Technische Universität München
Fakultät für Informatik

2nd Joint Advanced Summer School 2004
Course 3: Ubiquitous Tracking for Augmented Reality

Prof. Gudrun Klinker, Ph.D.
Martin Wagner

June 2004

Contents

1 Augmented Reality: Overview and Current Systems

— **Markus Michael Geipel**

1.1 A Brief Scenario for Motivation

Before we start defining Augmented Reality (AR), looking at specific technologies and current systems, let us imagine a brief scenario, well suited for the application of AR. In fact this scenario is an adopted version of the scenario of the heARt-Project [16, 32] on which we will have a closer view later on in section 1.5.

1.1.1 The Situation

A surgeon wants to conduct a cardiovascular surgery, basically a heart surgery. In order to operate successfully she needs to know the exact position of the surrounding bones of the chest, the veins and arteries, and last but not least the heart. As humans are not transparent, formally there was only one choice: The surgeon had to open the patient's body up, to see all necessary details with her own eyes. Note, that experience would not prevent this practice because of the fact that every human has a slightly different physiognomy [1]. Today new technologies, like Magnetic Resonance Imaging (MRI) and Computed Tomography (CT) supply the surgeon with additional information and thus enable minimal invasive surgery. But new problems arise.

1.1.2 The Problems

The additional information is presented on monitors. This is a problem in three ways:

1. The surgeon cannot look at the patient and the screen at the same time: The workflow is disrupted.

2. Looking at the screen while working on the patient is not ergonomic.

3. The screen is 2D whereas the patients body is 3D. So, the surgeon has to transfer and interpret the information which further complicates the work process.

As we will see later on in section 1.5, the characteristics of the surgeon's problem repeat themselves in many other fields: automotive industry, military and civil aviation, repair and maintenance, education and many more.

[1] There are known cases, where even the heart is not at the right, or more exactly, left place.

1.1.3 The Vision

What features should a technology, AR in this case, provide to tackle these problems?

1. The information should be provided where it is needed, directly at the patient.
2. The user, that is the surgeon should be enabled to access and interact with the information in an ergonomically way.
3. The information should be in three dimensions like the application of the subject: the patient's body.

How to achieve this vision, that's the core issue of Augmented Reality.

1.2 Defining Augmented Reality

Let us look at some definitions and how they relate to our scenario. We will see that according to Azuma [5, 6], three major aspects have to work together in order to define AR. More restrictive definitions, limiting AR only to the use of Head-Mounted-Displays (HMD), will not be considered here.

1.2.1 Combining Real and Virtual

In the surgeon's scenario we found out that the information should be provided directly at the patient's body. This means mixing real and virtual objects: the patient's body and the sensor data. And that is what is meant, when it is stated in "Confluence of Computer Vision and Augmented Reality" [19] that . . .

> Augmented Reality (AR) is a technology in which a user's view of the real world is enhanced or augmented with additional information generated from a computer model.

There are two facts to be kept in mind: First, Head-Mounted-Displays (HMD) are only one technology to combine the real and the virtual world, although it is the most frequent case. Besides other visual display technologies which will be discussed later in section 1.4, all other senses are potential subject to augmentation. Often it makes sense to incorporate acoustic as well as tactile interfaces. An example of an AR-Application using only tactile augmentation is the NAVI-Project [23]. Basically it provides an outdoor navigation system for visually disabled persons. Second, augmentation does not necessarily mean that something is added, although this is the common case. Augmentation also includes filtering out information.

Another example for the mixture of real and virtual objects [2] is special effects in movies. In the Ridley Scott film "Gladiator", [30] battle scenes in the Roman Colosseum are shown. The gladiators are real, the colosseum isn't! The question is: Is this also AR? The answer: no, it isn't. Clearly it is a mixture of real and virtual, but AR is more. This leads us directly to the next point.

[2] We will discuss the degree of mixing real and virtual in section 1.3 and technologies to accomplish it in section 1.4.3.

1.2.2 Interactive in Real Time

> Using AR technology, users can interact with a combination of real and virtual objects in a natural way.
> *from "Confluence of Computer Vision and Augmented Reality" [19]*

In contrast to special effects in a movie, an AR-Application has to be interactive in real time. In the movie you can't just turn your head to see what hides behind a computer generated statue of Markus Aurelius because hollywood needs hours or even days of computer as well as manpower to produce one single scene.

In the surgeon's scenario this means that the viewpoint has to be changed everytime the surgeon moves his head. All she sees in her HMD, is rendered at this very moment. But what if you watch a movie with a HMD? In the HMD the movie appears to be floating about one or two meters in front of you. So it's a mixture of the real and the virtual world. There is also interaction at real time: The picture is rendered into your field of view in real time. You are also able to somehow interact: push fast-forward, adjust the brightness and so on. Is this enough to be called AR? Like with the special effects the answers is no, AR is more.

1.2.3 Registered in 3-Dimensions

In order to enable complex interactions the real and the virtual world have to be tightly coupled. This is called "Registered in 3-Dimensions". In the surgeon's scenario this means that the sensor data that is mixed with the patient's body has to stick to the patient's body even when this body is moved or the surgeon changes her viewpoint. This does not mean, that virtual objects are "glued" to the markers, they have their own three dimentional positions. The markers are only orientation points to bring both worlds in sync.

Further more most virtual objects should mix in a convenient way with reality. This means that the real object may be occluded by real ones and vice versa according to their 3-dimensional position. To do this, the AR-System has to keep track of all the real objects that has to be aligned. This may be accomplished in a static or dynamic way. For example some objects in the environment, like the walls or machinery, could be assumed to be static. The computer is only fed once with the position data. Some objects cannot be assumed static: the people interacting with the system and movable objects like tools. These have to be tracked at real time. We will look at tracking later on in section 1.4.4.

To cut a long story short: If we mix reality and virtuality, if we do it in an interactive way in real time and if we align all objects, real and virtual ones with each other, the vision of the transparent patient may become true. This is the very essence of AR.

1.3 A Taxonomy of Reality and Virtuality

> Augmented Reality (AR) is a variation of Virtual Environments (VE), of Virtual Reality as it is more commonly called.[5]
> *Ronald T. Azuma*

How can the degree of "variation" be classified? Paul Milgram and Herman Colquhoun[22] suggest a Taxonomy consisting of three dimentions to classify applications dealing with mixtures of real and virtual ...

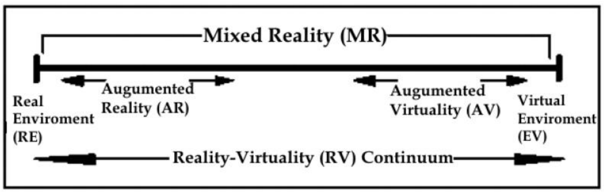

Figure 1.1:

The Reality-Virtuality Continuum Figure 1.1 shows this continuum. We see that Reality and Virtual Environments form the two poles. AR lies near to the real environment but can vary from application to application: If more of the world is modeled we go towards the VE pole whare as when less of the world is modelled we go towards the RE pole.

Centricity . . . is mainly concerned with the users view on the subject of manipulation: is it Exocentric or Egocentric? Exocentric for example would be a external, for example a bird's view, on the scene. The typical Egocentric situation is when the user sees the augmented world from his, her or the subjects postion. If we wanted to control a remote robot, viewing it through cameras ovserving the robot from outside would be external, viewing the world through cameras atached to the robot would be egocentric.

Control-Display Congruence Generally the higher the congruence, the more inuitive is the control. For example if you drive a car, the control metaphors are (forward, backward, turn left, turn right) this is congruent with the reactions you see through the windshield. But if you use a map for your navigation, the metaphors are (north, east, south, west) and not concruent with the controll before mentioned control metaphors[3] Another possibility might be an automatic car, that accepts acustic commants like "drive west 100 meters an then head north".

Ergo: Typical AR-applications lie near the RE pole of the continuum, often use an egocentric viewpoint and aim to minimize the Control-Display Congruence.

1.4 Technologies

1.4.1 Overview

What technologies are necessary to implement an AR-System? Figure 1.2 shows an overview. An AR-System has to be linked with a database containing information about the virtual world. The unidirectional connection with the real world is realized through sensing the environment. A tracking subsystem computes the position data to provide the system with an representation of the real world. Last but not least, the user interacts with the AR-System via an User-Interface. Of course, as the user is part of the environment, he may interact with it directly and thus interact indirectly with the AR-system.

[3] This fact maybe one reason for the success of navigationsystem, that translate the incongruant informations on a map to congruent ones like "please turn left in 100 meters".

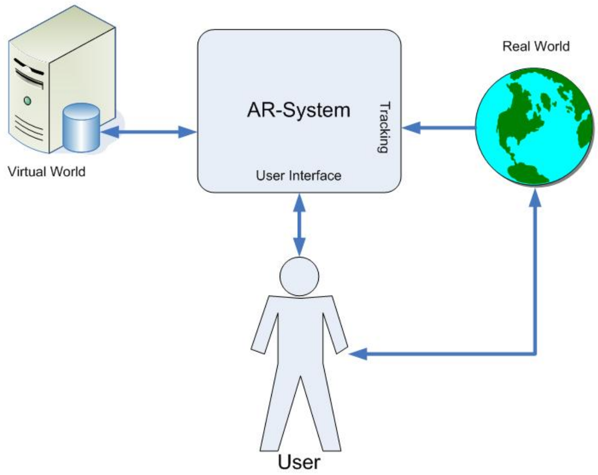

Figure 1.2: Scheme of an AR-System

1.4.2 User Interfaces

The first question is: How can a user interact with an AR-System? The traditional computer interaction with mouse and keyboard is definitely not appropriate for AR. Basically all senses humans use to communicate with their environment can also be used to communicate with an AR-System: Vision, Hearing, Touch. Voilá, a digest of possibilities and examples.

Visual As we have already seen, vision is the dominating user interface (UI) in AR so it will be discussed in depth in section 1.4.3. At this point let us only analyse how the user can give visual command to the AR-System. A typically visual interaction is the practice of gesture recognition: Through the use of image processing the AR-System recognizes gestures made by the user. This technique was incorporated in the MIT's Kids room [18, 27]. The users, kids in this case, were taught gestures by virtual monsters. The monsters then "danced" with the kids by doing the gestures that the kids did. Gestures included crouching, throwing arms-up to make a "Y", flapping, and spinning.

Acoustic Speech recognition and speech synthesis still suffer from technical problems [25, 20, 28]. To cite only one example: The english sentence "Time flies like arrows" can be interpreted by a computer, lacking the background knowledge, in four different ways[4]. So, purely acoustic interaction with computers like the one in Arthur C. Clarke's novel "2001, A Space Odyssey" is even now, in 2004, not reasonable[5]. Nevertheless acoustic signals can heavily support the UI: In the sheep-application [31], for example, a new sheep is inserted by the acoustic command "insert". The new sheep's position however is given by touching the virtual pasture with a "magic wand".

Tangible/Haptic Besides, the typical tangible UIs like a 3D-mouse or a joystick, also tangible objects, like the above mentioned "magic wand" can act as UI. A tangible object is a tracked real object with a representation in the virtual world and can thus be used

[4] To find them is left as an exercise.

[5] I guess a today's computer would quote the famous HAL sentence "I'm sorry, Dave, I can't do this!" if he only could.

as an input device. In the CAR-application [9] a tangible toy car is used to navigate through a virtual city projected on a desk, which is shown in figure 1.3. Additionally a monitor shows the view through the windshield. Also a world in miniature can be zoomed and tilted by moving a tangible cardboard. Another tangible UI: The Personal Interaction Panel (PIP) of the Studierstube [29]: A tracked cardboard in combination with a tracked pen.

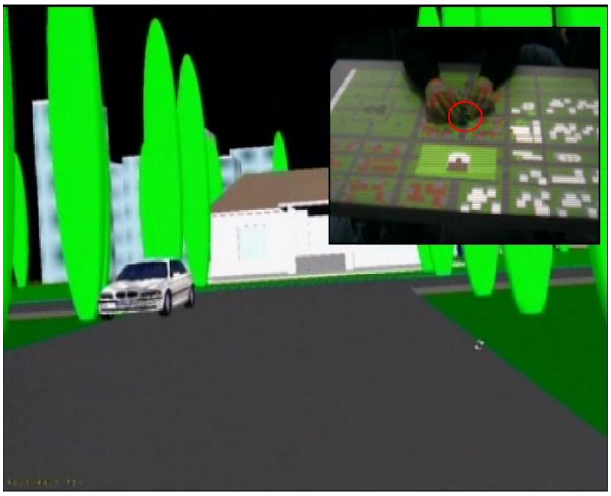

Figure 1.3: Embedded picture: The tangible car on the projected city map. Main picture: View throught the windshield.

Despite of the possibilities of every single presented interface it makes sense to combine the modalities to form an even more powerful UI. If we do this, the UI is called multi-modal. And in fact most AR-Systems of today, make heavy use of multi-modal UIs [13, 27, 29]

1.4.3 Information Representation

Although visualisation addresses only one of the possible modalities subject to augmentation, it will be discussed separately and in more depth. The reason: visualisation makes up the biggest part of the UI as well as one main part of implementation effort.

Display Devices

Head-Mounted-Displays The most common visualisation technique and also the most complicated.

PDAs (mobile displays) Especially for mobile applications small displays and computers are needed: PDAs. Since PDAs can also be equipped with small and relatively cheap cameras, they are a often used for AR. They are used for example in the Sheep-Application [27] or the PAARTI project [14].

Projection For applications with a varying user numbers where HMDs are just not convenient, projection is a handy alternative. Projection is also possible in 3-dimensions via shutter glasses, polarized projection or anaglyphic rendering[6].

[6]Two views are rendered: One in red and one in blue. Via colored glasses a 3D effect can be accomplished.

Monitors This is the smaller alternative to projection. Nevertheless, for some applications this is sufficient. Shutter glasses and anaglyphic rendering are also applicable for monitors.

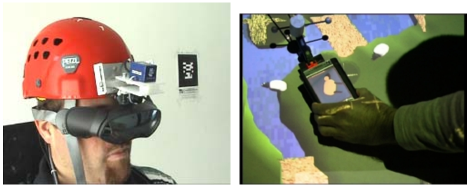

Figure 1.4: An HMD and a PDA in combination with a projection screen

How to Mix Virtuality and Reality

Basically there are two possibilities to mix virtual images with the real world:

Physical This can be accomplished via half transparent mirrors, for example in HMDs or on desks in combination with a projection screen. The advantage is that the real world can be seen with one's own eyes. The disadvantage: Virtual objects can not fully occlude reality.

Video see through Reality is filmed with cameras, so that the mixing problem is reduced to the problem of mixing two video streams. One way to mix these streams is chroma keying: The renderer responsible for the virtual part uses one specific color to mark transparent regions. The mixing process can be done by hardware mixers. This is exactly what is known as blue box-technique in film studios. Another way is the software approach: The "reality stream" is fed to the rendering software which uses it as background. The use of the blue box-technique is out dated. Today, even smaller computers are fast enough to use video streams as backgound in the rendering software.

The next question is how to master occlusions. Or more specifically: how does the render, knowing only a video stream, know what virtual objects should be occluded? Here are two methods to tackle occlusion:

Registering and Rendering Real Objects The idea is to keep track of all real objects and render them too, for example, in the chroma keying color, so that the video mixer will make them transparent. The obvious problem is, this approach is doomed to fail in an rapidly changing or uncontrolled environment: Keeping track of all real objects is simply impossible.

Depthmaps Through Stereo Pictures Depth maps of the environment can be constructed based on two images taken from two different viewpoints. These maps can then be used to compute the occlusions [19]. The appealing in this idea is its flexibility: Nothing has to be tracked. However there are situations, where this technique fails: If the environment lacks textures, for example, if it contains monochrome objects, the computation of depth maps is impossible.

Challenges

Some problems and drawbacks of visualization have already been mentioned in the preceeding section 1.4.3. But there are same more. Simply speaking, HMDs do not meet the quality standards of the human eye: The resolution, the brightness, the contrast, all these are not good enough to compete with reality. Some case studies: The eyes light sensitivity is logarithmic, but most cameras have a linear sensitivity. Imagine a light bulb. When you look at it you will see the filament as well as the bulb and its environment. Try this with a usual camera: the bulb will be just white, the surrounding just black. That is a problem for every technique that relies on cameras: Let us take video see through for example. Also optical see through suffers problems: On sunny days the augmentations have not enough contrast to be readable. At night they will glare and reality fades behind the half reflective glasses. Also safety is a problem. This might sound strange at first. The virtual objects will not bite, will they? No, they won't! But imagine the following: You are wearing a closed video see through HMD (so, you are only seeing a video image), showing some lurid augmentations, and you are driving with your car on a German autobahn at speed of, let's say, 210 kilometres per hour. Would you feel comfortable? Surely not! So, we see, in some environments, AR can be a safety problem. First by being a source of distraction and second by limitation the user's perception.

1.4.4 Tracking

As mentioned in section 1.2 the process of continously aligning virtual objects with the real world can be achieved by tracking the user's position. In this section we will discuss several tracking approaches, their strength and their drawbacks.

Ways of Tracking

Various techniques can be used to track objects or persons. These are the most important ones:

Time Frequency Measurements (Time of Flight) The best-known system in this category is surely the American "Global Positioning System" GPS [7]. GPS satellites transmit signals to equipment on the ground. These GPS devices on the ground receive passively the signals, including the satellites position and a timestamp, of at least four satellites. Taking in account the light speed-limit for the propagation of these signals, the GPS device can calculate its own position. Standard GPS has an accuracy of about 10 meters for non-military use. One meter accuracy can be reached by Differential GPS. At the moment GPS is without adversaries. This could change with the introduction of the European Galileo Positioning System, planned in 2008. However, both systems suffer a major drawback: Accuracy and the dependency of line of sight to the satellites. Another time of flight based system is ultrasound tracking. Basically ultrasound-tracking works similar to GPS, only in small and with ultrasound as signal medium.

[7] There are people holding the opinion that GPS is not a tracking technology because nothing is actively tracked, only anonymous time codes are received: This makes a GPS receiver only a box that knows its own position, therefore it is not a tracker. Nevertheless, we will consider GPS receivers as a tracking technology because they can be used to keep track of their own position.

Optical Tracking The basic principle is that one or more cameras detect one or more well-known markers. Two different approaches are possible. First, fixed cameras at known positions and moving markers: This is called "outside in" because the cameras are locking at the marked object. The second possibility is just the opposite: "inside out"; one camera on the object looks at markers at fixed and known positions. Both techniques depend on the line of sight between the markers and the cameras. The advantage: high accuracy.

Inertial Tracking This form of tracking includes gyroscopes and accelerometers that sense changes in orientation or speed. Inertial tracking is mainly used as an additive to the other forms of tracking because it suffers drift and thus has to be recalibrated very frequently.

Magnetic Tracking Magnetic tracking is based the fact that every magnetic coils cause a change in the sourounding magnetc field that can be sensed. Magnetic tracking provides good speed and high robustness. But it suffers the drawback that sensing falls victim to disturbances caused by every ferromagnetic object in the room and thus the precicion is reduced. And of course it can only be used in a closed environment.

