source coordinate change pattern

Inversion The third transformation of incremental rotation we need is the inversion, i.e. the exchange of source and target coordinate frames. We need to compute $\Delta r'$ of $r' = r^{-1}$:

$$\begin{aligned}\Delta r' &= r'^{-1}_{t_1} \cdot r'_{t_2} \\ &= r_{t_1} \cdot r_{t_2}^{-1} \\ &= r_{t_1} \cdot (r_{t_1} \cdot \Delta r)^{-1} \\ &= r_{t_1} \cdot \Delta r^{-1} \cdot r_{t_1}^{-1}\end{aligned}$$

This shows that it is not possible to invert an incremental rotation without knowing the absolute orientation. Thus, the spatial relationship pattern becomes:

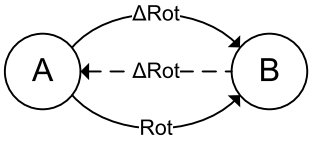


Figure 5: Incremental rotation inversion pattern

3.2 Rotation Velocities

So far, it did not matter, how exactly rotations were represented, as long as meaningful product and inversion operators where defined. For the rest of this paper, we will use quaternions, which represent rotations as 4-element vectors $q = (q_1, q_2, q_3, q_0)$ of norm $|q| = 1$. Quaternions have the advantage of a relatively low-dimensional representation (compared to matrices) and easy to compute product and inversion operations (compared to e.g. Euler angles). For more details on quaternions, we recommend [12].

Also, we have only dealt with incremental rotations, which have the disadvantage of depending on the time interval between the measurements. In order to be able to compute sampling-independent rotation velocities, a representation is required where multiplication and division by time are easy to compute. We therefore express rotation velocities as 3-element vectors $\mathbf{v}$ where $\mathbf{v}$ represents the axis of rotation and $|\mathbf{v}|$ the angle of rotation. By representing quaternions as 3-element vectors and a scalar $q = (\mathbf{q}, q_0)$, we can convert between incremental rotation quaternions and rotation velocities:

$$vel(q, \Delta t) = \mathbf{q} \frac{2\cos^{-1}(q_0)}{|\mathbf{q}|\Delta t}$$

$$quat(\mathbf{v}, \Delta t) = \left(\frac{\mathbf{v}}{|\mathbf{v}|} \sin \frac{\Delta t |\mathbf{v}|}{2}, \cos \frac{\Delta t |\mathbf{v}|}{2} \right)$$

In the rest of this paper we will use quaternions to represent rotations, and axis-angle vectors for rotation velocities. Note that the transformation formulas from the previous section are still valid for rotation velocities expressed as axis-angle vectors, if we silently extend them to a 4-element quaternion by setting $v_0 = 0$. The reason is that the product $q \cdot \Delta r \cdot q^{-1}$ (also called the quaternion rotation operator) only changes the direction of Δr but not its magnitude, given that $|q| = 1$.

4 Gyroscope Calibration

The spatial relationship graph of a typical gyroscope fusion situation is shown in figure 6. The gyroscope incrementally tracks its

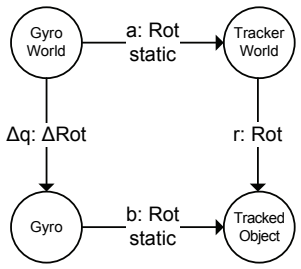


Figure 6: A typical tracking situation with a gyroscope

own orientation q with respect to some (usually unknown) world coordinate frame. The other tracker tracks the orientation of an object with respect to a different world coordinate frame. This world coordinate frame is statically related to the gyroscope world (for now, we can just ignore the direction of the transformation r). The key point here is that the gyroscope is rigidly connected to the tracked object, by some static, but unknown rotation b. As we have defined incremental rotation as $q_2 = q_1 \cdot \Delta q$, the direction of the Δq edge indicates that the rotation increment Δq is measured in the gyroscope coordinate frame.

In order to fuse the gyroscope with the absolute tracking, we need to bring the measurements Δq into the tracker's coordinate frames. As was shown above, the source coordinate frame can be moved without actually knowing a, but the rotation b needs to be known. Note that, for the same reason, small amounts of drift, which can be modeled as a change in the gyroscope's world coordinate frame, only minimally change the rotation increments. This small error can later be compensated by the fusion algorithm.

