Institut für Informatik der Technischen Universität München

Management of Tracking
and Tracking Accuracy in Industrial Augmented Reality Environments

Dissertation

Peter Keitler

Institut für Informatik der Technischen Universität München

Management of Tracking and Tracking Accuracy in Industrial Augmented Reality Environments

Peter Keitler

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender:		Univ.-Prof. Dr. N. Navab
Prüfer der Dissertation:	1.	Univ.-Prof. G. J. Klinker, Ph.D.
	2.	Prof. G. F. Welch, Ph.D., University of North Carolina / USA

Die Dissertation wurde am 27.01.2011 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 25.03.2011 angenommen.

Abstract

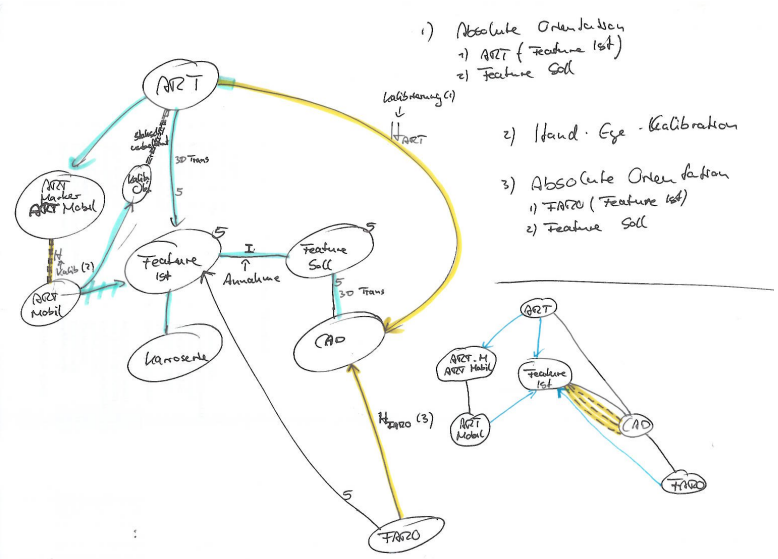
Industrial Augmented Reality (IAR) techniques can help to increase the productivity of certain work processes by augmenting the physical scene with virtual information. IAR brings together tasks that are traditionally performed off-line on purely virtual data, such as industrial design, geometric layout, or metrologic evaluations, with on-line tasks in the domain of physical objects such as prototypes, mockups, facilities or repetition parts. This allows for a broad range of new, integrated applications. Real-time position and orientation tracking of physical objects is needed to register these objects with the virtual world. It can be implemented based on a variety of sensors providing spatial measurements on an optical, inertial, or acoustic basis. However, tracking for IAR is often difficult to implement, due to the constraints imposed by the working environment such as electro magnetic interference, dirt/dust, noise, bad illumination conditions, vibrations, occlusion, and interference with existing work processes. Sometimes, the flexibility to quickly setup and dismantle the system is required. Nevertheless, the tracking has to be robust and meet the accuracy requirements imposed by the intended application. Often, this can only be accomplished by a heterogeneous multi-sensor tracking environment, a fact that complicates the task of registering the various coordinate frames of tracking systems, sensors, markers, and display devices with respect to one another.

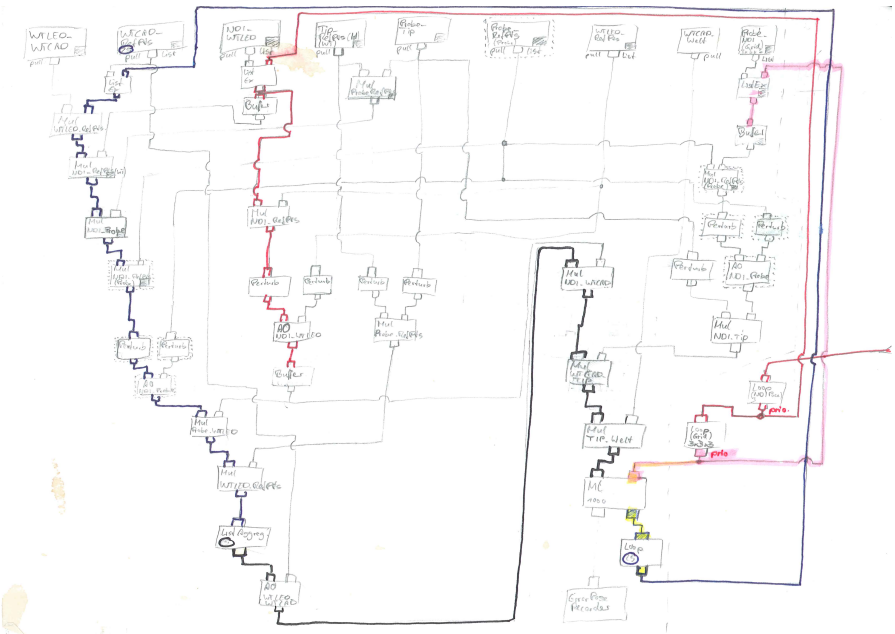
This thesis describes a generic approach to deal with the complexity of heterogeneous tracking environments. It supports the IAR engineer throughout the various design and implementation phases of an industrial IAR scenario. An abstract semantic modeling concept based on spatial relationship graphs (SRG) and its implementation in a graphical data flow editor is presented. Modeling is based on reusable design patterns which represent atomic sensor drivers and geometric algorithms as well as complex state-of-the- art solutions. Real-time data flow networks can be generated automatically from the SRG and are guaranteed to be semantically correct. Still, the data flow layer remains directly accessible through a round-trip engineering approach. Based on the SRG concept, simulation and analysis tools for a rigorous management of tracking accuracy are described. Monte Carlo simulation helps IAR engineers to understand the proposed system and to identify critical design issues, even before actual hardware deployment. Expert analysis tools help them in system validation and maintenance where simulation results have to be confirmed based on real measurement data. End-user analysis tools facilitate recurring quality checks by on-site personnel during regular system operation. The system has been implemented and used successfully in two real industrial settings. The described approach simplifies and standardizes the setup, operation, and maintenance of such IAR tracking environments.

Zusammenfassung

Techniken der Erweiterten Realität können die Produktivität von industriellen Arbeitsprozessen erhöhen, indem die Sicht des Nutzers mit virtuellen Informationen angereichert wird. Dazu muss Position und Orientierung realer Objekte in Echtzeit bestimmt werden, um sie mit der virtuellen Welt registrieren zu können. Für dieses Objekt-Tracking werden verschiedene Messprinzipien, insbesondere optische, inertiale oder akustische, verwendet. Oft ist eine Kombination verschiedener Verfahren notwendig, um die angestrebte Robustheit und Genauigkeit zu erreichen.

Diese Arbeit beschreibt einen generischen Ansatz zum Umgang mit heterogenen Tracking-Umgebungen, basierend auf räumlichen Abhängigkeitsgraphen. Das graphische Modellierkonzept vereinfacht den Umgang mit statischen und dynamischen Abhängigkeiten erheblich. Darauf aufbauend werden Simulations- und Analysewerkzeuge beschrieben, die einen umfassenden Umgang mit Tracking-Genauigkeit erlauben.


1) Absolute Orientation
1) ART (Feature Ist)
2) Feature Soll
2) Hand-Eye-Kalibration
3) Absolute Orientation
1) FARO (Feature Ist)
2) Feature Soll
ART
ART Marker ART Mobil
ART Mobil
Feature Ist
Feature Soll
Kamerale
CAD
FARO
3D Trans
Annahme



Acknowledgment

Ganz herzlich danken möchte ich Gudrun Klinker, für die aufopferungsvolle wissenschaftliche Betreuung in den vergangenen vier Jahren und für unzählige wertvolle Denkanstöße. Ich schätze das Vertrauen sehr, das Du in uns setzt.

Sincere thanks also go to Greg Welch who agreed to become my second supervisor without hesitating. Your feedback was very valuable for me.

Danken möchte ich auch Georg Klinker, der es mir ermöglichte, im industrienahen Umfeld zu promovieren, ohne dabei den Anschluss an die universitäre Forschung zu verlieren. Du hast mir gezeigt, den Blick auf das Wesentliche zu lenken.

Besonderer Dank gilt meinen hochgeschätzten Kollegen aus der Industrie. Michael Schlegel hat unzählige Stunden und Tage mit mir im Versuchsfahrzeugbau bei BMW verbracht. Diese wertvollen Erfahrungen ("Wer misst, misst Mist!") legten erst den Grundstein für ein strukturierteres Vorgehen. Das gemeinsame Auswerten der Messdaten war zuweilen frustrierend, hat aber auch viel Spass gemacht. Besonderer Dank auch an Benjamin Becker von EADS Innovation Works, der mich ebenfalls des öfteren mit praxisrelevanten Anforderungen versorgte. Seine Fähigkeit zur direkten Kritik schätze ich sehr. Die Zusammenarbeit mit Euch möchte ich nicht missen.

Des Weiteren möchte ich all jenen danken, die mit ihrer Mitwirkung an *Ubitrack* die Basis für diese Arbeit gelegt haben. Dank gilt insbesondere Florian Echtler, Manuel Huber und Daniel Pustka. Ein Dankeschön auch an alle weiteren Kolleginnen und Kollegen vom FAR. Nicht nur das Schlittenfahren hat mit Euch sehr viel Spass gemacht! Diese Kollegialität ist nicht alltäglich.

Danke auch an Martin Groher, Jörg Traub und Marco Feuerstein, die mit mir das Hauptseminar Augmented Reality besucht haben, was mich überhaupt erst auf das Thema AR gebracht hat. Die Gruppen-Diplomarbeit mit Euch und unter Gudruns Betreuung ist für mich noch immer der Inbegriff von gutem Teamwork.

Ich möchte auch meinen Eltern Inge und Josef danken, die es mir erst ermöglicht haben, ein Hochschulstudium aufzunehmen. Ein Privileg, dass eigentlich keines sein sollte. Danke auch für die zuverlässige Grundversorung während arbeitsintensiver Wochen auf dem Lande.

Ein ganz besonderer Dank gilt Agnes, für das in mich gesetzte Vertrauen, für das Ertragen von entbehrungsreichen Monaten, sowie für die Zubereitung schier unglaublicher Mengen grünen Tees. Ich liebe Dich!

Contents

Abstract i

Zusammenfassung ii

Acknowledgment iv

Overview x

I. Industrial Augmented Reality 1

1. Motivation 2
- 1.1. Mixed- and Augmented Reality . . . 2
  - 1.1.1. Architecture of an Augmented Reality Setup . . . 2
  - 1.1.2. Taxonomy of Mixed Reality Systems . . . 3
- 1.2. Prerequisites for Industrial Augmented Reality . . . 3
  - 1.2.1. Virtual Content . . . 4
  - 1.2.2. Visualization Techniques . . . 6
  - 1.2.3. Interaction . . . 8
  - 1.2.4. Tracking Systems . . . 10
- 1.3. Importance of Augmented Reality in Industry . . . 14
  - 1.3.1. Basic Benefit of Augmented Reality Techniques . . . 14
  - 1.3.2. Applications . . . 15
  - 1.3.3. Relations to Classical Mixed- and Augmented Reality . . . 21
  - 1.3.4. Relations to Virtual Reality . . . 22
  - 1.3.5. Relations to Metrology . . . 22
- 1.4. The Tracking Challenge . . . 23

2. Tracking to Meet Industrial Criteria 25
- 2.1. Requirements in the Literature . . . 25
- 2.2. IAR Design Phases . . . 26
- 2.3. Requirements for Management of Tracking . . . 27
  - 2.3.1. Guaranteed Performance . . . 27
  - 2.3.2. Sensor Fusion . . . 27
  - 2.3.3. Modularization . . . 29
  - 2.3.4. Maintainability . . . 29

3. Related Work **31**

4. Approach **34**
4.1. General Approach . . . 34
4.2. Focus . . . 35
4.2.1. Relations to other Aspects of IAR Applications . . . 35
4.2.2. Sensor Fusion . . . 35
4.2.3. Dynamic Reconfiguration of Sensors . . . 35
4.2.4. Dynamic State Changes . . . 36
4.2.5. Relations to Metrology . . . 37
4.3. Contribution . . . 38

II. Data Flow Management **39**

5. The Ubitrack Tracking Middleware **41**
5.1. Spatial Relationship Graphs . . . 42
5.1.1. Definition . . . 42
5.1.2. Edge Characteristics . . . 43
5.1.3. Related Concepts . . . 44
5.2. Data Flow Networks . . . 45
5.3. Spatial Relationship Patterns . . . 47
5.3.1. Definition . . . 47
5.3.2. A Catalogue of Spatial Relationship Patterns . . . 50
5.3.3. Node/Edge/Pattern Attributes . . . 54
5.3.4. The Module Mechansim . . . 56
5.3.5. Time-/Space-Expansion . . . 56
5.4. Data Flow Synchronization . . . 58
5.4.1. Synchronization of Data Flow Components . . . 58
5.4.2. Synchronization of the Data Flow Network . . . 59
5.4.3. The Time-Synchronization Problem . . . 62
5.5. SRG Design Activities . . . 64

6. Graphical Data Flow Modeling **66**
6.1. Graphical Layout . . . 66
6.2. trackman Architecture . . . 66
6.3. Interactive SRG generation . . . 68
6.4. Interactive Deduction of Spatial Relationships . . . 69
6.4.1. Edge Matching . . . 69
6.4.2. Ordering of Design Activities . . . 70
6.4.3. Round-Trip Engineering in SRG and DFG . . . 70
6.4.4. Automatic Sync Propagation . . . 71
6.5. trackman Editor Functionality . . . 76

7. Common Modeling Tasks **79**
7.1. Application SRG . . . 79
7.1.1. Example: Indirect Tracking for the Intelligent Welding Gun . . . 79
7.1.2. Example: Discrepancy Checks in the Airplane Cabin . . . 81
7.2. Calibration and Registration . . . 82
7.2.1. Basic Solution Patterns . . . 82
7.2.2. Example: Hand-Eye Calibration for Indirect Tracking Setup . . . 86
7.2.3. Example: Registration of the Reference Target in the Airplane Cabin . . . 86
7.3. Application Interfaces . . . 86

8. Advanced Graphical Modeling Concepts **89**
8.1. Semi-Automatic Modeling . . . 89
8.2. Meta-Patterns . . . 90
8.2.1. Definition . . . 91
8.2.2. Applications . . . 91
8.2.3. Integration in trackman . . . 92

9. Discussion **94**
9.1. Summary . . . 94
9.2. Advantages and Limitations of Graphical SRG Modeling . . . 94
9.3. Relationship between trackman and Dynamic SRG Modifications . . . 95
9.4. Possible Improvements . . . 95

III. Error Management **97**

10. Quantifying Measurement Uncertainties **99**
10.1. Representation and Categorization of Uncertainties . . . 99
10.1.1. Categorization(s) of Errors . . . 99
10.1.2. Basic Error Statistics . . . 103
10.1.3. Root-Mean-Square Error . . . 104
10.1.4. Expressing Orientation and Pose Error . . . 106
10.2. Error Standards for Spatial Measurements . . . 107
10.3. Propagation of Uncertainties . . . 109
10.3.1. Short Survey of Parameter Estimation . . . 110
10.3.2. Propagation of Uncertainties in the Literature . . . 111
10.3.3. Linear Propagation of Uncertainties . . . 113
10.3.4. Monte Carlo Simulation . . . 117
10.3.5. Elementary Specification of Uncertainty . . . 118
10.4. Conquering the Complexity of IAR Applications . . . 119
10.4.1. Proposed Approach . . . 120
10.4.2. Focus . . . 122
10.4.3. Outline . . . 123

11. Ubitrack Monte Carlo Simulation Framework **124**
11.1. The Chain of Uncertainty . . . 124
11.2. Simulation Data Flow . . . 125
11.2.1. Ground Truth Data . . . 126
11.2.2. Synthetic Measurements . . . 127
11.2.3. Registration/Tracking Algorithm: . . . 128
11.2.4. Covariance Estimation . . . 128
11.2.5. Data Flow Synchronization . . . 130
11.3. Computational Complexity . . . 132
11.4. Example: Comparison of Monte Carlo with Online Error Propagation . . . 132

12. Sensor Errors **137**
12.1. Assessment of Uncertainties in the Literature . . . 137
12.2. Elementary Specifiation of Uncertainties for Sensors . . . 138
12.3. Residual Error from Point-Based Registration . . . 140
12.3.1. General Non-Linear Solution: Helmert Transformation . . . 141
12.3.2. A Linear Approximation . . . 142
12.3.3. Practical Considerations . . . 149
12.4. Example: Specification of Elementary Uncertainty for the Airplane Cabin 152
12.4.1. Setup . . . 153
12.4.2. Noise . . . 154
12.4.3. Systematic Effects . . . 155
12.4.4. Indirect Tracking Experiment . . . 159
12.4.5. Summary . . . 161

13. Verification & Validation **162**
13.1. Example: Indirect Tracking of the Intelligent Welding Gun . . . 162
13.1.1. Correcting Rotational Errors . . . 162
13.1.2. In-vitro Validation of Indirect Tracking . . . 168
13.1.3. In-situ Validation of Application . . . 172
13.1.4. Verification of Mobile Tracking Setup . . . 176
13.1.5. Verification of Application . . . 180
13.1.6. Discussion . . . 181
13.2. Example: Probe Tracking in the Airplane Cabin . . . 183
13.2.1. Verification . . . 183
13.2.2. Validation . . . 187
13.2.3. Discussion . . . 187

14. Runtime Error Mitigation **189**
14.1. Robustifying Calibration & Registration Procedures . . . 189
14.2. Example: Loops in the Spatial Relationship Graph . . . 190
14.3. Example: Error Mitigation in the Airplane Cabin . . . 192

15. Discussion **194**

IV. Conclusion **196**

16. Fulfillment of Tracking Tasks **198**
16.1. Guaranteed Performance 198
16.2. Sensor Fusion 198
16.3. Modularization 199
16.4. Maintainability 199

17. Compliance with Industrial AR Design Guidelines **201**

Appendix **203**

A. SRGs and DFGs **204**
A.1. Application 204
A.2. Registration 209
A.3. Simulation 213

Glossary **239**

Acronyms **249**

Overview

A basic prerequisite for augmented reality is tracking, i.e., the determination of the position and orientation of physical objects in space. It is accomplished with the aid of various sensors in a commonly heterogeneous sensor infrastructure. It is crucial to provide means to setup and implement a robust and maintainable tracking infrastructure to allow AR applications to spread in the industrial domain. This thesis is split into four parts.

Part I approaches augmented reality from the point-of-view of industrial applications. Promising scenarios are discussed first. From the discussed scenarios special requirements towards tracking setups in industrial augmented reality (IAR) are be derived. The ultimate goal is a standardized method that allows for the efficient implementation of robust and reliably accurate IAR solutions.

The next two parts represent the main contribution of this thesis. Part II elaborates on a graphical method that eases the creation of tracking data flow networks that are necessary to process the measurements of all involved sensors efficiently in real-time. Typical tracking problems are analyzed and reusable best-practice solution patterns are provided. In particular, common registration methods are analyzed that are needed to align sensors and objects in the environment. The focus is on a generic approach that is applicable in various different scenarios.

Part III tackles the problem of tracking accuracy. It is an elementary problem for most IAR scenarios. The goal is to be able to guarantee a certain level of accuracy for reliable operation. A generic concept is developed. It combines verification by simulation with validation by empiric measurements in the target environment. By means of two exemplary scenarios based on optical infrared tracking, the feasibility of the chosen approach is demonstrated.

Finally, Part IV concludes the thesis. The chosen generic approach is reviewed according to the initial problem statement given in Part I.

Part I.

Industrial Augmented Reality

1. Motivation

Generally, Industrial Augmented Reality (IAR) encompasses the integration of Augmented Reality (AR) techniques into work processes with the goal of making the latter more reliable, accurate, and efficient. This thesis deals with tracking, the real-time determination of position and orientation of objects in space. It is an important prerequisite for AR and thus also for IAR applications.

In principle, *tracking* has to be distinguished from *detection* or *localization*. An object can only be tracked continually after it has once been detected successfully. However, this distinction is ignored in this context. Due to the common terminology in AR, tracking includes detection and pose estimation.

Unfortunately, the proper setup of a realiable tracking system is a rather difficult problem. To understand the tracking problem, the context of AR/IAR is clarified first. Section 1.1 outlines the basic ideas behind Mixed-/Augmented Reality. Next, the different subsystems necessary to build an IAR application are detailed in Chapter 1.2. The potential of these ideas is then reflected and refined in the context of IAR applications in Section 1.3. Based on this, Section 1.4 describes the various difficulties that arise when trying to setup tracking for an IAR application. In this thesis, generic methods are developed that simplify the implementation of robust and reliably accurate tracking environments.

1.1. Mixed- and Augmented Reality

Augmented reality (AR) aims at augmenting the user's perception of the real world by integrating additional, virtual information.

1.1.1. Architecture of an Augmented Reality Setup

An AR application is generally composed of different subsystems[Azum 97]. A world model represents the *virtual content* to be mixed with the real scene. A suitable *visulization system* is needed to mix both modalities by overlaying the virtual information with the current perspective of the real world. A *tracking system* is needed to update the rendering of the overlay with respect to movements of users or physical objects or both, particularly the users head or the display he is using. Finally, an AR system also has means for *interaction* with the virtual scene. These components are detailed in 1.2.

1.1.2. Taxonomy of Mixed Reality Systems

The often-cited definition of AR was given by Azuma in 1997. It lists three indispensable requirements for an AR application [Azum 97].

- The AR system combines real and virtual information
- The AR system is interactive in real-time
- Virtual content is registered in 3D

Although there might be other means to augment the user's reality, in the industrial context, nowadays, mainly visual information is relevant. Milgram introduced another definition in 1994, which has a different, more general viewpoint [Milg 94]. It defines a "mixing ratio" between real and virtual information. Pure reality and pure virtuality (also known as virtual reality, VR) represent two extreme values and span a continuum also known as Milgram's *Reality-Virtuality Continuum*. In between those two extremes, mainly two intervals can be constituted: *augmented reality* (AR) and *augmented virtuality*. Whereas in AR, some virtual information is incorporated into the otherwise real world, in augmented virtuality, some real information is incorporated into the otherwise virtual world. Both definitions form a general concept commonly called *mixed reality*. Extensions to this taxonomy of mixed reality exist. The "degree of modification of reality or virtuality", also called *mediality*, forms another continuum which is orthogonal to the reality-virtuality continuum [Mann 02]. Again, pure reality is one of the extreme values. On the other extreme, there is a purely imaginary world, having nothing to do with reality. In between, reality is changed to some degree. This also incorporates the term *diminished reality*, where information existing in reality is absent in the mixed perspective. A third, orthogonal continuum can be constructed by considering the "degree of ubiquity" of the application, ranging from monolithic mainframe computing to distributed pervasive computing, with classic PC, or wearable systems in between [Newm 07].

Milgram's continuum is quite helpful to classify industrial MR applications. Both, AR and VR play a major role, as shown next. The two continua orthogonal to Milgram's do not yet play a major role in the classification of industrial MR applications. There currently is not much interest to change reality towards an imaginary world in industrial applications. This might change in the future after IAR has become commonplace, e.g., to increase the level of comfortness for workers in exposed situations [Tonn 09]. Currently the goals are more concrete. Concerning ubiquity, industrial applications are still situated rather on the mainframe computing end of the continuum, but maybe this will change in the future.

1.2. Prerequisites for Industrial Augmented Reality

In this section, an overview of the subsystems needed for an IAR system is given. This resembles the outline of a general AR application shown in Chapter 1. Since in this

thesis, the focus is on tracking, the other components are treated only to the minimal extent that is necessary to understand the implications on questions of tracking. This overview also touches some important usability aspects that arise with the introduction of an IAR system in an already existing work process. More information about the individual subsystems necessary for IAR in general is given by Pentenrieder and by Becker [Pent 07] [Pent 09] [Beck 11].

1.2.1. Virtual Content

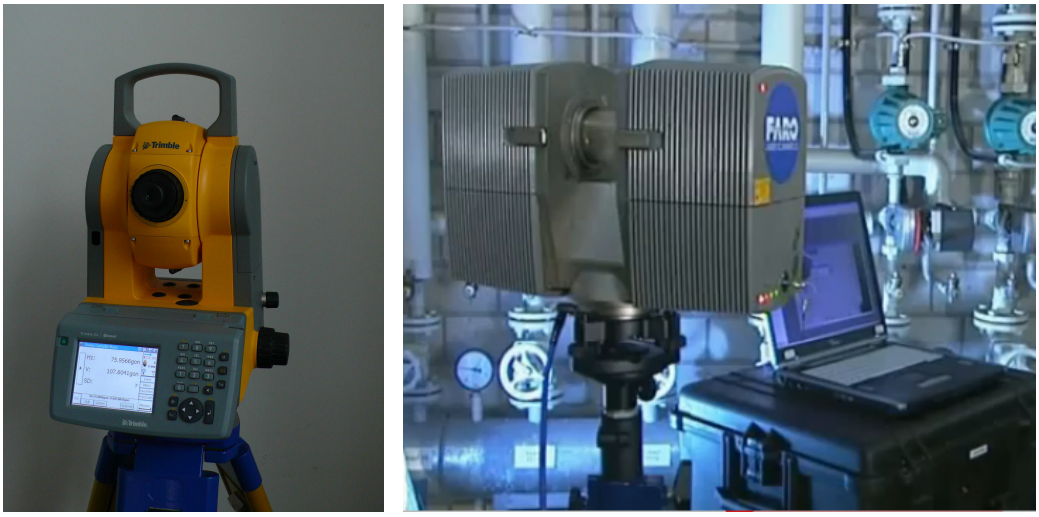
As stated above, virtual content is needed for any AR application. There are different ways to acquire such models.

In many industrial setups, a CAD model might already be available. This is particularly true for the augmentation of assembly parts which are mostly designed in digital design tools. A plethora of tools exists and is in productive use. They can be summarized by the term *digital factory* and aim at the integration of various steps in the development and production cycle, providing the necessary interfaces and data transformations [Pent 07] [Pent 08] [Pent 09]. In case a certain product is only available physically, e.g., a clay model from product design or a competitive product, current work processes mostly incorporate a (subsequent) digitalization step based on coordinate measurement machines (CMM), laser scanning or the projection of stripe patterns. An overview of such *reverse engineering* techniques is given for example in [Luhm 00a].

A more difficult question is how to obtain digital models of other objects such as buildings, machinery, installations, pillars, and so on. Even if CAD models for such objects are available, their quality and actuality is highly questionable. Maintaining physically applied changes in the underlying digital models is a difficult and time-consuming task. In particular, to obtain a high-quality model, digital planning is not sufficient but the implementation should be based on the digital model to reveal discrepancies. Metrology provides means to verify existing models and to obtain new digital models [Luhm 00a]. Different tools are used for this purpose. One can distinguish devices with *point-by-point probing* such as theodolites (see Figure 1.1(a)) and *surface probing* such as laserscanners (see Figure 1.1(b)).

Regardless the type of object to be reconstructed in terms of a digital model which can be rendered in an IAR application, the field of metrology provides suitable measurement devices and algorithms. However, metrology does not only solve the problem of reconstructing 3D models for rendering purposes. The classical field of application for metrology rather is in the spatial reconstruction of geometric relationships. Such information, though not suitable for visualization directly, also represents 3D content that might be valuable for the IAR application.

Classically, a theodolite is used to measure azimuth and elevation angles for dedicated points, from different viewing positions. To achieve good results, high-precision optics is incorporated. The resulting geodetic network is then used to compute point positions via trigonometric functions. Newer devices called *total stations* additionally include time-of-flight laser measurements to recover the point distance. This stabilizes the geodetic network and supersedes extra scale measurements to be taken. Recent devices automate



(a) Total station to measure 3D coordinates

(b) Laser scanner that produces dense 3D point clouds

Figure 1.1.: Large-area metrologic measurement devices (Courtesy of *FARO Europe GmbH & Co. KG*)

the adjustment of azimuth and elevation angles. A point probe with special optics is being continually tracked by a laser beam which originates in the base station. The latter is constantly adapting the angles automatically to follow the point probe. Of course, this only works for smooth movements of the point probe. Such a device covers ranges up to 60 m and comes close to what a typical tracking system provides. Therefore, the system could also be mentioned with good reason in subection 1.2.4 about tracking systems.

Metrology typically is a time-consuming offline process with two stages.

- In the first stage, measurements are collected using theodolites or total stations. This involves repositioning the device several times and measuring plenty of points from each position, probably requiring manual adjustments of the azimuth and elevation angles.
- Based on the collected data, the geodetic network is established and evaluated using methods of *adjustment theory* [Luhm 00a] [Niem 08] [Koch 97]. Degrees of freedom not determined by the measurements have to be treated by adding additional, functional constraints. Non-linear functional relationships are treated by solving the linearized model iteratively, good initial values are needed in this case in order not to converge to the wrong solution. The approach is statistically correct in the least-squares sense and yields also information about the accuracy of the incorporated measurements as well as the estimated parameters.

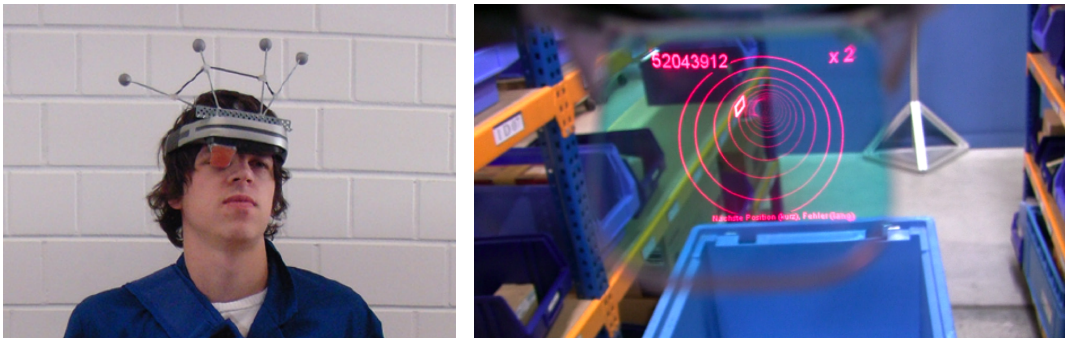
Metrologic systems with point-by-point probing play an important role in embedding an IAR application in the existing environment by registering the involved coordinate frames. This issue is tackled from different perspectives throughout this thesis.

1.2.2. Visualization Techniques

Virtual reality (VR) has been in productive use in industry for many years. A dedicated environment such as a *CAVE* [Cruz 93] or a *powerwall* are used for an immersive user experience. Typically, a stereo projector setup is needed for the powerwall or each of the at most six sides of the *CAVE*. The user's eyes (in terms of the glasses he is wearing) is tracked in the environment to adapt the stereo rendering. The images for the left and right eyes can be separated using shutter glasses that are synchronized with the projector andalternately serve both eyes. The single projector (per side) alternately provides images for the left and right eyes. Alternatvely, a dual-projector can be used, with orthogonal polarization filters in front of the lenses. The user then wears glasses having compatible polarization filters to admit only the corresponding image for each eye. This is also one of the major disadvantages of this technology. Only one user can perceive correct stereo visualizations while other persons can only observe a distorted visualization that gets worse the farther their eyes are displaced from the eyes of the primary user. Furthermore, *CAVE*s and powerwalls are suitable only to a limited number of applications, due to the fact that a dedicated environment is needed.

To mitigate this problem, the concepts of MR and AR are becoming more and more popular. These concepts promise to be useful in many situations since they are—at least in theory—compatible with existing environments and working processes. Differnt display technologies are available to overlay the user's view with virtual information. They are explained next, together with a discussion of the individual advantages and drawbacks.

In classical AR setups, *head-mounted displays* (HMD) are used [Azum 97]. They are worn by the user in front of the eyes, similar to normal glasses. Two types of HMDs can be distinguished, *video-see-trough* and *optical-see-through* displays.



(a) HMD (b) View through HMD

Figure 1.2.: Monoscopic head-mounted display (HMD)

Video-see-through displays only work in combination with at least one camera, two cameras are needed for stereo perception. A video stream of the pyhsical scene is captured and augmented with the virtual scene before it is presented to the user. In an *optical-see-through* display, the physical scene is observed directly. Such a device is shown

in Figure 1.2. The virtual scene is rendered and integrated into the user's field of vision by use of a beam combiner such as e.g., a semi-transparent mirror. The advantage of video-see-through displays is that the physical and virtual scenes are in sync since the video stream is presented to the user with the corresponding overlay. Unfortunately, this introduces a lag between the point in time when a physical activity takes place and the point in time it is observed by the user. Furthermore, the video provides only an indirect view of reality, disqualifying video-see-through for critical tasks. In case of a system failure, the user's visual perception is completely blocked. In optical-see-through displays, the augmentation lags behind physical movements, which can be annoying for the user and often leads to nausea. There are attempts to use HMDs even in underwater environments [Mora 09][Blum 09].

Mobile displays such as a tablet PC are another alternative. They are less immersive than HMDs. They are installed in the environment or mounted to the user's arm or just carried around. A mobile display can also be mounted to a tool, as shown in 1.3.

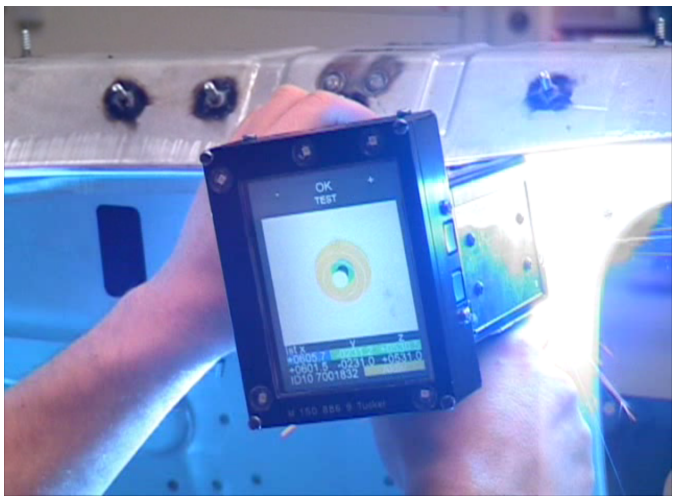


Figure 1.3.: Intelligent welding gun. The display shows guidance information indicating the next stud-welding position. Some already welded studs can be seen in the upper part of the image.

In this example of an intelligent, mobile stud-welding gun, the worker is guided to the next welding position by a purely virtual guidance shown on the display [Echt 03]. This is not a real AR application in the sense of Azuma's specification (see 1.1.2) since physical and virtual information is not combined [Azum 97]. Nevertheless, both modalities are registered in 3D and the application is fully interactive. In other contexts, the mobile display can also be combined with a camera to obtain again a video-see-through setup, as shown in 1.4 which shows a mobile phone AR viewer [Keit 09].

Mobile displays can principally be perceived by various users. With upcoming devices, even stereo-visualization is possible.

Spatial displays represent a third category of available visualization systems. A projec-

Figure 1.4.: Mobile AR viewer. A virtual object is integrated into the live video of the real scene.

tor can be used to project virtual information onto physical scene objects. Methods are available which yield correct visualizations even under the influence of complex object geometry or textured surfaces or both [Bimb 05]. However, such techniques are quite limited in the industrial context, due to complex calibration requirements and a rather low contrast. To overcome this problem, spatial laser projectors can be used, as shown in Figure 1.5 [Zaeh 06] [Schw 07].

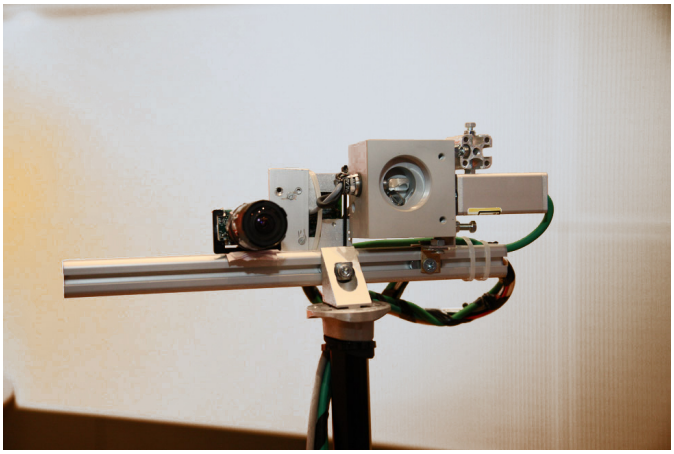
Although it is quite limited with respect to the complexity of the projected graphcis, the augmentation is clearly visible, even under disadvantageous lighting conditions.

Last but not least, there are many applications in industry that do not require at all an explicit visualization that is registered in 3D, especially in the context of metrologic applications. In 1.2.1, the complex offline process of metrologic measurements has been explained. Possibly, one might rather want to have an online metrologic system, despite the probably degraded accuracy. Although such applications can not be called AR any longer from the visualization perspective, such applications still require interactive registration of the virtual and real worlds, as the intelligent welding gun mentioned above. Due to this fact, and since the focus in this thesis is laid on tracking, such applications are explicitly included in all further considerations.

1.2.3. Interaction

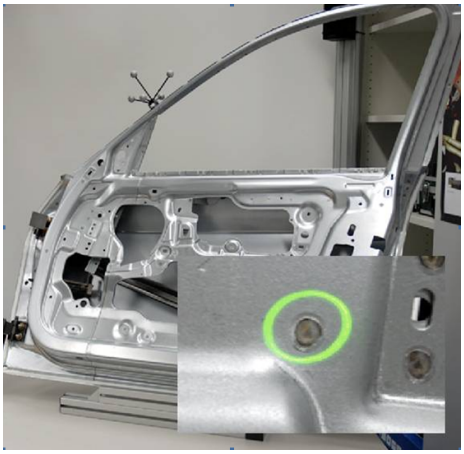
In the previous sections, the discussion was about virtual information that is registered in 3D with the physical scene and about how this information could be visualized to a user of the IAR system.

As a third category, interaction devices have to be considered. Such devices might occur in terms of physical tools that have been in use in the work process even before the introduction of the IAR system, such as a simple screw driver or the already mentioned



(a) Stationary laser projector

(b) Head-worn laser projector



(c) Laser projection in quality assurance application

Figure 1.5.: Laser projector as a spatial display

stud welding gun. Interaction devices are maybe more important than display devices. It has already been stated above (see 1.2.2) that IAR can do without any visualization that is registered in 3D. However, interaction in 3D is often indispensable in a productive application.

Using existing tools as interaction devices has the advantage of minimizing the changes that have to be applied to the existing work process when introducing the IAR system. In the stud-welding example depicted in Figure 1.3, the welding gun is not only used for aligning the physical stud with the planned stud position as given in the virtual model. Besides activating the welding process, pulling the trigger also changes the state of the IAR application by switching to the next stud position. Altogether, a good symbiosis of the existing task with the IAR improvements is obtained.

However, the tracking of such devices is offen difficult, as shown in the next subsection (1.2.4).

Besides reusing existing tools as interaction devices, also dedicated tools can be introduced, if needed. Special *pointing devices* or *measurement tools* can be used to directly measure coordinates in 3D, a requirement that newly arises through the AR system. Furthermore, they are suitable for selection of virtual objects or to "draw" virtual content in 3D. So-called *data gloves* are available for hand and maybe also finger tracking. They are mainly used for gesture recognition.

A third group of interaction devices is given in terms of mobile tablet PCs or handheld computers. They have already been described in the context of visualization techniques (see 1.2.2). The fact that those devices also offer buttons or a touch screen, often makes them a natural choice in applications where visualization and interaction is needed. A major drawback is fact that interaction is not available in 3D which highly complicates interaction with the 3D scene. Furthermore, frequently directing his attention at the mobile device might distract the worker from his actual task. If interaction should be performed with the IAR application rather than with the 3D scene, the 2D user interface provided by mobile devices may be better suited than a 3D interaction device, also because users are already accustomed to it.

1.2.4. Tracking Systems

The term *tracking* has already been used several times before in this thesis, without a proper definition yet. It means to determine the pose (position and orientation) of an object with respect to some coordinate system in real-time. A slightly different definition from the computer vision community refers to tracking as following the movements of an object once its initial position and or orientation has been determined. The process of retrieving the initial pose/position/orientation is then called *detection*. This definition is due to the fact that computer vision algorithms make heavy use of prior knowledge about the object position/orientation/pose from already processed video frames when processing the current frame. Due to this prior knowledge, tracking is a much easier task than detection, which has to be accomplished without any prior knowledge. In the context of this thesis, we stick to the more general definition of tracking given first, which also fits non-vision sensing systems such as gyroscopes, that do not distinguish between tracking and detection.

Tracking is needed to fulfill the following requirements:

1. Register the virtual 3D content that is supposed to be moved in space, with the real world.

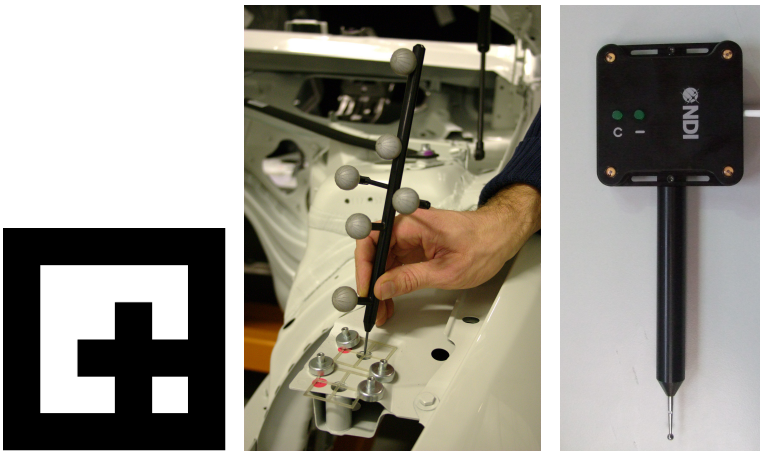
2. Register mobile displays such as HMDs or handheld devices (see 1.2.2) with the virtual world.

3. Register mobile physical objects such as tools or pointing devices with the virtual world.

Please note that this list resembles the three preceding sections. At least one of those requirements is typically needed for typical IAR applications.

Tracking represents a major challenge for IAR applications. It can be provided by a variety of sensors [Welc 02]. Classifications according to physical principles have been suggested by Meyer et al. [Meye 92] and Rolland et al. [Roll 00]. Optical cameras are often used to obtain high-quality data, not only in offline photogrammetry. Different approaches can be distinguished. They all have in common that certain 2D points in the image are used to recover 3D information by some geometric relationships [Hart 00] [Luhm 00a] [Atki 96].

Optical *marker-based* approaches use objects with a distinctive structure, so-called *fiducials*, that can be segmented easily from the rest of the scene. In industrial setups, often markers built from retro-reflective fiducials (passive) or infrared (IR) LEDs (active) are used. Such markers can be detected easily by an IR camera. This also holds for square markers printed on paper that have a distinctive shape and contrast and can be detected in a normal camera image. Such systems are available on a commercial basis. Typical markers and pointing devices are depicted in Figure 1.6.



(a) Square marker (b) Pointing device with passive retro-reflective fiducials (c) Pointing device with active IR LEDs

Figure 1.6.: Different types of markers and pointing devices

Pointing devices or probing devices often feature a spherical tip as depicted in Figure 1.6(c) instead of a pike such as shown in Figure 1.6(b), especially high-precision devices coming from the metrologic domain. Often drillings and clearances have to be measured. This can be achieved quite easily and with high accuracy by taking multiple measurements with a spherical tip, touching directly the object to be measured. For example, the center and diameter of a circular drilling can be inferred from at least three measurements of its rim, lying in a plane orthogonal to the drilling direction, taking the diameter of the spherical tip into account. A pike complicates touching the rim, in particular for drillings in sheet. It might furthermore scratch the surface.

Marker-less approaches do not rely on fiducials. They have to extract distinctive point (SIFT, SURF) or line features instead. The process of finding, recognizing and classifying features can be supported by a prior learning phase, or by using digital information of the scene, if available. Well-known examples of this category are [Klei 07] [Wagn 08] [Wagn 10] [Bles 06] [Wust 07b] [Wust 07a].

Global tracking systems such as GPS or compasses yield low-precision measurements that can, however, be valuable for the initialization of other trackers. See also the discussion about tracking and detection in 1.2.1. The use of a compass, however, does not make much sense in many industrial environments due to large deviations resulting from metal objects. Acoustic or magnetic systems are also in use and yield good results in certain environments.

Inertial sensors such as gyroscopes and accelerometers yield relative motion data. The major disadvantage of inertial sensors is that they deliver the rate of change of position or orientation speed, thus the acceleration. This is only the second derivative of the desired position/orientation in space. Integration always requires a starting condition and suffers from drift which accumulates over time. The amount of drift depends on the dimensioning of the hardware. In AR, devices typically have the size of a match box and drift invalidates the measurements after a few seconds, if no external reference is given. Ships and airplanes are equipped with more reliable devices providing accurate measurements for much longer periods of time.

Inertial sensors are often used in addition to other sensors which can provide the needed starting condition. Since they have a high update rate, they can help to stabilize tracking under fast movements and bridge the gap in case of a temporary blackout of the primary tracking system, e.g., due to occlusion problems when using optical tracking. Typical devices contain in one unit a gyroscope, accelerometers and a compass/inclinometer which yields the starting condition at least with respect to orientation.

Metrologic systems have already been treated in 1.2.1, in the context of derivation of digital 3D models and 3D reconstruction of objects. Those systems are typically operated in a time-consuming offline mode. However, there are also metrologic devices that provide pose data in real-time. Examples are mechanical measurement arms such as shown in Figure 1.7 or coordinate measurement machines (CMMs).

The tip of the arm is guided manually to the point of interest. At any point in time, its 3D position is reconstructed, based on the known geometry of the individual segments of the arm and the measured angles of its joints. This way, small volumes up to 4 m^3 can be covered with a precision of approximately 0.1 mm or lower.

So-called *laser trackers* extend this principle to larger volumes of up to 110 m diameter. A typical device[1] is shown in Figure 1.8. There is no longer a rigid connection between base system and probe. Instead, a laser beam emitted by the base station is used to measure the position of the probe. The latter is equipped with special optics that reflects the laser beam towards the base station which constantly updates the azimuth and elevation angles automaticalls to follow the movements of the probe. The achievable accuracy depends on the distance between the probe and the base station, in a distance

[1] *FARO ION* [Faro 11b]

Figure 1.7.: *FARO* mechanical measurement arm (Courtesy of *FARO Europe GmbH & Co. KG*)

Figure 1.8.: *FARO ION* laser tracker (Courtesy of *FARO Europe GmbH & Co. KG*)

of 10 m, the error is only 0.049 mm. Obviously, for a continual tracking, a direct line of sight is needed at all times. Furthermore, only careful movements are tolerated and multiple targets are not supported.

In the photogrammetry domain, *offline* and *online photogrammetric systems* are distinguished [Luhm 00b] [Luhm 00a]. This definition is extended to measurement systems in general in the context of this thesis. *Online* metrologic devices can be seen as normal tracking devices.

In optical or acoustic tracking, one can typically distinguish between *outside-in* and *inside-out* tracking [Roll 00]. The former refers to a setup where rigidly installed sensors

(e.g., cameras or microphones) observe objects with distinctive features (e.g., fiducials or microphones) moving in the tracking volume. The latter refers to a setup where one or several sensors moving around with the object to which they are attached, observe stationary features in the scene. In some cases, both, sensors and objects might be movable, this is sometimes referred to as *inside-in* tracking [Muld 94]. An example for this approach is the *NIKON iSpace* metrology and tracking system [Niko 11]. Flexible combinations of rigidly installed and mobile laser transmitters and sensors are used to cover large working volumes and to overcome line-of-sight problems. A tracked mechanical measurement arm can be seamlessly hooked into the working volume without additional referencing. A scenario based on IR tracking following a similar approach is discussed in this thesis (see 1.3.2).

1.3. Importance of Augmented Reality in Industry

The general definition of mixed and augmented reality systems is now refined in the context of industrial applications. Such applications are not limited to the production of goods but comprise also related fields for example in logistics or service. The described applications also show parallels to similar applications in other fields, such as medicine or military. Several researchers have investigated the potential of AR in industry, e.g., [Pent 09] [Beck 11]. Navab describes potential "killer apps for industrial augmented reality" in the fields of design, commissioning, manufacturing, quality control, training, monitoring and control, and service and maintenance [Nava 04]. A review of this work is presented in the remainder of this section, with special focus on the tracking requirements of the proposed applications.

1.3.1. Basic Benefit of Augmented Reality Techniques

Having a virtual model that is registered with the physical scene, provides a potential for industrial applications in two ways. An abstract overview of these two key concepts is presented next. They have been described similarly in [Pent 09]. Several implementations of these concepts are presented in 1.3.2.

Augmented Scene

By presenting the virtual model from the view of the user, things can be displayed that are not (yet) physically visible or existent. For example, the planned state of a facility or the final assembly state of a product can be visualized right from the beginning of the assembly process. This offers the opportunity to argue about arbitrary issues by a visual inspection of the augmented scene as well as a comparison of both, the real and virtual scene. Assembly instructions can be shown in the very location where they are needed. The attention of the user can be easily attracted to certain locations.

Online Metrology

In an IAR setup, it also is possible to measure positions and orientations in 3D, e.g., using a pointing device as depicted in Figure 1.6. I.e. one can recover distances and angles on a physical object almost in real-time, unlike with other offline metrologic systems that often require dedicated environments (e.g., CMM) or additional complex setup procedures (e.g., laser tracker). I.e., once an IAR tracking infrastructure is available, classical metrologic applications can be better intertwined with the existing work processes, especially if their accuracy requirements are moderate. If a registered virtual model or even some sort of visualization is available, the benefits of such a system are further increased. The metrologic aspects can be directly integrated with the visual aspects of the augmented scene. Luhmann distinguishes offline from online measurement systems in the realm of photogrammetry [Luhm 00b] [Luhm 00a]. This can be extended to the general context of online tracking systems.

1.3.2. Applications

An overview of typical fields of application is given in the following, describing both the problem to be solved and the IAR system proposed for this purpose. This overview is task-oriented. Some of the mentioned applications will be revisited several times throughout this thesis. See also [Rege 07] [Pent 09] for overviews of IAR applications. Furthermore, many IAR applications have been investigated in the context of the German research projects *ARVIKA* [ARVI 11] [Frie 04] and AVILUS [AVIL 11].

In an industrial environment, AR applications have the highest potential when there is a large amount of complex manual work to do. Nevertheless it is still not commonly used due to high investment needed as well as insufficient maturity and robustness of AR/MR technology.

Factory Planning and Change Management

Planning factories or facilities is a difficult task. A plant is only rarely constructed from scratch. More likely, legacy installations are already available that have to be considered, and different alternatives for modernization have to be pondered. A visual comparison of existent physical objects with virtual objects potentially to be installed can reveal answers to certain questions more easily than a comparison of old, maybe outdated paper plans with the new plans. If a digital model of the existing facility is available, it can be directly compared to the physical state of the facility. This can be useful to detect changes implemented physically and to update the existing digital model accordingly. Furthermore, deviations between both modalities can be detected qualitatively on a visual basis and measured quantitatively using online metrologic techniques.

Georgel proposes a marker-less approach to automatically detect in the image the anchor-plates that are available in many industrial environments [Geor 07]. These are points that are typically known with high accuracy from offline metrologic measurements. An example is depicted in Figure 1.9.

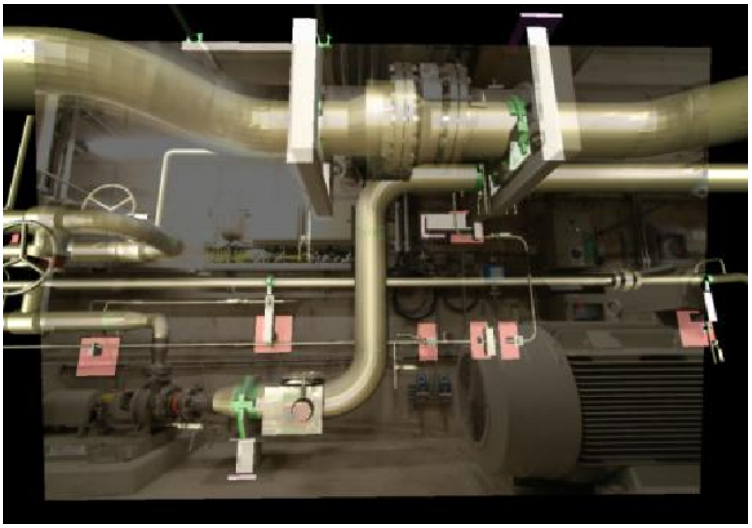


Figure 1.9.: Planning and change management. For referencing the indicated anchor plates are used. Their position is known with high accuracy. (Courtesy of Pierre Fite-Georgel [Geor 07])

Interfering edge analysis is an important topic in this area. Solutions are already available on a commercial basis[2]. Though the system is rather an offline metrologic application, it uses AR overlay techniques to detect interfering edges when putting a new car model on an existing assembly line. The analysis is based on high quality photographs containing optical square markers attached to various objects in the scene, as depicted in Figure 1.10(a).

(a) Setup (b) Clipping plane representing the limit

Figure 1.10.: Offline visual overlay for interfering edge analysis (Courtesy of *Adam Opel GmbH* [Pent 09])

After an offline processing step involving a bundle adjustment of all markers and

[2] e.g., *metaio* [meta 11]

all images from all positions, the virtual car model can be moved interactively on the assembly line as shown on the images in order to detect problems. Virtual measurement tools such as movable planes etc. are provided to support the interactive analysis, as shown in Figure 1.10(b). The relationship between the virtual measurement tool and the physical object in question is established via a marker that is attached to this object before taking the images. This explains why many markers have to be attached to the physical scene in advance. Their placement should be well-considered in order to avoid time-consuming iterations of the whole process when objects later turn out to be a potential collision candidate. More information on this application and tools for factory planning in general can be found in [Pent 07] [Pent 09].

Discrepancy Checks

Especially in the naval and avionic industry, products are very complex and highly customized. The high rate of manual work is an opportunity to create new tools using AR technology that have the potential to decrease production time and cost. Discrepancy checks aim at a direct comparison of the *actual* state of an object against its *set* state given in terms of the digital model. In a *qualitative discrepancy check* the real object is overlayed visually with the virtual model, or the virtual model is rendered from the perspective of the display. Figure 1.11 shows a qualitative discrepancy check for an airplane cabin [Keit 10b].

Figure 1.11.: Qualitative discrepancy check. The virtual model of an airplane cabin is rendered from the perspective of the display to allow for a straightforward review of the current state and correctness of the construction process. (Courtesy of *EADS Innovation Works* [Keit 10b])

A similar example is the visualization of underground urban infrastructure such as wires and pipes below the surface on a mobile tablet PC [Scha 09]. A sensor fusion approach is used to provide robust tracking. GPS and compass provide a rough estimate.

Based on this, optical markerless tracking and a gyroscope allow for good registration accuracy. In this case, the comparison aspect is less important than the mere visualization of the otherwise hidden information. Therefore, again a purely virtual visualization is used.

Besides that, also *quantitative discrepancy checks* are of major interest. With a pointing device such as shown in Figure 1.6, points, clearances, or drillings in the real model can be measured and immediately compared with the set values from the pre-registered virtual data. Already experts use metrology systems to provide measurements within the production environment. For these precise offline measurements within an environment as large as an aircraft, photogrammetric or laser based metrology systems are used, such as introduced in 1.2.1. Such systems are less easily integrated in the standard production processes and often require a fixed registration sensitive to vibrations and therefore interrupt the production process.

No AR visualization is actually needed for interactive quantitative discrepancy checks, though the virtual model must still be accessible in real-time by the pointing device. Nevertheless, an integration with qualitative discrepancy checks featuring also a VR/AR based visualization is often desired, e.g., in [Noll 06] [Pent 09] [Keit 10b]. To cover the large area of the aircraft where the current assembly procedures are performed, it is not feasible to deploy countless tracking systems. Therefore, a mobile tracking system is used that is able to reference itself within the aircraft and to perform the real-time tracking of the visualization and probing devices. This is done by adding reference markers to the aircraft whose static transform within the aircraft's coordinate frame is determined in a calibration routine using an offline metrological system. Such a setup is investigated thoroughly in this thesis.

In mass production, it is often required to apply statistical tests in order to guarantee a certain level of quality. Many factors still can only be checked by humans. To optimize the performance of the statistical approach, it might be necessary to randomize the tests to be performed such that no all tests have to be performed on all products. IAR can be used to visualize the tests to be performed chosen that have been chosen by the system according to some algorithm. The laser projectors depicted in Figures 1.5(a) and 1.5(b) have been used to project the chosen test locations directly onto the physical object, as shown in 1.5(c).

Typically, high-precision IR tracking is used for such purposes. Smaller objects can be aligned and fixed with a dedicated plate that is equipped with markers [Noll 06]. For larger objects, markers have to be deployed to the objects themselves [Pent 09][Keit 10b].

Assembly

IAR can help in the assembly of products by showing step-by-step instructions, highlighting objects of interest, or by supplying guidance in the usage of tools.

One of the first IAR applications was evaluated in the aviation industry [Caud 92]. It aimed at the assembly of cable harnesses and connectors in a dedicated AR setup focusing on a huge board to lay out the cables. The board was covered by a pattern of fiducial markers to allow for optical inside-out tracking. Step-by-step instructions were

shown to the workers in an HMD.

Another application, the intelligent stud-welding gun, has already been mentioned above, see 1.2.2 [Echt 03]. In this IAR application, the worker is guided to the next welding position by a purely virtual guidance shown on the display, as depicted in Figure 1.3. The virtual model is given in terms of a list of planned stud positions given in the CAD model of the car. It is registered with the physical car body by some reference points close to the axle mounting. Due to reasons related to the production process, those points are known to be quite consistent with the CAD model. Furthermore, they are distributed almost across the whole working volume which promises good registration results. To ease the welding process, the car body is not installed rigidly in the scene but tracked by infrared cameras mounted to the ceiling of the room. Retroreflective marker balls are attached to the car body for this purpose. The welding gun is also tracked by those cameras, using active LEDs installed on its surface. A natural problem of this setup are occlussion that occurs at locations inside the car or under the hood where the gun cannot be seen by the cameras as depicted in Figure 1.12.

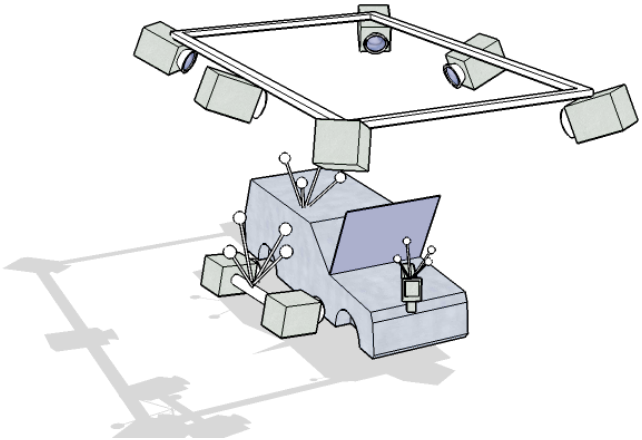


Figure 1.12.: Occlusion problem

An indirect tracking setup could be used to mitigate this problem, with a static camera setup mounted to the ceiling as above and a mobile camera setup mounted to a tripod. The mobile setup is equipped with a target such that its pose can be tracked by the static system. The mobile system in turn can "look into" the otherwise occluded areas. This topic about the intelligent welding gun and indirect tracking in particular will be revisited again throughout this thesis.

The relationship between the mobile setup and the tracked object could be described as *inside-in tracking* [Noll 06] since both entities are moved more or less dynamically in the scene. This term has been used first by Mulder [Muld 94], synonymously with the term *relative tracking*. It is not clear, however, whether the relationship between the mobile setup and common reference points as described above should be denoted as inside-in or rather inside-out tracking. This certainly depends on how mobile the

reference target actually is, whether it is considered part of the scene or rather of the background. We stick to the term inside-out tracking for this kind of relationship.

Several researchers have worked on the combination of optical inside-out and outside-in tracking systems. Bichlmeier et al. [Bich 06] used a reference target to combine outside-in instrument tracking with inside-out estimation of HMD pose. In this setup, a monocular tracking system, rigidly mounted above the HMD, tracks the reference target. The latter is simultaneously tracked by the outside-in tracking system, such that the pose of the HMD can be calculated with respect to the outside-in system. Fischer et al. [Fisc 04] presented a hand-eye calibration algorithm which utilizes the combination of a monocular inside-out and an n-ocular outside-in optical tracking system. Both works, however, did not address the error resulting from their combined setup.

Hoff et al. [Hoff 00] have also combined inside-out and outside-in optical tracking to improve the accuracy of tracking an HMD. They used two rigidly combined reference targets which are simultaneously tracked by an inside-out and outside-in optical tracking system. Based on accuracy specifications delivered by the manufacturer, they estimated the 6 DoF tracking accuracy for each involved target and for each tracking system. The estimated error covariances are used for sensor fusion of the inside-out and outside-in tracking system using Kalman filtering [Kalm 60]. This approach decreases the pose uncertainty by using two independent tracking informations.

Maintenance

As in assembly, IAR can also help in maintenance by showing step-by-step instructions for dismantling and reassembling a technical object, by highlighting objects of interest, or by supplying guidance in usage of tools. For example, a mobile laser projector (see also 1.5(a)) can be used to visualize such information for on-sight maintenance of military equipment [Schw 07] as shown in Figure 1.13.

Figure 1.13.: Maintenance instructions projected onto the missile launcher

In this example, the weapon is tracked by a markerless optical approach. Each maintenance step has to be trained in advance such that the object can be tracked throughout its stages of disassembling. The laser projector is situated in the same housing as the camera. This facilitates the calibration of the offset between those two devices that is needed for a correct augmentation.

Picking

Picking is a logistics application where people have to collect items from a warehouse according to a computer-generated list. This process is often not automatable, due to the complexity and variety of the objects which requires both, human intelligence and senses. HMDs or again a laser projector can be used as augmentation system to guide the user to the next shelf and to show him the number of items to collect.

An HMD-based solution has been investigated by Schwerdtfeger [Schw 09] [Schw 10]. He analysed the human factors of utility and usability arising with such an application. As an evaluation setup, a small warehouse was equipped with IR tracking cameras. A monocular, monochrome HMD[3] with a retinal laser projector was chosen for visualization, due to its high contrast.

Tracking is a difficult issue in this application, due to the large distances that have to be covered. IR tracking alone cannot be used for a full coverage of a typical warehouse due to its rather high price.

Workflow Analysis

Workflow analysis is not an IAR application in its own right. This task aims at recording and analyzing typical movements of users or tools in a certain work process. Using a tracking system, motion data is recorded such that it can be further analyzed. The final goal is an optimization of the work process. Typical movements of persons and objects can be extracted. This can e.g., give hints to reduce walking distances or to optimize the location of assembly parts. Furthermore, the data helps also in the installation of a permanent tracking system in the course of the integration of an IAR application. To give only one example, when installing cameras in the scene, the setup should be optimized with respect to the occlusions that might disrupt the prospective work process. An overview of workflow analysis in the medical domain is given in [Nava 07].

1.3.3. Relations to Classical Mixed- and Augmented Reality

Recapitulating the applications outlined in the previous subsection, IAR makes sense only in those work processes that are dominated by manual work. A process that can be completely automated by machines or robots does not benefit from IAR. Furthermore, robots are typically more accurate than online tracking systems since they can rely on mechanic tracking. IAR rather aims at a better integration of humans with the work

[3] *Microvision Nomad* [Micr 11]

process to make the most of their unrivaled capabilites in terms of senses and the ability to make the right decisions.

In the reality-virtuality continuum (see 1.1.2), the described applications mostly tend to the extreme left. Only selected virtual information is incorporated into the real world. The augmented airplane cabin features the highest degree of virtuality. Still, it is an AR application, not augmented virtuality. It can therefore be concluded that IAR rather brings the virtual realm into reality than vice versa.

Furthermore, it is noticeable that HMDs do not play a major role yet in IAR. Spatial displays and mobile displays are at least as important. If an HMD is used, it is probably monochrome and monoscopic. More immersive stereo displays are still too heavy, have a too low contrast. See [Schw 10] for a discussion of current HMDs.

Unlike in classical AR, an IAR application does not necessarily have a visualization of virtual information in 3D, though it is still an interactive real-time application. Even without the 3D visualization, virtual 3D content is still registered in 3D to make it available to the user via mobile displays such as with the intelligent welding gun, the visualization system for subsurface pipes [Keit 08] [Beck 11] [Keit 10b].

1.3.4. Relations to Virtual Reality

Unlike virtual reality (VR), that is traditionally being used in industry by engineers and designers in dedicated VR environments such as powerwalls or *CAVE*s, IAR intends to integrate well with some real work process to be performed on real objects, by the people that typically operate this process.

By integrating tasks that are performed traditionally on purely virtual data, such as industrial design, geometric layout or metrologic evaluations, with tasks carried out in the realm of real objects such as prototypes, mockups, facilities, tools or repetition parts, IAR enables a broad range of new, integrated applications that were not possible with a VR system. Such IAR applications can help in productive tasks such as factory planning, change management and discrepancy checks. They can provide support in assembly, maintenance, quality control and picking. Furthermore, they enable innovative solutions in desing and marketing, through the combination of a generic mockup with a plethora of virtual themes and desings.

1.3.5. Relations to Metrology

Many of the applications presented in 1.3.2 feature at least partially a metrologic purpose. Some of the mentioned tasks are traditionally solved by offline metrologic techniques. This brings up the question how offline metrology, online metrology, and IAR fit together. There is a tradeoff between the high accuracy of offline measurements made by laser-trackers or photogrammetric systems, high-precison online measurement systems such as mechanical measurement arms and CMMs, and online tracking systems suitable for IAR. As a consequence, some of the traditionaloffline metrologic applications might be migrated to IAR applications in the future. However, online IAR cannot fully replace offline metrology because the latter still provides a much higher accuracy [Luhm 00a].

However, AR techniques have already entered the domain of offline metrology, e.g., in terms of *metaio*'s *Roivis* factory planning toolchain (see 1.3.2). Classical AR techniques such as optical square markers and video overlay AR are combined with high-precision still images to result in an offline photogrammetric AR system. The decison for an offline or online approach finally depends on the needed accuracy.

Online metrology can easily be integrated into IAR. From the AR perspective, such devices can be considered as normal tracking systems, yet featuring a high accuracy. This has the potential to extend the range of possible applications as well as the integration level of existing work processes. For example, a stud-welding application similar to that described in 1.3.2 has recently been realized at Volkswagen using a mechanical measurement arm instead of optical IR tracking [Buck 10].

Besides directly using metrological equipment for tracking, there is also the possibility to supplement the classical offline metrologic work process by online IAR-supported steps. Thereby, steps that are critical for the overall accuracy can still be performed offline. For example, establishing a high-precision wide-area reference geometry can only be solved using thedolites or total stations and complex post processing of the measured data. Having such a reference geometry, an online system can be hooked up more easily and provide the needed accuracy at the same time. The discrepancy check application in the airplane cabin described in 1.3.2 belongs to this category. It is investigated in more detail throughout this thesis. Therby, productivity can be increased by porting time-consuming, repetitive measurement jobs to an IAR system. For example, measuring plenty of points on a complex physical geometry for variance analysis could be supported by an AR visualization for guidance. To achieve this goal of flexible, hybrid solutions, offline and online tasks have to be properly planned an integrated with each other. *Spatial Analyzer*[4] is an example of an offline metrologic framework and toolkit. But it also supports real-time tracking, e.g., for observing the assembly process and finally verifying the result of the assembly.

Generally, it can be expected that both fields can benefit from each other, which implies that offline metrologic and online tracking systems have to be integrated with one another.

1.4. The Tracking Challenge

Tracking is needed for VR, MR, and especially (I)AR. In VR environments, it is relatively easy to implement, due to the well-defined environment of a powerwall or *CAVE* where almost no constraints in terms of an already existing environment and work process have to be considered and where accuracy requirements play a minor role. Therefore, despite their limited use, such environments have been used for some while in industry.

IAR on the contrary promises a much broader range of applications, mainly due to its natural integration into existing environments and the corresponding workflows. However, exactly this fact highly complicates the tracking problem. Large areas have to be covered in some applications, e.g., in the airplane cabin scenario described in 1.3.2.

[4] *New River Kinematics* [Kine 11]

Stationary facilites and tracking systems have to be combined with mobile objects, tools and sensors. Disruptive conditions that are present in the environment have to be compensated for. Dirt, dust, and bad illumination conditions complicates the use of normal or IR cameras. Occlusion from physical objects constrains the usable tracking volume by impeding a direct line of sight between cameras and markers or features. Noise and vibrations have a negative influence on the proper calibration of all kinds of tracking systems. Metal and electro-magnetic interference almost inhibits the use of compasses.

Last but not least, changes to the existing environment caused by the installation of a tracking system have to be in accordance with the existing work processes. This constrains the modifications that can be applied to the environment and also the tools in use. Their usability with respect to the work processes must not be hindered by the introduction of IAR in every respect. This often requires the ability to quickly setup the system when needed and dismantle it afterwards to avoid disturbances with other work processes. This also brings up the need for robust methods to hook up tracking systems quickly, when needed.

Despite all these facts, the tracking has to be robust and accurate. This requirement can often only be accommodated by multi-sensor setups and *sensor fusion*. A tracking environment involving several sensors is also called a *heterogeneous tracking environment*. This particularly includes also the integration of metrologic devices into the process. Especially in wide-area IAR applications, offline metrologic devices can be used to establish a basic setup by providing an accurate global coordinate frame. This has to be properly interfaced with the real-time tracking systems (including the online metrologig devices) in terms of dedicated "link" coordinate frames.

The goal of this thesis is to investigate these tracking issues with respect to principal feasibility, robustness and accuracy.

2. Tracking to Meet Industrial Criteria

Several efforts exist in the literature that state requirements in general for IAR. A summary is presented first in 2.1. The requirements are then refined with a special focus on tracking and the phases to be passed through when making IAR productive 2.2. Then, concrete requirements are formulated toward a common platform for the management of IAR tracking systems in 2.3. Although we focus on industrial applications here, these requirements mostly apply also for other domains, e.g., for medical or military applications.

2.1. Requirements in the Literature

Navab lists three critical requirements in his general survey of IAR killer applications for a solution to be successful: reliability, user-friendlyness, and scaleability beyond simple prototypes [Nava 04]. Regenbrecht lists reasons for the failure of most IAR applications that have been implemented so far: lack of robustness, reliability, quality, and practical experience [Rege 07]. In both characterizations of the problem, tracking plays a major role for the fulfilment of the stated requirements.

Pentenrieder reviewed these criteria to derive a catalogue of requirements in the context of an offline IAR factory planning tool [Pent 09]. A special focus is thereby laid on usability of the system and also on the accuracy of spatial measurements. In particular, besides the requirements specific for factory planning, she lists the following basic requirements that are relevant for IAR applications in general:

- The tracking hardware must consist of simple standard components and provide a high performance, especially in terms of accuracy. The software should run on a standard PC. The tracking system should be usable by non-experts, easy to setup, robust, and accurate.

- An intuitive graphical interface is needed for the following tasks:
  - Scene management: Functionality is needed to handle all kinds of data that occurs in the course of an IAR application. In particular, it should handle real world image data, virtual 3D models, project configuration data, registration information describing the relationship between the real and virtual worlds, sensor calibration data and basic facilities for 3D object manipulation.
  - Accuracy: The whole tracking system consisting of hardware and software has to be precise enough to satisfy the application requirements. Errors in all influencing systems must be reduced or prohibited. In this respect,

the registration of virtuality and reality is very important for overall system performance. Furthermore, quality statements must be made available and presented to the user in an intuitive and nonetheless correct way.

– Process support: Usability is important not only for planning but also for the preparation and collection of configuration, calibration, registration, and model data. A special focus is laid on the registration of real and virtual world, since it has a considerable influence on the overall accuracy.

The cited requirements concern the most important aspects of IAR in general. It becomes clear'that tracking plays a major role in the future success of IAR applications.

2.2. IAR Design Phases

It is crucial to provide means to setup and implement a robust and maintainable tracking infrastructure to allow AR applications to spread in the industrial domain. Tools have to be provided to support this process throughout all its phases. Particular demands of these phases with respect to tracking have been elaborated in accordance with Becker [Beck 11] in the context of the augmented airplane cabin scenario discussed in 1.3.2. They are reviewed next:

- Definition: In the definiton phase of a proposed IAR application, facilities must be provided to outline the involved spatial relationships between all involved coordinate frames in a semantically correct way. In detail, this incorporates the specification of relationships between virtual models, world and object coordinate frames, tracking systems and offline metrologic equipment. In this phase, also the accuracy requirements for the application are defined. In this phase, the principal feasibility of the proposed system has to be verified. Regarding the tracking, it has to be ckecked whether the coverage of the proposed sensors and the estimated accuracy is in line with the requirements. This is mainly accomplished by simulation techniques such that the principal feasibility of the proposed system can be proven before expensive purchase decisions are made.

- Deployment: In this phase, the tracking system is deployed in the target environment. This involves initial registration of the rigidly installed sensors with the environement and also other static transformations. After that, the proposed system is validated against the IAR application requirements stated during the definition phase. A process is necessary to perform this validation, based on empirical measurements taken by the real-time tracking systems and additional offline metrologic equipment, if necessary. Ideally, these measurements confirm the proposed values from the definition phase.

- Operation and Maintenance: After approval in the validation phase, the system goes into productive use and is now operated by non-expert users. The process has to be continually monitored to guarantee the validated system properties.

Setup and dismantling of the system must be possible by on-site personnel, taking into account the assured properties of the system. This means in particular that necessary calibration and registration procedures can be carried out in a robust and intuitive way. The need for maintenance by experts has to be restricted to rare occasions.

Means have to be provided for proper documentation throughout all these phases.

2.3. Requirements for Management of Tracking

Based on the experiences stated in 1.2.4 and some additional considerations, the requirements toward a platform for the management of tracking in IAR are now formulated. The goal is to provide a generic software approach for tracking that is compliant with the design phases of an industrial application, as explained in 2.2. This does not contradict a later integration of the described concepts into integrated design tools that manage also other aspects of an IAR application (content, visualization, interaction, and tracking) as described in 1.2.

2.3.1. Guaranteed Performance

First of all, a software framework to solve the tracking problem must fulfil the requirements with respect to the quality of servcie as also stated in 1.2.4, namely reliability, robustness and accuracy. For industrial purposes, it is not sufficient to be able to setup such a system under idealized conditions. Rather, it should be possible to guarantee these properties throughout real, daily operation.

It must be possible to quantify the overall performance of the system in terms of accuracy. Depending on the intended application, this can be in terms of 2D overlay error, 3D position error, or 6DoF pose error. End-to-end error propagation in real-time is required to provide this information to the application for each measurement. Violations of the guaranteed quality level must be reliably detected by the system such that users can be informed immediately about potential error conditions. Furthermore, simulation facilites are needed to estimate the accuracy of a proposed setup based on simple assumptions, without relying on real measurement data.