Of course there are same more technologies like mechanical tracking. But let us take a glimpse on a special form of tracking that should not be left unmentioned: eye-tracking. The point is to calculate the user's direction of sight. This data can be the key to context aware applications as well as new forms of user interfaces. In the CAR Project, for example, eye tracking data is used to interact with an information panel [9]. Basically the point is: If the user looks at an item, auxiliary information is presented. Additionally the view cones can be visualized via AR for ergonomic studies.

Challenges

As we have seen in our short "tour de tracking", every tracking system has its drawbacks. Besides these, according to Azuma[5], two major problem classes exist:

Static errors These include mainly problems in the hardware configuration like optical distortion in camera based system, errors in the output data of tracking systems, mechanical misalignments or improper parameter configuration. The problems may look trivial but they are not: How should one measure output errors? With another tracker who's output isn't error free either? How can one guarantee that all "aligned" objects, like markers, to stay exactly at the same place?

Dynamic errors These are caused system delays: The time the signal needs from the sensing hardware to processing, to the renderer and finally to the screen. While static errors are always present, dynamic errors appear only during motion. Despite improving the processing speed, several techniques exist to reduce the lag: In video see through the video stream can be delayed a bit to match the rendered stream. The renderer may also render more than it is needed in order to be prepared for sudden changes of the view port. And last but not least the system can try to predict future view port changes.

Note, that these problems are that serious due to the extremely high accuracy standards: The angular accuracy needed for an acceptable visual augmentation is a small fraction of a degree! Virtual objects not properly aligned or lagging behind are not just unappealing; they are irritating and can even cause motion sickness.

1.5 Application

We have already seen one area of application in the surgeon's scenario. We will now give a brief survey of already addressed application domains. In order to structure the survey the applications are grouped by the type of information they present to the user. Of course this is just a theoretical scheme were also hybrids are possible. Nevertheless we will see that the vast majority of applications fit quite perfect into this scheme.

1.5.1 Annotating the Environment

Typical for this class of applications is to provide meta information about the environment. Applications of this class make up the biggest part of AR-applications because it comprises less difficulties than the other two classes: First, the visualisation is less demanding. Labelling or text has not to be realistic, it just has to be readable[8]. Second, also the tracking may be less precise: if a label, a minimap or a status bar is not exactly aligned there is not much of a problem; at least not so much as for the other classes. Let's browse some examples ...

Navigation Being an everyday feature of modern cars, a navigation system for pedestrians are quite novel. One of the first project in this field was the "Virtual Touring Machine" of the Columbia University [15]. The system aims to help the freshmen to orientate himself on the campus area by displaying labels on buildings and even further information, like opening times of libraries.

Figure 1.5: User with the "Touring Machine" equipment and a view through the HMD, (c) 1997, S. Feiner, B. MacIntyre, T. Hoellerer, and A. Webster, Columbia University

Military and Police In fact, for jet pilots AR has already become reality: The cockpit is equipped with a Head up-Display (HUD) which overlays the front view with tactical information, mainly vector graphics, for the pilot. The upcoming military application concentrates on the single infantry soldier. The US-Army's Concepts Division of the Marine Corps Combat Development Command states

[8] As we will see in section 1.4.3, even this can be a problem

> ... the success of a military operation in an urbanized environment depends crucially on being able to provide navigation and coordination information to the individual marine level. [24]

One Project that explores the possibilities of AR for this specific scenario is the BARS Project [24] which is based on the before mentioned "Virtual Touring Mashine" [15]. An AR-System is used to propagate the high-level knowledge of the strategic planning staff to the single soldier on the battlefield. The soldier is equipped with an wearable AR-System connected via wireless communication to the command central. Also for policemen AR has the possibility to become a vital technology. An extensive survey concerning the application of AR for the police forces can be found in "Improving Our View of the World: Police and Augmented Reality Technology" [12]. The authors think about incorporating AR in the following fields: Patrol, SWAT Operations, Criminal Investigations, Training and Supervision. A concrete idea for SWAT Operation for example is a "friend or foe classification"-system that could minimize collateral damage caused by "friendly fire".

Repair and Maintenance Especially in the sectors Repair, Maintenance and also Construction, AR can be very useful. In Germany, the ARVIKA consortium [4] brings together industry and research to explore new AR-solutions in this field. The Applications include AR-support for crash tests and measuring the air flow in the flume aswell as machine and plant construction. What else? In the field of aviation, Boeing is experimenting with AR-supported wire assembly [17]. An example application of AR in the automotive branche is the "Intelligent Welding Gun" [14]. In the construction of prototype cars steel studs need to be welded into the car frame at specific locations. The precise measurement and marking of the target locations is very time-consuming. As it is a prototype construction this process can't be automated. The solution: The welding gun is tracked and equipped with a small display, that guides the user to the next target location. The "Intelligent Welding Gun" is already productively used by BMW.

1.5.2 Displaying Auxiliary Sensor Information

The characteristic of that class of applications is the use of sensor information to extend the perception of the user. A significant example is ...

Medical There have been various projects exploring this application area: A research group at the University of North Carolina [8] has built a system conducting ultrasound scannings of pregnant women and presenting them three dimensions via an HMD to the user. As we have already seen in the surgeon's scenario in section 1.1, surgery is an ideal field for AR support: An example for heart surgery is the HeartProject. Its aim was to visualize the bones of the chest and the heart to provide a navigation aid for minimal invasive surgery [16, 32]. At MIT the AI lab developed an AR-Application that uses MRI and EEG scans of the patient's brain to support neurosurgery [10]. Here the difficulty for the doctor was to keep in mind the exact 3-dimensional target regions for the operation. The system provided a 3-dimensional model of a patient's brain as guidance during the operation. It was successfully evaluated in several real neurosurgeries.

1.5.3 Visualizing Artificial Data

The following class of applications uses artificial data the user wants to edit. Note that this data is not meta information but primary subject to data processing and manipulation through the user. All of the following examples are based on the Studierstube system[29] which is well suited for this class of applications. Here the examples ...

Design An interesting example of presenting design in AR is the "Virtual Showcase"[33]: A virtual exhibit is projected inside a glass housing mixed with real objects. No special glasses are needed to achieve this effect! It is based upon the mixture of real and virtual via semi-transparent, half-silvered mirrors.

Scientific Visualization and Education Generally numerical simulations produce huge 3-dimensional data sets. It is the task of scientists do interpret and often also to discuss these data sets. AR can bring these 3-dimensional structures into the real world. Scientists can walk around the structures manipulate and discuss them in a collaborative AR-application: This was, for example, shown in numerous application among them an application that targets especially collaborative work with the visualized data, described in "Collaborative Augmented Reality: Exploring Dynamical Systems" [1]. and one that explores techniques for the visualisation of 3-dimentional flows[2]. An area with very similar requirements is education, mainly in the field of mathematics: This is the aim of the Construct3D-application [11]. It too is based on Studierstube and shown in Figure 1.6.

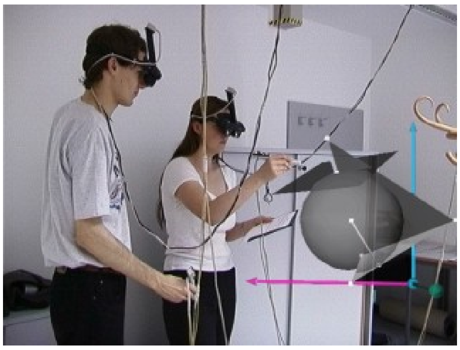

Figure 1.6: Math you can (almost) handle: Two students studying geometry in AR

1.6 Current Systems

1.6.1 Overview

So far we have seen lots of different applications. The questions is: Do they stand alone or do there exist Toolkits or Frameworks that make the implementation of AR-applications easier? Luckily, the answer is yes. There exists, for example, the ARToolKit [3], an open source vision tracking library, that enables easy development of a wide range of Augmented Reality applications. Applications using the ARToolKit include, for example, the MagicBook [21] of the the University of Washington. There also exist entire Frameworks for AR-applications:

The "Studierstube"-system [29] [9] uses a distributed heterogeneous architecture. It is designed to support the implementation of collaborative augmented reality application. In order to do this the "Studierstube" provides a framework for multi-windowing in three dimensions and ready to use UI-elements like the Personal Interaction Panel: A tablet, which is augmented with virtual widgets[10] and controlled via a tracked pen. Another AR-framework is the DWARF (the 'F' already stands for Framework) ...

1.6.2 DWARF, An Example AR-Framework

We will now look at a sample framework, the DWARF, in more detail. DWARF stands for Distributed Wearable Augmented Reality Framework. The speciality of DWARF is its structure: It is a network of dynamically cooperating services [7]. This yields several advantages:

Distribution The service concept allows free distribution of the services: The tracking services for example may run on one computer while another one is concerned with the rendering etc.

Heterogenity The computers running DWARF may use different operating systems and may be of different type: for tracking a desktop computer and for wearable visualisation PDAs for example. Thus DWARF is well suited to mix wearable and ubiquitous computing with stationary computing. Even at the service level, different programming languages can be used: for example, Python for prototyping, Java for application logic and C++ for time critical parts.

Modularity The service concept leads to high modularity all hardware details are abstracted by services. As we will see in the next section, a service that need tracking data, just advertises a Need for it in its Service Description. Thus Services become highly reusable and flexible.

Dynamically Cooperating Services

How do these dynamically cooperating services work? A Service is the basic software unit of DWARF. Services can have data sources and data sinks. The sources are called Abilities and the sinks, Needs. Abilities, Needs and Services are annotated with meta information, including the type of data they need or provide, their name, their host and many more. All this meta information for one Service is called the Service Description and can be written in XML. A new service may be started manually or by the middleware in case the Service provides data needed by other services. With the knowledge of the Service Description, the DWARF-Middleware connects the Needs and the Abilities dynamically at run-time. Thus the essential part of the Middleware is the Service-Manager which has to run on every computer, hosting Services. The whole communication is based on CORBA. Communication between Services can be established via three different types of channels, including shared memory, messages and call-backs. All running Services, their Needs and Abilities, their connections, their Attributes and status can be visualized via a graphical front end. Figure 1.7 shows a network of Services and their connections. With the front end, Service Descriptions of all services in the distributed system can even be edited during run-time.

[9] According to the authors, the word studierstube is derived from the study room of Goethe's Faust.
[10] The word "widget" is a contraction of "window" and "gadget".

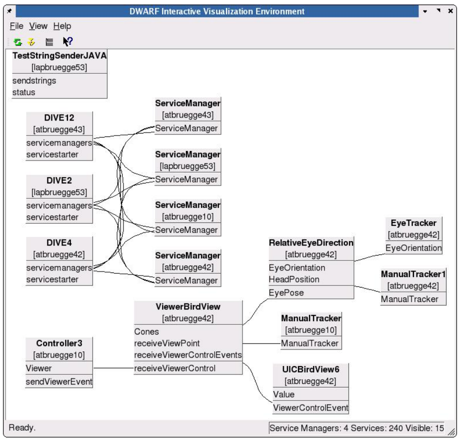

Figure 1.7: Dynamically cooperating services of DWARF visualized.

Subsystems

The DWARF Services are grouped in subsystems according to "Towards a System of Patterns for Augmented Reality Systems" [26] One subsystem of DWARF has already been mentioned. These are the other ones:

Tracking The tracking hardware is encapsulated in services of this subsystem. Further more this subsystem provides a calibration service to fine tune the positions of tracked objects at an abstract level.

Presentation The core service of this subsystem is the Viewer, an OpenGL-Renderer where VRML-Models can be loaded and registered with Needs for position data.

Interaction All services concerned with user interfaced belong to this subsystem. One main service for example is the UIController, a Petri net-based controller for multimodal user input.

Context Processing and gathering of context information belongs to services of the context subsystem.

Worldmodel All services modelling or representing virtual or real objects belong to the world model subsystem. For example in the Sheep-Application[31, 27] virtual sheep roam around a pasture projected on a desk. Every sheep is one service of this subsystem, providing position data and look and feel for the presentation services.

Application This subsystem is the place where the application developer can put the application-specific logic. This logic provides the glue, that holds together all services and subsystems, used for one specific application.

1.7 The Future

As we have seen, the applications of AR are as manifold as the technical problems. There are also manufacturers who have capitulated in the HMD sector: Sony for example has abandoned their "Glasstron" series. Surely AR for private use at home will take time to come: The hardware will not be affordable in the near future. But there are also domains in which AR has already, silently taken its place: television, for example. More and more TV-stations are augmenting their sport events. In Formula 1, the cars get dynamically labelled and in soccer the advertising is no longer taped to the rails of the station but digitally painted on them in the studio. So the question is not "Will AR come?" but "When will AR come and for whom?". Surely, sooner or later when the technologies become low-priced due to mass production, HMDs and AR will be as common as mobile phones and laptops are today.

Bibliography

[1] *Collaborative Augmented Reality: Exploring Dynamical Systems, pp. 459-462*, Los Alamitos, CA, USA, 1997, IEEE Computer Society Press.

[2] *Real-Time Techniques For 3D Flow Visualization. pp. 305-312*, North Carolina, USA, October 18-23 1998, Research Triangle Park.

[3] *Internet Presentation of the ARToolKit*, March 2004.

[4] *Internet Presentation of the ARVIKA Project*, March 2004.

[5] R. Azuma, *A Survey of Augmented Reality, pp. 355-385*, Presence: Teleoperators and Virtual Environments, vol. 6, no. 4, (1997).

[6] R. Azuma, Y. Baillot, R. Behringer, S. Feiner, S. Julier, and B. MacIntyre, *Recent Advances in Augmented Reality, pp. 34-47*, IEEE Computer Graphics and Applications, (2001).

[7] M. Bauer, B. Bruegge, G. Klinker, A. MacWilliams, T. Reicher, S. Ri, C. Sandor, and M. Wagner, *Design of a Component-Based Augmented Reality Framework*.

[8] R. D. Bergeron and A. E. Kaufman, eds., *Case Study: Observing a Volume Rendered Fetus within a Pregnant Patient. pp. 364-368*, Washington, DC, USA, October 17-21 1994, IEEE Computer Society.

[9] *Internet Presentation of the CAR Project*, March 2004.

[10] A. Chabrerie, F. Ozlen, S. Nakajima, M. E. Leventon, H. Atsumi, W. E. L. Grimson, E. Keeve, S. Helmers, J. Riviello Jr., G. Holmes II, F. Duffy, F. A. Jolesz, R. Kikinis, and P. Black, *Three-Dimensional Reconstruction and Surgical Navigation in Pediatric Epilepsy Surgery.*, Pediatric Neurosurgery 27: pp. 304-310, 1997. SPL Technical Report no. 104.

[11] *Internet Presentation of the Construct3D-Project*, March 2004.

[12] T. J. Cowper and M. E. Buerger, *Improving Our View of the World: Police and Augmented Reality Technology.*

[13] *DWARF Project Homepage.* DWARF Project Homepage, March 2004.

[14] 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 Virtual and Augmented Reality Applications in Manufacturing, Chapter 17, S. Ong and A. Nee, eds., Springer Verlag, 2003.

[15] S. Feiner, B. MacIntyre, T. Hllerer, and A. Webster, *A Touring Machine: Prototyping 3D Mobile Augmented Reality Systems for Exploring the Urban Environment, pp. 74-81*, Proceedings of ISWC, (1997).

[16] *heARt Project Homepage.* heARt Project Homepage, March 2004.

[17] N. Jim, *Wired Magazine: Issue 5.10, Wiring the Jet Set*, (1997).

[18] *Kidsroom Project Homepage.* WWW-White, March 2004.

[19] G. J. Klinker, K. H. Ahlers, D. E. Breen, P.-Y. Chevalier, C. Crompton, D. S. Greer, D. Koller, A. Kramer, E. Rose, M. Tuceryan, and R. T. Whitaker, *Confluence of Computer Vision and Interactive Graphics for Augmented Reality*, Presence: Teleoperations and Virtual Environments, Special Issue on Augmented Reality, (1997).

[20] R. Kurzweil, *When Will HAL Understand What We Are Saying? Computer Speech Recognition and Understanding", chapter 7*, The MIT Press, Cambridge and Massachusetts, 3rd ed., 2000.

[21] *Internet Presentation of the MagicBook Project*, March 2004.

[22] P. Milgram and C. Jr., *A Taxonomy of Real and Virtual World Display Integration, chapter 1, pp. 1-26*, IEICE Trans. Information Systems, vol. E77-D, no. 12.

[23] *Internet Presentation of the NAVI Project*, March 2004.

[24] *Battlefield Augmented Reality (BARS).* Office of Naval Research Homepage, retrieved January 28 2002.

[25] J. P. Olive, *"The Talking Computer": Text to Speech Synthesis, chapter 6*, The MIT Press, Cambridge and Massachusetts, 3rd ed., 2000.

[26] T. Reicher, A. MacWilliams, and B. Bruegge, *Towards a System of Patterns for Augmented Reality Systems*, in International Workshop on Software Technology for Augmented Reality Systems (STARS 2003), 2003.

[27] C. Sandor, A. MacWilliams, M. Wagner, M. Bauer, and G. Klinker, *SHEEP: The Shared Environment Entertainment Pasture*, in EEE and ACM International Symposium on Mixed and Augmented Reality ISMAR 2002, 2002.

[28] R. C. Schank, *"I'm sorry, Dave, I'm afraid I can't do that": How Could HAL Use Language?, chapter 8*, The MIT Press, Cambridge and Massachusetts, 3rd ed., 2000.

[29] D. e. a. Schmalstieg, *The Studierstube Augmented Reality Project, pp. 33-54*, Presence, vol. 11, no. 1, (2002).

[30] R. Scott, *Gladiator.* DVD, Dreamworks Pictures and Universal Pictures, 2000.

[31] *Internet Presentation of the SHEEP Project*, March 2004.

[32] J. Traub, M. Feuerstein, M. Bauer, E. U. Schirmbeck, H. Najafi, R. Bauernschmitt, and G. Klinker, *Augmented reality for port placement and navigation in robotically assisted minimally invasive cardiovascular surgery*, in 8th Annual Conference of the International Society for Computer Aided Surgery (ISCAS), 2004.

[33] *Internet Presentation of the Virtual Showcase-Project*, March 2004.

2 Ubiquitous and Context Aware Computing: Overview and Systems

— **Simon Bichler**

2.1 Motivation

2.1.1 Ubiquitous Computing

With more and more computing power becoming available, the limiting factor of computing is no longer processor speed or memory size but the user's attention. Over the next few years there will be a shift away from today's personal computer paradigm, that requires the user to pay full attention to the computer. The new ubiquitous computing paradigm, will remove the computer from the center of a users attention and into the background [2].

This shift in paradigm has been predicted by Mark Weiser over 10 years ago [12]. Mark Weiser used the example of writing to explain the idea of ubiquitous computing: Writing is an ubiquitous information technique today. Written information can not only be found in books or magazines, but everywhere around us: On street signs, billlboards, even on candy paper. In the ubiquitous computing paradigm computers will be just as immersed in our everyday lives as written information is today.

Another example Weiser uses in [12] is the electric motor. At the beginning of the 20th century in a factory there would be one engine, driving dozens to hundreds of different machines through a system of shafts and pulleys. With the advent of cheap electric motors each machine could have its own motor. Today there are usually several motors in a machine, for example a modern car. But the user does not have to be aware of all the electric motors in order to drive the car. In an analogy to this, in an ubiquitous computing environment the user will not have to be aware of all the computers around him in order to accomplish the task at hand.