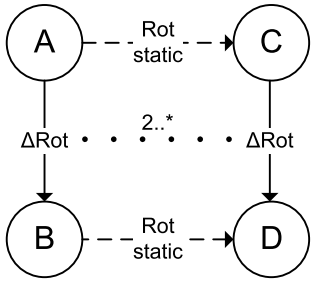


Figure 7: The rotation-only hand-eye calibration pattern

Hand-Eye Calibration The problem of determining the unknown but static transformations a and b given corresponding sets of incremental transformations Δq and Δr is well-known in the robotics community and called the "hand-eye calibration". It has also previously been applied to tracker alignment [4].

For the gyroscope calibration we use the Tsai-Lenz [19] algorithm, which directly uses quaternions and can easily be implemented, especially when only rotation is needed. Given pairs of corresponding incremental rotation quaternions $\Delta q_i = (\Delta\mathbf{q}_i, q_{0,i})$ and $\Delta r_i = (\Delta\mathbf{r}_i, r_{0,i})$, the algorithm solves a set of equations

$$Skew(\Delta\mathbf{q}_i + \Delta\mathbf{r}_i)\mathbf{b}' = \Delta\mathbf{r}_i - \Delta\mathbf{q}_i$$

where

$$Skew(\mathbf{v}) = \begin{bmatrix} 0 & -v_z & v_y \\ v_z & 0 & -v_x \\ -v_y & v_x & 0 \end{bmatrix}$$

As *Skew* is singular, at least two corresponding pairs are required to solve for $\mathbf{b}'$ in a least-squares fashion. The imaginary part $\mathbf{b}$ of the resulting quaternion b is determined from $\mathbf{b}'$ by computing

$$\mathbf{b} = \frac{\mathbf{b}'}{\sqrt{1+|\mathbf{b}'|}}$$

The real part b_0 of b is computed by requiring that $|b| = 1$.

Online Computation In order to avoid having to store long lists of measurements, we can implement the algorithm iteratively, using the Kalman filter equations:

$$\begin{aligned} \mathbf{b}_i &= \mathbf{b}_{i-1} + K_i(\mathbf{z}_i - H_i\mathbf{b}_{i-1}) \\ P_i &= (I - K_iH_i)P_{i-1} \end{aligned}$$

where

$$\begin{aligned} K_i &= P_{i-1}H_i^T(H_iP_{i-1}H_i^T + R_i)^{-1} \\ H_i &= Skew(\Delta\mathbf{q}_i + \Delta\mathbf{r}_i) \\ \mathbf{z}_i &= \Delta\mathbf{r}_i - \Delta\mathbf{q}_i \\ R_i &= I \end{aligned}$$

The initial rotation can be set to $\mathbf{b}_0 = \mathbf{0}$ if the initial covariance is set to $P_0 = Ic$, where c is a very high value. Note that this Kalman filter formulation is an optimal estimator for $\mathbf{b}$, as the underlying equation system is linear [11].

Measurement Selection The original Tsai-Lenz paper recommends to take a number of absolute measurements q and r, and use the incremental rotations $\Delta q_{ij} = q_i^{-1}q_j$ from all possible pairs q_i and q_j. While, this way, a maximum of information can be extracted from the given measurements, the approach is not suited to the gyroscope case. Rotation increments integrated over a long time suffer from the gyroscopic drift. On the other hand, incremental rotations computed from two consecutive measurements of the absolute tracker can be dominated by the tracker noise if the two rotations are not sufficiently apart. We therefore use both an upper bound on the time and a lower bound on the rotation angle for selecting incremental rotations used in the calibration. Additional robustness against outliers can be gained by requiring the incremental rotations of both systems to have a similar magnitude.

5 Gyroscope Fusion

Having properly calibrated the gyroscope with respect to the absolute tracker, we can think about fusing the two. We start this section with explaining the geometrical considerations before we come to the actual fusion algorithms.

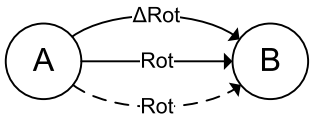


Figure 8: The outside-in gyroscope fusion pattern

5.1 Geometrical Considerations

In AR/VR setups, a gyroscope can be used in two different ways that have to be distinguished:

Outside-In Tracking In the first case, the gyroscope is attached to an object that is tracked by an outside-in tracking system. Neglecting the change in translation, the known pose can conceptually be split into rotation and translation, and the incremental rotation is multiplied to the absolute rotation:

$$r_i = r_{i-1} \cdot \Delta r_i$$

After that, rotation and translation are combined to a 6DOF pose again. The resulting spatial relationship pattern is depicted in fig. 8. Note that we assume that the gyroscope's incremental rotation is already transformed into the tracker coordinate frame using the source and target coordinate change patterns.