To obtain useful results from error propagation, it is necessary to quantify the results of all input data first. This comprises real-time measurements as well as static transformations that are estimated in advance. Tools have to be provided to assess the accuracy of both kinds in the concrete IAR setup. Vendor specifications are not always available and often cannot be given without knowing the concrete setup.

2.3.2. Sensor Fusion

For a robust system, facilities are needed to cope with a heterogeneous sensor infrastructure, as already stated in 1.4. Sensor fusion might help to mitigate sensor errors, provide the necessary degrees of freedom (DoF) or expand the tracking area. Durrant-Whyte

tries to formally distinguish three types of fusion, according to the usage of available information [Durr 88].

Competitive fusion The sensors provide information in the same location, having the same degrees of freedom. Thus, there is redundant information which can be used to improve the quality of the signal, e.g., by using a Kalman filter to fuse two 6DoF poses [Kalm 60]. Alternatively, particle filters can be used. Unlike the Kalman filter, they don't rely on a linearized description of the problem using a Taylor expansion that is truncated after the linear part and may therefore provide more accurate results. On the other hand, particle filters are more computationally expensive [Douc 01]. This type of fusion has also been called *concurrent fusion* [Broo 97].

Complementary fusion Complementary sensors provide information in the same location but for different degrees of freedom. A typical example is the combination of 3DoF translations from an accelerometer and 3DoF orientations from a gyroscope to obain 6DoFposes[1]. This could also be called *functionally complementary fusion*. Another variation is *spatially or temporarily complementary fusion* where different sensors are used in different areas or at different points in time.

Cooperative fusion One sensor relies on the data of another sensor. Durrant-Whyte gives as an example the "use of one sensor's information to guide the search for new observations". Using a compass to provide global directions to an inertial tracker would be another example. The use of offline metrologic equipment for global registration of online tracking devices can also be considered a member of this category. The indirect tracking setup introduced in 1.3.2 also belongs to this category. The mobile tracking system can be used only if its pose in the global reference frame is known. Another example is to use GPS to load the proper feature map for markerless tracking on a mobile phone [Reit 07]. Effectively, in many fusion schemes a measurement "depends" on prior measurements, e.g., in gating for acoustic signals or region based image processing.

Often, a mixture of these categories can be observed in typical scenarios. For example, with *SCAAT* (single constraint at a time) [Welc 96] [Welc 97], Welch describes a generic mathematical method based on a Kalman filter to determine the pose from various unsynchronized measurements that each, considered by themselves, would under-constrain the mathematical solution. This could be interpreted as complementary fusion. All measurements considered as a whole, however, the solution is over-determined, an indication for competitive fusion. The approach increases the update rate by incrementally computing a new estimate whenever a measurement becomes available, and it increases the accuracy by immediately incorporating each new measurement through statistical filtering. Furthermore, sensor devices are auto-calibrated on-the-fly, a characteristic of cooperative fusion. From this point of view, where all measurements are integrated in a

[1] This is typically handled internally, e.g., by the *XSens MTi-G* [XSen 11]

single functional and statistical model, the three fusion types mentioned above cannot be clearly distinguished. Rather, they can be considered ideal archetypes that are more or less prevalent in a concrete fusion setup. It will be seen however (cf. 5.3.2), that from the point of view of a tracking framework where reasoning is based on high-level measurements such as positional and rotational information, the made distinction becomes much clearer.

2.3.3. Modularization

The framework must provide a layer of abstraction for IAR applications to effectively separate them from the tracking infrastructure. This includes a flexible and maybe also parallel use of existing tracking facilities by different applications. A common standard is needed for the specification of relationships between sensors and objects and the exchange of tracking data associated with them. A similar approach is pursued for example in the field of ubiquitous computing and location-based services by the *context toolkit* [Dey 01] or the *Nexus* project [Hohl 99]. They integrate various sensors that provide rough spatial or other contextual information, though not in a sense that would allow for real-time augmentations.

The abstraction layer facilitates sharing and reuse of existing resources in terms of hardware and software, by providing modularization concepts. This is necessary to enable the development of best-practice solutions that can be easily adapted to new IAR applications.

A further benefit of the modularization is an increased flexibility. Ad-hoc installation and removal of hardware components, including the necessary calibration and registration procedures, is a key requirement, as already stated in 1.4.

The performane claims stated in 2.3.1 must not be compromised by the modularization and abstraction. Ideally, the system shall be as reliable, robust, and accurate than a hard-wired solution.

2.3.4. Maintainability

Maintainability is of course improved by the modularization claimed in 2.3.3. However, further aspects are relevant to accomodate all requirements arising throughout the various development phases of an IAR application (see 2.2).

The definition and deployment stages can assumed to be taken care of by experts. An "expert" in this context is supposed to be a person that has a general understanding of IAR and especially also of sensor hardware, calibration and registration methods. Such a person can be called an "IAR-engineer". This person shall however not necessarily be a good programmer. Expert tools must be provided for the specification of all spatial relationships, the calibration and registration of the sensors and static spatial relationships, to verify the system based on simulation, to validate it based on real measurements, and also to continually monitor it quickly and reliably. At any point in time, the IAR engineer needs a clearly documented understanding of the current configuration. It shall be

impossible for IAR-engineers to mistakingly give specifications of physically impossible tracking setups.

Once the system has been made ready for productive use, the responsibilites of the IAR-engineer are restricted to maintenance, similar to a system administrator in the IT department. Ideally, the system is then operated by the non-expert users of the IAR application alone. Maintenance tasks shall be institutionalized in terms of regular service intervals to allow for the schedulability of productive operation. The need for unscheduled maintenance has to be reduced as much as possible. This comprises also the regular setup and dismantling of mobile components, as well as their calibration and registration, that has to be repeated during productive use time and again. For this purpose, robust routines are needed, that are simple and intuitive enough to be used by a normal user. Instead of the flexibility of the expert tools, the end-user tools shall provide only the methods that have been selected for this purpose by the IAR-engineer during system validation. It must be possible for the end-user to monitor the system performance at all times, during calibration/registration as well as during runtime. Simple figures shall give information about whether the system is still running within the specifications made by the IAR-engineer.

3. Related Work

Many systems have been described which aim at the integration of multiple sensors and also at providing a layer of abstraction for the applications relying on them. Examples are the *context toolkit* [Dey 01] and the *Nexus* framework [Hohl 99] in the field of ubiquitous computing, that have already been mentioned in 2.3.3. For real-time tracking, there are several approaches, too. However, they are rather problem-specific and do not follow a general purpose approach, such as, for example, the perception and control techniques used in autonomous land vehicles [Dick 07] [Thru 05]. The architectures proposed for this field of application heavily rely on computer vision and make strong assumptions on the topology of the spatial relationships to be monitored [Darm 08] [Albu 02]. They were not designed with flexibility and maintainability in mind.

More generic architectures are based on so-called data flow networks, directed graphs whose nodes represent components for data acquisition from sensors or for data transformation [Morr 94]. Components can be flexibly cross-linked, based on a small set of data types allowed on component inputs and outputs, such as 3DoF position, 3DoF orientation, or 6DoF pose, a combination of both. Such data flow networks constitute one of the most important building blocks of ubiquitous tracking environments and provide the necessary transformation, synchronization, and network transport of tracking data.

In the domain of multimodal user interfaces, several component-based frameworks have been described that follow such a modular approach, though the processed data types differ, of course. They also feature graphical rapid prototyping environments to setup the data flow network [Serr 08] [Bouc 04] [Drag 01]. They allow for the integration of different input devices and provide algorithms for the flexible preparation and fusion of raw input events. A rapid prototyping environment allows the corresponding data flow to be configured graphically, according to the application's needs.

In rendering, scene graphs with the topology of a tree are in use for many years now as an abstract description of the scene. [Wern 93]. The structure has obvious advantages for understanding the topology of complex scenes in particular when spatial relationships change over time (animation). It provides the basis for simple graphical modeling in tools such as *CATIA* [CATI 11]. The spatial-relationship graph (SRG) concept to be introduced in 5 features some parallels but also differences to the scene-graph concept [Echt 08].

Matlab Simulink also follows an abstraction approach [Simu 11]. Basic bulding blocks for time-varying systems are provided in the domains of communications, controls, signal processing, video processing, and image processing. From these, arbitrary data flow networks can be modeled, simulated, and debugged. Existing building blocks can be combined recursively to form new building blocks. This eases reusability and results in a hierarchical data flow layout that can be edited at the level of abstraction needed in

each case.

For tracking, the two most prominent examples in the literature are *OpenTracker* [Reit 01] and *VRPN* [Tayl 01]. Whereas *VRPN*, coming from the VR domain, focuses on how various sensors can be linked together and how tracking data can be synchronized and transfered via network, *OpenTracker* originated in AR and also includes rudimentary support for sensor fusion.

Also noteworthy is the *OSGAR* system [Coel 04]. It builds upon tracking data amongst others given by *VRPN* and models tracked, statically registered, or deduced spatial relationships in a scene graph [Stra 92]. Assumed tracking and registration errors can be propagated along its branches in terms of covariances, thereby providing the application with a measure of the expected tracking accuracy.

Noteworthy is also the *DWARF* system to build and deploy AR applications [Baue 01] [Wagn 05]. In a peer-to-peer architecture, tracking sensors and AR applications are dynamically connected at runtime, using a protocol based on the description of *abilities* and *needs* for each participating peer. No predefined data flow specification is needed, though a particular data flow can be enforced by the proper specification of the involved abilities and needs.

Modeling data flow networks using those systems is a great relief, compared to coding all drivers and algorithms from scratch. However, it is still a tedious task and the maintainability requirements are rather delimited by the fact that invalid data flows are not excluded conceptually. Furthermore, none of the described systems provides support for registration of statically aligned entities, an obligation for modularization and real separation of tracking from the applications.

The probably most comprehensive system in the context of IAR has been described by Pentenrieder [Pent 09]. It follows an integrated approach comprising not only tracking but also content, visualization and interaction (see also 1.2). Concerning the tracking subsystem, it provides various means to register the real and virtual worlds with each other. End-to-end error propagation is also included to state the 2D overlay error of the augmentations. However, this system is rather an offline metrologic system. It supports only still images from high-precision cameras. This is also reflected in the restricted universality regarding the specification of spatial relationships. The system detects the pose of markers in the real image and inserts the virtual model, based on a static offset before projecting it onto the 2D image. Pentenrieder calls this a “flat SRG” concept, because only those spatial relationships are mapped that are needed for the factory planning scenario. Of course, this facilitates the creation of user-friendly wizards that guide through the application. However, the concept scales badly to more general spatial dependencies.

An interesting concept is used by *Spatial Analyzer* which comes from the offline metrology domain [Kine 11]. It aims at the integration of the 3D inspection, assembly-build, and reverse engineering tasks. It supports all kinds of metrologic measurement devices (laser trackers, theodolite, total station, CMM, laser scanner, GPS, photogrammetric products), it furthermore supports the *Optotrak HD* tracker [Opto 11]. A generic user interface is provided to make hardware interchangeable and increases productivity for the user. For example, various kinds of laser trackers from different vendors can be

controlled interchangeably in a single dialog. The full pipeline from importing CAD data from various sources, registering it with the physical measurements, interaction, and metrologic analysis and reporting is provided. Different algorithms for object registration are provided. However, *Spatial Analyzer* is not optimized for real-time tracking, in particular not for sensor fusion. Furthermore, there is no support for the rigorous handling of arbitrary SRGs.

4. Approach

At *Fachgebiet Augmented Reality* [FAR 11], we have been working across various projects (*DySenNetz* [DySe 11], *trackframe* [trac 11], *PRESENCCIA* [PRES 11], *AVILUS* [AVIL 11]) in a team of researchers, partially in collaboration with colleagues from *Technische Universität Graz—Institute for Computer Graphics and Vision* [ICG 11] and *University of Cambridge—Computer Laboratory* [CL 11], on a general purpose ubiquitous tracking framework called *Ubitrack*. In the following sections, the general approach, focus, and contribution of this thesis as a part of this effort is pointed out.

4.1. General Approach

One of the core concepts of *Ubitrack* is the *spatial relationship graph (SRG)* [Newm 04]. It is a directed cyclic graph whose nodes represent coordinate frames and whose edges represent spatial transformations between these coordinate frames. An SRG is composed of subgraphs, so-called *spatial relationship patterns*, atomary and reusable building blocks describing the spatial context of sensors and objects as provided by tracking systems and geometric algorithms [Pust 06b]. These patterns directly relate to the operational units needed for the real-time tracking data flow [Pust 06a], opening up the possibility for a semantic way of describing tracking infrastructures.

The requirements stated in 2.3 are tackled following a two-fold approach. First, the SRG approach is transformed into an intuitive user interface for IAR-engineers. It is realized in terms of a software suite that is integrated with the *Ubitrack* tracking framework [Hube 07] [Pust 06a]. Most notably, a graphical spatial relationship graph and data flow editor called *trackman* has been developed, combining the advantages of both, the semantic SRG and the operational data flow perspecitve, in a unique, dual approach. *trackman* is a generic tool and reflects the generic approach of *Ubitrack*. The experience of multiple AR/IAR scenarios was used to define its functionality.

In addition to this generic foundation, selected solutions are provided, with a particular focus on the investigated sample scenarios. The *Ubitrack* framework itself is extended for this purpose, where necessary, to provide the functionality needed for a solution. Other aspects that are not directly related to *Ubitrack* and real-time tracking, especially certain error analysis features, are tackled in the "appropriate manner". Partially, the required concepts are implemented in terms of *Ubitrack* data flows, such as a Monte Carlo simulation framework. In addition to that, external tools and scripts are also used.

4.2. Focus

The main goal of this thesis is to help in solving complex tracking problems in heterogeneous, wide-area industrial environments with a focus on sensor fusion. However, it follows a generic approach and is not targeted to specific application problems. All applications that rely on real-time tracking—not only in the industrial domain—can potentially benefit. In the following, the relations of this thesis to related questions in the context of (I)AR is clarified.

4.2.1. Relations to other Aspects of IAR Applications

The proposed approach aims particularly at solving the tracking problem. Other issues that arise in the context of IAR applications and have been mentioned in 1.2, such as virtual content acquisition and handling, visualization and display devices, are generally out of scope of this thesis. This is reasonable since from a software architecture point of view, tracking shall be completely separated from these issues (see 2.3). Nevertheless, there are some relationships to tracking that expand into this thesis. Displays have to be registered with the scene or tracked in real-time. Furthermore, content acquisition might also be related, as far as offline metrologic devices or real-time tracking systems are incorporated into this process.

4.2.2. Sensor Fusion

In 2.3.2, a taxonomy of different sensor fusion approaches has been given. In particular, techniques based on the Kalman filter [Kalm 60] have been subject to thorough investigations in the last years. A good introduction to Kalman filtering in general has been provided by Welch [Welc 01]. Based on this technique, he also developed the general *SCAAT* pose tracking fusion framework already mentioned in 2.3.2 [Welc 96] [Welc 97]. Hoff fused an outside-in IR tracker with an inside-out optical tracker to increase the head pose accuracy [Hoff 00]. Pustka fused 6DoF poses from an outside-in IR tracking system with orientation measurements of a gyroscope, to make HMD visualizations more stable under fast movements of the head [Pust 08].

The *Ubitrack* framework implements a variety of fusion facilities. Our reference scenarios, the intelligent welding gun and the augmented airplane cabin, both make use of cooperative fusion approaches. In both scenarios, independent measurement modalities including offline metrologic equipment are concatenated to transform the pose of the tracked object to world coordinates. In addition to that, a tailor-made competitive fusion approach is described to reduce rotation errors in the intelligent welding gun scenario. A more general treatment of sensor fusion is out of scope of this thesis, please refer to [Pust 06b] [Pust 08].

4.2.3. Dynamic Reconfiguration of Sensors

There are attempts to reconfigure automatically the constellation of sensors which contribute to tracking at runtime, in a way that is transparent for the application relying

on this tracking [Hube 07] [Pust 11]. Automatic reconfiguration of the sensors might be desirable in environments where the type or number of sensors changes dynamically at runtime.

In industrial setups, however, validated accuracy and robustness are required, which can only be achieved on the basis of a static and deterministic sensor infrastructure. Furthermore, the transparent introduction of a new sensor at runtime often requires the registration of either the sensor (outside-in) or markers (inside-out) with the scene to be known in advance. Otherwise, a dedicated calibration or registration routine would have to be executed. Such a routine, however, necessitates user interaction, rendering the automatic integration more or less useless.

Therefore, dynamic reconfiguration could make only sense when markers are assumed to be installed in the environment and an a priori unknown number of precalibrated inside-out trackers[1] were to be integrated on-the-fly. However, the verification and validation of industrial processes is difficult enough. To maintain the desired quality-of-service, concrete specifications are needed.

Dynamic reconfiguration could also play a role in very large-scale environments that contain too many sensors to be incorporated in one constellation without degrading overall system performance. This case, however, is far beyond the scope of the investigated scenarios. This, however, is not of interest for IAR.

Therefore, a strictly static sensor constellation is pursued in the context of this thesis. Some dynamics can still be achieved by a hard-wired sensor setup by following a complementary or competitive fusion approach (see also 2.3.1). This way, a dynamic reconfiguration can be circumvented by explicitly specifying all potential sensor configurations in advance, which means that the latter have to be known a priori. Furthermore, the approach does not scale very well to scenarios with large numbers of sensors or objects. Both problems are typically not present in the targeted IAR applications (see also 1.3.2).

4.2.4. Dynamic State Changes

Similar to the reconfiguration of sensors at runtime, one might also ask for an automatic adoption of the tracking to application state changes. Such state changes might be necessary due to a change of the involved sensors, virtual or physical objects, users, or interaction devices. The difference to dynamic reconfiguration of sensors above (see 4.2.3) is that this can and must not be handled transparently any more, rather the application initiates those state changes or has to react to them.

An application state change might require to change the constellation not only of involved sensors, but also other physical or virtual objects in the scene, e.g., a marker or the world coordinate frame. The SRG, and the data low description derived from it, are stateless, though. Each state change requires a change in the SRG. The IAR applications listed in 1.3 do not have a complex state logic, at least not concerning the tracking. Typically, only one or two items are to be tracked continually.

[1] For example the *NDI Optotrak* would fulfill this requirement

Generally, state changes are in the responsibility of the application. This responsibility can however be delegated to a workflow engine, an additional abstraction layer between the tracking and the application. Ambitions in this direction have been described by Misslinger in the context of the commercial *Unifeye* [meta 11] tracking suite [Miss 08]. It is based on a finite state machine. In a graphical editor, worksteps, so-called *actions* can be defined and arranged in the desired sequence. For each workstep, tracking, displays, virtual content, and interaction devices are parameterized accordingly. Transitions between worksteps can be triggered manually (for development) or by the occurence of standard or customized events defined by logical functions of user input, spatial relationships such as "adjacency", and automatic timers. Misslinger also gives a good review of previous work in this field. Another approach based on Petri nets to represent application states has been presented by Sandor in the context of the DWARF tracking framework [MacW 03] [Sand 05]. Noteworthy is also [Coqu 04] and *DART*, an AR add-on for *Macromedia Director* [Adob 11] [MacI 04].

Often, such state changes require the instantiation of a new data flow. These states are either known in advance and a set of manually pre-modeled SRGs can be provided to the application for these states. More difficult to handle are state changes that are unknown a priori. The AR workflow engine or application can then rely only on solution fragments. This issue is picked up again in 8.2.

An alternative to this deterministic way of data flow generation is the incorporation of a *Ubitrack* server [Hube 07]. As already stated above, this is mainly suited for large-scale distributed and dynamic environments. The server maintains a world-SRG based on the descriptions of basic spatial relationships provided by all registered clients. Based on this, it is able to serve client queries for derived spatial relationships, based on an automatic pattern-matching approach.

Workflow engines as well as the *Ubitrack* server are beyond the scope of this thesis, due to their limit relevance for IAR. The topic of integrating manually modeled static SRGs with these concepts are touched upon again in 5 and 8. It is shown that throughout all phases mentioned above, the SRG concept provides invaluable support.

4.2.5. Relations to Metrology

Metrology plays an important role for industrial applications. MR/AR interactive applications have to be integrated into this context. The SRG is in many respects similar to a geodetic network. There is a huge potential to integrate the presented concepts with design resources used in metrology [Niem 08].

This might apply for a tool like *Spatial Analyzer* [Kine 11] (see also Chapter 3). It comes from the offline metrology domain, supports all kinds of devices such as theodolites, total stations, laser scanners and photogrammetric devices. It follows an integrated approach considering 3D data acquisition, registration, visualization, real-time and offline measurements, data analysis, and reporting. The SRG concepts could well be adapted to such a tool, in order to improve its usability and reduce potential sources of error.

4.3. Contribution

In this thesis, generic planning and analysis tools are presented which support the IAR engineer throughout the various phases of two typical, industrial scenarios ranging from definition over deployment to operation and maintenance. Those tools are based on the semantic modeling concept called spatial relationship graph (SRG) which provides an abstraction layer on top of the algorithmic data flow layer. This concept eases the specication of known as well as the deduction of new spatial relationships between entities in the scene. A graphical SRG editor is presented for this purpose. Modeling is based on reusable patterns representing the underlying sensor drivers or algorithms. Recurring constellations in the scene can be condensed into easily reusable *meta-patterns* that lend themselves to the evolvement of best-practice solutions. Several exemplary meta-patterns are discussed in this context to emphasize their usefulness, a catalogue is likely to evolve in the future. The process is further simplified by semi-automatic modeling techniques which automize trivial steps. Dataow networks can be generated automatically from the SRG and are guaranteed to be semantically correct. Still, the data flow layer remains directly accessible through a unique *round-trip* engineering approach. Part II deals with these aspects of data flow management.

Uncertainty management is an important issue and has to be considered throughout all phases of application development. Critical steps are identified and generic tools and techniques are presented to solve these tasks. Simulation covers the definition stage where empirical measurements are not yet available. It is also useful during deployment since it makes empirical measurements more understandable. The assessment of accuracy of a concrete setup, based on real measurement data is indispensible to provide accuracy guarantees in the deployment phase. Many industrial procedures require quick setup and dismantling of the tracking system resulting in the need for simple recalibration procedures and maintenance effort. Ideally, such procedures can be performed through on site personnel without requirering detailed technological insight. Nevertheless, the accuracy of such an installation has to comply with production tolerances at all times. Methods are described to continually assert tracking accuracy during the operation and maintenance phase. Part III deals with these aspects of uncertainty management.

In summary, the proposed approach reduces the amount of expert knowledge that is needed for the administration of tracking setups. At the same time, sophisticated setups become manageable. Part IV reviews the approaches described in Parts II and III, according to the requirements specified in 2.3.

Part II.

Data Flow Management

Part II deals with the manual specification of efficient *Ubitrack* tracking data flows, based on the concepts of *SRGs* and *spatial relationship patterns* [Newm 04] [Pust 06b].

Despite the automatic data flow generation facilities referred to in 4.2, manual modeling facilities are indispensable. Even if a *Ubitrack* server for automatic pattern matching is available, domain-specific specifications still need to be done manually such that the server can compute useful results. This incorporates the specification of sensors, markers, and objects as well as the determination of static and tracked transformations. Furthermore, static transformations have to be estimated in advance, a task that is quite difficult to be automated, as will be seen. Concepts to reach that goal have been described in an accompanying thesis [Pust 11]. Last but not least, IAR environments need proven reliability instead of unreliable automatisms.

This part starts with a general review of the *Ubitrack* framework and the SRG concept in Chapter 5. This is followed by a description of the graphical modeling concepts in Chapter 6. Based on this, some common practical modeling tasks are presented in Chapter 7. This includes also the two sample scenarios described in 1.3 as well as best-practice solutions to common registration problems. These practical examples show that modeling on primitive *Ubitrack* data flow components can become quite complex. Therefore, advanced modeling concepts are described in Chapter 8. They aim at dealing with the complexity of complex scenarios by introducing a higher level of abstraction as well as enabling reusability of existing building blocks. This is followed by a review of the presented modeling concepts in Chapter 9.

5. The Ubitrack Tracking Middleware

The *Ubitrack* [Ubi 11] tracking middleware provides a higher level of abstraction, as compared to the data flow concepts of other tracking libraries referred to in Chapter 3 [Newm 04]. It features a layered architecture as depicted in Figure 5.1.

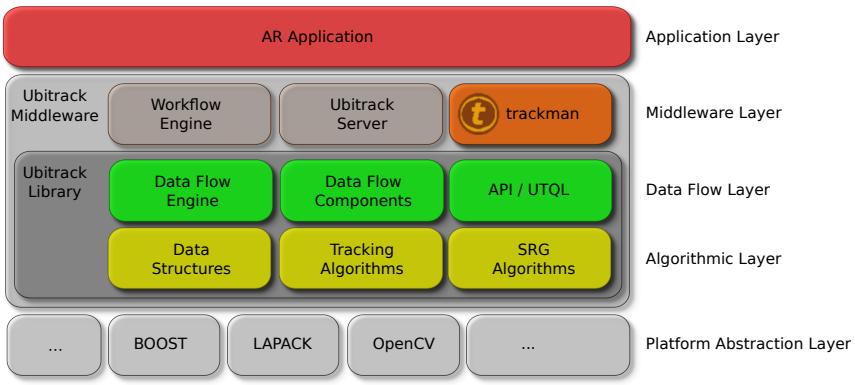


Figure 5.1.: Ubitrack layered architecture

The *algorithmic layer* provides elementary data structures for the representation of spatial measurements including timestamps, handling of spatial relationship graphs and data flow networks, numerical algorithms, as well as elementary registration and fusion algorithms.

The *data flow layer* provides a lightweight and dynamically reconfigurable framework for the instantiation of data flows having the topology of a directed graph. The data flow is energised by sensor measurements which are then processed by various data flow components and the results are passed on the sinks of the data flow [Pust 06a].

The algorithmic and data flow layers together represent the *Ubitrack* library, which can be used directly by third-party applications, via its API. Based on the common notion of spatial measurements and the lightweight implementation of a runtime engine for data flow components, the *Ubitrack* library eases the integration of additional functionality and allows for a generalized processing of spatial measurements. In his theoretic description of a fusion framework referred to in 2.3.2, Durrant-Whyte also mentions the necessity of a canocial description format for the representation of measurements from the involved sensors. He explicitly mentions geometric primitives such as line segments or planes as canonical formats, such as needed in computer vision algorithms. In the context of *Ubitrack*, the major canonical descriptions, in particular 3D positional and orientational transformations, are given in terms of the data structures provided by the

algorithmic layer.

On top is the *middleware layer*. It provides additional services which can be retained by IAR engineers and applications on an optional basis. The graphical tool *trackman* supports the process of interactively describing the physical sensor and marker arrangements. It is described in detail in Chapter 6. The resulting spatial relationship graphs (SRG) not only lend themselves to long-term documentation (and visualization) of the modeled sensor arrangement, as shown below. They also serve as a basis for the specification of data flow graphs (DFG) that can be directly instantiated by the *Ubitrack* library. These DFGs can either be created manually or semi-automatically in *trackman*, fully integrated with the already existing SRG, or generated fully automatically by a *Ubitrack* server [Hube 07]. Architecturally, the yet non-existing workflow engine would also reside on the middleware layer. The middleware layer thus provides a more intuitive way of describing data flows. Thereby, the risk of accidentally specifying physically impossible data flows is excluded conceptually. As already stated in 4.2, we focus on static tracking setups, i.e. on *trackman*.

The *Ubitrack* library provides a rather simplistic API for instantiating, starting, and stopping a data flow network. The complex specification of data flows for tracking setups is covered by UTQL, an XML-based description format that is used to parametrize the library [Pust 07]. This allows for more flexible ways of configuration on the middleware layer, instead of a mere programmatic setup by the AR application. This chapter concentrates on the *Ubitrack* library, reviewing the basic concepts of spatial relationship graphs, data flow networks and spatial relationship patterns, as far as graphical modeling is concerned. For more details, see [Hube 07] [Newm 04] [Pust 06a] [Pust 06b] [Pust 07] [Pust 08]. In the simplest case, the XML description needed to parametrize the library, can be edited manually by the IAR-engineer, though this is a tedious task. The following chapters present more advanced concepts based on manual and semi-automatic modeling.

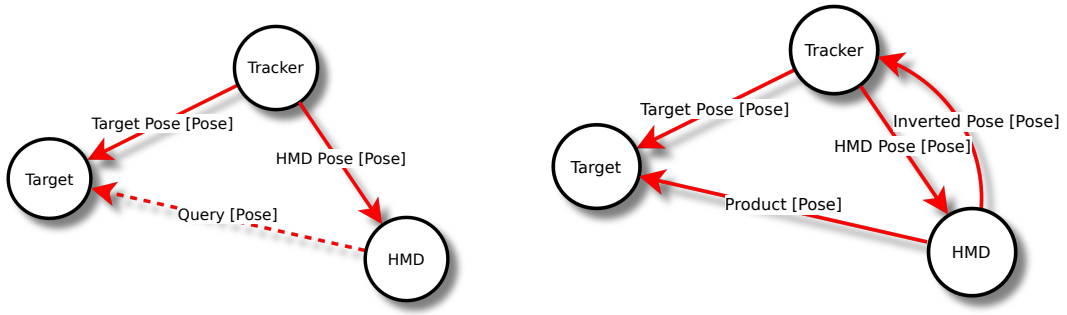
5.1. Spatial Relationship Graphs

The concept of *spatial relationship graphs (SRGs)* was proposed in [Newm 04]. It opens up the possibility for a semantic, high-level specification of tracking infrastructures.

5.1.1. Definition

A simple example is shown in Figure 5.2(a). SRGs are directed cyclic graphs which capture the structure of a tracking environment by describing the static and dynamic spatial properties of objects in the environment. The nodes of SRGs represent the local coordinate frames of real or virtual objects and sensors. A directed edge from A to B represents the spatial transformation $\boldsymbol{H}_A^B$ between coordinate frames A and B, i.e., the pose (position and orientation, 6DoF) of coordinate frame B relative to frame A. Analogously, the transformation $\boldsymbol{p}_{A_i} = \boldsymbol{H}_A^B \boldsymbol{p}_{B_i}$ transforms points $\boldsymbol{p}_{B_i}$ in frame B to $\boldsymbol{p}_{A_i}$ in frame A.

In Figure 5.2(a), the top node refers to a tracker that tracks the poses of two objects, represented by the bottom two nodes. The right one represents a head-mounted display, the left one refers to a generic target. For instance, the applications might want to render some virtual object on top of it. The directed solid edges represent the (dynamically changing) spatial relationships between the tracker and the two objects.



(a) Base SRG (b) Full SRG

Figure 5.2.: (a) Exemplary SRG with a sensor tracking two targets. The solid lines depict tracking information that is continually measured and updated. The dashed line depicts spatial information that can be derived from the tracking data. (b) Additional solid lines indicate the respective mathematical operations (inversion, concatenation) on the tracking data.

The dashed edge in the *base SRG* shown in Figure 5.2(a) describes a request to determine the spatial relationship between the tracked HMD and the tracked generic target. This relationship is not measured directly but can be derived by considering the known spatial relationships of both objects to the tracker. Figure 5.2(b) shows the resulting *full SRG* that does not contain any more dashed edges. Two new edges have been added to the SRG: one reversing the spatial relationship between the tracker and the HMD, and the other one replacing the dashed edge by concatenating the inverted edge from the HMD to the tracker with the edge from the tracker to the generic target.

5.1.2. Edge Characteristics

SRG edges represent a number of sensing characteristics that have to be considered in the manual or automated data flow construction process [Pust 06a]. The following two properties are immanent to the pattern and may only be changed at construction time.

Data type Most importantly, a spatial transformation can have varying degrees of freedom (DoF), ranging from 2 or 3 translational DoF for wide-area sensors such as GPS or WiFi based trackers, over 3 rotational DoF for sensors such as gyroscopes and compasses to full 6DoF poses of optical feature-based or marker-based trackers for small-area VR or AR setups [Pust 08]. Accordingly, the most important

data types are 2D Position, 3D Position, 3D Rotation, and Pose (6DoF position and orientation). They also exist in an extended version including uncertainty information, e.g., 3D Error Position and Error Pose. Instead of a single measurement, an edge can also represent a bunch of measurements. This is particularly useful to describe large or a priori unknown numbers of measurements, such as a cloud of feature points in 2D or 3D. For this, the data types 2D Position List, 3D Position List, 3D Rotation List, and Pose List are used. More complex aggregates are not allowed. Pose is the most used data type throughout this thesis; unlike in Figure 5.2 it is often not explicitly stated in the remaining SRG diagrams to compactify the representation.

Synchronization type Another important property of an edge describes synchronization issues. If the spatial transformation associated with the edge can be updated at discrete points in time only, e.g., by a sensor providing updates at a constant framerate, the synchronization type is PUSH. If the spatial transformation can be updated at any point in time, the synchronization type is PULL. The complexities arising from this distinction are discussed in 5.4. In SRG diagrams throughout this thesis, edges of type PUSH are depicted in a reddish color (as in Figure 5.2) whereas edges of type PULL are depicted in green.

At runtime, an SRG edge represents the *measurements* of a certain spatial transformation between two coordinate systems (nodes) over time. In general, new estimates for the spatial transformation may arrive in real-time. A measurement consists of a timestamp and the actual payload according to the chosen *data type*. The timestamp is relevant for data flow synchronization purposes, especially if several sensors are used.

5.1.3. Related Concepts

SRGs are similar to scene graphs in computer graphics [Stra 92]. Yet, in contrast to scene graphs, SRGs do not imply a pre-defined hierarchical ordering of the nodes. Rather, SRGs are cyclic directed graphs, unlike scene graphs that have a tree-like topology. Applications can request any node in an SRG to assume the role of a root node, requiring the traversal through parts of the SRG from this node to a specified leave node. For a detailed comparison of scene graphs and SRGs, see [Echt 08].

SRGs also resemble geodetic networks in metrology where 2D or 3D points are estimated with a theodolite [Niem 08]. Figure 5.3 depicts a simple geodetic network.

Usually, the theodolite is placed at different locations and the distances and angles towards other points are measured from there. In 3D, the coordinates of a new point can be determined from the azimuth and elevation angles as well as the distance measured from a single known point. Such measurements are provided by a total station, for example. Measuring the same point from different known locations results in an overdetermined estimation problem. In the example, there are no known points, rather all unknown points are estimated based on measurements from several other unknown points. Since no absolute coordinates are given, only the *inner geometry* of the network can be estimated. Note that the geodetic network contains cycles if and only if

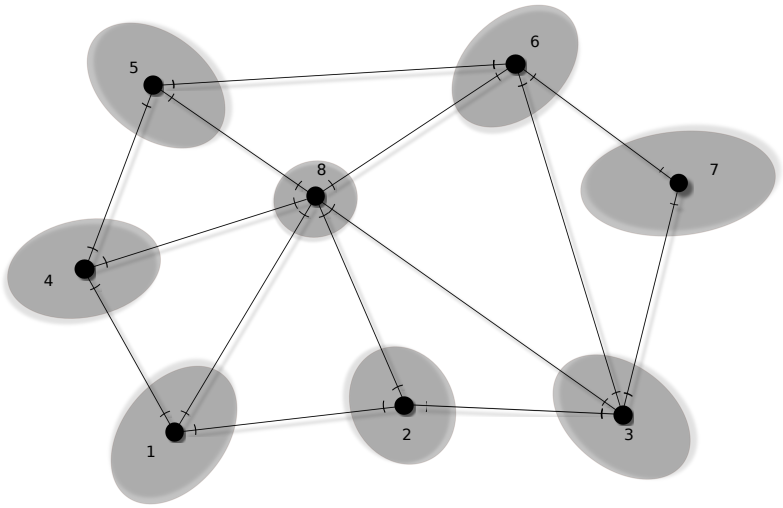


Figure 5.3.: Simple planar geodetic network. The points represent coordinates, measured distances are represented by edges, measured azimuth angles are indicated by small slashes orthogonal to the edges. The ellipses represent the covariances of the estimated points. (Example according to [Niem 08])

the estimation problem is overdetermined. By removing some of the edges of the network one would ultimately obtain a minimum spanning tree resembling a scene graph again. Therefore, a major difference between an SRG and a scene graph is the potential overdetermination of spatial measurements.

An SRG may also contain cycles in the non-overdetermined case, merely by making implicit information explicit, such as the SRG depicted in Figure 5.2(b). The edge between *HMD* and *target* does not represent new information, it is inferred from the other two edges.

If multiple independent trackers are tracking the same objects, however, and the transformations between those trackers are known, the SRG might contain multiple paths to determine the spatial relationship in question. Such cycles are essential for exploiting redundancy in tracking setups. They provide a means to register sensors or objects by using complementary tracking information (i.e., an alternate path in the graph). They also allow system monitoring and the detection of faulty (miscalibrated) sensors. This is picked up again in 14.2.

5.2. Data Flow Networks

The SRG provides an abstraction of the data flow layer. It is descriptive rather than an operational specification of a certain setup and is not directly usable by an application. Rather, for efficient use by the *Ubitrack* runtime system that is included into the appli-

cations, the SRG has to be converted into a *data flow network (DFN)*. DFNs consist of computational units that operate on tracking data.

A DFN is an instance of a *data flow graph (DFG)*. DFGs are directed graphs and their nodes specify the *data flow components* to be instantiated in the DFN. In general, a data flow component has *input ports* and *output ports*. An input port consumes data needed by the component; it is represented by a dashed edge in the SRG unless a provider of the data is specified (cf. Figures 5.2(a) and 5.2(b)). An output port provides the computation result; it corresponds to a solid edge in the SRG. In the DFG, edges represent the flow of tracking data between data flow components; they always connect an output port with an input port. An input port is connected to exactly one output port, whereas an output port can be connected to several input ports. Sources in a DFG generally represent sources of tracking data (i.e. tracking devices). Sinks correspond to interfaces to applications or to other data flow graphs.

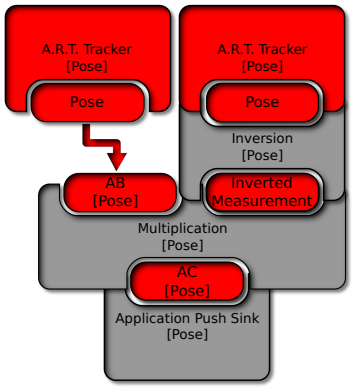


Figure 5.4.: DFG corresponding to the SRG shown in Figure 5.2(b). An inversion, followed by a multiplication is needed to compute the dashed edge of Figure 5.2(a) from the two solid edges. Connections between ports have been omitted where possible to compactify the representation.

Figure 5.4 shows the DFG that computes the spatial relationship between the HMD and the target in the SRG of Figure 5.2. The tracking data of the HMD is inverted and then concatenated with the tracking data of the target. The red color indicates that measurement updates are driven by the sensors, which are completely colored red. The short term *SRG* is sometimes used to denote a *full SRG* including all the derived spatial relationships and its corresponding DFG if it is clear from the context that the DFG is meant.

In general, cycles in the DFG are allowd, resulting in a directed cyclic graph structure. Often, however, the DFG adopts the topology of a tree with a single sink component (the transformation needed by the application) as its root, as in the example depicted in Figure 5.4.

The *Ubitrack* runtime environment executes such data flow networks. At this level,

Ubitrack is comparable to the approaches taken in other systems described in 3 such as OpenTracker [Reit 01]. In fact, we are able to export data flow networks that have been generated from SRGs into applications that use OpenTracker, using the Ubiquitous Tracking Query Language (UTQL) XML data exchange format [Pust 07][Newm 07]. As will be seen, the DFG is sometimes needed in addition to the SRG for expert modeling, especially when synchronization issues have to be solved.

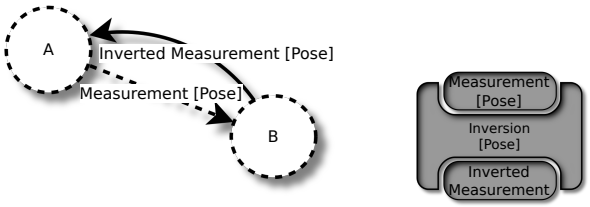
5.3. Spatial Relationship Patterns

The previous section have introduced the basic concepts of working with SRGs and patterns, using a rather simplistic example. This section describes formally, how SRGs can be transformed into DFGs. To this end, Pustka introduced the concept of *spatial relationship patterns* [Pust 06b]. They are the elementary building blocks that allow the user to make use of the diversity of sensor drivers, algorithms, and application interfaces provided by the *Ubitrack* framework.

5.3.1. Definition

A pattern is a template SRG that corresponds to a computational unit called a *data flow component*, i.e. a node, in a DFG. Each pattern / data flow component is associated with a *pattern ID* that is unique throughout the whole DFG. It is needed to interface particular data flow components from the outside. For this purpose, the pattern ID has to be chosen deliberately by the AR engineer for some components.

Tracking data is provided to the data flow component via its *input ports*. The data is used for the computation of some result which is then sent out via its *output*. Good examples are the Inversion and Multiplication components in the DFG shown in Figure 5.4.

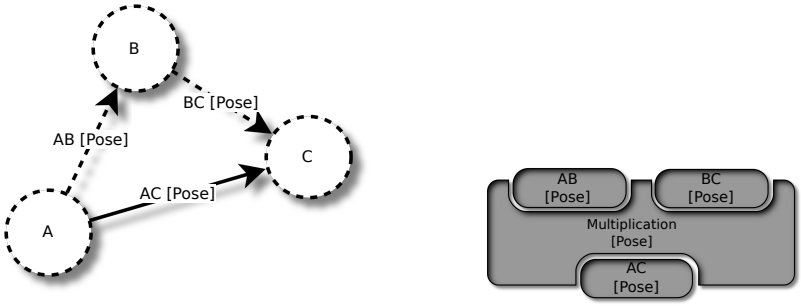


(a) Full pattern

(b) Data flow component

Figure 5.5.: Inversion basic spatial relationship pattern and corresponding data flow component for the inversion of spatial relationships or edges in the SRG

Spatial relationship patterns describe the effect of a computational unit on an SRG. For example, the Inversion pattern/component in Figure 5.5 states that, for a given SRG edge from node A to B, a new edge is added to the SRG going in the reverse direction. Similarly, the Multiplication pattern/component in Figure 5.6 states that an

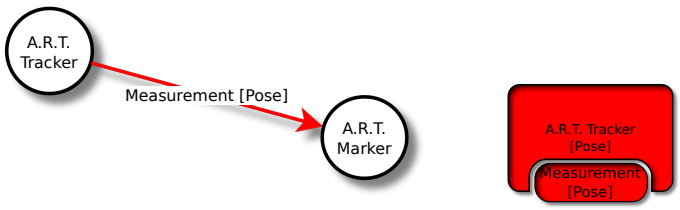


(a) Full pattern (b) Data flow component

Figure 5.6.: Multiplication basic spatial relationship pattern and corresponding data flow component for the concatenation of spatial relationships or edges in the SRG

SRG edge from A to B, and an edge from B to C can be concatenated by multiplying the transformations. As a result, a new edge from A to C can be added to the SRG.

Component inputs in the DFG correspond to *input edges* of the pattern. They are shown as dashed lines, and the associated *input nodes* as dashed circles. The resulting component output in the DFG corresponds to an *output edge* of the pattern. It is shown by a solid line, and the additionally added *output nodes* (if any) by a solid circle. In other words, the dashed part, the *input section* of the pattern has to be already available in the SRG such that the pattern can be applied and the solid part, the *output section*, can be added to the SRG.

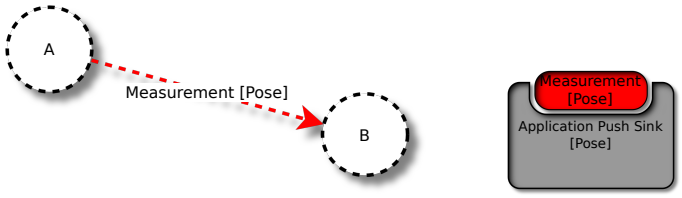


(a) Base pattern (b) Data flow source component

Figure 5.7.: Spatial relationship base pattern and corresponding data flow source component representing a typical tracking device

Syntactical compliance of the input section of an additional pattern with the existing SRG is the basic requirement for embedding this pattern. As an additional constraint, the data types and synchronization types (see 5.1.2) of the involved input and output edges have to comply or embeddeding is not allowed.

Some patterns/components, however, such as those depicted in Figures 5.5 and 5.6 above, have edges/ports whose synchronization type is initially undetermined. Unless



(a) Query pattern

(b) Data flow sink component

Figure 5.8.: Application Push Sink spatial relationship query pattern and corresponding data flow sink component representing a typical application interface to the *Ubitrack* data flow.

the synchronization type of such an edge/port becomes determined by embedding the pattern in the SRG, the edge is depicted in black and the data flow port in gray. The concrete mechanisms for this are detailed below in 5.4.1.

Either the input or output section of a pattern may be empty. According to this *pattern signature*, three different types can be distinguished:

Base patterns have an empty input section. The Tracker pattern depicted in Figure 5.7(a) is an example of that category. The corresponding component is depicted in Figure 5.7(b). Since it has an empty input section, no requirements with respect to the current SRG have to be fulfilled and it can always be added to the SRG. It is used to add tracking devices as source computational units to the DFG. One might think of an axiom in formal logic. From the data flow point-of-view, base patterns represent data sources injecting tracking data into the data flow network.

Full patterns such as shown in Figures 5.5 and 5.6 have both, a non-empty input and a non-empty output section. They may be applied if the graph structure described in the input section can be matched to the current SRG. The graph structure described by the output section is then added, resulting in an enriched SRG. This corresponds to a rule of inference in formal logic, having a set of premises as well as a conclusion. From the data flow point-of-view, full patterns deduce new information from existing information.

Query patterns have an empty output section. They connect the tracking setup to an application, in form of a query for information about a specific spatial relationship in a scene. An example is the pose of the target relative to the HMD in Figure 5.2(a). This is shown in Figure 5.8 on the example of the Application Push Sink pattern. Query patterns can be applied if the input section matches (like full patterns), however, no new information is added to the SRG. From the data flow point-of-view, query patterns represent data sinks transferring tracking data

to applications, storing it in a file or sending it to a corresponding data source in another data flow network.

Due to the correspondence of spatial relationhip pattern and computational unit, base/full/query patterns are atomic and cannot be dissected. They represent the elementary building blocks from which more complex SRG structures can be built.

5.3.2. A Catalogue of Spatial Relationship Patterns

A large number of patterns and associated computational units have already been integrated into *Ubitrack* [Pust 06b][1] On the one hand, this results from a great variety of sensor drivers and algorithms provided by the *Ubitrack* library. On the other hand, many patterns exist in multiple versions, differing only in the transformation type of their input and output edges. For example, among others, there are Inversion patterns for edges of type Pose, 3D Position and 3D Rotation. To support the user in dealing with this pattern diversity, a schema is needed for their categorization.

The most important aspect of a pattern is its structure, which allows for a categorization of patterns in base, full, and query patterns, as already stated in 5.3.1. From the data flow point-of-view, base patterns represent data sources that inject tracking data into the network, full patterns transform it until it finally reaches sinks in terms of query patterns.

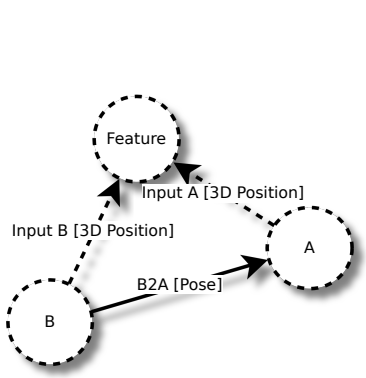
This point of view only considers the structural aspects of pattern, in other words their syntax. Alternatively, one might examine patterns according to their intended purpose, or their semantics. The following major categories can be distinguished:

Sensor-/tracker patterns describe how tracking data is provided to the data flow network. This mainly comprises driver components retrieving data from hardware, such as shown in Figure 5.7. Sensor patterns typically have synchronization type PUSH; they deliver updated measurements at the framerate of the sensor, e.g., 50 fps for a camera.
Other tracker patterns represent tracking algorithms that require input data from another sensor pattern, such as the PTAM (Parallel Tracking and Mapping) [Klei 07] natural feature tracking pattern depicted in Figure 5.11(a) which needs a camera image provided by some Framegrabber pattern in order to compute the pose of the Camera with respect to a pre-trained Feature Map.

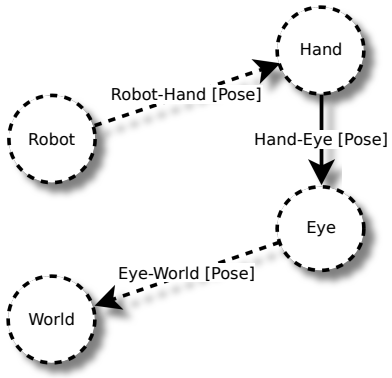
Basic patterns describe trivial transformation steps such as inversion or interpolation of a transformation or concatenation of two transformations. The patterns depicted in Figures 5.5 and 5.6 belong to this category.

Calibration/registration patterns represent algorithms for the estimation of static spatial transformations under certain boundary conditions.
The Absolute Orientation pattern (see Figure 5.9(a)) estimates the Pose transformation between two coordinate frames using at least three corresponding 3D Position

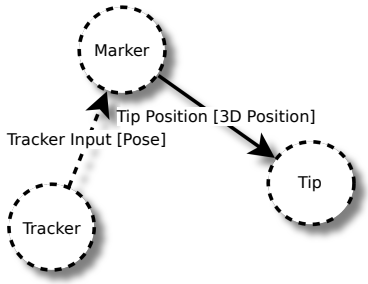
[1] Currently, there are already more than 200 patterns, and this number will probably further increase.



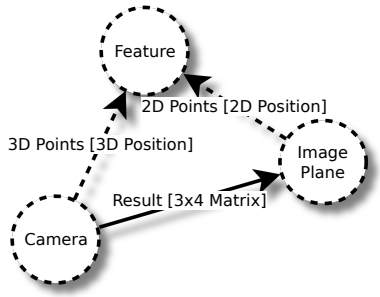
(a) Absolute Orientation



(b) Hand-Eye Calibration

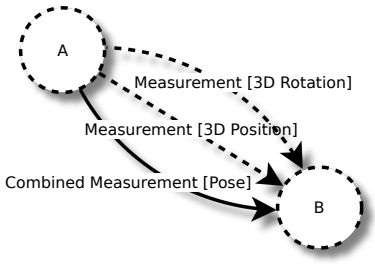


(c) Tip Calibration

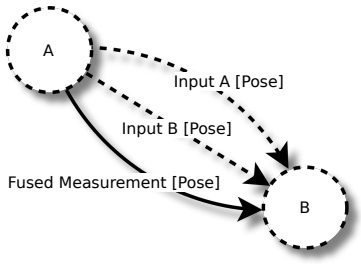


(d) SPAAM Calibration

Figure 5.9.: Selection of calibration and registration patterns

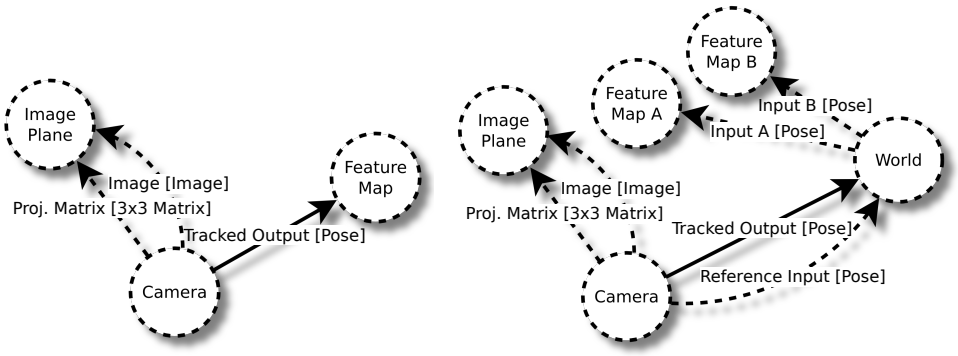


(a) Functional Fusion



(b) Pose Complementary Fusion / Pose Kalman Filter

Figure 5.10.: Selection of fusion patterns



(a) standalone variant (b) cooperative fusion variant

Figure 5.11.: Two variants of the PTAM (Parallel Tracking and Mapping) pattern [Klei 07]

measurements of a common Feature [Horn 87] [Egge 97]. The point features must be in a *general* constellation, i.e. they must not be collinear.
The Hand-Eye Calibration pattern (see Figure 5.9(b)) estimates the Hand-Eye Pose transformation using at least two corresponding Pose measurements for Robot-Hand and Eye-World [Tsai 88]. The underlying algorithm was first used in the robotics domain to estimate the offset of a camera mounted to the arm of a robot. Thereby, the pose of the arm is known from the robot kinematics, as well as the pose of a static reference target seen by the camera.
The Tip Calibration pattern estimates the 3D Position offset of a tip such as in Figure 1.6(b) and 1.6(c) with respect to the marker attached to the end of the device. Different Pose measurements have to be collected for the marker while the tip is kept in a constant position. For this, the tip is put on a surface or a special calibration adapter while the marker descripes a spherical movement. The SPAAM Calibration pattern (see Figure 5.9(d)) estimates a projection matrix with 11DoF (extrinsic and intrinsic camera parameters) from 2D Position and 3D Position measurements [Tsai 87][Tuce 02][Hart 00]. The registration is based on the manual alignment of displayed 2D points with a single 3D point given in world coordinates from various viewpoints. The SPAAM method assumes a marker to be attached rigidly to the display. Knowing the pose of this marker in world coordinates from the tracking system, the single 3D point can always be transferred to the marker coordinate system which allows the display (or head in case of an HMD) to be moved freely during the calibration procedure. At least six *corresponding measurements* in 2D and 3D are needed to solve for the 11DoF.
Many alternative algorithms exist for the solution of the mentioned registration problems. The referenced solution corresponds to the *Ubitrack* implementation. Bauer provides a much broader review [Baue 07]. Pentenrieder investigates different methods in the context of IAR [Pent 09]. The application of the patterns in realistic registration procedures for IAR are described in 7.2.

Fusion patterns represent algorithms for *competitive fusion* or *complementary fusion* (according to 2.3.2) which can be used to somehow fuse tracking data to obtain a better, more accurate or a more general result. The Functional Fusion pattern depicted in Figure 5.10(a) combines 3D Position and 3D Rotation measurements in a single Pose measurement (complementary fusion). Time-complementary fusion can be achieved easily, e.g., using the Pose Complementary Fusion pattern depicted in Figure 5.10(b). It conducts Input A if available, otherwise Input B. The signature of this pattern also describes a Pose Kalman Filter which combines two Pose measurements from independent sources to obtain a single Pose measurement with higher accuracy (competitive fusion). Other variants of the Kalman Filter pattern, differing in the types and numbers of input measurements, are conceivable or already exist in *Ubitrack*.
Simple forms of *cooperative fusion* can be accomplished by a suitable combination of general-purpose patterns in the SRG, rather than by a dedicated "competitive fusion pattern". To realize this, consider the indirect tracking setup introduced in 1.3.2. The mobile tracking system (dependent tracker) can be used only if its pose in the global reference frame is known (independent tracker). Basically, this is the concatentation of two sensor patterns (cf. Figure 5.7), using the Multiplication pattern (cf. Figure 5.6). In this simple example, the dependent tracker does not even know about the independent tracker. In the general cooperative fusion case, however, the dependent tracker obtains initialization data (e.g., for region based image processing) via a dedicated input edge in its pattern. The result is a hybrid tracker- and fusion pattern, such as the variant of the PTAM pattern depicted in Figure 5.11(b) [Klei 07]. In this variant, two pre-trained Feature Maps are available for tracking; their poses with respect to World need to be given. Depending on the Reference Input pose of the Camera with respect to the World, the implementation decides about which Feature Map to use for tracking. Initially, the Reference Pose has to be given by another tracker, in a cooperative fusion setup. Variants with more Feature Maps are available. See also 2.3.2 for a discussion.

Persistence patterns represent components that write tracking or calibration data to a file or read it from there. For persistence patterns, there exist pairs of corresponding base and query patterns, such as Player and Recorder for logging and replaying tracking data or Calibration Reader and Calibration Writer for maintaining the calibration or registration data of a static transformation in files.

Network patterns represent components that send/receive tracking data to/from the network. They are needed to link independent DFNs running on independent *Ubitrack* peers. Such as persistence patterns, network patterns also feature corresponding base and query patterns. Corresponding pairs of Network Sink and Network Source patterns are identified according to their identical pattern IDs.

Application patterns represent components that transfer or receive tracking data to / from an application. This principally enables applications not only to consume tracking data but also to transform it somehow and reinject it into the data flow

network. Such as persistence and network patterns, application patterns also feature corresponding base and query patterns.
The Application Push Source, Application Pull Source, Application Push Sink and Application Pull Sink patterns represent endpoints in the DFN which interface it to the application. The Application Pull Source is one of few data flow sources having type PULL. It retrieves current tracking data at any time via a callback interface from the application. Similarly, the Application Push Sink *pushes* data from the data flow into the application via a callback interface. In both cases, the DFN initiates the flow of tracking data. Application Push Source and Application Pull Sink, on the contrary, work without a callback mechanism and the application initiates the flow of tracking data by pushing updated measurements into the data flow or by requesting updated measurements, e.g., at the framerate of the renderer. The concept of PUSH and PULL is further detailed in 5.4. Regardless which interface is being used, the application identifies the corresponding data flow components by means of their pattern IDs.
Other patterns in this category include render components for development and debug purposes, which allow for simple OpenGL-based 3D graphics output based on tracking data.

Table 5.1 presents a representative subset of patterns, classified according to their structure and semantics. A comprehensive reference is provided at [Ubi 11].

Examples for base and query patterns are shown in Figures 5.7 and 5.8. Most base and query patterns feature the same layout of two nodes connected by a single edge, solid for base patterns and dashed for query patterns. Full patterns such as the Multiplication are more interesting to investigate since they exhibit a broader range of different forms. Most patterns have just been mentioned above, without giving a description of their semantics. More patterns are explained throughout the remainder of this thesis, in the context they are needed.

5.3.3. Node/Edge/Pattern Attributes

The essential characteristics of an edge, *data type* and *synchronization type* have already been introduced in 5.1.2. Furthermore, each node and each pattern is attributed with a *node ID* and a *pattern ID*, respectively. Both IDs have to be unique throughout the whole DFG. The *pattern ID* is needed to interface the data flow network from the outside, as described in 5.3.2. The *node ID* is needed only in special cases, see 5.3.4.

Some data flow components need additional information to provide the desired result. This information is provided to the *Ubitrack* runtime in terms of properties attributed to the output nodes, output edges, and patterns in the DFG. The location thereby depends on the type of information. A concrete pose value is attributed to the edge of e.g., the Static Transformation pattern. A marker ID may be attributed to either the node representing that marker or to the edge pointing towards the marker node. The choice depends on the concrete system. An optical square marker such as depicted in Figure 1.6(a) has a unique ID and can be attributed to the Marker node of the Optical

Table 5.1.: Pattern categorization matrix showing a representative subset of the existing *Ubitrack* patterns. The transformation types are neglected for the sake of readability.

		Syntax		
		Base pattern	Full pattern	Query pattern
Semantics	Sensor-/ tracker	A.R.T. Tracker Static Transformation Highgui Framegrabber	Optical Square-Marker-Tracker PTAM Tracker	
	Basic		Multiplication Inversion Buffer Interpolation Gate Time-To-Space-Expansion Converter Aggregator Signal Generator Trigger Trigger Loop Sampler	
	Calibration		Hand-Eye Calibration Absolute Orientation Tip Calibration SPAAM Calibration Tip Calibration	
	Fusion		Kalman Filter Complementary Fusion Functional Fusion	
	Persistence	Player Calibration Reader		Recorder Calibration Writer
	Network	Network Source		Network Sink
	Application	Application Push Source Application Pull Source		Application Push Sink Application Pull Sink X3D Object Background Image

Square-Marker Pattern. The ID of a tree target such as depicted in Figure 1.6(b) is determined in the proprietary administration software of the vendor. For the unlikely case of two independent tracking systems tracking the same marker, the ID has to be specified twice on the edges since the node then exists only once in the SRG. More general parameters that cannot be associated with a particular node or edge are typically attributed to the whole pattern. For more details about node/edge/pattern attributes,

refer to the UTQL specification [Pust 07].

5.3.4. The Module Mechansim

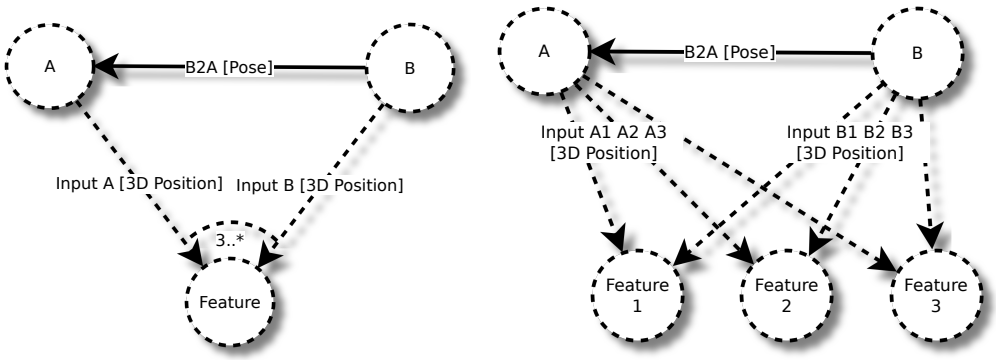
Logically, each pattern represents an independent data flow component. Adding the tracker pattern shown in Figure 5.7 twice would therefore mean to run two independent tracking systems, using two independent driver instances. While this behavior is also possible, the desired behavior probably is to use the same tracking system to track two different markers. However, this also means that both data flow components shall share a common driver instance.

For this, *Ubitrack* provides the *module* concept. It allows data flow components to share common resources. Modules are typically used for sensors drivers (e.g., A.R.T. Tracker) to share hardware devices or bus/network interfaces and also computer vision algorithms (e.g., Optical Square-Marker Tracker) to avoid the redundant execution of algorithms, such as in the example shown in Figure 5.2. A *module* encapsulates the usage of the resource and performs all necessary computations. Furthermore, it administrates a variable number of *thin* data flow components to interface with the data flow network.

To establish the $1 : n$ relationhip between various data flow components and their common module, *component keys* and a *module key* are used. The module key is typically specified in terms of a common node ID. In the example mentioned above, the two A.R.T. Tracker nodes could be parametrized with an identical ID. Note that this is identical to the unification of both nodes since the node ID has to be unique throughout the DFG. To obtain a *component key*, any attribute according to 5.3.3 can be used; its choice depends on the *Ubitrack* implementation of the module. In the mentioned example, the A.R.T. Body ID edge attribute of the Tracked Transformation edge is used. It determines the ID of the marker in the proprietary A.R.T. administration software. In the example of the Optical Square-Marker Tracker component, the module key is given in terms of the ID of the Camera node and the component key is given in terms of the Marker node. The component key has to be unique throughout all components managed by the same module. An important property of the module mechanism is that its complete spatial relationship pattern is replicated an arbitrary number of times.

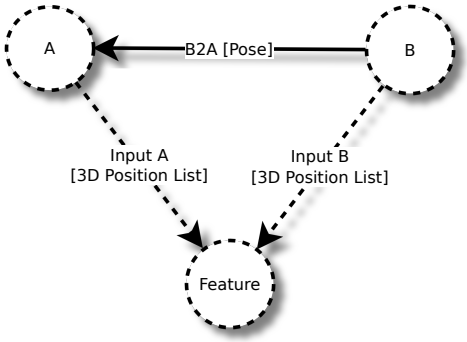
5.3.5. Time-/Space-Expansion

In contrast to the module concept where the complete pattern is replicated an arbitrary number of times, sometimes only those parts of the pattern have to be replicated that describe the input data. Some patterns require several (maybe corresponding pairs of) measurements to be able to compute a result. This applies in particular to the registration patterns introduced in 5.3.2. The Absolute Orientation pattern (see Figure 5.9(a)) for example requires at least 3 corresponding pairs of 3D position measurements. These measurements can either be collected at once or sequentially. Pustka introduced the concurrent concepts of *time-expansion* where the measurements are collected sequentially and *space-expansion* where the measurements are collected in one step, all measurements having the same timestamp [Pust 06b]. The two variants are depicted in Figure 5.12.



(a) Time-expansion

(b) Explicit space-expansion (three correspondences)



(c) Simplified space-expansion

Figure 5.12.: Time- and space-expansion variants of the Absolute Orientation pattern. At least three corresponding point measurements have to be given such that the pose from B to A can be computed.

With time-expansion, the Feature is supposed to be moved around and captured several times, as indicated by the "3..*" in Figure 5.12(a). With space-expansion on the contrary (see Figure 5.12(b)), the Feature node is replicated a fix number of times to make the spatial relationships and the concurrent measurement acquisition explicit. Note that the registration patterns depicted in Figure 5.9 are all represented in their time-expanded form.

Although space-expansion is intuitive for a few Features, it becomes quite confusing when plenty of Features are used. Furthermore, the number of Features to be used has to be known at configuration time, which is not always the case. Therefore, space-expansion can also be represented in the form depicted in Figure 5.12(c). The Feature node made way for a Feature Cloud node representing the concurrent measurements and the Input A and Input B edges convey corresponding lists of 3D position measurements the lengths of which are determined not until runtime. The topic of time-/space-expansion is picked up again in 7.2, where practical solutions for various registration problems are described.

The *Ubitrack* framework eases the implementation of data flow components supporting both time- and space-expansion. For this, the concept of *expansion input ports* is provided which encapsulate this behavior such that no additional effort is needed during implementation. The desired method can be configured by setting the corresponding attribute in the UTQL description [Pust 07]. In case of time-expansion and *explicit space-expansion*, the expansion input port takes care of aggegrating a list of measurements automatically. In case of *simplified space-expansion*, the list of measurements is directly provided to the component.

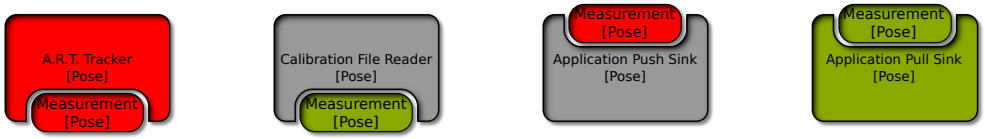
Examples for DFGs based on time-expansion and implicit space-expansion can be found e.g., in 7.2. An example using explicit space-expansion is discussed in 11.4.

5.4. Data Flow Synchronization

Before the synchronization of a DFN can be understood, the synchronization properties of individual data flow components needs to be investigated.

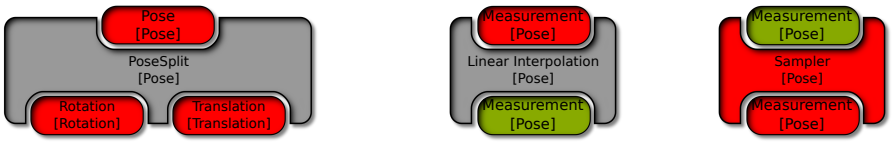
5.4.1. Synchronization of Data Flow Components

To properly handle measurements that are generated asynchronously by independent sensors, each edge in the SRG and thus each port in the DFG is attributed with its synchronization mode, either PUSH or PULL. For many of the patterns/components mentioned above in 5.3.2, the synchronization type has been fixed during implementation of the data flow component. The single output edge of a typical sensor pattern as well as the Player pattern in the SRG, which corresponds to a single output port in the DFG, has synchronization type PUSH. The StaticTransformation, CalibrationReader on the contrary have type PULL. Likewise, there are application patterns providing either a PUSH or PULL input. The IAR-engineer can choose the desired application interface. For these and some other patterns, the synchronization types are fixed. The situation is depicted in Figure 5.13. Also some full patterns use a fixed synchronization scheme, as depicted in Figure 5.14. The color of the ports indicates the synchronizatinon mode, red for PUSH and green for PULL.



(a) Source (PUSH) (b) Source (PULL) (c) Sink (PUSH) (d) Sink (PULL)

Figure 5.13.: Synchronization variants of data flow components for selected base and query patterns. Components driving a PUSH (in particular sensor drivers) or PULL action, are fully colored in red or green, respectively, otherwise only the respective data flow ports are colored.



(a) Pose Split (b) Interpolation (c) Sampler

Figure 5.14.: Data flow components for selected full patterns having a fixed synchronization mode. The Sampler component is fully colored in red because it represents a source of PUSH events.

For other patterns/components, mainly full patterns that compute some kind of result, the synchronization can be handled more flexibly at configuration time. For this, the *Ubitrack* framework provides the concept of *trigger components*. A subset of the input ports and at most one output port of the data flow component can be arranged in a *trigger group*. Unless the synchronization flag of an edge/port belonging to the trigger group has been fixed, it is set to AUTO. The edge in the pattern is depicted in black, the corresponding port is depicted in gray.

The synchronization flags on the inputs are thereby inherited from the outputs they are connected to. The constellation of synchronization flags on the inputs of the trigger group implies the synchronization flag for the single output, according to the following logic:

1. If all input ports of in the trigger group have type PULL then the output port also has type PULL.

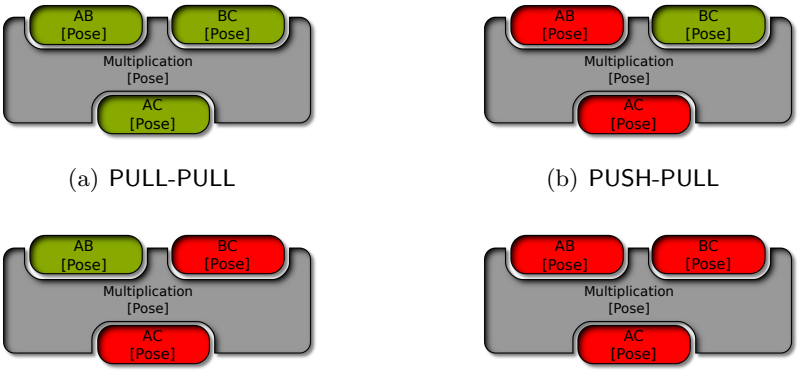
2. If one or more input ports have type PUSH, then the output port also has type PUSH.

3. Several input ports having type PUSH makes sense if and only if these ports are *synchronized* in terms of being fed with *corresponding measurements* having the same timestamp (Synchronized-PUSH). Otherwise, the component does not evaluate and no results will be pushed onwards on the output port.

Note that the Pose Split component depicted in Figure 5.14(a) cannot be implemented as a trigger component, since it has two output ports and only one output is allowd in the trigger group. The behavior of the Translation output when pulling on the Rotation output would be undefined and vice versa. The situation for trigger components is depicted in Figure 5.15 for the special but rather common case of two inputs.

5.4.2. Synchronization of the Data Flow Network

Pushed measurements travel downward from source toward sink through a DFN, e.g., when a tracker such as a camera sends new data into the network at its own speed. Pulled measurements are pulled upward in a DFN. A pull operation may be initiated for



(a) PULL-PULL (b) PUSH-PULL

(c) PULL-PUSH (d) Synchronized-PUSH

Figure 5.15.: Possible synchronization modes of the Multiplication component to demonstrate different combinations of synchronization modes for a trigger component with two inputs

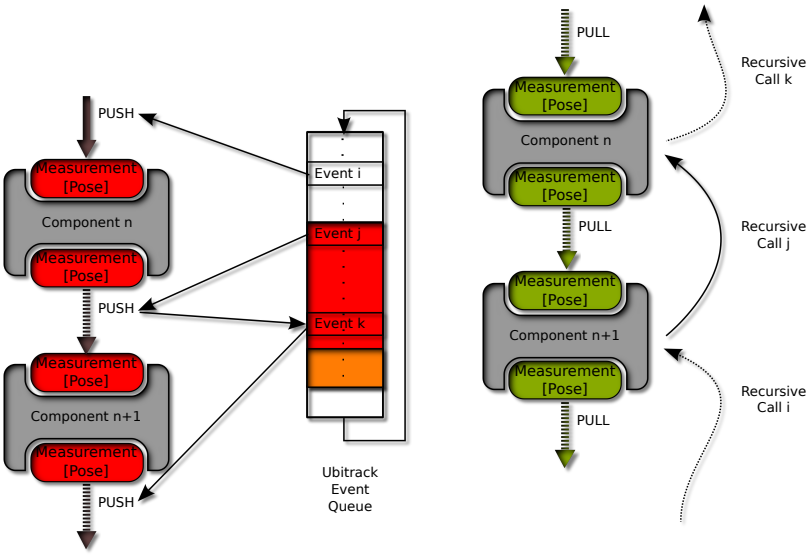
example by an application requesting measurements with a specific timestamp (see 5.1.2) via the ApplicationPullSink component. PUSH as well as PULL events are propagated recursively through the data flow network. Figure 5.16 presents both mechanisms and their implementation in *Ubitrack*.

In addition to the coloring of the ports, data flow components that drive the data flow by actively initiating downward push or upward pull operations are also colored, others are gray, see also Figures 5.4, 5.13, and 5.14.

Pulling measurements upwards in the DFN is implemented internally by recursive function calls. For propagating pushed measurement downwards in the DFN, the *Ubitrack* runtime environment operates an event queue to handle the pushed measurements in the proper sequence. The data flow components are thereby triggered in the direction from sources to sinks. To avoid race conditions, a strict execution sequence is established during instantiation of the DFG. Sinks obtain the lowest priority. In a depth-first graph search starting at the sink(s), the priorities are increased with the distance of the component to the respective sink. These priorities are used to sort events with equal timestamps in the event queue. As depicted in Figure 5.17, this ensures that intermediate results are evaluated in the correct sequence such that no result is reevaluated based on outdated measurements.

Problems arise when two or more unsynchronized inputs have to be combined by a computational unit, such as the Multiplication component, regardless of whether the component is a trigger component or not.[2] The measurements on all inputs need to be valid for the same point in time. When a PULL request occurs on the output, measurements have to be pulled for this timestamp on all inputs. When a push event occurs for

[2]The only difference betwenn a trigger component and a normal component is the point in time of the determination of the synchronization flags.



(a) PUSH (b) PULL

Figure 5.16.: Propagation of measurements in the data flow. The wide arrows represent the handover of measurements, dashed wide arrows represent handovers laying in the future of the depicted situation. (a) Association of handovers with entries in the event queue. The colored part represents yet unprocessed events. Event i has been processed already. It resulted in event j which is currently being processed. Event j will result in event k to be enqueued in the end of the queue for this timestamp. The orange part represents more recent events having a newer timestamp that might have been pushed into the data flow already. They will be processed after the red part either has been fully processed or dropped in case of heavy system load. (b) Requests for measurements are conveyed by recursive function calls. Handover of measurements is accomplished with their return. In the depicted situation, call j is about to execute.

one input edge, measurements for the same timestamp have to be pulled on the other inputs. The result can then be computed and pushed onwards on the output. Generally, it is not possible to have more than one input in PUSH mode. Therefore, all except one of them should be in PULL mode. To this end, suitable conversion facilities must be included, as shown exemplarily in Figure 5.18. The Buffer (constant interpolation), Linear Interpolation and Kalman Filter components convert measurements from PUSH to PULL (*push-pull conversion*) whereas the Sampler component converts from PULL to PUSH (*pull-push conversion*). Refer to [Pust 06a] for more details.