2.1.2 History of Paradigms for Computing Systems

According to [13] the ubiquitous computing environment is a logical continuation of the evolution of computer systems:

1. The mainframe: Many users per computer
2. The personal computer: One user per computer
3. The ubiquitous computing environment: Many computers per user

Currently we are moving from the personal computer paradigm to the ubiquitous computing paradigm.

When we use the world wide web, we use all three paradigms at once: Webservers can be compared to mainframes, both have many concurrent users. The webbrowser usually runs on a personal computer with only one user at a time. Also, during a typical web browsing session the user visits many different webservers. But he does not even have to be aware of the fact, that the pages he visits are served by several different computers. The webservers are computers that have moved to the background, away from the users immediate attention.

But in the future ubiquitous computing will be even more removed from the users awareness. Computers will be embedded everywhere: In walls, chairs, clothing, light switches, etc. Although many of today's household appliances do already use microprocessors, ubiquitous computing will go much further. In real ubiquitous computing these small computers will be using many sensors to become aware of what is happening around them. They will also be networked together to share this information with each other.

2.1.3 Context awareness

When humans communicate with each other, information is not only transmitted by spoken language. Facial expressions, emotions, past and future events, the existence of other people in the room, etc. are also important aspects of human communication. This information does not have not to be made explicit, as long as the human communication partners are both aware of it. However, when communicating with a computer, it has to be told explicitly, what it should do. The implicit information from the situation is usually lost.

Context awareness means that an application does not solely rely on explicit input from the user. It also uses implicit information available in the situation it is used in. This implicit information is called context and applications that use this context are called context aware [3]. Context aware applications should be able to come closer to doing what the user wants than applications that do not use context.

A variety of definitions for context has been stated in literature, a quite in-depth discussion of the different approaches can be found in [3].

The context toolkit, presented in [3], uses the following definition:

Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.

A system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task.

Examples for context information according to [3] are:

- Computing environment: available processors, devices available for user input and display, network capacity, connectivity and costs of computing
- User environment: location, collectin of nearby people, and social situation
- Physical environment: lighting and noise level

Context awareness in mobile and ubiquitous computing

Context awareness is especially important for mobile and ubiquitous computing:

Firstly, in a non-static setting, there is more context information available. For example, people and devices moving around and computing services dynamically becoming available.

Secondly, the attention of the user may be limited. He might be walking or driving a car while using an application. In that case the explicit information provided to the application may be sparse. If the application can use implicit information from the situation it may be better able to unterstand what the user wants it to do.

Difficulties of using context

The use of context in an application is difficult for several reasons [3]:

- Context is acquired from non-traditional devices. That means that special drivers are needed to access them.
- Context must be abstracted to make sense. The pure sensor data is usually not meaningful. To interpret this sensor data is usually difficult.
- Context may be acquired from multiple distributed and heterogenous sources. These sources have to be connected to a common system
- Context is dynamic. Therefor the application has to be able to adapt tp changes to its surroundings.

2.2 Ubiquitous Computing Systems

There is still a lot of work to be done before we can expect to use Ubiquitous Computing systems, that work in the way that was suggested in section 2.1.1. However, research is going on and several systems have been implemented that can serve as building blocks for future ubiquitous computing systems.

2.2.1 Active Badges

The Active Badge System was developed between 1989 and 1992 [11]. It consists of two basic components: A badge that is carried by employees when they are inside the company building and a network of receivers across the building. The receivers are able to detect the badges and therefor the position of the employees within the building.

The badges are only 55x55x7 mm small and weigh 40g. Every 15 seconds they send a pulse-width modulated infrared signal. This signal is detected by one of the sensors that are placed around the building. The sensors are connected to a network and send the information about the location of the badge to a master station that keeps a log of the position of all the badges.

Because the badge has to emit a signal only every 15 seconds, the battery lasts for about one year. The relatively long interval also helps to avoid interferences between several different badges. Because people move relatively slowly in an office building the position can still be measured with sufficient accuracy.

A power saving mechanism is included in the active badge, that turns off the device when in a dark surrounding. For example when it has been put in a desk drawer or at night. This eliminates the need for a power switch that would complicate the use of the active badge more than nessecary.

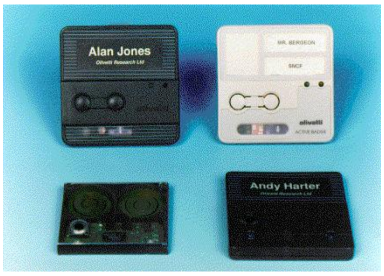

Figure 2.1: Several Active Badges at different stages of evolution

The initial application of the active badge system was to help a telephone receptionist to locate people and route incoming calls to a telephone close to their current position. Another application is to control access to secure areas.

2.2.2 i-Bean

The i-Bean is a short-range, ultra-low power wireless device. It can be used to acquire, process and transmit sensor data from an attached ring sensor. The ring sensor is fastendend around a persons finger and measures arterial blood volume waveforms and blood oxigen saturation. In combination this system allows online, long-term and continuous monitoring of vital signs.

The form of the sensor makes the i-Bean suitable for wearing over an extended period of time, where other sensor might become uncomfortabel.

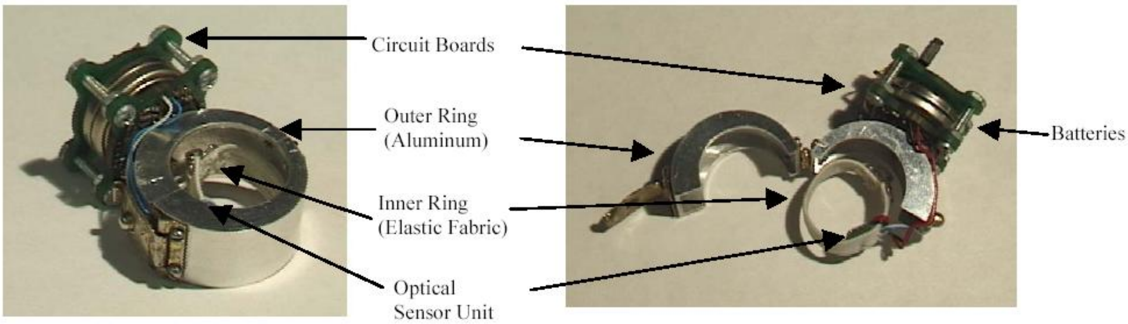

Figure 2.2: The i-Bean with attached finger ring sensor

To minimize the power-consumption of the i-Bean a wireless sensor network is used. It consists of many transceivers that pick up the signal from the i-Bean and route it to the monitoring computer via the shortest path available. So every single transmission has to travel only a short distance (up to 30 m) and can use less power.

2.2.3 Phidgets

Physical Widgets (Phidgets) [6] aim to be a physical analogy to graphical user interface widgets. Phidgets have been designed as building blocks for physical user interfaces. They support the development of such interfaces by providing a programming interface to a range of sensors and actuators. The interface is easy to use with a standard programming language.

The devices usable in this framework include servo motors, solenoids, LEDs, light, force and heat sensors, digital switches and a power bar that can switch each of its outlets seperatly.

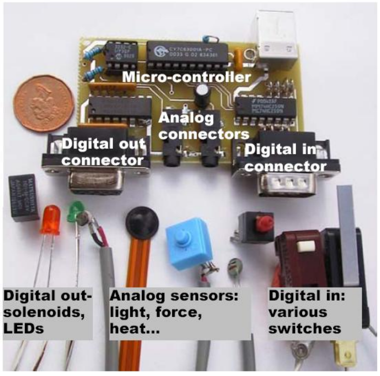

Figure 2.3: The phidget interface board and some devices that can be connected to it

Phidgets have some requirements that graphical widgets do not:

- Connection manager: While widgets are always available to an application, physical objects may be removed from or added to a system during runtime. The connection manager has to monitor the state of the physical devices and inform the application of changes.
- Identification: If there are several identical phidget devices connected to the system, it must be possible to distinguish between them.
- Simulation mode: Graphical widgets can be easily tested, but phidgets may not be available yet at the time the application is written. The framework therefor has to provide a simulation mode where the application can be tested without a physical device connected.

2.2.4 Project Aura

Project Aura [2] is trying to build an ubiquitous computing environment by integrating existing hardware and software. The two concepts that it applies are proactivity and self-tuning.

Proactivity means, that a system's layer should be able to anticipate requests from a higer layer instead of only reacting to it.

Aura is also self-tuning, that means that the system layers dynamically adapt their performance and resource usage to the demands made on them.

Some of the techniques implemented in project aura are:

- Cyber foraging: the use of staging servers close to a limited mobile device to reduce the impact of end-to-end Internet latency on intercative file-intensive applications.
- Wireless bandwidth advisor: the use of reasonable estimates of future available bandwidth to make informed decisions, e.g. on server selection.
- WaveLAN-based people locator: the use of signal strength and access point information of a WaveLAN to determine the position of persons carrying a laptop.

2.2.5 Portable Help Desk

The portable helpdesk (PHD) is an application built on top of the infrastructure of project Aura [2]. It allows the user to locate other people on his team and to find information about them.

Figure 2.4: The visual interface of the portable help desk

There are two user interfaces available: One is a visual interface (see figure 2.4), that shows a map of the surrounding area with information about people and resources nearby. The other is an audio interface that uses speech recognition and audio output. The audio interface is used, when the user cannot be distracted by looking at a screen. For example when he is walking somewhere. Both interface however use the same underlying database for the information.

2.2.6 DyPERS: Dynamic Personal Enhanced Reality System

DyPERS is a wearable system that uses augmented reality and computer vision. It is able to overlay real objects the user looks at with previously recorded video and audio material [8], [1].

The hardware of DyPERS consists of a head-up-display with a camera and a microphone, worn by the user. The display is connected via wireless audio/video transceivers to a graphic workstation (see figure 2.5). The workstation analyses the scene the user looks at. If it recognizes an object, that has a media clip associated with it, it replays that clip.

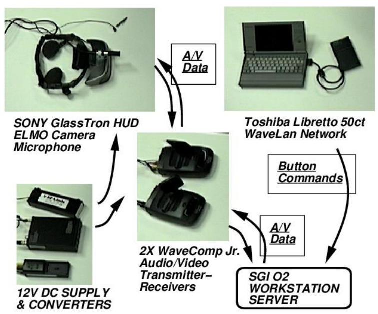

Figure 2.5: The hardware components of the DyPERS system.

In a first step the user has to record the video and audio clips that should later be associated with an object. This can be easily done by using the head mounted camera and microphone.

Then the recorded clip has to be associated with an object. Again, this can be done by looking at the object and signaling the graphics workstation to record the image. This image is then analyzed so it can be compared to the live video stream from the users head-mounted camera.

When the graphics workstation finds, that the live video matches the prerecorded image, it replays the associated video clip to the users head-mounted display. A statistical model is used to allow the recognition of the object even under different lighting conditions or from a different point of view.

This system could be used for a wide variety of scenarios. Some examples include:

- A TO-DO list could be stored on the users watch or on other personal items.
- A conversation can be recorded and associated with the business card of the person the user did talk to.
- A story teller can read a picture book and associate each picture with its spoken text passage. A child could then use the system to listen to the story while looking at the pictures in the book.
- A teacher could associate objects with the spoken names of the objects in a foreign language. A student could then use the system to learn the vocabulary of the language.
- Assembly instructions could be associated with the unassembled parts.
- A person with poor vision could listen to audio descriptions of the things he is looking at.

2.2.7 Disappearing Computer Initiative

The Disappearing Computer Initiative is a EU funded research project in ubiquitous computing [4]. It tries to focus on the following three objectives:

- Creating Artefacts: To embed computing and networking capabilities into common objects
- Emerging Functionality: To find ways of combining artefacts with limited capabilities in order to create new functionality
- People's Experience: To find ways in which the user's activities can be enhanced by ubiquitous computing

One of the many projects that are part of the Disappearing Computer Initiative is the Smart-Its project, described in the next section.

2.2.8 Smart-Its

Smart-Its [7] is a hardware platform that aims to provide a basis for the development of ubiquitous computing applications. It was developed as a collaboration between six partners in five countries: ETH Zurich (Switzerland), Interactive Institute (Sweden), Lancaster University (UK), University of Karlsruhe (Germany), the Victoria University (Sweden) and VTT (Finland). The research project was funded by the European Union.

The goal of the Smart-Its project is to provide a platform for researchers and developers of ubiquitous computing applications. With that platform it schould be possible to explore future applications with less overhead than is currently needed. The intention is to create a hardware analogy to a GUI toolkit for a desktop computer.

Smart-Its are used to embed sensors, computation and communication into common artifacts. The hardware design follows a modular approach that makes it easier to adapt the hardware to a specific situation. A Smart-It consists of the following modules:

- A Core-board with a wireless transceiver to communicate with other Smart-Its
- A standard sensor board for light, sound, pressure, acceleration and temperature
- Specialized sensor boards, e.g.: gas sensor, load sensor, video camera, etc.

Standard output facilities of a Smart-It are LEDs and a speaker. The wireless network that is used supports a speed of 125kbps. It allows 1024 nodes sending at one time.

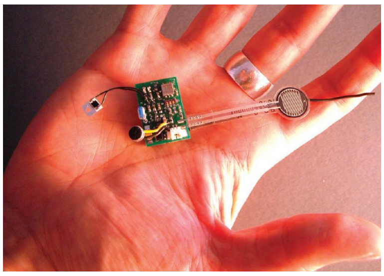

Figure 2.6: A Smart-It with several sensors attached

A Smart-It can be as small as 17 x 25 x 15 mm. It weighs 8 to 20 gramms including sensors and battery. It can be powered from several different sources, depending on the application: standard AAA batteries, rechargeable batteries or lithium coin cells. The Smart-Its platform supports power management for the core board, for the sensors and for other attached hardware. It has a lifetimes between several days to one year, depending on the amount of sensor use, processing and communication required by the application.

The Smart-Its perception API (PAPI) isolates application requests for sensor data from the internal sensor processing. Each Smart-It knows what sensors are attached to it and it shares this information with other Smart-Its. The sensor discovery range uses the physical distance to determine which sensor data is available. A Smart-It can request remote sensor values. It has four methods to request sensor data:

- Single value: One single value is read from the sensor
- Condition triggered: When the sensor data meets the condition the Smart-It is notified once
- Continous subscription: Every time the sensor data meets a specified condition the Smart-It is notified.
- Constant stream: Sensor data is read at full speed

2.2.9 Proactive Furniture Assembly

In a project described in [5] Smart-Its with several different sensors and LEDs were attached to an unassembled piece of flatpack furniture. The goal of the project was to immerse the assembly instructions for the furniture into the real world instead of printing them on paper or showing them on a computer screen.

The system was designed to recognize the users actions and translate them into the current state of assembly. It should then give recommendations for the next step to take to further assemble the furniture.

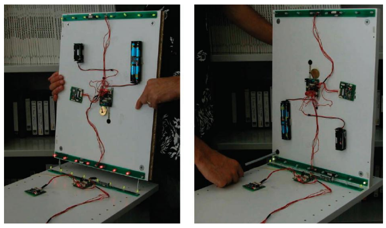

Figure 2.7: LEDs indicate the incorrect alignment of the boards (left) or show which screws have to be tightened (right)

The following sensors were attached to the furniture, using Smart-Its to allow them to communicate their measurements:

- Accelerometers to determine the orientation of each board
- Force sensors to observe screw tightening
- IR sensors to detect the co-location of boards

To present the instructions to the user several different techniques could be used. All of them have certain drawbacks:

Augmented Reality is too cumbersome and expensive.

Audible instructions make it difficult to address the different parts without teaching the user the right vocabulary first.

Instructions on a computer screen are not immediatly integrated with the task.

Therefor LEDs are used to suggest the next action to the user.

The system has several ways of giving feedback to the user:

- Blinking LEDs guide the users attention to the pieces he should start with.
- Positive feedback for correct actions: Correctly aligned edges are indicated by static green light patterns.
- Negative feedback for wrong actions: Wrong actions are signaled by red light patterns.
- Fine grain direction: Individual LEDs show where screws have to be tightened
- Notification of finished task: Synchronous flashing LEDs indicate that the task is finished

LEDs are able to give dynamic cues and enhance static affordances by dynamic instructions.

There are three principles that support learning by doing. These principles are all realized in this system.

- Explorability: The user can explore the possibilities of aligning the boards without penalization for aligning them incorrectly at first.
- Predictability: The furniture assembly works the way the user expects it to work. The hints given by the LEDs are consistent.
- Intrinsic guidance: The instructions are available without the user having to perform any special actions to access them

2.2.10 Load sensing Furniture

By adding load sensors to ordinary furniture it can be made aware of the actions perfomed on its surface. In the Smart-Its project this has been done with tables, shelves and even a whole office floor [10].

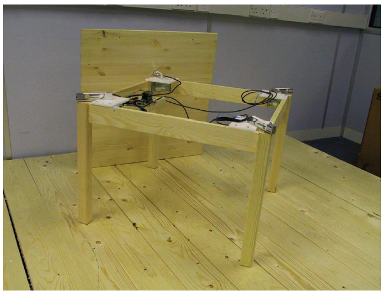

Figure 2.8: This dining table was augmented with load cells in each corner

Industrial load cell were attached at each corner of a dining table and connected to Smart-Its (see figure 2.8). The load cells thus became wireless networked sensors.

The load at each corner of the table was measured with a frequency of up to 200 Hz. It can of course be detected when an object is put down on the table or when one is removed from the table. But by comparing the relative loads at the corners, the center of gravity of the objects on the table can be found.

The motion of the center of gravity can be observed and used to detect interaction on the surface, for example someones finger moving across the table top. This effect has been exploited to use the entire table surface as a wireless pointing device to control a mouse cursor.

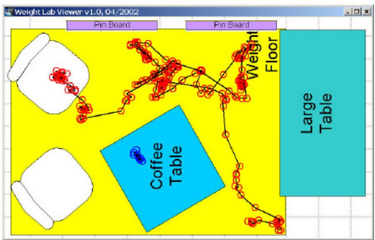

Figure 2.9: Actions in a load-augmented room can be tracked

When the floor of a room is put on load cells, the position of people inside the room can be tracked (see figure 2.10). An application using this information could easily be made context aware.

2.2.11 A-Life System

In [9] a system based on the Smart-Its platform is presented, that tries to aid avalanche rescue. It makes use of wearable sensors on mountaineers, skiers and snowboarders that are at risk of being buried by an avalanche.

Current radio-based tracking systems can only give the location of one victim at a time, but with multiple vitims the order in wich they are rescued can be crucial.

Figure 2.10: The A-Life system shows the victims positions, vital signs and surrounding conditions on a handheld device

The A-Life System connects oximeters, oxygen sensors and accelerometers with Smart-Its. The Smart-Its transmit environmental conditions and vital signs of a buried person to a handheld device. The rescuers can use this handheld device to find the position of the victims. He can also prioritize the victims by urgency based on their vital signs and surrounding conditions, e.g an air pocket at the victims face.

While it can be important to rescue the victims in a prioritized order, it also raises ethical questions, wether a computer should be allowed to decide which person to rescue first.

2.3 Issues of Ubiquitous and Context Aware Computing

The systems shown in the previous sections illustrate the state of ubiquitous computing today. The common problem seems to be, that the real world use value of all these applications is not very high yet.

For example, the use of Active Badges to route telephone calls seems absurd in a time, where everybody owns a mobile phone anway.