Inside-Out Tracking Inside-out tracking, where a camera is mounted on the user's head and normally looking in the same direction as the user, is frequently used in Augmented Reality applications. The advantage of inside-out tracking compared to outside-in is that the accuracy along the x and y axes is usually very good, with the trade-off of low accuracy in the depth (which is hardly noticeable).

We usually assume that some static object is observed by a moving camera (see Fig 9). The simplest way of fusing is to invert the absolute tracker's pose, apply the outside-in fusion and invert the result again. In some cases, however, this method may fail. If the camera observes a planar structure which is parallel to its image plane, the rotational accuracy of the inside-out tracker tends to be very bad (see e.g. [14]), which will add a lot of jitter to the translation of the inverted pose. If now the rotation is corrected by the gyroscope without considering the translation, this jitter will be visible in the image.

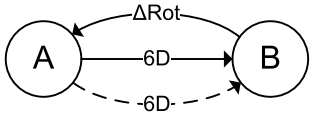


Figure 9: The inside-out gyroscope fusion pattern

We therefore need to explicitly model that the incremental rotation not only changes the absolute orientation, but also causes a change in the position of the observed world with respect to the camera. This rotation of the position vector is inverse to the incremental rotation. Also, the rotation increment must be converted to the other coordinate frame, before it can be applied to the world's orientation. Thus, a transformed rotation increment Δr, measured by the gyroscope, causes the following update of the pose $P = (t_p, r_P)$, measured by the camera:

$$\begin{aligned} t_{P,i} &= \Delta r_i^{-1} \cdot t_{P,i-1} \cdot \Delta r_i \\ r_{P,i} &= r_{P,i-1} \cdot r_{P,i-1}^{-1} \cdot \Delta r_i \cdot r_{P,i-1} \end{aligned}$$

The pattern of inside-out fusion is shown in fig. 9.

5.2 Fusion Technique

In case the gyroscope is to be used as a simple fall-back solution when the other tracker fails, the formulas given in the previous section can be directly applied. However, one should not forget to add timestamp-based interpolation of the rotation increments, as combined off-the-shelf trackers are unlikely to be synchronized. The drawback is that no real sensor fusion takes place. Therefore this would not result in an increased update rate or reduced jitter, compared to using the absolute tracker alone.

For that reason, we will concentrate in this section on the implementation of an (extended) Kalman filter for the fusion process. We assume that the reader already has a basic understanding of the Kalman filter's prediction-update cycle (see e.g. [20] for an introduction) and focus on the motion and measurement models. The required Jacobians necessary for the implementation can easily be computed using any computer algebra software. Other fusion techniques can be used instead, such as the Unscented Kalman Filter [9], which has shown to provide a slightly better accuracy at higher computational cost [1].

Motion Model The state of the observed transformation is modeled as a translation vector $\mathbf{t}$, a translational velocity vector $\mathbf{v}$, a rotation quaternion r and a rotation velocity vector $\mathbf{w}$. This 13-element state vector is associated with a 13×13-element covariance matrix P. In the time update step for a time increment of Δt, the following computations are performed:

$$\begin{aligned} \mathbf{t}_i &= \mathbf{t}_{i-1} + \Delta t\, \mathbf{v}_{i-1} \\ \mathbf{v}_i &= \mathbf{v}_{i-1} \\ r_i &= r_{i-1} \cdot quat(\mathbf{w}_{i-1}, \Delta t) \\ \mathbf{w}_i &= \mathbf{w}_{i-1} \\ P_i &= P_{i-1} + Q(\Delta t) \end{aligned}$$

where $Q(\Delta t)$ is the process noise covariance matrix. For the inside-out fusion, both translation and its velocity have to be rotated by the transformed rotation increments:

$$\begin{aligned} \mathbf{t}_i &= r_{w,i} \cdot (\mathbf{t}_{i-1} + \Delta t\, \mathbf{v}_{i-1}) \cdot r_{w,i}^{-1} \\ \mathbf{v}_i &= r_{w,i} \cdot \mathbf{v}_{i-1} \cdot r_{w,i}^{-1} \end{aligned}$$

with

$$r_{w,i} = r_{i-1} \cdot quat(\mathbf{w}_{i-1}, \Delta t) \cdot r_{i-1}^{-1}$$

Note that we do not integrate state elements for drift correction, as we assume this is already done inside the gyroscope.