If several data flow sources belong to the same physical sensor (as e.g., for several A.R.T. Tracker components), these source components belong to a common driver *module*

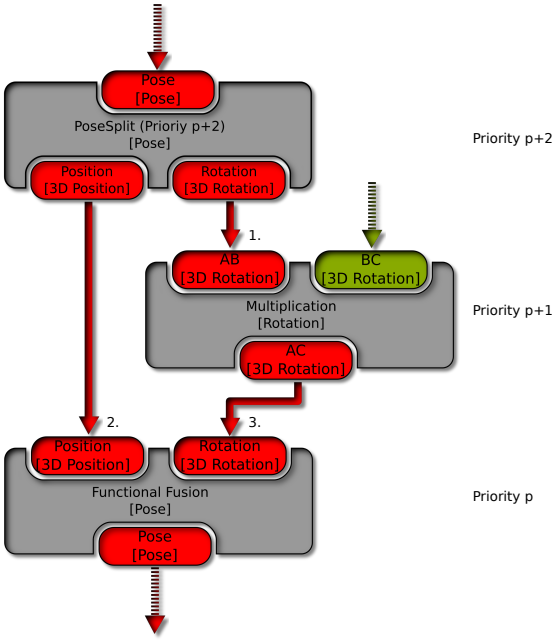


Figure 5.17.: Prioritization of PUSH operations in the data flow. Data flow components are sorted according to their "distance" to the sinks. The component priority is used to sort events with equal timestamps in the event queue.

and are thus in the same *trigger group*. This means that all PUSH events originating at these components obtain indentical timestamps. Therefore, no interpolation is necessary in the example DFG depicted in Figure 5.4. Unfortunately, there is currently no means to detect on the middleware layer whether two push sources belong to the same trigger group or not.

5.4.3. The Time-Synchronization Problem

The timestamps (see 5.1.2) associated with the individual measurements must represent the moment of the true measurement, otherwise the measurements cannot be associated properly. In general, this affects all kinds of fusion types. In complementary fusion, the joined degrees of freedom simply do not correspond. In competitive fusion, e.g., the motion model of a Kalman filter gives wrong predictions due to wrong computation of the time elapsed since the last measurement. In cooperative fusion, the needed interpolation (see 5.3.2) is also subject to the wrong computation of the weight between the involved supporting points. This simultaneity assumption has been discussed before by Welch [Welc 96], Welch and Bishop [Welc 97], and Pustka [Pust 11].

Synchronization can be accomplished in hardware, by using a common sync signal

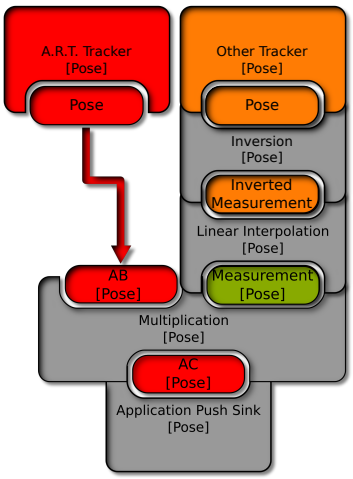


Figure 5.18.: A slightly modified version of the example DFG (cf. Figure 5.4). Two different trackers are used instead of just one tracker for two markers. A push-pull conversion is necessary to synchronize measurements from these two unsynchronized push sources. In a realistic scenario, the two trackers would probably also have to be registered to each other, which is not considered in the depicted data flow.

between the involved sensors[3]. Note that this is the case in the intelligent welding gun scenario where both, the static and the mobile tracking system, use the same sync signal.

If synchronization is not implemented in hardware, an inferior software solution has to be used. Latency in the communication between the involved devices makes an exact synchronization difficult. The most prominent reasons for latency are delay between the physical measurement and its digitization, overhead in network/bus communication, overhead in the implementation of the sensor drivers as well as the operating system. In addition to that, many platforms lack real-time capabilites such that an additional amount of time may be lost in process scheduling. These factors result in a typical delay for each sensor and for each platform. Therefore, timestamps which are generated in software, are not highly precise. Thereby, it does not matter whether the timestamps are generated upon arrival of the measurement or by prior soft-synchronization of the hardware clocks. Even if two measurements obtain the same timestamp, the corresponding physical measurements probably are not in sync.

A method for the relative temporal alignment of independent sensors has been developed by Huber et al. [Hube 09] and Schlegel [Schl 11]. The idea is to add a relative temporal offset to the initial timestamps of one sensor. For this, a temporal registration has to be performed before the spatial registration (according to 7.2), trying to maxi-

[3] Among others, commercial multi-camera IR tracking setups follow this approach, e.g., those from *A.R.T.* [ART 11].

mize the correlation of both signals by tuning the relative temporal offset. The approach principally works under the assumption of an only slowly changing relative time-delay. The authors also show that it is sometimes necessary to interleave the temporal and spatial registrations in an iterative manner to obtain optimal results.

In some special cases, however, synchronization is not necessary although sensor fusion is performed:

1. A single sensor can provide data to different tracking algorithms. From the *Ubitrack* point of view, the results of these algorithms can be treated using one of the fusion techniques described in 4.2.2. This leads to a locally synchronized subgraph of the DFG. For example, a USB webcam might be used for marker- and feature-based tracking. Although the true timestamp is unknown, both results still correspond to each other. Cf. the discussion about synchronized PUSH sources in 5.4.1.

2. The synchronization can be assured physically. This happens e.g., when a set of common reference points are measured twice by probing them one after another with two different measurement devices. Note that this is the standard case when using offline metrologic equipment.The method is often used to collect corresponding data for registration purposes and therefore also applies for our augmented airplane cabin scenario (see 1.3.2). It is however not very viable for real-timetracking.

To conclude, for the scenarios discussed in this thesis, time synchronization is not an issue.

5.5. SRG Design Activities

The construction of an SRG based on the available spatial relationship patterns (see 5.3) is an iterative process which consists of the following steps:

Description of the tracking environment All mobile and stationary sensors and all real and virtual objects are identified. Their known or tracked spatial relationships to one another are described. This activity mainly uses base patterns. It has to be executed manually to provide the domain knowledge to the *Ubitrack* system.

Deduction of indirect spatial relationships Full patterns are applied to suitable parts of an SRG to infer additional spatial relationships. This can be achieved either by an automatic pattern matching process or interactively by the IAR-engineer.

Definition of the runtime interface to the application On the basis of query patterns, application interface(s) are inserted into the SRG. This step has to be performed manually such that a defined API can be established. In particular, the application needs to know the IDs associated with the application patterns, as described in 5.3.2.

The definition of the tracking environment and of the runtime interface to the application require manual interaction. These steps result in the base SRG depicted in Figure 5.2(a). It still contains unresolved queries in terms of dashed edges. The deduction of indirect spatial relationships to resolve all unresolved input edges (cf. Figure 5.2(b)) can theoretically be automated, at least partially. In the following, the manual and semi-automatic techniques provided by the graphcial tool *trackman* for this purpose, are described.

6. Graphical Data Flow Modeling

The *trackman* graphical modeling tool for spatial relationship graphs provides a user interface to directly access both, the SRG and the DFG, in a *round-trip* engineering approach. It also provides interactive means to access all patterns that are known to the *Ubitrack* runtime system and to integrate them into the current configuration.

6.1. Graphical Layout

Figure 6.1 presents a screen dump of *trackman* showing the interactive construction of the SRG and DFG of Figures 5.2 and 5.4. The tree on the left shows excerpts of the list of all patterns, accessible both with respect to semantic and structural (layout) categories. Below, the property editor allows one to inspect and edit settings associated with the selected node, edge, or pattern. On the top left, a search facility allows the IAR-engineer to restrict the displayed elements in the tree to only those patterns that contain all specified strings.

The central area is tiled, showing the current SRG (top) and DFG (bottom). In the DFG pane on the right, data sources (corresponding to base patterns, green) are the uppermost components, followed by intermediate computational units (full patterns, cyan), and finally the lowermost data sinks (query patterns, yellow). The latter ones represent interfaces to applications. IAR-engineers can alter the tracking setup in the SRG window. Resulting updates are automatically brought to the DFG window. At intermediate stages of the configuration process, not all nodes in the DFG window need to be integrated into the data flow network. For example, the right green node in Figure 6.1 has not yet been connected to other modules. According to this, a part of the input section of the Multiplication pattern is not embedded in the SRG.

6.2. trackman Architecture

In order to keep *trackman* independent from *Ubitrack* development, and to ensure its compatibility with new spatial relationship patterns, it was designed as a lightweight and generic tool. Architecturally, it resides on the middleare layer, as depicted in Figure 5.1.

In general, *trackman* does not have special knowledge about the semantics of the patterns but rather imports the current set of available patterns from external description files that come with the *Ubitrack* runtime library[1]. They provide information about the signature or syntax of the patterns needed to enable the graphical modeling process. The

[1] Some minor exceptions arise in the context of registration and data flow surveillance tools as described in 7.2 and Chapter 14.

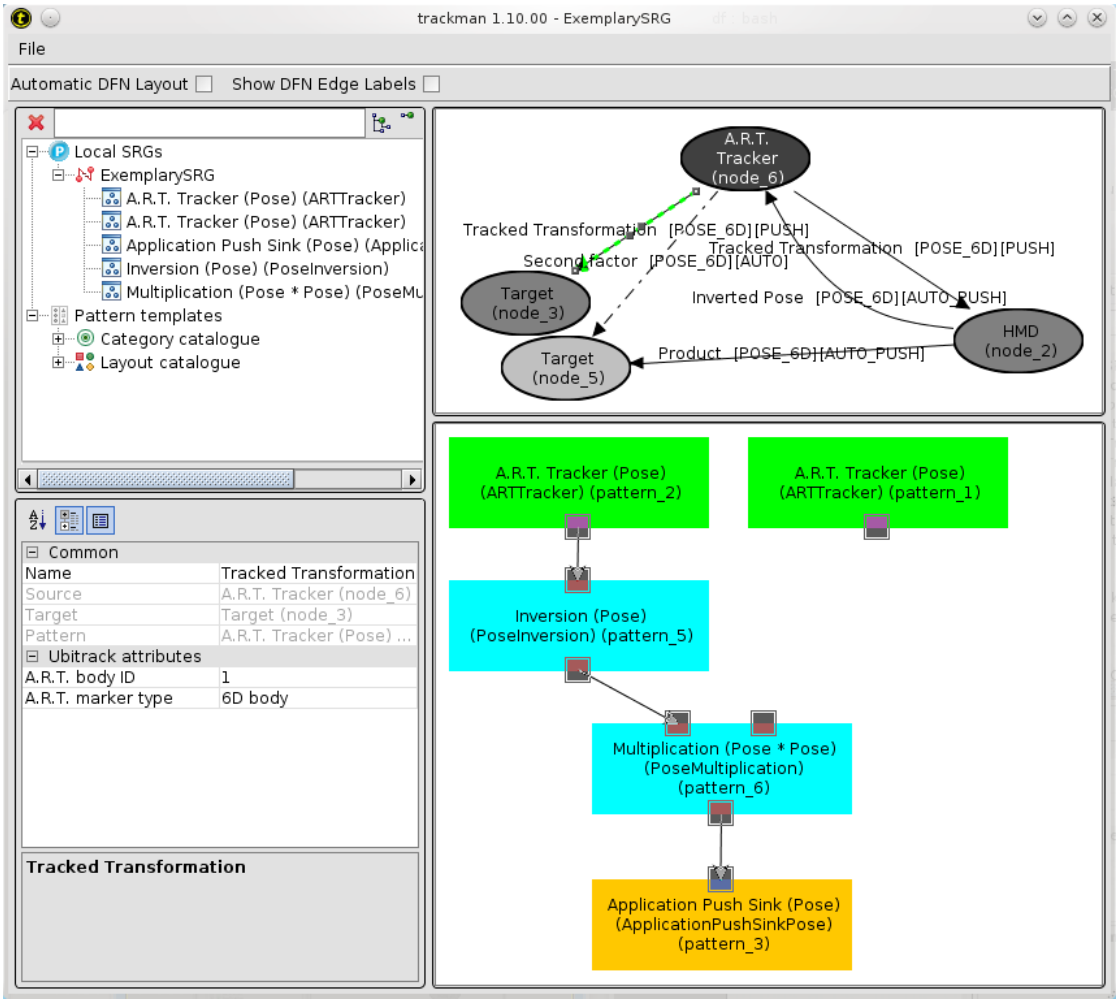


Figure 6.1.: The *trackman* graphical modeling tool for spatial relationship graphs. The example SRG from Figure 5.2 is shown on the editor pane. The resulting data flow is updated on-the-fly and is displayed on the right. Pattern templates are provided in the tree on the left, they can be dragged to the editor pane for modeling. The property editor on the bottom left is used to configure attributes provided by nodes, edges and patterns.

description language is based on the UTQL [UTQL 11] data exchange format [Pust 07]. In addition to the mere graph structure of the patterns, it also allows one to specify important meta information. Type information for node/edge/pattern attributes (see 5.3.3) as well as pattern documentation have to be provided. The resulting *pattern template specification language* XML schema [UTPa 11b] [UTPa 11a] allows for the formal description of available patterns. trackman uses the meta information to allow for convenient configuration of node, edge, and pattern attributes in its property editor and to display documentation to the user, as can be seen in Figure 6.1.

It shall be noted that in the current implementation of *trackman*, explicit space-expansion (cf. 5.3.5) is not directly supported. Rather, an extra hardcoded pattern template in XML has to be provided for each number of correspondences. Usually, however, the number of correspondences in explicit space-expansion is small, such that the resulting bloat of pattern templates is manageable. An example is given in 11.4.

6.3. Interactive SRG generation

Starting with an empty work area in *trackman*, we use base and query patterns similar to those shown in Figures 5.7(a) and 5.8(a) to describe the directly existing spatial relationships in the tracking environment and the application requests. To this end, they are dragged from the tree view on the left to the SRG editing workspace.

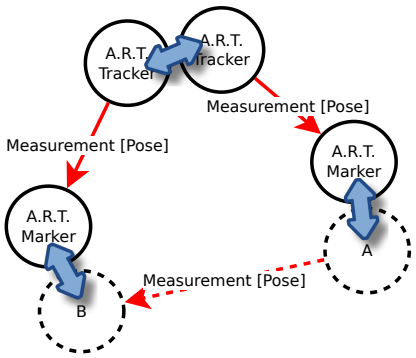


Figure 6.2.: Identification of coordinate systems via *node unification*. The indicated unification steps can be conducted in arbitrary order and result in the SRG shown in Figure 5.2(a).

For the SRG in Figure 5.2(a), the A.R.T. Tracker pattern is dragged twice into the work area, once for each of the two targets. Names, IDs, and other attributes are modified by selecting the respective node, edge, or pattern and applying the settings in the property editor. The query pattern Application Push Sink is dragged into the work space to describe the request that an edge be provided which describes the spatial relationship between the HMD and the target (see Figure 6.2).

Using the *node unification* interaction scheme, all three patterns can be merged to form a single graph. To express that two nodes belonging to different patterns are identical in the SRG, IAR-engineers can drag one node on top of the other one or select both before choosing the "unify nodes" operation from the context menu. As a result of this operation, the subgraphs are merged at this node. Node unification can be applied to all combinations of input and output nodes. Nodes belonging to a single pattern cannot be unified (principle of pattern atomicity). Figure 6.2 shows the node unification steps which lead to the SRG shown in Figure 5.2(a). See also 5.3.3 and 5.3.4 for the effects of node unification.

6.4. Interactive Deduction of Spatial Relationships

Another interaction scheme is needed to let IAR-engineers specify which operations shall be applied to the tracking data such that additional spatial relationships can be derived. To this end, full patterns have to be integrated into the SRG, thereby adding further (deduced) edges in terms of their output edges.

6.4.1. Edge Matching

By *edge matching*, an edge from the input section (dashed edge) of a new pattern is identified with an edge that already exists in the SRG and that is part of the output section of another pattern (solid edge). The operation also immediately updates the corresponding DFG, linking the input of the computational unit with the output of another component.

The edge matching operation is performed again either by dragging one of the edges on top of the other one or by selecting the two edges and then invoking the "match edge" operation from the context menu. Both edges must have the same edge characteristics in terms of data type and synchronization type (according to 5.1.2). Illegal matches are automatically inhibited.

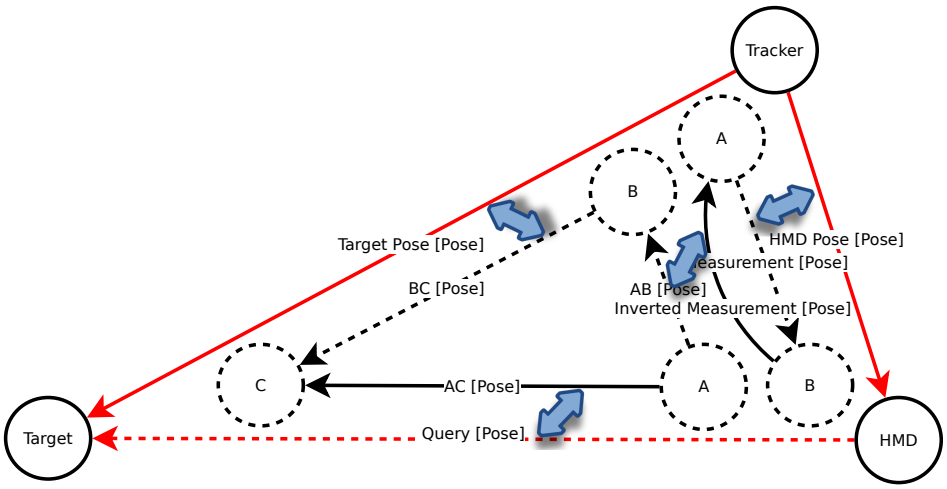


Figure 6.3.: Identification of data input and output of patterns via *edge matching*. The matching steps indicated by the blue arrows can be conducted in arbitrary order and result in the SRG shown in Figure 5.2(b).

Edge matching implies node unification according to 6.3 on the source and sink nodes, respectively, if necessary. Edges belonging to the same pattern cannot be matched (again due to pattern atomicity). The edge matching steps which lead to the SRG shown in Figure 5.2(b) are depicted in Figure 6.3.

6.4.2. Ordering of Design Activities

In 5.5, three elementary modeling tasks have been specified, the description of the tracking environment, the deduction of indirect spatial relationships, and the definition of the runtime interface to the application. It is up to the IAR-engineer to decide about a suitable design approach. Patterns may be added to the SRG in any sequence. Furthermore, patterns may be combined using the node unification and edge matching metaphors in any sequence. The output edge of a selected pattern may therefore be associated with a subsequent input edge, even though it is currently unclear how the output edge can be deduced since the input edges of that pattern haven't been matched yet. A valid DFG, of course, requires all input edges to have been matched properly.

The design process may therefore be started either with the environment or also the application interface. In general, two *modeling directions* can be distinguished.

Bottom-up approach The IAR-engineer starts with physical entities and refines information step-by-step, resulting in an application-level piece of information. This might also be denoted *induction*.

Top-down approach The IAR-engineer starts with the application interface and drills down through various algorithms to finally reach real-world sensors and objects. This might also be denoted *deduction*.

For clarification, going *up* according to the degree of abstraction from raw sensor measurements towards application-level data comes along with going *down* in the data flow from data sources to data sinks. In some cases, also come central full pattern serves as a seed for the modeling process and the corresponding real-world entities and the application interface is added to it in a mixed bottom-up / top-down approach. In general, *trackman* supports an arbitrary mixture of the bottom-up and top-down approach.

6.4.3. Round-Trip Engineering in SRG and DFG

The SRG editor on the top-right of Figure 6.1 is beneficial in the first place to argue about spatial relationships. The integration of spatial-relationship patterns in the SRG editor is intuitive and inhibits semantically wrong combination of patterns. Other issues such as the sequence of operations in the resulting data flow network (see 5.2) or the synchronization of the latter (see 5.4.2 are better observed in the DFG editor on the bottom-right.

Both visualizations represent the same internal model and are directly related. As explained in 5.2, an edge in the SRG relates to a port in the DFG, and edge-matching in the SRG relates to connecting ports in the DFG. Connecting two ports in the DFG can be accomplished by a click-and-drag operation from one port to the other. In a *round-trip engineering* approach, both views can be equally used to establish connections between patterns / data flow components. Changes effected in one of the views are always reflected directly in the other, and vice versa. Round-trip engineering is a common concept in software engineering. It keeps the consistency of the underlying model among

different representations. A typical example is the consistency of source code and the corresponding UML model in modern IDEs[2].

In case of modeling in the DFG, the semantic correctness of the corresponding SRG is always ensured and invalid operations result in an error message.

6.4.4. Automatic Sync Propagation

While the data type of an edge is determined in the pattern template and is immutable, the synchronization type of an edge belonging to the trigger group of a trigger component (see 5.4.1) is determined not until configuration time of the data flow. Directly after the pattern has been dragged to the SRG editor pane, the corresponding edge(s) of such patterns still have an undetermined synchronization type, as indicated by their black color during the construction of our example SRG in Figure 6.3. Note also the depictions of the Multiplication and Inversion spatial relationship patterns and data flow components in Figures 5.6 and 5.5. Finally, each edge must be associated with a distinct synchronization type, either PUSH or PULL. In our example, all SRG edges and their corresponding data flow ports obtain type PUSH, as indicated by the red color in Figures 5.2(b) and 5.4. Thereby, the rules for the synchronization of trigger groups stated in 5.4.1 must be obeyed. By repeated edge-matching operations, the yet undetermined synchronization flags are being constrained step by step.

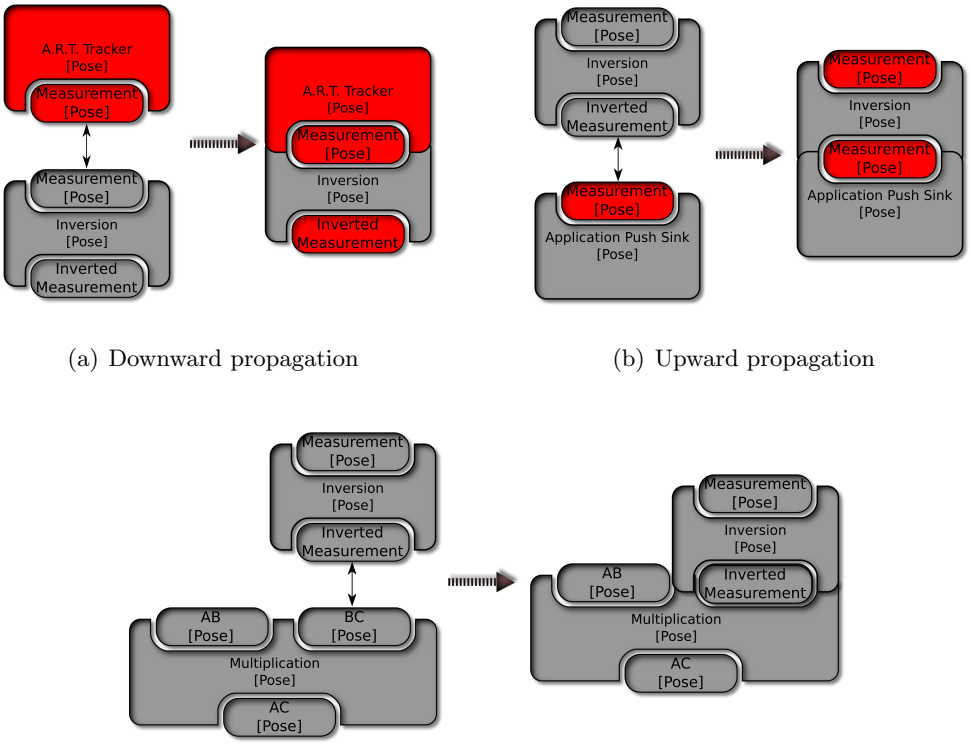
trackman supports the analysis of synchronization issues and ensures the consistency of the data flow synchronization at all times. During each edge-matching step, and in accordance with the modeling directions defined in 6.4.2, already fixed synchronization flags are propagated downwards (bottom-up modeling approach) or upwards (top-down modeling approach) in the DFG, according to the rules described in 5.4.2. Propagation thereby means to either imply an additional constraint to a formerly unconstrained port in the DFG, or to reject the matching operation because of an already existing contradiction. The propagation is trivial for components with single inputs and outputs, or if no constraint is set for propagation. These trivial cases are depicted in Figure 6.4. Upward-/downward propagation of PULL in the trivial cases 6.4(a) and 6.4(b) works analogously. For trigger patterns with multiple inputs[3], the situation is a bit more complex.

As Figure 6.5 shows, a PUSH has to be propagated downwards on the output as soon as *at least one* input is constrained to have type PUSH. In contrast, a PULL has to be propagated on the output only if *all* inputs are already constrained to have type PULL. In the upward direction, the situation is mirrored. Constraining the output to type PULL immediately implies that all inputs must also have type PULL. Constraining the output to type PUSH implies a constraint for an input only if it is the last unconstrained input and no other input already has type PUSH.

Input and output edges (data flow ports) with contradicting synchronization modes PUSH and PULL (or vice versa) may not be connected. Such a conflict has to be resolved manually, as in Figure 5.18. The Buffer (constant interpolation), Linear Interpolation, or

[2] Integrated Development Environment

[3] Note that trigger patterns were defined to have at most one output.



(a) Downward propagation

(b) Upward propagation

(c) No propagation

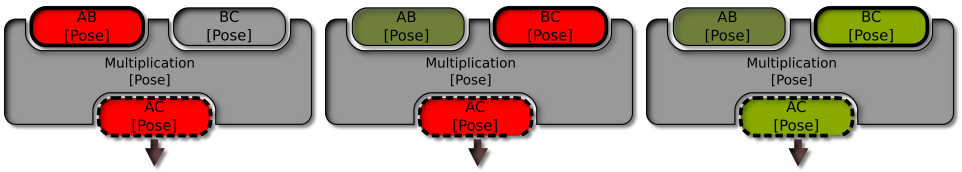
Figure 6.4.: Propagation of synchronization modes in trivial cases. (a,b) For trigger components having only one input, the mode is immediately propagated in both directions. (c) Nothing happens when connecting two yet unconstrained ports.

Kalman Filter components convert an edge with mode PUSH to an edge conveying the same quantity[4] but having type PULL. The opposite effect can be achieved using the Sampler component.

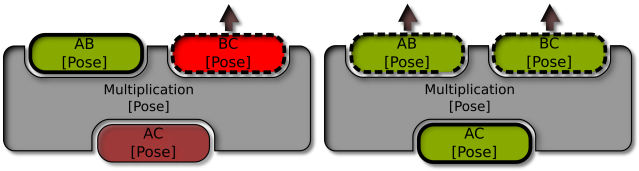
If multiple trigger components are combined according to Figure 6.4(c), initially without any constraints, the propagation of synchronization flags subsequently has to be performed recursively in the DFG, once constraints apply. Thereby, a downward-propagation may result in recursive upward-propagations and vice-versa. Figure 6.6 shows an example.

Ports can not only be connected, but also disconnected. As a consequence, the synchronization modes in the neighborhood of the released ports have to be updated, according to the removed constraint(s). For this, it is important to keep track for each port from which direction (upwards/downwards) potential constraints have been propagated

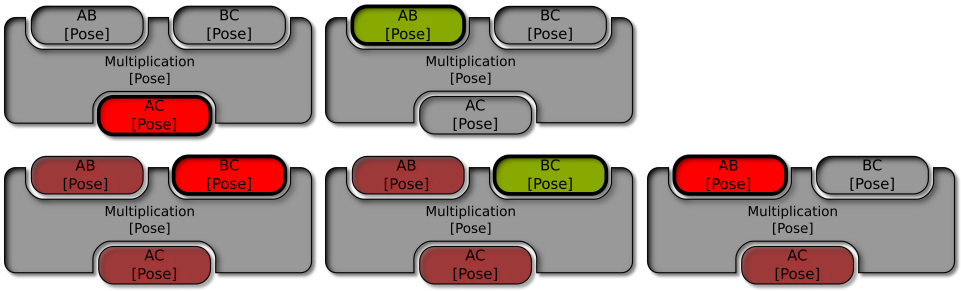
[4] though with other time and accuracy properties



(a) Downward propagation



(b) Upward propagation



(c) No propagation

Figure 6.5.: Propagation of synchronization modes in complex cases. Ports colored lightly indicate the initial condition before the additional constraint becomes effective. The port imposing the additional constraint is depicted with a thick, solid border. Ports that are affected by the additional constraint (and therefore change its synchronization type from AUTO to either PUSH or PULL, are depicted by a thick, dashed border, the potential recursive implication on other ports by an arrow.

to this port before. Internally, two synchronization flags are maintained for each edge/-port, one for upward and one for downward propagation. Both can either be undefined (AUTO) or have one of the values PUSH or PULL. However, they may never contradict each other; operations that would lead to a contradiction are not allowed (see above). The effective synchronization mode of the port is determined by the more distinctive one of the two flags. When a constraint from downward (upward) propagation ceases to exist, the flag for downward (upward) propagation is reset to AUTO; the new effective synchronization mode then depends on the value of the flag for upward (downward) propagation. If the latter is also undefined, the effective synchronization mode is also reset. Note that the effective synchronization mode is irrelevant for upwards and downwards sync propagation, only the other two flags are important. The effective mode is just used

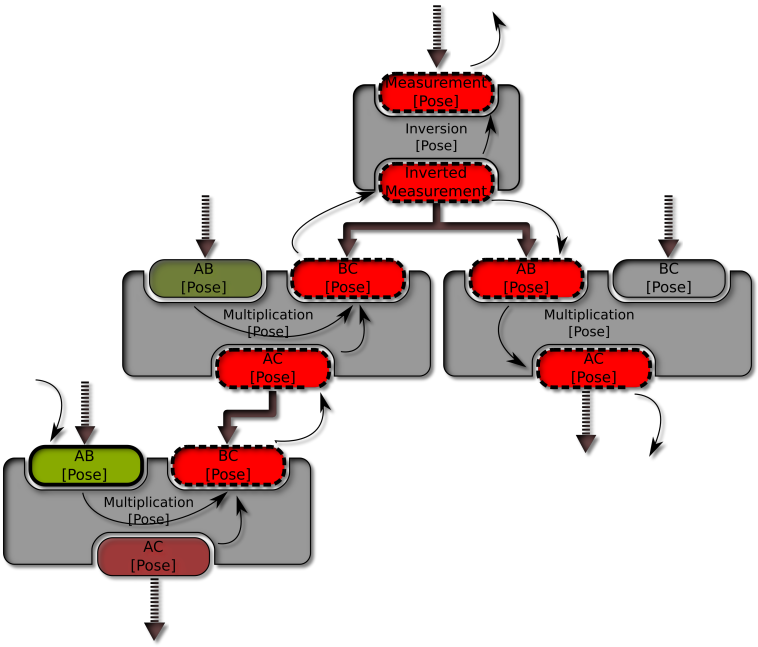


Figure 6.6.: Example for recursive upwards/downwards sync propagation. The synchronization of the pale ports is assumed to be already determined. An additional constraint (bold, solid border) leads to recursive implications for other ports (bold, dashed border). The recursion propagates upwards and downwards in the DFG, as indicated by the thin arrows.

to update the GUI. The downward (upward) recursion terminates whenever a trigger component is visited where the propagation does not result in further implications (cf. Figure 6.4(c)) or when a non-trigger component is visited.

Let us consider the directed graph G_p of synchronization dependencies with the individual data flow ports as its nodes, and edges between those ports that might affect each other's synchronization mode. G_p includes dependencies between input ports and their matching output ports in the DFG (inter-component port dependencies), though maybe in opposite direction, depending on the propagation direction. Furthermore, G_p includes dependencies between ports inside individual data flow components (intra-component port dependencies) according to Figures 6.4 and 6.5. A partial exemplary G_p is depicted in Figure 6.6 in terms of the data flow ports and the thin arrows between them.

G_p contains dependencies only between two auto-triggered ports. By assumption, as soon as at least one port with a fixed synchronization mode is involved, the match can be directly validated or rejected without prior recursion. However, the subsequently carried out recursive propagation may still reveal conflicts in other parts of G_p in case it is not a tree, as shown in Figure 6.7.

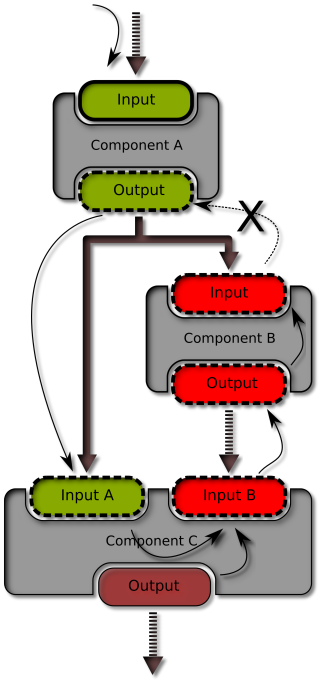


Figure 6.7.: Conflict during recursive sync propagation arising from a loop in the DFG. The Output of Component C is assumed to be already determined. The additional constraint implied on Component A reveals a contradiction on its Output port. The conflict is resolved automatically by releasing the connection between Component A and Component B.

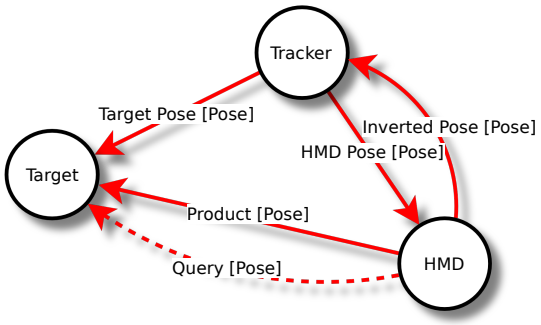
In most practical cases, however, G_p is a tree. In general, loops in the data flow are allowed by UTQL. In practice, they occur mostly in the context of data flow steering mechanisms as discussed in 11.2, though with fixed synchronization flags only. For an example, refer to Figure 11.3. None of the DFGs discussed in this document results in a cyclic G_p. In the rare case of a cylic G_p, special considerations are needed. The loop has to be detected during recursion and terminated, such that G_p can again be considered a tree. The problem is to justify the termination of recursion in such a loop. Two possible cases exist when a loop leads to a sync flag to be visited a second time: either the propagated sync flag is consistent with the set sync flag or not. In the former case, the recursion can trivially terminate here without further actions. The latter case is more problematic. The only way to terminate the recursion here is to release the contradictory match and to inform the user about this operation. She may then decide about measures to mitigate the contradiction, e.g., by manual insertion of a Linear Interpolation component and establishing the matching again afterwards. Of course, releasing the contradictory matching results in two additional recursive propagations of AUTO, one going upwards, starting at the released output port and one going downwards, starting at the released input port. This relaxation of the constraints, however, may not

lead to any additional contradictions.

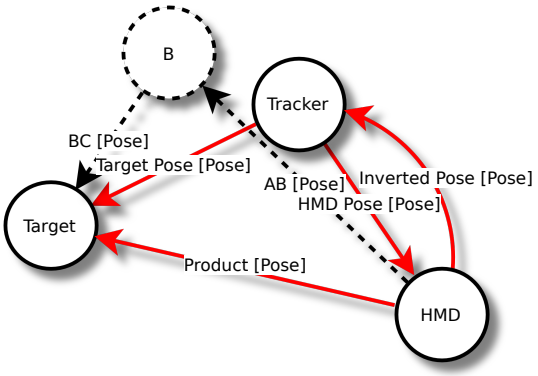
Consequently, G_p can always be considered to have a tree-like structure. This allows for efficient reevaluation, following the partial order imposed by G_p and thus supports interactive editing even for large DFGs.

6.5. trackman Editor Functionality

With node unification and edge matching, SRGs can be constructed from scratch. Additional functionality is needed when dealing with existing SRGs. This is important for the maintenance of existing setups and also to recover from modeling mistakes. Therefore, *trackman* also provides the following interaction schemes.



(a) Isolate pattern outputs



(b) Isolate pattern inputs

Figure 6.8.: Result of the isolate pattern outputs and inputs operations, invoked on the Multiplication pattern contained in Figure 5.2(b).

- Isolate Pattern Outputs: It is possible to separate the output edges of the pattern

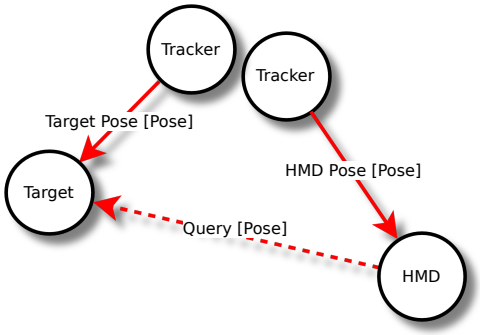


Figure 6.9.: Result of the split node operation on the A.R.T. Tracker node in the exemplary base SRG (cf. Figure 5.2(a))

from connected nodes/edges in the input section of other patterns. The effect of this operation is shown exemplarily for our example SRG (cf. Figure 5.2(b)). In Figure 6.8(a), the operation has been invoked on the Multiplication pattern; its single output edge has been isolated from all matched input edges, in this case only the edge named Query.

- Isolate Pattern Inputs: To complement the previous scheme, one can also separate the input edges of a pattern from its context, effectively annulling all dependencies between these input edges and corresponding output nodes/edges of other patterns. Invoking this operation on the same Multiplication pattern results in Figure 6.8(b). This implies also the separation of those input nodes that are neither source nor sink of an output edge of the pattern, such as node B of the Multiplication pattern.

- Isolate Pattern: This operation combines the two operations above and brings the pattern back to its atomic form.

- Delete Pattern: The pattern is removed from the current SRG. It does not matter whether the pattern was integrated in some larger SRG structure or existed in its atomic form. In the former case, an isolate pattern step is implicitly performed first.

- Hide Pattern: Parts of the SRG are hidden to provide an abstracted, clearer view of the SRG in the editor window. *trackman* provides this functionality on a per-pattern basis.

- Hide Edge: Individual edges of the SRG can be hidden by double-clicking them, to provide a clearer view on the SRG. Hidden edges are revealed temporarily by selecting either its source or sink node. Subsequently, they can be made visible permanently again by another double-click on the edge.

- Split node: This operation revokes the unification of nodes, though without breaking the graph structure and without undoing any edge-matchings. Applying this feature to the A.R.T. Tracker node in Figure 5.2(a) would result in two independent A.R.T. Tracker nodes, as depicted in Figure 6.9. However, it would have no effect on the same node in Figure 5.2(b), due to the constraints implied by the Multiplication pattern which cannot be reduced beyond its atomic form.

The Hide Pattern and Hide Edge features are crucial to maintain clarity in large SRGs such as the one shown in Figure 13.11 which consists of more than 70 patterns. Indeed most of the SRGs discussed in the remainder of this document show only the relevant information.

7. Common Modeling Tasks

Based on the concepts described so far, more complex solutions for realistic scenarios are now provided in terms of SRGs and the corrsponding DFGs. This comprises exemplary application data flows and also several common registration problems in IAR, where static spatial transformations between rigidly installed sensors, markers, and other objects are determined. Though registration problems actually have to be solved before the application data flow can be used, the sequence of their treatment is interchanged because the application data flows are easier to understand.

7.1. Application SRG

In the following, application data flow descriptions are derived for the two exemplary scenarios introduced in 1.3.2, the intelligent welding gun and the augmented airplane cabin. This also incorporates data flows that could not be modeled based on the "flat SRG" concept described by Pentenrieder [Pent 09].

7.1.1. Example: Indirect Tracking for the Intelligent Welding Gun

One of the currently most common tracking setups for AR and VR applications consists of an outside-in configuration with a number of infrared cameras mounted rigidly to the environment, observing a fixed volume within their midst. The camera arrangement imposes restrictions on the tracking of moveable objects inside/below/behind other opaque objects in the scene. This occlusion problem is not generally solvable by adding additional cameras to the classical outside-in setup since, first, occlusions generated by scene objects cannot always be known in advance, and second, the scene may offer only small and varying viewing angles to the outside, which cannot simply be covered by adding some more cameras. This is especially true for trackable objects surrounded by other objects, e.g., a tool inside a car body.

The *indirect tracking* approach already introduced in 1.3.2 adds an additional, mobile IR tracking system which can be placed in the scene on-the-fly such that it can see trackable objects that are hidden to the stationary cameras. The mobile setup itself is equipped with a marker such that its pose can be tracked by the stationary setup (see Figure 7.1).

The spatial relationships of the proposed tracking approach are depicted in Figure 7.2. The car body is mobile; it is tracked with respect to the Stationary Cameras (world coordinate system) using the A.R.T. Tracker pattern. The Welding Gun is tracked in the same way (direct tracking).

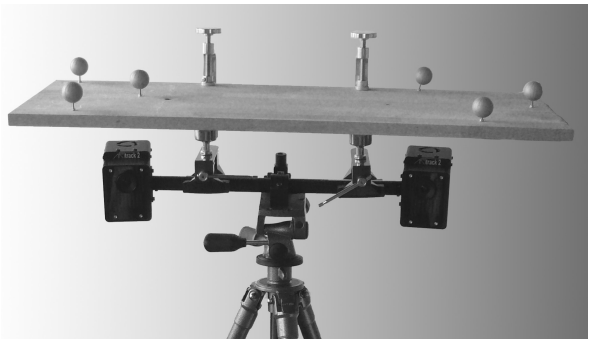


Figure 7.1.: Mobile tracking system with an attached marker consisting of six retroreflective fiducials

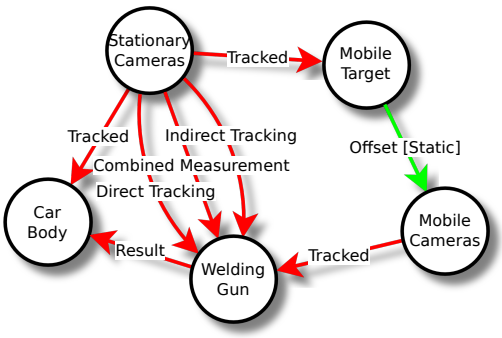


Figure 7.2.: SRG for indirect tracking of the intelligent welding gun. All edges represent pose transformations. Only the important spatial relationships are shown, most intermediary results are hidden for the sake of clarity.

The same transformation can also be estimated via indirect tracking. It is then then given by a concatenation of several transformations via the Mobile Target and the Mobile Cameras. The mobile target is again tracked by the A.R.T. Tracker pattern. The transformation between the Mobile Target the Mobile Cameras is static and it has to be registered before, e.g., using the method described in 7.2.2; it is incorporated via the Calibration Reader component. The quality of this registration is crucial, it is therefore further investiaged in 13.1.2. The last step in indirect tracking is the transformation of the Welding Gun with respect to the Mobile Cameras; it is again provided by the A.R.T. Tracker pattern. Of course, the direct transformation between the Stationary Cameras and the Welding Gun can still be used, when available. A Time Complementary Fusion component prioritizes the latter over the indirectly tracked transformation (Combined Measurement). It also provides a useful reference value for evaluating the quality of indirect tracking 14.2. The complete SRG and DFG can be found in Appendix A.1.

7.1.2. Example: Discrepancy Checks in the Airplane Cabin

The scenario of discrepancy checks in an airplane cabin has been introduced in 1.3.2. Due to the large volume to be covered, and the requirement for quick setup and dismantling procedures for the tracking system, a mobile IR tracking system mounted on a tripod is used that references itself using various reference markers mounted in the cabin. This is another variant of indirect tracking (see also 7.1.1).

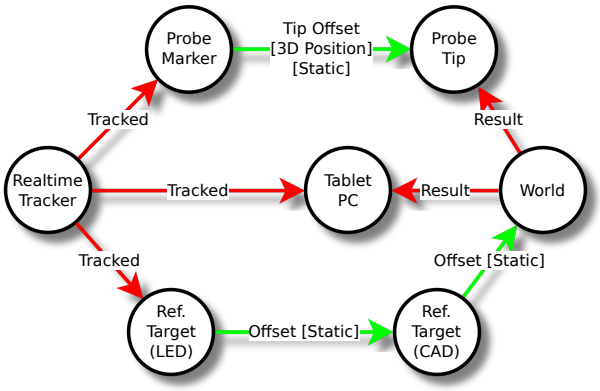


Figure 7.3.: SRG for discrepancy checks in an airplane cabin. Edges represent pose transformations unless otherwise noted. Only the important spatial relationships are shown, all intermediary results are hidden for the sake of clarity.

Figure 7.3 shows the spatial relationships between the involved entities. The world coordinate system is defined by the CAD model of the airplane cabin. The goal is to track in this coordinate system the position of the tip of a probe such as depicted in Figure 1.6(c) for quantitative discrepancy checks, as well as the pose of a Tablet PC depicted in Figure 1.11 for qualitative comparison. A mobile IR tracker[1] is used for real-time tracking.

The IR tracker references itself via reference markers that are installed in the scene. Only one such reference target is represented in the SRG. The Reference Target (CAD) node represents the coordinate system of the CAD model of the precisely manufactured target. Another node represents the distinct coordinate system spanned by the six IR LEDs mounted to the reference target, as it is used by the real-time tracking system. The transformations between World and Reference Target (CAD) as well as between Reference Target (CAD) and Reference Target (LED) are static. Their estimation is described below in 7.2. For now, it is assumed that they are already known such that they can be inserted into the SRG in terms of two Calibration Reader patterns. The other two solid edges are given in terms of NDI Optotrak patterns, one to track the Reference Target (LED) and another to track the Tablet.

[1] *NDI Optotrak Certus HD* [NDI 11]

The edges can be concatenated using the Inversion and Multiplication patterns to compute the desired transformations. Since all NDI Optotrak patterns belong to the same trigger group, no push-pull conversion is needed. The complete SRG and DFG can be found in Appendix A.1.

7.2. Calibration and Registration

Calibration typically denotes the estimation of parameters which describe the exact behavior of an individual physical device or object. From the Ubitrack point of view, the calibration of individual sensors or trackers is typically accomplished by the proprietary methods of the system vendors, which are beyond the scope of this thesis.

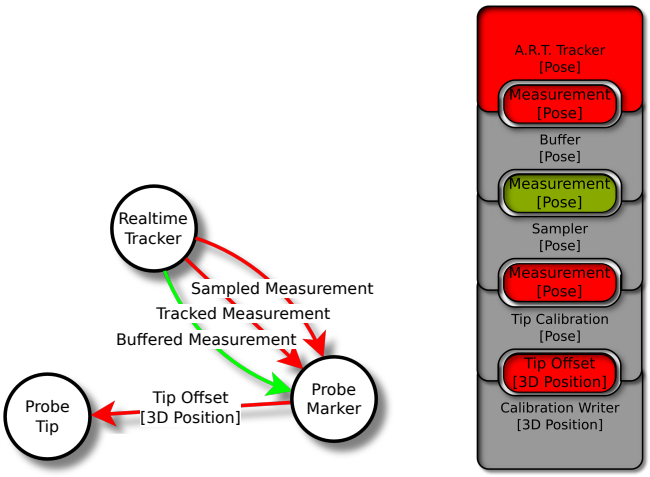
In 5.3.2, common registration patterns were introduced to determine static offsets *between* the coordinate systems represented by sensors, markers, or other objects. Confusingly, some of them carry the word "calibration" in their names instead of "registration", such as the Tip Calibration, Hand-Eye Calibration, and SPAAM Calibration, due to historical reasons. Others are called 2D-3D Pose Estimation or Absolute Orientation (also known as *3D-3D pose estimation*). These terms are maintainted when referring to the concrete algorithm. However, the term *registration* is used when referring to the estimation of static spatial relationships in the SRG in general, regardless the algorithm(s) that could be used to obtain a solution. Abstractly speaking, a node B is registered relative to another node A when the static edge from A to B is known.

7.2.1. Basic Solution Patterns

As a matter of principle, a registration data flow is constructed in the same way as any other tracking application data flow; it can be described fully by SRGs and DFGs. Generally speaking, in a typical registration process, the IAR-engineer has to move object(s) according to some rules depending on the chosen registration method while the object(s) is (are) tracked by sensor(s) in the environment. The tracking data of each sensor(s) is stored, and the desired static transformation can be determined from this data.

Some processes require IAR-engineers to align several objects, e.g., the 2D image displayed by the HMD with a given 3D coordinate for SPAAM Calibration or the tip of a probing device with drillings known in world coordinates. In such cases, the engineer has to signal when he is ready to take a measurement. When enough measurements have been collected, the desired registration can be computed. The minimum number of measurements depends on the chosen method, e.g., at least 3 corresponding 3D point measurements are needed in two coordinate systems to use the Absolute Orientation pattern. Normally, much more than the minimum number of measurements should be collected to reduce errors by a least-squares optimization process.

During the registration process tracking measurements are streamed (pushed) from one or more trackers into the DFN. They can be conditioned in three different ways before being passed to the actual registration algorithm:



(a) SRG (b) DFG

Figure 7.4.: SRG and DFG for tip calibration based on discrete measurements. The SRG shows all intermediary results.

Discrete measurements In the simplest case, all measurements provided by the tracker are being continually collected, at the discrete update rate of the tracker. They can either be directly fed into the time-expanded registration algorithm or stored using the Recorder pattern. If the update rate is too high, it can be sub-sampled. For this purpose, the collection mechanism is extended by adding the Buffer component[2] and Sampler components upstream in the data flow, as depicted in Figure 7.4(b). The desired frequency can be specified as an attribute of the Sampler component. The SRG and DFG are depicted in Figure 7.4. The SRG is just a bloated version of the Tip Calibration pattern, it does not reveal much more information. The DFG however presents the necessary sequence of operations quite intuitively.

User-triggered measurements Instead of sampling automatically at regular time intervals, measurements are taken upon user interaction. An asynchronous event injected into the data flow via the Application Push Source Button component triggers the Gate component to let pass a single measurement. This method fits well with registration methods that require the manual alignment of objects. Figure 7.5 demonstrates this concept by means of the absolute orientation problem. The SRG is omitted this time, it corresponds to the Absolute Orientation pattern depicted in Figure 5.9(a). A generalized approach for tracker/display calibration by user-triggered measurements has been presented by [Bail 03].

Some calibrations just need measurements from one tracking modality as input. The

[2] Using Linear Interpolation would degrade the quality of the data

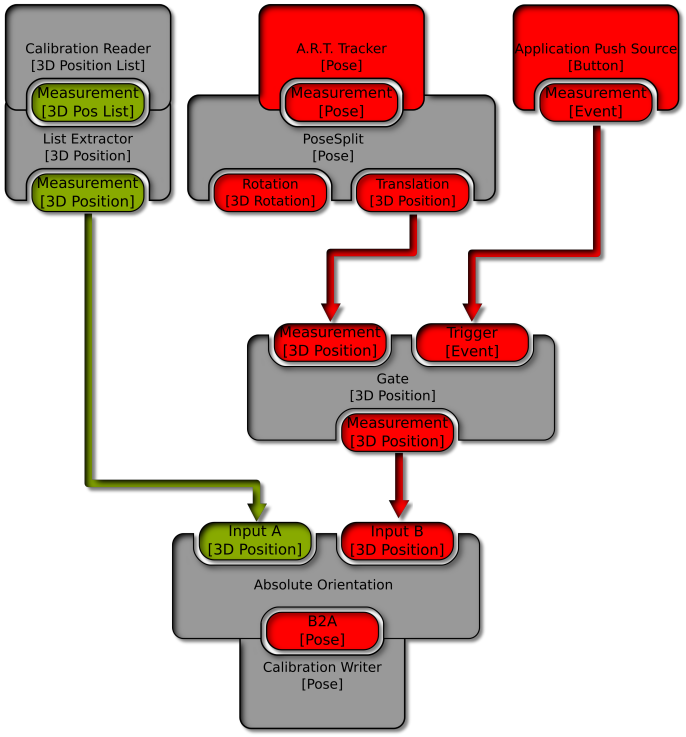


Figure 7.5.: DFG to solve the absolute orientation problem based on user-triggered measurements

tip calibration depicted in Figure 7.4 is one example. Registering a tracker coordinate frame with the world by probing a list of 3D coordinates on the object whose counterparts in world coordinates are already known (e.g., from the CAD model), is similar. Data can then flow directly in the DFN from the data collection components to the registration components. This allows for *online incremental parameter estimation* as soon as the minimal number of measurements are provided. If desired, more measurements can be taken in order to incrementally improve the registration until the residual error is considered to have converged against a reasonable value[3]. This is picked up again in 14.1.

In many calibration and registration processes, however, data streams of different trackers need to be associated with one another to obtain pairs of *corresponding measurements*. There are mainly two interaction methods for IAR-engineers to establish such correspondences between measurements from several trackers:

Simultaneous measurements Both measurements can then be fed directly into the time-expanded registration component in the DFG and one SRG is still sufficient to describe the entire procedure. An example is shown in Figure 7.6. The asynchronous timestamps of the two trackers have to be balanced, e.g., using the Linear Interpolation component, if the two trackers do not correspond to the same trigger

[3] under the ideal assumption of no outliers

group; otherwise, the DFG simplifies a bit. The negative effects of this interpolation, as well as means to mitigate them, are described in detail in [Pust 11]. Most importantly, the measurements with the higher update rate should be interpolated to minimize interpolation errors. Using this solution, we can still benefit from on-line incremental parameter estimation. It can be used with *discrete measurements* as well as with *glspluser-triggered measurement*. The DFG in Figure 7.6 uses *user-triggered measurements*. For *discrete measurements*, the combination of Buffer and Sampler shown in Figure 7.4(b) would have to be replicated for both inputs of the Hand-Eye Calibration component. This method is well-suited for the Hand-Eye Calibration pattern since the poses of two distinct entities Hand and Calibration Object can easily be tracked simultaneously, as depicted in Figure 5.9(b). Similarly, the measurements for SPAAM calibration (cf. Figure 5.9(d)) should also be collected simultaneously. On the contrary, to register two trackers with each other via Absolute Orientation, a single pointing device that can be tracked by both trackers, such as shown in Figure 1.6(b) or 1.6(c), would have to be moved around to generate 3D point correspondences; this works only if two compatible trackers are used.

Reproduction of measurements The idea is to conduct corresponding measurements (e.g., a set of 3D points in space) sequentially, using one tracking system then the other. Thus, a two-step approach for data-acquisition is needed, resulting in two data flow descriptions.
First, a list of *user-triggered measurements* using tracking modality A is aggregated via the Time-To-Space-Expansion Converter component and stored in a file via the Calibration Writer component. This represents a form of implicit space-expansion. Analogously to Figure 7.5 above, an Application Push Source Button and a Gate component are used for the acquisition of user-triggered measurements from modality A.
In the second data flow, the actual registration can take place, using both, measurements from the stored file via the Calibration Reader pattern and *user-triggered measurements* from tracking modality B, again analogously to Figure 7.5). The implicit space-expansion of measurements from modality A is revoked by the List Extractor component to obtain single time-expanded measurements again; they can be processed, together with time-expanded measurements from modality B, by the time-expanded version of the registration component. This method does not work well with the *discrete measurements* method described above since it is rather difficult to exactly reproduce the complete trajectory of an object, except if you have a robot at your disposal. Often, aids are installed in the environment to ease the reproduction of 3D point measurements. In metrology, adapters with standardized drillings are used for this purpose which allow for repeated probing using devices from different vendors having spherical tips with a common diameter (see for example Figure 1.6(c)).

7.2.2. Example: Hand-Eye Calibration for Indirect Tracking Setup

Two alternative methods can be used to estimate the static offset (green edge in Figure 7.2) between the Mobile Target and the Mobile Cameras. One solution to this problem is to use the Hand-Eye Calibration pattern (cf. Figure 5.9(b)), with the Stationary Cameras being the Robot, the Mobile Target being the Hand, and the Mobile Cameras being the Eye. A marker visible to the Mobile Cameras can be placed rigidly in the scene as the Calibration Object. The problem can hence be solved using the data flow description depited in Figure 7.6. However, moving around the mobile setup and rotating about various axes as needed by the algorithm, turned out to be difficult in practice, due to the weight of the marker board and the deflections of the rigid connection resulting from it.

The second alternative is a bit more elaborate. The Stationary Cameras and Mobile Cameras are placed rigidly in the scene. Then, the Absolute Orientation scheme with *simultaneous measurements* (see 7.2.1) is used to register both trackers with each other based on the measurement of corresponding 3D point features. Concatenating the resulting static transformation with the tracked transformation of the Mobile Target with respect to the Stationary Cameras yields the desired result.

7.2.3. Example: Registration of the Reference Target in the Airplane Cabin

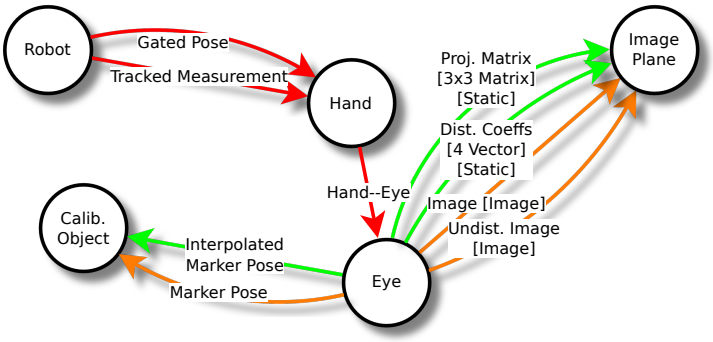
Each reference target is a plate precisely manufactured from a CAD model, with drillings that can be probed using a laser tracker metrologic measurement device as depicted in Figure 1.8 and 25 other drillings that can be probed with the IR tracker and a suited probing device such as shown in Figure 1.6(c). For, this, the 3D coordinates of all drillings that are known with high precision from the CAD model of the reference target are provided in two text files suitable for the Calibration Reader component. Measuring at least three drillings on the reference marker with the laser tracker allows computing its rigid transformation in the World (aircraft) according to Figure 7.5.

When looking at the SRG in Figure 7.3, the only missing registration is between the CAD model and the LED marker of the reference target. The LEDs have been attached manually to the plate and are therefore not known with high precision. A registration can be obtained by probing the 25 drillings known in the CAD model with the probe, again according to Figure 7.5. The rather large number of 25 points shall reduce registration errors from the rather inaccurate real-time tracking.

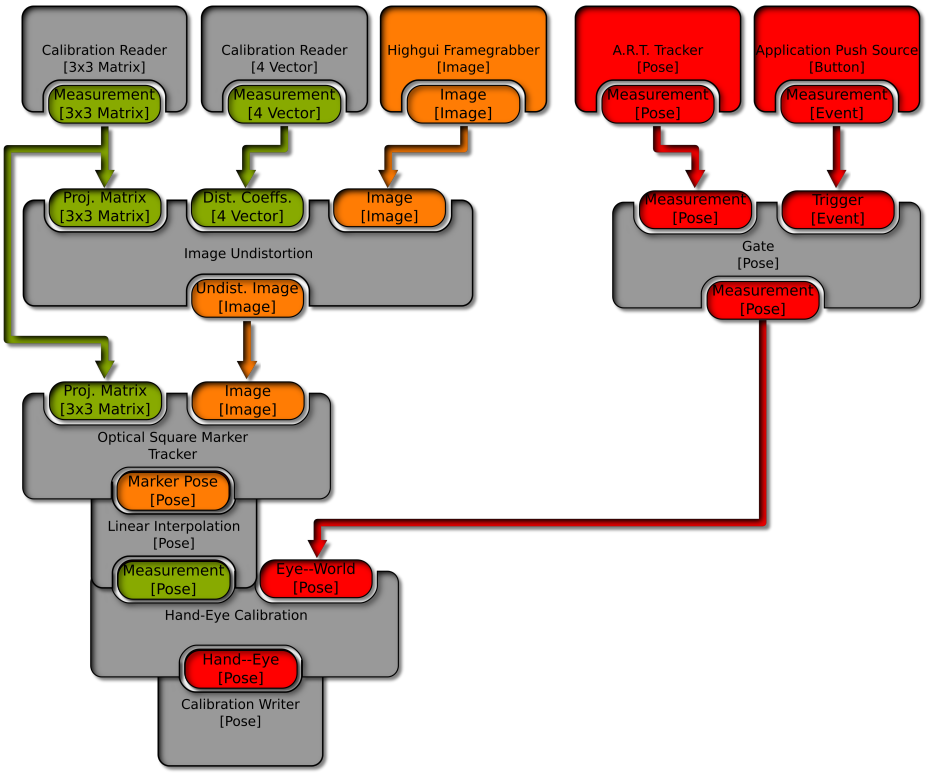
7.3. Application Interfaces

Typically, the data flow is driven by trackers that push their measurements into the data flow network. Therefore, the sink(s) of the data flow (resulting transformation) also have type PUSH if not explicitly changed by insertion of a push-pull conversion component. The default thus is to use the Application Push Sink Pose pattern to convey the information to the application. This and other patterns to model application interfaces hav already been listed in 5.3.2.

In some situations, it makes sense to use other means. In the intelligent welding gun scenario (see 7.1.1), a legacy application without indirect tracking was already available. It did not depend on *Ubitrack* but rather used the tracking data from the IR tracking system directly which are transmitted via UDP packets over the network. Therefore, a component called A.R.T. Sender was implemented which emulates the tracking system by sending UDP packets with the same format. The component serves as a drop-in replacement for the Application Push Sink Pose component. Therefore, no changes in the actual IAR application were needed for the migration of the tracking system to *Ubitrack* and indirect tracking.



(a) SRG



(b) DFG

Figure 7.6.: SRG and DFG to solve the hand-eye calibration problem using simultaneous user-triggered measurements. The SRG shows all intermediary results. Note that the event generated by the Application Push Source Button component does not have a representation in the SRG since it does not have a spatial interpretation.

8. Advanced Graphical Modeling Concepts

The implementation of the sample scenarios above reveals major potential for improvements in the graphical modeling process. This section describes two techniques that can further ease the SRG modeling process. *Semi-automatic modeling* automates simple operations and lets the user focus on the essential deduction steps. *Meta-patterns* provide best-practice solutions to well known problems, reducing the modeling problem to the addition of a few patterns only. Both techniques significantly reduce the number of modeling operations that have to be performed manually.

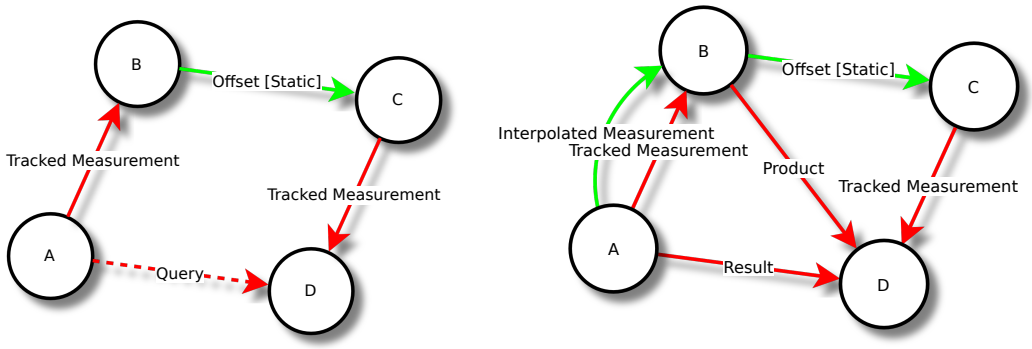
8.1. Semi-Automatic Modeling

Manual pattern matching can become a very tedious procedure. In more complex setups, the number of patterns to be integrated in the SRG increases quickly. A concatenation of n edges requires $n-1$ applications of the Multiplication pattern. In addition to that, some edges have to be inverted. In practice, approximately half of all matchings of full patterns fall upon the Inversion and Multiplication patterns (e.g., 5 out of 12 in Figure 7.2 and 6 out of 13 in Figure 7.3).

Automatic pattern matching can relieve the user from the trivial aspects of these and other modeling operations. Figure 8.1 depicts a typical modeling situation. The transitive transformation from A to D shall be deduced using known transformations from A to B, B to C, and C to D, respectively. Three full patterns are necessary to solve this simple problem and overall six solutions exist, one of them is shown in Figure 8.1(b). It first deduces a transitive transformation from B to D (Multiplication), then converts the synchronization mode of transformation A to B from push to pull (Linear Interpolation) and finally concatenates both to the desired result (Multiplication).

Fully automatic pattern matching is offered by the *Ubitrack* server. [Hube 07] [Pust 11]. In principle, any edge in the SRG can be deduced automatically, if there is a solution at all, by iteratively applying full patterns in a brute-force approach. Though the SRG concepts guarantee that the automatic pattern matcher only inducts semantically meaningful SRG edges, the approach has its limitations in selecting optimal patterns for every purpose. It is not easy to ensure that the chosen deduction steps meet the AR engineer's notion of the solution. The distributed DWARF tracking framework uses a path search strategy ondirected graph model to insert trivial operations such as inversion and multiplication automatically [Wagn 05]. This algorithm performes much faster than brute-force pattern matching. However, it works only for trivial operations.

In particular, the many push/pull variations may require fine-tuning by the engineer, once the overall setup has been configured. Assuming that the two tracked edges in Figure 8.1 offer comparable tracking quality at different frequencies, the location of the



(a) Deduction problem

(b) One possible solution

Figure 8.1.: Typical modeling situation which requires many applications of full patterns.

Interpolation component in the DFG influences the resulting quality and one would want to interpolate between measurements of the faster tracker. Differing tracking qualities between both trackers further complicates the consideration.

To exploit the best of both options, *trackman* provides *semi-automatic modeling* facilities. It uses the same pattern-matching algorithm as the *Ubitrack* server. During manual operations, IAR-engineers can enable automatic pattern matching for individual groups of patterns (e.g., for all variations of the Multiplication pattern or the Inversion pattern) while keeping other, more critical patterns such as the LinearInterpolation under strict manual control.

The automatic pattern matcher can be invoked in two ways. The first is to select the source and then the sink node of the transformation to be deduced and then to activate the matcher in the menu. The second method is to first integrate a query pattern into the SRG by unifying its source and sink nodes and then invoke the matcher on the corresponding dashed input edge.

Both, manual and automatic pattern matching have their advantages and drawbacks. This is reflected in the semi-automatic modeling approach, and also in the fact that *trackman* and the *Ubitrack* server use the same SRG and data flow description formats such that both concepts can be further interlocked on the middleware layer in the future.

8.2. Meta-Patterns

Another approach to simplify the modeling task is to provide reusable template solutions for common, recurring problems in terms of *meta-patterns*.

8.2.1. Definition

Basically, a meta pattern is an SRG consisting of more than one pattern that can be embedded in other SRGs for reuse. It contains only those patterns that belong to the reusable core of the solution. Interchangeable aspects can be left open. In particular, the meta-pattern may contain unmatched input edges (dashed). A meta-pattern can be embedded in an SRG like any other pattern. It differs from a normal pattern (cf. 5.3) in that it is non-atomtic and can always be dissected in its building blocks.

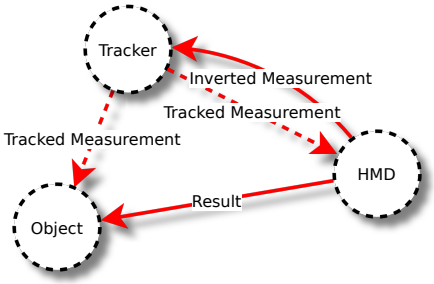


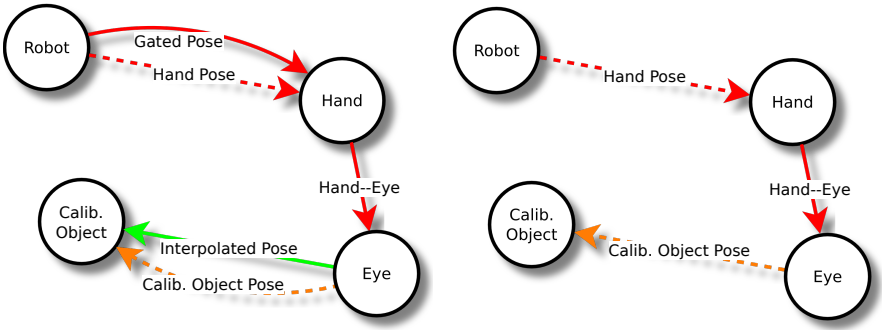
Figure 8.2.: Meta pattern describing the principal layout and application interface of the sample SRG in Figure 5.2(b).

To illustrate the idea behind meta patterns, Figure 8.2 shows the sample SRG from Figure 5.2(b) with all base patterns removed, i.e. nodes have a dashed rather than a solid line (see also Figure 6.8). This meta pattern still conveys the basic structure of the sample application, with HMD and target being tracked by a single tracking system, as well as the application interface. It can be completed by simply matching an arbitrary tracker pattern (providing a push measurement of type Pose) twice with the input edges of the meta pattern in order to obtain a valid data flow description again.

8.2.2. Applications

Registration problems are especially suitable for a description in terms of meta patterns. The estimation of a static transformation is generally constructed around a fundamental registration algorithm. Furthermore, a more or less constant set of additional patterns is needed which are responsible for steering the process of measurement acquisition. Figure 8.3 shows an exemplary meta pattern built around the Hand-Eye Calibration pattern according to the user-triggered simultaneous measurement acquisition scheme described in 7.2.1 (cf. Figure 7.6).

Only the Pose measurements of two trackers have to be added to complete the meta pattern to a full SRG (dashed edges). In this example, the measurements are collected simultaneously, manually upon user interaction (cf. 7.2.1). Furthermore, the Calibration Writer component has to be parameterized with a suitable path. The other solution patterns for registration problems provided in 7.2.1 can be packaged in meta-patterns in the same way.



(a) Fully visible (b) Interface only

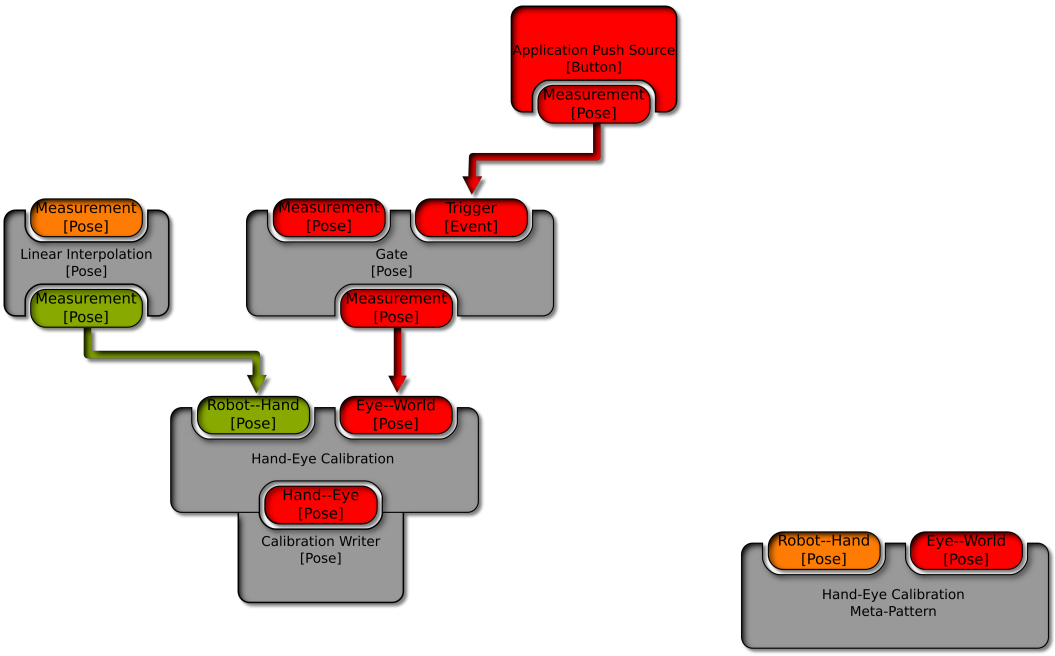
Figure 8.3.: Meta pattern for hand-eye calibration based on user-triggered simultaneous measurements. (a) With all patterns and edges visible. (b) With all except the interface hidden (cf. 6.5)

Meta patterns provide a quick solution for algorithm-centric problems such as registration procedures where complex data flow operations such as synchronization, aggregation and user interaction are integrated around a central algorithmic pattern. Particularly these complex data flow operations highly increased the complexity of modeling the data flow. They cannot be treated intuitively in the SRG (cf. 6.4.3), requiring the IAR-engineer to switch to the low-level DFG editor. Meta-patterns thus help to reduce the amount of expert knowledge needed to solve particular problems.

8.2.3. Integration in trackman

The current implementation of *trackman* does not yet handle meta patterns conceptually, i.e. there is no explicit category or treatment for them. Rather, meta patterns can be stored to or loaded from a UTQL file, like any other SRG. Also, they can be imported into existing SRGs. *trackman* does not have a notion of a meta pattern as a single entity; rather, the set of atomic patterns it consists of are handled individually. This also means that meta patterns are not supported in automatic pattern matching. At least, the edge and pattern hiding mechanisms (cf. 6.5) can be used to mask internal functionality of the meta pattern in the SRG editor and present to the user only its interface in terms of those input and output edges that are relevant for embedding the meta pattern, as demonstrated in Figures 8.3(a) and 8.3(b).

However, these internals cannot be hidden in the DFG view. Instead, it should be possible to treat the meta pattern as a single pattern in the GUI and to construct meta patterns containing other meta patterns, recursively. In the DFG view, the meta pattern could then be compacted to its interface, too, as depicted in Figures 8.4(a) and 8.4. This could simplify the DFG to a great extent. A *group* feature would be necessary to create ad-hoc meta patterns. An *ungroup* feature would allow one to "enter" the meta pattern again for later changes. Such a functionality is known from graphics and presentation software suites such as *Powerpoint*, to group primitive geometric shapes.



(a) DFG (b) Compacted component

Figure 8.4.: DFG of meta pattern for hand-eye calibration based on user-triggered simultaneous measurements. (a) With all data flow components visible. (b) Desired representation as a compacted data flow component revealing the interface of the meta pattern only.

It is also available in *Matlab Simulink* [Simu 11] where data flow components can be nested recursively (cf. 3).

This would allow for a consistent assembly/disassembly of the meta pattern; removing it from the SRG again would be a single operation only. The performance in *trackman* would increase, in particular the performance of automatic sync propagation (cf. 6.4.4). Last but not least, meta patterns could also be used in automatic pattern matching.

Remark: In case of doubt, new *Ubitrack* data flow components shall be designed as fine-grained as possible to allow for maximum flexibility. If functionality can be distributed to several components, this is the way to go. A bad example is the DirectShow Framegrabber component. Unlike the other framegrabbers in *Ubitrack* (see 5.3.2, it already contains the radial undistortion, although this functionality also exists as dedicated Radial Undistortion component. Although this results in DFGs containing more atomic data flow components, the complexity, and also the number of matching operations in *trackman* can be kept low using meta-patterns; the DFG also remains maintainable.