The main reason for selling furniture unassembled is, that it is cheaper to transport and store it that way.When you put several hundred Dollars worth of sensors and computers on a 50$ piece of furniture, the economic advantage is gone. In many cases the commercial use of ubiquitous and context aware coputing just makes no sense yet.

Some years into the future however, the calculation might look different. Computing power will continue to become cheaper, wireless communication will become more and more common. One day it will become economically feasible to implement systems similar to those shown above.

Bibliography

[1] B. Schiele, T. Jebara, and N. Oliver, *Sensory Augmented Computing: Wearing the Museum's Guide*, 2001.

[2] A. S. D. Garlan, D. Siewiorek and P. Steenkiste, *Project aura: Toward distraction-free pervasive computing*, 2002.

[3] A. Dey, *Providing Support for Building Context-Aware Applications*, Master's thesis, Georgia Institute of Technology, 2000.

[4] *Disappearing Computer Initiative*, 2001.

[5] Florian Michahelles, Stavors Antifakos, Jani Boutellier, Albrecht Schmidt and Bernt Schiele, *Instructions immersed into the real world – How your Furniture can teach you*, in The Fifth International Conference on Ubiquitous Computing, Seattle, USA, October 2003.

[6] S. Greenberg and C. Fitchett, *Phidgets: Easy development of physical interfaces through physical widgets*, in UIST, 2001, pp. 209–218.

[7] L. E. Holmquist, H.-W. Gellersen, G. Kortuem, A. Schmidt, M. Strohbach, S. Antifakos, F. Michahelles, B. Schiele, M. Beigl, and R. Maze, *Building Intelligent Environments with Smart-Its*, IEEE Comput. Graph. Appl., 24 (2004), pp. 56–64.

[8] T. Jebara, B. Schiele, N. Oliver, and A. Pentland, *DyPERS: Dynamic Personal Enhanced Reality System*, 1998.

[9] F. Michahelles and B. Schiele, *A-Life: Saving Lives in Avalanches*, in Fifth International Conference on Ubiquitous Computing, Seattle, USA, October 2003.

[10] A. Schmidt, M. Strohbach, K. van Laerhoven, A. Friday, and H.-W. Gellersen, *Context Acquisition Based on Load Sensing*, in Proceedings of the 4th international conference on Ubiquitous Computing, Springer-Verlag, 2002, pp. 333–350.

[11] R. Want, A. Hopper, V. Falcao, and J. Gibbons, *The Active Badge Location System*, Tech. Rep. 92.1, ORL, 24a Trumpington Street, Cambridge CB2 1QA, 1992.

[12] M. Weiser, *The Computer for the 21st Century*, Scientific American, 265 (1991), pp. 66–75.

[13] M. Weiser and J. S. Brown, *The coming age of calm technolgy*, Beyond calculation: the next fixty years, (1997), pp. 75–85.

3 Tracking - Overview and Mathematics

— **Christoph Krautz**

3.1 Motivation of Tracking

Tracking is the repeated localization of the position and the orientation (pose) of one or several real physical objects. It has many applications, for example in robotics or virtual and augmented reality. In augmented reality, where it is a core technique, it is needed for integrating virtual objects into the real world, for example when blending additional data on a head-mounted display. In the next chapters the most common technologies used for tracking and the mathematics of tracking will be described and explained.

3.2 Overview Over Tracking Technologies

Several technologies for tracking have been invented. One way to classify the different types is by physical medium. In the next chapters the most common technologies will be described using that classification. In addition, the advantages, the problems that can occur and which sources of error can falsify the tracking results will be dealt with.

A typical source of error is distortion due to noise within the working volume. If this distortion is not repeatable, it can not be calibrated out of the system. A further source can be long-term variations that cause readings to change from one day to the next day. Moreover dynamic tracker errors can occur when the tracking system neglects the target's motion with respect to its measurement interval. From this it follows that a low update rate also restricts the range the target can move between two measurements.

3.2.1 Acoustic Tracking

Description

Geometry Acoustic trackers typically use ultrasonic sound waves for determining the position of a target. Starting with one transmitter located at fixed point in the space and one receiver which is located on the target one gets a distance measurement between the transmitter's location and the target. This defines a sphere on whose surface the target is located. Adding a second transmitter or receiver leads to a restriction of the surface to a circle (see figure 3.1). The usage of a third transmitter or receiver again restricts the circle to two points (see figure 3.1), one of which can normally be discarded because it would be to far away, for example. Thus, a determination of the 3D position of the target is achieved.

As one can see for acoustic position-tracking either three transmitters and one receiver or one transmitter and three receivers are needed to compute the position of a object in the three-dimensional space. For additionally estimating the orientation, three transmitter/receiver pairs are required. The orientation is then computed from the different positions of the receivers on the target that were computed as shown above.

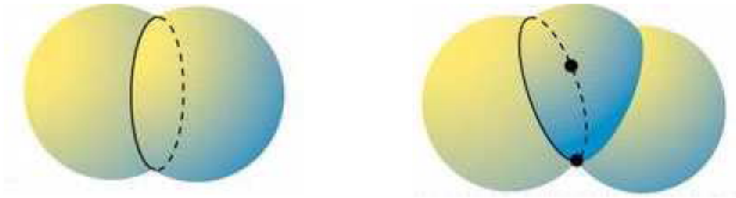

Figure 3.1: Intersection of two spheres (a circle) and three spheres (two points) [1]

Techniques Typically one of the following two techniques, that both use the speed of sound (in air at 0 degree celsius 331 [m/s]) for conversion of time to distance, are employed by acoustic trackers to estimate position and orientation:

- Time of Flight and
- Phase Coherence

Time of Flight

The time of flight method computes the distance d by the multiplication of the speed of sound v by the measured time that a sound wave needs to travel from a transmitter to a receiver (see equation 3.1).

$$d[m] = v[\frac{m}{s}] \cdot t[s] \tag{3.1}$$

Phase Coherence

The phase coherence method computes the distance by measuring the phase difference Φ between the sound wave at the receiver and the transmitter at time t_1, at which the position of the target is known, and at time t_2. The difference $\Phi(t_2) - \Phi(t_1)$ can be converted to the distance the target has moved within the time interval $[t_1, t_2]$. As one can see that method is not able to estimate the absolute position of the target without knowing the previous position. A further aspect to note is that a phase $\Phi + (n \cdot 2\pi)$ leads to the same measurement as a phase Φ, what results in an ambiguity in distance. By assuming phase changes are small within a time interval of measurement this can usually be resolved.

Advantages

Acoustic trackers are small and lightweight because microphones sound-wave transmitters have been highly miniaturized in the last years. They are furthermore not sensitive to influences from the environment except to noise in the ultrasonic range.

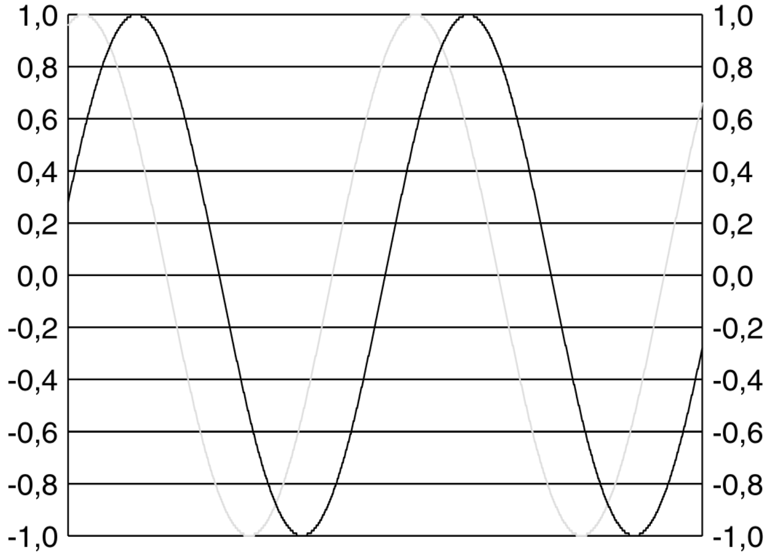

Figure 3.2: Cosine functions with phases of 0 and $\pi/4$ [radians] [1]

Problems And Sources Of Error

The first problem to mention is that the speed of sound is usually not constant. It varies, for example with temperature, pressure and humidity. But non only the variance of the speed of sound is problematic, also the "slowness" and the low update rate resulting from it has to be noted. If the target changes its position and/or orientation during the measurement the result can be inexact due to the inherent delay in waiting for the signal to travel from the transmitter to the receiver. Moreover the above mentioned sensitivity to ultrasonic noise from the CRT sweep frequency or disk drives, for example, causes errors in the tracking results.

Existing Systems

- Logitech 3D Mouse
- InterSense IS-300, IS-600 (http://www.isense.com/)

Speed	**Precision**	**Robustness**	**Range**
+	-	+	-

from [9]

3.2.2 Global Positioning System (GPS)

Description

GPS based tracking systems use time-encoded radio signals from satellites with atomic clocks to calculate positions accurate to a matter of meters. Advanced forms of GPS provide measurements to better than a centimeter. GPS uses time-of-flight measurements as explained above. But the big distance between the satellites and a receiver on the earth causes a problem, the way of synchronization of the clocks of the satellites and the clock of the receiver.

This can only be achieved by a little trick because integrating a atomic clock in every receiver would be to expensive. So, only "normal" clocks are employed. The synchronization of the receiver clock is performed as follows. The position calculated from the signals of three satellites is cross-checked by the signal of a fourth satellite. If that signal does not match the estimated position the receiver knows that its own clock is out of sync. Then it computes a correction factor, so that all four signals lead to one position. This factor is used to correct the clock of the receiver. The disadvantage of that approach is that at least 4 satellites are needed to find out the position of the receiver. (from [4])

Advantages

GPS is a world-wide available system.

Problems And Sources Of Error

GPS based tracking systems provide only a slow update rate. The accuracy depends on how many satellites are on the line of sight of the receiver. However, an maximum accuracy of two meters is still not enough for many tracking systems.

Existing Systems

- Garmin (http://www.garmin.com/)
- Trimble (http://www.trimble.com/)

Speed	**Precision**	**Robustness**	**Range**
–	-	+	++

from [9]

3.2.3 Inertial Tracking

Description

Inertial trackers utilize accelerometers to measure an object's position and gyroscopes to measure the orientation of an object. They rely on Newton's second law of motion $F = ma$ and its rotational equivalent $M = I\alpha$. Ideally, the accelerometer as well as the gyroscope are deployed in orthogonal triples (for 3D position and for 3D orientation) to estimate the 6D pose.

Accelerometers An accelerometer measures the linear acceleration of the object to which it is attached. Position information is obtained from an accelerometer by twice integrating the resulting acceleration, assuming that the initial conditions of the target (position and speed) are known.

A simple accelerometer consists of a spring that a mass (proof-mass) is attached to (see Figure 3.3). Actually it measures the force exerted on the mass (see Figure 3.4) since the acceleration cannot be measured directly. When there is no acceleration imposed upon the accelerometer, the spring is at rest and exhibits zero displacement. If a force is applied to the accelerometer, it will accelerate but inertia causes the suspended mass to lag behind, resulting in a displacement. By the relationship $F = ma$ the measured force is transformed into a measure of acceleration. Finally, by twice integrating the estimated acceleration over the measurement time the searched position information can be obtained. From $a = \frac{d^2x}{dt^2}$ follows that $x = \int\int a\ dt^2$.

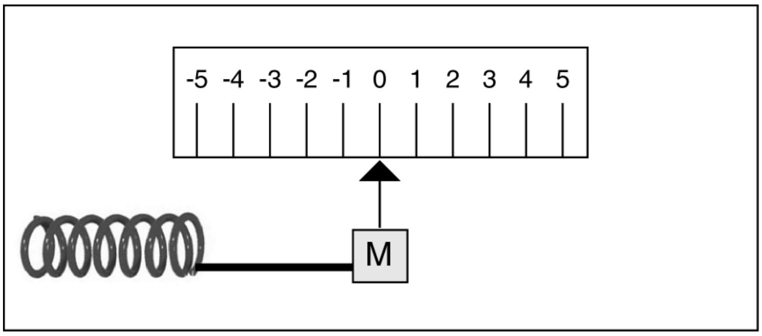

Figure 3.3: Spring at rest with zero displacement [1]

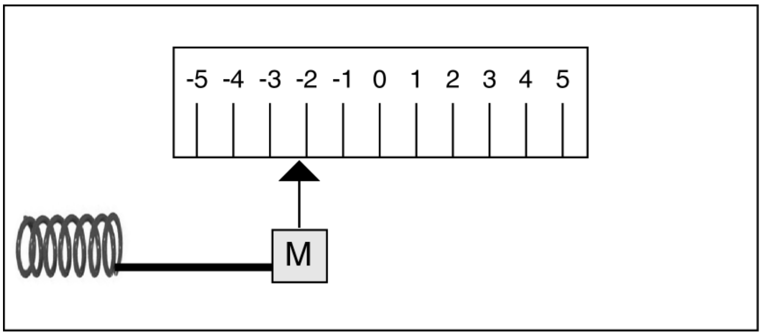

Figure 3.4: Spring under acceleration with displacement [1]

Gyroscopes Gyroscopes base on the principle of conservation of angular momentum, "which states that an object rotated at high angular speed in the absence of external moments, conserves its angular velocity." [7] This principle is employed by two different types of gyroscopes. The first one uses a wheel that is mounted on a frame. The rotation-axis can rotate with three degrees of freedom and external moments due to friction are minimized. So the frame can be frame can be turned "around the wheel without experiencing a change in the direction of its axis" [7]. The orientation of the wheel relative to the frame is computed from the angles measured by rotational encoders attached on the frame.

The second type of gyroscopes uses a phenomenon called "precession" that is caused by the principle of conservation of angular momentum. "If torque is exerted on a spinning mass, its axis of rotation will precess at right angles to both itself and the axis of exerted torque. If the mass spins very fast, it will have a large angular momentum that will strongly resist changes in direction." [1] This phenomenon is illustrated in figures 3.5 and 3.6.

If the spinning mass, e.g. a wheel, is mounted on gimbals the torque can be measured (see figure 3.7). Three rate gyroscope, one for each axis, are required for computing the orientation of an tracked object.

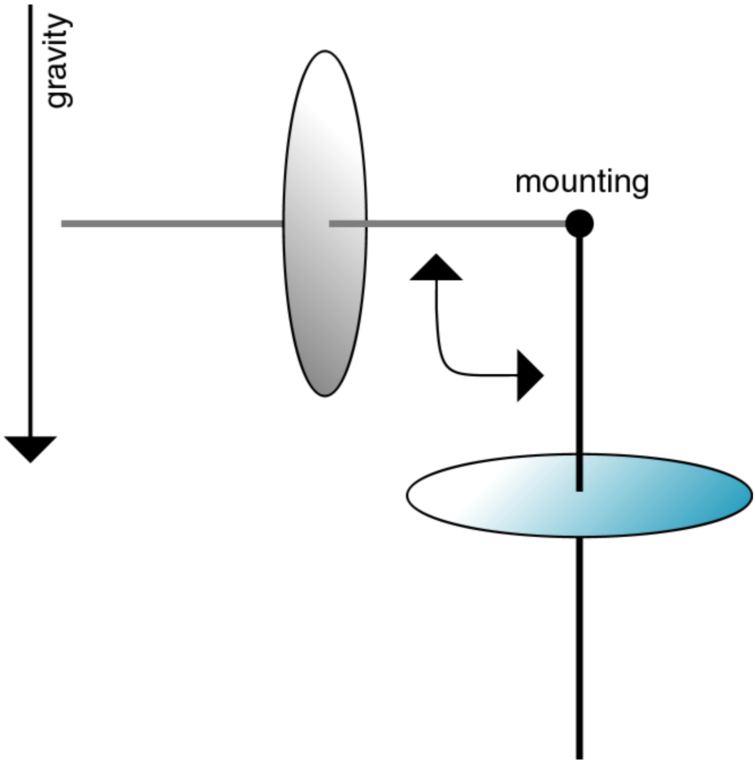

Figure 3.5: Non-Spinning Mass

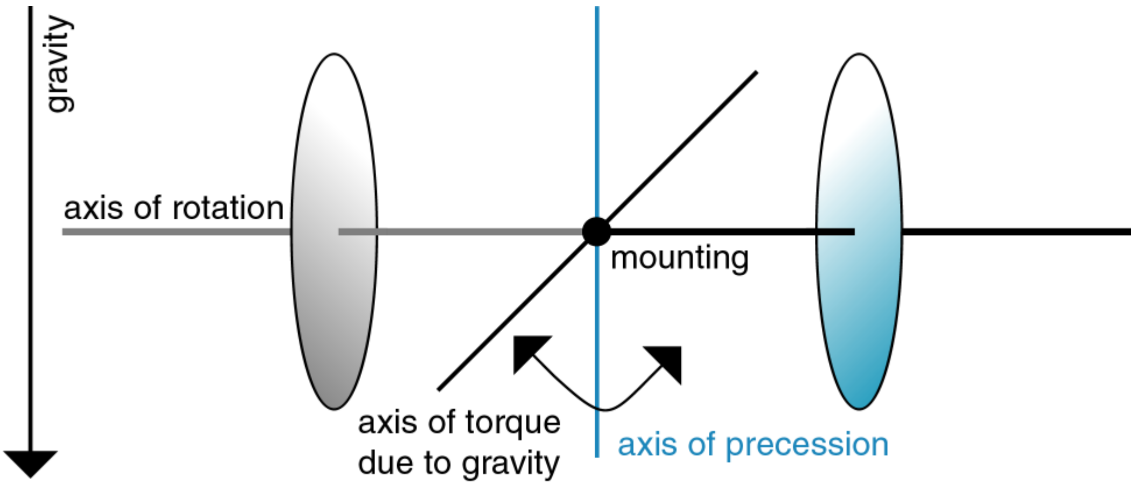

Figure 3.6: Spinning Mass

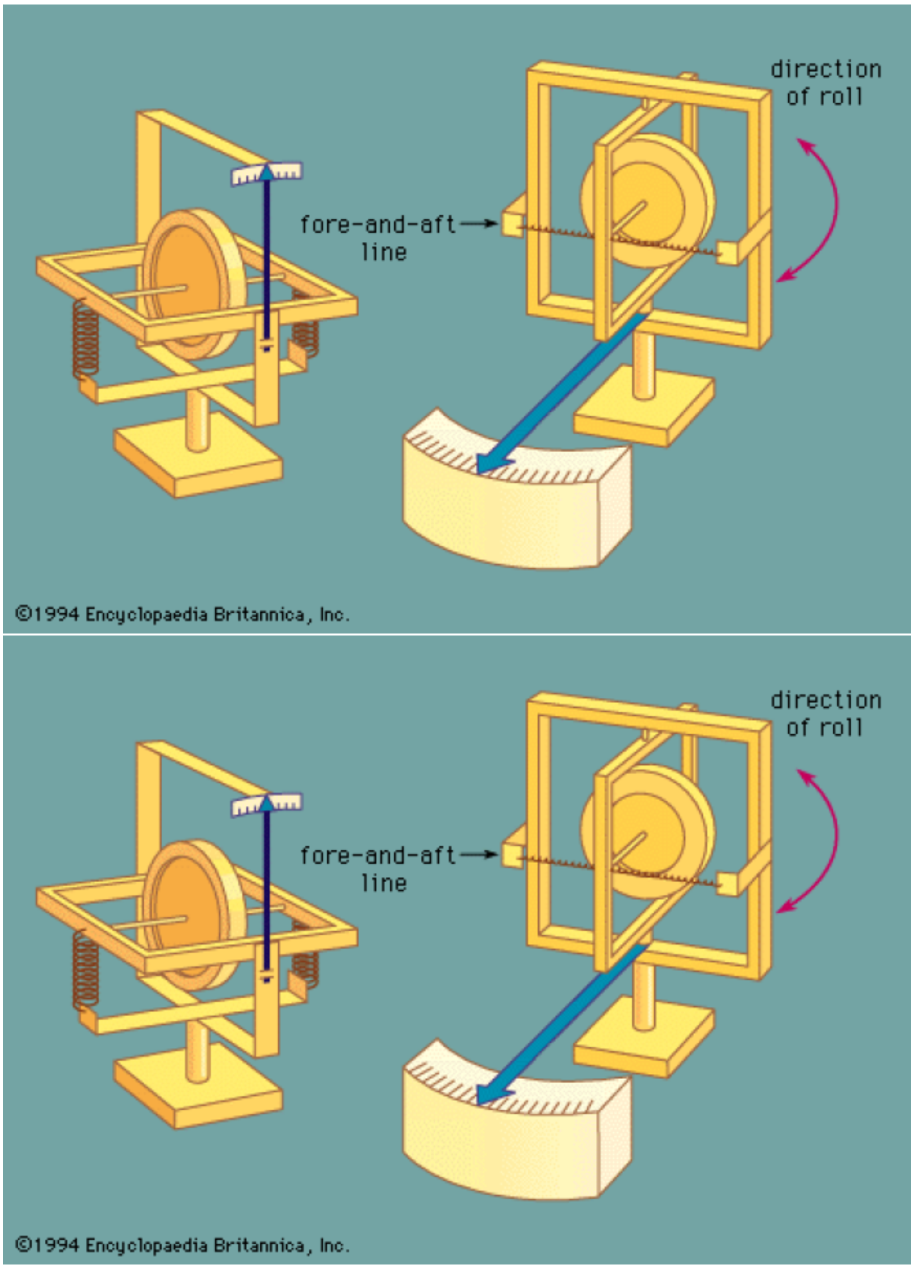

Figure 3.7: Rate Gyroscopes for measuring the rate of turn (left) and the rate of roll (right) [1]

Advantages

Accelerometers usually are lightweight (micro-machined accelerometers are available), and all accelerometers have an absolute reference.