Measurement Models The measurement equation h_a for the absolute tracker is trivial, as the measurements directly correspond to elements of the state vector:

$$h_{a,i} = (t_i^T, r_i^T)^T$$

Integrating the gyroscope measurements in the outside-in case also is simple:

$$h_{oi,i} = \mathbf{w}_i$$

For the inside-out case, the gyroscope measurement equation is the same as the incremental rotation inversion equation given above, as the gyroscope measurements are made in the coordinate frame of the camera.

$$h_{io,i} = r_i \cdot \mathbf{w}_i^{-1} \cdot r_i^{-1}$$

In all cases, the quaternion part of the state vector needs to be normalized to $|r| = 1$ after the measurement integration. As the quaternions q and $-q$ describe the same rotation, and it is usually not possible to predict which version a tracker returns, the one which is closer to the state variable must be chosen for the measurement update.

For prediction, the time update step is applied without changing the state. In order to determine the process noise matrix Q, we use a non-linear minimization method (simplex), similar to what is used in [3]. The function to be minimized is the sum of differences between the predicted pose and the actual tracker measurement, computed over a recorded sequence of gyroscope and absolute tracker measurements. Note that this minimization does not always converge to the same minimum. Therefore, the initial value must be chosen carefully. For optimal performance, the covariance matrices of the individual tracker measurements should be known. A method to compute them is described in [5]. The gyroscope covariances are set to a very low value.

6 Algorithmic Evaluation

In order to evaluate whether the distinction between the inside-out and outside-in cases is necessary, we performed an experiment, for which we used an Xsens MT-9 inertial sensor. The magnetometers were turned off, as we found that the magnetic field distortions introduce additional error. Without the magnetometers, the sensor has a bit of drift, but this is easily compensated by the fusion algorithm.

The hardware for this experiment consisted of a Logitech QuickCam Pro 4000 USB web cam which was rigidly connected to the inertial sensor. The camera was calibrated using OpenCV and ran a square marker tracker also based on OpenCV, similar to the AR Toolkit. Our marker tracker provides a covariance matrix with every measurement which was used in the measurement update of the Kalman filter. The camera ran at a rather low frame rate of 15Hz.

To get comparable results for the different fusion methods, we recorded time-stamped measurements of the inertial sensor and the marker tracking and ran the fusion algorithms off-line. Four different camera motions were used:

- a "still" sequence where the camera was held by hand in a position where the marker and the camera sensor were approximately parallel. In this case, the error in the measured rotation is maximal. This is confirmed e.g by [14].
- a "slow rotation" sequence where the camera looked at the marker in an angle of about 45 deg and was rotated such that no motion blur occurred and the marker tracker was able to follow the marker.
- a "fast rotation" sequence which was similar to the slow rotation, but with many fast rotations which caused the marker tracker to frequently loose the marker.
- a "full motion" sequence, where the camera was both rotated and translated.

For quantitative results, we calculated how well the Kalman filter was able to predict the next measurement of the marker tracker. The translational error was converted to screen pixels (320×240 resolution) using the known calibration matrix. The rotational error was directly computed in degrees. We compared four different prediction methods:

- The outside-in motion model, evaluated on the transformed gyroscope rotation and the inverted (with covariance propagation) marker pose.
- The outside-in motion model, but without the gyroscope.
- The inside-out motion model, where the Kalman filter state directly corresponds to the marker pose and the gyroscope measurements are inverted.
- The inside-out motion model without the gyroscope data.

	outside-in		inside-out	
	w/ gyro	w/o gyro	w/ gyro	w/o gyro
still	22.8	68.0	1.2	1.2
slow rotation	2.6	5.6	3.0	5.2
fast rotation	9.1	94.1	10.8	91.0
full motion	6.5	11.0	7.3	9.3

Table 1: Average prediction error in pixels

	outside-in		inside-out	
	w/ gyro	w/o gyro	w/ gyro	w/o gyro
still	5.2	10.0	3.3	4.5
slow rotation	0.53	1.0	0.51	1.0
fast rotation	1.2	11.0	1.2	9.6
full motion	0.91	2.0	0.87	2.0

Table 2: Average angular prediction error in degrees

We computed different tuning parameters for each of the four cases, using the "full motion" sequence.

The results are shown in tables 1 and 2. The advantage of using a gyroscope for stabilization is clear, especially for the fast rotations. Both rotational and positional prediction accuracy is improved. The inside-out and outside-in motion models produce comparable results is most cases, with the exception of the "still" sequence where the marker tracking has a very bad rotational accuracy. Here the inside-out motion model has a clear advantage.