9. Discussion

9.1. Summary

Unlike other graphical data flow editors such as [Serr 08], *Matlab Simulink* [Simu 11], and *Spatial Analyzer* [Kine 11], *trackman* provides a two-layered graphical modeling approach. The SRG allows for an intuitive understanding of the spatial relationships and guarantees the semantic correctness of the resulting DFG. In a round-trip engineering approach, data flow oriented operations can be directly performed on the DFG, without loosing the semantic correctness property. IAR tracking scenarios with unforseen complexity could be realized using this tool. Pattern templates can be imported for modeling from their XML specifications, keeping *trackman* compatible with upcoming *Ubitrack* versions.

During an offline planning phase, the tracking problem at hand is described by iteratively inserting spatial relationship patterns (components) into an initially empty SRG (DFG) and combining them by node unification and edge matching (port connection) operations. The sequence of modeling steps is almost arbitrary, bottom-up and top-down modeling being the two border cases. Many pattern application steps can be automized which simplifies the design process a lot. In this semi-automatic modeling process, the IAR-engineer always keeps control of granularity and scope of automatically-made decisions. Furthermore, meta patterns provide default solutions to common problems, reducing especially calibration problems to a few mouse clicks. The result of the modeling process is a data flow description that can be used by an application, together with the *Ubitrack* library, to instantiate a tracking data flow. This enables us to change or modify an existing tracking environment very quickly, without the need to alter the AR application at all.

9.2. Advantages and Limitations of Graphical SRG Modeling

The SRG provides an intuitive representation of spatial relationships between different coordinate frames of trackers, markers as well as real and virtual objects. It lends itself to a documentation of the implemented concepts.

However, some non-spatial operations such as user-triggered events (cf. Figure 8.3(a)) or image undistortion (cf. Figure 7.6(a)) are not manageable intuitively in the SRG. It should rather be restricted to spatial transformations. Non-spatial operations also represent a big challenge in implementing the system. However, due to round-trip engineering, the SRG integrates well with the fully featured DFG and the SRG helps a lot to understand the otherwise unintuitive DFG. One could think about restricting such

steps to the DFG only to avoid having many barely distinguishable edges in the SRG. Alternatively, a hiding mechansim could be provided on an optional basis. Already, the problem can be mitigated to some degree using meta patterns and the edge hiding feature.

Even though many operations are better performed in the DFG than in the SRG, the latter still allows for consistency checks regarding the spatial correctness of each modeling step. However, procedural correctness is still difficult to ensure. For example, the synchronicity of two PUSH inputs served by two *Ubitrack* tracker components belonging to the same *Ubitrack* module cannot be verified, since the module concept is encapsulated inside the *Ubitrack* library. It should be made explicit in the pattern templates used by *trackman*.

9.3. Relationship between trackman and Dynamic SRG Modifications

In the context of IAR, the need for dynamic SRG modifications has been received low priority in 4.2.3 and 4.2.4. Nevertheless, *trackman* was designed as a generic tool. Therefore, the interoperability of *trackman* and the *Ubitrack* server has been kept in mind, even though it has not been pursued with a high priority. Both use the UTQL XML dialect to convey SRGs and DFGs.

Base SRGs created in *trackman* can already be sent to the *Ubitrack* server for integration into the world SRG. In the other direction, *trackman* can open the returned full SRGs, with some deductions in the graphical layout of the SRG and DFG. A continual surveillance of the world SRG incorporating dynamic changes on-the-fly could easily be implemented, if necessary. Furthermore, the pattern matching algorithm used to implement the semi-automatic modeling techniques described in 8.1 is identical to that the *Ubitrack* server is using.

9.4. Possible Improvements

trackman could be enhanced in many ways. Currently, each SRG is treated individually. As shown in Chapter 7, typically several SRGs are needed for a certain scenario, at least one describing the application itself, and various more for prior registration tasks. A project management facility would simplify the managemant of SRGs belonging together. In particular, paths to calibration files used by Calibration Writer and Calibration Reader components could be handled consistently. Furthermore, wizards could easily be provided to guide through the various registration steps in the correct order.

Wizards could also be provided for the solution of typical tracking scenarios, based on the already available meta patterns. They should furthermore point out to missing SRG edges to be provided as well as to mandatory or optional node/edge/pattern attributes to be specified.

Especially during development and testing of an SRG, it should be possible to deactivate individual patterns or meta patterns such that different variants of the data flow can

be tested without changing the SRG. This would especially help in cross-platform development where patterns representing sensor drivers, such as for example framegrabbers, sometimes have to be switched.

Patterns could be handled in an elaborate way. The pattern catalogue 5.3.2 lists more than 200 patterns. The existing categorization in terms of pattern syntax and semantics is quite helpful and the concept of trigger components (cf. 5.4.1) already constrains the length of the catalogue by providing a single component only for n^2 different sync configurations, n being the number of inputs in the trigger group. An intelligent handling of not only the synchronization type but also the data type would further constrain the combinatorial explosion. Currently, there exist for example eleven versions of the Multiplication component.

Regarding the GUI itself, several improvements lend itself to be implemented. Different colors should be used for SRG output edges, depending on whether they are static (base pattern), tracked (base pattern), or derived (full pattern). Additionally, output edges directly connected to a query pattern (data flow sink) should be highlighted. An example is shown in Figure 13.11.

Furthermore, an automatic hiding mechanism for edges and patterns would be helpful to ease clarity. Edges could be shown/hidden according to the above-mentioned characteristics also used for coloring. In addition to that, it would make sense to show/hide output edges according to whether any input edges match against them or not. By this, "used" edges could be hidden, accomodating the fact that many edges are needed exactly once and afterwards only clutter the representation . This happens whenever chains of data flow components occur in the DFG, as depicted in Figure 7.6(b).

In the DFG, a coloring scheme would be helpful to visualize data flow synchronization. Showing pushed edges in red and pulled edges in green[1] would make the DFG more intuitive than in the screenshot of *trackman* (cf. Figure 6.1). Consider Figure 11.3 as an example. It even distinguishes different sources of PUSH events.

An undo functionality would complement the existing set of operations. In fact, all matching operations in the SRG/DFG can already be undone by selecting the proper operation but a primitive undo is easier to use.

[1] As for most of the DFG graphics in this chapter

Part III.

Error Management

Tracking accuracy plays an important role in many AR applications, not only in the IAR scenarios described in 1.3.2. In this part, a generic approach for the management of tracking errors is formulated that is embedded in the methodology described in the previous part. It is based on verification by simulation paired with validation by empirical measurements.

The chosen approach is evaluated, mainly based on our two exemplary scenarios, the intelligent welding gun and the airplane cabin, as well as some additional preexaminations. Related concepts that are not directly applicable to our scenarios, are also integrated into the context. All relevant steps for error management are considered, ranging from the *elementary specification of sensor uncertaintys* via the tracking of individual markers and the registration of objects in the scene with respect to each other, to the overall IAR application.

The subsequent chapters describe integral components of this general approach. Chapter 10 starts with a review of statistical methods for the description and propagation of uncertainties. This covers also existing standards in the field of industrial metrology. Chapter 11 treats the integration of a Monte Carlo simulation framework into *Ubitrack*, as a generic means for the propagation of uncertainties through the data flow towards the spatial relationship that is relevant for the IAR application. This is based on an *elementary specification of uncertainties* that is assumed to be known for the involved sensors. Chapter 12 elaborates on how to provide this elementary specification of uncertainty. It shows how to determine a suitable degree of granularity to specify the basic sensor noise for a given system and how to assign a realistic quantity with it. In Chapter 13, the generic approach of verification and validation is finally applied to the main scenarios, the intelligent welding gun and the augmented airplane cabin application. Based on this, Chapter 14 demonstrates concepts for error mitigation during runtime. A wrap-up of concepts to deal with measurement uncertainties is presented in Chapter 15.

10. Quantifying Measurement Uncertainties

This chaper reviews state-of-the-art methodolgies that exist in the field of AR for the treatment of uncertainties. It starts with the statistical foundations for expressing uncertainties. Common standards in the field of accuracy assessment for spatial sensing devices are then presented. This is followed by a review of error propagation techniques which also incorporates its relations to the field of metrology and nonlinear least-squares parameter estimation. Finally, a generic verification and validation approach is formulated, based on extensive simulation paired with a restricted number of empirical measurements.

10.1. Representation and Categorization of Uncertainties

In this section, common error classification schemes are presented. Furthermore, basic error statistics are introduced that are used throughout the remainder of this part.

10.1.1. Categorization(s) of Errors

Azuma distinguishes between *static error* and *dynamic error* [Azum 97]. The latter is caused by various kinds of system delays and lags that are introduced during sensing, data procssing, and communication between subsystems. See [Pust 11] for the impact of dynamic errors that originate from within the tracking infrastructure. They impact the overlay accuracy (e.g., in the HMD) during movement of the viewpoint or tracked objects. Dynamic errors are not an issue as long as all objects remain still. For industrial IAR applications that provide quantitative measurements to the user, static errors are relatively more important and thus are investigated more thoroughly here.

Static errors are introduced mainly in the tracking subsystem, as well as in display calibration. For the quantitative measurements needed in our scenarios, the latter can be negelected since no AR overlays are diplayed. However, having an accurate real-time tracking system is not sufficient to eliminate static errors. The overall static error that is relevant for the application, is composed of various potentially independent sources: sensor calibration, offline metrologic measurements, registration of coordinate frames, and real-time tracking [Holl 95] [Holl 97].

Another often-cited classification distinguishes between *precision* and *accuracy* [Baue 06] [Baue 07] [Pent 09]. The term *precision* denotes the characteristics of the random distribution of samples around the sample mean whereas the term *accuracy* denotes a systematic deviation of the sample mean from the true value. In this model, it is assumed that the random error in a certain measurement results from many different influences which therefore can be modeled by a Gaussian distribution, according to the

central limit theorem. The systematic error, however, is caused by a few unknown and uncontrolled effects.

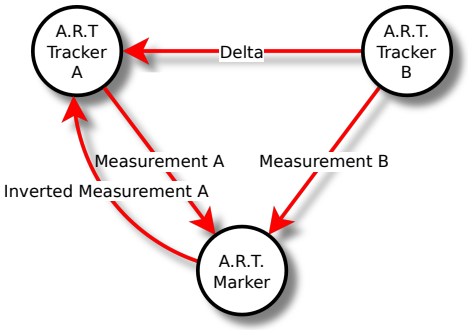
Azuma used the term *non-systematic* to characterize random effects [Azum 97]. Pentenrieder also makes this distinction in her work about AR based factory planning [Pent 09]. Ideally, systematic errors can be fully removed by appropriate calibration and registration procedures. An example is the undistortion of the magnetic field which is warped under the presence of metal objects [Zach 97]. In effect, due to dynamic scene changes (vibrations, dilatation due to temperature changes) and simplified physical models, systematic errors can never be fully eliminated.

Real-time tracking devices may provide many measurements at almost no cost, e.g., 50 poses of a marker when the camera pose is kept constant for one second and the camera provides 50 frames per second. It is questionable, however, whether such a simple repetition of the measurement under *repeatability conditions* yields much additional information in terms of a realistic sample distribution. Rather, the observed sample distribution is often rather small. To get better results, Niemeier and Luhmann suggest to repeat the experiment under *reproducibility conditions* [Niem 08] [Luhm 00a]. In the marker tracking example, this could mean to measure the poses of two markers instead of just one, and to observe the relative pose offset between the two markers instead. If the (unknown) true pose of two markers remains constant while the camera is moving to capture the scene from different perspectives, the relative pose varies. The experiment reveals an additional influence of error which depends systematically on the pose of the camera with respect to the rigid arrangement of markers. But does this experiment already reveal the full extent of systematic error that is present in the setup? Further influences might be investigated that were kept constant so far, such as the distance or the relative orientation between the two markers.

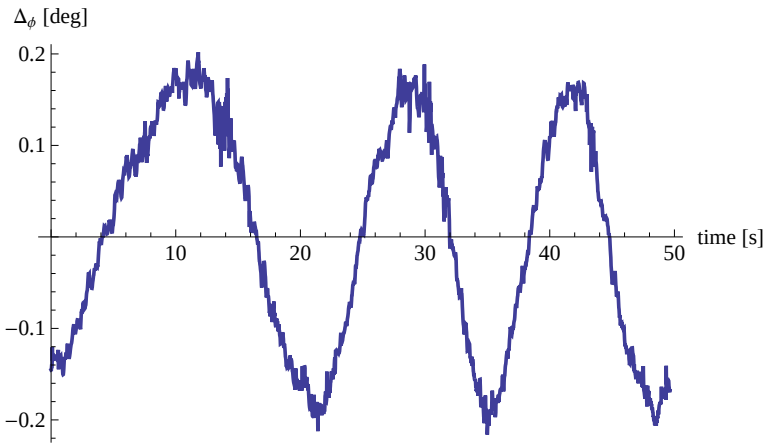
To demonstrate the importance of such systematic effects, we conducted a similar experiment. Two independent IR tracking setups observe the pose of the same marker which is lying on a table in the center of both tracking volumes. As Figure 10.1(a) shows, the relative offset between the two trackers can be deduced from these two pose measurements. Since the trackers are not moved during the experiment, the offset is supposed to remain constant as the target is being rotated thrice about an axis orthogonal to the table. Using Equation 10.21, the rotational part of the relative offset can be expressed in terms of a rotation axis and a single angle describing the rotation about this axis. Figure 10.1(b) depicts this angle, relative to its mean value. Instead of a constant value, a sinusoidal behavior is observed. This simple experiment already reveals a large amount of systematic error.

The question remains whether all systematic influences have already been revealed, or whether maybe more measurements under yet unexplorerd conditions (e.g., rotating the marker about another axis than orthogonal to the table), would maybe reveal even larger effects. To conclude, the assessment of all kinds of potential systematic influences quickly becomes expensive or even infeasible.

This simple example poses several important questions. How can we obtain comparable values from different kinds of measurements? Which canonical representation format shall be used to represent all the errors? Is there a simpler experiment which yields the



(a) SRG



(b) Deviation of Δ_ϕ from its mean

Figure 10.1.: Simple demonstration of systematic error. A 6DoF pose offset $\Delta_{6\text{DoF}}$ relates both tracking systems. A single, relative orientation angle Δ_ϕ is derived by converting the quaternion to the axis-angle represenation using Equation 10.21 and subtracting its mean value. One would expect Δ_ϕ to be independent of the pose of the marker. However, it shows a large sinusoidal oscillation, as the marker is rotated thrice on the table.

same results?

The *ISO* "Guide to the Expression of Uncertainty in Measurement" (GUM) [ISO 08] suggests a pragmatic approach for the treatment of errors. It introduces the broader, "operational" term *uncertainty*. It subsumes both, precision and accuracy and deliberately makes a distinction to the classical, "ideal" term *error*. The latter, by definition, can only be stated if the true value of the measurand is known, a requirement that can hardly be met. The following elementary terms and definitions are directly quoted from the GUM:

- *Accuracy* is the "closeness of the agreement between the result of a measurement and a true value of the measurand. [...] The term *precision* should not be used for

accuracy."

- *Repeatability* is the "closeness of the agreement between the results of successive measurements of the same measurand carried out under the same conditions of measurement"
- *Reproducibility* is the "closeness of the agreement between the results of measurements of the same measurand carried out under changed conditions of measurement"
- *Uncertainty* is a "parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could reasonably be attributed to the measurand"
- *Error* is the "result of a measurement minus a true value of the measurand".
- *Random error* is the "result of a measurement minus the mean that would result from an infinite number of measurements of the same measurand carried out under repeatability conditions". [...] "Random error is equal to error minus systematic error."
- *Systematic error* is the "mean that would result from an infinite number of measurements of the same measurand carried out under repeatability conditions minus a true value of the measurand". It is "equal to error minus random error". Furthermore, "like the true value, systematic error and its causes cannot be completely known".

The GUM distinguishes two methods for the evaluation of uncertainties. *Type A* is the "evaluation of uncertainty by the statistical analysis of series of observations". This comprises e.g., the a posteriori variances and covariances resulting from adjustment computation which is often performed in conjunction with non-linear optimization [Niem 08]. *Type B* is the "evaluation of uncertainty by means other than the statistical analysis of series of observations". This incorporates among others the usage of uncertainties specified by the vendor or computed from previous measurement data. Type B uncertainties can even be guessed, based on experience and general knowledge. This is reflected in the following quotation: "The proper use of the pool of available information for a Type B evaluation of standard uncertainty calls for insight based on experience and general knowledge, and is a skill that can be learned with practice. It should be recognized that a Type B evaluation of standard uncertainty can be as reliable as a Type A evaluation, especially in a measurement situation where a Type A evaluation is based on a comparatively small number of statistically independent observations." [ISO 08].

According to the GUM, an uncertainty value, be it expressed as a Type A or B value, shall incorporate not only random effects (precision) but also all those systematic effects (accuracy) that can be somehow assessed. This specification of one uncertainty value that comprises both effects has several advantages. Forward and backward error propagation techniques can be applied directly to derive the combined uncertainty of another

result in which the initial result is used. Confidence intervals can be directly computed. Last but not least, no explicit distinction is needed. This is of great importance, since the true value is normally unknown, so it is difficult to state systematic effects explicitly. For this pragmatic approach, particularly the Type B specification of uncertainties, the GUM has also been critized; nevertheless, it remains the state-of-the-art approach [Niem 08].

10.1.2. Basic Error Statistics

The derivation of the underlying error statistics would go beyond the scope of this thesis. Therefore, just the notations are introduced, and the reader is referred to the literature [Koch 97] [Niem 08] [Hart 00] [Baue 07] [Pent 09] [Pust 11].

Traditionally, error is specified as the deviation of a single sample x_i of a measurand X from the expected (true) value μ. The true value can be approximated by the arithmetic mean (average) computed over the n sample values x_i.

$$\mathbf{E}[X] = \mu \approx \bar{X} = \frac{\sum_{i=1}^{n} x_i}{n} \tag{10.1}$$

Deviations from this mean are typically characterized by the standard deviation σ, which is approximated by the empirical standard deviation s.

$$\mathbf{VAR}[X] = \sigma^2 \approx s^2 = \frac{\sum_{i=1}^{n}(x_i - \mu)^2}{n-1} \tag{10.2}$$

It represents the expected deviation of the sample from μ. For larger n, the empirical s approaches the unknown true standard deviation σ and the -1 can be omitted. Often, its squared value, the variance σ^2 is specified. For multi-dimensional measurements such as 3D position measurements, or 6DoF poses, the covariance matrix $\mathbf{\Sigma}$ replaces the variance σ. $\mathbf{\Sigma}$ is a positive semi-definite matrix in general. Its computation is described below in 11.

The distribution of the empirical sample mean $\bar{X}$ is again a random variable. If it is computed from uncorrelated X_i, it has the variance

$$\mathbf{VAR}[\bar{X}] = \mathrm{VAR}[\frac{\sum_{i=1}^{n} X_i}{n}] = \frac{\sum_{i=1}^{n} \mathrm{VAR}[X_i]}{n^2} = \frac{\sigma^2}{n}. \tag{10.3}$$

An important property of the variance is used here which states that the sum (or the difference) of uncorrelated random variables is the sum of their variances. As a consequence, the estimate of a quantity, e.g., an uncertainty, becomes better (decreasing variance), the more (uncorrelated) samples are incorporated and converges to 0.

Often, for random variables, a Gaussian distribution function can be assumed [Koch 97] [Niem 08] [Gelb 74]. In the general, multidimensional case, it is defined by

$$f(x) = \frac{1}{(2\pi)^{\frac{n}{2}}(\det\mathbf{\Sigma})^{\frac{1}{2}}} e^{-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T\mathbf{\Sigma}^{-1}(\boldsymbol{x}-\boldsymbol{\mu})}. \tag{10.4}$$

In some cases, uncertainties might be better represented by a uniform distribution than by a Gaussian distribution [ISO 08]. For this case, a conversion is necessary between the sample variance and the minimum and maximum devations that characterize the uniform distribution. The variance of a uniform distribution is given by

$$\sigma^2 = \frac{1}{12}(max - min)^2. \tag{10.5}$$

Assuming that $|min| = |max|$ and solving for $|max|$, we obtain

$$|max| = \sqrt{3}\sigma. \tag{10.6}$$

This allows one to represent the empirical standard deviation s to be used with a one-dimensional uniform distribution and is needed below to generate artificial measurement data for simulation purposes. The generation of general uniform distributions from a given covariance matrix $\mathbf{\Sigma}$ is non-trivial, since uniformity, unlike normality, is not preserved by linear transformations. A method for the generation of multi-dimensional correlated random variables according to a general covariance matrix has been described by Fackler [Fack 91].

10.1.3. Root-Mean-Square Error

The root-mean-square (RMS) error ϵ_{RMS} is a scalar, absolute error measure and represents the root of the mean of the squared errors. It is defined as follows [Baue 07] [Wile 05]:

$$\epsilon_{\text{RMS}} = \sqrt{\frac{\sum_{i=1}^{n} \epsilon_i^2}{n}} = \sqrt{\frac{\sum_{i=1}^{n} (x_i - \mu)^2}{n}} \tag{10.7}$$

Thus, the computation of the ϵ_{RMS} is based on the typically unknown ground truth value μ. As a replacement, the empirical mean $\bar{X}$ can be used. Another term for ϵ_{RMS} is *residual error*.

An interesting relationship can be established between the RMS error ϵ_{RMS} on the one hand and mean and standard deviation on the other hand [Wile 05]. Expanding Equation (10.2), one obtains

$$\begin{aligned}
\sigma^2 &= \frac{\sum_{i=1}^{n} (\epsilon_i^2 - 2\epsilon_i \mu + \mu^2)}{n-1} \\
&= (\frac{n}{n-1})(\frac{\sum_{i=1}^{n} \epsilon_i^2}{n} - 2\mu \frac{\sum_{i=1}^{n} \epsilon_i}{n} + \mu^2 \frac{\sum_{i=1}^{n} 1}{n}) \\
&= (\frac{n}{n-1})(\epsilon_{\text{RMS}}^2 - 2\mu\mu + \mu^2) \\
&= (\frac{n}{n-1})(\epsilon_{\text{RMS}}^2 - \mu^2)
\end{aligned} \tag{10.8}$$

Isolating ϵ_{RMS} in Equation (10.8) yields

$$\epsilon_{\text{RMS}} = \sqrt{(\frac{n-1}{n})\sigma^2 + \mu^2}, \tag{10.9}$$

and for large n

$$\epsilon_{\text{RMS}} \approx \sqrt{\sigma^2 + \mu^2}. \tag{10.10}$$

Thus, at least theoretically, the overall error can be seperated in its systematic (accuracy) and random (precision) components. However, this explicit separation is difficult to obtain and useless for practical purposes. Following the principles of the GUM [ISO 08], we therefore just estimate the overall ϵ_{RMS}, interpret it as the *uncertainty* and model it in terms of σ, distributing the deviations around 0.

Note that this is exactly what happens when we switch from repeatability conditions to reproducibility conditions: with each additional systematic effect μ_i that is "revealed", the overall empirical $\mu = \frac{\sum_{i=1}^{n} \mu_i^2}{n}$ approaches 0 since the systematic effects cancel each other out globally whereas σ increases. Fortunately, it is enough to conduct these experiments under reproducibility conditions to obtain a realistic σ, an explicit statement of the various systematic effects μ_i is typically not needed. They can be subsumed in the single quantity σ.

For multidimensional error distributions, this result can be extended [Kana 93] [Kana 96].

$$\epsilon_{\text{RMS}} = \sqrt{\text{tr}(\boldsymbol{\Sigma}) + \|\boldsymbol{\mu}\|^2}. \tag{10.11}$$

Here, $\text{tr}(\boldsymbol{\Sigma})$ represents the trace norm of the covariance matrix $\boldsymbol{\Sigma}$ and $\|\boldsymbol{\mu}\|$ is the Euclidean norm of the expected error $\boldsymbol{\mu}$. If we assume unbiased, independent, and isotropic errors, we obtain

$$\text{tr}(\boldsymbol{\Sigma}) = \sum_{i=1}^{N} \sigma_i^2 = \epsilon_{\text{RMS}}^2. \tag{10.12}$$

Thus,

$$\sigma_i^2 = \frac{1}{n} \epsilon_{\text{RMS}} \tag{10.13}$$

Equations (10.12) and (10.13) allow for the simple conversion between general covariance matrices and the scalar RMS value, based on the simplified assumption of unbiased, independent, and isotropic errors.

The RMS error is particulary useful for the specification of errors for 3D position measurements since its value then corresponds to the average Euclidean distance error. When matching point clouds from different coordinate frames, it is defined as follows [Wile 05]:

$$\epsilon_{\text{RMS}} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} \| \boldsymbol{H}\boldsymbol{p}_i - \boldsymbol{q}_i \|^2} = \sqrt{\text{tr}(\boldsymbol{\Sigma}_{\text{pos}}^{3\times3})} = \sqrt{\mu^2 + \sigma^2} \tag{10.14}$$

Thereby, $\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ are corresponding 3D position measurements in different coordinate frames P and Q, related by the 6DoF pose $\boldsymbol{H}$. The root of the mean of the squared Euclidean distances $\| \, . \, \|$ of the mapped point sets is equivalent to the root of the trace tr of the 3D covariance matrix $\boldsymbol{\Sigma}_{\text{pos}}^{3\times3}$. Assuming an isotropic error model, ϵ_{res} can also be expressed in terms of the expected deviation μ (systematic error) and its standard

deviation σ (random error). When observing several point clouds measured in the same coordinate frame, Equation (10.14) is slightly modified to obtain the RMS per point i

$$\epsilon_{\mathrm{RMS}_i} = \sqrt{\frac{1}{m}\sum_{j=1}^{m} \parallel \boldsymbol{p}_{i,j} - \bar{\boldsymbol{p}}_i \parallel^2} \tag{10.15}$$

where $\bar{\boldsymbol{p}}_i$ is the mean of all measurements j at position i. The overall RMS error then is

$$\epsilon_{\mathrm{RMS}} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\frac{1}{m}\sum_{j=1}^{m} \parallel \boldsymbol{p}_{i,j} - \bar{\boldsymbol{p}}_i \parallel^2)}. \tag{10.16}$$

10.1.4. Expressing Orientation and Pose Error

In this context, rotations are expressed in terms of unit quaternions [Kuip 02]. Altogether, a 6DoF pose transformation of a 3-vector $\boldsymbol{x}$ is represented as follows.

$$\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{t} + \boldsymbol{q}_r \boldsymbol{x} \boldsymbol{q}_r^*. \tag{10.17}$$

Thereby, $\boldsymbol{t}$ is a 3-vector describing the translation and $\boldsymbol{q}_r$ is a unit quaternion representing the rotation. The $*$ is the quatnerion conjugation. See also [Pust 04] for the representation of spatial transformations in *Ubitrack*.

A rotational error can be expressed in terms of an unnormalized "small" quaternion

$$\boldsymbol{q}_e \approx (e_{r,x}, e_{r,y}, e_{r,z}, 1)^T \tag{10.18}$$

that is multiplied with the actual rotation. This yields the following pose transformation including error:

$$\boldsymbol{x}_{\mathrm{new}} = \boldsymbol{t} + \boldsymbol{t}_e + \boldsymbol{q}_r \boldsymbol{q}_e \boldsymbol{x} \boldsymbol{q}_e^* \boldsymbol{q}_r^*. \tag{10.19}$$

The uncertainty of this transformation can be expressed in terms of a 6x6 covariance matrix $\boldsymbol{\Sigma}$.

$$(t_{e,x}, t_{e,y}, t_{e,z}, q_{e,x}, q_{e,y}, q_{e,z})^T \sim N(\boldsymbol{0}, \boldsymbol{\Sigma}) \tag{10.20}$$

A much simpler representation of rotational error can be obtained by converting q_e to the axis-angle representation

$$\begin{aligned} \phi &= 2\arccos(q_w) \\ a_x &= q_x / \sqrt{(1 - q_w^2)} \\ a_y &= q_y / \sqrt{(1 - q_w^2)} \\ a_z &= q_z / \sqrt{(1 - q_w^2)} \end{aligned} \tag{10.21}$$

where $\boldsymbol{a}$ is the associated axis and ϕ is the angle of rotation about this axis [Hart 00]. The conversion back to the quaternion is given by

$$\boldsymbol{q} = (\boldsymbol{a} sin(\frac{\phi}{2}), cos(\frac{\phi}{2}))^T \tag{10.22}$$

The axis associated with the erroneous orientation is often of secondary importance only. More interesting is the associated angle. It provides an intuitive scalar error measure, as can be seen in Figure 10.1.

10.2. Error Standards for Spatial Measurements

Several standards are available that describe in detail the assessment of uncertainties of metrologic measurement devices.

- ISO 10360 (Parts 1-13):Acceptance and re-verification tests for all kinds of coordinate measuring machines
- VDI/VDE 2634: Optical 3D measuring systems
  - Part 1: Imaging systems with point-by-point probing. This includes devices using optical (triangulation) and tactile probing.
  - Part 2: Imaging systems for surface probing
  - Part 3: Multiple view systems based on area scanning. This describes devices using triangulation from multiple viewpoints for 3D reconstruction. This part explicitly excludes those devices where the sensor is positioned using translational and rotational axes.
- VDI/VDE 2617, Parts 1-13: Accuracy of coordinate measuring machines—Parameters and their reverification. Accuracy assessment for all kinds of coordinate measurement machines. E.g., part 9 describes mechanical measurement arms.
- ASMEB89.4.22: Methods for Performance Evaluation of Articulated Arm Coordinates Measuring Machines. Similar to VDI/VDE 2617 (Part 9).

A brief overview of these standards is given in the following, focussing on the issues relevant for this thesis. An in-depth treatment has been provided by Chmill [Chmi 08]. Their primary goal of the mentioned standards is the standardization of

- acceptance tests, to be carried out by the system vendor, maybe at the customer's site.
- reverification test, to be performed at the customer's site
- between-process inspection and testing, is performed frequently with mechanic CMMs or laser trackers by returning to a *home* calibration sphere inside the measurement volume to confirm system consistency by confirming the reproducibility of point probing or the estimation of the diameter of special calibration spheres from various directions.

Depending on the type of device, the measurement prescriptions vary. Deviations exist mainly in the regulations for the placement and orientation of measurement devices, probes and reference objects, as well as the repetition of measurements under reproducibility conditions (cf. 10.1.1). The quantities to be measured mainly are

- probing error, that is the measurement of position, diameter, and shape (roundness) of individual rigidly installed reference spheres in the volume by touching them from different directions

- length measurement error, assessed using length standards and artefacts which can be measured by optical or tactile probing, like other typical workpieces, for different orientations of the artefact in the volume

Length measurements might seem unintuitive. Practically, however, they are often easier to obtain. When combining two sensor systems, e.g., in a cooperative fusion setup, the errors of both devices have to be considered when performing the registration (cf. 7.2) between them. Note that perfect length measurements in both modalities also yield perfectly matching point clouds in both modalities and therefore an error-free registration. This emphasizes the usefulness of the generic length measurements.

In VDI/VDE 2634 (Part 1), only length measurements are conducted explicitly. The probing errors are incorporated implicitly in these measurements since all measured lengths depend on the proper probing of the points defining the length. An example from this norm is shown in Figure 10.2.

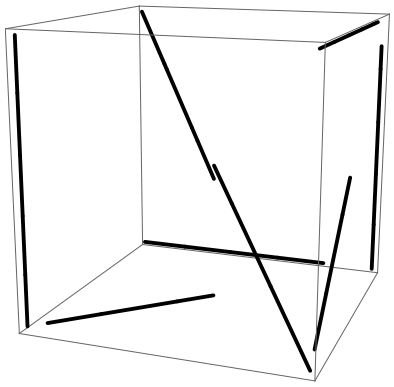


Figure 10.2.: Recommendation for distances to be measured in a given volume according to VDI/VDE 2634 (Part 1) [VDI 02]

For example, VDI/VDE 2617 (part 9) for measurement arms requires three spheres to be probed in the measurement volume at different distances (arm deflection lower than 30%, between 30% and 70%, and higher than 70%) from the arm's major axis. For reproducibility, these spheres have to be touched from five different directions. This ensures that different joint configurations are used and thereby facilitates the revelation of systematic errors. The ASME B89.4.22 goes a bit further than VDI/VDE 2617 (Part 9). It requires error statistics to be computed instead of a simple compliance with the maximum deviation values. It also provides guidlines for machine classification, environmental specifications (temperature, humidity, vibration, oscillation, external forces e.g., on device carrier), and machine performance in terms of comparison tests.

The described standards only cover classical measurement setups. Often, these systems are rather simple and their operating conditions are well-defined, facilitating a validation based on empirical measurements. However, even closely related devices, such as a mechanical measurement arm equipped with a laser scanner unit, are not covered.

To assess the accuracy of such systems, the existing standards have to be interpreted accordingly. The described uncertainties for point probing and length measurements generally keep their worthiness, only the testing modalities have to be adapted to the charachteristics of the new device. An example for such an adaption is given in [Chmi 08].

The complexity of the adaption thereby depends on the complexity of the sensor system. Measuring the distance between tactile points is rather straightforward for a mechanical measurement arm, not however for an IR tracking system consisting of severall cameras, because the performance now also depends on the camera setup (extrinsic camera parameters) as well as the number and distribution of fiducials on the probe and the tip offset. The degree of complexity is also increased drastically when multiple devices are incorporated in a single setup because then, also uncertainties from the registration of the devices with respect to each other have to be considered. This however cannot be accomplished in a general way by the system vendor any more. In 2.3.4 it has been mentioned that these registrations might have to be repeated frequently, depending on the definition of the production process.

As a consequence, acceptance tests for such systems tend to be performed according to the customer's requirements, at the customer's site, and no longer in a general fashion in the lab of the vendor. Some vendors resorted to publishing their own guidelines for the assessment and reverification of the accuracies of their sensors. For example, the guidelines published by *NDI* for their *Polaris* and *Vicra* position sensors [NDI 11] explicitly mention the German VDI/VDE 2617 and the American ASME B89.4.22 as its basis [Wile 05]. This approach however does not scale very well to the combination of devices from different vendors.

The increased effort for accuracy assessment can be mitigated by extensive use of simulation models such that the number of empirical measurements can be kept at an acceptable level, as shown in 11. This can be accomplished by simulation and error propagation models which do not only incorporate the uncertainties of the individual sensors, but also the geometric constellations they are used in.

In IAR, the challenge is to combine all uncertainty specifications in an overall application error. Offline metrologic devices still affect the registration accuracy which in turn burdens the real-time application with systematic error. Errors in the real-time tracking add to this error, resulting in a complex error behavior that cannot be easily described.

10.3. Propagation of Uncertainties

Whenever unknown parameters are estimated from known measurement data, it is desirable to propagate the uncertainties associated with the measurements to the estimated parameters. This allows for statements about the uncertainty that are meaningful for the intended IAR application. Various methods are available for the propagation of uncertainties. An overview is given in the following, together with related work in the field of AR. This is the basis for our propagation framework to be discussed in Chapter 11. First, however, a brief overview of parameter estimation is given, as parameter estimation and the propagation of uncertainties are inextricably connected, as we will

see.

10.3.1. Short Survey of Parameter Estimation

The field of adjustment theory generally seeks to solve problems of the type

$$f(\boldsymbol{P}) = \boldsymbol{L} \qquad \boldsymbol{P} \in \mathbf{R}^M,\ \boldsymbol{L} \in \mathbf{R}^N,\ M < N, \tag{10.23}$$

which is a mapping from some unknown values in parameter space $\mathbf{R}^M$ to measurements in measurement space $\mathbf{R}^N$. $\boldsymbol{P}$ denotes the unknown parameters and $\boldsymbol{L}$ the typically erroneous measurements that can be represented by a functional model f in terms of the parameters [Koch 97] [Niem 08] [Luhm 00a] [Hart 00].

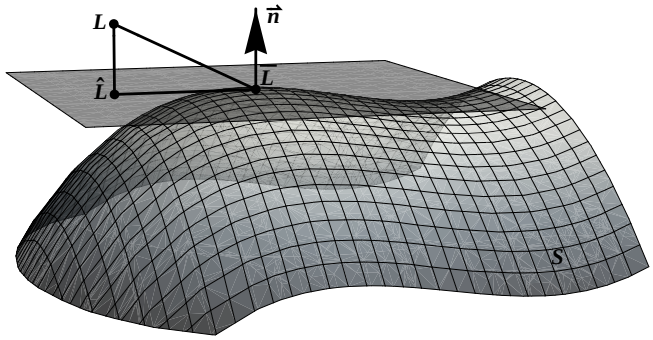


Figure 10.3.: Relationship between the functional model and the measurement space

Assuming (unknown) true values $\bar{\boldsymbol{L}}$, there exists a vector of parameters $\bar{\boldsymbol{P}}$ and the range of $f(\bar{\boldsymbol{P}})$ is a submanifold $\mathbf{S}$ of $\mathbf{R}^N$ with dimension M of all potentially valid measurement vectors $\boldsymbol{L}_i$. This is sketched in Figure 10.3. In the overdetermined case, due to errors, in general $\boldsymbol{L} \notin \mathbf{S}$. The maximum-likelihood estimate (MLE) $\hat{\boldsymbol{L}}$ of the unknown true values $\bar{\boldsymbol{L}}$ is the point on $\mathbf{S}$ (or its linear approximation as shown in Figure 10.3) closest to the erroneous measurements $\boldsymbol{L}$. $\|\hat{\boldsymbol{L}} - \boldsymbol{L}\|$ is the overall residual error. It can be computed when estimates $\hat{\boldsymbol{L}}$ are available. $\|\bar{\boldsymbol{L}} - \hat{\boldsymbol{L}}\|$ is the overall estimation error. It can only be computed for synthetic data for which $\bar{\boldsymbol{L}}$ is known. The following Pythagorean equality holds:

$$\|\boldsymbol{L} - \bar{\boldsymbol{L}}\|^2 = \|\boldsymbol{L} - \hat{\boldsymbol{L}}\|^2 + \|\bar{\boldsymbol{L}} - \hat{\boldsymbol{L}}\|^2 \tag{10.24}$$

MLE methods are typically applied whenever there is an overdetermination of the equation systems to be solved. Assuming a Gaussian error model, the estimation error in associated with the unknown parameters decreases with an increasing number of observations. In case of a linear functional relationship $\boldsymbol{L} = f(\boldsymbol{P})$, this is called linear least-squares (LLSQ). In case of non-linear functional relationships, equation systems are formulted by linearization of f using a Taylor series aborted after the linear part

and a solution is obtained by an iteration until convergence. This technique is also known as non-linear least-squares estimation (NLLSQ). A reasonable approximation of the solution must be provided for the iteration to converge to the true solution.

Bundle adjustment is a prominent example for the application of these concepts to the computer vision domain where problems typically are highly overdetermined [Doyl 64] [Luhm 00a] [Atki 96]. The techniques are integrated in many current computer vision algorithms, especially in real-time tracking techniques based on simultaneous localization and mapping (SLAM) [Bles 06]. [Klei 07].

NLLSQ is also useful for the implementation of calibration and registration algorithms that estimate some geometric properties or spatial relationships between different coordinate frames (cf. 7.2) from noisy measurement data. The uncertainties of the estimated parameters are obtained as a direct result of the optimization process. Furthermore, NLLSQ methods often yield superior results, as compared to the corresponding closed-form solution. Dornaika in 1998 for example compared different linear closed form and non-linear solutions for the hand-eye calibration problem (see 7.2.2). He obtained the best results with the NLLSQ solution, for uniform as well as for Gaussian noise [Dorn 98]. The result is also backed by Zhuang [Zhua 94]. This fact could however be outdated already by the findings of Daniilidis who described a linear close-form solution based on dual quaternions that gets along with one single solution stage [Dani 99]. Prior closed-form solutions always needed two stages to solve for rotation and then for translation, e.g [Tsai 88].

Our own experience in relation with the absolute orientation problem (cf. 13.2.1) does not yield any preferences. Still, NLLSQ has the advantage that it provides Type A uncertainties (cf. 10.1.1) out-of-the-box. Furthermore, it allows for statistical hypothesis tests to be implemented right away [Niem 08] [Koch 97]. This general overview is substantiated for the case of point-based registration in 12.3.

10.3.2. Propagation of Uncertainties in the Literature

There exist many prior works in the field of online and offline error estimation and propagation to understand uncertainties in tracking setups. Error propagation techniques have been applied successfully to assess the accuracy of individual tracking or metrological systems. Pentenrieder discusses the assessment of uncertainties for an optical square marker tracker [Pent 06]. This is achieved by automatic generation of artificial camera images. She also validates the simulation approach using high-precision ground truth measurements. Similarly, Hastedt uses a Monte Carlo approach to simulate the behavior of a photogrammetric system [Hast 04]. Similarly, Davis et al. predict the accuracy in pose estimation for marker-based tracking, resulting in suggestions for advantageous marker design [Davi 03][Davi 04]. In such propagation approaches, critical elementary influence factors can be determined which are not easily available for direct measurements.

Error propagation has also often been useful to assess the performance of a certain algorithm, e.g., [Dani 99] [Hart 00]. Furthermore, an important field of application for propagation techniques is to predict the overall system uncertainties that are relevant

for the application. One of the oldest works to predict pose error is by Woltring et al. [Wolt 85] who analytically derive the effects of isotropic 3D error on an isotropic distribution of fiducials and who observe that the error is minimal at the centroid of the fiducials. Holloway used maximum error statistics to derive uncertainties in a medical HMD application [Holl 97] [Holl 95] with an outside-in head tracking system. It incorporates static and dynamic errors according to Azuma [Azum 97], registration, tracking, visual overlay errors. Fitzpatrick [Fitz 98] give a formula for estimating the target registration error based on the simplifying assumption of an isotropic fiducial location error.

Allen et al. follow a general approach for the estimation of asymptotic or steady-state positional/orientational error at systematically chosen points throughout the tracking volume, incorporating also a motion model [Alle 05] [Alle 07]. The variances for raw sensor measurements are assumed to be known. Using 3D visualization techniques, the system gives insight into "variations in the expected performance throughout the desired working volume for a particular design choice, as well as the relative global effects of variations between candidate designs, independent of the tracking algorithm chosen for the real system". According to the authors, the approach could be adapted to "virtually anytracking or motion capture system". It aims at a lower level, optimizing the choice and constellation of sensors, as well as their sampling frequency, for the intended working volume and expected scene dynamics, irrespective of the chosen tracking algorithms and motion paths. Furthermore, it does not consider uncertainties arising from the alignment of static objects in space (registration, cf. 7.2).

There also exist several online error propagation approaches to predict the current tracking error of the application in real-time [Baue 06] [Fitz 98] [Hoff 00]. Those approaches simulate the error only for the current system state, which allows assessing the currently available accuracy. This is typically not possible for legacy systems since the internal error prediction is not accessible. Also a prediction throughout the entire error propagation chain requires linearization of each computational step. Additionally, for online capabilities the estimation often relies on a linear propagation model which might not be as precise but is fully sufficient to give the user an insight of the current system performance. Also, in IAR, the validation of a tracking system requires assessment of the overall system performance before deployment, that is what offline error estimation is used for.

Pentenrieder et al. describe an error propagation framework for industrial metrology based on photographs [Pent 09]. Coelho et al. [Coel 04] use the unscented transformation [Juli 04] to propagate tracking errors through a scene graph into the image shown to an AR user.

The online error propagation systems described so far either assume sensor errors to be known a priori [Hoff 00], or underestimate the total error by considering only jitter, thereby neglecting the systematic errors that are often even more important [Holl 95] [Baue 06]. Experiments with optical IR tracking systems reveal that the overall error largely consists of systematic error [Baue 06] [Keit 08].

10.3.3. Linear Propagation of Uncertainties

The GUM suggests to use a linear error propagation model to assess the uncertainties of a measurand $\boldsymbol{L} \in \mathbf{R}^N$ that is assessed indirectly by measuring a random vector $\boldsymbol{P} \in \mathbf{R}^M$, based on the the functional relationship $\boldsymbol{L} = f(\boldsymbol{P})$, see e.g., [ISO 08] [Niem 08]. Hoff used this technique for the estimation of head tracking accuracy in a medical application with a combination of optical outside-in and inside-out tracking systems [Hoff 00]. Bauer applied these concepts in the case of optical tracking where 2D uncertainties on the image plane are propagated to the uncertainty of the tip of a tracked instrument [Baue 06]. Pustka described their application in the context of *Ubitrack* in general [Pust 04] [Pust 11] and in particular in the context of IR fiducial marker tracking [Pust 10]. A review of the basic formulas is given next. A detailed derivation can be found in the literature, e.g., [ISO 08] [Koch 97] [Hart 00] [Niem 08].

Forward Propagation

The basis for linear error propagation is a linear or linearized functional model. If f is a non-linear function, a Taylor-series approximation can be used according to

$$\boldsymbol{L} = f(\boldsymbol{P}) \approx f(\bar{\boldsymbol{P}}) + \boldsymbol{J}_f(\boldsymbol{P} - \bar{\boldsymbol{P}}) \tag{10.25}$$

where $\bar{\boldsymbol{P}}$ is the expectation value of the parameters $\boldsymbol{P}$, and $\boldsymbol{J}_f$ is the partial derivative matrix (Jacobi matrix) of f at $\bar{\boldsymbol{P}}$. This linearization can be omitted, if the function is already linear. The mean and covariance matrix of $\boldsymbol{L}$ can be obtained from $\bar{\boldsymbol{P}}$ and the covariance matrix $\boldsymbol{\Sigma}_P$ of $\boldsymbol{P}$ by *forward propagation* according to

$$\begin{aligned} \bar{\boldsymbol{L}} &= f(\bar{\boldsymbol{P}}) \\ \boldsymbol{\Sigma}_L &= \boldsymbol{J}_f \boldsymbol{\Sigma}_P \boldsymbol{J}_f^T. \end{aligned} \tag{10.26}$$

A derivation can be found e.g., in [Koch 97] [Niem 08] [Hart 00]. From this elementary rule, concrete propagation formulas for all trivial data flow operations such as inversion or multiplication can be derived [Baue 07].

Backward Propagation

Often, the explicit observations are given in terms of $\boldsymbol{L}$ and the unkown parameters $\boldsymbol{P}$ of f are to be estimated. In this case, the covariance matrix $\boldsymbol{\Sigma}_L$ is given. By *backward-propagation*, the covariance $\boldsymbol{\Sigma}_P$ of $\boldsymbol{P}$ can be estimated according to

$$\boldsymbol{\Sigma}_P = (\boldsymbol{J}^T \boldsymbol{\Sigma}_L^{-1} \boldsymbol{J})^+, \tag{10.27}$$

where $+$ denotes the pseudo-inverse [Koch 97] [Niem 08] [Hart 00]. Bauer used backward propagation to estimate the uncertainty in 6DoF pose estimation from 2D image observations [Baue 06].

Relation to Adjustment Theory

Backward propagation of uncertainties is an integral part of all adjustment computations as described in 10.3.1. In the general case of a non-linear functional model f, the adjustment problem can only be solved by an iterative approach, relying on a repeated linearization of f at the last estimate of $\boldsymbol{P}_0$. The linear part of this Taylor series is often denoted as the *design matrix* $\boldsymbol{A}$, as a synonym for the Jacobi matrix. If $\boldsymbol{A}$ has full rank then $\boldsymbol{\Sigma}_P$ is positive definite [Koch 97] [Niem 08] [Hart 00].

Thereby, the assumed a priori covariance matrix $\boldsymbol{\Sigma}_L$ of the measurements $\boldsymbol{L}$ is transformed to an a posteriori covariance matrix incorporating the actual uncertainties of the observations. Figure 5.3 shows an example for the a posteriori covariances of a planar geodetic network from surveying a district; the overall covariance matrix has been split into covariances of the individual 2D points which are represented in terms of confindence ellipses [Niem 08] [Baue 06]. Although equal a priori covariances were assumed, the a posteriori covariances become larger in the boundary region of the network and smaller in its center. This is a natural consequence of least-squares optimization; it tries to balance the errors to keep the overall sum of squares small.

However, it is not sufficient to consider the uncertainties of the measurements only. For example, the covariance of a homography H estimated by some registration algorithm (cf. 7.2) depends on many additional factors [Fitz 98] [Baue 07]:

- The sensitivity of the chosen registration method: Different algorithms may have a different sensitivity with respect to measurements uncertainties or differ in their numerical stability.

- Number of correspondences: More correspondences improve the accuracy of the estimated homography. Imagine a homography that was estimated from the minimum number of correspondences. $\boldsymbol{\Sigma}_P$ would be 0 in this case, nevertheless the estimate probably is rather coarse.

- Spatial distribution of correspondences: The distribution of measured point correspondences should cover the intended working volume. Imagine a homography that was estimated from almost colinear points. Residual error is the same as for an equally distributed point cloud, but the estimated homography probabably contains large errors in the orientation around the axis formed by the measured points, due to the degenerate configuration.

The forward and backward error propagation techniques can be easily integrated into the (N)LLSQ parameter estimation techniques since the (linearized) functional relationship is already given and the computation requires the same intermediary steps. In adjustment theory, the terms *functional model* and *statistical model* are used to determine the (linearized) functional relationship and the input covariances that are needed to setup the adjustment computation.

Now, the elementary formulas for the NLLSQ estimation are stated, along with some background that is relevant in this context. For a complete description, please refer

to [Niem 08]. First, the functional model f is linearized around the initial solution $\boldsymbol{P}_0$:

$$f(\boldsymbol{P}_0 + \boldsymbol{p}) \approx f(\boldsymbol{P}_0) + \boldsymbol{A}\boldsymbol{p} \tag{10.28}$$

This yields the *shortened parameters* as well as the *shortened observations*

$$\begin{aligned} \boldsymbol{p} &= \boldsymbol{P} - \boldsymbol{P}_0 \\ \boldsymbol{l} &= \boldsymbol{L} - \boldsymbol{L}_0 = \boldsymbol{L} - f(\boldsymbol{P}_0). \end{aligned} \tag{10.29}$$

The constant part $f(\boldsymbol{P}_0)$ can be omitted from now on, only the linear part is important for parameter estimation and error propagation. The following problem is to be solved:

$$\hat{\boldsymbol{l}} = \boldsymbol{l} + \boldsymbol{v} = \boldsymbol{A}\hat{\boldsymbol{p}} \tag{10.30}$$

The stochastic model is given in terms of an a priori covariance matrix $\boldsymbol{\Sigma}_L$ as follows:

$$\boldsymbol{\Sigma}_L = \sigma_0^2 \boldsymbol{Q}_L \tag{10.31}$$

$\boldsymbol{Q}$ is called the *cofactor matrix*. The unknown variance factor σ_0^2 is estimated by the NLLSQ process. This partitioning represents the fact that only the relative weighting of elements of $\boldsymbol{\Sigma}_0$ is of importance for MLE estimation, the a priori σ_0 can be chosen arbitrarily; it can also be ignored.

The MLE estimate of the (incremental) parameters $\hat{\boldsymbol{p}}$ minimizes the sum of squared corrections that have to be applied to $\boldsymbol{L}$ such that the equation is balanced. These corrections represent the residual error in Figure 10.3.

$$\|\boldsymbol{L} - \hat{\boldsymbol{L}}\|_{Q_L}^2 = \boldsymbol{v}\boldsymbol{Q}_L^{-1}\boldsymbol{v} \rightarrow min. \tag{10.32}$$

Now, (10.30) can be solved using the so-called *normal equations* to find the (incremental) MLE estimate $\hat{\boldsymbol{p}}$ of the unknown (incremental) parameters $\boldsymbol{p}$.

$$\hat{\boldsymbol{p}} = \underbrace{(\boldsymbol{A}^T\boldsymbol{Q}_L^{-1}\boldsymbol{A})^{-1}\boldsymbol{A}^T\boldsymbol{Q}^{-1}}_{\boldsymbol{F}}\boldsymbol{l} \tag{10.33}$$

From this, also the cofactors $\boldsymbol{Q}_P$ of the estimated parameters can be derived, using the rule of forward error propagation (cf. 10.3.3),

$$\begin{aligned} \boldsymbol{p} &= \boldsymbol{F}\boldsymbol{l} \\ \boldsymbol{Q}_P &= \boldsymbol{F}\boldsymbol{Q}_L\boldsymbol{F}^T, \end{aligned} \tag{10.34}$$

plugging in (10.33), and simplifying the term.

$$\boldsymbol{Q}_P = (\boldsymbol{A}^T\boldsymbol{Q}_L\boldsymbol{A})^{-1} \tag{10.35}$$

By application of forward propagation to $\hat{\boldsymbol{l}} = A\hat{\boldsymbol{p}}$ stated in (10.33), the cofactors $\boldsymbol{Q}_{\hat{L}}$ of the corrected observations can be obtained.

$$\boldsymbol{Q}_{\hat{\boldsymbol{L}}} = \boldsymbol{A}\boldsymbol{Q}_P\boldsymbol{A}^T \tag{10.36}$$

A slightly more compelex derivation yields the cofactors $\boldsymbol{Q}_v$ of the corrections $\boldsymbol{v} = \boldsymbol{A}\hat{\boldsymbol{x}} - \boldsymbol{l}$. One has to plug in (10.33) for $\hat{\boldsymbol{p}}$, then use (10.35), factor out $\boldsymbol{l}$, and simplify the term.

$$\boldsymbol{Q}_v = \boldsymbol{Q}_L - \boldsymbol{A}\boldsymbol{Q}_P\boldsymbol{A}^T \iff \boldsymbol{Q}_L = \boldsymbol{Q}_v + \boldsymbol{Q}_{\hat{L}} \tag{10.37}$$

Thus, the cofactor matrix of the corrected observations $\boldsymbol{Q}_{\hat{L}}$ is smaller than the unknown true cofactor matrix of the original observations. This directly corresponds to the Pythagorean equality (10.24) stated above.

An estimate $\hat{\sigma}_0^2$ for the unknown variance factor σ_0^2 can be derived from the sum of squared corrections in (10.32).

$$\hat{\sigma}_0^2 = \frac{\boldsymbol{v}\boldsymbol{Q}_L^{-1}\boldsymbol{v}}{n-u} \tag{10.38}$$

Here, n is the number of observations, i.e. the dimension of $\boldsymbol{L}$ and u is the degrees of freedom of the problem, i.e. the dimension of $\boldsymbol{P}$. This estimate is correct only if the functional and stochastic model assumed above were correct. With this a posteriori estimate of the variance factor σ_0^2, an a posteriori covariance can be derived using (10.31). In an *estimation of variance components*, it is even possible to estimate multiple variance factors related with a partionned covariance matrix [Niem 08].

It shall be mentioned that the covariances that result from (non)-linear least-squares optimization in the overdetermined case represent Type A uncertainties in the sense of the GUM. This is concretized in the context of point-based registration below 12.3, including also an example of the above-mentioned formulas.

Another advantage of the concepts of adjustment theory is the straightforward implementation of *statistical hypothesis tests* [Niem 08] [Luhm 00a] [Koch 97] [Atki 96]. Global hypothesis tests are available to evaluate the consistency of the chosen functional and statistical model with the empirical measurements. Other tests evaluate the adherence to given constraints such as the identity points, collinearity of several points, right angles, ... Last but not least, statistical tests guess the adherence of the individual measurements with the estimated parameters. They can be used for a semi-automatic detection of outliers. Unfortunately, hypothesis tests are beyond the scope of this thesis.

Linear Propagation of Uncertainties in Ubitrack

Linear propagation of uncertainties has the advantage to be quite efficient, since only matrix operations are needed. This renders it quite suitable for real-time processing. Bauer used it to assess the accuracy of a multi-camera IR tracking setup, based on the assumption of 2D Gaussian noise on the image planes of the involved cameras [Baue 06]. In this context, he also gave a review of propagation rules for the trivial data flow operations such as Inversion or the Multiplication of one or two erroneous measurements [Baue 07]. The concepts have been incorporated into the *Ubitrack* tracking framework conceptually [Pust 04]. The transformation types Error Pose and 3D Error Position (cf. 5.1.2) represent spatial transformations with their covariance matrix attached. These data types are also supported by the elementary base patterns (e.g., Calibration Reader, Static

Transformation), the mentioned trivial operations plus Interpolation as well as the important query patterns (e.g., Application Sink, Calibration Writer, Recorder).

Also, the Optical Square Marker Tracker pattern provides an Error Pose on its output, which is obtained by backward-propagation from the assumed 2D uncertainties of the four corner points for a square marker [Pust 11]. Furthermore, generic Kalman Filter patterns are provided, e.g., for the fusion of several generic Error Poses [Pust 06b] or of an Error Pose with rotational information from a gyroscope, e.g., to improve 6DoF tracking under the influence of fast head movements [Pust 08].

However, linear error propagation always requires an explicit implementation in terms of a linearization of the underlying functional model of the component. Therefore, propagation support still lacks in many components, in particular for the essential registration algorithms (cf. 7.2), as already mentioned above. Suitable approaches are available in the literature, e.g., [Zhua 94] for the hand-eye calibration.

10.3.4. Monte Carlo Simulation

The Monte Carlo technique is generally attributed to Metropolis who used it for the analysis of differential equations in 1949 [Mete 49]. According to Dimov, "Monte Carlo methods are methods of approximation of the solution to problems of computational mathematics, by using random processes for each such problem, with the parameters of the process equal to the solution of the process. The method can guarantee that the error of Monte Carlo approximation is smaller than a given value with a certain probability" [Dimo 05]. And "usually Monte Carlo methods reduce the problem to the approximate calculation of mathematical expectations" [Dimo 05]. A Monte Carlo algorithm gives more precise results the longer it is run[1].

Monte Carlo simulation represents one of two major fields of Monte Carlo methods where random variables are investigated following the underlying "physical, chemical, or biological processes under consideration" [Dimo 05]. This is also called *Monte Carlo integration* and should be distinguished from Monte Carlo numerical algorithms which solve deterministic problems based on an artificial random process modeled e.g., as a Markov chain. A prominent example is the *Metropolis-Hastings* algorithm and the *simulated annealing* method derived from it [Metr 53] [Kirk 83].

Applied to the problem at hand, Monte Carlo simulation means to work on given spatial transformations as ground truth data (parameters of the problem) to which artificial noise is added. This perturbed data is then fed into the *Ubitrack* data flow (physical model) to propagate the assumed uncertainties to the desired spatial transformation (solution of the process).

The Monte Carlo method is a good alternative for the propagation of uncertainties. Based on the generally non-linear functional relationship $\boldsymbol{L} = f(\boldsymbol{P})$, the parameters $\boldsymbol{P}$ are sampled many times from any probability distribution and the behavior of $\boldsymbol{L}$ is observed by forward propagation. Alternatively, backward propagation yields the behavior of $\boldsymbol{P}$ if measurements $\boldsymbol{L}$ with associated uncertainties are provided. Requirement is the

[1] as opposed to a Las Vegas algorithm that gives the right answer, but whose run-time is indeterminate

existence of some means to evaluate f^{-1}, be it an analytic inversion of f or an NLLSQ method (cf. 10.3.3).

There is no requirement to use Gaussian probability density functions. It does not require a linearization of the functional model and is therefore always applicable, even for problems where an analytic propagation is difficult to achieve. This simple mechanism makes Monte Carlo a good choice for many purposes. Often however, it is not suitable for real-time processing since the computational complexity increases exponentially with the dimensionality of the parameter vector $\boldsymbol{P}$.

The *unscented transform* is also sometimes used for error propagation. It has similarities with the Monte Carlo method. However, the number of samples is decreased drastically by assuming again a Gaussian distribution. This allows one to observe the behavior of a few so-called *sigma points* in the parameter space only, instead of hundreds or thousands of samples. A detailed description of this method can be found in [Juli 04]. A review of related work is provided by [Baue 07]. The unscented transform is generally considered to be superior to the linear propagation of covariance that is implemented in *Ubitrack*, due to better handling of non-linearities. It has been shown, however, that the Monte Carlo method yields similar results to the linear error propagation in the context of fiducial marker tracking [Pust 10].

A Monte Carlo simulation framework for *Ubitrack* is described in 11. It is shown that it solves all practical issues related to IAR.

10.3.5. Elementary Specification of Uncertainty

Regardless of whether an analytic propagation model (cf. 10.3.3) or simulation (cf. 10.3.4) is used, the uncertainties that go into the simulation have to be specified. For this, the following questions have to be answered:

Abstraction Level The data can be given at different levels of abstraction, in particular as 2D or 3D positions or also as 6DoF poses. The term *abstraction* thereby represents the fact that typically, higher-dimensional information is computed from lower-dimensional data, e.g., 3D positions from 2D positions by triangulation or 6DoF poses from 3D positions by absolute orientation (cf. 7.2). The decision on the granularity highly depends on the used system setup. If the layout and the number of fiducials allowed for marker design are predefined, it could make sense to specify a 6DoF pose error. Similarly, in an optical multi-camera tracking setup with arbitrary camera arrangement, rather a 2D value is specified since the uncertainty of a 3D position of a fiducial depends on the number, distances, and distribution of cameras. Sensor uncertainties can and should be specified at a level that allows for maximum generality. This concerns specifications provided by system vendors as well as third-party accuracy assessments.

Error Distribution According to the GUM [ISO 08] sensor noise may follow different error distributions. By default, one might assume a Gaussian error distribution. However, there are tracking systems where global systematic distortions of the

tracking volume dominate the overall error behavior [Keit 08]. In such cases, a uniform distribution could be preferred in order not to underestimate extreme values. Also a combination of different error distribution might be used since systems often suffer from sensor noise as well as systematic errors.

Error Magnitude Finally the magnitude of the error has to be provided. Also the degree of the noise depends on the setup and on environmental influences. If the actual amount of error is not known a priori, simulation with different assumed error levels can still help to define the maximum allowed sensor noise to stay in-line with the overall application specifications.

The means to address these issues are often non-trivial. It was denoted the *elementary specification of sensor uncertainty* above. This corresponds to the specification of a functional and a statistical model in adjustment theory (cf. 10.3.3).

Ideally, the information is provided by the system vendor, e.g., [Wile 05]. Typically, high-precision ground truth measurements are needed for this purpose. However, this might not be applicable for all kinds of systems, e.g., the provider of a markerless tracking algorithm. The accuracy of a markerless tracking system is highly impacted by the used optics as well as the illumination and texture properties of the concrete environment. Therefore, no general statements can be given. In such cases, accuracy assessments as described in 12 might help. As a last resort, an a priori covariance can be assumed as an initial guess. It can then be refined in an iterative approach until accordance. This method has been formalized in the field of adjustment theory under the name *variance component estimation* [Koch 97] [Niem 08].

10.4. Conquering the Complexity of IAR Applications

Due to the reasons explained in 1.4, IAR tracking setups may become quite complex, requiring the interaction of various sensor devices and objects. Especially for those complex IAR setups it is challenging to assure compliance with production tolerances. This has to cover the entire chain of uncertainty from registration to real-time tracking. Typical generic vendor specifications are often insufficient to provide a reliable assessment of the tracking system performance in the explicit use case scenario.

Following the ideas described in 10.2, the application would have to be validated based on exhaustive empirical measurements in the target environment since the individual use cases and sensor constellations cannot be handled in a generic way be the system vendors. However, it is often very difficult to provide a sufficient number of ground truth measurements in the target environment. Other means are needed to strengthen the confidence in the tracking infrastructure.

We propose to combine a small set of representative empirical measurements with a comprehensive simulation to cover all possible constellations and system configurations in the use case scenario. Therefore we describe a framework that allows for a standardized and integrated implementation of all necessary computations.

The goal is to generalize the concepts for the assessment of uncertainties described in 10.2 and integrate them with the SRG and *Ubitrack* concepts described in the previous part of this document about data flow management. Therefore, statistical simulation based on the error propagation concepts described in 10.3.4 is incorporated. The following three sections describe the approach in more detail and distinguish it from related concepts.

10.4.1. Proposed Approach

Following industrial standards, classical validation requires exhaustive empirical measurements to validate the entire system operation [ASME 06] [VDI 02]. The combination of extensive statistical simulation, backed by selected empirical measurements is motivated by the ASME *guide for verification and validation in computational solid mechanics* [ASME 06]. The *Verification and Validation* [VeVa 11] addon for *Matlab Simulink* for example provides integrated verification and validation facilities using exactly these concepts, though not for tracking. The consistency of simulation, analytic error propagation, and empirical measurements is important to gain confidence in the correctness of the assumptions that were made regarding the error model. For this approach to be feasible, it has to be proven first that the simulation system yields realistic results.

This overall concept is integrated with the design phases of an IAR application according to 2.2, as depicted in Figure 10.4. During its definition phase, a verification of

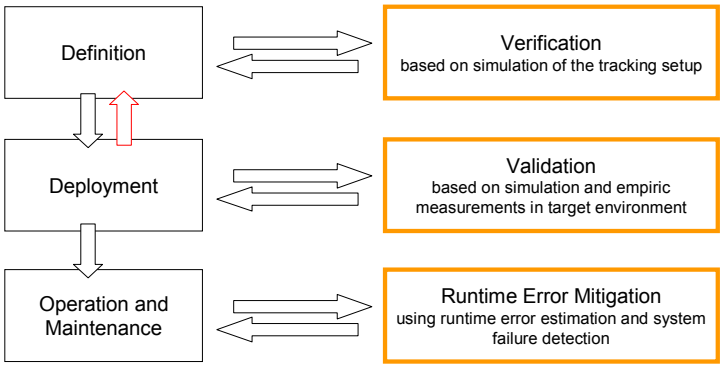


Figure 10.4.: IAR design phases

the intended IAR application is performed. It is mainly based on statistical simulation.

First, the correctness of the simulation system needs to be proven to assure that the assumptions made are specific enough to obtain realistic results. An important question is about the assumptions to be made for the type and magnitude of the basic sensor noise. For the example of an optical tracking system, uncertainties may be assumed either for the 2D observations on the image plane or for 3D observations in the tracking volume. As will be seen, this specification depends on the chosen sensor type(s) and the underlying measurement principles. It is also shown that simplifications regarding

the error model may be necessary and reasonable in order to handle the complexity of measurement uncertainty.

This step is typically based on extensive empirical measurements according to the concepts described in 10.2, paired with a detailed understanding of the underlying measurement principles. In many cases, the necessary information is already available, either in terms of vendor specifications or in terms of evaluations described in the literature. Having proved the validity of the simulation model, one can evaluate easily various aspects or hypothetical variations of the system. This would be very expensive if based on empirical data, or even impossible in case the assumed hardware is not at one's disposal. Proving the correctness of the simulation system can be quite complex. The good news is: it has to be shown only once for a certain constellation of sensors.

Once the correctness of the simulation system has been proven and a basic sensor noise assumption has been made, it is possible to benchmark various hypothetical scenarios even using different tracking systems, since no hardware is needed yet. Forward propagation can be applied to propagate the assumed uncertainties through the entire chain of uncertainty. By this, the behavior of the overall system and also of its individual components can be analyzed. Also, the influence of critical individual factors on the overall performance can be evaluated.

The final goal of the simulation is to verify the overall concept and to decide on a certain variant before deployment. It can help to state the general feasibility of a proposed setup, to justify purchase decisions, or to decide on which particular algorithms to use. To represent the system behavior correctly, it might also be necessary to simulate critical subsystems by refining black box systems down to the internal estimation algorithms. This depends on the basic sensor noise assumptions already mentioned above. The better the whole system is understood by simulation, the more accurate the subsequent validation will be.

After deployment, this is complemented by a validation based on empirical measurements in the target environment. For a realistic result, these measurements should incorporate the whole chain of uncertainties. This however makes measurement acquisition quite cumbersome, since a valid ground truth must be provided. In the target environment, this is typically possible only for very a small number of measurements. It shall be mentioned that for validation, it is not enough to use relative distance or point probing measurements as described in 10.2. This would neglect relevant parts of the chain of uncertainties, in particular the static uncertainties in registrations of markers and sensors in the world. Such relative measurements are only representative for uncertainties of individual systems, relative to their individual coordinate systems.

Basically, the findings from the verification stage are confirmed by these measurements. Ideally, the validation is as simple as this. In case of any contradictions, plausible explanations have to be found. In the worst case, the verification and validation process has to be reiterated, involving a refinement of the simulation model.

Last but not least, runtime error mitigation is needed in the operation & maintenance phase in terms of runtime consistency checks and techniques for the straightforward localization of system failures. A detailed system knowledge has been acquired during

verification and validation that increases the confidence into the overall system. This allows one to implement runtime error mitigation in terms of consistency checks for the automatic identification of possible system malfunction. These checks use confidence intervals for measurement noise and systematic error in the system to detect these error cases. Those can be interpreted and traced back to a certain system component to help guide mitigation steps.

To summarize the approach, the effort for validation based on empirical measurements is reduced. Instead of individually validating all IAR applications by exhaustive empirical measurements in the target environment, the simulation system shall be validated in a generic and reusable way. Still, exhaustive measurements are needed for this, but only once. Furthermore, this proof can be rendered in a dedicated laboratory environment instead of in adverse industrial target environments. Adaptation to the target environment then happens mostly by means of simulation. Only few empirical measurements are then needed to validate the IAR application finally in its target environment.

In 13, the feasibility of this approach is demonstrated in a test scenario using high precision ground truth data that is compared with the results of our simulation framework.

10.4.2. Focus

The proposed verification and validation approach is integrated with the *Ubitrack* and SRG concepts described in Part II. Verification could thereby be based on linear error propagation, as described in 10.3.3. For the purpose of an efficient and generic implementation, it has been decided to follow the approach of a Monte Carlo simulation, as described in 10.3.4. It has the advantage of supporting all existing and new *Ubitrack* functionality right away, without the need for providing a linearized functional model. Furthermore, the real-time capabilites of linear error propagation are not considered necessary for the described IAR applications. The focus is on deriving reliable, static guidelines for the IAR applications that can be incorporated into the exact specification of an industrial work process. Runtime uncertainty information is not considered here.

The verification & validation concepts are tested exemplarily on mainly two IAR scenarios. These two scenarios make heavy use of cooperative fusion approaches (cf. 2.3.2). Competitive fusion concepts based on Kalman [Kalm 60] [Koch 97] [Welc 01] [Hoff 00] or particle filters [Douc 01] also heavily rely on measurement uncertainties. They are not covered by subsequent evaluations, though.

Clipp et al. described a filtering concept that could make sense in combination with an industrial measuring application [Clip 07]. They run two Kalman filters, a “current” and a “delayed” one. The delayed filter is fed with the “good” observations only, according to the information of the current filter about e.g., reprojection error. This should result in a higher quality of the delayed signal. The current signal could be used for real-time visualization, the delayed signal for interactive measuring where a slight lag can be easily tolerated.

Related to this issue is the tradeoff between motion and measurement uncertainty. The measurement noise decreases over time due to averaging whereas the motion noise

increases over time due to increasing uncertainty of the motion prediction. Therefore, an "optimum frequency" can be found. It would be interesting to test the described concepts in IAR. However, this is beyond the scope of this thesis, refer to [Pust 11] for a treatment of this topic.

Furthermore, this part is restricted to the analysis of statistical errors. The problem of blunders that are often caused by human failure in the described scenarios, are not treated any further. Statistical tests as those mentioned in 10.3.3 could be applied for their treatement, as well as other robust appraoches such as RANSAC [Fisc 81] [Hart 00].

Azuma distinguished between static and dynamic errors. The latter are caused by two phenomena. Firstly, lag in communication paths always results in a delay between the physical action and the reaction of the AR system. Secondly, in a sensor fusion setup, the relative time-synchronization of the sensors with respect to each other is always an issue (cf. 5.4.3). Simple strategies are followed to mitigate the problem in this context. For a general solution, refer to [Schl 11].

10.4.3. Outline

To implement the described concepts, a Monte Carlo simulation system is integrated into the *Ubitrack* tracking framework in terms of several data flow components. This integration is described in Chapter 11. The simulation approach requires a detailed understanding of the used tracking devices, first of all to support the validity of the simulation system, in particular the inherent assumptions about the granularity of uncertainty specifications, and secondly also to quantify the individual sensor uncertainties. This analytic evaluation of individual sensors is treated in Chapter 12. In Chapter 13, the described verification & validation concepts are then applied to the exemplary IAR scenarios, the intelligent welding gun and the augmented airplane cabin. On this basis, an approach for runtime error mitigation is formulated in Chapter 14. A discussion in Chapter 15 wraps up the concepts for error management.

11. Ubitrack Monte Carlo Simulation Framework

For now, we assume the elementary specification of uncertainties of the involved sensors to be known. Different ways to determine them are described in 12. First, the adaption of Monte Carlo simulation techniques to *Ubitrack* is described. This starts with a closer look at the underlying simulation model, the *Ubitrack* data flow network in 11.1. Next, the required data flow components are described in 11.2. This is followed by an investigation of the computational complexity of Monte Carlo simulation in 11.3. This is completed by several exemplary simulations that give insight into typical registration and other problems.

11.1. The Chain of Uncertainty

The Monte Carlo simulation is to be implemented on the basis of an already existing *Ubitrack* data flow for the sake of simplicity. Furthermore, its implementation shall again yield a valid *Ubitrack* data flow for straightforward development.

From the *Ubitrack* point of view, the tracking data flow is fed by its source components. Driver components provide erroneous sensor measurements. Static spatial transformations are provided to the data flow, too, e.g., by the Static Transformation or Calibration Reader components. They have been computed by a precedent instance of another (registration) data flow and are also subject to error. This can be regarded as a recursive problem, tracing back all erroneous signals to non-static errnoneous signals of some sensors. A single sensor may even contribute several times to the overall application error if it is used not only in the application data flow, but also in precedent registration data flows. Pentenrieder called this backtrace of errors the *chain of uncertainty* [Pent 09]. An important facet of this problem is the incorporation of errors arising from the combination of offline metrologic and online tracking systems, as needed for example by the airplane cabin scenario (cf. 7.1.2).

General approaches to deal with this problem are not yet available. Pentenrieder lists various influences along the chain of uncertainty in a factory planning application. These are the uncertainties from the detection of markers, camera calibration (intrinsic parameters), referencing offset (registration), and model (rather the deviation of physical model from the CAD model). The author itself calls this strict sequence of a “flat SRG” concept. She describes forward uncertainty propagation rules to compute in real-time the overall application error in this case from the known input uncertainties, based on a linearized functional model. Different registration techniques based on 2D, 3D and 6DoF measurements are evaluated and compared based on empirical ground truth

measurements. The focus was on the practical viability of the registration methods. However, no rigorous comparison based on elementary specification of uncertainty is given. Furthermore, Pentenrieder conducted a simulation experiment based on synthetic camera images to assess the quality of optical square marker tracking. In this case, the application error is considered to be the 2D overlay error for the augmentation on the still image [Pent 06] [Pent 09].

Pentenrieder's chain of uncertainty actually has the topology of a chain and its chain links have the strict order given above. A *Ubitrack* data flow, however, has thetopology of a cyclic graph in general. This can be seen for example in the airplane cabin exemplary scenario, see 7.1.2 and 7.2. The two data flows logically belong together. They can be linked at the Calibration Writer component for the offset between the Reference Target (LED) and Reference Target (CAD) in the registration data flow and the corresponding Calibration Reader component in the application data flow. It becomes obvious that the tracking data of the probe tip is used twice, ending up in a single Application Push Sink Position component interfacing to the application, cf. also the complete data flows in Appendix A.1. Thus, the two paths originate and terminate at a common component, resulting in a cyclic data flow graph.