Because no external reference is required to estimate the position and the orientation there are not limits on the working volume and the user is able to move around unencumbered in the environment.

Furthermore inertial trackers provide a very good performance at high frequency and over short time intervals.

Problems And Sources Of Error

Due to the numeric integration errors in position accumulate over time. These errors can be minimized with periodic recalibration.

The main problem of gyroscopes, however, is caused by the remaining friction between the axis of the spinning mass and the bearings. This leads to inaccurate estimations with time.

Existing Systems

- Intersense InertiaCube (http://www.isense.com/)
- Xsense (http://www.xsense.com/)

Speed	**Precision**	**Robustness**	**Range**
++	-	++	++

from [9]

3.2.4 Optical Tracking

Description

Optical tracking systems user either an image-based approach or the determination of sweep-beam angles to compute the position and orientation of a given target.

Two types of targets can employed by optical tracking systems: active and passive targets. Active targets are for example light-emitting diodes (LEDs). Often infrared LEDs are used to prevent a disturbance due to ambient light. Passive patterns consist of reflective materials or high contrast patterns.

Charge Coupled Devices (CCDs) A CCD array usually consists of a 1D or 2D collection of light-sensitive cells (see figure 3.8).

Quad Cells Quad cells consist of four photosensitive cells (see figure 3.9).

Lateral Effect PhotoDiodes (LEPDs) One-dimensional LEPDs are made up of a silicon photosensitive region at which two terminals are attached on either side (see figure 3.10). The electrons produced by an incident light beam flow laterally towards the terminals causing an measurable amount of current that depends on the distance of the centroid of the incident beam from the terminals. Equal current values are measured at each terminal, if the centroid occurs at the center of the photosensitive region. A two-dimensional LEPD is made up of two

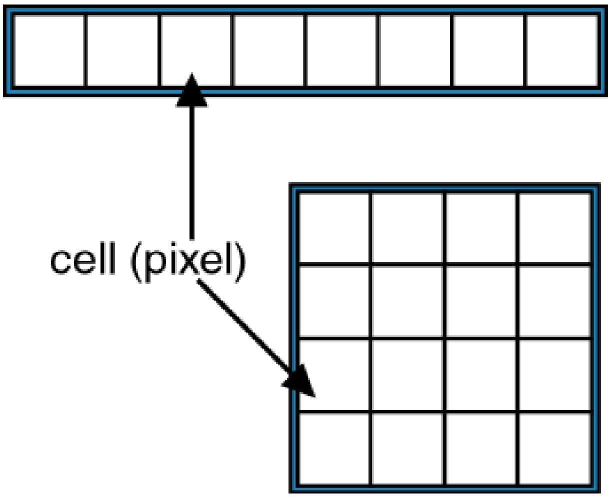

Figure 3.8: 1D and 2D CCD Detector Arrays [1]

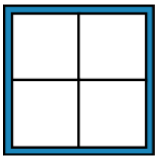

Figure 3.9: A Quad Cell [1]

one-dimensional sensors that a located orthogonal to one another. A LEPD can be employed for the detection of both the intensity and the position of an incident light beam.

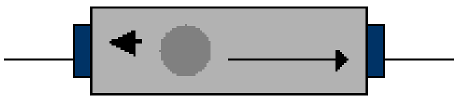

Figure 3.10: A 1D Lateral Effect PhotoDiode

Outside-in versus Inside-out

A sub-classification of optical systems is *outside-in* versus *inside-out*.

Outside-in systems (see figure 3.11) are widely employed by designers of tracking systems. They usually employ sensors that are mounted at a fixed location (position and orientation) in the scene. The tracked objects are marked with passive or active landmarks. The number and/or the shape of the landmarks varies depending on the number of degree of freedom with each object is to be tracked. At least three landmarks are needed to estimate the position and the orientation of an object. For providing redundancy, in order to avoid false estimations due to occlusion, or for improving the position and orientation estimation, additional landmarks can be used.

In contrast to outside-in systems inside-out systems (see figure 3.12) utilize sensors that are attached directly to the object to be tracked for position and/or orientation estimation. These sensors observe the scene which is marked with landmarks. The number and the shape of

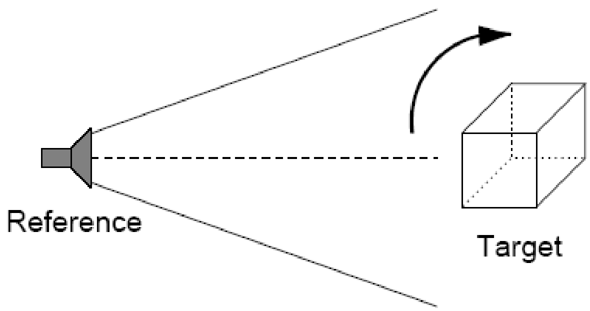

Figure 3.11: A outside-in configuration [6]

the landmarks again depends on the same criteria as described for outside-in systems already. Inside-out systems are rarely used for indoor applications and are required outdoors.

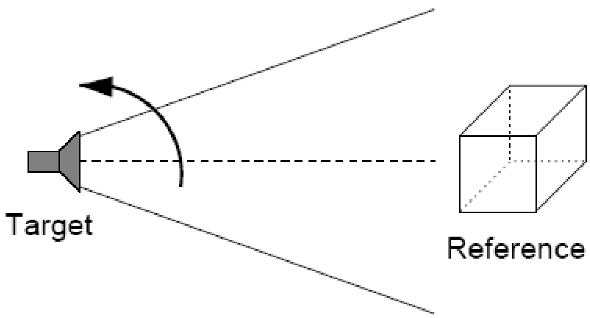

Figure 3.12: A inside-out configuration [6]

System Configuration

Principally the estimation of position and orientation works as follows. A single point on a 2D detector, produced by a incident light beam from a marker, provides a ray in the 3D-space defined by that pixel and the center of the projection of the detector (see figure 3.13). The usage of at least two detectors results in two rays that have to be intersected to compute the position of the target. The orientation can be obtained by comparing the position information of three different markers.

Advantages

Optical tracking systems typically have good update rates because they interact with the environment at the speed of light. Therefore they are well suited to real-time systems.

Problems And Sources Of Error

The accuracy and the resolution of the measurement decreases with the distance of the target from the sensors. The first type of inaccuracy occurs, if the sensors a close together. The lines to the object determined by each sensor then are nearly parallel which results in poor

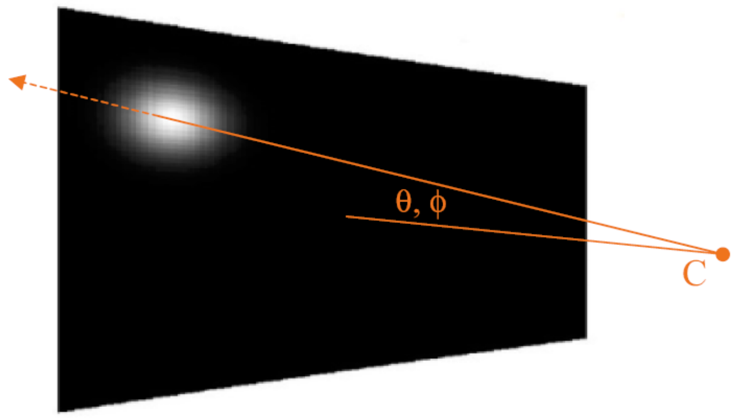

Figure 3.13: Incident light-beam on a 2D-sensor [2]

positional accuracy due to the almost equal line equations. For minimizing this problem the sensors should be far apart and at nearly right angles to one another (see figure 3.14).

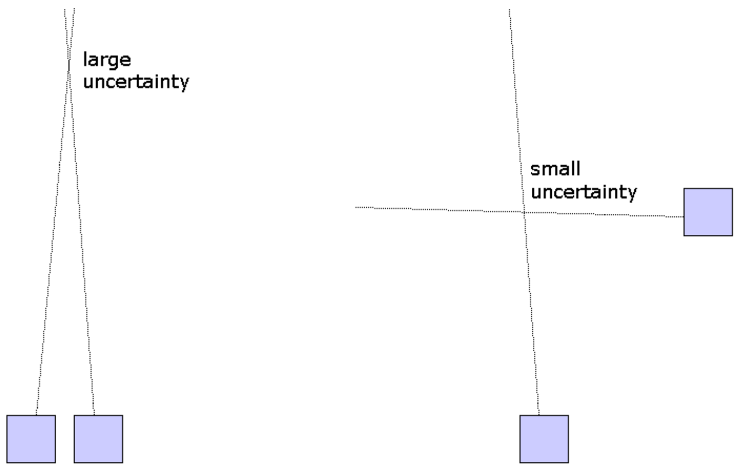

Figure 3.14: Relation between uncertainty and sensor position [1]

The second type occurs, if the markers cannot be resolved spatially in an easy way. This can happen because their relative distance on the sensor image appears smaller as the target gets farther away.

Further sources of error are optical noise and spurious light. Thus, optical tracking systems usually employ infrared light to minimize thereby caused errors.

Finally, errors due to occlusion of landmarks / features have to be mentioned, which can fortunately be controlled by a clever placement of the landmarks / features.

Existing Systems

- ART (http://www.ar-tracking.de/)
- Northern Digital Polaris (http://www.ndigital.com/)

Speed	Precision	Robustness	Range
-	++	-	+

from [9]

3.2.5 Magnetic Tracking

Magnetic trackers measure the position and the orientation by utilizing magnetic fields. Usually low frequency AC fields or pulsed DC fields are generated in each of three orthogonal triaxial coils at the transmitter. Together with the three orthogonal triaxial coils at the receiver position and orientation measurements can be provided.

Magnetic Fields

Generating magnetic fields Magnetic fields are generated by using current carrying coils. The magnetic field produced by a circular coil with a single-turn winding is illustrated in figure 3.15. H_r and H_Θ are the radial and tangential components of the field and described by

$$\begin{aligned} H_r &= \frac{M}{2\pi d^3}\cos(\Theta) \\ H_\Theta &= \frac{M}{4\pi d^3}\cos(\Theta) \end{aligned} \tag{3.2}$$

where M is the magnetic moment of the loop ($M = NIA$), A and N are the area enclosed by the current loop and number of turns of the loop or winding, I is the current carried by the coil, d is the distance to the center of the field and Θ is the off-axis angle.

Detecting magnetic fields A magnetic field that varies with time will induce a voltage in a coil. This voltage can be measured electrically and is proportional to the the angle between the axis of the inducing coil and the axis of the coil the voltage was induced in as well as to the distance between the two coils. Disregarding the distance the maximum voltage is induced, if the transmitting coil is oriented in the same direction as the receiving coil.

System Configuration

A one-dimensional magnetic tracking system is made up of a transmitter coil and a receiver coil. As explained above the induced voltage level provides information about the both distance from the transmitter to the receiver and the axis-alignment between them. Three

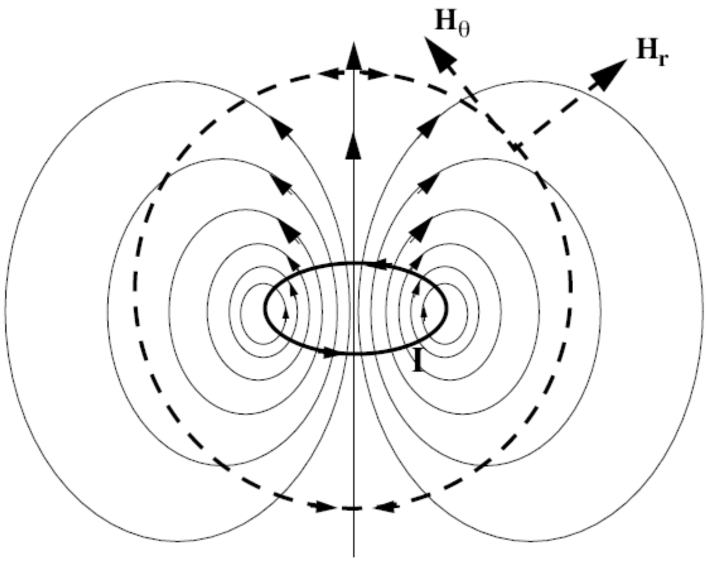

Figure 3.15: Magnetic Dipole [1]

separate orthogonal triaxial coils utilized for both the transmitter and the receiver of a three-dimensional magnetic tracking system. The three source coils are activated serially and the induced signal in each of the three receiving coils is measured. This leads to three values (one of each receiving coil) for each source coil and the resulting nine-element measurement is used to compute the position and the orientation of the receiver relative to the transmitter. The strength of the induced signals can be compared to the known strength of the transmitted signals to find distance. The strength of the induced signals are compared to each other to find orientation.

Advantages

Magnetic trackers are inexpensive and small. In contrast to optical trackers they don't suffer from occlusion problems. They provide a high update rate and a low lag.

Problems And Sources Of Error

The main disadvantage of AC magnetic sensors is the distortion of the magnetic field geometry caused by ferromagnetic and other conducting objects within the sensor space. The magnetic fields of the source coils induce so called eddy currents in these objects and thus small magnetic fields are generated around them. The fields cause distortions in the source fields and that results in inaccurate position and orientation estimates. As this is particularly a problem if AC transmitters are utilized (AC signals are continuously varying) the usage of DC transmitters solves the eddy current interference problem. Unfortunately, distortions due to ferromagnetism are still a problem for DC systems.

Another restriction of magnetic trackers is the small size of the working volume due to the rapid decrease of a magnetic field with respect to the distance of the source coils. Furthermore magnetic tracking systems are sensitive to electromagnetic noise.

Existing Systems

- Ascension: Flock of Birds (http://www.ascension-tech.com/)
- Polhemus FASTRAK (http://www.polhemus.com/)

Speed	Precision	Robustness	Range
+	-	++	–

from [9]

3.2.6 Mechanical Tracking

Description

Mechanical trackers measure the joint angles and lengths between joints. One position in the set-up is known, so all other absolute positions can be derived from the relative joint measurements.

Mechanical trackers are often used in motion capture.

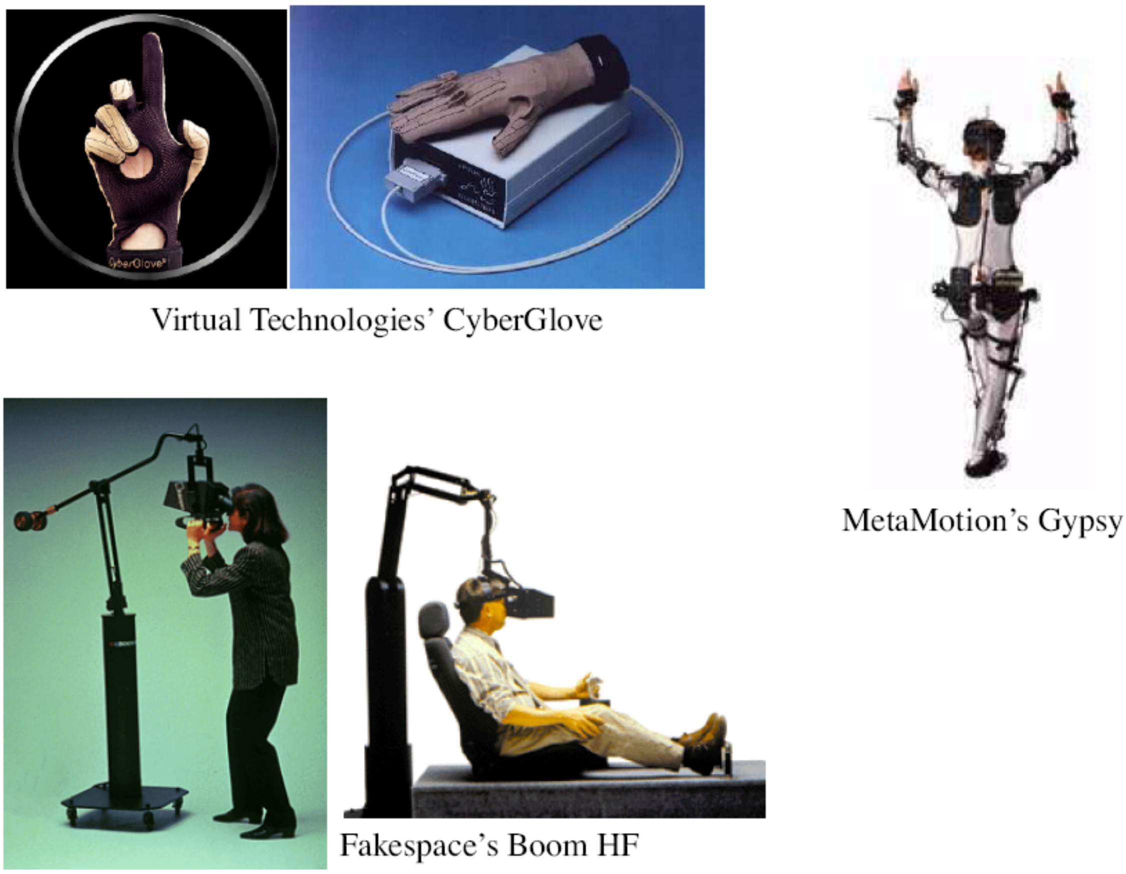

Figure 3.16: Some example mechanical tracking systems [1]

Advantages

Mechanical trackers have a good accuracy and provide a high update rate. The don't suffer from environmental linked errors.