7 Dynamic Gyroscope Fusion

After the more theoretical description of the algorithms used for transformation, calibration and fusion of gyroscope data, we come to an actual application in a real Ubitrack system. The general architecture of our system is presented in [8], and in this section we will focus on describing the concrete system instance as well as the reconfiguration mechanism that was added.

7.1 Setup

Our scenario consists of a mobile user, equipped with some tracking hardware, who is standing in the hallway in front of our lab. Using his camera, the user can track a marker in front of the door and see a virtual sign inviting him to come in. Looking in through the door, he can already see an augmentation, a virtual sheep, standing in the middle of the lab. As he is entering the lab, the marker tracking becomes unavailable, but the outside-in tracking inside the lab is – transparently to the application – taking over. A map of the situation and a picture of the mobile setup is shown in figure 11.

The hardware setup is centered around an Xsens MT-9 inertial sensor, which is dynamically fused with three other tracking systems:

- The first tracker is a Ubisense wide-area radio frequency location system that covers the whole lab. It gives position updates at a frequency of about 1Hz with an accuracy of about 15cm. As no orientation is provided, the "fusion" consists of combining the magnetometer-stabilized orientation from the inertial sensor with the 3D-position from the Ubisense system. Due to magnetic field distortions, the resulting orientation is accurate to about 10 deg. The Ubitag (RF emitter) is the black box in the middle of picture 11.

- In the center of the lab we have installed a high-precision ART infrared-optical tracker, which covers an area of about 4×4 meters using three cameras. In order to be tracked by this system, the mobile setup includes a set of retro-reflective marker balls. Fusion of the ART with the gyroscope is performed using the outside-in fusion technique described in section 5.2. A 6×6 covariance matrix for each measurement is computed using the method described in [5].

- Tracking outside the lab is provided by our own marker tracking algorithm, which works similar to the AR Toolkit. The most notable difference is that our system provides a 6×6 covariance matrix that describes the accuracy of each measurement. This information is used in the inside-out fusion algorithm. Note that the firewire camera on the mobile setup is used both for marker tracking and to provide video-see-through AR during the entire demonstration.

The spatial relationship graph of the setup is shown in figure 10. Note that the edges representing the ART data, the Ubisense position and the marker tracker can be unavailable, depending on the current situation.

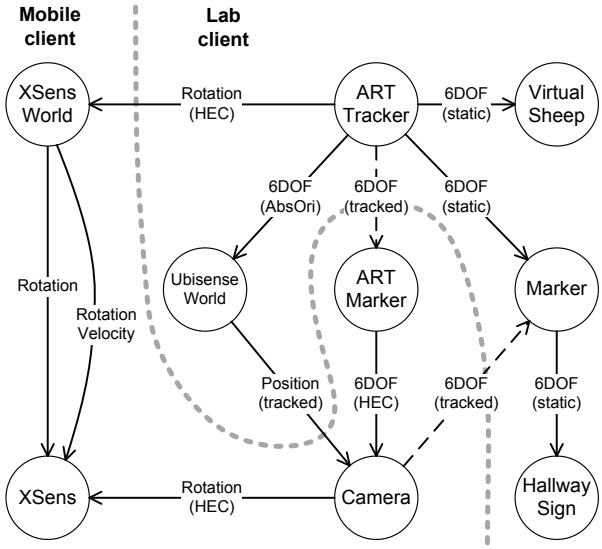


Figure 10: Spatial Relationship Graph of the dynamic fusion setup

7.2 Software Architecture

Our software system, shown in figure 12 consists of a Ubitrack server and three different clients. In detail, the components fulfill the following functions:

Ubitrack server The central component of our system is the Ubitrack server, which coordinates the various clients. After startup, the internal SRG of the server is empty, as is its list of patterns. When a client connects, it sends the server a list of the SRG fragments it knows about, and a list of patterns. As each pattern is associated with a data flow component available at the client, this list of patterns essentially describes the processing capabilities of the client. Additionally, the client can send a set of queries for nodes and/or edges.

From the list of partial SRGs sent by the clients, the server assembles the global SRG and applies the patterns to infer new edges. When a query matches an edge, the server generates a data flow description by descending the tree of dependent edges. If multiple clients are involved, the data flow is distributed among these clients. This way, one client can be instructed to send data to another client to fulfill its queries. Note that the communication of tracking data at runtime is done in a peer-to-peer fashion, rather than over the server, in order to reduce the system's latency. As the communication is coordinated by the server, we call this a "centrally coordinated peer-to-peer architecture".