The application error can be a 2D overlay error as above, or any other data type supported in *Ubitrack*. For our IAR applications, rather 3D position or 6DoF pose measurements are relevant. In this thesis, the focus regarding error management is built on a consistent handling of the chains of uncertainties contributing to the overall application errors.

11.2. Simulation Data Flow

A generic simulation system is described next which is able to propagate forward such an elementary specification of uncertainty to simulate the uncertainty of the algorithm's outcome. It is integrated into the *Ubitrack* framework to allow for the simulation of arbitrary *Ubitrack* data flows, in the style of the *Matlab Simulink* [Simu 11] simulation concepts.

A *Ubitrack* simulation data flow consists of four elementary steps. First, ground truth data has to be provided. Second, synthetic measurements can be computed by adding noise to certain spatial transformations. Third, these measurements are fed repeatedly into the actual registration / tracking algorithm. Forth, from the results, a covariance can be estimated. These steps are be detailed in 11.2.1 - 11.2.4. A simulation data flow can be distinguished from a normal data flow in that it does not contain any real sensors. The special synchronization issues resulting from this are discussed subsequently in 11.2.5.

The necessary steps for construction of a simulation data flow are demonstrated step-by-step, using the example of a typical absolute orientation registration data flow based on user-triggered point measurements (cf. 7.2.1 and Figure 7.5). The complete SRG and DFG are depicted in Figure 11.1 and 11.2.

11.2.1. Ground Truth Data

The simulation system shall imitate the expected constellations of the planned real system. For this, hypothetical data has to be provided in terms of assumed spatial relationships. Hypothetical trajectories for moving objects can be provided in terms of previously recorded or synthetically generated continuous movements or alternatively in terms of a discrete grid. The Player or Calibration Reader components can be used to inject this data into the data flow network. Furthermore, static transformations describing the setup have to be provided. This synthetic data represents a ground truth; it does not contain any measurement errors.

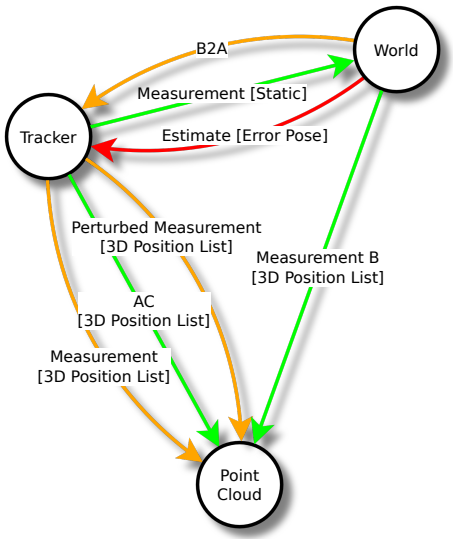


Figure 11.1.: SRG for solving the absolute orientation problem based on user-triggered measurements (cf. Figure 7.5). See also the corresponding DFG in Figure 11.2.

For the ground truth data to be consistent, it is necessary to avoid over-determination. This is equivalent to avoiding cycles in the SRG. Figures 11.1 and 11.2 depict the simulation SRG and DFG for our example. It is enough to specifiy the 6DoF transformation between Tracker and World (Static Pose) as well as a 3D Point Cloud known in the World (Calibration Reader). Instead of explicitly specifying the ground truth data in textual lists, suitable data can also be generated on-the-fly, using the Perturbation component. In our example, e.g., the Point Cloud could be created by sampling with a uniform distribtion from a specified volume. This often speeds up the specification of a suitable ground truth. This is sufficient to derive the Point Cloud also in the Tracker coordinate frame, are described next.

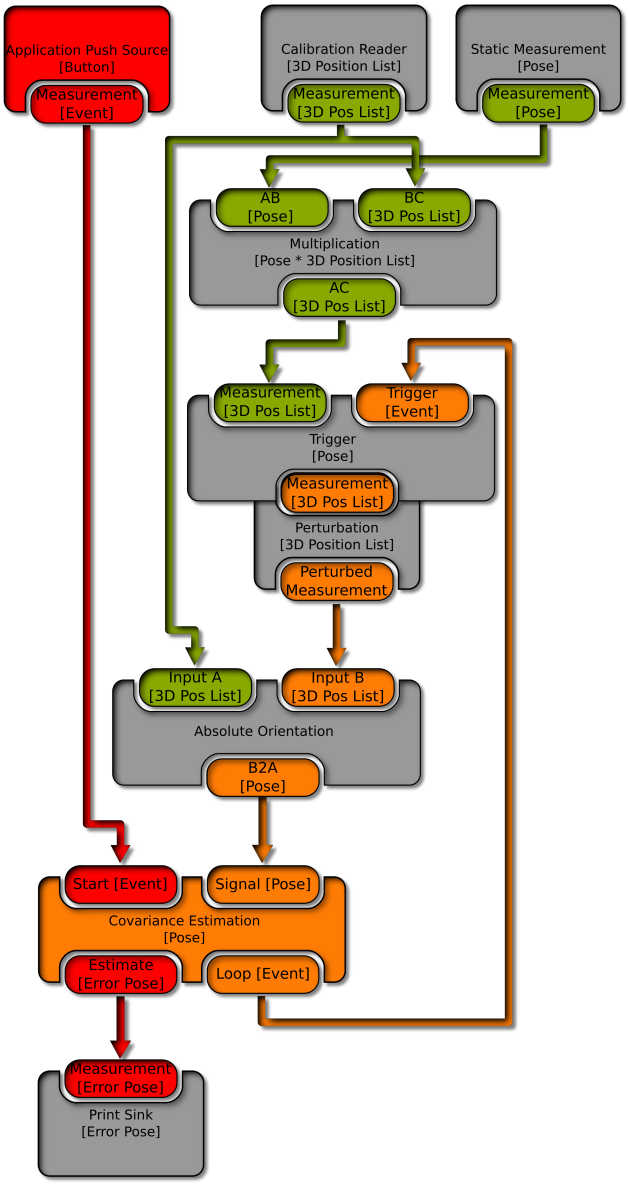


Figure 11.2.: DFG for solving the absolute orientation problem based on user-triggered measurements (cf. Figure 7.5). See also the corresponding SRG in Figure 11.1.

11.2.2. Synthetic Measurements

Based on the ground truth data specified according to 11.2.1, synthetic measurements are generated. For this, the ground truth data first needs to be transformed into the coordinate systems in which the elementary specification of uncertainties of the involved sensors is provided. The usual spatial-relationship patterns are used for this. In our

example in Figures 11.1 and 11.2, the Multiplication component transforms the Point Cloud to the Tracker coordinate frame. Then, by sampling from the given probability density function, the data is perturbed accordingly. The Perturbation component is again used for this. The Trigger component standing in-between are explained below. This is the general procedure for the generation of synthetic measurements.

Note that ground truth data does not necessarily have to be given at the abstraction level of the elementary specification of uncertainty. For example, the ground truth may be specified conveniently in terms of 6DoF poses whereas error is assuemd in 2D pixel coordinates. Another example making use of this property are provided below in 11.4.

In case a static transformation has already been estimated in a precedent simulation step, its mean can be directly adopted and its covariance can be used to parametrize the Perturbation component, instead of using the elementary specification of uncertainty of an individual sensor.

The GUM suggests to use either a Gaussian, a uniform, or a triangular distribution for the specification of measurement uncertainties [ISO 08]. In a sensitivity analysis of the hand-eye calibration problem, Dornaika used either uniform or Gaussian distributions [Dorn 98]. The Perturbation component also supports uniform and Gaussian distributions. In case an estimated variance or covariance matrix shall be used to parametrize a uniform distribution, the corresponding minimum/maximum values have to be computed according to Equation (10.6). The Perturbation component currently only supports an uncorrelated and isotropic error model. The current implementation could be improved by a method for the generation of multidimensional uniformly distributed random variables. It should support general covariance matrices having also off-diagonal entries. See also 10.1.2.

11.2.3. Registration/Tracking Algorithm:

The perturbed measurements are propagated through the data flow network of the algorithm to be simulated. This resembles the normal calibration/registration or application data flow. Various kinds of spatial relationship patterns are at our disposal, besides the trivial Inversion and Multiplication, this comprises many common calibration and registration methods, see also 5.3.2. In our example in Figure 11.2, the data flow to be simulated consists of the single Absolute Orientation component only. However, more complex data flows would also be possible. Unlike in 7.2.1, the pattern is now used in the implicit space-expanded form. The measurements actually do not have to be captured manually, thus, a whole list of measurements can be processed in one step.

11.2.4. Covariance Estimation

The previous two steps, generation of synthetic measurements (cf. 11.2.2) and running the registration/tracking algorithm (cf. 11.2.3) are repeated iteratively. Samples are produced by perturbation and then propagated through the data flow by running the corresponding algorithms. Each iteration results in a single output of the simulated algorithm. Accumulating these output values, using descriptive statistics, the covariance

associated with the registration or tracking result can be estimated by the Covariance Estimation component [Niem 08] [ISO 08] [Koch 97] [Dimo 05]. After the specified number of iterations, a single mean and covariance is dumped, in our example in Figure 11.2 to the Print Sink component.

For efficient incremental estimation of the pose covariance, the following relationship can be exploited [Koch 97]:

$$\mathbf{VAR}[\boldsymbol{X}] = \mathbf{E}[(\boldsymbol{X} - \boldsymbol{\mu})^2] = \mathbf{E}[\boldsymbol{X}^2] - \mathbf{E}[\boldsymbol{X}]^2. \tag{11.1}$$

Thus, the empirical mean value μ does not have to be known in advance. The two expectation values $\mathbf{E}[\boldsymbol{X}^2]$ and $\mathbf{E}[\boldsymbol{X}]^2$ can be estimated incrementally, by representing Equation (10.1) as a weighted sum. In a recursive approach, the first summand represents all previous measurements, the second summand represents the current measurement. Assuming that $n-1$ measurements were processed already, the $\mathbf{E}[\boldsymbol{X}^2]_n$ can be computed according to

$$\mathbf{E}[\boldsymbol{X}^2]_n = \frac{n-1}{n}\mathbf{E}[\boldsymbol{X}^2]_{n-1} + \frac{1}{n}\boldsymbol{x}_n^2. \tag{11.2}$$

Similarly, $\mathbf{E}[\boldsymbol{X}]_n^2$ is obtained by the recursive formula

$$\mathbf{E}[\boldsymbol{X}]_n = \frac{n-1}{n}\mathbf{E}[\boldsymbol{X}]_{n-1} + \frac{1}{n}\boldsymbol{x}_n \tag{11.3}$$

and applying the square after the last measurement has been incorporated.

For multidimensional measurements, the scalar product in the above equations is replaced by the outer vector product and one obtains the covariance matrix $\boldsymbol{\Sigma}_X$ instead of $\mathbf{VAR}[\boldsymbol{X}]$. Therefore, the application of Equations (10.1) and (10.2) is straightforward for the expression of 2D or 3D positional errors.

For the representation of orientation, however, additional steps are necessary. For a further description of quaternions in general, see also 10.1.4 and [Horn 87] [Kuip 02] and [Pust 04, Pust 08]. The *additive* 7x7 covariance matrix estimated using Equations (11.1), (11.2), and (11.3) has to be converted to a *multiplicative* 6x6 covariance matrix compatible with Equations (10.19) and (10.20). In other words, one would like to represent an arbitrary estimated rotation $\boldsymbol{q}$ by a multiplication of the unknown true quaternion $\boldsymbol{q}_0$ with a small multiplicative error quaternion $\boldsymbol{q}_e$ according to Equation (10.18).

$$\boldsymbol{q} = \boldsymbol{q}_0\boldsymbol{q}_e = \boldsymbol{q}_0(\boldsymbol{q}_{\text{id}} + \boldsymbol{q}_{\text{add}}) \tag{11.4}$$

Here, $\boldsymbol{q}_{\text{id}}$ is the identity quaternion $((0,0,0),1)$ and $\boldsymbol{q}_{\text{add}}$ is a quaternion with expectation $((0,0,0),0)$ and a 3×3 covariance matrix covering only the imaginary part. The variance of the real part is assumed to be 0 to obtain a non-over-paramterized uncertainty specification, see also [Mark 04]. Together $(\boldsymbol{q}_{\text{id}} + \boldsymbol{q}_{\text{add}})$ represents a small quaternion $\boldsymbol{q}_e = ((e_{rx}, e_{ry}, e_{rz}), 1)$.

When estimating the mean and covariance of the quaternion according to Equations (11.1), (11.2), and (11.3), however, one obtains the following additive representation instead:

$$\boldsymbol{q} = \boldsymbol{q}_0 + \boldsymbol{q}'_e \tag{11.5}$$

Note that q_e and q'_e are not identical. Equating Equations (11.4) and (11.5), this yields

$$\begin{aligned} \boldsymbol{q}_0(\boldsymbol{q}_{\text{id}} + \boldsymbol{q}_e) &= \boldsymbol{q}_0 + \boldsymbol{q}'_e \\ \boldsymbol{q}_{\text{id}} + \boldsymbol{q}_e &= \boldsymbol{q}_0\boldsymbol{q}_0 + \boldsymbol{q}_0^{-1}\boldsymbol{q}'_e \\ \boldsymbol{q}_e &= \boldsymbol{q}_{\text{id}} + \boldsymbol{q}_0\boldsymbol{q}'_e - \boldsymbol{q}_{\text{id}} \\ \boldsymbol{q}_e &= \boldsymbol{q}_0\boldsymbol{q}'_e \end{aligned} \tag{11.6}$$

Thus, one has to rotate the distribution by the inverted quaternion $\boldsymbol{q}_0$. The variance of the real part can then be discarded, as it should be ≈ 0. This can be seen from Equation (10.22): for small ϕ, the gradient of $cos(\phi)$ is small, and so is the variance of $\boldsymbol{q}_{e,w}$.

11.2.5. Data Flow Synchronization

The simulation has been implemented in terms of a few additional data flow components. Their interaction is discussed next.

Basic Operation

Two components occuring in Figure 11.2 have not been described so far, the Application Push Source (Button) component and the Trigger component. The former is used to kick-start the simulation once in the beginning. Unlike during normal operation where at least one sensor pushes events into the data flow network, thereby triggering the computation of a result (cf. 5.4), in the simulation data flow there is no natural source of PUSH events. Therefore, the Covariance Estimation component itself drives the data flow; indeed it runs a dedicated thread for this. Whenever a new iteration is initiated, an event is issued on one of its output ports. Generally speaking, this event activates upstream parts of the data flow such that a new outcome of the data flow algorithm can be computed. The Covariance Estimation component then just waits for the outcome to be computed before initiating the next iteration. This goes on until the preconfigured number of outcomes has been incorporated and the estimated covariance can be dumped. In our example, the event activates the Trigger component. Whenever it receives an event, is pulls a measurement and pushes it onwards to (again) initiate a new sample of the outcome of the algorithm.

First of all, this push-style synchronization scheme ensures that the simulation data flow runs efficiently, as a new iteration is triggered immediately after the previous iteration has been completed. Furthremore, it also allows for nesting of data flow operations, as discussed next.

Systematic Sampling

In case one wants to systematically vary some of the spatial assumptions (cf. 11.2.1), e.g., to sample a predefined grid of positions within the tracking volume, the Covariance Estimation has to be repeated for each grid position. To automate this, the Trigger Loop component can be used, as depicted in Figure 11.3.

As with the Covariance Estimation, the Trigger Loop is triggered externally and runs its own thread internally, allowing for recursive nesting. It also issues events to initiate the next iteration. In each iteration of the loop, one or several Buffer components can be updated; the necessary ground truth information can e.g., be fetched from preconfigured lists of data in textual representation using the Calibration Reader and List Extractor patterns.

Note that the initialization, i.e. updating the Buffer, is guaranteed to be executed before the Covariance Estimation component is triggered anew since the latter has a lower priority in the *Ubitrack* data flow. See also 5.4. Another interesting aspect revealed in Figure 11.3 is the Synchronized-PUSH operation in the inner covariance estimation loop. The two Trigger components assure that Paramter B and Parameter C both obtain the timestamp associated with the Loop [Event] issued by the Covariance Estimation. Therefore, the Black Box Algorithm can process them, see also 5.4.

Instead of a single covariance, many covariances are generated. They can be gathered in a text file for post-processing in an external tool, using the Recorder pattern. This strategy was pursued in the evaluations of the industrial use case scenarios discussed in Part 13.

Nesting Operations

In the case where an application data flow requires the prior registration of a static transformation, the question arises whether to simulate everything in a single monolithic data flow or to determine the uncertainty of the registration in a prior simulation step. The latter method is simpler to implement, however useful information may get lost by describing the uncertainty in terms of a covariance and injecting it into the subsequent application data flow, together with the assumption of a uniform or Gaussian distribution. Therefore, the implemented simulation concept also allows for the nesting of such operations.

A practical application of this technique is given in the validation of the airplane cabin exemplary scenario in 13.2.2. The metrological probe whose uncertainty is to be simulated is also used in the prior registration step for the reference target (cf. 7.2.3). In a monolithic data flow, a list of 25 perturbed positions of the probe tip is generated on-the-fly for all 25 drillings in the reference target before generating one perturbed sample of the offset between the CAD model and the LEDs. The basic data flow design pattern is depicted in Figure 11.4. This technique will be picked up in 13.2. The complete SRG and DFG can also be found in Appendix A.1.

Often, nested loops are needed to construct lists of artificial measurements on-the-fly. The Time-To-Space-Expansion Converter component is used for this purpose in the depicted data flow. Note that *Ubitrack* supports one-dimensional lists of measurements only to be conveyed between output and input ports. If each element of a list A shall be used to generate a whole list of measurements B_i, the creation of the lists B_i in general has to be rolled out as a nested operation.

In the example, the nested loop is located right in the beginning of the covariance estimation loop, therefore, the Trigger Loop is triggered directly by Covariance Estimation

component. For an arbitrary other location, the Trigger Loop would have to be activated by another Signal Generator component. Recursive nesting of operations is feasible with this technique.

11.3. Computational Complexity

An interesting question is the number of samples needed for a sufficiently accurate estimate of the covariance. I have chosen to present a rather informal consideration in this context. It neglects the influence of multiple sources of uncertainties as well as multidimensional correlated random variables. Some aspects of these issues are also discussed in [Dimo 05].

For estimating a mean value, Equation (10.3) gives insight into the convergence rate of the Monte Carlo method.

$$\mathbf{VAR}[\bar{X}] = \frac{\sigma^2}{n} \Rightarrow \sqrt{n} = \frac{\sigma}{s_{\bar{X}}} \tag{11.7}$$

Thus, if we want to achieve a standard deviation for the mean of 1/10 of the assumed uncertainty σ, then 100 samples are needed.

Our main interest however is the estimation of a covariance matrix. For the variance of the empirical variance, the following relationship can be shown [Kenn 52]:

$$\mathbf{VAR}[s^2] = \sigma^4(\frac{2}{n-1} + \frac{\kappa}{n}) \tag{11.8}$$

where κ is the *excess kurtosis*, which is 0 for a Gaussian distribution. Consequently, if we want to achieve a standard deviation for the empirical standard deviation of 1/10 of the assumed uncertainty σ, then $\approx$ 20000 samples are needed. This consideration just sheds light on convergence given an assumed standard deviation σ. This σ is not only influenced by the uncertainty of the observations but also heavily depends on the sensitivity of the estimation algorithm itself [Robe 04]. No general statement can be given. Therefore, the number of samples should be chosen adaptively, taking into consideration the problem at hand as well as the desired degree of approximation [Cox 06]. An example for the convergence of the absolute orientation algorithm [Horn 87] is discussed below in 12.3.2.

11.4. Example: Comparison of Monte Carlo with Online Error Propagation

The following example demonstrates the results of Monte Carlo simulation in a direct comparison to analytic error propagation (cf. 10.3) [Pust 10]. The task is the estimation of pose tracking uncertainty for a point-of-interest (POI) to which a fiducial marker similar to 1.6(b) is rigidly attached. Both propagation techniques use identical assumptions, in particular the elementary specification of uncertainty is given in 2D. Ground truth

data is assuemd in terms of 6DoF poses. The SRG and the corresponding explicitly space-expanded pattern (cf. 5.3.5) are depicted in Figure 11.5.

The pattern does not represent a full-fledged pose estimation; rather only the NLLSQ optimization is performed, based on the given Initial Pose of the Body with respect to the World. This is enough for our simulation purposes since the required initialzation can be taken directly from the ground truth data. Furthermore, the intrinsic (projection matrix) and extrinsic (pose of Camera with respect to the World) camera parameters have to be known. From this, the corresponding 2D coordinates on the individual Image Planes can be derived by projection and artificial measurments can be generated by perturbation. From this, the pose of the Body can be computed again.

The space-expansion affects the number of cameras available in the pattern, two in this case. The time-expanded version of the pattern would consist of only one, though movable camera. Unlike for the Absolute Orientation pattern, time-expansion would not make much sense here since the extrinsic camera parameters are assumed to be known, which means that the camera would have to be tracked such that the pattern can be applied. With space-expansion, the extrinsic camera parameters are assumed to be constant. This demonstrates that not all expansion types make sense for all algorithms [Pust 06b].

The simulation is a simplification of the real situation, in particular the effects of fully or partially hidden fiducials are neglected; rather, only their 2D and 3D coordinates are used. The analytical model uses backward propagation (identical to NLLSQ) to estimate the 6DoF uncertainties from 2D pixel uncertainties. It then uses forward propagation to propagate the 6DoF error to any Point of Interest (POI). The simulation uses the same NLLSQ algorithm for pose estimation. 1,000,000 samples were used to estimate the pose covariance. The complete SRG and DFG can be found in Appendix A.1.

The purpose of this evaluation, that is not reproduced in detail here, was to estimate the point-of-minimum-error (PME) in the coordinate system of the fiducial marker at which the translational uncertainty is minimized. It could be shown that the PME is generally distinct from the center-of-gravity (COG) of the fiducial marker. Furthermore, a method is provided that allows one to deduce the PME given the covariance matrix at the COG, the POI, or any other point in the coordinate system of the fiducial marker. For details, please refer to [Pust 10]. The important fact that is relevant in this context, is just stated here without reproducing the data. It says that the analytic error propagation and the Monte Carlo simulation yielded the same result, aside from some small differences in the off-diagonal entries of the covariance matrix.

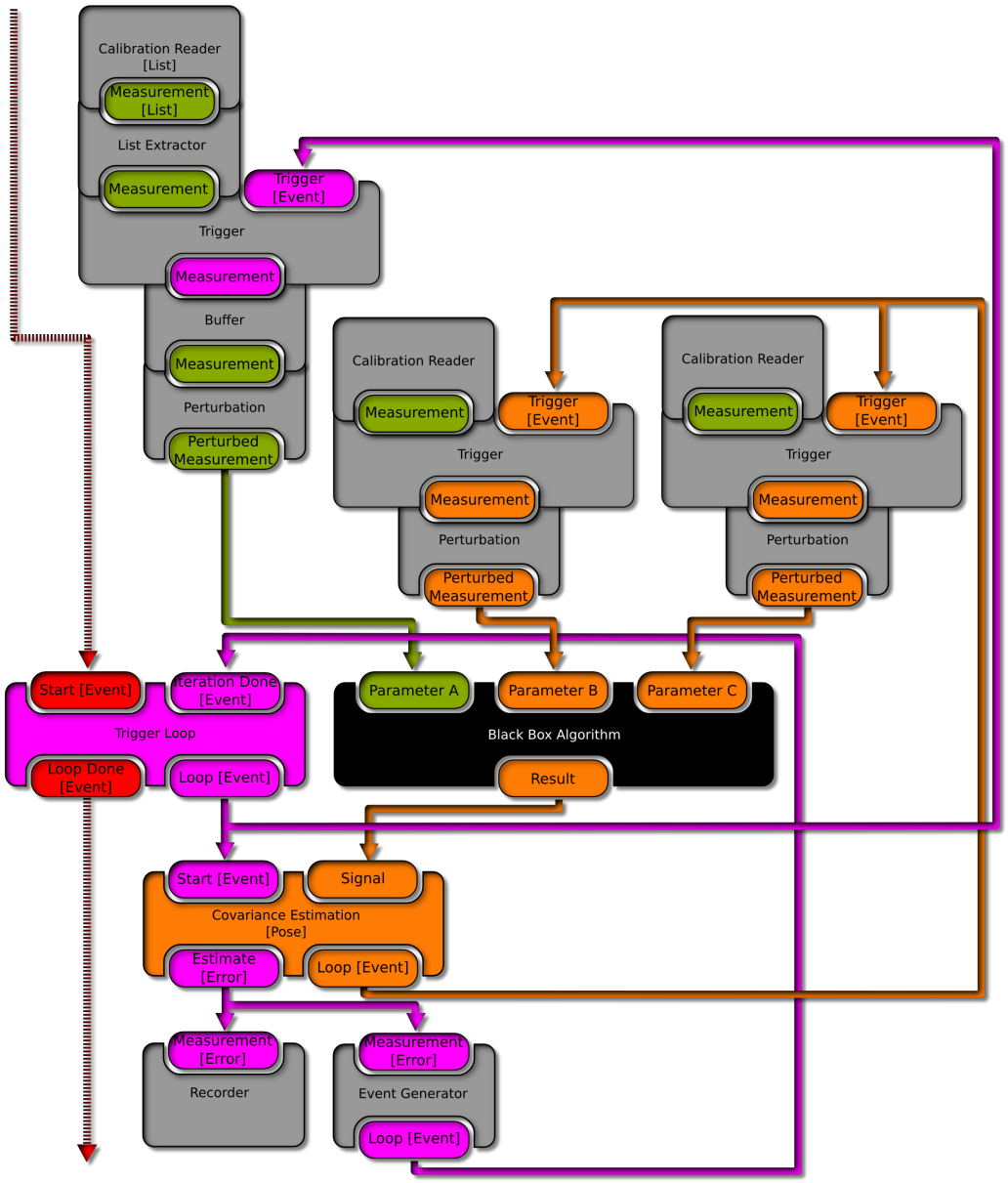


Figure 11.3.: Simulation data flow with systematic sampling from a list of data. The Trigger Loop systematically varies Parameter A of the algorithm to be simulated between individual runs of the Covariance Estimation whereas Parameter B and Parameter C are just reused over and over again. The Event Generator informs the outer Trigger Loop about the termination of the inner Covariance Estimation loop. The three reddish colored arrows indicate the three different levels of PUSHes propagated through the data flow and reveal the nested loop structure at the same time. The list of estimated covariances is stored by the Recorder component.

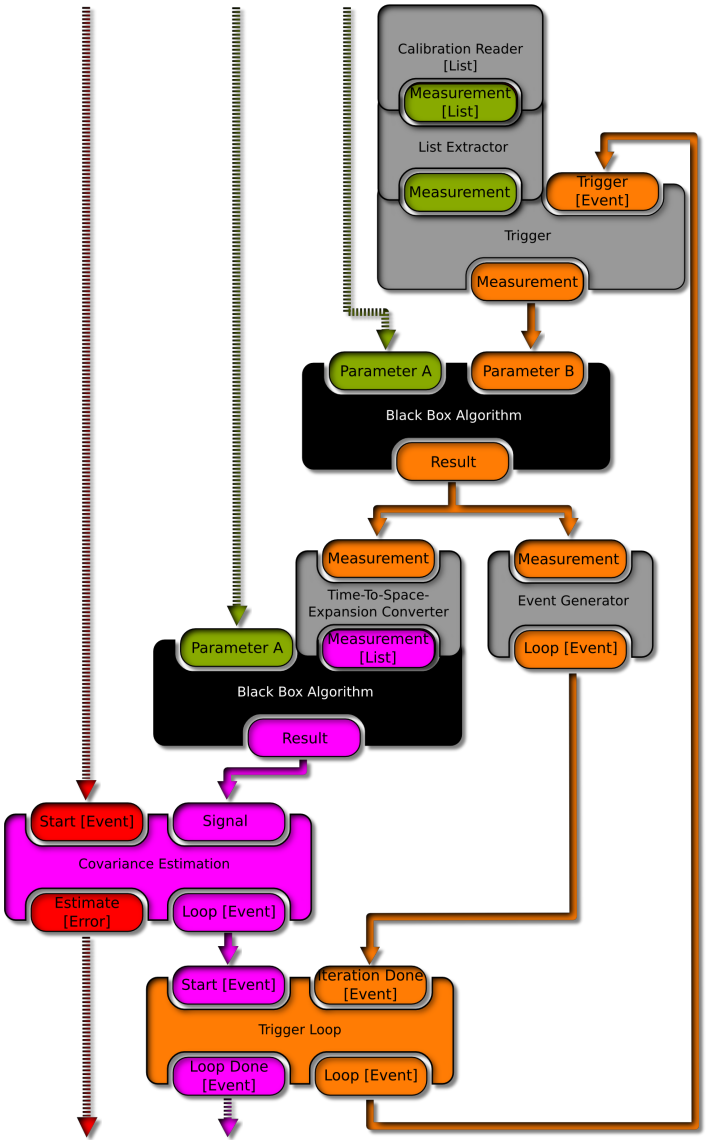
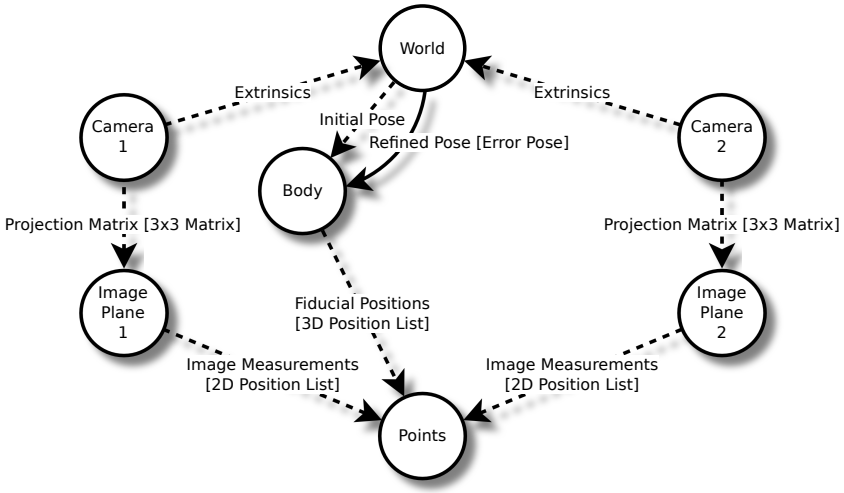
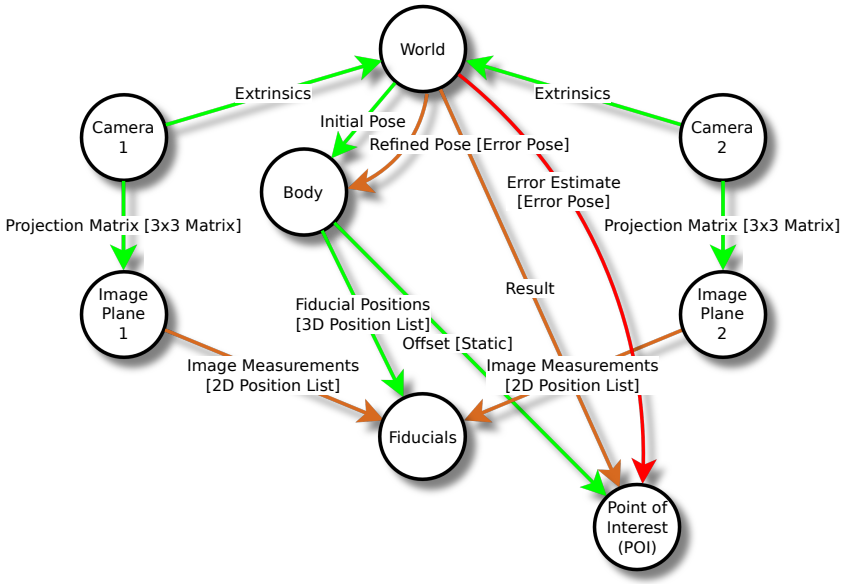


Figure 11.4.: Simulation data flow with nested on-the-fly generation of a list of synthetic measurements. In each iteration of the Covariance Estimation, a nested loop is executed, steered by the Trigger Loop and Event Generator components. Both components syntactically confine the nested loop; whenever the Event Generator receives a pushed measurement, it informs to Trigger Loop to initiate the next iteration of the nested loop. When the preconfigured loop counter is reached, the Trigger Loop issues a final event on its other output port. The nested loop feeds individual measurements into the Time-To-Space-Expansion Converter which aggregates a list. As soon as the list contains the determined amount of data, the outer Covariance Estimation loop continues operation.



(a) SR pattern



(b) SRG

Figure 11.5.: SR pattern and SRG for the 2D-6D pose estimation problem. Two cameras track the fiducials of a marker (Body) inside the commonly visible volume. Their 6DoF poses in the world are known (Extrinsics), as well as their projection matrices. From the 2D observations of both cameras, a 6DoF pose is reconstructed by a NLLSQ method. The uncertainty is estimated for an assumed Point of Interest (POI) that is rigidly connected with the marker. In the SRG, only the important spatial relationships are shown, most intermediary results are hidden for the sake of clarity.

12. Sensor Errors

In 10.3 and Chapter 11, the propagation along the chain of uncertainties was described. So far, the *elementary specification of sensor uncertainty* was assumed to be given. In some cases, it is given by the system vendor, however often, the desired specification is not available. Important standards and guidelines for the determination of uncertainties for spatial measurement devices were already introduced in 10. This chapter addresses the topic of formulating an elementary specification of uncertainty in some more detail.

First, a review of related work is given in 12.1 regarding the assessment of uncertainties for sensors typically used in AR. Next, the term *elementary specification of uncertainty* introduced in 10.3.5 is further motivated in 12.2. Then, two different techniques are described in 12.3 that allow for an assessment of the sensor uncertainties based on 3D position measurements. The presented concepts are not exhaustive, rather, the focus is laid on optical tracking systems and online metrologic measurement devices such as mechanical measurement arms, coordinate measurement machines (CMM), and laser trackers. This is accompanied by practical examples and case studies. A detailed example in 12.4 completes the treatment of sensor uncertainties.

12.1. Assessment of Uncertainties in the Literature

Ideally, the vendor itself publishes a basic sensor uncertainty specification for his products. This is a must for vendors in the metrology domain where specifications are often published on their website[1]. Some vendors provide an even more detailed insight into their systems, thereby allowing for detailed reverification tests at the customer's site [Wile 05]. Schmidt et. al use a high precision linear stage to evaluate various NDI tracking systems along the three spatial axes [Schm 09].

The situation is less straightforward for tracking systems that emerged in the MR domain, where the assessment of uncertainties is often still a research issue.

Satoh et al. use an industrial robot to move an HMD through the tracking volume to evaluate the *VICON* [Vico 11] and *laserBIRD* [lase 11] optical tracking systems [Sato 06]. It allows for systematically reproducible measurements with arm poses known with high precision. The pose error of the HMD as well as registration error in pixels is evaluated. Thereby, jitter at a constant pose is distinguished from systematic error in relative position and orientation changes. Using this technique, substantial parts of the chain of uncertainty from tracking to visualization can be covered, though e.g., the world registration or the arrangement and design of markers, still lacks. A

[1] Examples are *FARO* [Faro 11a] and *NDI* [NDI 11]

framework is presented for this purpose which is applicable also to other systems. However, it requires dense ground truth data from a high-precision industrial robot. We believe that the major contribution of such works—besides validating a particular AR environment—is restricted to an elementary statement about the investigated tracking systems only.

Rohling et al. conducted a comparison of the accuracy of a mechanical measurement arm[2] and an optical position tracker[3] for image-guided neurosurgery. Thereby, IR LEDs are mounted rigidly near the tip of the probe. Relative distances are compared against the gold standard obtained from a calibration device [Rohl 94].

Other works restrict themselves to the assessment of an individual system. Much of the work has been conducted in the field of optical square marker tracking [Pent 06] [Zhan 02] [Malb 02] [Abaw 04]. Malbezin et al. pointed a camera towards the marker from different positions and angles. They assessed the error in camera pose of the *ARToolKit* tracker using physical measurements as a reference. The latter were obtained using complex manual measurements and geomteric constructions on the plane containing the marker as well as the tripod carrying the camera with a plumb-line mounted below. Maximum errors of these more or less rude methods and their contribution to the pose error are pre-estimated and considered in the analysis of the results [Malb 02].

Zhang et al. compare four different available optical square marker tracking systems according to various factors, including accuracy [Zhan 02]. They measure feature registration error on the image plane. As a ground truth they use the *well established* corner detection algorighm of *OpenCV* as well as an alternative line-fitting algorighm to estimate the features *with the best possible accuracy*. Thereby, they did not rely on pose errors or back-projection since those methods depend on camera internal parameters and calibration algorithms.

In contrary to that, Abawi et al. estimated pose error, again using physical measurements [Abaw 04]. Thereby, systematic error and standard deviation are treated separately. Both may have huge impact on the overall error. From this, an accuracy function is derived for the prediction of accuracy based on the pose of the camera with respect to the marker.

Lieberknecht et al. use a *FARO* measurement arm to generate ground truth image data which is then used to assess the quality of different markerless tracking algorithms [Lieb 09].

12.2. Elementary Specifiation of Uncertainties for Sensors

The assessment of uncertainties as described above in 12.1 is tedious work and often also requires the use of expensive equipment to obtain ground truth measurements. Unfortunately, only some of the described works is useful in a general means for the purposes of propagation of uncertainties according to 10.3. For example, the knowledge of pose errors in monocular square marker tracking for a certain constellation of camera

[2] *FARO*
[3] *NDI Optotrak*

linearization model, marker tracking algorithm, marker type and size, camera model, and lens model is not generally useful to predict the overall application error in similar setups.

To get the most benefit from expensive accuracy assessments, a standardization of the process is required. Uncertainties must be assessed and specified in a way that allows for flexible reuse of the data.

In the mentioned example it would be more useful to have knowledge about 2D uncertainties on the image plane of a certain constellation of linearization model, camera, and lense. It could be reused to predict pose errors also in similar setups using other tracking algorithms, marker layouts and dimensions. Probably, an additional uncertainty value has to be estimated in 2D for the feature detection algorithm being used. However, the uncertainties of image acquisition should be distinguished from those of the feature detection for maximum reusability.

Similarly, in a multi-camera IR tracking setup, pose uncertainties heavily depend on the arrangement of the cameras (extrinsic camera parameters). Knowledge of the pose uncertainties in a given setup is not transferable to other setups, even if the same number of cameras is used. If the uncertainties were specified in terms of 2D image plane errors for individual fiducials, pose errors for any setup using such cameras could be easily predicted.

For multi-camera setups installed in a single, rigid frame, the specification of uncertainties for individual fiducials in also 3D comes into consideration. NDI follows this approach. In this case, the manufacturer's calibration protocol that has been published for customer reverification tests, is also based on ground truth measurements in 3D [Wile 05]. This way, the specification can be used to predict pose errors for targets of arbitrary geometry and numbers of fiducials.

The challenge is to state the uncertainties in a way that makes them maximally reusable and guarantees consistent simulation results at the same time. This shall be called the *canonical representation* of the uncertainty specification and corresponds to the definition of the elementary specification of uncertainties given in 10.3.5. First of all, a data type has to be chosen for a particular system that represents *best* the properties of the system. Then, the magnitude of the basic uncertainty has to be either directly measured, or inferred by some suitable means. Methods to accomplish this in the context of optical IR tracking is presented in the remainder of this chapter. A transfer to other sensor systems is considered necessary by the author, it is however beyond the scope of this thesis. The correctness of the chosen uncertainty specifications regarding their capability to yield realistic simulation results is analyzed in this context.

It shall be noted that the basic uncertainty specification should not be restricted to errors that are caused by the physical entity one would intuitively associate with the data type chosen for this specification. In other words, if 2D image plane errors are chosen as the error model, then the specified quantity should not only comprise CCD pixel noise, as assumed by Bauer [Baue 06] [Baue 07]. It should also contain for example those uncertainties that result from the inadequateness of the camera model or fabrication tolerances of the used optics. Bauer already observed that such systematic effects dominate the overall tracking accuracy. Nevertheless, he excluded them from his real-

time prediction system for instrument tracking. He stated that such systematic effects should rather be removed by improved calibration procedures. Indeed, it is desirable to reduce uncertainties by using better models and calibration routines. Yet, they will never be fully eliminated. Following the philosophy of the GUM, they should rather be accepted and incorporated, subsumed unter the pragmatic term *uncertainty*.

12.3. Residual Error from Point-Based Registration

Many registration methods are based on corresponding 2D or 3D position measurements. Several such methods are available in *Ubitrack*, e.g., the Absolute Orientation or SPAAM Calibration (cf. 5.3.2 and 7.2). The 3D-2D problem can be solved using DLT, followed by an iterative NLLSQ optimization that minimizes geometric error [Hart 00]. For the 2D-2D and 3D-3D problems, linear closed-form solutions exist that directly yield the least-squares solution. See also 10.3.1.

Point correspondences have to be collected for all these algorithms. In the following two methods are discussed that allow one to assess sensor uncertainties from this data. Basically, it does not matter whether the point correspondences are collected for the sake of registration or quality assessment or both. Our pragmatic approach aims at estimating the accuracy of one or several tracking devices, after they have been installed in a certain setup and also after they have been calibrated using the proprietary methods provided by the vendor. This also allows one to perform the analysis for the typical tracking volume to be used later on in the application. Furthermore, typical pointing devices, measurement tools or other targets can be incorporated, as long as they allow for reproducible point measurements.

The following treatment mainly discusses the 3D-3D case, although the proposed methods can also be used in the other cases. Two techniques are discussed in 12.3.1 and 12.3.2, the NLLSQ Helmert transformation used in metrology, and a linear approximation to this concept. Initially, the registration incorporates ground truth data that can be obtained, e.g., from high-precision coordinate measurement machines or mechanical measurements arms. The final goal is the derivation of a method that allows one to assess uncertainties even *without* having ground truth data.

Assuming n corresponding point measurements $\{p_i \leftrightarrow q_i\}$, a pose transformation matrix $\boldsymbol{H}$ can be estimated such that

$$\boldsymbol{p}_i = \boldsymbol{H}\boldsymbol{q}_i \qquad \text{for all } i \in [1..n]. \tag{12.1}$$

$\boldsymbol{p}_i$ and $\boldsymbol{q}_i$ are thereby assumed to be homogeneous vectors. For tracking scenarios, $\boldsymbol{H}$ normally is an invertible rigid (euclidean) transformation of 2D (3 DoF) or 3D (6 DoF) space, or a 3D-2D projection (11 DoF) [Hart 00]. If the problem is overdetermined, the measurements won't fit perfectly and Equation (12.1) becomes $\boldsymbol{p}_i \approx \boldsymbol{H}\boldsymbol{q}_i$ instead. The main idea is the statistical analysis of residual errors arising from this over-determination.

12.3.1. General Non-Linear Solution: Helmert Transformation

In adjustment theory, the problem of matching point clouds is treated by an NLLSQ approach (cf. 10.3) called *Helmert transformation*. It allows one to estimate spatial transformations in 1D/2D/3D having various DoF. Free and commercial tools are available for this purpose. For our purposes, the free *Java Graticule 3D* [Grat 11] is used. It implements the method described by [Jage 05]. Therefore, only the fundamental idea is sketched here.

Two point clouds are assumed to be given, measured independently in coordinate frames P and Q, potentially using distinct devices. One of these point clouds, say P, is defined to be the *target* coordinate system. These observations are gathered in the observation vector $\boldsymbol{L}$, consisting of vectors $\boldsymbol{L}_P$ and $\boldsymbol{L}_Q$. The basic idea is to estimate from all oberservations $\boldsymbol{L}_P$ and $\boldsymbol{L}_Q$ a consistent point cloud $\hat{\boldsymbol{X}}_P$ in coordinate system P as well as transformation parameters $\hat{\boldsymbol{H}}$.

This functional model f is linearized around the initial solution $\boldsymbol{X}_0$ and $\boldsymbol{H}_0$ and contains only their corrections $\hat{\boldsymbol{x}}_P = \hat{\boldsymbol{X}}_P - \boldsymbol{L}$, $\hat{\boldsymbol{h}} = \hat{\boldsymbol{H}} - \boldsymbol{H}$. Since only the corrections are included, this is also called *shortened observation equations*.

$$\hat{\boldsymbol{l}} = \underbrace{\begin{bmatrix} \boldsymbol{l}_P \\ \boldsymbol{l}_{Q\text{in}P} \end{bmatrix}}_{\boldsymbol{l}} + \underbrace{\begin{bmatrix} \boldsymbol{v}_P \\ \boldsymbol{v}_{Q\text{in}P} \end{bmatrix}}_{\boldsymbol{v}} = \underbrace{\begin{bmatrix} \boldsymbol{I} & \boldsymbol{0} \\ \boldsymbol{I} & \boldsymbol{A}_f \end{bmatrix}}_{\boldsymbol{A}} \underbrace{\begin{bmatrix} \hat{\boldsymbol{x}}_P \\ \hat{\boldsymbol{h}} \end{bmatrix}}_{\hat{\boldsymbol{p}}} \tag{12.2}$$

Here, $\boldsymbol{A}_{f(\boldsymbol{H}_0)} = \boldsymbol{A}_f$ is the design matrix (Jacobi matrix) of the spatial transformation parameterized by $\boldsymbol{H}_0$. $\boldsymbol{l}_P$ and $\boldsymbol{l}_{Q\text{in}P}$ have been computed based on the initial solution of the parameters and the observations according to

$$\begin{bmatrix} \boldsymbol{l}_P \\ \boldsymbol{l}_{Q\text{in}P} \end{bmatrix} = f(\boldsymbol{X}_{P,0}, \boldsymbol{H}_0) - \begin{bmatrix} \boldsymbol{L}_P \\ \boldsymbol{L}_{Q\text{in}P} \end{bmatrix}. \tag{12.3}$$

The $\boldsymbol{v}$ are the corrections to be applied to these shortened observations to fulfill the functional model. The first line of Equation (12.2) is a simple identity transformation since the point cloud to be estimated is in coordinate system P. The second line represents the yet to be transformed point cloud Q, it also involves the design matrix $\boldsymbol{A}_{f(H_0)}$.

The stochastic model is described by

$$\boldsymbol{\Sigma}_{\boldsymbol{L}} = \begin{bmatrix} \boldsymbol{\Sigma}_P & 0 \\ 0 & \boldsymbol{\Sigma}_{Q\text{in}P} \end{bmatrix}. \tag{12.4}$$

Here, $\boldsymbol{\Sigma}_P$ is the a priori covariance matrix associated with $\boldsymbol{L}_P$. $\boldsymbol{\Sigma}_{Q\text{in}P}$ is the a priori covariance associated with $\boldsymbol{L}_Q$, but propagated to the target coordinate system A based on $H_A, 0^B$ using forward error propagation according to 10.3.3:

$$\boldsymbol{\Sigma}_{Q\text{in}P} = \boldsymbol{A}_f \boldsymbol{\Sigma}_Q \boldsymbol{A}_f^T \tag{12.5}$$

The NLLSQ process described in 10.3.3 computes estimates $\hat{\boldsymbol{X}}_P$ and $\hat{\boldsymbol{H}}$ for the unknown parameters $\bar{\boldsymbol{X}}_P$ and $\bar{\boldsymbol{H}}$ such that the necessary corrections $\boldsymbol{v}_P$ and $\boldsymbol{v}_{Q\text{in}P}$ are

minimized. Obtained estimates $\hat{\boldsymbol{X}}_P$ and $\hat{\boldsymbol{H}}$ become the initial solution $\boldsymbol{X}_{P,0}$ and $\boldsymbol{H}_0$ for the next iteration until convergence. In particular, the covariance matrix $\boldsymbol{\Sigma}_{Q\text{in}P}$ has to be adapted in each iteration. Depending on the weighting of $\boldsymbol{\Sigma}_P$ and $\boldsymbol{\Sigma}_{Q\text{in}P}$ in $\boldsymbol{\Sigma}_L$, the estimated point cloud $\hat{\boldsymbol{X}}_P$ is closer to $\boldsymbol{L}_P$ or $\boldsymbol{L}_{Q\text{in}P}$. Each estimated point $\hat{\boldsymbol{p}}$ will end up on a straight line between $\boldsymbol{p}$ and $\boldsymbol{Hq}$, all other choices of $\hat{\boldsymbol{p}}$ would result in a greater norm of $\boldsymbol{v}$, regardless the chosen a priori covariance matrix $\boldsymbol{\Sigma}_L$.

12.3.2. A Linear Approximation

Next, a linear method is described for the estimation of uncertainties from point-based registration. The derivation assumes isotropic error in coordinate frames P and Q. It relates the residual error (cf. 10.1.3) with the concepts of MLE estimation (cf. 10.3.1). It helps to develop a method for the estimation of uncertainties even without having ground truth data.

Residual Error Revisited

Assuming error in coordinate frame Q only, *residual error* and *estimation error* per measurement are defined as

$$\begin{aligned} \epsilon_{\text{res}} &= \sqrt{\frac{1}{dn}\sum_{i=1}^{n}\|\boldsymbol{q}_i - \hat{\boldsymbol{q}}_i\|^2} \\ \epsilon_{\text{est}} &= \sqrt{\frac{1}{dn}\sum_{i=1}^{n}\|\bar{\boldsymbol{q}}_i - \hat{\boldsymbol{q}}_i\|^2}, \end{aligned} \tag{12.6}$$

where d is the dimensionality of the problem, e.g., 3 for 3D points, $\|\cdot\|$ is the Euclidean distance and $\hat{\boldsymbol{q}}_i$ is the maximum likelihood estimate (MLE) of the unknown true point $\bar{\boldsymbol{q}}_i$. This is a just a reformulation of Equation (10.14).

For the sake of completeness, a related, more general error norm shall be mentioned. The *Mahalanobis distance* weights the measurements according to their covariance matrix $\boldsymbol{\Sigma}$. Using the Mahalanobis distance, however, one sacrifices the nice property of a direct, metric interpretability of ϵ_{res} and ϵ_{est}.

$$\begin{aligned} \epsilon_{\text{res},\Sigma} &= \sqrt{\frac{1}{dn}\sum_{i=1}^{n}(\boldsymbol{q}_i - \hat{\boldsymbol{q}}_i)\boldsymbol{\Sigma}(\boldsymbol{q}_i - \hat{\boldsymbol{q}}_i)} \\ \epsilon_{\text{est},\Sigma} &= \sqrt{\frac{1}{dn}\sum_{i=1}^{n}(\bar{\boldsymbol{q}}_i - \hat{\boldsymbol{q}}_i)\boldsymbol{\Sigma}(\bar{\boldsymbol{q}}_i - \hat{\boldsymbol{q}}_i)}, \end{aligned} \tag{12.7}$$

For the general case of errors in both measurements, residual and estimation errors

become

$$\epsilon_{\text{joint res}} = \sqrt{\frac{1}{2dn}\sum_{i=1}^{n}(\|\boldsymbol{p}_i - \hat{\boldsymbol{p}}_i\|^2 + \|\boldsymbol{q}_i - \hat{\boldsymbol{q}}_i\|^2)}$$

$$\epsilon_{\text{joint est}} = \sqrt{\frac{1}{2dn}\sum_{i=1}^{n}(\|\bar{\boldsymbol{p}}_i - \hat{\boldsymbol{p}}_i\|^2 + \|\bar{\boldsymbol{q}}_i - \hat{\boldsymbol{q}}_i\|^2)}. \tag{12.8}$$

The factor 2 in the denomenator reflects the fact that errors arise in both coordinate frames now. Equations (12.6) and (12.8) compute a per single coordinate measure of the standard deviation of measured minus estimated (residual) and estimated minus unknown true points. In other words, the uncertainty is balanced equally between both modalities.

Residual error is also known as *geometric error*. Please note that it is different from *symmetric transfer error* [Hart 00].

$$\epsilon_{\text{sym}} = \sqrt{\frac{1}{2dn}\sum_{i=1}^{n}(\|\boldsymbol{H}^{-1}\boldsymbol{p}_i - \boldsymbol{q}_i\|^2 + \|\boldsymbol{p}_i - \boldsymbol{H}\boldsymbol{q}_i\|^2)} \tag{12.9}$$

Though easier to compute since no point estimates $\hat{\boldsymbol{p}}$ are needed, the latter does not directly relate to the standard deviations of the two sensors and is therefore not very helpful for our purposes.

Parameter Estimation

In case of error in one frame only, the true points $\bar{\boldsymbol{p}}_a$ are known and $\boldsymbol{P}$ consists only of the parametrization of the homography H to be estimated, e.g., 3 translational and 3 rotational parameters for 3D-3D pose estimation. In case of errors in both coordinate frames, $\boldsymbol{P}$ also contains dn entries for $\hat{\boldsymbol{p}}_a$, thus altogether $6+dn$ entries. $\boldsymbol{L}$ consists of all measurements, either in frame Q only, or both, stacked together. It therefore contains $N = dn$ or $N = 2dn$ entries.

Hartley and Zisserman [Hart 00] derived formulas for the expectation values of ϵ_{res} and ϵ_{est} that depend on the assumed standard deviation in measurement space, the number of measurements and the degrees of freedom of the problem.

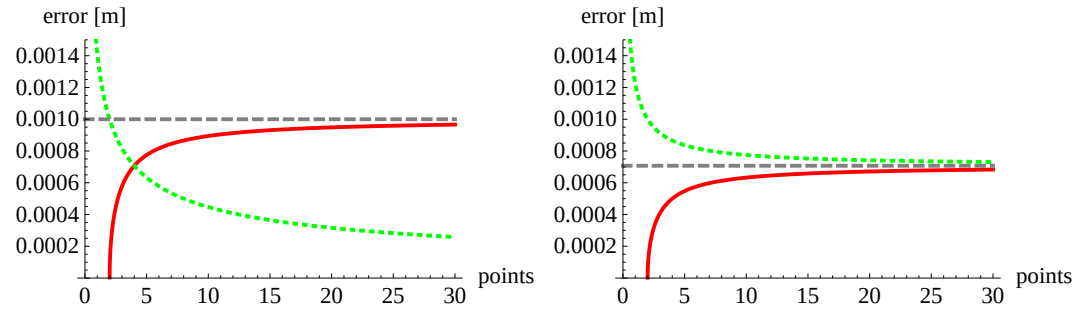
$$\epsilon_{\text{res}} = \sqrt{\mathbf{E}[\frac{\|\hat{\boldsymbol{L}} - \boldsymbol{L}\|^2}{N}]} = \sqrt{\frac{\sigma^2(N-M)}{N}}) = \sigma\sqrt{(1 - \frac{M}{N})} \tag{12.10}$$

$$\epsilon_{\text{est}} = \sqrt{\mathbf{E}[\frac{\|\hat{\boldsymbol{L}} - \bar{\boldsymbol{L}}\|^2}{N}]} = \sqrt{\sigma^2\frac{M}{N}} = \sigma\sqrt{\frac{M}{N}} \tag{12.11}$$

The derivation is based on the total variance associated with the measurements $\boldsymbol{L}$ which is the trace of the covariance matrix. In the case of isotropic error, total variance

thus becomes $N\sigma^2$. According to Equation 10.24, estimation error is the deviation of estimated measurements $\hat{\boldsymbol{L}}$ from the true measurements $\bar{\boldsymbol{L}}$, which is given by a projection of the Gaussian distribution of $\boldsymbol{L}$ on the submanifold $\mathbf{S}$ of $\mathbf{R}^N$ (cf. Figure 10.3).

Errors in directions orthogonal to $\mathbf{S}$ become $\mathbf{0}$ by this projection and the remaining total variance of $\hat{\boldsymbol{L}} \in \mathbf{S}$ becomes $M\sigma^2$. Note that M is the dimension of $\mathbf{S}$. The derivation for ϵ_{res} is similar. Figures 12.1(a) and 12.1(b) show the behavior of ϵ_{res} and ϵ_{est} assuming $\sigma = 1$ mm for the 3D-3D pose estimation problem as a function of the number of point correspondences for the cases of error in frame Q only and errors in both frames.



(a) Error in one coordinate frame only

(b) Error in both coordiante frames

Figure 12.1.: Behavior of residual error ϵ_{res} (solid, red) and estimation error ϵ_{est} (dotted, green) in 3D-3D pose estimation as a function of the number of point correspondences. A standard deviation of $\sigma = 0.001$ m per measured coordinate is assumed.

In both cases, residual error increases quickly after the minimum number of measurements has been taken. At the same time, estimation error decreases quickly. For errors in frame Q only, residual error for single coordinates (q_x, q_y, q_z) of the points $\boldsymbol{q}_i$ converges against the assumed standard deviation of $\sigma = 1$ mm whereas estimation error for these coordinates converges against 0. This becomes clear from the fact that both $\hat{\boldsymbol{L}}$ and $\bar{\boldsymbol{L}}$ are elements of the only 6-dimensional submanifold $\mathbf{S}$. With more and more measurements available, the orthogonal projection referred to above assigns more and more of the total variance of the measurements to directions that are orthogonal to $\mathbf{S}$. In case of errors in both frames, however, the dimension of $\mathbf{S}$ increases with the number of measurements. The common asymptotic value of $\epsilon_{\text{joint res}}$ and $\epsilon_{\text{joint est}}$ is $\sigma/\sqrt{2}$ because the true coordinates are not known any longer but are estimated from two measurements each, one in frame P and another in Q, related by the estimated homography $\boldsymbol{H}$. And with two measurements for each coordinate, the variance of the estimate (mean value) is $1/2$ (cf. Equation (10.3)) of the variance of the single measurement. This directly corresponds to the Pythagorean equality in Equation 10.24 stated above. For errors in frame Q only, $\|\bar{\boldsymbol{L}} - \hat{\boldsymbol{L}}\|^2$ converges to 0 and $\|\boldsymbol{L} - \hat{\boldsymbol{L}}\|^2$ converges to the real error. For errors in both frames, $\|\bar{\boldsymbol{L}} - \hat{\boldsymbol{L}}\|^2$ never becomes 0.

Using the Mahalanobis distance, a more general form of Equation (12.10) can be derived. For this, the Euclidean distance is just replaced by the Mahalanobis distance

and the σ is removed.

$$\epsilon_{\text{res}} = \sqrt{\mathbf{E}[\frac{\|\hat{\boldsymbol{L}} - \boldsymbol{L}\|_{\Sigma}^2}{N}]} = \sqrt{\frac{(N-M)}{N}}) = \sqrt{(1 - \frac{M}{N})} \tag{12.12}$$

$$\epsilon_{\text{est}} = \sqrt{\mathbf{E}[\frac{\|\hat{\boldsymbol{L}} - \bar{\boldsymbol{L}}\|_{\Sigma}^2}{N}]} = \sqrt{\frac{M}{N}} \tag{12.13}$$

The computed norms can be plotted similarly to Figure 12.1, though without the metric interpretability. Still, such plots could provide valuable information to the user during measurement acquisition for registration procedures, see also 14.1.

Estimation of Corrected Points

In the case with error in frame Q only, estimated points $\hat{\boldsymbol{q}}$ are trivially obtained from the true points $\bar{\boldsymbol{p}}$ by $\hat{\boldsymbol{q}}_i = \hat{H}^{-1}\bar{\boldsymbol{p}}_i$, using the transformation $\hat{H}$ estimated before (see 12.3.2). In the general case, estimates $\hat{\boldsymbol{p}}$ and $\hat{\boldsymbol{q}}$ can not be obtained as easily. For this, we use a modification of an error function proposed by Sampson for conic fitting [Hart 00] [Samp 82]. It gives us the MLE of the points $\hat{\boldsymbol{p}}$ and $\hat{\boldsymbol{q}}$ which allows us to compute the residual error using Equation (12.8).

The Sampson approximation is normally used for the iterative estimation of both the unknown parameters of the function to be estimated and corrections to the noisy measurements in an NLLSQ approach. Since we already have the MLE estimate of the parameters from the closed-form solution, Sampson's formula gives us the estimated corrections of the noisy measurements in one step.

In the general case of a projective transformation in 3D, Equation (12.1) can be written as

$$k\begin{pmatrix} p_x \\ p_y \\ p_z \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} & h_{14} \\ h_{21} & h_{22} & h_{23} & h_{24} \\ h_{31} & h_{32} & h_{33} & h_{34} \\ h_{41} & h_{42} & h_{43} & h_{44} \end{pmatrix} \begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix} = \begin{pmatrix} \boldsymbol{h}_1^T \\ \boldsymbol{h}_2^T \\ \boldsymbol{h}_3^T \\ \boldsymbol{h}_4^T \end{pmatrix} \begin{pmatrix} q_x \\ q_y \\ q_z \\ 1 \end{pmatrix}. \tag{12.14}$$

k is some constant that allows one to have 1 in the forth row of the homogeneous vectors on both sides.[4] Setting $k = \boldsymbol{h}_4^T\boldsymbol{q}$ yields

$$\begin{pmatrix} \boldsymbol{h}_4^T\boldsymbol{q}p_x \\ \boldsymbol{h}_4^T\boldsymbol{q}p_y \\ \boldsymbol{h}_4^T\boldsymbol{q}p_z \end{pmatrix} = \begin{pmatrix} \boldsymbol{h}_1^T\boldsymbol{q} \\ \boldsymbol{h}_2^T\boldsymbol{q} \\ \boldsymbol{h}_3^T\boldsymbol{q} \end{pmatrix}. \tag{12.15}$$

[4] Note that $k = 1$ for affine transformations for which $h_{41} = h_{42} = h_{43} = 0$ and $H_{44} = 1$. In particular, pose transformations fall in this category.

Sorting this equation system by the entries of $\boldsymbol{H}$ yields

$$\begin{pmatrix} \boldsymbol{q}^T & \boldsymbol{0}^T & \boldsymbol{0}^T & -p_x\boldsymbol{q}^T \\ \boldsymbol{0}^T & \boldsymbol{q}^T & \boldsymbol{0}^T & -p_y\boldsymbol{q}^T \\ \boldsymbol{0}^T & \boldsymbol{0}^T & \boldsymbol{q}^T & -p_z\boldsymbol{q}^T \end{pmatrix} \begin{pmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \boldsymbol{h}_3 \\ \boldsymbol{h}_4 \end{pmatrix} = \boldsymbol{A}\boldsymbol{h} = 0. \tag{12.16}$$

In the overdetermined case, Equation (12.16) $\neq 0$ but it yields the algebraic error vector

$$\epsilon_{alg} = \boldsymbol{A}_i\boldsymbol{h} = \begin{pmatrix} \boldsymbol{A}_1 \\ \vdots \\ \boldsymbol{A}_n \end{pmatrix} \boldsymbol{h} \tag{12.17}$$

for each point correspondence $\{p_i \leftrightarrow q_i\}$. It does not have a geometrically meaningful interpretation except for the case when it is 0. In this case, the point correspondence fits perfeclty to the given homography $\boldsymbol{H}$ and geometric error becomes 0, too.

We introduce the 6-vector $\boldsymbol{L}_i = (\boldsymbol{p}_i^T, \boldsymbol{q}_i^T)^T$ and the $6n$-vector $\boldsymbol{L} = (\boldsymbol{L}_1^T, \ldots, \boldsymbol{L}_n^T)^T$. The algebraic error vector can be approximated by a Taylor expansion around the given point measurements.

$$\epsilon_{\text{alg}}(\boldsymbol{L}_0 + \boldsymbol{\Delta}_L) \approx \epsilon_{alg}(\boldsymbol{L}) + \boldsymbol{J}_{\epsilon_{\text{alg}}(L_0)}\boldsymbol{\Delta}_L \tag{12.18}$$

with $\boldsymbol{\Delta}_L = (\hat{\boldsymbol{L}} - \boldsymbol{L})$. Given a certain homography $\boldsymbol{H}$, the goal therefore is to compute points $\hat{\boldsymbol{p}}$ closest to our measurements $\boldsymbol{p}$ that also fulfill Equation (12.16). This is a minimization problem with constraints and can be solved using Lagrangian multipliers. A derivation is given in [Hart 00]. To solve the problem, the Jacobian

$$\boldsymbol{J}_{\epsilon_{\text{alg}}(L_0)} = \frac{\delta}{\delta \boldsymbol{L}}\epsilon_{alg}|_{L_0} = \frac{\delta}{\delta L}\boldsymbol{A}\boldsymbol{h}|_{L_0} \tag{12.19}$$

of ϵ_{alg} w.r.t. the measurements $\boldsymbol{L}$ has to be computed. It has block-diagonal structure, with

$$\boldsymbol{J} = \begin{pmatrix} \boldsymbol{J}_1 & 0 & \ldots & 0 \\ 0 & \boldsymbol{J}_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & \boldsymbol{J}_n \end{pmatrix} \tag{12.20}$$

and the 3×6 matrices $\boldsymbol{J}_i$ according to Equation (12.21). Please note that the first-order approximation in term of $\boldsymbol{J}$ is exact in the case of an affine transformation. This becomes clear by setting $h_{41} = h_{42} = h_{43} = 0$ and $h_{44} = 1$.

$$\boldsymbol{J}_i = \begin{pmatrix} h_{11} - p_x h_{41} & h_{12} - p_x h_{42} & h_{13} - p_x h_{43} & -\boldsymbol{h}_4^T\boldsymbol{q} & 0 & 0 \\ h_{21} - p_y h_{41} & h_{22} - p_y h_{42} & h_{23} - p_y h_{43} & 0 & -\boldsymbol{h}_4^T\boldsymbol{q} & 0 \\ h_{31} - p_z h_{41} & h_{32} - p_z h_{42} & h_{33} - p_z h_{43} & 0 & 0 & -\boldsymbol{h}_4^T\boldsymbol{q} \end{pmatrix}. \tag{12.21}$$

We obtain corrections for the measurements $\boldsymbol{P}$ by

$$\boldsymbol{\Delta}_P = -\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{J}^T)^{-1}\epsilon_{alg}. \tag{12.22}$$

In the general case, the covariance matrix would have to be incorporated:

$$\boldsymbol{\Delta}_P = -\boldsymbol{\Sigma}_L\boldsymbol{J}^T(\boldsymbol{J}\boldsymbol{\Sigma}_L\boldsymbol{J}^T)^{-1}\epsilon_{alg}. \tag{12.23}$$

This allows us to compute the MLE estimates of the measured coordinates $\hat{\boldsymbol{P}} = \boldsymbol{P} + \boldsymbol{\Delta}_P$. Since the Sampson approximation to NLLSQ is correct for our affine case, so are the computed corrections, provided that H indeed was the MLE estimate. This requirement is met e.g., by the absolute orientation algorithm used for 3D-3D pose estimation [Horn 87]. Having this, geometric error according to Equation (12.8) can be computed. The derivation assumed measurements in 3D space. It is similar for 2D measurements.

Estimation of Variance Components

In the following, we use again the simplified assumption of an isotropic error model. As already stated in 12.3.1, the joint residual error remains constant, regardless how we distribute the uncertainties to the involved modalities. The estimated (corrected) point just moves on a straight line between the two erroneous measurements since all other estimates would result in a higher residual, regardless which a priori variance or covariance is assumed. It might be wrong but the estimated joint residual error remains constant. Assuming independent and isotropic errors in each of the two frames, (12.10) can be converted to

$$\begin{aligned}\epsilon_{\text{joint res}} &= \sqrt{\frac{\sigma^2_{\text{joint}}(N-M)}{N}} \approx \sqrt{\frac{\sigma^2_{\text{joint}}}{2}} = \sqrt{\frac{0.5(\sigma_p^2+\sigma_q^2)}{2}} \\ 2\epsilon_{\text{joint res}} &= \sqrt{\sigma^2_{\text{joint}}} = \sqrt{\sigma_p^2+\sigma_q^2}.\end{aligned} \tag{12.24}$$

The true joint uncertainty σ_{joint} is composed of the unknown individual true uncertainties σ_p and σ_q. σ_{joint} can be computed from the computed residual $\epsilon_{\text{joint res}}$. This also explains again the nature of the factor $\sqrt{2}$ appearing in the denominator of Equation 12.8 and the lower limit of the residual error in Figure 12.1(b), as compared to Figure 12.1(a). Note that the assumption of isotropic error allows us to neglect the otherwise necessary transformation of the covariance according to Equation (12.5) since by a rigid transformation, the norm of the RMS is not changed. Therefore, the RMS values of both modalities can simply be added.

The final goal is to estimate σ_p and σ_q. From a single set of point correspondences, this is not possible. At least three independent sets of correspondences are needed in order to obtain unique results by solving the following equation system:

$$\begin{aligned}\sigma_a^2 + \sigma_b^2 &= \sigma^2_{\text{joint}_{ab}} \\ \sigma_b^2 + \sigma_c^2 &= \sigma^2_{\text{joint}_{bc}} \\ \sigma_a^2 + \sigma_c^2 &= \sigma^2_{\text{joint}_{ac}}\end{aligned} \tag{12.25}$$

As explained above, the MLE estimate $\hat{\boldsymbol{p}}$ moves freely between $\boldsymbol{p}$ and $\boldsymbol{Hq}$, depending on the choice of variance or covariance in the the coordinate systems P and Q. The joint residual error ϵ_{res} is not influenced by this a priori assumption of the uncertainties. The assumption of uncertainty in one frame only, say Q, is just an extreme case. Instead of (12.24), we then obtain

$$\epsilon_{\text{res}} = \sqrt{\frac{\sigma^2_{\text{joint}}(N-M)}{N}}) \approx \sqrt{\sigma^2} = \sqrt{\frac{(\sigma_p^2 + \sigma_q^2)}{2}}. \tag{12.26}$$

The system of equation (12.25) remains unchanged. This simplification renders the explict computation of corrected points $\hat{\boldsymbol{p}}_i$ according to 12.3.2 useless. It can be omitted if the corrected points are not explicitly needed, simplifying the whole process.

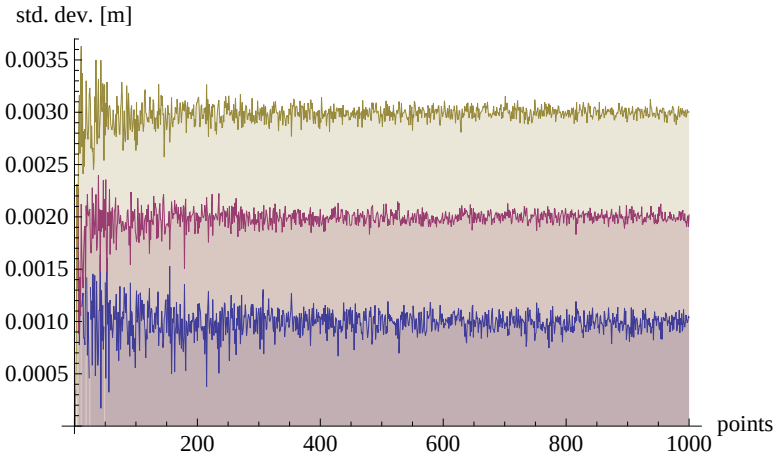
Example: Simulation Experiment

A simulation based on synthetic data shows the principal feasibility of our approach. This is supported by two experiments based on real sensor data. Three corresponding sets of 3D points are given in coordinate frames P, Q, and R. They are chosen to be uniformly distributed on a unit sphere. The three frames are related by an identity transform, i.e. no translation, no rotation. Then, isotropic Gaussian noise is applied to all points. Different standard deviations σ_p, σ_q and σ_r are used for the three point clouds.

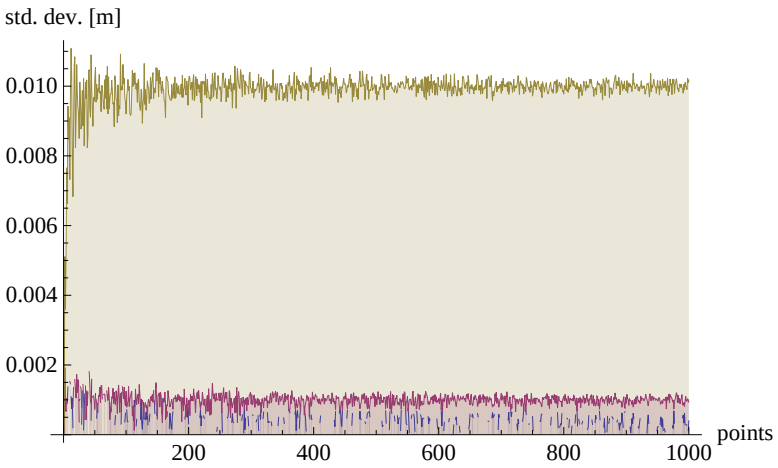
From this data, the three rigid transformations $\boldsymbol{H}_{AB}$, $\boldsymbol{H}_{BC}$, and $\boldsymbol{H}_{AC}$ are computed using Horn's abosolute orientation algorithm [Horn 87] applied to the noisy data. Then, point corrections $\boldsymbol{\Delta}_{ab}$, $\boldsymbol{\Delta}_{bc}$, and $\boldsymbol{\Delta}_{ac}$ are computed using (12.22). Using (12.24), they result in the joint residual errors $\hat{\sigma}_{ab}$, $\hat{\sigma}_{bc}$, and $\hat{\sigma}_{ac}$ that are used to solve for the desired standard deviations $\hat{\sigma}_a$, $\hat{\sigma}_b$, and $\hat{\sigma}_c$ according to Equation (12.25).

The experiment is conducted twice, once with σ_i being close together ($\sigma_a = 1$ mm, $\sigma_b = 2$ mm, and $\sigma_c = 3$ mm) and once with σ_i differing by an order of magnitude ($\sigma_a = 0.1$ mm, $\sigma_b = 1$ mm, and $\sigma_c = 10$ mm). In each experiment, the size of the point sets is increased one by one, starting with the minimum number of three points, stopping at 1000 points per set. For each number of points, a new sample is drawn from the uniform distribution, and a new error is added to each set, drawn from the isotropic Gaussian distribution.

The results are summarized in Figure 12.2. In both cases, with increasing number of points, the estimated standard deviations $\hat{\sigma}_i$ become more and more stable and approach the true standard deviations $\bar{\sigma}_i$ chosen before. See also 11.3. Quantities lower than 50-70 don't seem to yield stable results. Note that the (12.25) is not always solvable with real numbers, especially for few point correspondences and σ being close to 0. The corresponding measurements are missing in the plots, especially for σ_a in Figure 12.2(b). As a workaround, one could compute the absolute value of the variance before extracting the root to compute the standard deviation.



(a) $\sigma_a = 1$ mm, $\sigma_b = 2$ mm, and $\sigma_c = 3$ mm



(b) $\sigma_a = 0.1$ mm, $\sigma_b = 1$ mm, and $\sigma_c = 10$ mm

Figure 12.2.: Behavior of estimated standard deviation as a function of the number of point correspondences

12.3.3. Practical Considerations

In the case where ground truth data for comparison is available, both, the Helmert transformation and the linear approximation can be used to assess the unknown uncertainties of a tracking device. This can be achieved at several levels of abstraction. Position measurements can be generated by a single fiducial, as shown in Figure 12.3. Alternatively, the tip of a pointing device (see Figure 1.6(b)) can be used as well.