Problems And Sources Of Error

Problematic is the limitation of motion due to the mechanical linkage with the reference. A low encoder resolution can cause an inaccuracy in the measurement.

Existing Systems

- Immersion CyberGlove (http://www.immersion.com/)
- Meta Motion (http://www.metamotion.com/)

Speed	Precision	Robustness	Range
+	++	++	−

from [9]

3.3 Mathematics of Tracking

For a full description of the position and orientation of an object in the three-dimensional space, six degrees of freedom are needed. For specifying the position the Cartesian coordinates x, y, and z with respect to a given reference coordinate system are required. The orientation can be defined by three angles (α, β, γ), also known as pitch (elevation), roll and yaw (azimuth) (for details see section 3.3.5). Equivalently, a pose can be represented by a 4x4 homogeneous transformation matrix:

$$H = \begin{pmatrix} R & T \\ 0 & 1 \end{pmatrix}$$

where R is a 3x3 rotation matrix, defining the same rotation as the three angles (α, β, γ), and T is a translation-vector (x, y, z). Tracking algorithms have to transform measured values in order to estimate the pose of a tracked object with respect to a given reference coordinate system. In the following sections the fundamental operators needed for those transformations will be explained. For didactic reasons first the operators will be defined in the two-dimensional space and afterwards in the three-dimensional space. Problems, that occur when several transformations have to be concatenated, will also be dealt with.

In the following sections the source-point is denoted by P and the transformed point is denoted by P'.

3.3.1 Transformations in the Two-dimensional Space

Translation

A translation is simply a shift of a point P by adding a translation-vector (a, b) to it.

$P = (x, y) \longrightarrow P' = (x + a, y + b)$

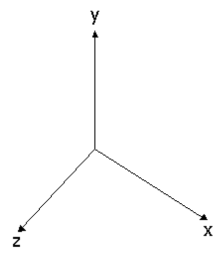

Figure 3.17: Right-handed coordinate system [1]

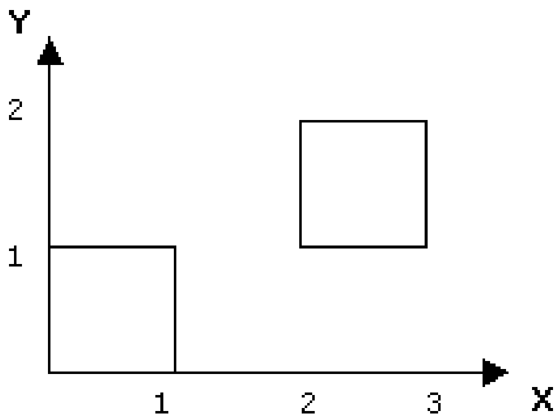

Figure 3.18: Translation in 2D space

Scale

Scaling is simply the multiplication of each component of a point P with an arbitrary value. It can be performed by multiplying each entry of P with an scale-factor.

$P = (x, y) \longrightarrow P' = (s_1 x, s_2 x)$

But also a 3x3-matrix can be used to represent the scale transformation above.

$P' = SP$

$S = \begin{pmatrix} s_1 & 0 \\ 0 & s_2 \end{pmatrix}$

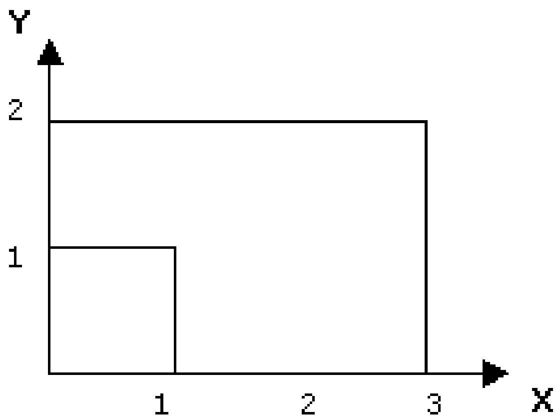

Figure 3.19: Scale in 2D space

Rotation

Rotation means that a point P is rotated about the origin of the reference coordinate system through an angle α.

$P = (x, y) \longrightarrow P' = (x', y')$

where

$x' = x \cos\alpha - y \sin\alpha$

and

$y' = x \sin\alpha + y \cos\alpha$

Written as a matrix the rotation-transformation looks as follows.

$$P' = RP$$

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

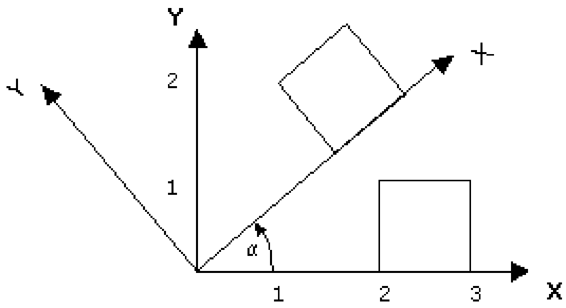

Figure 3.20: Rotation in 2D space

But what happens, if we want to combine several transformations, for example first a rotation and then a translation. We have to apply first the rotation to all of our points and then we have to apply the translation to all of our rotated points. But it would be more efficient, to first concatenate both transformations and then to apply the resulting transformation to all of our points. Here, it has to be mentioned, that scaling and rotation are multiplicative transformations and translation is an additive transformation. In order to be able to concatenate all transformations we have to make them all multiplicative. For this reason *Homogeneous Coordinates* will be introduced in the next section.

Homogeneous Coordinates

A two-dimensional coordinate (x, y) is represented by a homogeneous coordinate $(x, y, 1)$. In general a two-dimensional homogeneous coordinate is defined as (x, y, w) (see figure 3.21).

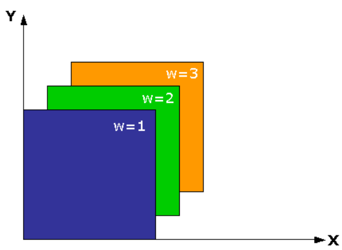

Figure 3.21: 2D homogeneous coordinate space [1]

The above defined operators for translation, scale and rotation now have to be redefined as follows.

Translation:

$$T = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \tag{3.3}$$

so that

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = P' = TP = \begin{pmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x + a \\ y + b \\ 1 \end{pmatrix} \tag{3.4}$$

Scale:

$$S = \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.5}$$

so that

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = P' = SP = \begin{pmatrix} s_1 & 0 & 0 \\ 0 & s_2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} s_1 x \\ s_2 y \\ 1 \end{pmatrix} \tag{3.6}$$

Rotation:

$$R = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{3.7}$$

so that

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = P' = RP = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} x\cos\alpha - y\sin\alpha \\ x\sin\alpha + y\cos\alpha \\ 1 \end{pmatrix} \tag{3.8}$$

3.3.2 Concatenation of Transformations in the Two-dimensional Space

An important operation during tracking is the concatenation of several transformations for calculating the pose of a target-object. In order to apply translation and rotation in one step to a point homogeneous coordinates have been introduced. But still one problem comes up when concatenating transformations. In which order should the different transformations be applied to the origin set of points.
Consider the following example for illustration:

Let $P = (1, 0, 1)$ be the original point in homogeneous coordinates and let

$$T = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

be a translation operation, that shifts the x-coordinate about one and let

$$R = \begin{pmatrix} \cos(\pi/2) & -\sin(\pi/2) & 0 \\ \sin(\pi/2) & \cos(\pi/2) & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

be a rotation operation, which rotates a coordinate system through 90 degrees.

These operations shall be applied to the point P.

If T is applied first and subsequently R, the resulting point P' is

$$\begin{aligned} P' &= RTP \\ P' &= RT(1, 0, 1)^T \\ P' &= R(2, 0, 1)^T \\ P' &= (0, 2, 1)^T. \end{aligned}$$

If R is applied first and subsequently T, the resulting point P' is

$$\begin{aligned} P' &= TRP \\ P' &= TR(1, 0, 1)^T \\ P' &= T(0, 1, 1)^T \\ P' &= (1, 1, 1)^T. \end{aligned}$$

As one can see from this example the order of the transformations matters, wherefore it has to be dealt with carefully.

3.3.3 Transformations in the Three-dimensional Space

Analogous as in the two-dimensional space the representation of a three-dimensional coordinate (x, y, z) as a homogeneous coordinate is $(x, y, z, 1)$.

The transformation operators for translation, scale and rotation are then defined as follows.

Translation

$$T = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

so that

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = P' = TP = \begin{pmatrix} 1 & 0 & 0 & a \\ 0 & 1 & 0 & b \\ 0 & 0 & 1 & c \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} x+a \\ y+b \\ z+c \\ 1 \end{pmatrix} \tag{3.10}$$

Scale

$$S = \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

so that

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = P' = SP = \begin{pmatrix} s_1 & 0 & 0 & 0 \\ 0 & s_2 & 0 & 0 \\ 0 & 0 & s_3 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} s_1x \\ s_2y \\ s_3z \\ 1 \end{pmatrix} \tag{3.12}$$

Rotation

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.13}$$

so that

$$\begin{pmatrix} x' \\ y' \\ z' \\ 1 \end{pmatrix} = P' = RP = \begin{pmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} = \begin{pmatrix} r_{11}x + r_{12}y + r_{13}z \\ r_{21}x + r_{22}y + r_{23}z \\ r_{31}x + r_{32}y + r_{33}z \\ 1 \end{pmatrix} \tag{3.14}$$

The rotation-matrix entries r_{11} to r_{33} have of course to be assigned properly. The different ways of doing that are described among other things in section 3.3.5.

3.3.4 Concatenation of Transformations in the Three-dimensional Space

Three-dimensional homogeneous transformations can be combined by matrix multiplication analogous to the concatenation of transformations in the two-dimensional space. However, the concatenation of several rotations contains some special aspects and the next section will give a detailed insight.

3.3.5 Rotation-Sequences in the Three-dimensional Space

A rotation-sequence consists of several concatenated rotations. There exist several techniques to realize a rotation-sequence, which will be described in the following sections, including their properties and the problems that occur with them.

Rotation Matrices

As described in one of the preceding sections a rotation can be uniquely represented by a rotation matrix. Every such matrix stands for a rotation about a certain axis through a certain angle. The axis and the angle can be easily extracted from the matrix, but this will not be described here. Several rotations about arbitrary axes through arbitrary angles can be be combined by simple matrix multiplications, for which very efficiently implemented methods are available. A disadvantage is that they are inappropriate for filtering and interpolation.

Euler-angles

Leonard Euler (1707-1783) stated and proved that general rotations in 3D can be expressed as three (not more than three) successive rotations about different axes. Usually the coordinate axes x, y and z are used as the rotation axes. The rotation angles are called Euler-angles and a Euler-angle rotation sequence is written in the form $R(\alpha, \beta, \gamma)$. For example, a transformation from reference axes to a new coordinate frame may be expressed as follows.

$$rotation\ \gamma\ about\ z\ axis, R_1 = \begin{pmatrix} \cos\gamma & \sin\gamma & 0 & 0 \\ -\sin\gamma & \cos\gamma & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.15}$$

$$rotation\ \beta\ about\ y\ axis, R_2 = \begin{pmatrix} \cos\beta & 0 & -\sin\beta & 0 \\ 0 & 1 & 0 & 0 \\ \sin\beta & 0 & \cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.16}$$

$$rotation\ \alpha\ about\ x\ axis, R_3 = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha & 0 \\ 0 & -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.17}$$

Finally, the transformation can be expressed as the product of these three separate transformations.

$$\begin{aligned} R &= R_3 R_2 R_1 \\ R &= \begin{pmatrix} \cos\gamma\cos\beta & \sin\gamma\cos\beta & -\sin\beta & 0 \\ \sin\alpha\cos\gamma\sin\beta - \cos\alpha\sin\gamma & \cos\alpha\cos\gamma + \sin\alpha\sin\gamma\sin\beta & \sin\alpha\cos\beta & 0 \\ \cos\alpha\cos\gamma\sin\beta + \sin\alpha\sin\gamma & \cos\alpha\sin\gamma\sin\beta - \sin\alpha\cos\gamma & \cos\alpha\cos\beta & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned} \tag{3.18}$$

It has to be mentioned that in principle the order of the concatenation of the rotations is arbitrary. In the example above the order *zyx* was used. Within an application this order has to be defined explicitly, which is the first problem of the Euler-Angles. They are an ambiguous representation of a rotation in the 3D-space. This ambiguity still remains even if the order of concatenation is defined explicitly because the rotation $R(\pi, 0, 0)$ describes the same rotation as $R(0, \pi, \pi)$, for example.

Another problem arising when using Euler-angles is the so called *Gimbal Lock*. Due to the interaction of the Euler-angles, α and γ reduce to one degree of freedom when setting β to $\pi/2$ because the chosen value for α can be compensated by an accordingly chosen value for γ. When assigning β with $\pi/2$, the rotation matrix R above reduces to the following matrix.

$$\begin{aligned} R &= \begin{pmatrix} 0 & 0 & -1 & 0 \\ \sin\alpha\cos\gamma - \cos\alpha\sin\gamma & \cos\alpha\cos\gamma + \sin\gamma\sin\beta & 0 & 0 \\ \cos\alpha\cos\gamma + \sin\alpha\sin\gamma & \cos\alpha\sin\gamma - \sin\alpha\cos\gamma & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \\ &= \begin{pmatrix} 0 & 0 & -1 & 0 \\ \sin(\alpha-\gamma) & \cos(\alpha-\gamma) & 0 & 0 \\ \cos(\alpha-\gamma) & \sin(\alpha-\gamma) & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{aligned}$$

Unfortunately the three Euler angles don't indicate this to us at all. An attempt to filter or to interpolate the three angles independently would ignore exactly this critical interaction.

Yaw, Pitch and Roll

There exist other names for each Euler-angle which historically have been used in navigation such as on ships and in planes. These names are *yaw*, *pitch* and *roll* and still people use these terms when referring to the orientation of an object. Consider as an example the orientation of a person's head. If the person is sitting upright, looking straight ahead, *yaw* would refer to rotating the head to the left or right around the axis of the neck and the spine, *pitch* would refer to elevating or declining the chin up or down and *roll* would refer to leaning the head towards one shoulder or the other. Thus, placing a right-handed coordinate system at the base of the person's head such that the z axis is up and the person is looking in the direction of the y axis, *yaw*, *pitch* and *roll* would correspond to a rotation about the z, y and y axes respectively.

Quaternions

Introduction Quaternions are an extension of the complex numbers. They are defined as a complex number with one real part r and three imaginary parts x, y and z.

$$q := s + ix + jy + kz = (s, \vec{v}), \vec{v} = (x, y, z), \tag{3.19}$$

in which i, j and k are the three imaginary units.

If s is zero a quaternion represents an ordinary vector; if x, y and z are zero, it represents an ordinary real number. A unit quaternion has the sum of the squares of its four elements equal to 1.

$$s^2 + x^2 + y^2 + z^2 = 1$$

From the geometric point of view a unit quaternion $(s, \vec{v})$ represents a rotation about the axis $\vec{v}$ through the angle $2arccos(s)$.

Here are some other simple rotations:

$$\begin{aligned}
90\ degrees\ about\ Y &= [\frac{1}{\sqrt{2}}, (0, \frac{1}{\sqrt{2}}, 0)] \\
270\ degrees\ about\ Z &= [\frac{-1}{\sqrt{2}}, (0, 0, \frac{1}{\sqrt{2}})]
\end{aligned}$$

Multiplication of quaternions P = QR is defined as

$$[P_r, P_v] = [Q_r R_r - Q_v \cdot R_v, Q_r R_v + R_r Q_v + Q_v \otimes R_v] \tag{3.20}$$

.

The result is a rotation P composed by the rotations Q and R.

A quaternion $[s, (x, y, z)]$ can be easily converted to an equivalent (homogeneous) rotation matrix:

$$R = \begin{pmatrix} 1-2y^2-2z^2 & 2xy+2sz & 2xz-2sy & 0 \\ 2xy-2sz & 1-2x^2-2z^2 & 2yz+2sx & 0 \\ 2xz+2sy & 2yz-2sx & 1-2x^2-2y^2 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.21)$$

Advantages Quaternions have several advantages over the matrix representation and the Euler-angle representation.

- They don't suffer from gimbal lock.
- A user of quaternions does not have to think about certain convention of the order of rotation about explicit axes.
- Interpolations among quaternions are properly and elegant carried out with spherical interpolation on the 4-sphere (see [8]).
- In contrast to the matrix representation the quaternion representation is more compact because due to its four-dimensionality it only contains four degrees of freedom whereas a 3x3 rotation matrix contains nine degrees of freedom.
- The composition of quaternions can be easily performed as described in the preceding section.

3.3.6 Usage of Homogeneous Coordinates in Computer Graphics

Homogeneous coordinates are a very important concept in Computer Graphics because matrix multiplications are easy and, what is most important, very efficient to implement on the graphics hardware.

In OpenGL, which is a low-level API to the graphics hardware, every vertex (one point of a geometric primitive with additional attributes as e.g. its color) is sent through the geometry pipeline during rasterization. OpenGL uses one transformation matrix that every vertex is applied to. So every translation, rotation, scaling, shearing and various projection transformations is performed by multiplying the new transformation by the old transformation. Therefore homogeneous coordinates are needed to provide the possibility of an easy concatenation of all the described transformation types.

Another concept that is in contrast to OpenGL a higher-level API is the so called concept of scenegraphs (e.g. OpenInventor). Every part of the scene to render is stored in a node of a tree, e.g. the vertices, the light-settings and naturally the transformations. In order to render the whole scene the rendering algorithm walks through the whole tree and sends the data of every node of the tree to the graphics hardware using a low-level API as OpenGL. Also the transformations are applied one after another to the scene's transformation.

3.3.7 Usage of Transformations in Tracking

Tracking algorithms use the same transformations as Computer Graphics, but in a different way. In Computer Graphics known transformations are applied to the system as described above. A tracking algorithm has to reconstruct the transformations that describe the pose of a tracked object from the information it gets from the sensors. So the number of unknown parameters of a transformation has to be minimized in order to reduce the time needed for calculation. Using homogeneous rotation matrices as representation for transformations, the algorithm has to determine nine parameters, but only four parameters can be computed from the sensor's data. The missing parameters have to be calculated by using the constraints of a rotation matrix (each row has to be a unit vector and the columns have to be mutually orthogonal). The maintenance of the rotation matrix properties leads to a non-linear algorithm. Here, quaternions provide a better non-redundant representation of a rotation. Only one constraint (the quaternions must have unit-length) has to be maintained (for further details see [3] and [5]).

3.4 Conclusion

Several different tracking technologies were shown in this paper. Everyone has its own advantages and disadvantages. In order to minimize the occurring problems and errors multi-sensor-fusion can be used. On the one hand this technique complicates the mathematics, but on the other hand it leads to much better tracking results than using only one single tracking technology. In the second part the basic mathematics of tracking were described and different problems that can occur were discussed.