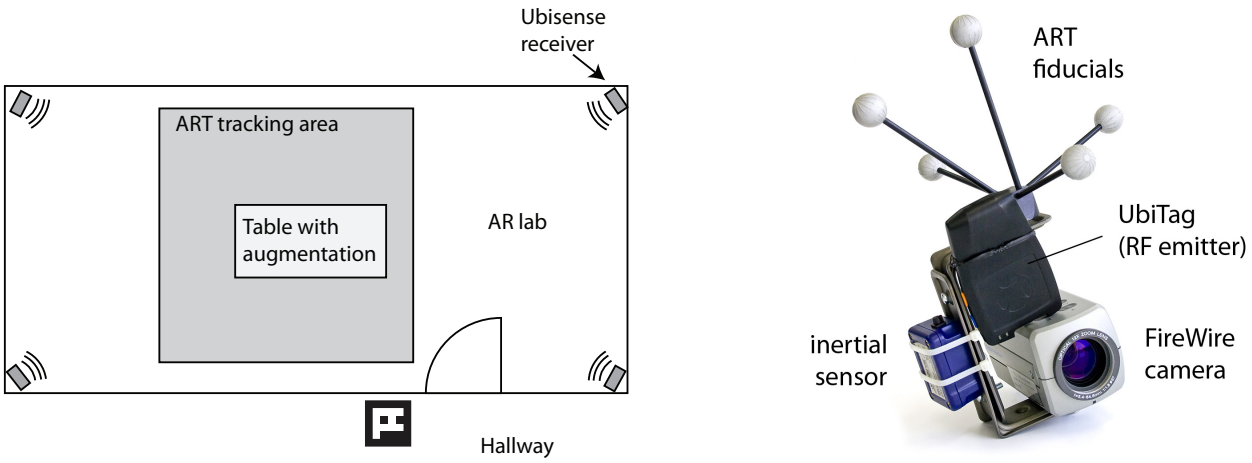


Figure 11: Map of our lab and picture of the mobile tracking setup

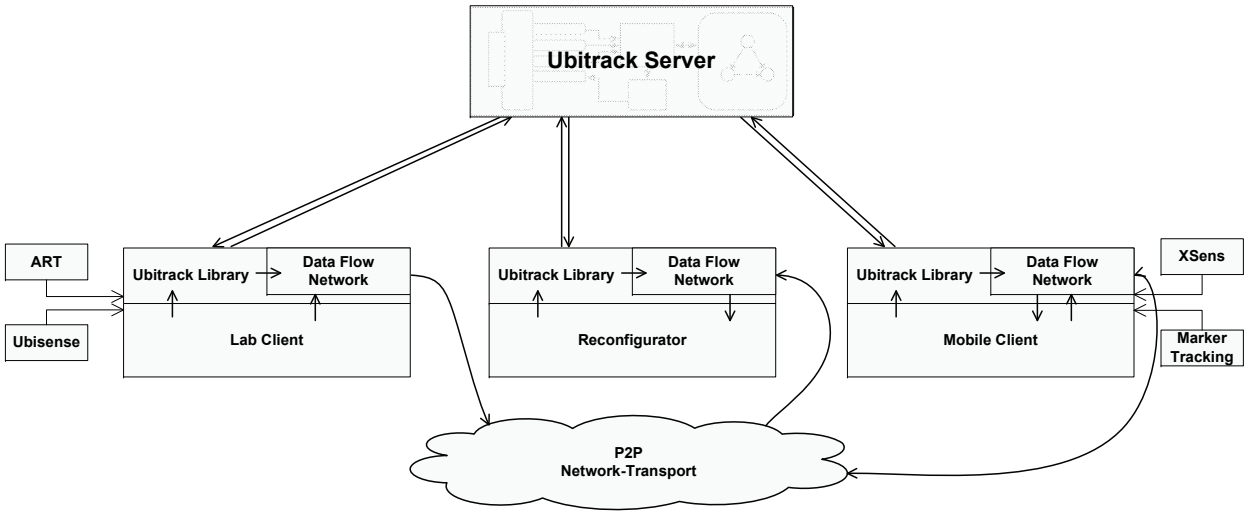


Figure 12: Software architecture of the dynamic gyroscope fusion setup

Mobile client The mobile client is responsible for the SRG that describes the mobile setup (left of the gray dashed line in figure 10), which it sends to the server at startup. As the mobile client contains the rendering "application", it also sends a query for the pose of all renderable objects wrt. to the camera coordinate frame. In the concrete setup, these are the "virtual sheep" and the "hallway sign" objects. When the client receives a data flow description as a response from the server, it is instantiated and the resulting tracking information is used to overlay the virtual objects onto the camera image.