With a pointing device the correspondence property between measurements of different modalities can be established easily by repeatedly probing the same physical points. It might therefore be easier to handle, provided that suitable points are available for probing. With respect to a generally useful elementary specification of uncertainty, the

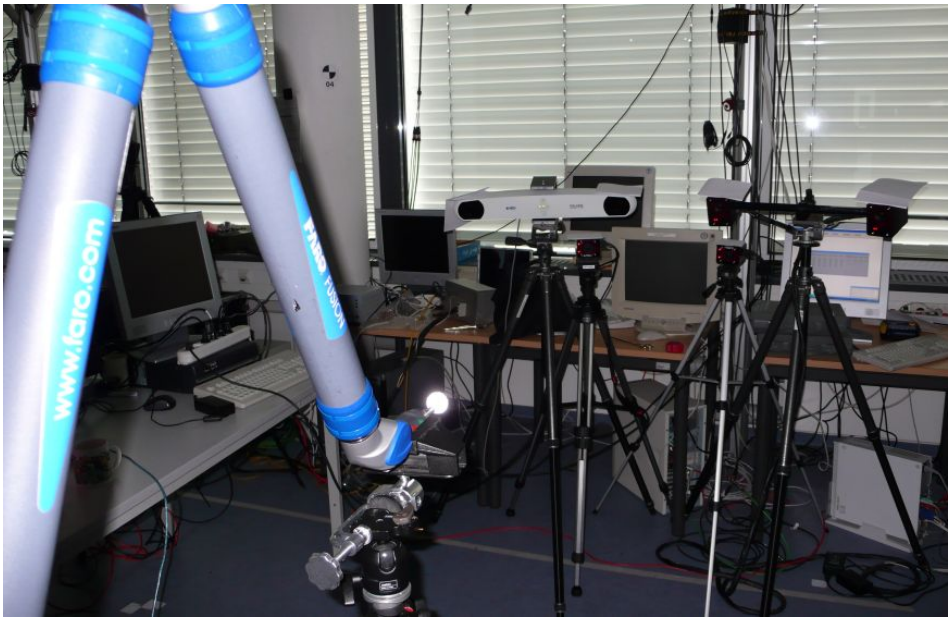


Figure 12.3.: Three-sensor evaluation setup

former method should be preferred though. The results can then be applied to arbitrary marker layouts.

When capturing position measurements synchronously, irrespective of whether a single fiducial or a pointing device is used, a proper time-synchronization is indispensable. This can be achieved in general by a proper synchronization of the associated timestamps, cf. 5.4.3. Also, a high-precision robot could be used to keep constant the position for the time span that is needed for a safe acquisition of measurements with all involved devices (cf. 12.4), superseding a proper time-synchronization. Alternatively, special plates or 3D structures can be used that allow for reproducible fiducial placement. High-precision variants of the latter are typically used in close-range photogrammetry, see for example [Luhm 00a].

In the experiment depicted in Figure 12.3, three different IR stereo tracking systems were compared against ground truth measurements of a *FARO* arm. The tracking systems had a baseline of 50 cm each, but considerably different focal lengths. A retroreflective fiducial was mounted as the tip of the mechanical tracking arm. Obviously, it can be tracked by the IR tracking systems simultaneously, resulting in measurements of the center-of-gravity of the sphere. Furthermore, the ball was calibrated as the tip of the arm, instead of the default 3 or 6 mm ceramic measurement sphere. In this way, *corresponding measurements* could be recorded simultaneously by all four devices. For this, the arm was moved to different positions approximately aligned with a $4 \times 4 \times 4$ grid, resulting in 64 point measurements. Point measurements were acquired discretely and manually, point by point; at each position, the arm was fixed by a screw clamp to circumvent the problem of not properly synchronized timestamps.

The three point clouds obtained from IR tracking can be matched with the ground truth point cloud, yielding a residual error according to Equation (12.6). Alternatively the Helmert transformation could also be used. The former method was applied to the measurements from the described experimental setup, resulting in the uncertainties given in Table 12.1.

Table 12.1.: RMS error [mm] from point-based registration of three different IR tracking systems using ground truth measurements. A 6 DoF rigid body transformation and a 12 DoF affine transformation are used for registration.

		RMS error	
IR tracking system	Focal length [mm]	6 DoF matching	12 DoF matching
NDI Polaris	6	0.4	0.4
A.R.T. SmARTrack	3.5	2.3	1.0
A.R.T. TrackPack	2	12.4	5.2

The low RMS error for the *NDI Polaris* shows that the calibration of the retro-reflective fiducial as the tip of the *FARO* arm is quite accurate, otherwise the error would have been higher. The comparison is somewhat unfair, due to the quite different focal lengths. The huge tracking volumes of the *A.R.T.* systems had to be restricted to the much smaller volume of the *NDI Polaris* for the acquisition of identical point clouds. Unlike the *NDI Polaris*, the *A.R.T.* systems have to be calibrated on-site before usage. For this, a wand with a defined length and a fiducial at each of its ends is used. Since we suspected scaling problems caused by this calibration procedure, we also computed a 12 DoF affine transform (non-isotropic scaling along arbitrary axes) to match the point clouds, using the DLT method [Hart 00]. For the A.R.T. systems, the error dropped by more than 50 percent, suggesting that the prescribed calibration procedure with the wand is critical and should be optimized. For the *NDI Polaris*, the 12 DoF transformation did not make a difference, suggesting that its factory calibration is quite optimal, at least globally (considering the complete tracking volume).

Unlike in the described test setup, also tracking modalities based on distinct measurement principles can be compared. In this case, the different fiducials can be mounted rigidly to a common target that can be moved in space. The individual fiducials will have an offset in between, but if the offset can be kept constant by the experimental setup, the point clouds can still be matched. This approach is further detailed below in 12.4.

Now, the assessment of uncertainties *without* ground truth data is discussed. Having at least three corresponding point clouds, this can be achieved by solving Equation (12.25). Four point clouds are available from the experiment described above, considering the *FARO* measurements not any longer as a ground truth. Table 12.2 lists the results for the four possible combinations of three point clouds, respectively.

The last two combinations correspond quite nicely to the results in Table 12.1 whereas the first two combinations yield rather different results. Combination two is obviously heavily corrupted since the estimated uncertainty for the *FARO* arm is far too high.

Table 12.2.: RMS error [mm] from point-based registration of four different tracking modalities without assuming ground truth data. Four different combinations of three corresponding point-clouds are given.

Combination	*FARO* arm	*NDI Polaris*	*A.R.T. SmARTrack*	*A.R.T. TrackPack*
1	-	4.8	5.3	13.4
2	4.8	-	5.4	13.3
3	0.8	0.9	-	12.4
4	0.5	0.3	2.2	-

This evaluation reveals an interesting aspect. Solving Equation (12.25) seems to yield reasonable results if the majority of the involved systems are globally scaled correctly. The method fails if two of the involved systems suffer from a globally wrong scaling as shown in Table 12.1. The method then is no longer capable of re-establishing the true scaling, resulting in wrong error estimates.

A further, interesting possibility is briefly sketched in this context. It would allow for the assessment of uncertainty for a single tracking modality only, which is more desirable than the assessment of three systems in parallel. The difficulty in practice is to measure the same point cloud several times and with different orientations inside the tracking volume to "reveal" the systematic effects. A physical setup is needed for a reproducible fiducial placement. It should be large enough to cover a considerable part of the whole volume. From at least two point clouds (the more, the better) captured this way, the uncertainty could then be estimated from the joint residual (cf. Equation (12.8)) by splitting it up in equal parts to the involved poses of the physical setup (cf. Equation (12.24)). Unlike the methods to provide ground truth discussed above, the geometry of this physical setup can be unknown, it just has to be rigid. Therefore, its construction might be simpler and cheaper. This method would not reveal isotropic scaling effects, due to the unknown geometry of the setup. However, it could reveal the non-isotropic scaling effects that were not treated correctly for the first two combinations in Table 12.2. An unknown distance should remain constant for different orientations in the volume.

A similar situation arises in case that we already know the uncertainty of one device, σ_a, maybe from a previous inspection. We can then allocate the remainder of the error to the σ_b to be estimated, again using Equations (12.8) and (12.24). It is important, however, to use the system under equal operating conditions such that the a priori accuracy is really valid.

12.4. Example: Specification of Elementary Uncertainty for the Airplane Cabin

In the following, a reasonable elementary specification of uncertainty is provided for the exemplary airplane cabin scenario (see also 1.3.2 and 7.1.2) [Keit 10b]. It represents the basis for verification and validation of this application in the target environment in 13.2.

However, it is relevant also for other scenarios based on the same tracking equipment and can be directly reused.

The validity of the elementary specification of uncertainty is supported by a simulation experiment of the indirect tracking scenario, following the concepts described above in 11. The consistency of simulation and empirical data is shown based on extensive measurements in a controlled setup.

12.4.1. Setup

The IR real-time tracking system (see foreground of Figure 12.4) consists of three line cameras, mounted rigidly in a single housing. Extrinsic and intrinsic parameters have already been estimated by the vendor before delivery. It provides 6DoF poses of the probe and reference targets. Nevertheless it is not feasible to specify the error at that abstraction level since we use custom marker layouts. The error of 6DoF pose tracking highly depends on layout and dimension and therefore cannot be stated in a general way by the vendor, cf. 13.1.5. Applying the error to the 3D positions of the individual LEDs is more appropriate. This quantity can be estimated in a general way, due to the rigid arrangement of cameras. The vendor specifies an RMS uncertainty for the position of a tracked LED that depends on the depth with respect to the cameras [Opto 11]. The uncertainty in the horizontal/vertical/depth directions is 0.1/0.1/0.15 mm at 2 m depth, 0.15/0.15/0.25 mm at 4 m depth, and 0.25/0.25/0.45 mm at 6 m depth.

An uncertainty specification on the sensor's image plane using 2D noise would also be possible. However, internal details about the intrinsic and extrinsic camera parameters would be required for this. Therefore, we stick to the above uncertainty specification in 3D.

In the experiments described next, we first try to reproduce this elementary specification of uncertainty. Then, a simulation experiment is conducted to prove the principal suitability of this elementary specification of uncertainty to obtain reasonable results.

To evaluate the consistency of the simulation, we rely on distance measurements as proposed by international metrological standards [VDI 02] [ASME 06], as well as point-based registration and adjustment theory [Niem 08]. We designed a test target (small picture in Figure 12.4) that consists of a tracking target for the online metrological system (black LED target) and a laser reflection target (silver sphere) for the offline metrological system, a high-precision laser tracker[5]. Both tracking systems are located in front of the target to be able to identify the reference target's positions.

A high precision coordinate measurement machine[6] (CMM) is used as a reference. It allows us to move the test target to an arbitrary specified location within a volume of 6x4x2.5 m^3, with an RMS uncertainty of 0.01 mm + 0.014 mm/m its coordinate origin. The laser scanner is specified with 0.049 mm RMS at 10 m distance.

We programmed the CMM to scan the entire measurement volume as shown by the small spheres in Figure 12.5, stopping at each position for 3 seconds. The distance between the grid positions was 20 cm in each direction. We furthermore scanned three

[5] *FARO Ion* [Faro 11b]
[6] *DEA Lambda*

Figure 12.4.: Laboratory setup for the derivation of a consistent elementary specification of uncertainty. A high-precision CMM and a laser tracker provide ground truth measurements to estimate the uncertainty of 3D position tracking of a single IR LED. (Courtesy of *EADS Innovation Works* [Keit 10b])

orthogonal lines along the coordinate axes of the real-time tracker, at a regular distance of 1 cm. The frustum of pyramid in Figure 12.5 indicates the tracking volume as specified by the vendor.

Pausing at every position for 3 seconds allowed us to synchronize programmatically the three systems afterwards, as well as to assess the precision (affected by noise) and accuracy (affected by systematic errors) of LED tracking. The real time system runs at 30 Hz and theoretically provides 90 samples for each measured position. However, the window of 3 seconds was clipped at its beginning to remove inertial effects coming from the sudden stop of the movement of the CMM. For each measured position, there remain 60 valid samples.

12.4.2. Noise

The first experiment estimates the influence of random noise on LED tracking. The reason to estimate noise separately is the hope of lowering measurement errors by averaging

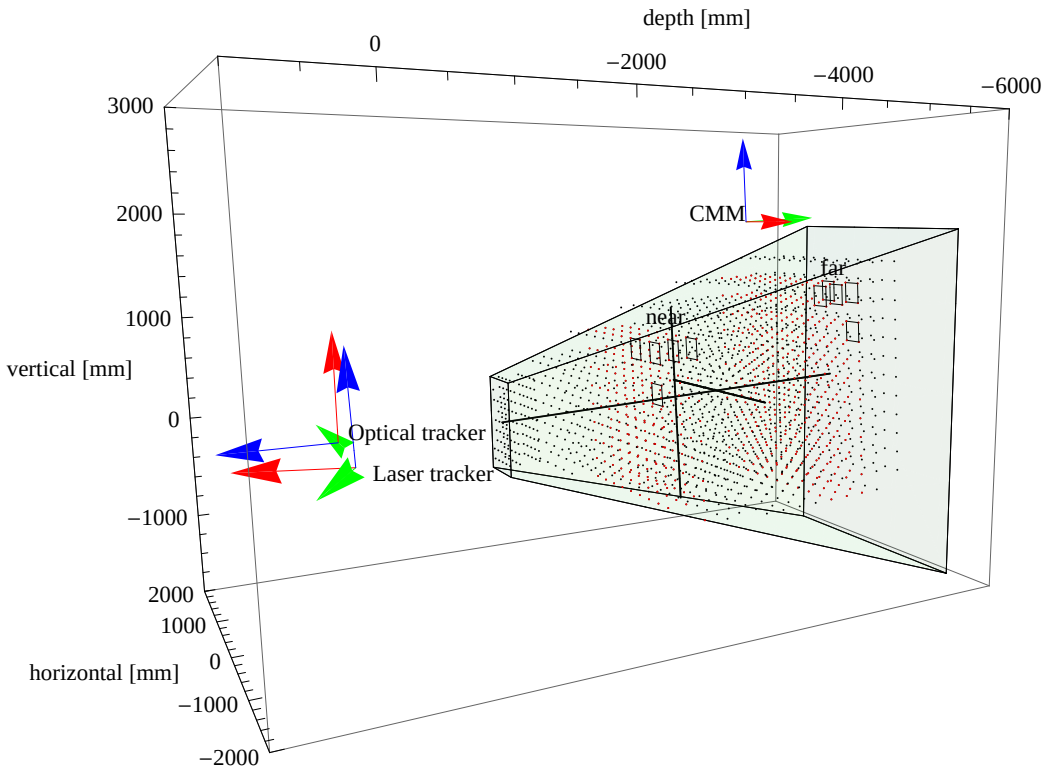
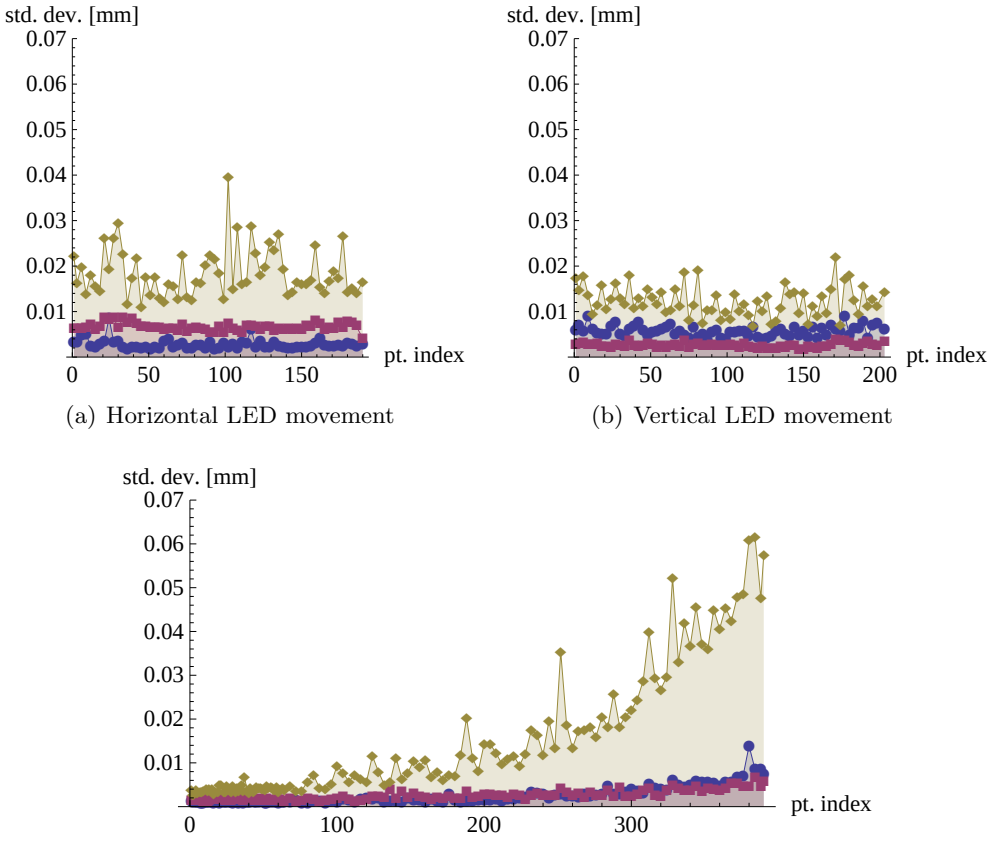


Figure 12.5.: Measurement volume, point grid and lines along coordinate axis. The "simulated" reference target is located in the areas marked "near" and "far", as indicated by the quadrangles.

in our exemplary scenario. The result is shown in Figure 12.6. For each position on the three densely sampled lines (x-axis), an RMS residual error is computed according to (10.15). The evaluation shows that noise is constantly low in the horizontal and vertical directions. It increases however with the depth, assumably due to a decreasing number of affected pixels on the sensors. Still, random noise is of secondary importance, with a maximum RMS of 0.06 mm in the far end of the volume. More important are the systematic effects described next.

12.4.3. Systematic Effects

The accuracy of distance measurements between LED positions on the lines is evaluated by a comparison against the corresponding distance measurements of the CMM. Noise has been practically eliminated before by averaging over all 60 samples per position. The result is shown in Figure 12.7. Three different distances are considered in the range of 10 mm up to 100 mm. Each of these distances was moved incrementally along the three densely sampled axes (x-axis). The deviation of two corresponding lengths is plotted at that position (y-axis).



(a) Horizontal LED movement

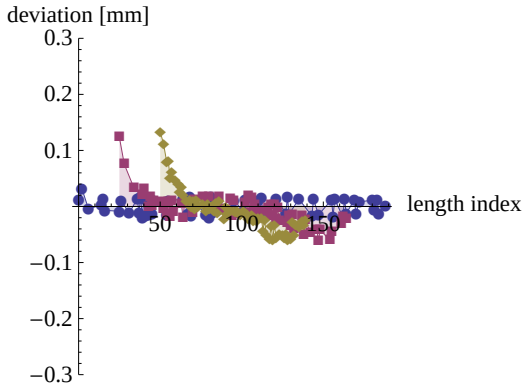
(b) Vertical LED movement

(c) LED movement in depth direction

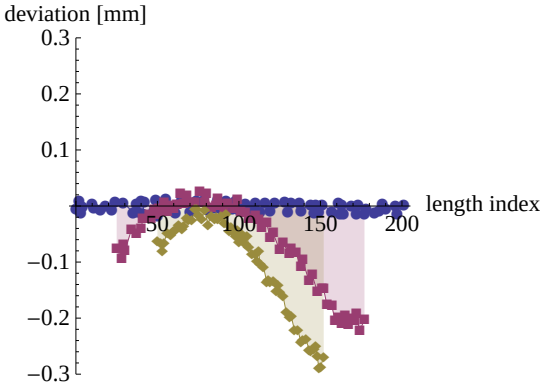
Figure 12.6.: Noise: standard deviation [mm] of LED position along horizontal/vertical/depth lines, computed from 60 samples per position, plotted separately for each coordinate (◯ (blue): horizontal, □ (purple): vertical, ◊ (brown): depth)

There is no systematic effect for smaller distances. The error is small and increases in magnitude for larger depths. For larger distances, however, there is a tendency to overestimate lengths in the near part and to underestimate them in the far part of the volume. This indicates that systematic errors take effect rather globally than locally.

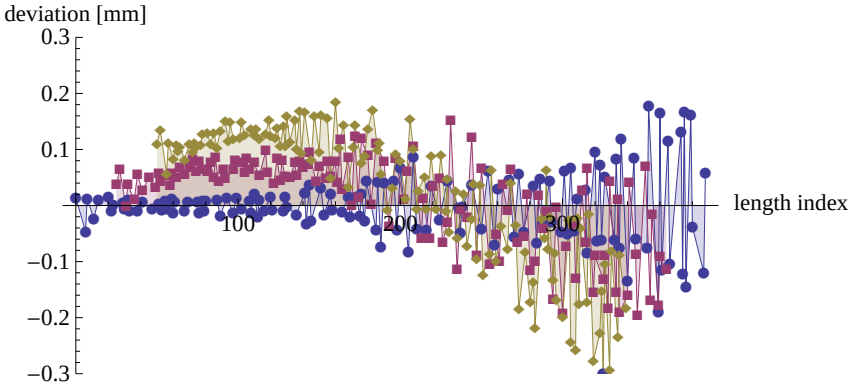
Next, systematic effects are analyzed in the grid covering almost the entire tracking volume. By each of the three measurement systems, a point cloud as shown in Figure 12.5 has been recorded. Matching these point clouds by a 6DoF Helmert transformation $\boldsymbol{p}_i = \boldsymbol{H}\boldsymbol{q}_i$ (cf. 12.3.1) or the linear approximation method (cf. 12.3.2) gives information about the accuracy of the individual systems [Niem 08]. Noise again has been practically eliminated before. The CMM measured its tip position, not the position of the LEDs. However, the orientation of the tip remained constant, so the desired positions of the LEDs could be translated by a constant offset from the actual measurements. This is



(a) Horizontal axis



(b) Vertical axis



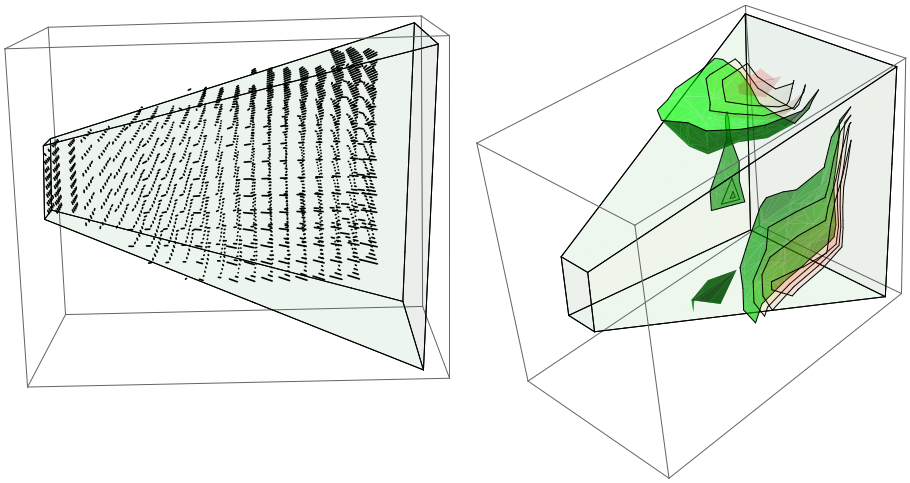
(c) Depth axis

Figure 12.7.: Systematic error: deviation [mm] of distance between two measured LED positions (mean of 60 samples) from the corresponding reference distance (◯ (blue): 10 cm, □ (purple): 50 cm, ♢ (brown): 100 cm)

handled implicitly by the transformation $\boldsymbol{H}$. The same holds for the offset between the LEDs and the laser reflection target. For the Helmert transformation, a random subset of 200 points was used, instead of the almost 2000 available points, due to computational complexity and limitations in *Java Graticule 3D*.

Matching the point clouds from CMM and laser tracker with the linear method results in an RMS of only 0.0369 mm. This overall RMS covers the errors of both systems. Its low value is in accordance with the specified accuracies. In conclusion, both systems, CMM and laser tracker, provide a good ground truth for testing the accuracy of the real-time tracking system.

Ideally, a 6DoF rigid body transformation should suffice to match the real-time tracking system with the CMM. The Helmert method yields a joint RMS of 0.440 mm, which is mostly caused by the real-time tracker, since the error of the CMM is by an order of magnitude lower, as specified by the vendor and shown above. The linear approximation method (cf. 12.3.2) results in a similar value of 0.406 mm. Matching the real-time tracking with the laser tracker using the linear method results in 0.410 mm. Considering the vendor specification, Equation (10.14), and the fact that most grid points are located in the far part of the pyramid, these are reasonable values. The resulting deviation vectors are shown in Figure 12.8, scaled by a factor of 100 for better visibility.



(a) Deviation vectors, scaled by 100

(b) Error isocontours for 0.5722 / 0.8 / 1.0 / 1.2 mm RMS

Figure 12.8.: Systematic deviations in tracking volume

The plot reveals that there are systematic effects, especially in the outer areas of the volume. The computed $\boldsymbol{H}$ minimizes the RMS. Therefore, errors are lower in the center of the volume and increase towards the boundaries. This is the same effect as in Figure 5.3. Furthermore, errors on the boundaries are balanced through the minimization. According to Equation (10.14), the RMS value specified by the vendor and confirmed

by our measurements, consists of a dominating systematic error μ and a minor noise component σ. Computing a more general 7 DoF similarity transform using the Helmert method yields a similar value of 0.439 mm RMS. However, the 12 DoF affine transform reduces the error to 0.253 mm RMS, indicating that a better pre-calibration of the system on behalf of the vendor would be possible. This is similar to the system we evaluated in [Keit 08].

It is worth noting that the RMS error for matching the point clouds of two LEDs (e.g., LED 1 and 2) is only one fourth of the global RMS (0.116 mm). This indicates that there is a strong local dependency of the deviation vectors, in other words, a systematic error.

To give an intuitive example of the consequences, one might consider the probe target shown in Figure 1.6(c) which has an edge length of 5 cm. The span of this target is small compared to the distances at which systematic distortions take effect (cf. Figure 12.7 and Figure 12.8), and therefore all LEDs are affected by a similar deviation vector. This results in a direct error in the position estimate. Its orientation, however, is estimated better than the specified error model predicts because the relative constellation of the four LEDs is barely affected. This fact has an impact on the experiment described next.

The captured point clouds lend themselves to also test the estimation of variance components (cf. 12.3.2) to assess the uncertainties of all involved devices without any prior assumptions about their accuracy. The result is an RMS error of 0.0286 mm for the CMM, 0.0467 mm for the laser tracker, and 0.407 mm for the LED. The latter value is almost identical to the results obtained above, assuming the CMM / laser tracker as a ground truth. Also, the estimated values for CMM and laser tracker come pretty close to the specification of the vendor (see above). Given enough measurements, the technique thus yields very accurate results.

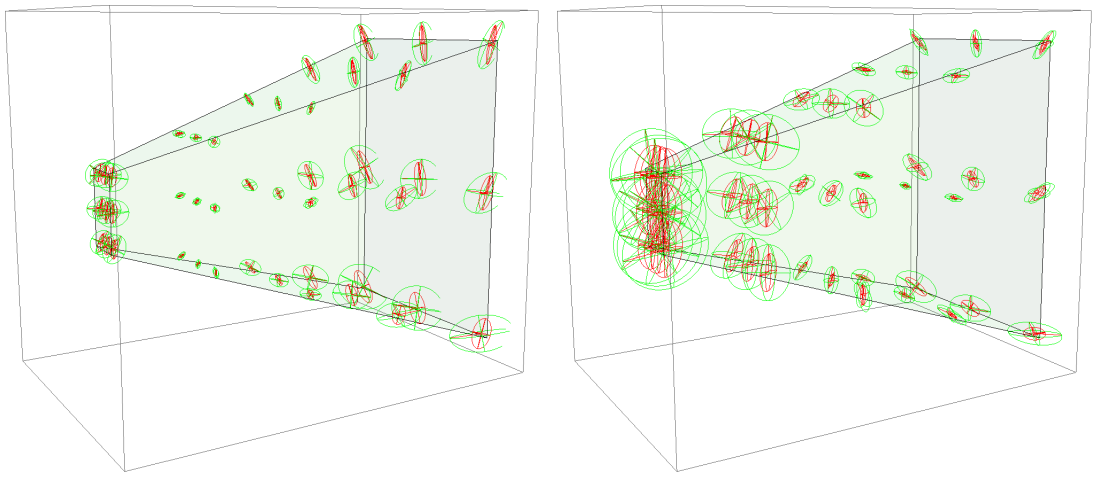
12.4.4. Indirect Tracking Experiment

The consistency of our simulation model is supported by a comparison with extensive empirical measurements in the CMM setup. For this, we focus on the accuracy of indirect tracking since it has a major impact on the overall accuracy of our sample scenario and dominates the impact of probe tracking, tip calibration, and world registration. Thus, we estimate how accurate a given POI can be estimated in world coordinates, under the influence of an error prone tracking of the reference target. Based on the grid of measured LED coordinates, we defined a virtual reference target consisting of four adjacent grid positions (20x20 cm^2). As highlighted in Figure 12.5, the virtual reference target is iterated through the two marked areas in the center (near) and at the end of the pyramid (far).

Indirect tracking uses the reference target to self-localize the tracking system and thereby derive the estimate for the POI. This means the given POI should remain constant in world coordinates for any position of the reference target. Due to errors in LED tracking, this is not perfectly true. A 3×3 covariance can be estimated for the POI, once for the empirical LED positions, and once for the simulation, perturbing the ground

truth grid positions using the vendor's uncertainty specification. In case the specification and our simulation model are correct, the resulting covariances should match.

The result is shown in Figures 12.9(a) for the near and 12.9(b) for the far location of the reference target.



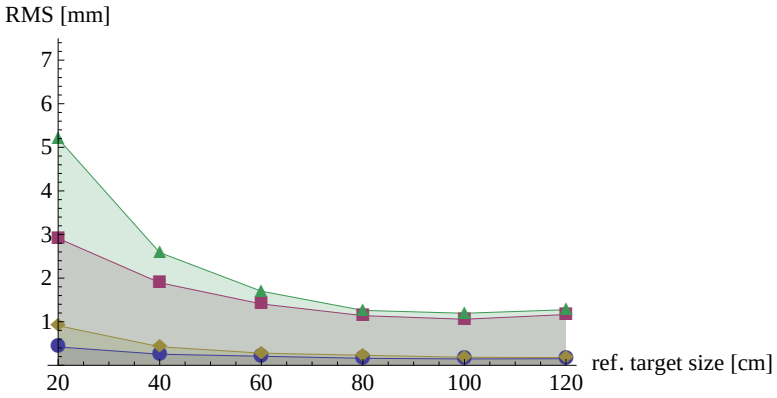
(a) near reference target (b) far reference target

Figure 12.9.: Comparison of simulation (green) and empirical measurements (red)

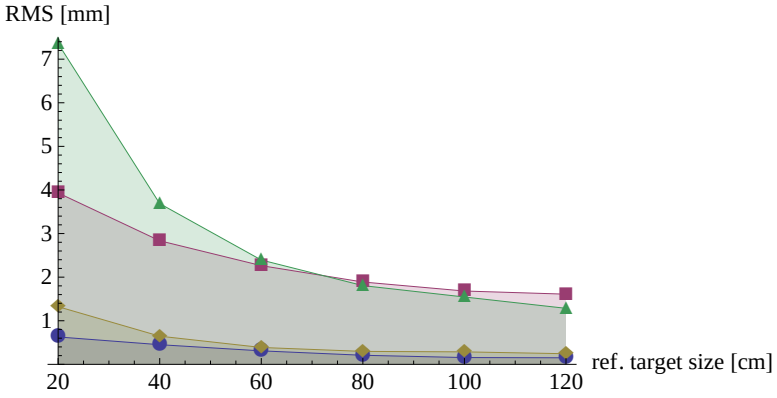
The plots show the confidence ellipsoids corresponding to these covariance matrices (enlarged by a factor of 100) for various POIs throughout the volume. First of all, it can be seen quite clearly that errors increase with an increasing distance of the probe with respect to the reference target, an important fact for the definition of future work processes. Furthermore, the simulated covariance matrices always enclose the empirical covariance matrices but otherwise have a similar shape. This means the simulation yields a reasonable qualitative description of the error, but in this case is an upper bound.

There is a simple explanation for this fact: the rather low frequency of the systematic distortions mentioned above. It results in a lower rotational error than the assumption of a purely Gaussian error would predict which in turn increases the performance of indirect tracking. The noise specified by the vendor is globally correct but has local dependencies as seen in the error analysis before. The overestimated rotational error in the simulation is propagated over a long axis and therefore leads to an over-estimation of the indirect tracking error.

The experiment has been repeated using other, larger sizes of the reference target, up to an edge length of 6 grid positions (120x120 cm^2). Figure 12.10 shows the empirical and simulated minimum and maximum errors. The larger the target, the better the empirical results are approximated by the simulation.



(a) near reference target



(b) far reference target

Figure 12.10.: POI RMS errors [mm] for indirect tracking (△ (green): simulation maximum, □ (purple): empirical maximum, ◊ (brown): simulation minimum ◯ (blue): empirical minimum) depending on the size of the reference target (1: 20 cm, 2: 40 cm, 3: 60 cm, 4: 80 cm, 5: 100 cm, 6: 120 cm)

12.4.5. Summary

Altogether, the evaluation of the optical IR tracker in the CMM setup coincides with both, the vendor specifications and prior evaluations [Wile 05] [Schm 09]. We could verify its magnitude and also its correlation with depth. Nevertheless, some important additional facts are revealed for the subsequent accuracy analysis in the target environment. Systematic errors dominate the overall error, especially for marginal positions. Thus, for critical applications, some outer parts of the pyramid might be clipped. Likewise, only small benefits can be expected from computing mean values for positions measurements. Also the strong local coherence of the systematic error leads to an overestimated positional error for small targets whereas the orientational error is underestimated, compared to the assumption of a globally random error.

13. Verification & Validation

In the following, the approach of integrated verification & validation is applied in terms of comprehensive evaluations of the two sample scenarios. This makes heavy use of the concepts for simulation described in Chapter 11 and also the basic uncertainty specifiations described in Chapter 12.

13.1. Example: Indirect Tracking of the Intelligent Welding Gun

In theory, indirect tracking as described in 7.1.1 works properly. In practice, however, the indirection via the Mobile Target and the Mobile Cameras can result in serious errors estimating the pose of the Welding Gun with respect to the Stationary Cameras which determines the world coordinate system. The goal is to reduce those errors and thereby ideally close the gap in tracking accuracy between indirect and direct tracking [Keit 08].

Positional errors in the concatenated transformations just accumulate when transforming the measurement of the mobile cameras back into our reference frame defined by the Stationary Cameras. Any rotational error, however, results in positional errors at the region of interest which increase linearly with the distance [Holl 95] [Holl 97] [Baue 06]. This particularly holds for the orientation of the Mobile Target, the *orientation* estimation of which we assume to be rather imprecise. To clarify the importance of accuracy for the proposed setup, one may imagine that a deviation of 0.1 in the estimated pose of the Mobile Cameras results in a displacement of 1.7 (3.4) mm between estimated and true position of the Welding Gun in a distance of 1 (2) m from the mobile setup. This follows directly from the *sine* function. The general idea for reduction of this error is to use *common reference points* somewhere in the scene which are visible to both, the Stationary Cameras and the Mobile Cameras, in order to estimate the orientation of the Mobile Cameras as good as possible, as depicted in Figure 13.1.

The work on this scenario was initiated in 2006. At this time, the Monte Carlo simulation techniques (cf. Chapter 11) were not yet at our disposal. Also, ground truth measurements were not available right from the beginning. Results of this evaluation are presented in chronological order. Experiments for the validation are presented first. They are backed by a retrospective validation based on simulation.

13.1.1. Correcting Rotational Errors

As already stated above, to improve the accuracy of indirect tracking setups, it is crucial to minimize the error in orientation estimation of the mobile tracking setup. In this

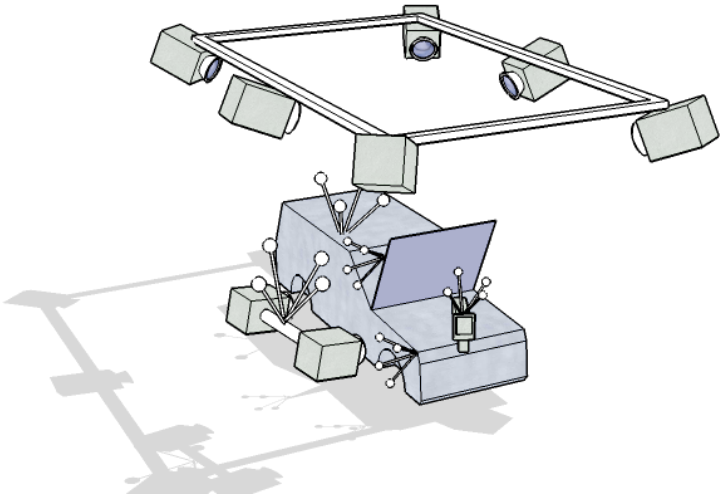


Figure 13.1.: Indirect tracking setup to mitigate the occlusion problem. Two reference targets are used to correct for rotational errors in tracking the mobile camera system, see also Figure 1.12.

section, methods are presented which allow for on-the-fly compensation of this error without extensive recalibration. The first method assumes that the mobile setup is being tracked and constantly corrects the orientation by one or two additional reference points in the scene. The second method gets by without tracking the mobile setup. It needs at least three reference points for pose estimation. Even though the latter approach is not new, we incorporate it in our evaluation for the sake of completeness and also as a reference for our correction methods.

Tracking of Mobile Setup

One (implicit) reference point correspondence is already given by the center-of-gravity (COG) of the Mobile Target if we assume that *position* tracking of the Mobile Target with respect to the Stationary Cameras is not error-prone and that the Mobile Target is registered well with the Mobile Cameras. Additional reference points may come from marker balls or complete 6 DoF targets that are suitably placed in the scene and that are seen by both tracking systems.

Especially when using a big mobile target (such as shown in Figure 7.1), it is important to apply the rotational correction in a coordinate frame with the COG of the target marker balls as its origin because it probably is the center of rotation where the rotational error gets added to the true orientation of the target in the pose estimation algorithm used by the vendor [Egge 97][1]. Usually, however, the target frame defined by the vendor differs. The situation is depicted in Figure 13.2.

Via the static transformation $\boldsymbol{T}_{\text{CMT2MC}}$ between the Corrected Mobile Target and the

[1] See also 11.4 for a discussion about the correctness of this assumption

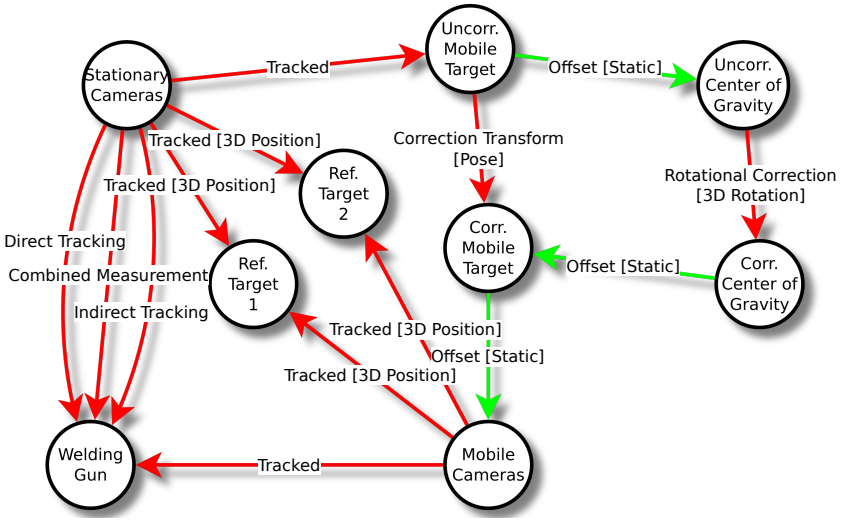


Figure 13.2.: SRG for indirect tracking of the intelligent welding gun. Using at least one Reference Target that is seen by Stationary Cameras as well as the Mobile Cameras, a Rotational Correction for the error of outside-in tracking of the Uncorrected Mobile Target can be computed. Only the important spatial relationships are shown, most intermediary results are hidden for the sake of clarity.

Mobile Cameras and the translational offset $\boldsymbol{T}^{-1}_{\mathrm{MT2COG}}$ between the Mobile Target and the Center of Gravity, we are able to transform the positional measurements of the Reference Target(s) made by the Mobile Cameras to the Corrected Center of Gravity coordinate frame. $\boldsymbol{T}_{\mathrm{MT2COG}}$ can be obtained from the body calibration of the Mobile Target. Estimation of $\boldsymbol{T}_{\mathrm{CMT2MC}}$ is a bit more complicated (cf. section 13.1.2). Concatenation of both transformations results in a 3D position of the reference point(s) (as seen by inside-out tracking) in the Corrected Center of Gravity coordinate system. We call these the *reference points*

$$\boldsymbol{p}_{\mathrm{ref}} = \boldsymbol{T}^{-1}_{\mathrm{MT2COG}} \boldsymbol{T}_{\mathrm{CMT2MC}} \boldsymbol{p}_{\mathrm{mobile}}. \tag{13.1}$$

The *corresponding measurement* made by the stationary tracking system $\boldsymbol{p}_{\mathrm{stationary}}$ can be transformed to the Uncorrected Center of Gravity frame. However, since the transformation between the Stationary Cameras and the Mobile Target $\boldsymbol{T}_{\mathrm{SC2UMT}}$ is perturbed by rotational error by assumption, this yields a wrong 3D position of the reference point(s) (as seen by outside-in tracking) in the Uncorrected Center of Gravity coordinate system. We call these the *erroneous points*

$$\boldsymbol{p}_{\mathrm{err}} = \boldsymbol{T}^{-1}_{\mathrm{MT2COG}} \boldsymbol{T}^{-1}_{\mathrm{SC2UMT}} \boldsymbol{p}_{\mathrm{stationary}}. \tag{13.2}$$

Based on the reference point correspondence(s), a rotational correction R_{corr} is computed between Uncorrected Center of Gravity and Corrected Center of Gravity, using one of the methods described in the following subsections. The indirect tracking equation

then becomes

$$\boldsymbol{p}_{\text{stationary}} = \boldsymbol{T}_{\text{SC2UMT}} \boldsymbol{T}_{\text{corr}} \boldsymbol{T}_{\text{CMT2MC}} \boldsymbol{p}_{\text{mobile}} \tag{13.3}$$

with the additional correction transform

$$\boldsymbol{T}_{\text{corr}} = \boldsymbol{T}_{\text{MT2COG}} \boldsymbol{R}_{\text{corr}} \boldsymbol{T}_{\text{MT2COG}}^{-1} = \begin{bmatrix} \boldsymbol{R} & -\boldsymbol{R}\boldsymbol{t} + \boldsymbol{t} \\ 0 & 1 \end{bmatrix}. \tag{13.4}$$

One Reference Point Correspondence Let us assume for now that rotational error occurs only in directions that are orthogonal to the line joining the center of gravity of the mobile target and a known reference point in the scene. We call this line the *direction of the reference point*. Let us further assume that the position of the mobile target is tracked perfectly and that the 3D position of the reference point is known perfectly in both tracker frames.

The geometry of this problem is shown in Figure 13.3. The COG of the mobile target lies in the center of a unit sphere. By normalization, a general point in the scene can be converted to a point on the surface of the sphere. It is sufficient to consider this directional vector. Every rotation about some axis through the rotation center lets this direction vector describe a circular orbit on the surface of the sphere.

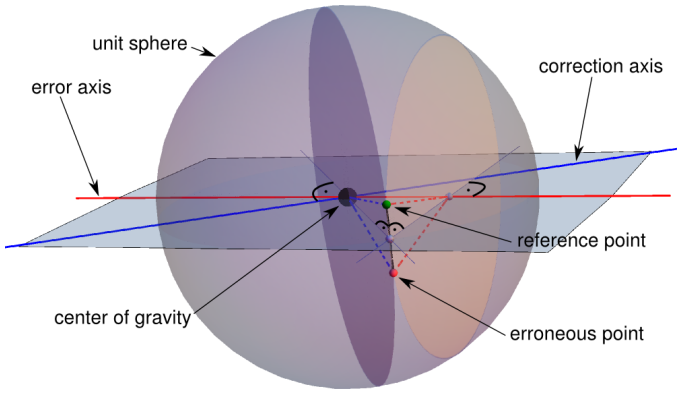


Figure 13.3.: *Simple correction* of rotation using a single reference point in the scene. Only the part of the rotational error that is orthogonal to the line joining the center-of-gravity (COG) of the mobile target and the reference point itself can be corrected.

First, the erroneous point as well as the reference point are projected onto this sphere. Thereby, they define the normal vector to a plane through the COG. It contains all the axes that would be suitable for mapping the erroneous point back to its reference position. Furthermore, all axes pass through the rotation center. The rotation axis about which the original error rotation took place, is shown in the Figure 13.3 but its orientation in the plane is unknown. However, we obtain another (normalized) rotation

axis from the vector cross product of the two measurements of the reference point.

$$\boldsymbol{a} = \frac{\boldsymbol{p}_{\text{ref}} \times \boldsymbol{p}_{\text{err}}}{\|\boldsymbol{p}_{\text{ref}} \times \boldsymbol{p}_{\text{err}}\|} \tag{13.5}$$

With a rotation about this axis, by the angle between the two directions of the erroneous point and the reference point, it is always possible to map the erroneous point back to its reference position. The rotation angle ϕ is obtained by

$$\phi = cos^{-1}(\boldsymbol{p}_{\text{ref}}^T \boldsymbol{p}_{\text{err}}). \tag{13.6}$$

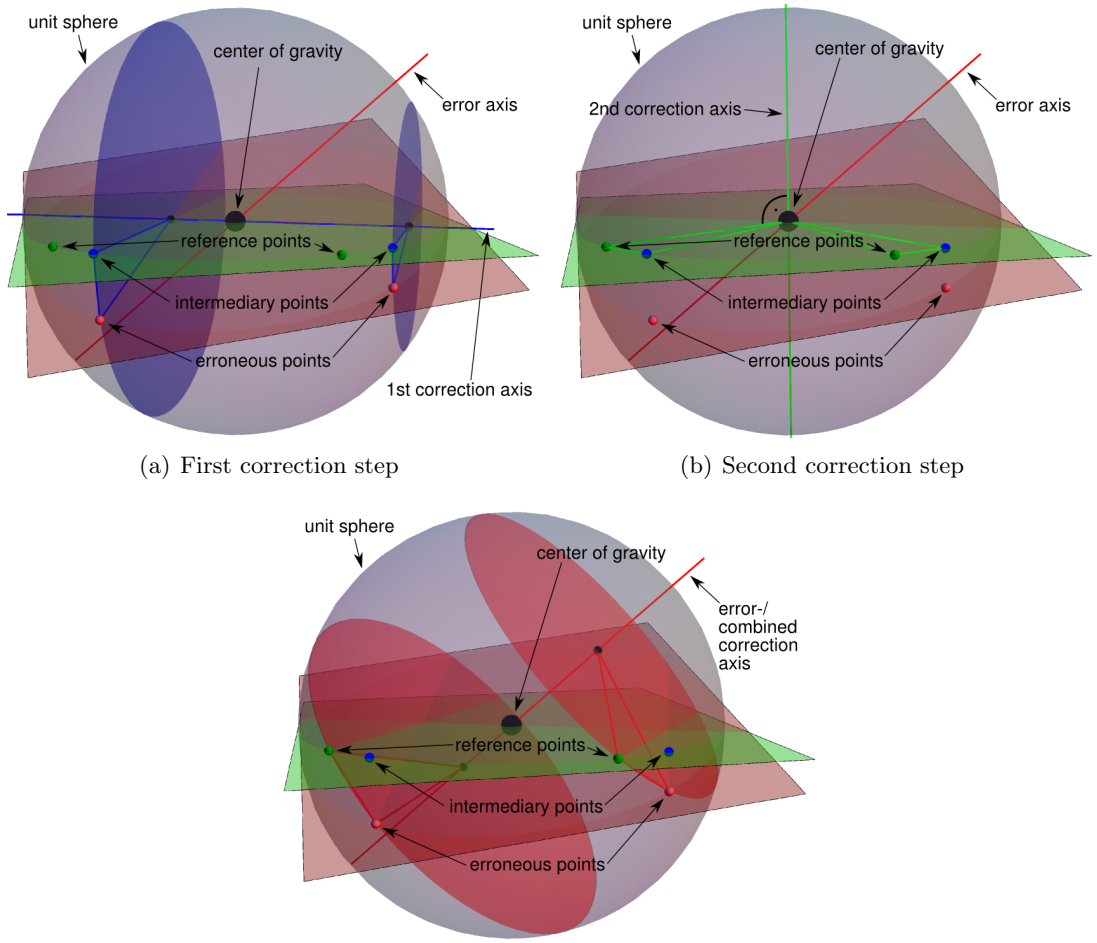
This rotational correction can then be in terms of a quaternion using Equation (10.22).

For points different from the erroneous point, this correction yields wrong results as soon as the error rotation axis contains the direction of the reference point in its linear combination. In most cases, the error can still be reduced. Due to its lengthy and flat shape, e.g., our mobile target (see Figure 7.1) probably exhibits the greatest rotational uncertainty in its pitch axis (parallel to the long edge). If the reference target is placed in front of the cameras, the orthogonality assumption might be save.On the other hand, an error with its axis being aligned with the direction of the reference point can not even be detected using a single reference point since the point is mapped onto itself by the error. For the sake of brevity, this approach will also be called *simple correction* in the following.

Two Reference Point Correspondences As an alternative, we use two additional point correspondences A and B, thus an overall count of three if the implicit COG is also counted. The spatial relationships are again depicted in Figure 13.2. The availability of three non-collinear point correspondences principally enables us to solve the absolute orientation problem (7 DoF similarity transformation between two point sets) between the stationary and mobile tracking system using any absolute orientation algorithm (cf. section 13.1.1, see also [Egge 97]). However, the problem at hand allows us to make various simplifications which lead to a neat solution of the problem.

- It is not necessary to find a scaling factor because both tracking systems already provide metric measurements. This reduces the problem by one degree of freedom.
- If we again transform the known 3D measurements of the two reference targets from their respective coordinate frames of the stationary and the mobile tracking system into the frame of the mobile target's COG, then both coordinate systems already share the same origin. Thus, it is not necessary to solve for the translational part. This reduces the problem by another three degrees of freedom.

The remaining problem is a 3 DoF rotation about the COG of the mobile target. A nice solution has been presented by Horn for estimating the orientation in the special case of coplanar point sets [Horn 87]. It is particularly suitable for three point correspondences which are always coplanar. A sketch is given in Figure 13.4. Since one



(a) First correction step

(b) Second correction step

(c) Combined

Figure 13.4.: *Full correction* of rotation using two reference points in the scene. (a) First, the plane defined by the erroneous points (red) and the center-of-gravity (COG) is mapped to the plane defined by the reference points (green) and the COG by a rotation about the intersection line of these two planes. This maps the reference points to the intermediary points. (b) An in-plane rotation about the COG then maps the intermediary points to the reference points. (c) Effect of both correction steps combined. The true error axis is reconstructed.

correspondence is given by the common origin of the COG reference frame, the remaining two correspondences result in two independent position vectors $\boldsymbol{p}_{\mathrm{err}_A}$ and $\boldsymbol{p}_{\mathrm{err}_B}$ for the erroneous points and $\boldsymbol{p}_{\mathrm{ref}_A}$ and $\boldsymbol{p}_{\mathrm{ref}_B}$ for the reference points.$\boldsymbol{p}_{\mathrm{err}_i}$ and $\boldsymbol{p}_{\mathrm{ref}_i}$ define two planes, one containing the erroneous points and one containing the reference points. An informal description of the algorithm is given in the following.

1. The first step is depicted in Figure 13.4(a). Normals to both planes are computed using the vector cross product. From these normals, a rotation axis is deduced, again by the cross product. The angle between the plane normals is the searched for rotation angle. By using Rodrigues' formula for rotating one plane by the computed angle about the computed axis, both sets of coplanar points are mapped into a single plane. This step of course only yields correct results if either all the points are lying in an exact plane or only three points are used. This corresponds to a mapping of the plane spanned by the erroneous points and the origin (COG) to the plane spanned by the reference points and the origin. Thereby, the erroneous points are mapped to the intermediary points.

2. Step two according to Figure 13.4(b) is a rotation in the reference plane about the normal to that plane, which maps the intermediate points from the last step to the reference position. Estimating the angle which yields the least-squares solution is now only a one DoF problem. In contradiction to Horn's solution, however, we do not consider the distance between the original points, but the distance of the normalized points on the unit sphere. Thus, the optimal angle is given by the mean of the angles between points A and B. This corresponds to a mapping of the intermediary points to the reference points.

The result of combining these two steps is depicted in Figure 13.4(c). It will also be called *full correction* in the following.

General Absolute Orientation Approach

Another approach to solve the indirect tracking problem is to abandon the idea of a tracked mobile setup in combination with one or two additional reference points in the scene and use a large number of point correspondences instead. This solution would require a brief calibration process after movement of the mobile setup and before the ongoing tracking process where the tracked object is being panned within the close-up range of the mobile system which is also visible by the stationary system. The resulting point correspondences could be used to solve the general absolute orientation problem each time the pose of the mobile setup has changed [2].

Several closed-form least-squares solutions exist for the absolute orientation problem. A comparison conducted by Eggert et al. [Egge 97] did not yield a particular preference for any of them. For our experimental evaluations, we used Horn's quaternion-based solution, in a slightly stripped-down version which does not optimize for the scaling factor [Horn 87]. This version of the algorithm is also contained in *Ubitrack* (cf. 5.3.2).

13.1.2. In-vitro Validation of Indirect Tracking

The methods for correction of rotational error described in the previous section have been evaluated on real tracking data in the lab. All tests were carried out using a

[2]The mobile target might still be useful in order to detect these position changes.

stationary *A.R.T.* system consisting of three *ARTtrack 1* cameras and a mobile *A.R.T. SmARTrack* system mounted on a tripod, consisting of two *ARTtrack 2* cameras with a baseline of approximately 0.5 m. The latter is depicted in Figure 7.1. The acquisition of measurement data was carried out using the the *Ubitrack* framework (cf. 5) and with the aid of *trackman* (cf. 6). Data was evaluated offline with *Mathematica*[3].

Calibration and Registration

For calibration of the camera setups and the involved tracking targets, the mechanisms prescribed by the vendor were used. Each target was calibrated in the stationary tracking setup. It is supposed to yield better results than a two-camera setup with a small baseline of 50 cm. The mobile target was mounted to the mobile setup before calibration of extrinsic camera parameters in order to avoid negative effects of this mechanical intervention with regard to both, camera and target geometry.

It is crucial to register the mobile target and the mobile cameras very accurately. Based on our experiences, though without any explicit evidence yet, the method based on the average of multiple absolute orientations was chosen for this (cf. 7.2.2). The topic is picked up again below in 13.1.4. Assuming that orientation tracking of the mobile target suffers from pose-dependent systematic errors, we decided to use measurements from several poses of the mobile setup in the registration process.

For each pose of the mobile setup, points were measured with a pointing device in both tracker coordinate systems. Pointing at a rigid surface ensures that the tool tip remains constant during the acquisition of its position by the two measurement systems even under the influence of misaligned sensor timestamps. In a proper setup allowing for continual measurements, one would have to synchronize the cameras using a common sync source which should not be a problem with quality tracking equipment.

Next, the absolute orientation problem was solved independently for each pose of the mobile setup (cf. section 13.1.1). The solutions corresponding to the different poses were then combined to a single transformation in order to obtain a single calibration of higher precision.This was done by averaging the translational offsets in each axis as well as the Euler rotation angles, as described in 7.2.2.

Evaluation of the Indirect Tracking Setup

The pointing device was also used to make measurements. Unlike in the intended use-case, it was always visible in both tracking systems. By this means, a relative comparison was possible of 3D points measured indirectly via the mobile system against those measured directly by the stationary system.

For the acquisition of evaluation data, the mobile system was sequentially set up in two distinct poses *one* and *two* within the tracking volume of the stationary cameras. For each pose, 35 point correspondences were measured on a rigid surface in the common tracking volume of both systems. No averaging was performed for the point measure-

[3] *Wolfram Research*

ments. Figure 13.5 shows the measurements of the stationary tracking system as well as the COG of the mobile target for pose *one* as spheres.

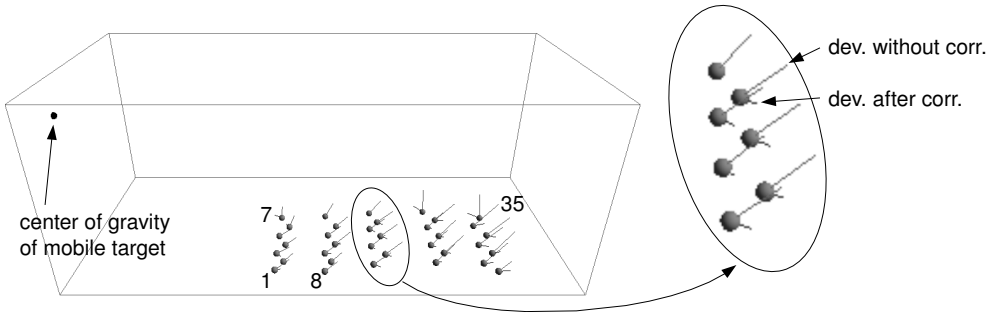


Figure 13.5.: Evaluation setup for pose *one*. The vectors show for each point the deviation of indirect tracking from direct tracking whereas larger deviations result from indirect tracking without rotational correction. Smaller deviations in this example result from the simple correction using reference point 18.

Instead of placing reference targets in the scene, different subsets of the measured points were used as reference positions in the offline evaluation of the data.

For both poses of the mobile system, the three approaches described in section 13.1.1 were tested and compared to the naive indirect tracking approach which works without any correction at all. In Figure 13.5 the distance of the indirectly tracked points obtained from the naive approach from the corresponding directly tracked points are depicted by light-colored lines, lengthened by a factor of 50 for better visibility. Results for other poses of the mobile setup feature a similar systematic behavior, and the orientation of deviation vectors varies depending on the chosen pose. This suggests a significant amount of systematic error which we cannot fully explain at the moment. The shorter dark lines depict—exemplarily for the simple correction using the point with ID 18 as reference point—the distance of the indirectly tracked points to their directly tracked correspondences, again lengthened by a factor of 50. All conducted evaluations are described in the following.

Simple correction (1 additional point) Based on one reference point correspondence, a correction of the orientation of the mobile target was carried out following the approach described in section 13.1.1. As stated there, this method only corrects those parts of the original rotational error that are orthogonal to the line joining the COG of the mobile target and the reference point. The correction results are presented in table 13.1 for six reference points with varying positions relative to the mobile cameras.

Full correction (2 additional points) The information of two of the measured points was used to compute an orientation correction according to the method described

in section 13.1.1. The correction result is presented for six pairs of reference points with different positions relative to the mobile cameras. As a test for robustness, the first three pairs were chosen to contain points very near together. The last three pairs eachcontain points on opposite sides of the volume.

General absolute orientation (5 points) A subset containing five of the measured 35 points is used to compute a rigid transformation with an absolute orientation algorithm(cf. 13.1.1). The correction result is presented for three sets of five points each, one near and one far to the mobile cameras as well as one random sample of all points.

General absolute orientation (all points) All point correspondences are used to compute the absolute orientation. Since the resulting rigid transformation is optimal in a least-squares sense (cf. 13.1.1), the measurement residuals give information about how well the actual mapping can be described by a rigid transformation.

For each indirect tracking approach, the RMS deviation was computed in mm and the results are shown in Table 13.1.

Table 13.1.: RMS error of rotational correction approaches. For all evaluated constellations of correction approach, mobile setup pose, and reference points, the RMS euclidean distance [mm] between indirectly and directly tracked points is given.

No corr.		Simple corr.			Full corr.			Abs. orient.			Gen. abs. orient.	
Pose 1	Pose 2	Pt.	Pose 1	Pose 2	Pts.	Pose 1	Pose 2	Pts.	Pose 1	Pose 2	Pose 1	Pose 2
3.1	2.7	1	1.2	1.8	4,11	1.8	1.6	1,4,7,	1.2	1.6	1.0	1.2
		15	1.1	1.4	18,25	1.2	1.5	9,13				
		29	1.1	1.4	25,32	1.2	3.4	22,25,28,	1.0	1.3		
		4	1.1	1.7	1,14	1.4	1.8	31,34				
		18	1.1	1.3	15,21	1.1	1.4	1,10,22,	1.1	1.4		
		32	1.1	1.6	22,35	1.1	1.4	32,35				

The residuals from the absolute orientation with all points represent the greatest lower bound for all error corrections based on optimizing a rigid transform. Lower residuals could only be achieved by fitting a more general, maybe even non-linear mapping. However, for a metric measurement system, this is not desirable. There must be a considerable amount of probably systematic error left that can only be explained by discrepancies between the two tracking systems. Possible reasons are the following:

- Deficiencies in the room calibration algorithms of the manufacturer which determine the intrinsic and extrinsic camera parameters
- Errors introduced during body calibration
- Errors introduced during tip calibration of the measurement tool

- Insufficient number of point correspondences during those two types of calibration
- Inaccurate 3D point measurements due to merging markers which is when two marker balls come close to the same line of sight and converge to a single point measurement on the image plane
- Tolerances in the manufacturing process, each marker ball is laminated with retro-reflective foil by hand.

All correction mechanisms perform quite well in converging to the theoretical lower bound. In most constellations, at least two third of the residuals can be explained and corrected by the assumption of erroneous orientation estimation of the mobile target. The full correction seems to have problems with point correspondences that are chosen too close together, especially, if they are far away (points 25 and 32) and therefore result in very similar direction vectors. Thus, reference points should be chosen carefully in order to avoid such degenerate configurations.

Astonishingly, the simple correction is not worse than the full correction. This could be explained by the layout of our mobile target which suggests that pitch movements of the mobile setup can be detected worst by the stationary cameras. Since this pitch axis is roughly orthogonal to the line joining the COG of the mobile target and the reference point, these errors can be corrected quite well. In general, the full correction with a reasonable pair of reference points should be preferred, if it is feasible. A good alternative is the general absolute orientation, if enough point correspondences are available or can be obtained without much effort.

13.1.3. In-situ Validation of Application

In the next step, the evaluation described above was repeated in the intended working environment.

All tests were carried out using a stationary *A.R.T.* system consisting of five *ARTtrack 1* cameras and a mobile *A.R.T. SmARTrack* system mounted on a mobile workshop cart, consisting of two *ARTtrack 2* cameras with a baseline of 1 m. The setup is depcited in Figure 13.6.

Additionally, ground truth measurements were conducted using a *FARO* mechanical tracking arm similar to the one shown in Figure 1.7. The arm was registered with the stationary cameras in advance, using the absolute orientation method based on 10 measured 3D point correspondences (cf. 7.2.1). An idential registration was also made for a particular setup of the mobile cameras, to assess their quality for direct tracking. This allows one to use the SRG depicted in Figure 13.7 for evaluation. The distance of the mobile cameras to the measurement volume was approximately 1.5 m, the distance of the stationary cameras approximately 3.0 m.

For validation, the deviations of direct and indirect tracking with respect to the ground truth measurements made with the mechanical arm were assessed. Coordinates were measured on a front vehicle as depicted in Figure 1.6(b). A point probe (as depicted in the same image) was used instead of the real welding gun to ensure consistent point

(a) Stationary camera system (b) Mobile camera system

Figure 13.6.: Tracking systems used in in-situ validation of indirect tracking

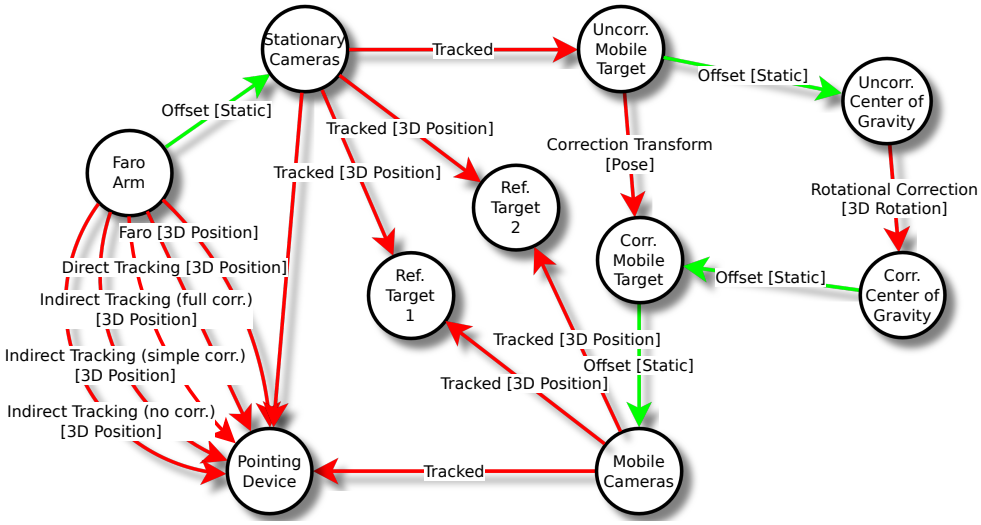


Figure 13.7.: SRG for in-situ validation of indirect tracking setup. Direct tracking, as well as indirect tracking in its corrected and uncorrected versions can be directly compared against the ground truth measurements made in advance with a *FARO* mechanical tracking arm. Only the important spatial relationships are shown, most intermediary results are hidden for the sake of clarity.

probing using the probe and the tracking arm. 14 drillings were prepared on the steel sheet to ensure the reproducibility of point-probing.

First, the quality of *direct tracking* was assessed. The 14 points were measured with

both, the stationary and the mobile system, the latter being at the pose used for registration (see above) . All measurements of the different optical tracking approaches were transformed to the *FARO* reference coordinate system for comparison, via the registration described above. For the stationary cameras, a residual error of 0.77 mm RMS was measured, the mobile cameras yielded 0.73 mm RMS. Each system on its own thus provides the necessary accuracy.

For *indirect tracking*, the hand-eye problem first had to be solved. Again, the method based on averaging multiple absolute orientations was used (cf. 13.1.2 and 7.2.2). For this, the mobile cameras were brought to five different poses throughout the measurement volume of the stationary cameras and the 10 points used already above where probed for each pose of the mobile cameras.

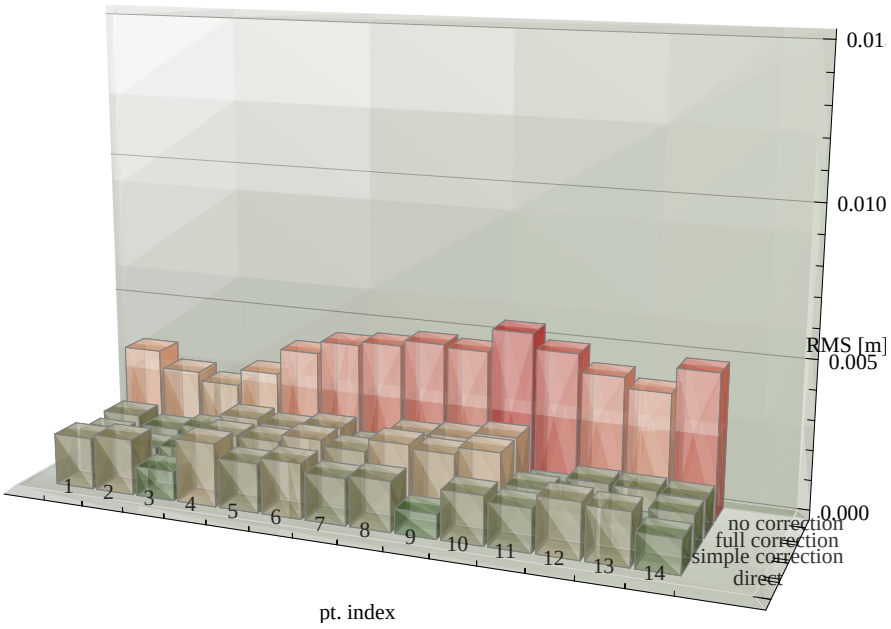
For the indirect tracking measurements, two reference targets were placed in the scene, similar to Figure 13.1. Indirect tracking was performed without any reference targets (no correction), with one (simple correction), and with two (full correction). Six poses of the mobile system were used for measurement acquisition, the latter two of them at a target distance to the measurement volume of 3.0 m.

Figure 13.8 shows the deviations of all 14 points, exemplarily for a near (3rd) and a far (6th) pose of the mobile system. The bars represent the euclidean distance computed for all 14 points, using the different optical tracking modalities with respect to the ground truth measurements of the mechanical arm. With correction, the quality of indirect tracking almost corresponds to that of direct tracking. Surprisingly, for some points (1, 2, 4, 11, 12, and 13), the error is even lower with indirect tracking than with direct tracking. The error of approximately 1 mm measured for these points comes close to the internal accuracy of the stationary system (0.76 mm, see above). Maybe, tracking can even benefit from indirect tracking under certain conditions. A justification for this could be the fact that the large mobile target with 3 cm markers can be tracked better than the small probe. The results for all poses of the mobile system are summarized in Table 13.2.

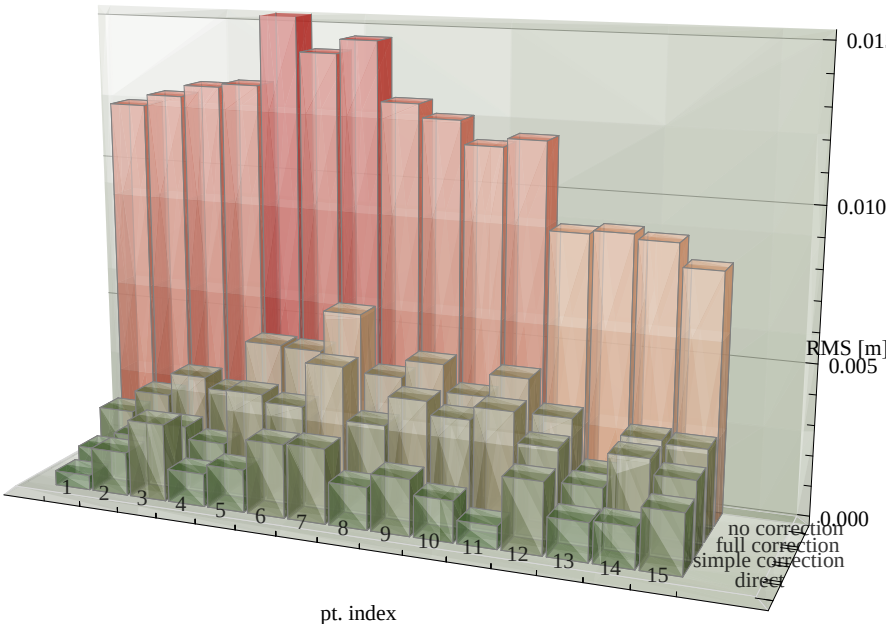
Table 13.2.: RMS error [mm] of rotational correction approaches versus ground truth measurements

Pose / app. distance	Indirect Tracking			Direct Tracking
	No corr.	Simple corr.	Full corr.	
1st / 1.5 m	2.3	2.1	1.9	1.4
2nd / 1.5 m	2.6	1.7	1.6	1.4
3rd / 1.5 m	4.3	1.7	1.8	1.6
4th / 1.5 m	3.2	3.1	3.1	1.5
5th / 3.0 m	13.8	2.5	3.2	1.6
6th / 3.0 m	12.1	3.0	3.8	1.7

Direct tracking reproducibly yields similar results; the different poses of the mobile setup ideally should not have any influence on this. The slight deviation between 1.4 mm in the beginning and 1.7 mm in the end can be traced back to probing the same



(a) 1.5 m distance (3rd pose)



(b) 3.0 m distance (6th pose)

Figure 13.8.: RMS error [mm] of indirect tracking without correction, with simple correction, with full correction, and direct tracking with stationary cameras for two exemplary poses of the mobile system

points several times under repeatability conditions. Maybe, the front vehicle was slightly moved during the experiment.

Interestingly, there is a discrepancy between the results for the static tracking experiment described above which led to a deviation of less than 0.8 mm RMS. They could be a sign of an displacement of the front car between the direct and indirect tracking experiment. The front car was mounted on a dolly that could have been moved easily by an unintentional hit because only two of its four wheels could be locked.

For indirect tracking, the correction methods lead to a significant improvement in all cases. For larger distances of the mobile setup, the deviations slightly increase. This is nothing unexpected since any remaining error in the orientation of the mobile setup propagates to a positional error that increases linearly with its distance. In particular for larger distances, the correction yields a tremendous improvement over the uncorrected indirect tracking and thereby demonstrates the principal correctness of the chosen correction approach. For the first three poses, even the uncorrected indirect tracking features quite low deviations, rendering the advantage of the correction approaches being not so drastic. For the 4th pose , there is no improvement at all from the correction, however the correction does not harm. Maybe, a partial occlusion of one or both reference targets existed, hindering the correction to be really effective.

Since the accuracies of the individual systems is approximately 0.8 mm RMS for both, at a distance of 1.5 m to the measurement volume, the obtained results for indirect tracking with correction are close to what is possible at all. Note that even with rotational correction, the positional errors of both systems still add. In summary it can be said that indirect tracking is not much worse than direct tracking. This is balanced by a much higher flexibility due to the extended tracking range.

13.1.4. Verification of Mobile Tracking Setup

After the simulation concepts described in 11 had become available, they were applied to investigate the until then open question of how to determine the offset between the mobile target and the mobile cameras as well as to simulate the overall behavior of the indirect tracking setup.

Basic Setup

Two variants have been discussed in 7.2.2 to estimate the offset between the mobile target and the mobile cameras. The first is a straightforward hand-eye calibration, the second computes the average of multiple absolute orientations. Unlike in the SRG depcited Figure 13.2 that was used for validation, no shift of the origin of the mobile target to its COG is needed. Rather, the simulation data flow implicitly assumes the origin to be in the COG.

Both variants were evaluated based on the assumption of an isotropic and uniformly distributed error for the transformation between Stationary Cameras and Mobile Target as well as between Mobile Cameras and Probe. The uniform distribution was chosen because of the systematic error behavior of the used tracking systems, see also Figure 10.1(b).

Rough indications from prior experiments were used to derive a 6DoF elementary specification of uncertainty. For the positional uncertainty, the RMS errors from the direct comparison against the *FARO* arm in 13.1.3 can be consulted. They actually describe the uncertainty of the tip of the Probe, 0.77 mm for the static and 0.73 mm for the mobile tracking system. The approximately 0.8 mm RMS is converted by the Perturbation component to a maximum deviation using Equation (10.6), resulting in 1.39 mm. It shall be noted that the large mobile target is probably tracked more precisely than the rather small probe used in the referred experiments; thus, these figures might be too pessimistic for the mobile target.