Bibliography

[1] G. Bishop, G. Welch, and B. D. Allen, *Tracking: Beyond 15 Minutes of Thought*, in SIGGRAPH 2001 Course Notes, Chapel Hill, 2001.

[2] G. Bishop, G. Welch, and B. D. Allen, *Tracking: Beyond 15 Minutes of Thought*, in SIGGRAPH 2001 Course Slides, Chapel Hill, 2001.

[3] E. B. Dam, M. Koch, and M. Lillholm, *Quaternions, Interpolation and Animation*, tech. rep., 1998.

[4] *All about GPS*. Trimble, May 2004.

[5] J. B. Kuipers, ed., *Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace, and Virtual Reality*, Springer, Vienna, New York, 4th ed., 1997.

[6] M. Ribo, *State of the Art Report on Optical Tracking*, tech. rep., 2001.

[7] J. Rolland, L. Davis, and Y. Baillot, eds., *A Survey of Tracking Technology for Virtual Environments, in Augmented Reality and Wearable Computers, Ch. 3*, Ed. Barfield and Caudell, Mahwah, NJ, 4th ed., 2001.

[8] K. Shoemake, ed., *Animating rotation with quaternion curves*, Computer Graphics, 19(3):245-254, 1985.

[9] *Einfuehrung in die Erweiterte Realitaet - 9. Tracker*. TUM, Prof. Dr. Gudrun Klinker, January 2004.

4 Sensor Fusion Systems: Overview and Mathematics

— **Bjoern Griesbach**

4.1 Introduction

In this paper a short overview of the current state of research on the topic of Multi Sensor Fusion is given. Sensor Fusion is a highly important feature of modern Augmented Reality (AR) Applications. Naturally a high priority in AR is set to the process of tracking objects. There are various different kinds of trackers based on different technologies such as magnetic or optical sensors for instance. In addition even the same tracker from a technological point of view can vary due to its position: Either a tracker is fixed at a specific location (stationary tracker) or it is placed on a head mounted display (HMD) of a person (mobile tracker). Each possibility has its pros and cons in terms of how exact, reliable or fast a specific object can be tracked. Obviously using the data of multiple sensors would most probably lead to a higher quality in tracking objects. The purpose of this paper is to give the reader an idea of how Sensor Fusion works and to show its advantages.

In the first section different systems which make use of Multiple Sensor Fusion will be shown. The second section will cover the mathematical background of Sensor Fusion.

4.2 Existing Multi Sensor Fusion Systems

In this chapter three different Multi Sensor Fusion Systems will be illustrated: Fusion of data from head mounted and fixed sensors, fusion of magnetic and optical trackers and fusion of a gyroscope and an optical tracker. In the last part the software framework Open Tracker will be discussed for the use in Augmented Reality with multiple sensors.

4.2.1 Fusion of data from head mounted and fixed sensors

This chapter refers to the work of William A. Hoff [2]. The problem solved in this paper is a well known issue in advanced indoor AR systems. In an indoor AR lab there are several optical sensors and several objects to be tracked. Some of the sensors are mobile, i.e. head mounted ("inside-out") while others are fixed ("outside-in"). The goal of this system is to track the same object by using the data from several sensors. The solution provided here is thereby also known as a hybrid system combining the "inside-out" and "outside-in" approach. As this paper concentrates on Sensor Fusion, only this part will be focused. Suppose, for example, the system wants to track the pose of a person's head by using the data of a sensor mounted on that head and data of a fixed sensor. Pose is defined as $\vec{x} = (x, y, z, \alpha, \beta, \gamma)$ where the first three attributes specify the 3D position and the latter ones the orientation of the object. Due to no limitation in size and weight, the fixed optical sensor certainly has a significantly higher precision in tracking the 3D position in the room. On the other hand, tracking the orientation definitely can be tracked much better from a sensor mounted directly on the object of interest, i.e. the head. Only a slight rotation of the head results in a large shift of a fixed target in the image taken by the head mounted camera.

Suppose now, that after previous transformations of the available date into the same coordinate system, one gets two measurement vectors $\vec{z}_m$ and $\vec{z}_f$ (from two different sensors M and F) which specify the pose of the same object i.e. the head. Each measurement is correlated with a certain variance, which represents the level of accuracy. These variances are represented in the "vector world" by two matrices, say C_m and C_f. In our example, the covariance matrix of the mobile sensor should obviously give the attributes of the orientation parameters a higher accuracy than the covariance matrix of the fixed sensor does. A higher accuracy is represented by lower variances. Given this data, the optimal combined estimation $\vec{x}$ of the object's pose is the result of the following equation:

$$\vec{x} = \frac{C_f}{C_f + C_m}\vec{z}_m + \frac{C_m}{C_f + C_m}\vec{z}_f$$

with the combined covariance matrix

$$C = \frac{C_f}{C_f + C_m}C_m$$

Using this approach W. Hoff [2] could improve the accuracy of tracking objects by 90% to 50% in an experiment. In figure 4.1 the accomplished result of tracking a third object with a fixed and a head mounted sensor is visualized.

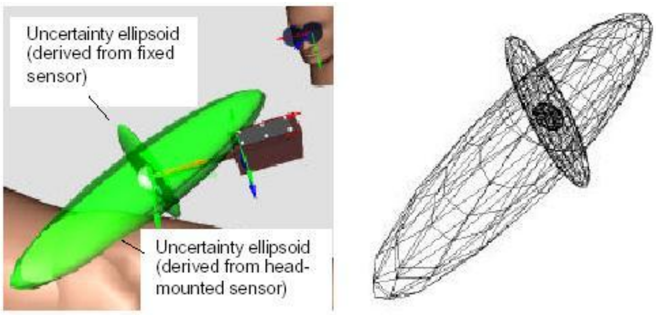

Figure 4.1: This figure [2] points out the fusion of data from the fixed sensor and the head mounted one. The covariances of the measurements are represented by the ellipsoids. Note the small ellipsoid of the combined estimation indicating the much higher accuracy of the tracked object.

4.2.2 Fusion of data from magnetic and optical trackers

This section refers to the papers of [1] and [10]. In the previous chapter the focus was on fusing the data of sensors with a different location. Now we want to concentrate on fusing data of different kinds of sensors in terms of technology. The main idea of fusing the sensor data of magnetic and optical trackers is to make use of the advantages of each technology and to exclude the disadvantages: The pros and cons of a magnetic tracker can be shortly summarized as:

- A magnetic tracker especially in contrast to an optical one is much more reliable and stable
- The computational complexity of getting useful data out of the sensor is comparatively low

An optical tracker on the other hand has the following properties:

- An optical tracker serves with data of a higher accuracy
- An optical tracker is much more time consuming due to high computational complexity of image analysis

Therefore people have researched to find a hybrid solution of combining these two technologies. The approach of [10] is to use an optical landmark tracker supported by a magnetic tracker in order to estimate the head pose in an AR system. The supporting part done by the magnetic tracker is specified as a so called landmark predictor. This device predicts in each frame small areas in the image, taken by the optical sensor, in which landmarks can be found with a high probability. Thereby the comparatively slow image analyzer needs to search for landmarks only in small areas of the image instead of the entire one. The landmark predictor computes the predicted areas with information from previous time steps and the data of the magnetic tracker. Figure 4.2 shows the data flow within this hybrid tracking approach.

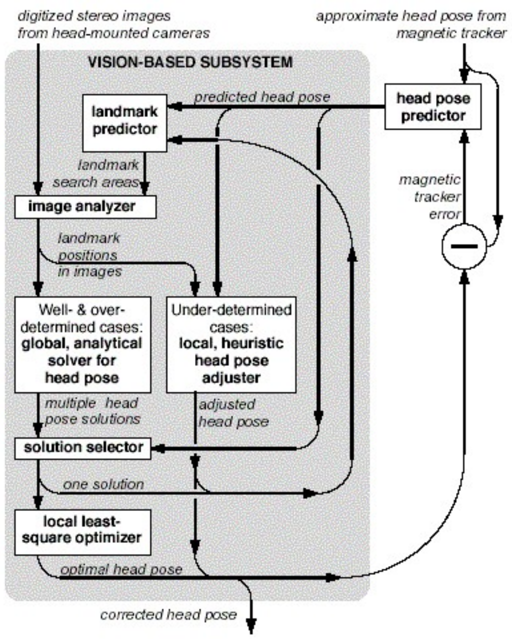

Figure 4.2: This figure[10] points out the fusion of data from the optical sensor and the magnetic one (coming from the top).

4.2.3 Fusion of data from a gyroscope and an optical tracker

The previously demonstrated Multi Sensor Fusion Systems were indoor AR systems. What about Sensor Fusion in the outdoor world? [10] gives an example of Sensor Fusion for outdoor wearable AR systems. This approach uses a hybrid method to estimate head orientation with a gyroscope and an optical sensor. In general a gyroscope serves highly precise data of the head's orientation. The only issue that occurs is that after some time a drift error arises. That is, that tracked objects are still positioned correctly, but with a slight deviance. This happens due to some drift error of the gyroscope. The result of this approach uses a so called vision based drift compensation algorithm to reduce this effect. The concept is similar to the system of fusing magnetic and optical tracking technologies demonstrated in section 4.2.2: The optical sensor supports the gyroscope by giving orientation estimations computed by landmark detection. Thus whenever drift error occurs, the system harks back to the optical tracker. In an experiment the drift error was completely compensated. The cost of the additional computation was small enough to ignore it: When drift compensation was not conducted a frame rate of 22 fps was accomplished, whereas with the new algorithm 18 fps were achieved.

4.2.4 Open Tracker: an open source framework for Sensor Fusion

In this section a standard framework for setting up augmented reality systems is presented. A big issue of current research in augmented reality systems is the fact, that most experiments, regardless which tracking technologies and algorithms they use, are set up in a very idiosyncratic way. This implies that AR systems, once they are set up, usually are not portable. In order to standardize augmented reality systems G. Reitmayr and D. Schmalstieg [6, 7] developed an open source framework (Open Tracker) to ease setting up AR systems. The main purpose of Open Tracker is thereby to combine hardware (trackers, computers, network) and software in a standardized way.

Open Tracker tries to find similarities among all AR systems. Basically every AR application implements some kind of data flow concept. For instance data is generated by some hardware trackers, afterwards forwarded to some filters or fusion algorithms respectively and finally sent via network connections to other hosts, in order to allow distributed computing, for instance. Open Tracker uses XML as a description language for this kind of data flow. The XML configuration file implements a directed graph which represents the data flow (see figure 4.3). There are three different types of nodes in the graph: source, filter and sink nodes.

Source nodes: encapsulate a device driver that directly accesses a specific tracking device such as a magnetic or visual tracker.

Filter nodes: receive data from one or more child nodes, e.g. other filter- or source nodes. Upon receiving data from their child nodes they compute their own state based on the collected data. They are of high interest in this paper, because they can be used for Sensor Fusion. Filter nodes can be split up into different types, e.g.:

- Transformation filters perform geometric transformations of their children's values.
- Noise filters deal with inaccuracies of the data of their child nodes. The Kalman Filter (see section 4.3.1) could be used here.
- Merge filters can be used in particular for Sensor Fusion. They are supposed to merge different parts of the data values of several children. The way sensor data was merged in section 4.2.1 could be described by such a node for instance.

Sink nodes: are similar to source nodes but distribute data rather than collect it. For example, previously collected data could be send to network multicast groups or to a specific user interface, etc.

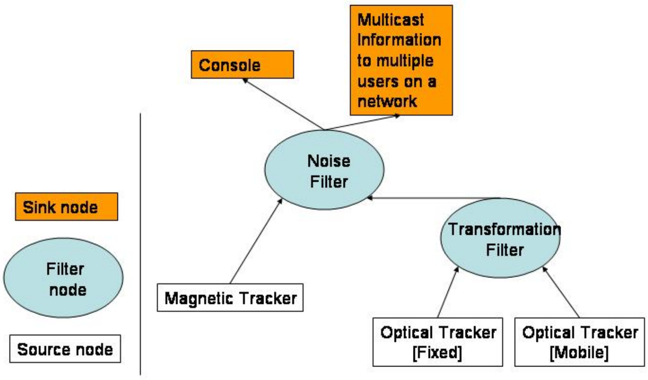

Figure 4.3: An example data flow represented with graph according to the Open Tracker Standard.

Open Tracker is definitely an important step towards standardizing AR applications. Actually all the systems described in the previous sections could be described and set up with this approach. Making use of such standards would most probably decrease the often time consuming process of setting up AR systems and could lead to faster progress in this field of research.

4.3 Mathematics of Sensor Fusion

In this section we will have a short look at two different mathematical tools often used for Sensor Fusion. In the first section the Kalman Filter will be introduced. In the second section a short perspective is given on Particle Filters which can cope with conditions where the Kalman Filter fails (non Gaussian-Noise systems).

4.3.1 Kalman Filter

The Kalman Filter [5, 11, 9, 4] is a well known mathematical tool for various kinds of applications. All those applications have in common, that based on some noisy measurement data, one wants to get an optimal state estimate of a specific dynamic system with discrete time. In our case for example, one wants to get an optimal state estimate (i.e. head pose at time step t) based on some measurement data (i.e. data from magnetic or optical sensors etc.).

In the first section there will be a short description to the Basic Kalman Filter. This algorithm is like the base of multiple extensions in various different research fields. One of those extensions – using the Kalman Filter for Multi Fusion Systems – will be illustrated in the second section.

Short introduction to the Kalman Filter

The Kalman Filter is an recursive algorithm based on Bayes' Rule. It provides a method to compute an optimal estimate of the state of a dynamic system at time step t, based on all measurements taken so far (i.e. in all previous time steps 0..t). In its simplest case, the system measures the position of a stationary object. Measurements thus correspond to a constant state variable subjected to noise. Incremental estimates essentially compute the mean value of a series of measurements. In a dynamic model, the object is mobile. The state variables then reflect the motion parameters of the mobile object (such as velocity or even acceleration). The estimation process of the Kalman Filter is divided into two phases: prediction and correction. In the prediction phase, the estimated state values x_{t-1} of the previous iteration $t-1$ are used to predict the new state x_t^- (Note the superminus, indicating that this variable specifies a prediction, not the actual state). For example, the new position of a mobile object x_t is predicted from its previous position x_{t-1} and altered according to its current motion parameters u_t, such as speed and possibly acceleration. The equation modeling this step is called the **process equation** (or state transition equation). It can be seen as a first order Markov process. In the correction phase, the new measurement z_t at time t is compared with the predicted object position x_t^-. This comparision will correct the prediction and thus compute the actual object position at time t: x_t. If there is a difference between measurement and prediction, the motion model is updated to optimally fit the new motion path. The equation modeling this step is called the **measurement equation**. Figure 4.4 shows the relation of the process and the measurement equation to a certain system. Furthermore the idea of prediction and correction is visualized.

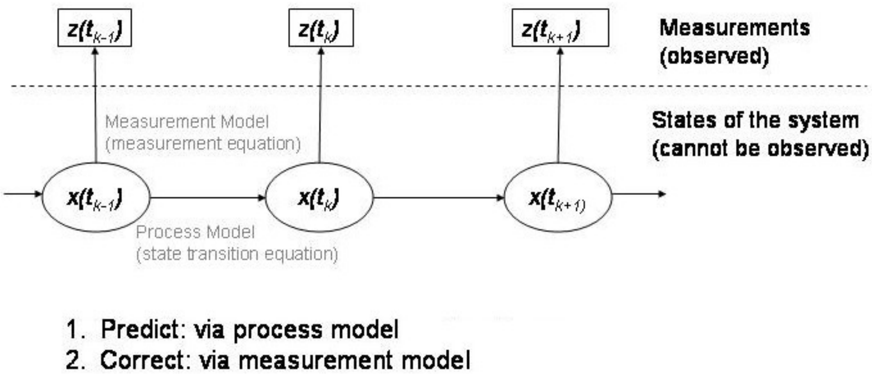

Figure 4.4: The process and measurement model of the Kalman Filter

Modeling the process to be estimated Suppose it is possible to describe the state of a System S in a vector $\vec{x} \in \mathbf{R}^n$ which incorporates all necessary information of S. For example consider a System in Augmented Reality where one wants to observe the position of a certain object. The state or the position respectively of this object could be represented by a vector $\vec{x} = (x, y, z)$.

Of course a System S is not static but dynamic, which means it should be possible to represent S by a recursive process equation. As we observe a system in real live, most probably some noise is correlated with the process. Therefore we speak of a stochastic process.

The Kalman Filter provides a general recursive framework to simulate the stochastic process (**process equation**) of a dynamic system:

$$\vec{x_t} = A\vec{x}_{t-1} + B\vec{u}_t + \vec{w}$$

This recursive equation specifies the dynamic process of an observed system S. In words this equation basically means: The state of the system S at time step t equals the state at time step $t-1$ times a so called state transition matrix A plus some system input $\vec{u}$ plus some noise $\vec{w}$. Matrix $B \in \mathbf{R}^{n \times l}$ relates the input $\vec{u} \in \mathbf{R}^l$ to the dimension of the state vector.

The noise vector $\vec{w}$ represents the normal probability distribution of S with mean 0 and the covariance matrix Q.

$$\mathcal{P}(w) \sim N(0, Q)$$

When all the variables in this framework ($A, B, \vec{u}_t$ and Q) are specified, the Kalman Filter finally knows how this system works. But later on, the Kalman Filter should be able to compute a good estimate of the system's state with the help of some measurement data. Therefore the Kalman Filter needs to have some knowledge about the measurement model as well. A general model is provided by the following **measurement equation**

$$\vec{z_t} = H\vec{x}_t + \vec{v}$$

Again, in words, this equation means that the measurement taken at time t equals the current state $\vec{x}_t$ times a so called measurement relation matrix H plus some measurement noise $\vec{v}$. The noise vector $\vec{v}$ represents the normal probability distribution of the taken measurement of S with mean 0 and the covariance matrix P.

$$\mathcal{P}(v) \sim N(0, R)$$

If one has specified H and R the Kalman Filter finally knows everything that is necessary in order to start computing estimations.

Note that depending on the observed System S some parameters, either in the process or in the measurement model, can be omitted, while others could be extended. For example, if it is possible to represent a system transition only by a matrix multiplication with A, the input vector $\vec{u}$ and its relation matrix B can be skipped by setting $\vec{u}$ to $\vec{0}$. On the other hand the complexity of a system representation can be increased by making the matrices and vector variables time dependent (e.g. $A(t)$, if transition matrix A changes with each time step).

The Basic Kalman Filter algorithm In order to understand the Kalman Filter in a better way, one should keep the following properties in mind:

- The Kalman Filter algorithm is a recursive algorithm. Each estimate of the system's state is based on the previous estimate. Note that an estimation is marked with a hat (e.g. $\hat{x}$).
- The Kalman Filter algorithm can be described best as a Predictor – Corrector–Algorithm. Each time step the filter runs through one prediction – correction cycle.
- In the prediction part the filter predicts a state estimation of the system based on its knowledge about the system (i.e. according to the process model) and the previous state estimate. Note that this predicted estimation is marked with a superscript minus.

- In the correction part the filter updates or corrects respectively the predicted estimate with the new measurement data.

Given the process and measurement model above the Basic Kalman Filter Algorithm would do the following in time step t.