Lab client This client runs on a computer in the lab and knows about the ART and Ubisense trackers, as well as the virtual objects and some static transformations in between (right of the gray dashed line in the SRG). It is a pure tracking client, and thus has no queries of its own.

Reconfigurator Given only the server and the mobile and lab clients, the system would not be able to present any augmentations. As the actual tracking edges connecting the mobile setup with the lab part do not belong to either of the above clients, they are not part of the initial global SRG. Theoretically, it would be possible to make the mobile setup query the system for all "marker" objects (nodes), and similarly, the lab clients could query for all infrared fiducial nodes and add the respective edges to the SRG. However, adding edges to all available markers/fiducials would cause the server to assume that they are actually visible to the tracker and start using them in data flows. In a larger setup with multiple rooms and many markers, this would waste processing time and possibly cause confusions if fiducials are not globally unique. Also, the server could prefer an unavailable tracker to an available but less a accurate one. Therefore, an edge should only be added to the SRG when there is a good chance of actually receiving tracking data from it.

In our Ubitrack system, we solve this by assuming that there is always enough information available to at least decide which fiducials are close to a given sensor. This kind of coarse tracking information could come e.g. from a WLAN tracking system. In our case, we are using the Ubisense for this task.

The reconfigurator client monitors the data from the Ubisense system and, based on this information, adds or removes edges in the SRG. The reconfigurator is given an XML configuration file with a number of regions that can be defined by either the convex hull of a set of points or by presence/absence of tracking data. Depending on whether the client is inside or outside a region, SRG edges are added or removed. In the dynamic reconfiguration scenario, we use one "lab" region that is defined by availability of Ubisense data and

one smaller "ART" region inside the lab.

7.3 Calibration and Results

In order to align the various sensors with each other, different calibration methods have been applied. The gyroscope was aligned with the camera using the rotation-only hand-eye-calibration (HEC) described in section 4. After that, a traditional offline hand-eye-calibration algorithm was used to determine the transformation between the camera and the ART marker. In order to align the Ubisense with the ART system, we applied an absolute orientation algorithm on data gathered by moving around a target trackable by both systems. As the world coordinate frame of the XSens is defined by gravity and the magnetic field, we were able to compute the static rotation between the XSens and the ART using the online hand-eye-calibration algorithm again.

The system can be seen in action in the accompanying video. It starts with the mobile setup standing in the hallway. The marker tracking is fused with the gyroscope to display a welcome sign. Because of the fusion, the orientation tracking continues even when the marker is not visible. In the SRG, the marker is connected to the visualization inside the lab, and the user can look through the door to see it. When the user enters the lab, the Ubisense system starts tracking, the marker tracking edge is removed from the SRG, and the server reconfigures the data flow to use the Ubisense system together with the compass-stabilized gyroscope. As the mobile setup enters the ART tracking region, the reconfigurator client detects this from the Ubisense data and adds an ART edge to the SRG. The server reacts to this by sending the mobile client a new data flow, now fusing the ART with the gyroscope. The robustness of the fusion is again demonstrated by putting a plastic bag over the infrared fiducial. In this case, the inertial unit still tracks the orientation, but no positional updates are available. As the user leaves the lab, the whole process runs in reverse order.

8 Conclusion

In this paper we have shown that the Ubiquitous tracking tool set, consisting of spatial relationship graphs and patterns is very useful to analyze tracking setups including gyroscopes. It allows a Ubitrack system to automatically infer occasions for gyroscope fusion in dynamically changing tracking situations. We have derived new patterns that allow the transformation of incremental rotation and rotation velocity between different coordinate frames, which enable a deeper understanding of the situation. By looking at the SRG it became clear that the gyroscope alignment is related to the well-known hand-eye calibration problem, for which we presented a recursive solution based on the rotation part of the Tsai-Lenz algorithm. For the fusion of absolute tracking with the gyroscope, an extended Kalman filter implementation was described, with different motion models and measurement equations for the inside-out and outside-in case. This was first evaluated in a small experiment and then applied in a real Ubiquitous tracking system, where the gyroscope was dynamically fused with three different other trackers as a user moved from one room into another, based only on the available SRG structure.

Acknowledgements

This work was supported by the Bayerische Forschungsstiftung (project trackframe, AZ-653-05) and the PRESENCCIA Integrated Project funded under the European Sixth Framework Program, Future and Emerging Technologies (FET) (contract no. 27731).