For the rotational error, a maximum deviation of 0.2° was directly assessed in Figure 10.1(b). It is assumed for both tracking systems. A more fine-graned elementary specification of uncertainty would be desirable, in particular a distinction of the orientational uncertainties between the mobile two-camera and the stationary five-camera setup. Unfortunately, the corresponding ground truth measurements were not available.

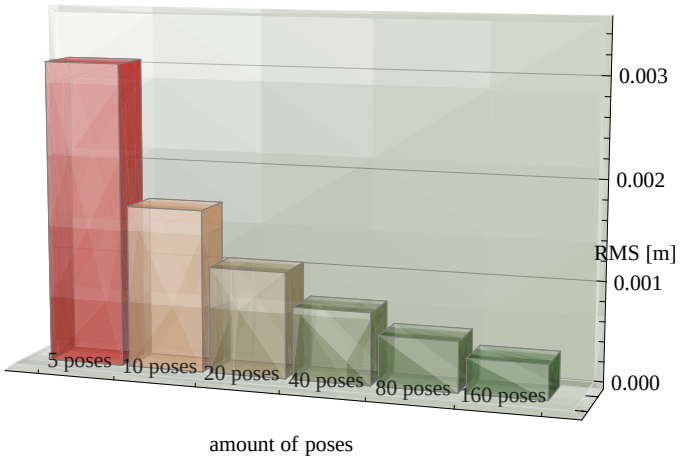
Furthermore, an offset of 0.4 m was assumed between the mobile target and the mobile cameras. The orientation is set to identity for all involved coordinate frames, for error propagation, only the distance has an impact on the resulting uncertainties. Depending on the chosen registration variant, different movements are necessary for the Probe, as well as for the mobile setup consisting of the Mobile Target and the Mobile Cameras. These movements are created on-the-fly in the simulation data flow, based on a common *basic pose*, using the Perturbation component. Further details about theses movements are given below. The basic pose of the Probe/Calibration Object is assumed to be 3 m below the Stationary Cameras. The basic pose of the Mobile Target is assumed to be 2 m below the Stationary Cameras.

The corresponding data flows are described next. They are based on the concepts described in 11.2 and are also available for reference in Appendix A.1.

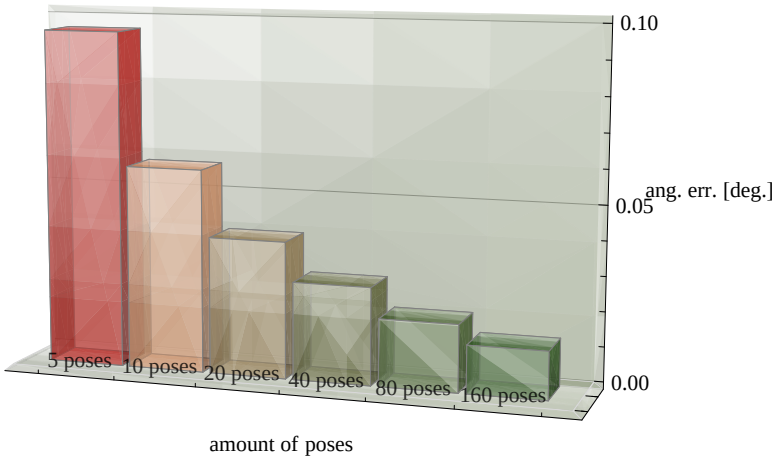
Simulation of Hand-Eye Calibration

Using this method, the Calibration Object is assumed to remain constant at its basic pose (see above). The movement of the mobile setup is generated on-the-fly in the data flow by a perturbation of the basic pose. In successive runs of the data flow, 5/10/20/40/80/160 different poses where sampled using a uniform distribution from inside a unit sphere with a diameter of 4 m and arbitrary orientation. The data flow was reconfigured after each run to increment the number of poses; it was executed five times. 1000 samples where used in each execution of the data flow to estimate the behavior of the registration procedure. Figure 13.9 shows the results.

With an increasing number of poses, the estimation error decreases. The evaluation shows that 20-40 poses represent a good trade-off between registration effort for collection of measurements and the resulting estimation error. With fewer poses, the error is unnecessarily high, more poses only yield a slight improvement.



(a) Positional error



(b) Rotational error

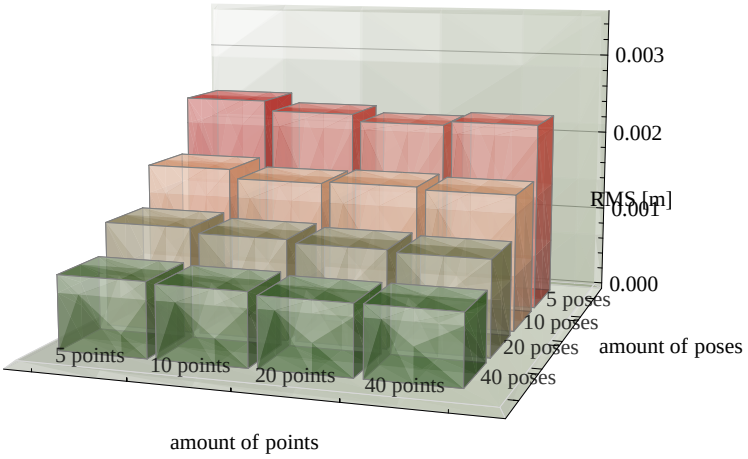
Figure 13.9.: Result of simulation for hand-eye calibration of the offset between mobile target and mobile cameras

Simulation of Absolute Orientation

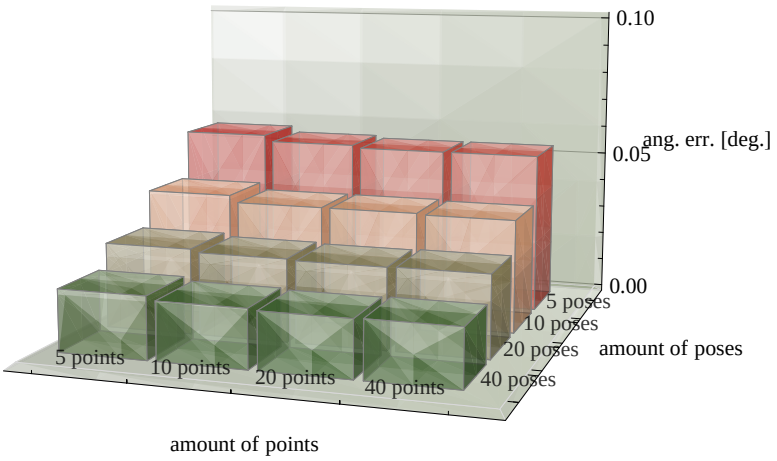
As opposed to the hand-eye calibration variant, the absolute orientation variant depends on mainly two factors, the number of 3D positions measured with the Probe for each computation of the absolute orientation as well as the number of absolute orientations used to compute a mean value. The movements are again generated on-the-fly, by a perturbation of the respective basic pose (see above). The fact that the absolute orientation is performed many times for each pose of the mobile setup is reflected in a nested layout of the corresponding data flow given in Figure A.16 in Appendix A.1. In fact, the Covariance Estimation component is used twice; in the inner loop it is "abused" to compute perturbed samples of the desired mean value, the also computed covariance

is discarded. For nested simulation data flows, see also 11.2.5.

The 3D points to be probed were assumed to be distributed uniformly inside a unit sphere with a diamter of 2 m located at the basic pose of the Probe. The evaluation was performed for 5/10/20/40 corresponding point measurements. For the mobile setup, 5/10/20/40 different poses where sampled, again using a uniform distribution from inside a unit sphere with a diameter of 4 m and arbitrary orientation. Figure 13.10 shows the results.



(a) Positional error



(b) Rotational error

Figure 13.10.: Result of simulation for mean of absolute orientation of the offset between mobile target and mobile cameras.

As expected, position and rotation errors decrease with increasing number of point measurements as well as poses of the mobile setup. However, the benefit from a larger number of poses of the mobile setup is much higher than the impact from an increased number of points.

Conclusion

The evaluation agrees with our initial experiences registering the offset between mobile target and mobile cameras in 13.1.2. Even with a small number of user-triggered point measurements (cf. 7.2.1), the absolute orientation method yields better results for the orientational part. For example, using the hand-eye method, 10 poses of the mobile setup result in a positional error of 1.6 mm RMS and in a standard deviation of 0.058° for the rotation. With the abosolute orientation method, the same 10 poses, paired with only 5 additional position measurements per pose, result in 1.7 mm RMS but only 0.038°. Especially the orientation seems to benefit from the widely distributed reference points.

For practical implementation, the fact has to be taken into account that the stationary and mobile trackers do not belong to the same trigger group; therefore, a convenient discrete collection (cf. 7.2.1) of many pose measurements was not an option and a manual arrangement of the mobile setup is time-consuming. Some user-triggered point measurements, however, are easily obtained.

Whereas the simulation of the two alternative registration methods helps to obtain a deeper understanding, it might not provide reliable quantitative figures for the concrete setup used in 13.1.3, due to the rather rough elementary specification of uncertainty used. At least it can be stated that a the result could have been improved much by more than only 5 poses of the mobile setup for registration.

13.1.5. Verification of Application

For verification of the application, the SRG depicted in Figure 13.2 was embedded into a simulation data flow (see also Figure A.20 in Appendix A.3) according to the methods described in 11.

The same elementary specification of uncertainty was used as above in 13.1.4. Furthermore, the simulation results from 13.1.4 were reused for the offset between mobile target and mobile cameras. During validation (see 13.1.2), 5 poses where used, with 10 point correspondences each. According to Figure 13.10, this yields a positional error of 2.3 mm RMS and a rotational error of 0.055°. These values are used to perturb the offset between the mobile target and the mobile cameras again.

As above, the assumed offset between mobile target and cameras is again 0.4 m. Furthermore, the distance between the mobile cameras and the probe was assumed with 1.5/3.0 m in horizontal direction and between the stationary cameras and the mobile target with 3.0 m in vertical direction. The positions of the two reference targets are also varied. Twice, they were placed close to the probe, 0.5 m closer to the mobile cameras than the probe itself, thus at 1.0/2.5 m distance. Once, they were placed far away at 1.0 m while the probe was at 3.0 m. The distance between the reference targets was constantly set to 1.0 m.

The described ground truth data is provided in terms of the SRG edges between Stationary Cameras and Probe, Stationary Cameras and Mobile Target, and Mobile Target and Mobile Cameras. The forth transformation between Mobile Cameras and Probe is

derived from the other three edges by closing the loop in the SRG, to obtain a consistent set of ground truth data. Table 13.3 shows the simulation results for the described assumptions. Both correction approaches reduce the errors of indirect tracking, full

Table 13.3.: Simulated RMS error [mm] of rotational correction approaches

Probe dist. / ref. target dist.	Indirect Tracking		
	No corr.	Simple corr.	Full corr.
1.5 / 1.0 m	4.2	2.3	2.0
3.0 / 1.0 m	6.4	4.0	3.5
3.0 / 2.5 m	6.4	2.1	1.9

correction performing slightly better than simple correction. These results will now be discussed in more detail.

13.1.6. Discussion

Only a rough elementary specification of uncertainty was used for verification. Nevertheless, the simulation results in Table 13.3 come quite close to the empirical results in Table 13.2, at least when the probe is 1.5 m away from the mobile cameras. The position of the reference targets seems crucial for the performance of rotational correction, as indicated by the second row. This somehow foils the idea of indirect tracking for the reduction of occlusion effects; at least, indirect tracking cannot reach very deep into occluded areas without loosing too much accuracy.

This might also explain why the simulated uncertainties for a long distance between probe and mobile cameras of 3.0 m are better (up to 1.9 mm with full correction) than in in-situ validation ($>$ 3.1 mm with full correction). In the latter experiment, the probe was used at distances from the reference targets that were significantly larger than 0.5 m for most of the measured points, up to 1.5 m for some.

Nevertheless, the verification does not really explain why indirect tracking without correction performed so badly ($>$ 12 mm) over a distance of 3 m during validation. There are three possible explanations:

1. The registration of the mobile target and the cameras could accidentially have been corrupted during the experiment, especially the orientational part of that pose. This would explain the constantly bad results for indirect tracking (mostly $>$ 3 mm) with correction during the second half of the experiment (fourth/fifth/sixth pose in Table 13.2) as compared to the first half ($<$ 2 mm). It would also explain the bad performance for indirect tracking without correction for the last two poses. It would however not explain why indirect tracking without correction for the forth pose (3.2 mm) was not much worse than for the first three poses (2.3/2.6/4.3 mm); it should have also been affected.

2. Therefore, it is also reasonable that one of the reference targets was badly tracked by the stationary cameras during the second half. It could have been touched

during the experiment, resulting in merging markers or some of the spheres moving fully or partially outside of the tracking volume of the stationary cameras. Note that the mobile cameras were moved, which would probably have resolved such a problem with the next pose. A badly tracked reference target could explain the annihilation of the correction effects for the forth pose and a degradation for the far aways fifth and sixth pose. After all, the simulation also indicates that the correction works better for longer distances between mobile cameras and probe/reference targets. However, the huge error for indirect tracking without correction for the fifth and sixth pose remains curious.

3. The empirical measurements for indirect tracking without correction (cf. Table 13.2) suggest a lower positional error but a higher rotational error for the offset between mobile target and cameras. The lower positional error could be explaned with the rather huge dimensions of the mobile target and the individual spheres mounted on it; the elementary specification of uncertainty for positional error was based on the rather small pointing device. A higher rotational error on the contrary could also be explained. Unlike in the simulation and unlike in Figure 10.1(b), the pose of the mobile setup was not varied freely in the real setup. Due to practical reasons, in particular the used tripod and the fact that the mobile target had to be visible for the stationary cameras, the orientation could only be changed by approximately $+/-30°$ about the yaw and pitch axes. Some of the systematic effects therefore might have been missed. See also 10.1.1.

Probably, a combination of several effects was at work during validation.

An interesting aspect is the equal performance of the simple correction during validation (cf. Tables 13.1 and 13.2), as compared to the full correction. The simulation however shows an advantage for the full correction, as one might intuitively assume. However, the flat and lengthy layout of the mobile target was not implemented in the simulation, due to the elementary specifiation of uncertainty having a 6DoF granularity. On that condition, the simple correction could have performed better during verification.

For both correction approaches, coordinates were shifted to the COG of the mobile target. Our recent findings suggest however that the COG is not ideal [Pust 10]. Neither does the COG represent the point with the lowest positional uncertainty nor does it provide a good separation (in terms of low correlation) between positional and rotational uncertainty. See also 11.4. The correction approaches should be revisited in this respect.

To conclude, this subsequent verification is useful to understand some effects but it quickly reaches its limits. The validation would certainly have been conducted with more attention on certain aspects, especially the positions of the reference targets or the registration of the offset between mobile target and cameras. Furthermore, the 6DoF elementary specification of uncertainty seems to provide only a rough approximation. Ideally, a proper assessment of the elementary uncertainties should be the first step. Next, a subsequent verification step should be conducted to reveal important aspects for accuracy. Then, validation can be performed, with the critical aspects properly defined for maximum explanatory power of the results.

13.2. Example: Probe Tracking in the Airplane Cabin

Now, the accuracy of quantitative discrepancy checks using a metrological probe in the airplane cabin (see 7.1.2) is analyzed [Keit 10b]. First, the expected performance in the target environment is verified in 13.2.1. This step is based on exhaustive simulation and can happen even before actual deployment of the hardware. Having this, the application is validated in 13.2.2, based on some selected measurements in the target environment. Both steps also include the relevant registration steps for the reference target (see 7.2.3).

13.2.1. Verification

Using the simulation setup, we do not only want to analyze a specified setup. Also, critical design issues and purchase decisions are investigated in more detail. For verification by simulation, we decided to use a non-isotropic Gaussian error distribution to approximate the real error distribution, following the vendor's elementary specification of uncertainty with an overweight error in the depth direction. It has been shown above in 12.4 that this is a reasonable elementary specification of uncertainty.

Simulation Setup

First of all, the level of abstraction in the elementary specification of uncertainty requires to explicitly formulate the tracking algorithms needed to determine the 6DoF pose of the Probe and Reference Target in real-time. They are actually performed inside the black-box tracking system but have to be reproduced such that the sensor uncertainty description for the individual LEDs can be used. For this, the SRG depicted in Figure 7.3 has been refined, resulting in Figure 13.11.

The bold edges are obtained by an absolute orientation each, according to 7.2.1. They emulate the black-box marker tracking algorithms.

Based on this, simulation data flows were created, using the concepts described in 11. Since the simulation has to cover the entire error chain, not only the application data flow (cf. 7.1.2) but also several registration data flows (cf. 7.2.3) have to be incorporated. The measurements for tracking the Probe Tip in the World (actual application data flow) are depicted in red. The calibration of the offset between Reference Target (CAD) and Reference Target (LED) is nested in the same SRG according to 11.2.5), unlike in 13.1. The corresponding measurements for this task are depicted in orange. The remaining static offsets depicted by green edges are taken from external sources.

The calibration of the Tip Offset was performed with the proprietary software of the tracking system vendor. The routine also computes an RMS value for the tip; it can therefore be perturbed accordingly in the monolithic main simulation data flow. Similarly, the Offsets of 4 Probe Marker LEDs, 6 Reference Target LEDs, and 25 Tactile Points are simply taken from the body calibration files and CAD model, respectively. No error was assumed for the manufacturing of the reference target from its CAD model.

The World Registration of the Reference Target (CAD) with the World (aircraft) was simulated externallyl, using the method described in 11.2. For the laser tracker, the specified uncertainties (see above) were used.

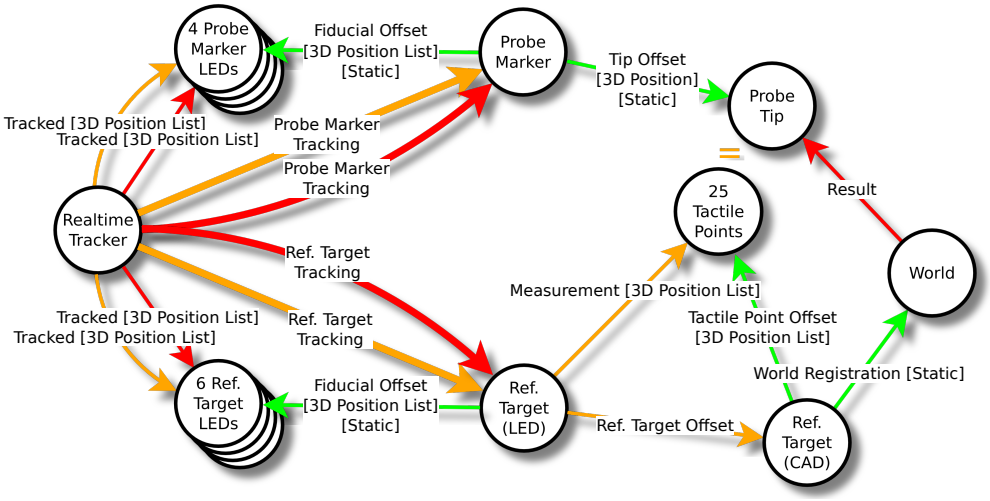


Figure 13.11.: Refined version of the SRG shown in Figure 7.3. The pose estimation based on visible IR LEDs is explicitly modeled (bold edges). The SRGs combines the spatial relationships for tracking of the Probe Tip in the World at runtime (red measurements) with the spatial relationships needed for calibration of the offset between Reference Target (LED) and Reference Target (CAD) (orange).

An outline of the main simulation data flow algorithm is depicted in Algorithm 1. The complete data flow can be found in Figure A.14 in Appendix A.3.

For the calibration of the reference target (orange edges in SRG), the Realtime Tracker is virtually positioned such that the reference target is at an optimal distance (< 2 m) for minimal LED tracking error. To apply the noise in our setup, we use the ground truth data of the Probe, keeping the tip fixed on one of the 25 Tactile Points. We derive the location of the 4 Probe Marker LEDs and the 6 Reference Target LEDs in the coordinate frame of the real-time tracking. There, Gaussian noise is applied to each LED position. Using an absolute orientation algorithm, error-prone estimates for the Probe and the Reference Target (see above, cf. Figure 13.11) and consequently for the 25 Tactile Points and the Probe Tip are computed. Iterating over the tactile points, we derive 25 corresponding 3D point pairs (inner loop in Algorithm 1). The erroneous calibration of the offset between Reference Target (LED) and the Reference Target (CAD) is then obtained by another absolute orientation between the 25 Tactile Points as given in the CAD model and as measured by the real-time tracking.

A newly perturbed sample of the calibrated offset is incorporated in an enclosing simulation loop, for the generation of samples for the runtime system (red edges in SRG). We specify several poses of the Realtime Tracker with respect to the Reference Target (LED, as well as grid of assumed Probe Tip poses in the tracking volume. The simulation iterates over these poses, perturbing the pre-calibrated transformations as well as of each of the currently visible LED positions to sample the covariances of the

Algorithm 1 Outline of data flow algorithm for the simulation of the uncertainty of tip tracking in the target environment. All relevant errors are considered.

Require: Ground truth data: positions of Reference Target LEDs w.r.t. Reference Target, positions of Probe Marker LEDs w.r.t. Probe Marker, position offset of Probe Tip w.r.t.Probe Marker, positions of 25 Reference Points w.r.t. Reference Target, pose of Reference Target w.r.t. World, pose(s) of Realtime Tracker w.r.t. Reference Target, pose(s) of the Probe Tip w.r.t. the Real Time Tracker, assumed poses of Probe Tip.
Uncertainties: elementary specification of uncertainty for 3D position of a single LED, covariances of Reference Target LEDs w.r.t. Reference Target (from body calibration), covariances of Probe Marker LEDs w.r.t. Probe Marker (from body calibration), covariance of Probe Tip w.r.t. Probe Tip (from tip calibration), covariance of Reference Target w.r.t. World (from prior simulation).

```
for all defined poses Realtime Tracker w.r.t. Realtime Tracker do
    Derive g.t. Reference Points w.r.t. Realtime tracker
    Initialize list of covariances
    for all defined poses of Probe Tip do
        Initialize covariance
        for count = 1 to 1000 do
            Perturb Reference Target LEDs w.r.t. Reference Target
            Derive g.t. Reference Target LEDs w.r.t. Realtime Tracker
            Perturb Reference Target LEDs w.r.t. Realtime Tracker
            By Absolute Orientation, derive perturbed pose of Realtime Tracker w.r.t. Reference Target
            for all 25 Reference Points do
                Perturb Probe Marker LEDs w.r.t. Probe Marker
                Derive g.t. pose of Probe Tip w.r.t. Realtime Tracker for Reference Point
                Derive g.t. Probe Marker LEDs w.r.t. Probe Marker
                Perturb Probe Marker LEDs w.r.t Probe Marker
                By Absolute Orientation, derive perturbed Probe Marker w.r.t. Realtime Tracker
                Perturb Probe Tip w.r.t. Probe Marker
                Derive perturbed Reference Point w.r.t. Reference Target (LED)
            end for
            Derive perturbed Reference Target (LED) w.r.t. Reference Target (CAD)
            Perturb Reference Target LEDs w.r.t. Realtime Tracker
            By Absolute Orientation, derive perturbed pose of Realtime Tracker w.r.t. Reference Target
            Derive g.t. pose of Probe Marker w.r.t. Realtime Tracker
            Derive g.t. positions of Probe Marker LEDs w.r.t. Probe Tracker
            Perturb Probe Marker LEDs w.r.t. Probe Marker
            Perturb Probe Marker LEDs w.r.t. Realtime Tracker
            By Absolute Orientation, derive perturbed pose of Probe Marker w.r.t.
            Realtime Tracker
            Derive perturbed Probe Tip w.r.t. World
            Update covariance
        end for
        Append covariance to list of covariances
    end for
    Output list of covariances
end for
```

Probe Tip at the different positions in the tracking volume.

Verification Experiments & Results

The simulation results of 12.4 already pointed out that a reference marker dimensioned to 300x300 mm^2 seems to be a good optimum to get a high accuracy and still allow handling in the aircraft.

Next we analyzed the world registration procedure using the *FARO* metrological system that had an RMS error of 0.018 mm in our test case. To understand the impact on the entire application, we use the simulation to propagate this error to the tip in world coordinates. For different numbers of samples used for the registration procedure, the resulting error in the tip is displayed in Table 13.4. When comparing the resulting error for 4, 6, and 8 samples, it seems sufficient to rely on 4 since the over determination allows one to identify user errors and the resulting error does not influence the entire system error strongly.

Table 13.4.: Propagated RMS errors [mm] for the world registration

number of points	min. error	max. error
3	0.0790	0.690
4	0.0590	0.488
6	0.0547	0.475
8	0.0474	0.405

For the registration between the reference marker and it's CAD model we depend on the less accurate real-time tracking system with an RMS point error of 0.2 mm, computed from the vendor specification using (10.14) assuming 2 m depth. To understand the impact on the entire application, we again use the simulation to propagate this error to the tip in world coordinates. In contrary to the prior analysis and due to the reduced accuracy we decided to rely on more samples and to compare different designs of the probing device. Table 13.5 indicates that for the registration of the reference target the type of probe is less important.

Table 13.5.: Propagated RMS errors [mm] for the reference target

used probing device	min. error	max. error
small probe (5x5 cm^2)	0.208	2.23
large probe (10x10x5 cm^2)	0.192	2.11

A typical setup we want to validate has the reference target in the far field (Figure 12.5). Following the results from Figure 12.9, we prohibit the use of the probe too far away from the reference target and therefore clip the front part of the pyramid to avoid occlusion and a too high error value.

As Table 13.6 illustrates, the error in the worst case estimation of our scenario ranges between 0.79 mm and 4.86 mm. As already stated, the error increases with the distance

Table 13.6.: Simulated RMS errors [mm] for the tip position

simulation setup	min. error	max. error
far reference target	0.789	4.86
far reference target (pro)	0.154	1.43
far multi target	0.710	1.57
far multi target (pro)	0.137	0.377

between probing device and reference target, as in Figure 12.9(b). This becomes clear when we use four small 3 LED reference targets distributed in the working volume (multi target); the maximum error drops to 1.57 mm whereas the minimum error is nearly unchanged. This coincides with the basic rule in metrology to always measure inside the cloud of reference points used for registration [Niem 08]. Also we simulated the expected error for an alternative "pro" version of the hardware[4]. It has a much lower uncertainty; in the horizontal/vertical/depth directions, it drops to 0.02/0.02/0.06 mm (at 2 m) and 0.06/0.06/0.15 mm (at 6 m), compared to the 0.1/0.1/0.15 mm at 2 m depth, 0.15/0.15/0.25 mm at 4 m depth, and 0.25/0.25/0.45 mm at 6 m depth of the normal hardware.

13.2.2. Validation

As proposed we validate our simulation results in the real environment. Starting with the noise of the direct and indirect probe tip tracking. In the direct probe tracking the results matched the expected values and the standard deviation for various measurements was 0.03 to 0.04 mm.

When measuring the tactile points of one of the reference targets, the back projection error was too large. Recalibration with the 25 tactile points solved that issue. Also for the just calibrated reference targets, the accuracy was as expected, the error ranged from 1 mm to 3 mm with a standard deviation of approximately 0.5 mm over 100 samples - depending on the distance between reference target and probe. If the application requires a precision below 1 mm, the more expensive "pro" system needs to be used. Also, the degradation in the quality of the calibration of the reference targets over time has to be validated in a more detailed system analysis.

13.2.3. Discussion

Since complex tracking scenarios cannot be covered by a generic, application-level uncertainty specification, an elementary specification of uncertainty has to be provided for each of the involved sensors. By a propagation throughout the complete chain of errors, an application-level uncertainty specification can be obtained. Unlike the elementary specification of uncertainty which has to be provided for a certain kind of sensor only, the application-level uncertainty specification depends on the concrete setup.

[4] *NDI Optotrak Pro System*

It has been demonstrated that using a simulation framework, design decisions in the definition phase of an industrial tracking system become more transparent. Various different hardware platforms and concepts can be easily benchmarked even without the need to use real hardware. Furthermore, the task of validation in the target environment could be simplified tremendously, since many critical aspects could be solved before the actual deployment.

14. Runtime Error Mitigation

Error mitigation strategies are needed in the operation & maintenance phase of the IAR application. Most tracking setups change physically over time, due to various reasons such as temperature changes, fatigue of material, or (un)intentional mechanical influences. Therefore, while the tracking system is operational, one would like to know whether static transformations from previous registration steps are still valid. Furthermore, registration methods have to be robustified such that they can be used by on-site personnel.

14.1. Robustifying Calibration & Registration Procedures

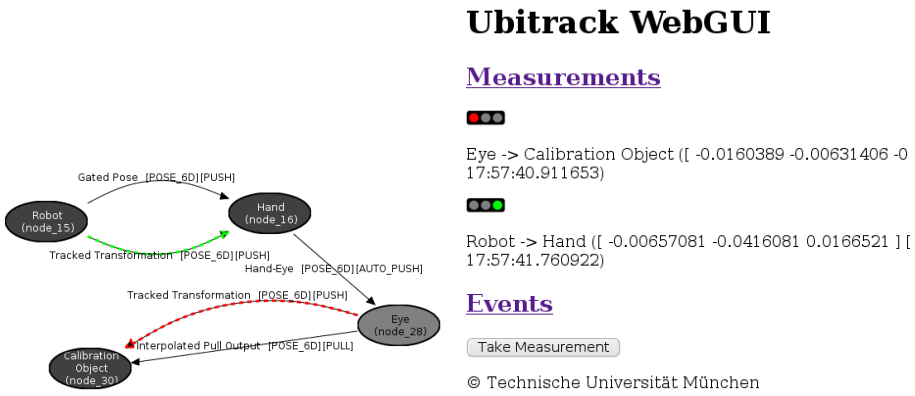
trackman provides a generic means to carry out registration procedures including the necessary user interactions. SRGs can be directly instantiated in *trackman* such that no additional implementation is necessary to solve the registration problem. A binding for the Application Push Source Button components is provided for measurement acquisition. Various data flows have been discussed in 7.2.

Especially for optical tracking systems, it is desirable to have feedback in the user interface about the current refresh rate or timeliness of the last measurement received. For example when using a probe such as in Figure 1.6(b) to collect point measurements, this is of utmost importance. It is not sufficient to merely ensure that the probe tip is at the correct position when the user presses the button. Additionally, it also has to be ensured that preferably all fiducials are seen by the cameras to obtain a good measurement. Intuitive tools are needed, tailored to the target user and also the IAR application to adhere to this simple constraint. Examples are depicted in Figure 14.1.

Currently missing observations are indicated in red. The user has to actively remedy such sitations, possibly by removing occlusion conditions. The timing of the visualization has to be adapted to the update rate of the involved sensors.

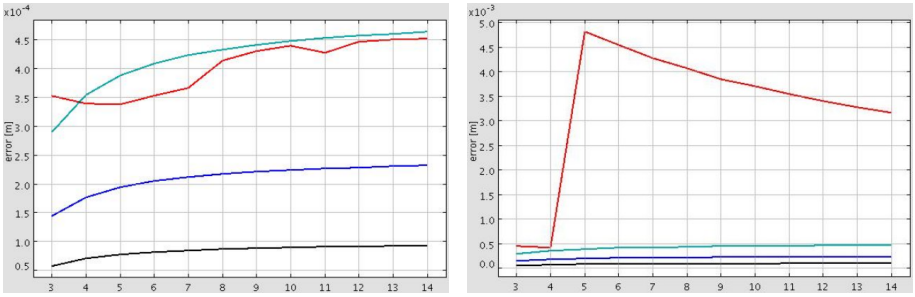
Also, information about the quality of the current measurement residual can be displayed. It can be observed while collecting measurments to see whether it converges against a reasonable value, see also 12.3.2. This is shown exemplarily for the absolute orientation in Figures 14.2(a) (expected convergence) and 14.2(b) (outlier).

Despite these efforts, it remains a huge potential for improvements here, especially in manual measurement acquisition for registration procedures. This becomes even more crucial in case of several interdependent registration steps. To accomplish this tedious task efficiently, all involved measurements have to be very accurate and outliers should be avoided.Therefore, it is desirable to incorporate statistical tests (cf. 10.3.3) which provide immediate feedback about the consistency of measurement data. Resulting actions can



(a) *trackman* (b) Web front-end

Figure 14.1.: Availability of Tracking Data. All relevant transformations should be green. Red indicates currently missing observations.



(a) Expected convergence (b) Outlier

Figure 14.2.: Behavior of residual error in point-based registration. The expected characteristics is plotted for several apriori uncertainty levels. The actual convergence (red) is plotted on-the-fly, as new measurements are taken.

be the exclusion of single measurements or also the repetition of the whole measurement sequence.

14.2. Example: Loops in the Spatial Relationship Graph

During runtime, loops in the SRG can help to perform online consistency checks. The basic idea is to exploit redundancies in the SRG, similar to how over-determination of a problem is exploited in adjustment theory to perform hypothesis tests on the input data [Koch 97, Niem 08], see also 10.3.3. A similar strategy is used with coordinate measurement machines (CMM) and laser trackers (see 1.2): from time to time, the tip is moved to a dedicated calibration sphere mounted rigidly to the machine in order to

assure that the tip calibration is still valid and the CMM is working properly.

Assuming that a certain transformation is given by following two different paths in the SRG, the two alternatives should ideally not deviate from each other. As an example, consider the SRG depicted in Figure 13.2. It contains four different ways to track the pointing device in the world. The deviation between these paths does not only reveal how good indirect tracking with or without rotational correction works, it is also an indicator for potential inconsistency in the setup [Keit 10a]. An increase suggests that static transformations have changed over time and a re-calibration/re-registration of the setup becomes necessary.

A comparison of two alternative SRG paths has been implemented exemplarily in *trackman*. Again, a dedicated user interface would be required in a productive environment. The reference measurement is thereby retrieved via an Application Push Sink Pose, in our example the result obtained from direct tracking (cf. 13.1.2). The alternatives are retrieved via Application Pull Sink Pose components to ensure synchronized measurements (cf. 5.4). They are compared relatively to the reference measurement. An exemplary plot captured during in-vitro verification of indirect tracking (cf. 13.1.2) is shown in Figure 14.3.

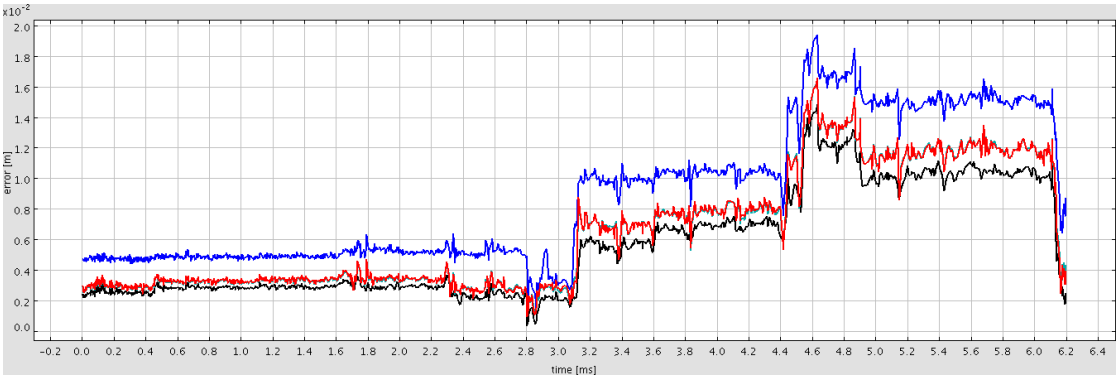


Figure 14.3.: Exemplary comparison plot showing positional deviations of three indirect tracking variants (blue: no correction / red: simple correction / black: full correction) from direct tracking. As the probe is moved further away from the mobile cameras over time, the deviation with respect to the reference increases.

The x-axis of the plot represents time. The y-axis represents positional deviations (euclidean distance) of the three alternative paths (indirect tracking variants) in the tracking data flow relative to the reference path (direct tracking). Similarly, orientational deviations can be plotted based on the axis-angle representation (see Equation (10.18)).

14.3. Example: Error Mitigation in the Airplane Cabin

In the next example, we go one step further. The goal is not only to detect potential system failures, but to describe strategies to trace back and eliminate them efficiently and reliably. The expected accuracy and precision have been analyzed in the verification and validation procedures (cf. 13.2). This allows one to define constraints within the tracking setup's SRG to compare current measurements with the expected system behavior at run time.

The simplest way to perform a run-time test is a ground truth measurement. When probing a known point it is possible to compare the measurement to the expected value. Using the same statistical tests as during validation (cf. 13.2.2), it is possible to determine whether a measured point set violates the expected confidence interval. In case the user can reproduce the effect, it is not caused through a user error like imprecise probing as for example slipping or measuring the incorrect point. This indicates a malfunction.

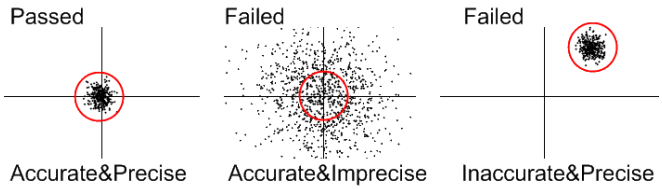


Figure 14.4.: Runtime checks, ◯ represents the confidence interval.

To locate the error source in such a complex setup, it is possible to run the validation tests simply in reverse order. The first step validates the probe tracking in the complete system. Measuring with the probe tip on a known fixed point (ground truth) while rotating the probe results in a position and a covariance ellipsoid of the tip in the world.

The resulting measurement can violate the specified error level in two ways. In case the covariance is larger than the simulation predicts, the source of error should be in the direct probe tracking (Figure 14.4, accurate & imprecise). Recalibration of the probe target and the tip calibration should solve that issue. Otherwise, the error has the correct magnitude of noise but the measured position is incorrect (Figure 14.4, inaccurate & precise).

In the latter case a second test needs to be performed by probing at least three known tactile points on the reference target and comparing the measurement to the ground truth values from the CAD model in the coordinate frame of the reference target. If these points violate the confidence interval, this indicates that the transformation between the 25 tactile points and the marker of the reference plate is invalid. Recalibration of the reference target solves this issue.

If the test is passed and the points are correctly mapped, the only left registration is between the reference marker and the aircraft (world). A recalibration of this reference using the offline metrological system is necessary.

After this procedure, the final system test is repeated by measuring a known point in the aircraft. In case the distance between the measured point and the ground truth

exceeds the confidence interval, there are two severe error sources left. Either the offline metrological system was not registered properly to the aircraft in the calibration procedure or the real-time tracking system is outside its specification.

Performing this procedure at runtime asserts that the tracking system and the work procedures relying on the new application are in line with the specification. This corresonds to repeately probing the calibration sphere with a CMM or laser tracker, as described above in 14.2.

15. Discussion

From the review of standards available in the field of industrial metrology and measuring technology in general, this thesis has formulated an approach for the treatment of uncertainties in IAR tracking environments. The approach is based on a combination of extensive verification by simulation paired with selected measurements in the target environment of the IAR application. The simulation was integrated in the SRG and *Ubitrack* concepts to benefit from the graphical modeling facilities described in Part II.

It has been shown that an elementary specification of all uncertainties is needed for the simulation to yield realistic results. This specification should be as generic as possible, for maximum reusability in similar scenarios and also for comparability of competing accuracy assessments. Furthermore, it should incorporate both, systematic and random errors, though without the need for an explicit separation. Methods have been described to derive such generic specifications, on the example of optical IR tracking. The approach has been demonstrated and shown to be useful in conjunction with two industrial scenarios. However, the specification should also be specific enough to describe adequately the behavior of the system. The specification based on 3D uncertainties yielded much more accurate results than the specification based on 6DoF uncertainties. The author hopes to motivate others to promote this generalization approach by formulating uncertainty standards also for other types of sensors.

Additional complexity arises when looking at heterogeneous systems. In the considered cases, only IR tracking systems have been considered. There was no need to consider temporal calibration of tracking systems since either only one system was used, or they were synchronized in hardware. Furthermore, all registrations and all measurements were conducted with manual fixation to exclude the negative influence of lag. This is not a restriction for interactive measurements such as for quantitative discrepancy checks were objects have to be probed anyway but it is a restriction for classical AR applications with interactive visualization. In general, combining measurements with different timestamps requires inter-/extrapolation and therefore the error management would have to be extended, too.

The described simulation framework was implemented mainly on the *Ubitrack* data flow layer. This allows for graphical configuration using the *trackman* tool described in II. However, standardized functionality for the preparation of ground truth data, as well as the for the evaluation of simulation results are still missing. *Mathematica* notebooks and other scripts were used for pre-/post-processing instead. A graphical tool on the middleware layer would be desirable. It should allow one to arrange tracking devices, markers, and other entities of the SRG directly in a 3D environment and also present the simulation results in this environment.

It also turned out that modeling nested simulation data flows quickly becomes awk-

ward and difficult to debug. The data flow editor could provide graphical support for nesting operations. As Figures 11.3 and 11.4 suggest, it would already help to present different "push-paths" in the data flow with distinct colors. The meta-pattern concept described in 8.2 could be extended to classify sets of patterns according to their affiliation with such "push-paths". Certain levels of detail could then be investigated in detail or hidden completely.

An even more general approach could be to equip the middleare simulation tool with a dedicated simulation workflow engine. In 4.2 it was claimed that the data flow is stateless and that logic shall be dealt with on the middleware layer. This principle has been somewhat softened by allowing for nested loops, though branches are still not possible. The simulation workflow engine should be intertwined with the interactive setup of the environment in 3D and visualization of the simulation results. Ideally, it would automatically take care of the generation of synthetic measurements and the proper sequence of data flow algorithms, depending on the user's needs.

Part IV.

Conclusion

To conclude, the demonstrated methods are now discussed according to their eligibility to solve the problems discussed in Part I. This encompasses the technical requirements towards tracking in IAR setups in 16. Aspects regarding the integration into industrial processes are reviewed separately in 17.

16. Fulfillment of Tracking Tasks

In 2.3, the technical requirements towards tracking in IAR setups have been discussed.

16.1. Guaranteed Performance

The "guaranteed performance" of the system comprises a warranty for its reliability, its robustness, as well as its accuracy & precision.

To ensure reliability and robustness, concepts have been described in 14 that aim at continual consistency checks at runtime, mainly based on loops in the SRG. In 14.3, also ideas for a systematic traceback of system failures have been given. However, much work remains to be done in this field.

Much effort has been put into pointing out relations to industrial metrology and adjustment theory. For IAR to be successful, trusted methods from this field have to be adopted and IAR has to be dovetailed with existing processes. Unfortunately, real-time tracking equipment is often more complex to handle than offline metrologic equipment, due to the odds described in 1.4. Available standards from metrology have to be interpreted and adapted to IAR. As an important contribution to this problem, the elementary specification of uncertainties has been introduced and shown to yield realistic results in industrial scenarios. Hopefully, system vendors will adopt such a standardized approach in the future.

A major contribution also is the description of an end-to-end error propagation framework based on Monte Carlo simulation (cf. 11). It is based on the elementary specification of uncertainties and gives concrete statements about the expected performance of a proposed system (verification). It has been shown in 12.4 and 13 that the simulation is consistent with empirical measurements in the target environment (validation). The combination of verification and validation thus is an integrated means to provide a guaranteed level of accuracy & precision.

To increase the realiability of (re-)registration procedures, the importance and restriction of measurement residuals has been pointed out in Chapter 10. This should be better integrated in the future with error propagation techniques and guidelines to collect meaningful point/pose correspondences. Also outlier detection and global hypothesis tests from adjustment theory should be incorporated to increase robustness.

16.2. Sensor Fusion

Mainly cooperative and complementary fusion approaches have been treated. Competitive fusion could also be helpful for IAR but has been neglected in this context. This

reflects the necessity of an integration of offline metrology and real-time tracking. It has to be solved before complementary fusion can further increase the tracking performance.

The major contribution here consists of providing a graphical editor to quickly implement the registration and runtime data flows. Particularly registration quickly becomes complex to handle. With *trackman*, setups of unprecedented complexity can be constructed with relative ease.

16.3. Modularization

Modularization is key to the provision and reuse of common best-practice solutions. This in turn is needed for a maximum flexibility regarding the ad-hoc installation and removal of components, including the implied calibration and registration procedures.

The modularization is accomplished to a big part by *Ubitrack* already. A *Ubitrack* data flow runs almost as efficiently as a hard-wired solution. Patterns can be used either in their time-expanded or in the implicitly/explicitly space-expanded form. Furthermore, trigger components adapt themselves to various constellations of synchronization on their input ports. This is a huge source of flexibility (cf. 5.3). *trackman* maps this flexibility to a user interface that allows one to model new data flows with relative ease, making use of the *Ubitrack* components and their flexible expansion variants.

A distinct lower level of modularization is introduced at a higher level of abstraction by the meta-pattern concept (cf. 8.2) that allows one to provide best-practice solution patterns for recurring registration and tracking problems. Various such solutions patterns have been described throughout this thesis, e.g., in Chapter 7.

The modularized approach also eases wrapping any existing registration of tracking data flow in a Monte Carlo simulation data flow (cf. 11.2). Though the provision of ground truth data and the analysis of the simulation results are not yet solved in general for arbitrarily complex setups (cf. Chapter 15), the provided methods are already quite useful in many situations (cf. Chapter 13).

16.4. Maintainability

Maintainability during the definition and deployment stages mainly affects expert IAR-engineers. They are supported by a graphical SRG and data flow editor that highly reduces the effort to realize a planned tracking setup (cf. Chapter 6). The proven spatial correctness inherent to the pattern approach spares additional time for troubleshooting. The fact that some aspects (in particular related to data flow synchronization, cf. 5.4) cannot be intuitively represented on the SRG level, is mitigated by a round-trip engineering approach (cf. 6.4.3 and Chapter 9).

In the operation & maintenance stage, support for non-expert users is needed. They should be able to operate setup and dismatingling procedures including the implied calibration and registration procedures. Furthermore, they should be able to verify the proper functioning of the system at all times. The challenge here is to find generic solution patterns. It is beyond question that a tailored software solution can meet these

criteria. It is also beyond question that in a productive environment with non-expert users, a tailored software solution is needed. The question rather is how the creation of a tailored solution can be supported by generic best-practice solution patterns. Even though several ideas are sketched throughout this thesis, no comprehensive answer has been given so far. Ideas are sketched in the following.

To support calibration and registration procedures, the residual error resulting from over-determination surely provides valuable information (cf. 12.3). Furthermore, the number and distribution of measurements has to be controlled (cf. 10.3.3). Both concepts can be found already in some specialized proprietary calibration routines of individual system vendors. A more standardized approach would be desirable. Regardless the chosen algorithm, a generic visualization of the convergence of residual error according to Figure 14.2) can be provided to the user. What remains is a suitable guidance for the acquisition of measurements, which heavily depends on the algorithm and involved sensors; it cannot be handled in a generic way.

Unlike the calibration of an individual sensor, the registration of several sensors to one another (cf. 7.2) incorporates a much higher risk of blunders during measurement acquisition. This problem has already been solved in the field of metrology, by means of a plethora of hypothesis tests that can be applied individually to single measurements as well as globally to obtain a conclusive statement. These relationships have only been mentioned briefly in 10.3. However, such strategies are typically applied by experts during offline analysis of measurement data. More work is necessary in this area to adapt the strategies to intuitive online procedures.

Finally, concepts for runtime surveillance have been provided, based on loops in the SRG (cf. 14). Concepts for a systematic troubleshooting and traceback of errors have been demonstrated exemplarily for the airplane cabin scenario (cf. 14.3).

17. Compliance with Industrial AR Design Guidelines

In 2.3, phases in the implementation of new productive processes have been discussed. For AR to spread in the industrial domain, support is needed during all these phases. Answers have been given mainly in Part III.

First, an elementary specification of uncertainties is needed for all involved sensors (cf. 10.3.5 and Chapter 12). It has to incorporate systematic and random error and shall be as generic as possible and as specific as necessary. Methods for its acquisition have been discussed in relation to existing standards for metrologic devices and demonstrated for the case of optical IR tracking. It could be shown that the methodology of the GUM to subsume systematic and random error in one quantity yields reasonable results. An adaption to other sensors and measurement types remains to be done.

Based on the elementary specification of uncertainties, the definition phase can be supported by extensive simulation (cf. Chapter 11). By this means, critical questions of detail as well as the overall setup can be investigated before the actual purchase decision. This highly simplifies the verification process.

During deployment, the installed setup has to be validated for the new process to be approved. Due to the complexity of the tracking setups, the classic approach would require many iterations with extensive empirical measurements to obtain a satisfactory result. The knowledge from prior verification speeds up this process tremendously; much less empirical measurements are needed.

After the setup has been consolidated by verification & validation, the determination of admissible tolerances is straightforward. Their continual surveillance is indispensable for the reliability of the industrial IAR process.

The concepts could be successfully demonstrated by means of two exemplary IAR applications (cf. Chapter 13). These examples also show that a slightly more specific (but still rather generic) elementary specification of uncertainty yields more acccurate results in terms of compliance of the simulation with the empirical measurements in the target environment (cf. 13.2).

Altogether, this thesis presents a very useful toolbox for the IAR-engineer. The generic SRG/DFG modeling concepts allow for the flexible treatment of arbitrary tracking situations by simply combining the correct spatial relationship patterns in a graphical editor. This allows one to realize scenarios with unforseen complexity with relative ease. Furthermore, the pragmatic approach to the assessment and treatment of sensor errors according to the GUM [ISO 08] provides valuable insights into the error characteristics of a given setup at affordable costs, for the Monte Carlo simulation framework directly

integrates with the SRG/DFG modeling concepts. Although the presented toolbox is not yet complete in various respects, this thesis provides a framework that can be extended in the future, in particular toward solutions for the proper time-synchronization of sensors [Schl 11], generic methods for competitive filtering [Pust 08], and statistical methods for outlier detection [Niem 08], as already described in 4.2.

Appendix

A. SRGs and DFGs

All SRGs and DFGs described throughout this thesis, are depicted in the remainder of this appendix. Some of the graphics are rather small when viewed in the printed version, due to their complexity. Furthermore, the graphics may not rasterize well, depending on the used printer driver or PDF viewer. Nevertheless, they have been kept in a vector-format such that zooming works properly when using the electronic version of this document. It can be downloaded from the author's website [Keit 11a].

All depicted SRG files can be downloaded from [Keit 11b] for inspection and editing in *trackman*. They are compatible with *trackman* version 1.10.00 which can be downloaded from [Ubi 11]. A *subversion* checkout of *Ubitrack* as of November 13, 2010 (revision 1453) from the *Ubitrack* subversion repository is also needed such that the pattern templates are available for *trackman* [UbiS 11]. Ensure that the `trackman.conf` file in your *trackman* installation points to the pattern template catalogue shipped with *Ubitrack*, instead of the outdated catalogue shipped with the *trackman* release. To this end, set option `PatternTemplateDirectory` to the full path of the `doc/utql/patterns` subdirectory of your *Ubitrack* installation. See also the *trackman* user manual for the setup and usage of *trackman*. A build of *Ubitrack* is *not* necessary to open and edit the SRGs in *trackman*.

In order to actually instantiate the corresponding data flows, however, a build of *Ubitrack* is needed. Please refer to the *Ubitrack* website for further information [Ubi 11]. Additional settings need to be done in `trackman.conf` to enable the data flow instantiation facilities. Refer to the *trackman* user manual for details.

A.1. Application

The following SRGs and DFGs are discussed in 7.1.

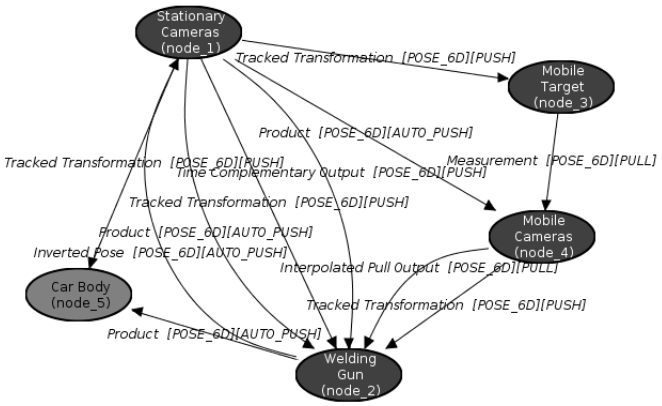


Figure A.1.: SRG for indirect tracking of the intelligent welding gun

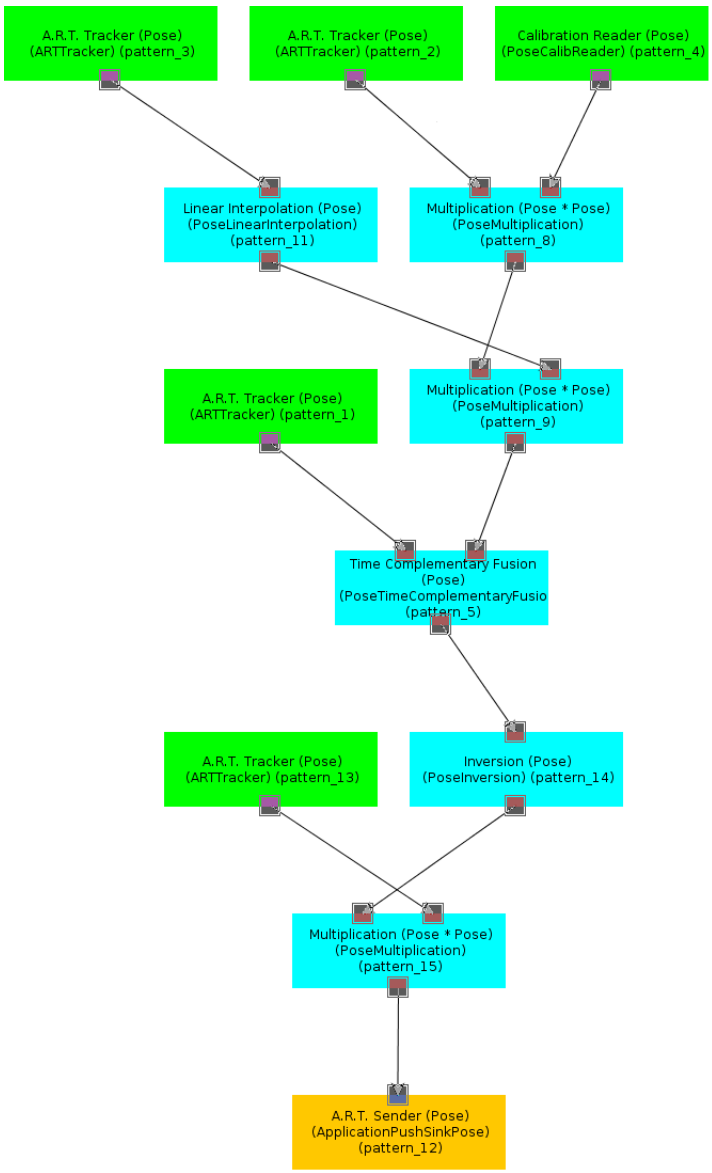


Figure A.2.: DFG for indirect tracking of the intelligent welding gun

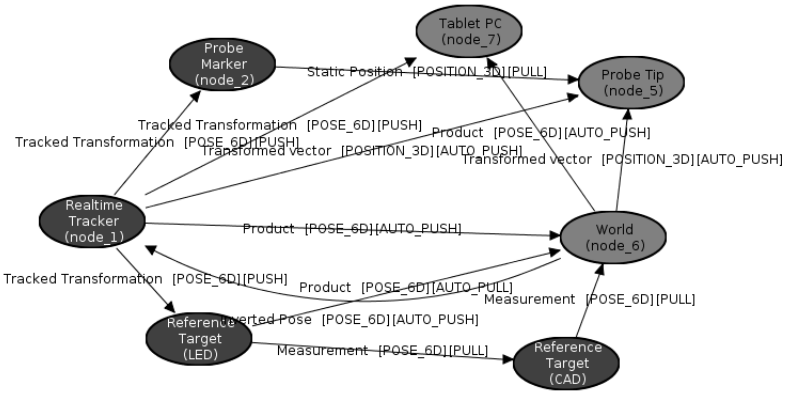


Figure A.3.: SRG for probe tracking in the airplane

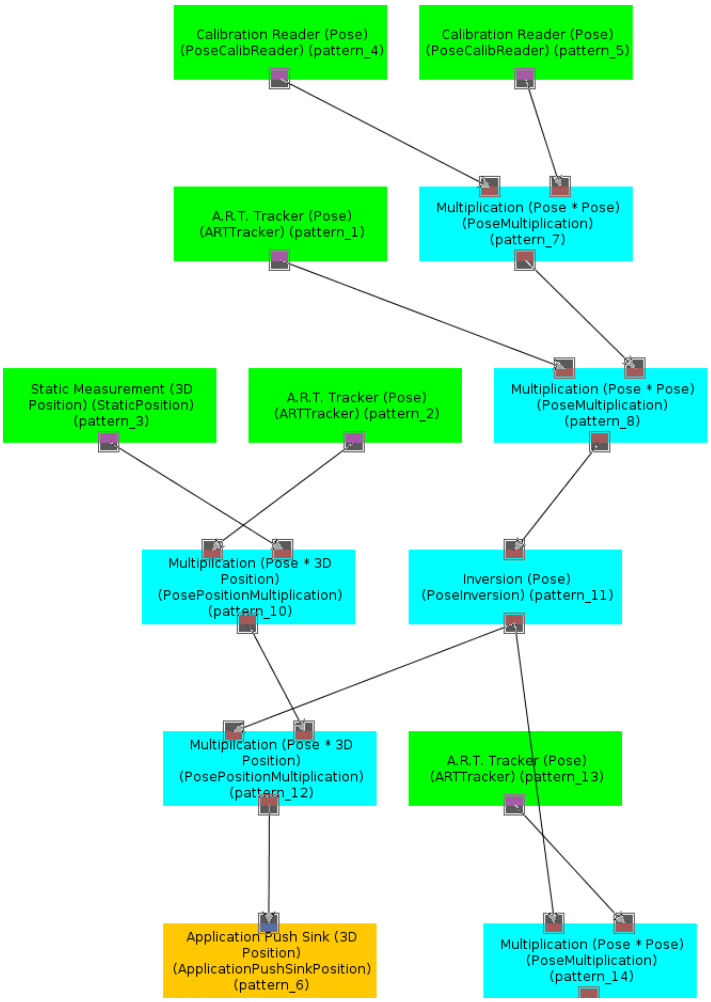


Figure A.4.: DFG for probe tracking in the airplane

A.2. Registration

The following SRGs and DFGs are discussed in 7.2.

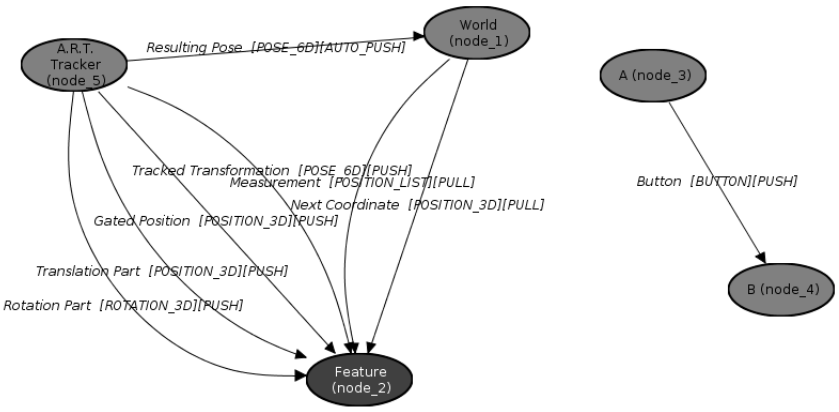


Figure A.5.: SRG for absolute orientation based on user-triggered measurements

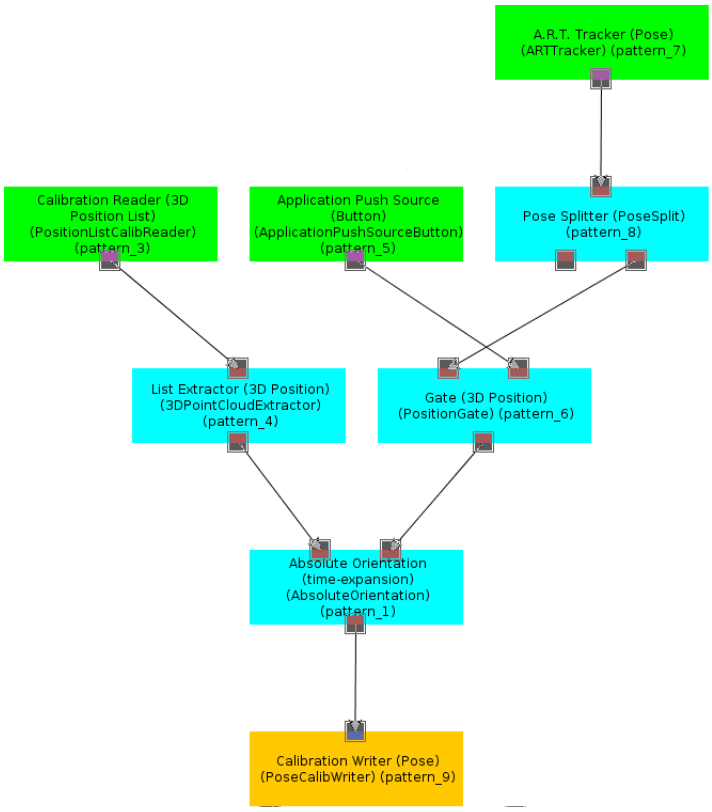


Figure A.6.: DFG for absolute orientation based on user-triggered measurements

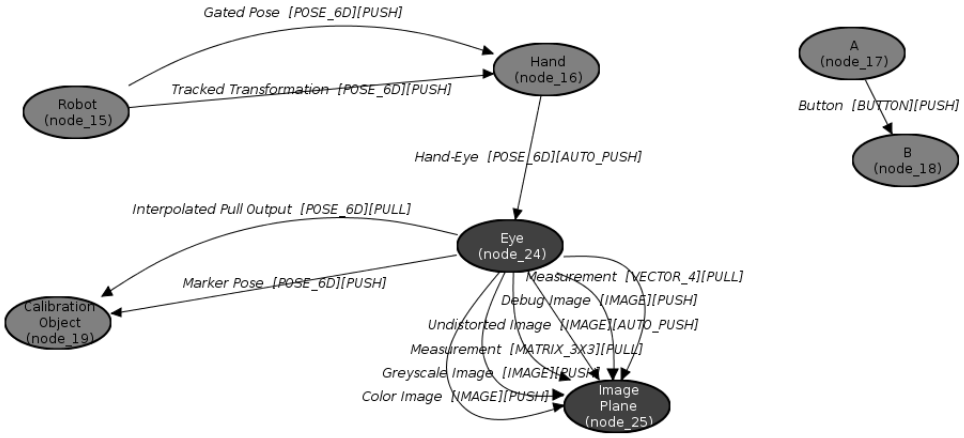


Figure A.7.: SRG for hand-eye calibration based on user-triggered measurements

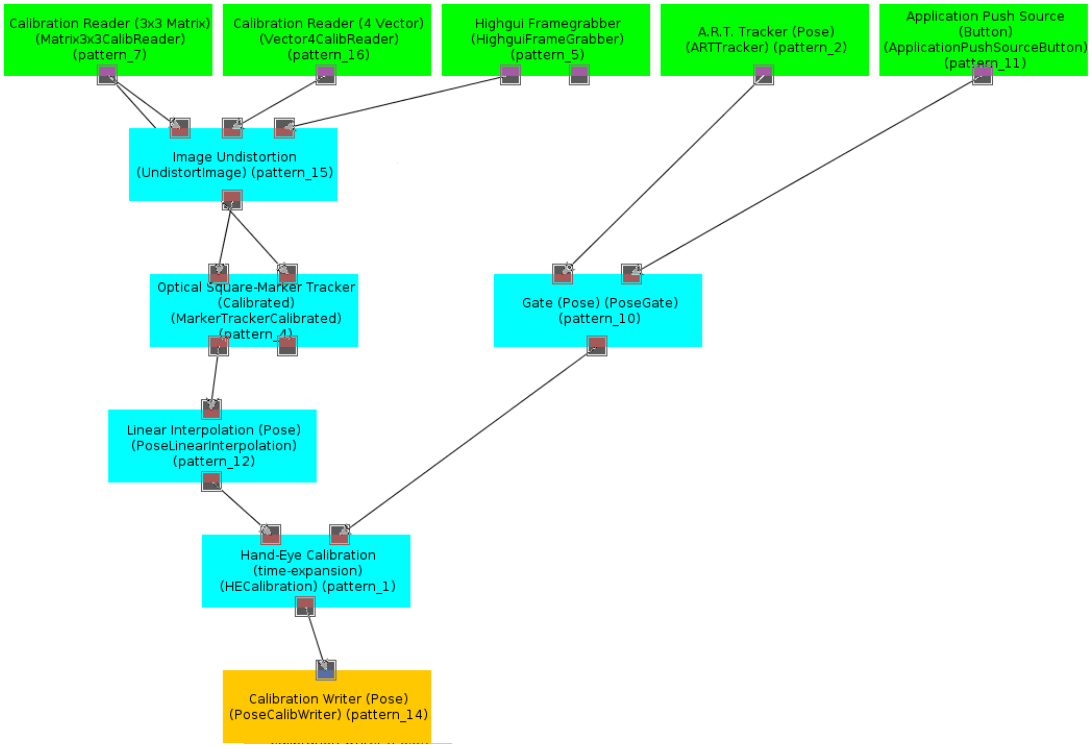


Figure A.8.: DFG for hand-eye calibration based on user-triggered measurements

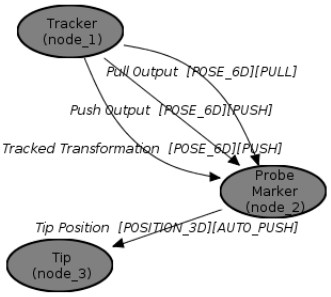


Figure A.9.: SRG for tip calibration based on discrete measurements

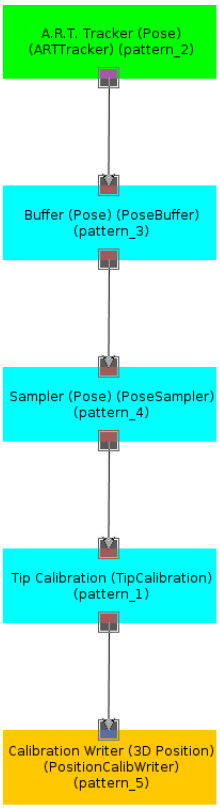


Figure A.10.: DFG for tip calibration based on discrete measurements

A.3. Simulation

The following SRGs and DFGs are discussed in 11.2.