References

[1] L. Armesto, S. Chroust, M. Vincze, and J. Tomero. Multi-rate fusion with vision and inertial sensors. In *Proceedings of the 2OO4 IEEE International Conference on Robotics & Automation*, 2004.

[2] M. Aron, G. Simon, and M.-O. Berger. Handling uncertain sensor data in vision-based camera tracking. In *Proc. of the Third International Symposium on Mixed and Augmented Reality (ISMAR'04)*, 2004.

[3] R. Azuma and G. Bishop. Improving static and dynamic registration in an optical see-through hmd. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 197–204, New York, NY, USA, 1994. ACM Press.

[4] Y. Baillot, S. Julier, D. Brown, and M. Livingston. A tracker alignment framework for augmented reality. In *Mixed and Augmented Reality, 2003. Proceedings. The Second IEEE and ACM International Symposium on*, pages 142–150, 2003.

[5] M. Bauer, M. Schlegel, D. Pustka, N. Navab, and G. Klinker. Predicting and estimating the accuracy of vision-based optical tracking systems. In *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'06)*, Santa Barbara (CA), USA, October 2006.

[6] E. Foxlin. Inertial head-tracker sensor fusion by a complementary separate-bias kalman filter. In *Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996*, pages 185–194, 267, 1996.

[7] D. Hallaway, T. Hoellerer, and S. Feiner. Bridging the gaps: Hybrid tracking for adaptive mobile augmented reality. *Applied Artificial Intelligence, Special Edition on Artificial Intelligence in Mobile Systems*, 25(5), July 2004.

[8] M. Huber, D. Pustka, P. Keitler, E. Florian, and G. Klinker. A System Architecture for Ubiquitous Tracking Environments. In *Proceedings of the 6th International Symposium on Mixed and Augmented Reality (ISMAR)*, Nov. 2007.

[9] S. Julier and J. Uhlmann. A new extension of the kalman filter to nonlinear systems. In *Int. Symp. Aerospace/Defense Sensing, Simul. and Controls*, Orlando, FL, USA, 1997.

[10] G. S. W. Klein and T. W. Drummond. Tightly integrated sensor fusion for robust visual tracking. *Image and Vision Computing*, 22(10):769–776, Sept. 2004.

[11] K.-R. Koch. *Parameterschätzung und Hypothesentests in linearen Modellen*. Dümmler, Bonn, 1980.

[12] J. B. Kuipers. *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*. Princeton University Press, 2002.

[13] J. Newman, M. Wagner, M. Bauer, A. MacWilliams, T. Pintaric, D. Beyer, D. Pustka, F. Strasser, D. Schmalstieg, and G. Klinker. Ubiquitous tracking for augmented reality. In *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'04)*, Arlington, VA, USA, Nov. 2004.

[14] K. Pentenrieder, P. Meier, and G. Klinker. Analysis of tracking accuracy for single-camera square-marker-based tracking. In *Proc. Dritter Workshop Virtuelle und Erweiterte Realitt der GI-Fachgruppe VR/AR*, Koblenz, Germany, September 2006.

[15] D. Pustka. Construction of data flow networks for augmented reality applications. In *Proc. Dritter Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR*, Koblenz, Germany, September 2006.

[16] D. Pustka, M. Huber, M. Bauer, and G. Klinker. Spatial relationship patterns: Elements of reusable tracking and calibration systems. In *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'06)*, October 2006.

[17] M. Ribo, P. Lang, H. Ganster, M. Brandner, C. Stock, and A. Pinz. Hybrid tracking for outdoor augmented reality applications. *Computer Graphics and Applications*, 22(6):54–63, Nov/Dec 2002.

[18] K. Satoh, M. Anabuki, H. Yamamoto, and H. Tamura. A hybrid registration method for outdoor augmented reality. In *Augmented Reality, 2001. Proceedings. IEEE and ACM International Symposium on*, pages 67–76, 2001.

[19] R. Tsai and R. Lenz. Real time versatile robotics hand/eye calibration using 3d machinevision. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 554–561, 1988.

[20] G. Welch and G. Bishop. Course 8 – An introduction to the Kalman filter. In *SIGGRAPH 2001 Courses*, 2001.

[21] S. You, U. Neumann, and R. Azuma. Hybrid inertial and vision tracking for augmented reality registration. In *Virtual Reality, 1999. Proceedings., IEEE*, pages 260–267, 1999.