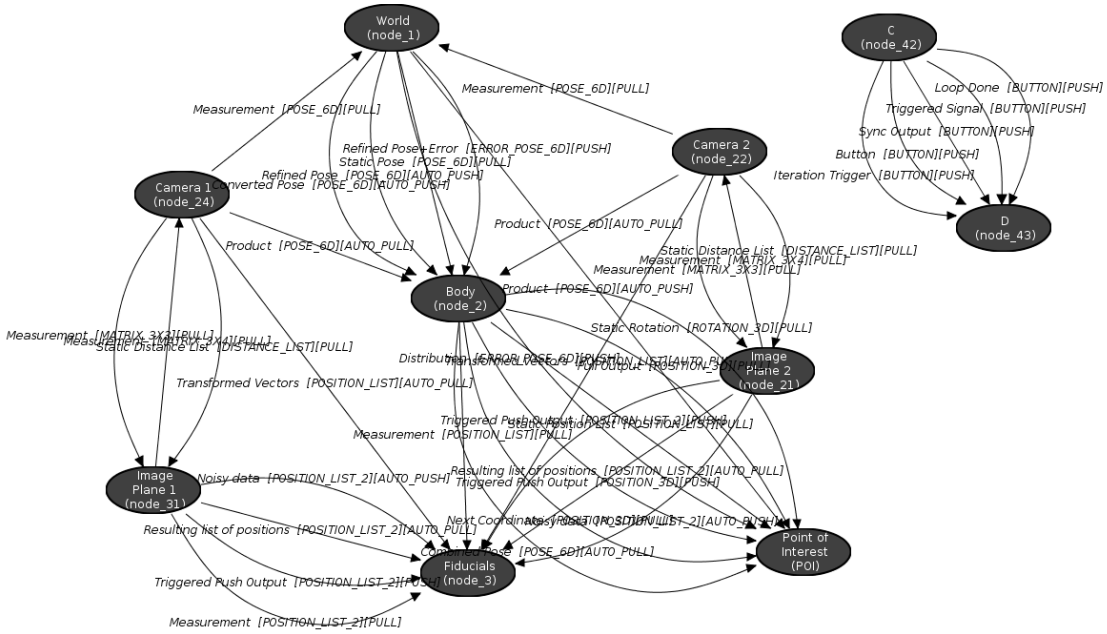


Figure A.11.: SRG for simulation of the 2D-6D pose estimation problem

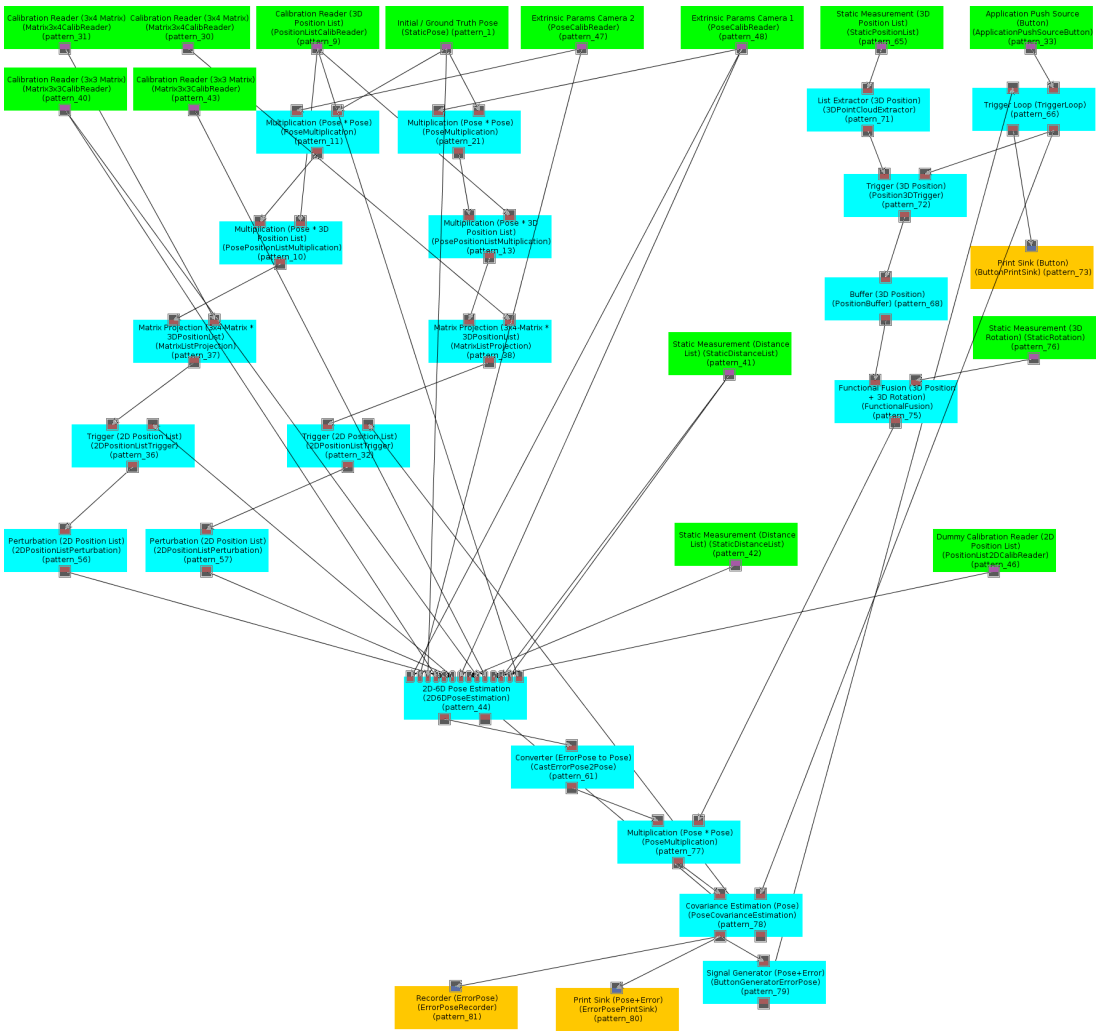


Figure A.12.: DFG for simulation of the 2D-6D pose estimation problem

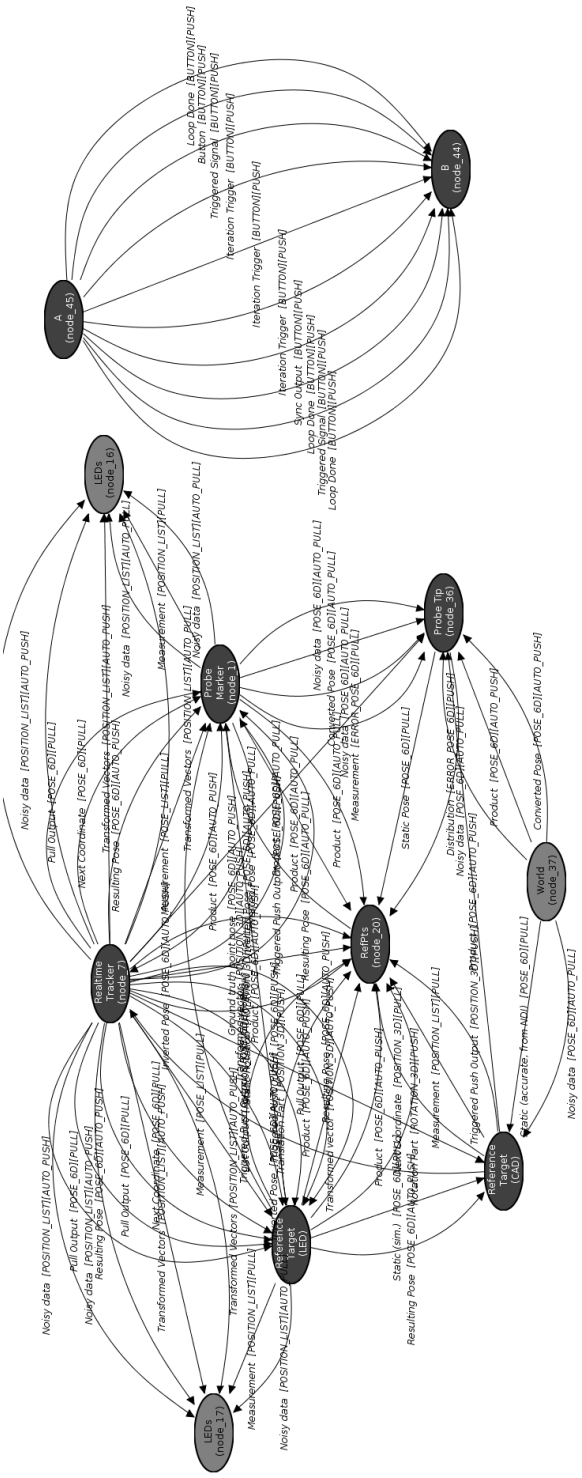


Figure A.13.: SRG for simulation of probe tracking in the airplane

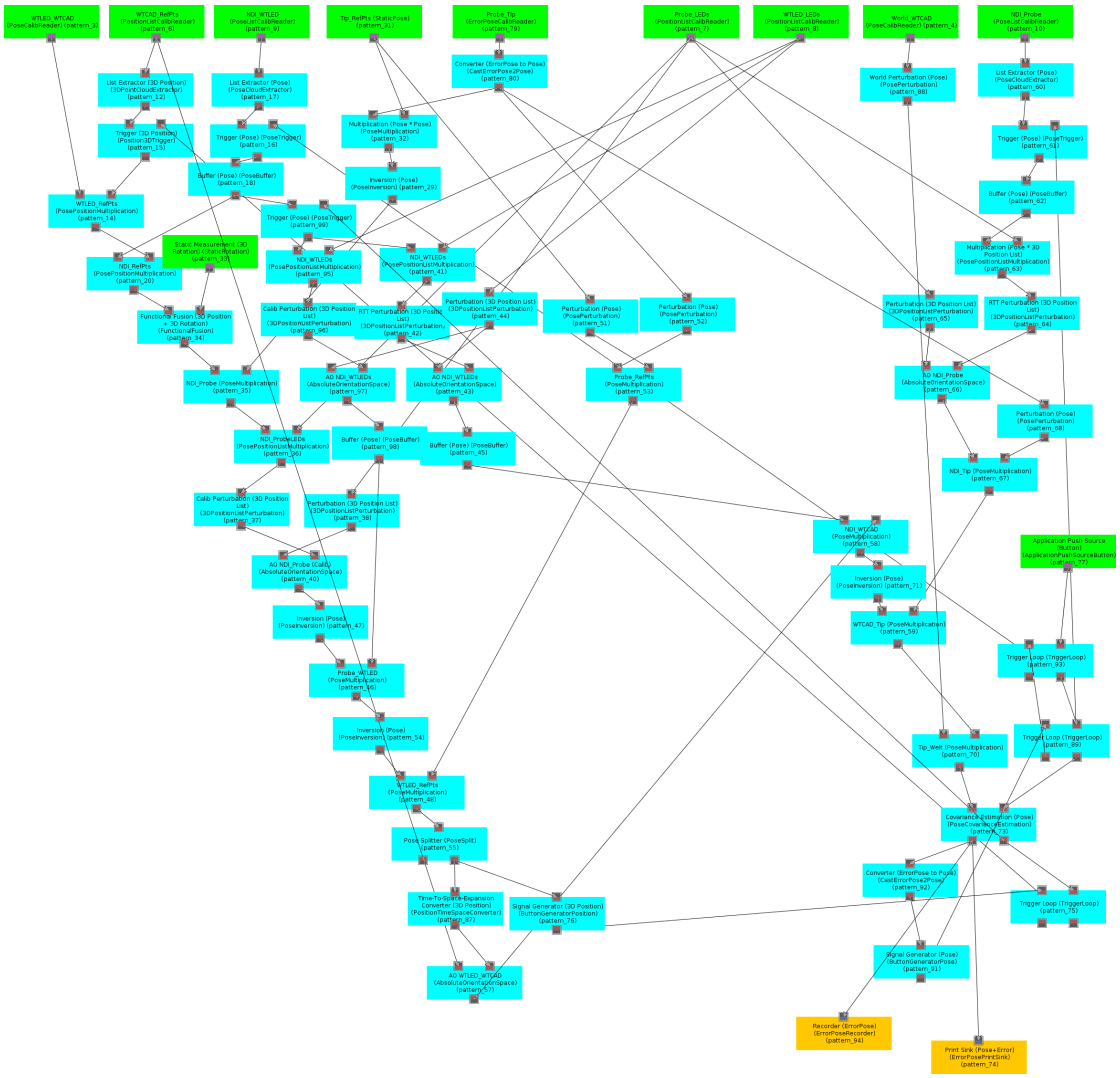


Figure A.14.: DFG for simulation of probe tracking in the airplane

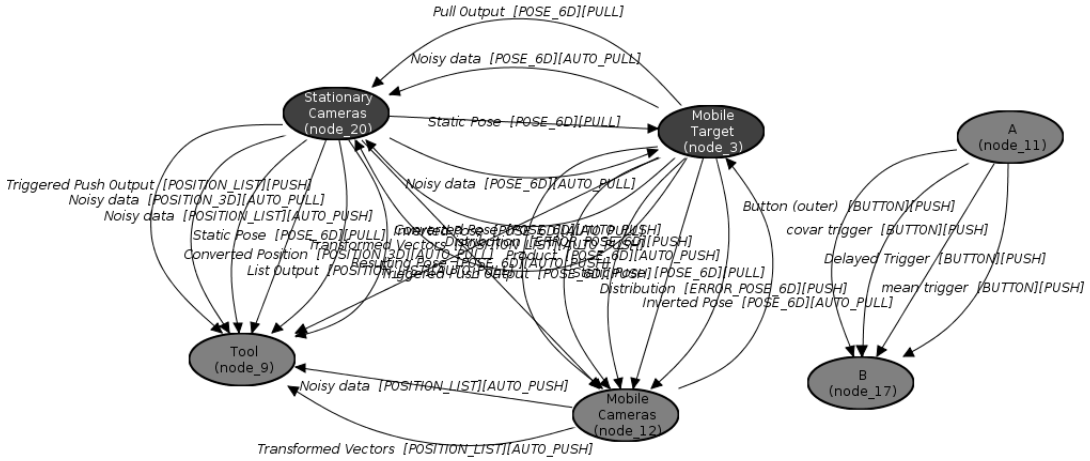


Figure A.15.: SRG for simulation of hand-eye calibration for indirect tracking of the intelligent welding gun

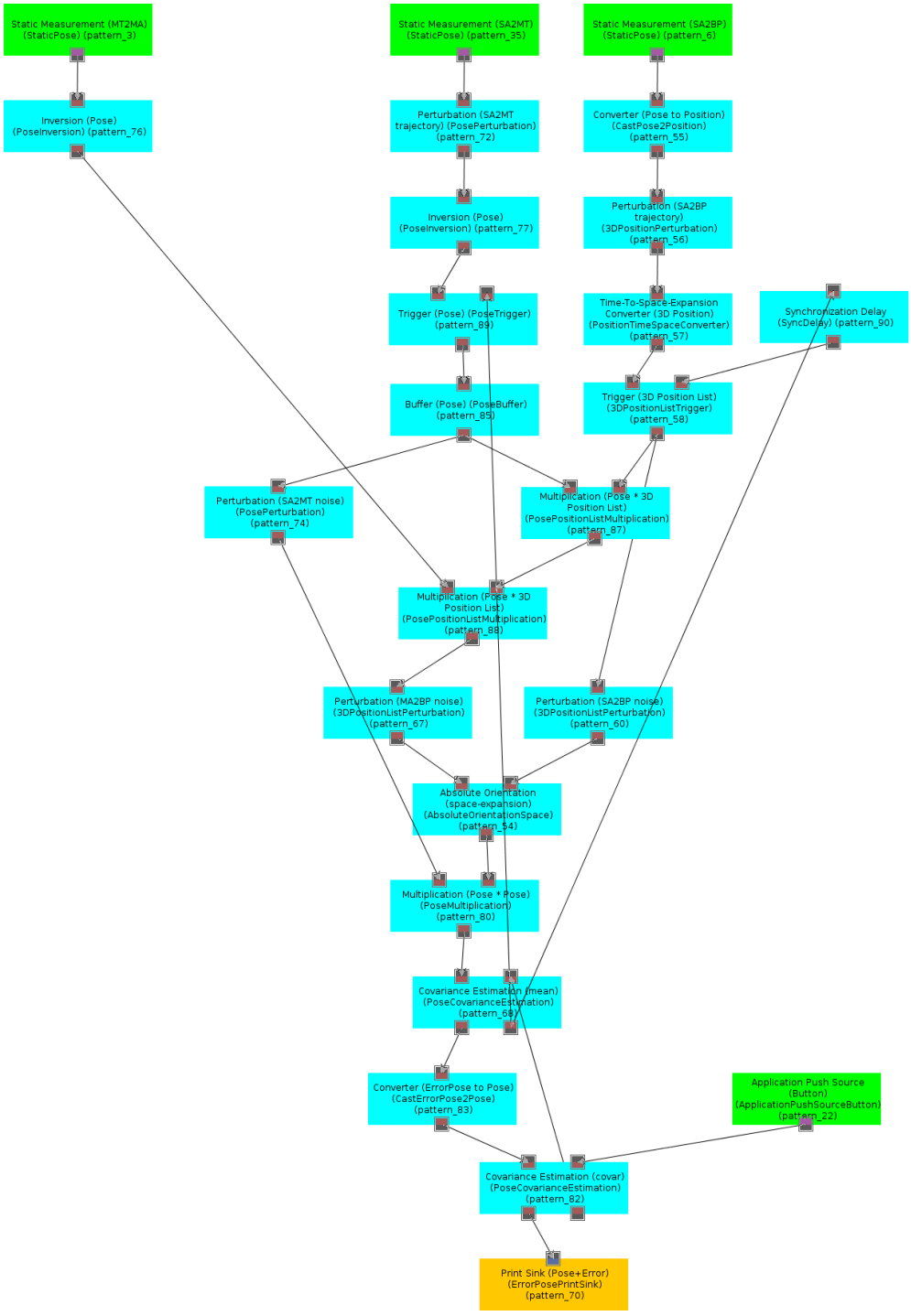


Figure A.16.: DFG for simulation of hand-eye calibration for indirect tracking of the intelligent welding gun

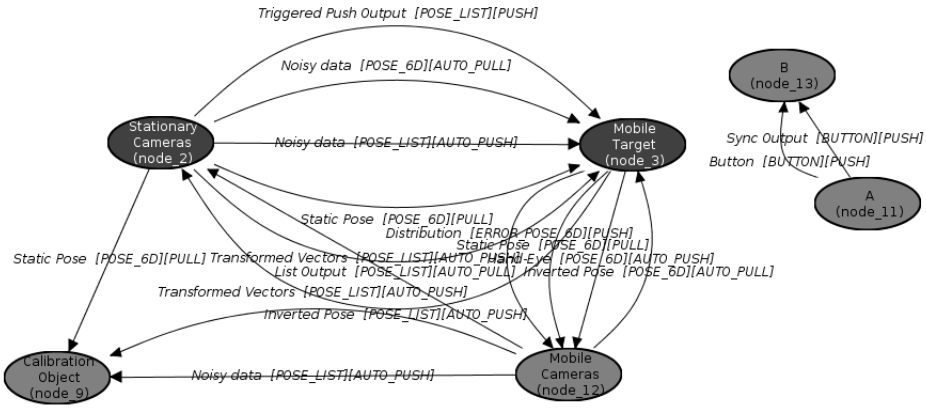


Figure A.17.: SRG for simulation of absolute orientation for indirect tracking of the intelligent welding gun

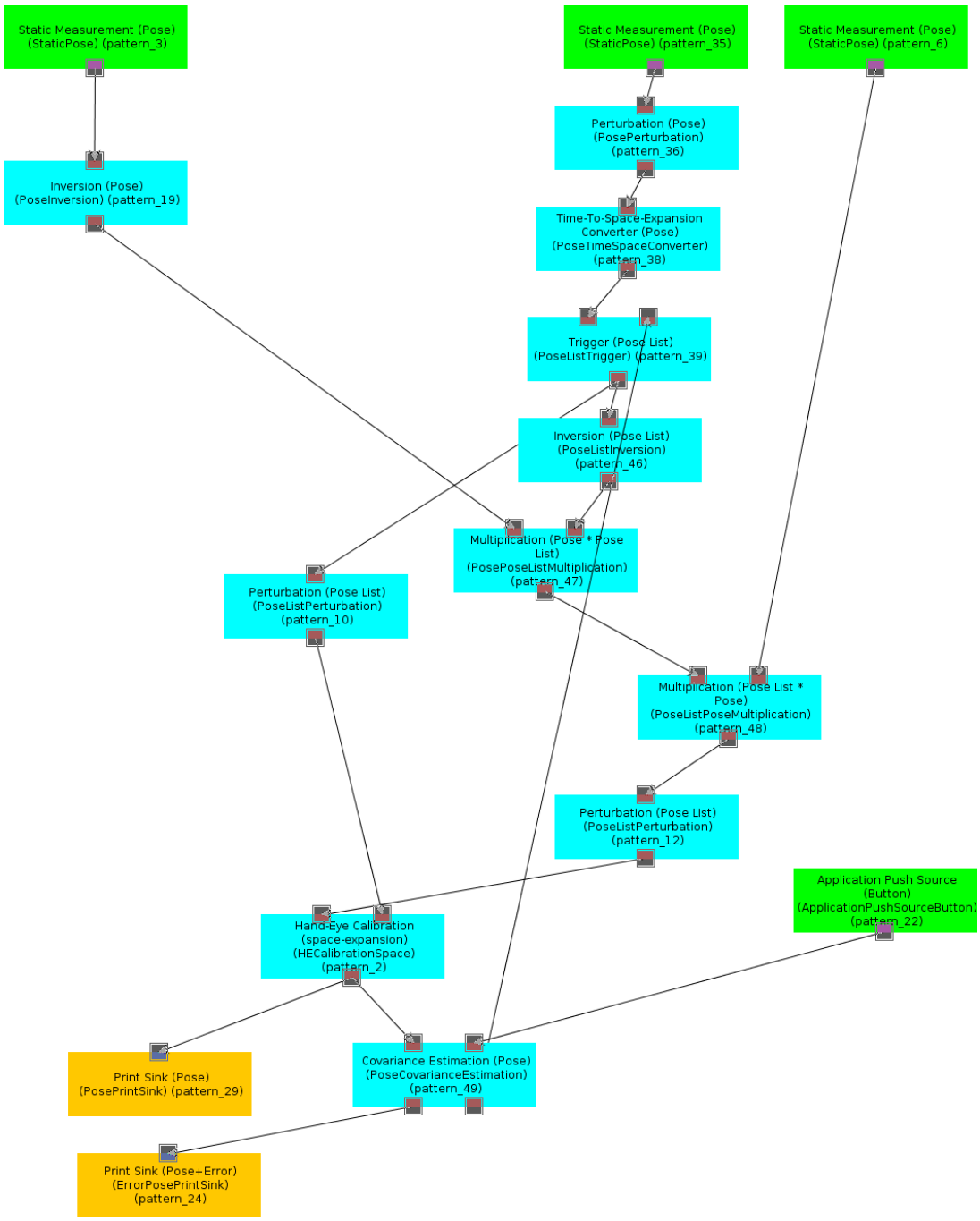


Figure A.18.: DFG for simulation of absolute orientation for indirect tracking of the intelligent welding gun

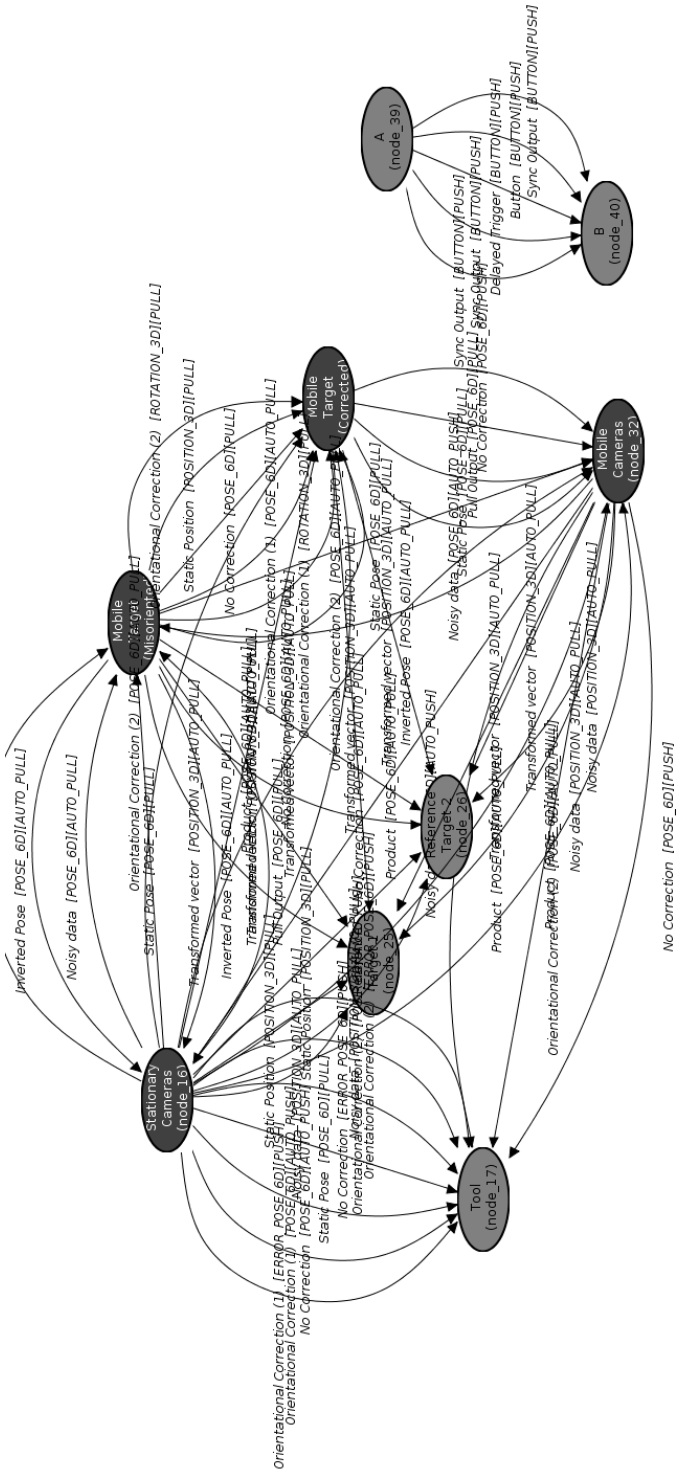


Figure A.19.: SRG for simulation of indirect tracking of the intelligent welding gun

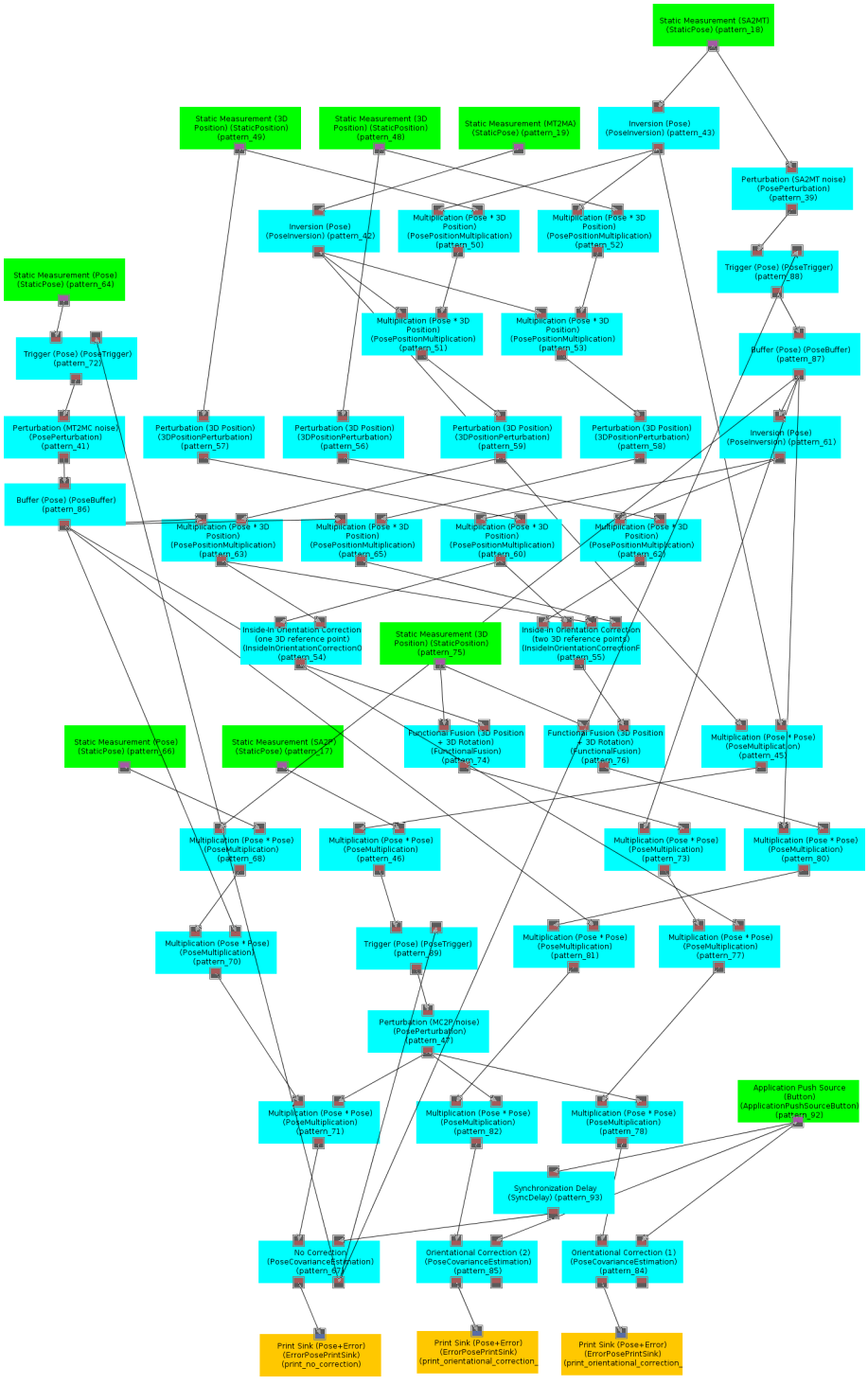


Figure A.20.: DFG for simulation of indirect tracking of the intelligent welding gun

Bibliography

[Abaw 04] D. Abawi, J. Bienwald, and R. Dorner. "Accuracy in optical tracking with fiducial markers: an accuracy function for ARToolKit". In: *Proc. 3rd IEEE/ACM International Symposium on Mixed and Augmented Reality*, pp. 260–261, IEEE Computer Society Washington, DC, USA, 2004.

[Adob 11] "*Adobe* company website". http://www.adobe.com. Accessed January 12, 2011.

[Albu 02] J. Albus. "4D/RCS: a reference model architecture for intelligent unmanned ground vehicles". In: *Proc. SPIE*, p. 303, SPIE, 2002.

[Alle 05] B. D. Allen and G. Welch. "A general method for comparing the expected performance of tracking and motion capture systems". In: *Proc. ACM symposium on Virtual reality software and technology (VRST)*, pp. 201–210, ACM Press, New York, NY, USA, 2005.

[Alle 07] B. Allen. *Hardware design optimization for human motion tracking systems*. PhD thesis, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA, 2007.

[ART 11] "*Advanced Realtime Tracking* company website". http://www.ar-tracking.com. Accessed January 12, 2011.

[ARVI 11] "*ARVIKA* project website". http://www.arvika.de/www/index.htm. Accessed January 12, 2011.

[ASME 06] "Verification and Validation in Computational Solid Mechanics". American Society of Mechanical Engineering (ASME) Guide, October 2006.

[Atki 96] K. Atkinson, Ed. *Close range photogrammetry and machine vision*. Whittles Publishing Caithness, UK, 1996.

[AVIL 11] "*AVILUS* project website". http://www.avilus.de/. Accessed January 12, 2011.

[Azum 97] R. Azuma *et al.* "A survey of augmented reality". *Presence: Teleoperators and Virtual Environments(1054-7460)*, Vol. 6, No. 4, pp. 355–385, 1997.

[Bail 03] Y. Baillot, S. Julier, D. Brown, and M. Livingston. "A tracker alignment framework for augmented reality". In: *Proc. 2nd IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'03)*, p. 142, IEEE Computer Society, 2003.

[Baue 01] 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: *Proc. International Symposium on Augmented Reality (ISAR)*, Oct. 2001.

[Baue 06] M. Bauer, M. Schlegel, D. Pustka, N. Navab, and G. Klinker. "Predicting and estimating the accuracy of n-occular optical tracking systems". In: *IEEE/ACM International Symposium on Mixed and Augmented Reality, 2006. ISMAR 2006*, pp. 43–51, 2006.

[Baue 07] M. Bauer. *Tracking Errors in Augmented Reality*. PhD thesis, Technische Universität München, 2007.

[Beck 11] B. Becker. *Efficient Application of Digital Models using Mixed Reality Tools*. PhD thesis, Technische Universität München, 2011.

[Bich 06] C. Bichlmeier, T. Sielhorst, and N. Navab. "The Tangible Virtual Mirror: New Visualization Paradigm for Navigated Surgery". In: *AMIARCS—The Tangible Virtual Mirror: New Visualization Paradigm for Navigated Surgery*, MICCAI Society, Copenhagen, Denmark, October 2006.

[Bimb 05] O. Bimber and R. Raskar. *Spatial augmented reality: Merging real and virtual worlds*. AK Peters Ltd, 2005.

[Bles 06] G. Bleser, H. Wuest, and D. Strieker. "Online camera pose estimation in partially known and dynamic scenes". In: *IEEE/ACM International Symposium on Mixed and Augmented Reality, 2006. ISMAR 2006*, pp. 56–65, 2006.

[Blum 09] L. Blum, W. Broll, and S. Müller. "Augmented reality under water". In: *SIGGRAPH'09: Posters*, p. 97, ACM, 2009.

[Bouc 04] J. Bouchet, L. Nigay, and T. Ganille. "ICARE software components for rapidly developing multimodal interfaces". In: *Proc. 6th international conference on Multimodal interfaces*, pp. 251–258, ACM, 2004.

[Broo 97] R. Brooks and S. Iyengar. "Real-time distributed sensor fusion for time-critical sensor readings". *Optical Engineering*, Vol. 36, p. 767, 1997.

[Buck 10] K. Buckl. "Stud-welding at Volkswagen using a mechanical measurement arm". personal communication, 2010.

[CATI 11] "*Dassault Systemes CATIA* website". http://www.3ds.com/products/catia. Accessed January 12, 2011.

[Caud 92] T. Caudell and D. Mizell. "Augmented reality: an application of heads-up display technology tomanual manufacturing processes". In: *Proc. Twenty-Fifth Hawaii International Conference on System Sciences*, 1992.

[Chmi 08] C. Chmill. *Untersuchungen zur Eignung eines Messgelenkarms mit optionalem 3D-Scanner für die Flugzeugstrukturmontage bei Airbus*. Master's thesis, Fachhochschule Oldenburg Ostfriesland Willhelmshaven, 2008.

[CL 11] "Website of *Computer Laboratory* at *University of Cambridge*". http://www.cl.cam.ac.uk/. Accessed January 12, 2011.

[Clip 07] B. Clipp, G. Welch, J.-M. Frahm, and M. Pollefeys. "Structure from Motion via a Two-Stage Pipeline of Extended Kalman Filters". *Proceedings of the British Machine Vision Conference (BMVC 2007)*, September 10–13 2007.

[Coel 04] E. Coelho, B. MacIntyre, and S. Julier. "OSGAR: A Scene Graph with Uncertain Transformations". In: *International Symposium on Mixed and Augmented Reality (ISMAR 2004)*, pp. 6–15, 2004.

[Coqu 04] S. Coquillart and M. Göbel. "Authoring of Mixed Reality Applications including Multi-Marker Calibration for Mobile Devices". In: *Proc. Eurographics Symposium on Virtual Environments*, pp. 1–9, Citeseer, 2004.

[Cox 06] M. Cox and P. Harris. "Software Support for Metrology Best Practice Guide No. 6: Uncertainty Evaluation". Tech. Rep. DEM-ES 011, ISSN: 1744-0475, National Physical Laboratory, Teddington, UK, September 2006.

[Cruz 93] C. Cruz-Neira, D. Sandin, and T. DeFanti. "Surround-screen projection-based virtual reality: the design and implementation of the CAVE". In: *Proc. 20th annual conference on Computer graphics and interactive techniques*, p. 142, ACM, 1993.

[Dani 99] K. Daniilidis. "Hand-Eye Calibration Using Dual Quaternions". *The International Journal of Robotics Research*, Vol. 18, No. 3, p. 286, 1999.

[Darm 08] M. Darms, P. Rybski, C. Urmson, C. Inc, and M. Auburn Hills. "Classification and tracking of dynamic objects with multiple sensors for autonomous driving in urban environments". In: *Intelligent Vehicles Symposium, 2008 IEEE*, pp. 1197–1202, 2008.

[Davi 03] L. Davis, E. Clarkson, and J. P. Rolland. "Predicting Accuracy in Pose Estimation for Marker-based Tracking". In: *Proc. Second IEEE and ACM International Symposium on Mixed and AugmentedReality (ISMAR)*, p. 28, 2003.

[Davi 04] L. Davis, F. G. Hamza-Lup, and J. P. Rolland. "A Method for Designing Marker-Based Tracking Probes". In: *Proc. Third IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 120–129, 2004.

[Dey 01] A. Dey, G. Abowd, and D. Salber. "A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications". *Human-Computer Interaction*, Vol. 16, No. 2, 3 & 4, pp. 97–166, 2001.

[Dick 07] E. D. Dickmanns. *Dynamic Vision for Perception and Control of Motion*. Springer Verlag, Berlin, 2007.

[Dimo 05] I. Dimov. *Monte Carlo methods for applied scientists*. World Scientific, 2005.

[Dorn 98] F. Dornaika and R. Horaud. "Simultaneous robot-world and hand-eye calibration". *Robotics and Automation, IEEE Transactions on*, Vol. 14, No. 4, pp. 617–622, 1998.

[Douc 01] A. Doucet, N. de Freitas, and N. Gordon. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag, 2001.

[Doyl 64] F. Doyle. "The historical development of analytical photogrammetry". *Photogrammetric Engineering*, Vol. 30, No. 2, pp. 259–265, 1964.

[Drag 01] P. Dragicevic and J. Fekete. "Input device selection and interaction configuration with ICON". In: *People and Computer XV-Interaction without Frontier (Joint proceedings of HCI 2001 and IHM 2001)*, pp. 543–448, 2001.

[Durr 88] H. Durrant-Whyte. "Sensor models and multisensor integration". *The International Journal of Robotics Research*, Vol. 7, No. 6, p. 97, 1988.

[DySe 11] "*DySenNetz* project website". http://campar.in.tum.de/Chair/ProjectDySenNetz. Accessed January 12, 2011.

[Echt 03] F. Echtler, F. Sturm, K. Kindermann, G. Klinker, J. Stilla, J. Trilk, and H. Najafi. "The Intelligent Welding Gun: Augmented Reality for Experimental Vehicle Construction". In: S. Ong and A. Nee, Eds., *Virtual and Augmented Reality Applications in Manufacturing, Chapter 17*, Springer Verlag, 2003.

[Echt 08] F. Echtler, M. Huber, D. Pustka, P. Keitler, and G. Klinker. "Splitting the Scene Graph—Using Spatial Relationship Graphs Instead of Scene Graphs in Augmented Reality". In: *Proc. 3rd International Conference on Computer Graphics Theory and Applications (GRAPP)*, Jan. 2008.

[Egge 97] D. Eggert, A. Lorusso, and R. Fisher. "Estimating 3-D rigid body transformations: a comparison of four major algorithms". *Machine Vision and Applications*, Vol. 9, No. 5, pp. 272–290, 1997.

[Fack 91] P. Fackler. "Modeling interdependence: an approach to simulation and elicitation". *American Journal of Agricultural Economics*, Vol. 73, No. 4, p. 1091, 1991.

[FAR 11] "*Technische Universität München, Fachgebiet Augmented Reality* website". http://campar.in.tum.de/Chair/ResearchAr. Accessed January 12, 2011.

[Faro 11a] "*FARO* company website". http://www.faro.com. Accessed January 12, 2011.

[Faro 11b] "*FARO ION* lasertracker website". http://www.faro.com/lasertracker/. Accessed January 12, 2011.

[Fisc 04] J. Fischer, M. Neff, D. Freudenstein, and D. Bartz. "Medical Augmented Reality based on Commercial Image Guided Surgery". In: *Proc. Eurographics Symposium on Virtual Environments (EGVE)*, pp. 83–86, 2004.

[Fisc 81] M. A. Fischler and R. C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". *Commun. ACM*, Vol. 24, No. 6, pp. 381–395, 1981.

[Fitz 98] J. M. Fitzpatrick, J. B. West, and C. R. Maurer, Jr. "Predicting Error in Rigid-Body Point-Based Registration". *TMI*, Vol. 14, No. 5, pp. 694–702, 1998.

[Frie 04] W. Friedrich. *ARVIKA: Augmented Reality für Entwicklung, Produktion und Service*. VCH, 2004.

[Gelb 74] A. Gelb. *Applied optimal estimation*. The MIT press, 1974.

[Geor 07] P. Georgel, P. Schroeder, S. Benhimane, S. Hinterstoisser, M. Appel, and N. Nassir. "An Industrial Augmented Reality Solution For Discrepancy Check". In: *Proc. 6th International Symposium on Mixed and Augmented Reality (ISMAR)*, Nov. 2007.

[Grat 11] "*Java Graticule 3D* website". http://derletztekick.com/software/koordinatentransformation. Accessed January 12, 2011.

[Hart 00] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, UK, second Ed., June 2000.

[Hast 04] H. Hastedt, T. Luhmann, and W. Tecklenburg. "Bestimmung von Einflussgrößen in der Nahbereichsphotogrammetrie mittels Monte-Carlo-Simulation". *Publikationen der DGPF*, Vol. 13, pp. 359–366, 2004.

[Hoff 00] W. Hoff and T. Vincent. "Analysis of Head Pose Accuracy in Augmented Reality". In: *IEEE Transactions on Visualization and Computer Graphics*, pp. 319–334, IEEE Computer Society, 2000.

[Hohl 99] F. Hohl, U. Kubach, A. Leonhardi, K. Rothermel, and M. Schwehm. "Next Century Challenges: Nexus—An Open Global Infrastructure for Spatial-Aware Applications". In: *Proc. Fifth Annual International Conference on Mobile Computing and Networking (MobiCom)*, pp. 249–255, Universität Stuttgart : Sonderforschungsbereich SFB 627 (Nexus: Umgebungsmodelle

für mobile kontextbezogene Systeme), Seattle, WA, USA: not available, August 1999.

[Holl 95] R. Holloway. *Registration Errors in Augmented Reality Systems*. PhD thesis, University of North Carolina, 1995.

[Holl 97] R. Holloway. "Registration Error Analysis for Augmented Reality". *Presence: Teleoperators and Virtual Environments*, Vol. 6, No. 4, pp. 413–432, 1997.

[Horn 87] B. Horn, H. Hilden, and S. Negahdaripour. "Closed-form solution of absolute orientation using unit quaternions". *Journal of the Optical Society of America A*, Vol. 4, No. 4, pp. 629–642, 1987.

[Hube 07] M. Huber, D. Pustka, P. Keitler, F. Echtler, and G. Klinker. "A System Architecture for Ubiquitous Tracking Environments". In: *Proc. 6th International Symposium on Mixed and Augmented Reality (ISMAR)*, Nov. 2007.

[Hube 09] M. Huber, M. Schlegel, and G. Klinker. "Temporal Calibration in Multisensor Tracking Setups". In: *Proc. Shortpapers and Posters of the 8th IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR 2009)*, Orlando, USA, October 2009.

[ICG 11] "Website of *Institute for Computer Graphics and Vision* at *Technische Universität Graz*". http://www.icg.tu-graz.ac.at/. Accessed January 12, 2011.

[ISO 08] "Uncertainty of measurement—Part 3: Guide to the expression of uncertainty in measurement (GUM)". International Organization for Standardization (ISO/IEC) norm 98-3:2008, 2008.

[Jage 05] R. Jäger, T. Müller, H. Saler, and R. Schwäble. *Klassische und robuste Ausgleichungsverfahren*. Herbert Wichmann Verlag, Heidelberg, Germany, 2005.

[Juli 04] S. Julier, J. Uhlmann, I. Ind, and M. Jefferson City. "Unscented filtering and nonlinear estimation". *Proc. IEEE*, Vol. 92, No. 3, pp. 401–422, 2004.

[Kalm 60] R. Kalman. "A new approach to linear filtering and prediction problems". *Journal of Basic Engineering*, Vol. 82, No. 1, pp. 35–45, 1960.

[Kana 93] K. Kanatani. "Unbiased estimation and statistical analysis of 3-D rigid motion from two views". *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, Vol. 15, No. 1, pp. 37–50, 1993.

[Kana 96] K. Kanatani. *Statistical optimization for geometric computation: theory and practice*. Elsevier Science Inc. New York, NY, USA, 1996.

[Keit 08] P. Keitler, M. Schlegel, and G. Klinker. "Indirect Tracking to Reduce Occlusion Problems". In: *Advances in Visual Computing, Fourth International Symposium, ISVC 2008 Las Vegas, USA, December 1-3*, pp. 224–235, Springer, Berlin, Deutschland, 2008.

[Keit 09] P. Keitler, F. Pankratz, B. Schwerdtfeger, D. Pustka, W. Rödiger, G. Klinker, C. Rauch, A. Chathoth, J. Collomosse, and Y.-Z. Song. "Mobile Augmented Reality based 3D Snapshots". In: *Proc. Sechster Workshop Virtuelle und Erweiterte Realität der GI-Fachgruppe VR/AR*, Braunschweig, Germany, November 2009.

[Keit 10a] P. Keitler, D. Pustka, M. Huber, F. Echtler, and G. Klinker. *The Engineering of Mixed Reality Systems*, Chap. Management of Tracking for Mixed and Augmented Reality Systems. Springer Verlag, 2010.

[Keit 10b] P. Keitler, B. Becker, and G. Klinker. "Management of Tracking for Industrial AR Setups". In: *Proc. 9th International Symposium on Mixed and Augmented Reality (ISMAR)*, October 2010.

[Keit 11a] P. Keitler. "Peter Keitler's website at Technische Universität München". http://campar.in.tum.de/Main/PeterKeitler, 2011. Accessed January 12, 2011.

[Keit 11b] P. Keitler. "SRG files discussed in this thesis". http://campar.in.tum.de/twiki/pub/Main/PeterKeitler/SRGs.zip, 2011. Accessed January 16, 2011.

[Kenn 52] J. Kenney and E. Keeping. *Mathematics of Statistics, Pt. 2*. Princeton, NJ: Van Nostrand, 1952.

[Kine 11] "*New River Kinematics* company website". http://www.kinematics.com. Accessed January 12, 2011.

[Kirk 83] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi. "Optimization by simulated annealing". *Science*, Vol. 220, No. 4598, p. 671, 1983.

[Klei 07] G. Klein and D. Murray. "Parallel tracking and mapping for small AR workspaces". In: *6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007. ISMAR 2007*, pp. 1–10, 2007.

[Koch 97] K. Koch. *Parameterschätzung und Hypothesentests in linearen Modellen*. Dümmlers Verlag, Bonn, Germany, third Ed., 1997.

[Kuip 02] J. Kuipers. *Quaternions and rotation sequences: a primer with applications to orbits, aerospace, and virtual reality*. Princeton Univ Pr, 2002.

[lase 11] "*Ascension laserBird 2* website". http://www.ascension-tech.com/realtime/laserBIRD2.php. Accessed January 12, 2011.

[Lieb 09] S. Lieberknecht, S. Benhimane, P. Meier, and N. Navab. “A Dataset and Evaluation Methodology for Template-based Tracking Algorithms”. In: *ISMAR*, 2009.

[Luhm 00a] T. Luhmann. *Nahbereichsphotogrammetrie*. Herbert Wichmann Verlag, Heidelberg, Germany, 2000.

[Luhm 00b] T. Luhmann. “Photogrammetrische Verfahren in der industriellen Messtechnik”. In: *Publikationen der DGPF*, DGPF, 2000.

[MacI 04] B. MacIntyre, M. Gandy, S. Dow, and J. Bolter. “DART: a toolkit for rapid design exploration of augmented reality experiences”. In: *Proc. 17th annual ACM symposium on User Interface Software and Technology*, pp. 197–206, ACM, 2004.

[MacW 03] A. MacWilliams, C. Sandor, M. Wagner, M. Bauer, G. Klinker, and B. Brgge. “Herding sheep: Live system development for distributed augmented reality”. In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*, Tokyo, Japan, October 2003.

[Malb 02] P. Malbezin, W. Piekarski, and B. Thomas. “Measuring ARTootKit accuracy in long distance tracking experiments”. In: *Proc. 1st IEEE International Augmented Reality Toolkit Workshop*, p. 2, 2002.

[Mann 02] S. Mann. “Mediated Reality with implementations for everyday life”. In: *Presence Connect, the on line companion to the MIT Press journal PRESENCE: Teleoperators and Virtual Environments*, MIT Press, 2002.

[Mark 04] F. Markley. “Multiplicative vs. additive filtering for spacecraft attitude determination”. In: *Proc. 6th Conference on Dynamics and Control of Systems and Structures in Space (DCSSS)*, pp. 467–474, 2004.

[meta 11] “*metaio* company website”. http://www.metaio.de. Accessed January 12, 2011.

[Mete 49] N. Meteopolis and S. Ulam. “The monte carlo method”. *Journal of the American Statistical Association*, Vol. 44, No. 247, pp. 335–341, 1949.

[Metr 53] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller. “Equations of State calculations by fast computing machines”. *Journal of Chemical Physics*, Vol. 21, pp. 1087–1092, 1953.

[Meye 92] K. Meyer, H. Applewhite, and F. Biocca. “A survey of position trackers”. In: *Presence: Teleoperators and Virtual Environments*, pp. 173–200, 1992.

[Micr 11] “*Microvision* company website”. http://www.microvision.com. Accessed January 12, 2011.

[Milg 94] P. Milgram and F. Kishino. “A taxonomy of mixed reality visual displays”. *IEICE Transactions on Information and Systems E series D*, Vol. 77, pp. 1321–1321, 1994.

[Miss 08] S. Misslinger. *Design, Implementation and Evaluation of a Workflow System for Augmented Reality Applications*. Master’s thesis, Technische Universität München, 2008.

[Mora 09] R. Morales-García, P. Keitler, P. Maier, and G. Klinker. “An Underwater Augmented Reality System for Commercial Diving Operations”. In: *OCEANS 2009 MTS/IEEE, Biloxi, Mississippi, USA, November 2009*, 2009.

[Morr 94] J. Morrison. *Flow-based programming*. Van Nostrand Reinhold, 1994.

[Muld 94] A. Mulder. “Human Movement Tracking Technology”. Tech. Rep. 94-1, School of Kinesiology, Simon Fraser University, Burnaby, B.C., Canada, 1994.

[Nava 04] N. Navab. “Developing killer apps for industrial augmented reality”. In: *Computer Graphics and Applications, IEEE*, pp. 16–20, 2004.

[Nava 07] N. Navab, J. Traub, T. Sielhorst, M. Feuerstein, and C. Bichlmeier. “Action- and Workflow-Driven Augmented Reality for Computer-Aided Medical Procedures”. *IEEE Computer Graphics and Applications*, Vol. 27, No. 5, pp. 10–14, September/October 2007.

[NDI 11] “*NDI* company website”. http://www.ndigital.com. Accessed January 12, 2011.

[Newm 04] 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.

[Newm 07] J. Newman, A. Bornik, D. Pustka, F. Echtler, M. Huber, D. Schmalstieg, and G. Klinker. “Tracking for Distributed Mixed Reality Environments”. In: *Proc. IEEE VR 2007 Workshop on ’Trends and Issues in Tracking for Virtual Environments’*, IEEE, Shaker Verlag, Aachen, Germany, 2007.

[Niem 08] W. Niemeier. *Ausgleichsrechnung—Statistische Auswertungsmethoden*. Walter de Gruyter Verlag, Berlin, Germany, second Ed., 2008.

[Niko 11] “*Nikon iSpace* metrology and tracking system”. http://www.nikonmetrology.com/products/large_volume_tracking_positioning/ispace/. Accessed January 12, 2011.

[Noll 06] S. Nölle. *Augmented Reality als Vergleichswerkzeug am Beispiel der Automobilindustrie*. PhD thesis, Technische Universität München, 2006.

[Opto 11] "*NDI Optotrak Certus HD* website". http://www.ndigital.com/industrial/certushd.php. Accessed January 12, 2011.

[Pent 06] 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*, 2006.

[Pent 07] K. Pentenrieder, C. Bade, F. Doil, and P. Meier. "Augmented Reality-based factory planning-an application tailored to industrial needs". In: *6th IEEE and ACM International Symposium on Mixed and Augmented Reality, 2007. ISMAR 2007*, pp. 1–9, 2007.

[Pent 08] K. Pentenrieder and P. Meier. "Registration Approaches for Augmented Reality—A Crucial Aspect for Successful Industrial Application". In: *Third International Conference on Computer Graphics and Applications (GRAPP)*, Funchal, Portugal, January 2008.

[Pent 09] K. Pentenrieder. *Augmented Reality based Factory Planning*. PhD thesis, Technische Universität München, München, 2009.

[PRES 11] "*PRESENCCIA* project website". http://www.presenccia.org. Accessed January 12, 2011.

[Pust 04] D. Pustka. *Handling Error in Ubiquitous Tracking Setups*. Master's thesis, TU München, Munich, Germany, August 2004.

[Pust 06a] 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.

[Pust 06b] D. Pustka, M. Huber, M. Bauer, and G. Klinker. "Spatial Relationship Patterns: Elements of Reusable Tracking andCalibration Systems". In: *Proc. IEEE International Symposium on Mixed and Augmented Reality (ISMAR'06)*, October 2006.

[Pust 07] D. Pustka, M. Huber, F. Echtler, and P. Keitler. "UTQL: The Ubiquitous Tracking Query Language v1.0". Tech. Rep. TUM-I0718, Institut für Informatik, Technische Universität München, 2007.

[Pust 08] D. Pustka and G. Klinker. "Dynamic Gyroscope Fusion in Ubiquitous Tracking Environments". In: *Proc. 7th International Symposium on Mixed and Augmented Reality (ISMAR)*, September 2008.

[Pust 10] D. Pustka, J. Willneff, O. Wenisch, P. Lükewille, K. Achatz, P. Keitler, and G. Klinker. "Determining the Point of Minimum Error for 6DOF Pose Uncertainty Representation". In: *Proc. 9th International Symposium on Mixed and Augmented Reality (ISMAR)*, October 2010.

[Pust 11] D. Pustka. *Automatic Data Flow Construction for Ubiquitous Tracking Systems*. PhD thesis, Technische Universität München, 2011.

[Rege 07] H. Regenbrecht. "Industrial Augmented Reality Applications". In: *Emerging Technologies of Augmented Reality: Interfaces and Design*, pp. 283–304, Idea Group, 2007.

[Reit 01] G. Reitmayr and D. Schmalstieg. "An open software architecture for virtual reality interaction". In: *Proc. ACM symposium on Virtual reality software and technology*, pp. 47–54, ACM New York, NY, USA, 2001.

[Reit 07] G. Reitmayr and T. Drummond. "Initialisation for visual tracking in urban environments". In: *IEEE/ACM International Symposium on Mixed and Augmented Reality (ISMAR'07)*, 2007.

[Robe 04] C. Robert and G. Casella. *Monte Carlo statistical methods*. Springer Verlag, 2004.

[Rohl 94] R. Rohling, P. Munger, J. M. Hollerbach, and T. Peters. "Comparison of Relative Accuracy Between a Mechanical and an Optical Position Tracker for Image-Guided Neurosurgery". In: *Journal of Image Guided Surgery*, pp. 277–282, Wiley and Sons Inc, 1994.

[Roll 00] J. Rolland, L. Davis, and Y. Baillot. *Augmented Reality and Wearable Computers*, Chap. A survey of Tracking Technology for Virtual Environments. Lawrence Erlbaum Press, 2000.

[Samp 82] P. Sampson. "Fitting conic sections to very scattereddata: An iterative refinement of the Bookstein algorithm". *Computer Graphics and Image Processing*, Vol. 18, No. 1, pp. 97–108, 1982.

[Sand 05] C. Sandor and G. Klinker. "A Rapid Prototyping Software Infrastructure for User Interfaces in Ubiquitous Augmented Reality". In: *Journal for Personal and Ubiquitous Computing*, Springer Verlag, 2005.

[Sato 06] K. Satoh, K. Takemoto, S. Uchiyama, and H. Yamamoto. "A registration evaluation system using an industrial robot". In: *IEEE/ACM International Symposium on Mixed and Augmented Reality,2006. ISMAR 2006*, pp. 79–87, 2006.

[Scha 09] G. Schall, E. Mendez, E. Kruijff, E. Veas, S. Junghanns, B. Reitinger, and D. Schmalstieg. "Handheld augmented reality for underground infrastructure visualization". *Personal and Ubiquitous Computing*, Vol. 13, No. 4, pp. 281–291, 2009.

[Schl 11] M. Schlegel. *Zeitkalibrierung in Augmented Reality Anwendungen*. PhD thesis, Technische Universität München, 2011.

[Schm 09] J. Schmidt, D. Berg, H. Ploeg, and L. Ploeg. “Precision, repeatability and accuracy of OptotrakSUP alignright174;/SUP optical motion tracking systems”. *International Journal of Experimental and Computational Biomechanics*, Vol. 1, No. 1, pp. 114–127, 2009.

[Schw 07] B. Schwerdtfeger and G. Klinker. “Hybrid Information Presentation: Combining a Portable Augmented Reality Laser Projector and a Conventional Computer Display”. In: *Proc. Shortpapers and Posters of the 13th Eurographics Symposium on Virtual Environments, 10th Immersive Projection Technology Workshop (IPT - EGVE 2007)*, July 2007.

[Schw 09] B. Schwerdtfeger, R. Reif, W. A. Günthner, G. Klinker, D. Hamacher, L. Schega, I. Böckelmann, F. Doil, and J. Tümler. “Pick-by-Vision: A First Stress Test”. In: *Proc. 10th IEEE and ACM International Symposium on Mixed and Augmented reality (ISMAR’09)*, October 2009.

[Schw 10] B. Schwerdtfeger. *Pick-by-Vision: Bringing HMD-based Augmented Reality into the Warehouse*. PhD thesis, Technische Universität München, München, 2010.

[Serr 08] M. Serrano, D. Juras, and L. Nigay. “A three-dimensional characterization space of software components for rapidly developing multimodal interfaces”. In: *Proc. 10th international conference on Multimodal interfaces*, pp. 149–156, ACM New York, NY, USA, 2008.

[Simu 11] “*Mathworks Matlab Simulink* website”. http://www.mathworks.com/products/simulink/. Accessed January 12, 2011.

[Stra 92] P. Strauss and R. Carey. “An Object-Oriented 3D Graphics Toolkit”. *Computer Graphics*, Vol. 26, No. 2, pp. 341–349, July 1992. Siggraph ’92.

[Tayl 01] R. M. Taylor, T. C. Hudson, A. Seeger, H. Weber, J. Juliano, and A. T. Helser. “VRPN: a device-independent, network-transparent VR peripheral system”. In: *Proc. ACM symposium on Virtual reality software andtechnology*, pp. 55–61, ACM Press, 2001.

[Thru 05] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. MIT press, Cambridge, Massachusetts, USA, 2005.

[Tonn 09] M. Tönnis and M. Huber. “May AR Manipulate Users Subconsciously?”. In: *Proc. 1st European Future Technologies Conference and Exhibition (FET)*, Apr. 2009.

[trac 11] “*trackframe* project website”. http://www.trackframe.de. Accessed January 12, 2011.

[Tsai 87] R. Tsai. "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses". *IEEE Journal of robotics and Automation*, Vol. 3, No. 4, pp. 323–344, 1987.

[Tsai 88] R. Tsai and R. Lenz. "Real time versatile robotics hand/eye calibration using 3 D machine vision". In: *Proc. IEEE International Conference on Robotics and Automation*, pp. 554–561, Philadelphia, USA, 1988.

[Tuce 02] M. Tuceryan, Y. Genc, and N. Navab. "Single-Point Active Alignment Method (SPAAM) for Optical See-Through HMD Calibration for Augmented Reality". *Presence: Teleoperators and Virtual Environments*, Vol. 11, No. 3, pp. 259–276, June 2002.

[Ubi 11] "*Ubitrack* and *trackman* project website". http://campar.in.tum.de/UbiTrack/WebHome. Accessed January 12, 2011.

[UbiS 11] "*Ubitrack* project subversion repository". https://camplinux.in.tum.de/ubitrack. Accessed January 12, 2011.

[UTPa 11a] "*UTQL* data types XML schema". http://ar.in.tum.de/static/files/ubitrack/utql/utql_types.xsd. Accessed January 12, 2011.

[UTPa 11b] "*UTQL* pattern template XML schema". http://ar.in.tum.de/static/files/ubitrack/utql/utql_templates.xsd. Accessed January 12, 2011.

[UTQL 11] "*UTQL* XML schema". http://ar.in.tum.de/static/files/ubitrack/utql/utql.xsd. Accessed January 12, 2011.

[VDI 02] "Optical 3D measuring system—imaging systems with point-by-point probing". VDI/VDE guideline 2634/1, 2002.

[VeVa 11] "*Verification and Validation in Mathworks Matlab Simulink* website". http://www.mathworks.com/products/simverification/. Accessed January 12, 2011.

[Vico 11] "*Vicon* company website". http://www.vicon.com. Accessed January 12, 2011.

[Wagn 05] M. Wagner. *Tracking With Multiple Sensors*. PhD thesis, Technische Universität München, 2005.

[Wagn 08] D. Wagner, G. Reitmayr, A. Mulloni, and D. Drummond, T. andSchmalstieg. "Pose tracking from natural features on mobile phones". In: *7th IEEE/ACM International Symposium on Mixed and Augmented Reality, 2008. ISMAR 2008*, pp. 125–134, 2008.

[Wagn 10] D. Wagner, A. Mulloni, T. Langlotz, and D. Schmalstieg. "Real-time Panoramic Mapping and Tracking on Mobile Phones". In: *Proc. IEEE Virtual Reality*, Boston, USA, 2010.

[Welc 01] G. Welch and G. Bishop. "An Introduction to the Kalman Filter". In: *Computer Graphics, Annual Conference on Computer Graphics & Interactive Techniques (SIGGRAPH 2001)*, ACM Press, Addison-Wesley, Los Angeles, CA, USA, August 2001.

[Welc 02] G. Welch and E. Foxlin. "Motion tracking: No silver bullet, but a respectable arsenal". *IEEE Computer Graphics and Applications*, Vol. 22, No. 6, pp. 24–38, 2002.

[Welc 96] G. Welch. *SCAAT: Incremental Tracking with Incomplete Information*. PhD thesis, University of North Carolina at Chapel Hill, TR 96-051, 1996.

[Welc 97] G. Welch and G. Bishop. "SCAAT: Incremental tracking with incomplete information". In: *Proceedings of the 24th annual conference on computer graphics and interactive techniques*, pp. 333–344, ACM Press/Addison-Wesley Publishing Co., 1997.

[Wern 93] J. Wernecke. *The Inventor Mentor: Programming Object-Oriented 3D Graphics with Open Inventor (TM), Release 2*. Addison-Wesley, Reading, MA, 1993.

[Wile 05] A. Wiles, D. Frantz, D. Swart, *et al.* "NDI Accuracy Assessment Kit Guidelines". Tech. Rep., Northern Digital Inc., 2005.

[Wolt 85] H. Woltring, R. Huiskes, A. De Lange, and F. Veldpaus. "Finite centroid and helical axis estimation from noisy landmark measurements in the study of human joint kinematics". *Journal of biomechanics*, Vol. 18, No. 5, pp. 379–389, 1985.

[Wust 07a] H. Wüst, A. Pagani, and D. Stricker. "Feature management for efficient camera tracking". In: *Proc. 8th Asian conference on computer vision*, pp. 769–778, Springer-Verlag, 2007.

[Wust 07b] H. Wüst, F. Wientapper, and D. Stricker. "Adaptable model-based tracking using analysis-by-synthesis techniques". In: *Computer Analysis of Images and Patterns*, pp. 20–27, Springer, 2007.

[XSen 11] "*XSens* website". http://www.xsens.com. Accessed January 12, 2011.

[Zach 97] G. Zachmann. "Distortion correction of magnetic fields for position tracking". In: *Proc. Cmputer Graphics International*, pp. 213–220, 1997.

[Zaeh 06] M. Zaeh and W. Vogl. "Interactive laser-projection for programming industrial robots". In: *Proc. Fifth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR)*, pp. 125–128, IEEE Computer Society, 2006.

[Zhan 02] X. Zhang, S. Fronz, and N. Navab. “Visual marker detection and decoding in AR systems: A comparative study”. In: *Proc. 1st International Symposium on Mixed and Augmented Reality*, p. 97, IEEE Computer Society, 2002.

[Zhua 94] H. Zhuang and Z. Qu. “A new identification Jacobian for robotic hand/-eye calibration”. *Systems, Man and Cybernetics, IEEE Transactions on*, Vol. 24, No. 8, pp. 1284–1287, 1994.

Glossary

accuracy The degree of absence of systematic error in overall error. Or, the "closeness of the agreement between the result of a measurement and a true value of the measurand [ISO 08]. 99, 101, 248

backward-propagation Propagation of uncertainties toward the parameters of a functional model the result of which is known, effectively inverting the functional model. 113

base pattern Spatial relationship pattern that only has an output section. Typically, such a pattern represents some tracking system or a static transformation (cf. Figure 5.7(a)). 49, 239–241, 244, 246

base SRG Spatial relationship graph that consists of base patterns only and therefore does not contain any deduced spatial relationships (cf. Figure 5.2(a)). 43, 241

calibration Estimation of the constants that describe the behavior of a physical system or device. 82

chain of uncertainty Errors of various sources contribute to the overall error of the result in typical functional models (or data flow graphs) for IAR. 124, 243

competitive fusion An over-determined problem is solved by the fusion of measurements by some statistical means that minimizes the uncertainty of the result. Also called concurrent fusion. 28, 53, 239

complementary fusion Two measurements are combined to obtain a result that none of the involved tracking systems could deliver alone. 28, 53

component Short form of data flow component. 244, 245

component key Unique identifier of a logical data flow component belonging to a module. In case of a single tracking system observing multiple markers, for example, the component keys are given in terms of the IDs of the observed markers (cf. Figure 5.2(a)). 56, 239

concurrent fusion Synonym for competitive fusion. 28

cooperative fusion One tracking system provides measurements that are needed by the other tracking system to work. In some cases, one tracking system is needed to

initialize the other, e.g., to constrain the search window for fiducials (see also detection), in other cases, just several spatial transformations need to be concatenated to obtain the desired result. 28, 53

corresponding measurement A pair of measurements provided by two distinct modalities, e.g., tracking systems, that adheres to certain temporal and/or spatial constraints. Typically needed for registration. 52, 84, 150, 164, 241, 245, 246

data flow component Operational counterpart of a spatial relationship pattern. Each data flow component represents the implementation of a driver or algorithm in Ubitrack. Measurements are consumed on the input ports and produced on the output ports. Data flow components corresponding to base patterns only have output ports whereas data flow components corresponding to query patterns only have input ports. 46, 47, 239–243, 245, 247

data flow graph Directed, cyclic graph with data flow components as its nodes and communication pathes between data flow components as edges. It represents the operational perspective of a full SRG (cf. Figure 5.4). It can be instantiated using the Ubitrack library to obtain a real-time tracking data flow network. 239–243

data flow network Instance of a data flow graph. 240, 246

data type Type of a measurement, particularly important are position, orientation, or pose. Both, edges in the spatial relationship graph and ports in the data flow graph are normally annotated with the data type in square brackets, for example [3D Position] for position. An absent data type implies type pose. An input edge/input port can only be matched against / connected with an output edge/output port if the data types are equal. 43, 44, 54, 240–244, 246, 247

detection The initialization of tracking. Often detection is a harder problem than tracking. In computer vision, for example, known fiducials have to be initially found in the first frame before they can be tracked in subsequent frames. Also called *localization*. 2, 10, 240, 247

discrete measurement Automatic acquisition of measurements in a discrete time interval. 83, 85

dynamic Opposite of static. A dynamic spatial transformation may change over time. This is the default and not explicitly stated in the graphical representation of the corresponding edge (cf. Figure 5.2(b)). 240, 246

edge See input edge and output edge. 240, 241, 243, 244, 246

edge matching User interface metaphor used for the identification of exactly one input edge (the *matching edge*) with exactly one output edge (the *matched edge*) belonging to distinct spatial relationship patterns (cf. Figure 6.3). Edge matching

implies node unification on the respective source and sink nodes of both involved edges. This operation effectively constructs the data flow graph.. 69, 246, 247

elementary specification of sensor uncertainty Specification of the granularity (data type) as well as the magnitude of error that should be reasonably expected for an assumed tracking system. Should be as generic as possible but as specific as necessary. 98, 119, 137

error "Result of a measurement minus the true value of the measurand" [ISO 08]. 101, 102, 239, 241, 244–247

estimation error Norm of the estimated minus the true parameters. 142

expansion Mechanism to group similar measurements having the same data type and synchronization type on the expansion input ports of a data flow component. Used in particular by registration patterns to form two groups of corresponding measurements. There are mainly two expansion types, space-expansion and time-expansion. 241, 246, 247

expansion input port Special type of input port that allows for different ways of aggregation of similar measurements by expansion. 58, 241

explicit space-expansion Variant of space-expansion to group a smaller, fixed number of similar measurements that are already known at configuration time. Each measurements is represented explicitly by a distinct input edge in the corresponding explicitly space-expanded spatial relationship pattern (cf. Figure 5.12(b)). 58, 246

fiducial Characteristic object or feature used for tracking. 11, 240, 242, 244, 247

forward propagation Propagation of uncertainties toward the result of a functional model the parameters of which are known. 113, 243

full pattern Spatial relationship pattern that has a non-empty input section and a non-empty output section. Typically, such a pattern represents some tracking algorithm (cf. Figure 5.6(a)). 49, 241, 242, 244, 246

full SRG Spatial relationship graph that consists of base patterns, full patterns, and query patterns, and which in particular contains all the spatial relationships needed by the AR application, as opposed to the base SRG (cf. Figure 5.2(b)). For each full SRG, a data flow graph exists that represents the operational perspective corresponding to this semantic description of spatial relationships. 43, 46, 240

functional model Mathematical description that maps a measurement (parameters) to another measurement (result). In this context, the functional model is given in terms of a data flow graph. 114, 239, 241, 243, 244

functionally complementary fusion Combination of measurements with different data types resulting in a more general measurement. For example, a position and an orientation can be combined to a pose. 28

fusion Combination of measurements from different tracking systems. 239, 246

heterogeneous tracking environment Environment equipped with various tracking systems of distinct types. 24

input edge Spatial transformation that is assuemed to be given by a pattern, associated with a distinct data type and synchronization type. Part of the pattern's input section. Graphically represented by a dashed arrow between two input nodes (cf. Figure 5.5(a)). Counterpart of an input port in the data flow graph. 48, 240–242, 244–247

input node Coordinate frame contained in the pattern's input section. Both, input edges and output edges can originate in or point at an input node. Graphically represented by a dashed circle (cf. Figure 5.5(a)). 48, 242, 243

input port Measurement input consuming the parameters needed for the computations inside a data flow component, associated with a distinct data type and synchronization type. Counterpart of an input edge in the spatial relationship graph. 46, 47, 240–242, 244–247

input section Constellation of input nodes and input edges that fully determines the prerequisites for embedment of the pattern in a spatial relationship graph. Graphically represented by dashed arrows and circles. Only full patterns (cf. Figure 5.6(a)) and query patterns Figure 5.8(a) have an input section. 48, 241, 242, 244–246

inside-in Combination of the outside-in and inside-out paradigms. 14, 19

inside-out Variant of tracking where the sensors are rigidly mounted to the moving object and observe static fiducials in the environment. 13, 242

marker Special object providing several fiducials for tracking. 239, 242–244, 246

marker-based Optical tracking using markers. 11

marker-less Optical tracking without using markers, i.e., solely based on natural features as fiducials. 12

measurement In this context, typically a spatial information of type position, orientation, or pose that is provided by a tracking system. 44, 239–248

measurement tool Synonym for pointing device. 10

meta-pattern Part of a spatial relationship graph (i.e. a combination of patterns), typically represents a reusable best-practice solution for a common problem (cf. Figure 8.3). 38, 89, 90

module Concept used in Ubitrack to manage multiple logical data flow components belonging to a single physical resource, for example a tracking system observing multiple markers. 56, 61, 239, 243

module key Unique identifier of the physical resource represented by a module. In case of a single tracking system observing multiple markers, the module key is given in terms of the ID of the single node that remained after the node unification of all output nodes representing the same single physical tracking system (cf. Figure 5.2(a)). 56

Monte Carlo Method for the forward propagation of uncertainties along the chain of uncertainty by systematic variation of the input measurements based on an assumed statistical model and repeated evaluation of the given functional model. Does not require analytical treatment of the functional model. 117

node See input node and output node. 241, 246

node unification User interface metaphor used for the identification of various input nodes and output nodes belonging to distinct spatial relationship patterns in the spatial relationship graph (cf. Figure 6.2). 68, 241, 243, 246

offline Not allowing for real-time tracking. 13

online Allowing for real-time tracking. 13

orientation Data type describing the 3D orientation of an object in space, represented by a [3D Rotation] label on the edge/port. 162, 240, 242, 244, 247

output edge Spatial transformation that is provided by a pattern, associated with a distinct data type and synchronization type. Part of the pattern's output section. Graphically represented by a solid arrow originating from an input node or output node and pointing at an input node or output node (cf. Figure 5.5(a). Counterpart of an output port in the data flow graph. 48, 240, 242–245, 247

output node Coordinate frame contained in the pattern's output section. Only output edges can originate in or point at an output node. Graphically represented by a solid circle (cf. Figure 5.7(a)). 48, 243, 244

output port Measurement output providing the computational result of a data flow component, associated with a distinct data type and synchronization type. Counterpart of an output edge in the spatial relationship graph. 46, 240, 243–245, 247

output section Constellation of output nodes and output edges that fully determines the additional spatial relationships added to a spatial relationship graph by the embedment of this pattern. Graphically represented by solid arrows. Only base patterns (cf. Figure 5.7(a)) and full patterns (cf. Figure 5.6(a)) have an output section. 48, 239, 241, 243, 244, 246

outside-in Variant of tracking where the sensors are rigidly mounted to the environment and observe moving fiducials. 13, 242

pattern Short form of spatial relationship pattern. 241–245

pattern signature Syntactical properties of a pattern, as determined by its input section and output section. 49

point probing The activity of touching a certain point with a pointing device to measure its position. 107, 121, 172, 244

point-by-point probing Point probing of multiple points in sequential order. 4

pointing device Device with a tip to point at certain points to measure their position, equipped with a marker for tracking. 10, 11, 242, 244

port See input port and output port. 243, 244, 246

pose Data type describing the 6DoF position and orientation of an object, represented by a [Pose] label on the edge/port (cf. Figures 5.2(b) and 5.4). If not explicitly stated otherwise, edges/ports in this context have data type pose (cf. Figure 7.2). 240, 242, 244, 247

position Data type describing the 3D position of an object in space, represented by a [3D Position] label on the edge/port (cf. Figure 7.3). 163, 240, 242, 244, 247

precision The degree of absence of random error in overall error. 99, 248

probing device Synonym for pointing device. 11

propagation of uncertainties Investigation of the error behavior of a certain measurement that is related to some other measurement with an associated statistical model by a known functional model. Also called error propagation. 239, 241

PULL Measurement updates are triggered (requested) on some input edge/input port upon demand, typically by the AR application. Graphically represented by a light green (cf. Figure 7.4). 44, 244–247

pull-push conversion By a pull-push conversion pattern/component, a measurement of synchronization type PULL on the input edge/input port can be converted to PUSH on the output edge/output port. For example, the Sampler component can be used. 61, 244

PUSH Measurement updates are provided on some output edge/output port with a constant update rate, typically by a tracking system driver. Graphically represented by a reddish color associated with the corresponding push source, typically a light red, or another reddish color in case of multiple push sources (cf. Figure 7.4). 44, 244–247

push source Source of PUSH measurement events that features a distinct timing and update rate, consisting of at least one data flow component. A synchronization of push sources may be beneficial, if supported. It is typically accomplished in hardware by using a common clock and sync signal, or in software by a relative harmonization of timestamps. All output edges/output ports belonging to the same push source share the same reddish color in their graphical representation. In addition to that, the respective data flow component or data flow components that generate the events, are also represented in the same red (cf. Figures 5.2(b) and 5.4). 245, 247

push-pull conversion By a push-pull conversion pattern/component, a measurement of synchronization type PUSH on the input edge/input port can be converted to PULL on the output edge/output port. Any interpolation component can be used, e.g., the Linear Interpolation component. 61, 245

query pattern Spatial relationship pattern that only has an input section. Typically, such a pattern represents an interface to the AR application (cf. Figure 5.8(a)). 49, 240–242, 246

random error Characterization of a jittery or Gaussian error behavior. Or, the "result of a measurement minus the mean that would result from an infinite number of measurements of the same measurand carried out under repeatability conditions" [ISO 08]. "Random error is equal to error minus systematic error." [ISO 08]. Also called *non-systematic* or *dynamic error* [Azum 97]. 102, 244, 245, 247

registration Estimation of the static spatial transformation between two coordinate frames. 82, 240, 241

repeatability The "closeness of the agreement between the results of successive measurements of the same measurand carried out under the same conditions of measurement" [ISO 08]. 100, 102, 245, 247

reproducibility The "closeness of the agreement between the results of measurements of the same measurand carried out under changed conditions of measurement" [ISO 08]. 100, 102

reproduction of measurement A pair of corresponding measurements is recoreded at different points in time. 85

residual error Norm of the measured minus the estimated parameters. 104, 142

sensor Device, e.g., a camera, that allows for the measurement of spatial parameters of an object using some physical principle. A sensor or a combination of sensors constitutes a tracking system. 242, 244, 246, 247

sensor fusion See fusion. 24

simplified space-expansion Variant of space-expansion to group larger, previously unknown numbers of similar measurements. The whole group of measurements is represented by a single input edge having a list data type in the corresponding simply space-expanded spatial relationship pattern (cf. Figure 5.12(c)). 58, 246

simultaneous measurement A pair of corresponding measurements is recoreded at the same time. 84, 86

space-expansion Opposite of time-expansion. Variant of expansion where multiple similar measurements are provided simultaneously. Can be accomplished either by explicit space-expansion or by simplified space-expansion. 56, 241, 246, 247

spatial relationship graph Directed, cyclic graph that describes the spatial arrangement of tracking systems, markers, and other real or virtual objects of a tracking infrastructure. Nodes represent coordinate frames, edges represent transformations between these coordinate frames. The spatial relationship graph is constructed from spatial relationship patterns, using the node unification (cf. Figure 6.2) and edge matching (cf. Figure 6.3) user interface metaphors. 239–244, 246

spatial relationship pattern Directed, cyclic graph that describes the spatial arrangement of coordinate frames that is relevant for a given particular tracking system or tracking algorithm. A full pattern consists of an input section and an output section whereas a base pattern lacks the input section and a query pattern lacks the output section. 34, 40, 47, 239–241, 243–247

spatially or temporarily complementary fusion Combination of measurements with identical data types to overcome temporal or spatial restrictions of the involved tracking systems. For example, Two identical optical tracking systems cover two distinct tracking volumes that form a common working area. 28

static Opposite of dynamic. A static spatial transformation is assumed to be rigid and must not change over time. This is explicitly stated in the graphcial representation of the edge in the spatial relationship pattern by the [Static] keyword (cf. Figure 7.2), if applicable.. 239, 240, 242, 245, 246

statistical model Specification of the error behavior attributed to a measurement. 114, 243, 244

synchronization type Describes how measurement updates are processed in the data flow network, either PUSH or PULL. Graphically represented by the color of the corresponding edge/port (cf. Figure 7.4). An input edge/input port can only be

matched against / connected with an output edge/output port if the synchronization types are equal. 44, 54, 241–245, 247

synchronized-PUSH A spatial relationship pattern/data flow component may specify some or all of its PUSH input edges/input ports to have equal timestamps. In this case, they have to belong to the same trigger group, i.e., they have to originate from the same push source. 59

systematic error Characterization of an error that does not have an expectation of **0**. Or, the "mean that would result from an infinite number of measurements of the same measurand carried out under repeatability conditions minus a true value of the measurand" [ISO 08]. It is "equal to error minus random error" [ISO 08]. Furthermore, "like the true value, systematic error and its causes cannot be completely known" [ISO 08]. Also called *static error* [Azum 97]. 102, 239, 245, 247

time-expansion Opposite of space-expansion. Variant of expansion where multiple similar measurements are provided sequentially, represented by a single input edge having a primitive (non-list) data type in the corresponding time-expanded spatial relationship pattern (cf. Figure 5.12(a)). 56, 241, 246

tracking In this context: the continual determination of an object's position, orientation, or pose in real-time. In computer vision, for example, the movement of fiducials is tracked between consecutive frames. Successful detection is a prerequisite for tracking. 2, 10, 240–244, 247

tracking system Technical system consisting of sensors that allows for tracking. In the context of Ubitrack, a tracking system is a black-box system that typically provides position, orientation, or pose measurements. Also called *tracker*. 2, 239–243, 245, 246

trackman Graphical SRG and DFG editor and Ubitrack frontend. 34, 42, 66

trigger component Special type of a data flow component where the synchronization types of all input ports and the single output port of the trigger group are not hard-coded but determined at configuration time. Corresponding input edges and input ports are depicted in black and gray, as long as their synchronization type is not yet determined (cf. Figure 5.6), red or green otherwise (cf. Figures 5.2(b) and 5.4). 59, 247

trigger group A subset of all input ports, as well as the single output port of a trigger component together form the trigger group. Typically, all input ports of the trigger component belong to the trigger group, e.g., in the Multiplication component. Two simple rules apply. First, if at least on input port is forced by edge matching to have synchronization type PUSH, the single output port must also have synchronization type PUSH. Second, if the single output port is forced by edge matching to have synchronization type PULL, all input ports automatically must also have synchronization type PULL. 59, 62, 247

Ubitrack Tracking framework and library developed at *Fachgebiet Augmented Reality* [FAR 11]. 34, 40, 240, 243, 247

uncertainty A "parameter, associated with the result of a measurement, that characterizes the dispersion of the values that could reasonably be attributed to the measurand" [ISO 08], thus a reasonable combination of precision and accuracy in one single quantity. 101, 102, 105, 239, 243

user-triggered measurement The acquisition of each measurement is triggered by the user. 83

Acronyms

DFG data flow graph. 46, 247

DFN data flow network. 46

SRG spatial relationship graph. 34, 40, 42, 46, 247