1. Predict
   - Estimate the state ahead: $\hat{x}_t^- = A\hat{x}_{t-1} + Bu_t$
   - Estimate error covariance ahead: $P_t^- = AP_{t-1}A^T + Q$
2. Correct
   - Update state estimate: $\hat{x}_t = \hat{x}_t^- + K[z_t - H\hat{x}_t^-]$
   - Update error covariance: $P_t = [I - KH]P_t^-$
   - with Kalman Gain: $K = \frac{P_t^- H^T}{HP_t^- H^T + R}$

The Kalman Gain is a variable used in both the state estimation update as well as in the error covariance update. It represents the gain which can be taken from the received measurement. The Kalman Gain relies on the knowledge about the variances of all the previously taken measurements and incorporates the variance of the new measurement in each step.

Please note that the equations of the algorithm are related to the particular measurement and process model above. As mentioned earlier those models can vary not only in their particular values of the variables but also in the function itself. Therefore see figure 4.5 which shows the two step Kalman Filter algorithm in a general way. The measurement and process model are stated as general stochastic functions $z = h()$ and $x = f()$. Thus this figure shows the basic idea of the Kalman Filter. All further specifications depend on the system it will be used for.

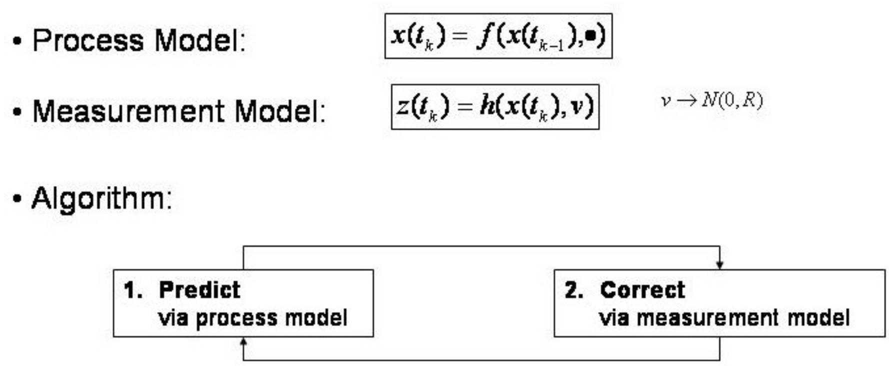

Figure 4.5: The Kalman Filter

For further readings about the Kalman Filter I recommend the slides of Matthias Mühlich [5] and Martin Spengler [9] as well as chapter 1.5 "A Simple Example" in the book of Peter S. Maybeck [4].

The Extended Kalman Filter Algorithm Up to now the Kalman Filter as explained in the previous paragraphs is based on the assumption of a linear process equation of the form $x_t = Ax_{t-1} + Bu_t + w$ (simplified). But what happens if it is not possible to describe a system's process in a linear way? Given this situation, the Basic Kalman Filter was upgraded to the so called Extended Kalman Filter (EKF). This Filter relies on a process model of the form $x_t = f(x_{t-1}, u_t, w_{t-1})$, where f is a non linear function. More information on the EKF can be found in [11] and [9].

How to apply a Kalman Filter In the previous paragraph we have seen how the Kalman Filter works in general. Apparently this is just a very abstract framework. Therefore I want to point out again the things which have to be done before applying a Kalman Filter in "real" life.

1. Find a appropriate representation of the state in a system.
2. Find a process model representing the state transition.
3. Find a measurement model

These theses should point out that there are sometimes many ways of applying the Kalman Filter. This particularly depends on one's process and measurement model. Either way can be more or less efficient and successful, though.

Multi Sensor Fusion with the Kalman Filter

We have seen the Kalman Filter as an algorithm which computes an estimate of the current state of a system represented by a stochastic recursive process. Up to now this estimation was based on previous estimations and *one* measurement $\vec{z}_t$ at each time step.

The new aim is to use the Kalman Filter in order to combine all available measurement data from different sensors to get an optimal estimation. The basic idea is to weight the different mediums most heavily in the circumstances where they each perform best. If the system can be described with a linear model and both the system error and the sensor error can be modelled as white Gaussian Noise, then the Kalman Filter will provide a statistically optimal estimate for the fused data.

As mentioned in the previous section, there are many ways of applying the Kalman Filter to a certain system, though. In the following two paragraphs two possibilities are illustrated, how to use the Kalman Filter for Multi Sensor Fusion. The first proposal can be understood as a synchronized Multi Sensor Fusion Kalman Filter. In each time step the filter will compute a state estimate based on all the measurement data of each sensor. The second proposal can be rather seen as an asynchronous Kalman Filter: Each time a new measurement from whatsoever sensor becomes available the filter computes a new estimation. In each proposal the used process and measurement model will be shown as well as the applied filter algorithm itself. Both proposals will realize a typical task of augmented reality: estimate the pose of certain object at time t, where pose is defined as a vector $\vec{x} = (x, y, z, \alpha, \beta, \gamma)$.

Synchronous Sensor Fusion with the Kalman Filter This proposal uses the Basic Kalman Filter with a simple static process model. The state of the system is described by the vector $\vec{x}$ as defined above.

- Process model: $\vec{x_t} = \vec{x}_{t-1} + \vec{w}$
- Measurement model: $\vec{z_t} = H\vec{x}_t + \vec{v}$

The basic idea of this implementation of the Fusion Kalman Filter is to incorporate the measurement data from all the sensors in $\vec{z}$. Assuming we have two sensors S_m and S_f the measurement vector $\vec{z}$ eventually looks (after previous transformations into the same coordinate system) like this: $\vec{z} = (x_{S_m}, y_{S_m}, z_{S_m}, \alpha_{S_m}, \beta_{S_m}, \gamma_{S_m}, x_{S_f}, y_{S_f}, z_{S_f}, \alpha_{S_f}, \beta_{S_f}, \gamma_{S_f})$ The measurement relation matrix $H \in \mathbf{R}^{6\times 12}$ of course has to be set in a wise way. Suppose S_m is a mobile head mounted sensor and S_f a fixed sensor. Given our knowledge from section 4.2.1 most probably H should weight the position data x, y, z of S_f higher than the data of S_m and vice versa for the orientation data α, β, γ.

The algorithm itself is identical to the Basic Kalman Filter algorithm with the appropriate parameters (i.e. A and $\vec{u}$ are both 0).

Asynchronous Sensor Fusion with the Kalman Filter This approach is based on the work of G. Welch and G. Bishop [11]. As this is a much more sophisticated approach, explaining all the details would go beyond the scope of this document. Thus the following description simply gives a rough idea of this concept but does not go into detail.

The basic idea of this approach starts with the use of an **extended state description vector**:

$$\vec{x} = (x, y, z, \alpha, \beta, \gamma, \dot{x}, \dot{y}, \dot{z}, \dot{\alpha}, \dot{\beta}, \dot{\gamma})$$

This vector also incorporates the current movement of the system, by adding the derivations of the pose attributes. The appropriate **process model** is of the form:

$$\vec{x_t} = A(\delta t)\vec{x}_{t-\delta t} + \vec{w}_{\delta t}$$

with $\vec{w}_{\delta t} \sim N(0, Q_{\delta t})$. The state transition matrix A predicts the future state $\vec{x}$ by implementing the following relationships (e.g.):

- $y_t = y_{t-\delta t} + \dot{y}_{t-\delta t}\delta t$
- $\dot{y}_t = \dot{y}_{t-\delta t}$

Another atypical extension of the Kalman Filter is the use of **multiple measurement models** for each sensor i:

$$\vec{z}_{i,t} = H_i\vec{x}_t + \vec{v}_{i,t}$$

with $\vec{v}_{i,t} \sim N(0, R_{i,t})$. Finally the algorithm itself looks just slightly different than the original Basic Kalman Filter algorithm. Whenever a new measurement from some sensor becomes available the Kalman Filter runs through the following prediction–correction cycle:

1. Predict
   - Estimate the state ahead: $\hat{x}_t^- = A(\delta t)\hat{x}_{t-\delta t}$
   - Estimate error covariance ahead: $P_t^- = A(\delta t)P_{t-\delta t}A^T(\delta t) + Q(\delta t)$
2. Correct with measurement i
   - Update state estimate: $\hat{x}_t = \hat{x}_t^- + K[z_{i,t} - H_i\hat{x}_t^-]$

- Update error covariance: $P_t = [I - KH_i]P_t^-$
- with Kalman Gain: $K = \frac{P_t^- H_i^T}{H_i P_t^- H_i^T + R_{i,t}}$

Due to problems in computational complexity of this algorithm this filter is further extended as can be found in [11].

4.3.2 Particle Filters

As shown in the previous section the Kalman Filter can cope with dynamic systems with Gaussian Noise. If noise in a certain system is not Gaussian however, one can use another interesting tool: the Basic Particle Filter, also known as Bootstrap Filter. The main duty of this algorithm is to handle non linear processes with non Gaussian Noise. Like the Kalman Filter the Particle Filter is also based on the Bayes's Rule. But where the Kalman Filter can make use of Gaussian Noise in order to simplify the computation of the Bayes's Rule, the Particle Filter uses a different concept. A good introduction to the Basic Particle Filter can be found in [5] and [3]. Furthermore [8] is an interesting article, where a Distributed Particle Filter for Decentralized Sensor Fusion is introduced. As Particle Filters require an even higher computational power than the already complex Kalman Filter, this paper provides a scalable possibility for Sensor Fusion with distributed Particle Filters.

4.4 Conclusion

This paper provided a short overview of Sensor Fusion in Augmented Reality applications and the mathematics used in this field. Sensor Fusion, to some extent, is of increasing interest due to needs of accuracy and reliability in tracking objects. At the same time Sensor Fusion relies on complex mathematical tools and algorithms which often surpass the limit of today's computational possibilities. Finding methods to cope with this problem is the aim of current and future research in this field.

Bibliography

[1] T. Auer and A. Pinz, *The integration of optical and magnetic tracking for multi-user augmented reality*, Graz, Austria, 1999.

[2] W. Hoff, *Fusion of Data from Head-Mounted and Fixed Sensors*, in First International Workshop on Augmented Reality, San Francisco, Golden, Colorado, U.S., 1999.

[3] C. Hue, J.-P. Le Cadre, and P. Perez, *A Particle Filter to Track Multiple Objects*, Rennes Cedex, France, 2000.

[4] P. Maybeck, ed., *Stochastic models, estimation and control*, Academic Press New York San Francisco London, Ohio, U.S., 1st ed., 1979.

[5] M. Mühlich, *Particle Filters: an overview*, in Filter Workshop Bucuresti, Frankfurt, Germany, 2003.

[6] G. Reitmayr and D. Schmalstieg, *An Open Software Architecture for Virtual Reality Interaction*, Vienna, Austria, 2001.

[7] G. Reitmayr and D. Schmalstieg, *Open Tracker - An Open Software Architecture for Reconfigurable Tracking based on XML*, Vienna, Austria, 2002.

[8] M. Rosencrantz, G. Gordon, and S. Thrun, *Decentralized Sensor Fusion with Distributed Particle Filters*, Pittsburgh, U.S., 2000.

[9] M. Spengler, *Sensor Fusion using Kalman Filter*, in Data Fusion Seminar, Zuerich, Switzerland, 2001.

[10] A. State and G. Hirota, *Superior Augmented Reality registration by Integrating Landmark Tracking an Magnetic Tracking*, North Carolina, U.S., 2000.

[11] G. Welch and G. Bishop, *An Introduction to the Kalman Filter*, North Carolina, U.S., 1999.

5 The mathematics of (Auto-)Calibrating AR Systems

— **Benjamin Fingerle, Christian Wachinger**

5.1 Calibration in Augmented Reality Environments

5.1.1 Introduction

"Calibration is the process of instantiating parameter values for mathematical models which map the physical environment to internal representations, so that the computers internal model matches the physical world."
Mihran Tuceryan

5.1.2 Requirements for Calibration in AR Environments

In an Augmented Reality environment reality is modeled in a virtual world by arranging digital counter parts of various real objects positioned and oriented based on data gathered by tracking technology. Since this digital representation of the real world - the virtual world - is then enriched with context sensitive information and this augmented virtual reality somehow projected back to the real world, any inaccuracy in estimating position or orientation of real world objects leads to a loss of realism in the augmented scene. A second source of impreciseness forms the projection of the augmented virtual world back to the real world.

For realism of the augmented scene determining the usability the preciseness of underlying tracking technologies forms the major success factor for AR applications. To reach highest accuracy an adequate calibration method for each tracker is needed.

Besides accuracy AR requires calibration methods to be as autonomous as possible to reach a convenient calibration process and to turn the number of user related errors down. Since AR often requires realtime calibration methods have to be computationally efficient. Additionally calibration methods should be designed to be as versatile as possible to enable them for reuse in different AR setups. [4]

5.2 Motivating AR Scenario

In our scenario a mobile user - we call him Gerhard - is wearing a Head Mounted Optical See Through Display (OST-HMD). When standing in front of an - from our point of view - apparently empty table this table seems not to be empty for Gerhard. Through the OST he sees the table with several virtual 3D objects placed on top of it. Additionally Gerhard is enabled to move these virtual objects with his hands wearing special gloves for this purpose.

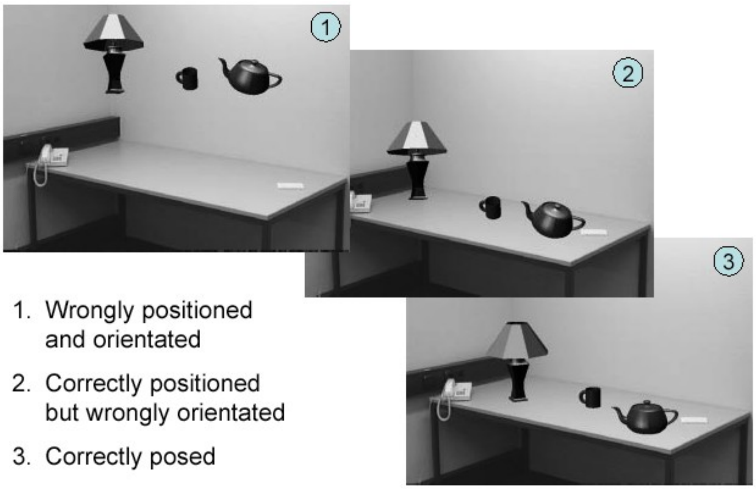

Figure 5.1: Virtual objects placed on table [5]

5.2.1 Objects to calibrate

In the specified scenario following parameter have to be estimated:

- Pose of the table relatively to the room
- Pose of Gerhard's head relatively to the room
- Pose of Gerhard's hands relatively to the room
- Parameters of Gerhard's OST-HMD

These estimations are done by using following technologies described in the subsequent chapters:

- Table: 3DOF-magnetic pointer based object calibration
- Gerhard's head pose: 6DOF-magnetic tracking - the magnetic marker rigidly fixed at Gerhard's HMD
- OST-HMD: SPAAM method for estimating parameters
- Gerhard's hands' pose: Stereovision based tracking

For using magnetic tracking a magnetic tracker transmitter has to be calibrated previously.

5.3 Pointer Calibration

In this section we want to describe the calibration of a pointer device like it is used within the GRASP system [5] and in our scenario. It enables us to easily calibrate other objects by just pointing at them. The pointer is a stick where a tracker receiver is mounted to get the pose. This is done with the help of a 6DOF tracker transmitter. The calculated parameters stay valid until the position of the tracker receiver on the stick is changed. The schematic layout is shown in Figure 5.2.

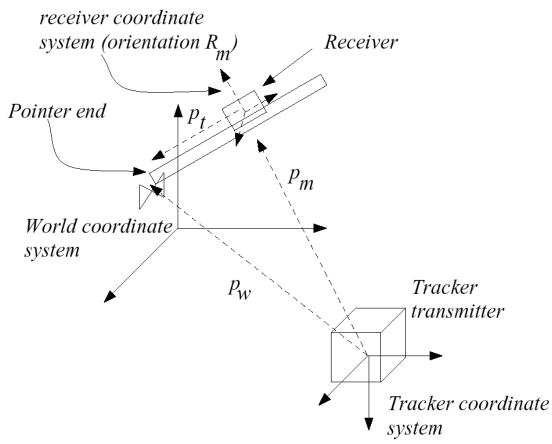

Figure 5.2: The pointer device [5]

During the calibration procedure the 3D vectors p_m, p_w, p_t and the 3 x 3 matrix R_m are interesting. p_m is the measured position of the tracker receiver, p_w is the position vector representing the tip of the pointer in tracker coordinates, p_t is the receiver attachment offset and R_m is the rotation matrix defining the orientation of the receiver as measured by the tracker. The formula

$$p_w = p_m + R_m p_t \tag{5.1}$$

shows the correlation of the different parameters like it is also pictured in Figure 5.2.

To calculate the unknown parameters p_w and p_t we have to pick a 3D point n times (n is a number between 3 and 6), each time with an other orientation of the stick. (see Figure 5.3)

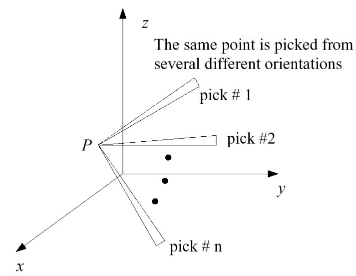

Figure 5.3: Pointer calibration procedure [5]

So we receive several equations with constant values p_w, p_t and measured values p_m, R_m. This leads to the following equation which has to be solved:

$$\begin{pmatrix} I & -R_{m1} \\ I & -R_{m2} \\ \vdots & \vdots \\ I & -R_{mn} \end{pmatrix} \begin{pmatrix} p_w \\ p_t \end{pmatrix} = \begin{pmatrix} p_{m1} \\ p_{m2} \\ \vdots \\ p_{mn} \end{pmatrix} \tag{5.2}$$

where I is a 3 x 3 identity matrix. The linear equation system (5.2) is over-determined as we have six unknowns (three for p_t and three for p_w) and 3n rows. Thus the least squares method is applied to solve it.

5.4 Pointer based Object Calibration

The problem of object calibration is to get the position and orientation of a real-world object. Therefore certain landmark points of the object have to be identified. Afterwards the relationships between the coordinates of the landmark points represented in the object coordinate system and the world coordinate system have to be calculated. As the landmark points should be easy to identify they should correspond to natural features of the objects. There exist two ways to localize the landmark points. One is a image-based approach which is already explained in the tracking section. The other approach, based on the previously calibrated pointer, is presented here. The calibration procedure assumes that a virtual model of the real-world object is already available. This geometric model may be rendered manually or it could come from a CAD system.

The following formula states the rigid transformation from coordinates in world coordinate system p^w to object's local coordinate system p^l:

$$p^w = \mathbf{R}p^l + \mathbf{T} \tag{5.3}$$

where $\mathbf{R}$ is a 3 x 3 rotation matrix and $\mathbf{T}$ is a 3D translation vector.

The aim of the calibration procedure is to estimate $\mathbf{R}$ and $\mathbf{T}$. This done by picking n landmarks of the object which has to be calibrated with the pointer device. Leading to

$$p_i^w = \mathbf{R}p_i^l + \mathbf{T} \tag{5.4}$$

where i is in the range from 1 to n. In this linear system we have 12 unknowns. As each measurement delivers three rows, four of them are necessary to get a unique solution. But because there are always calibration errors approximately 10 measurements should be taken and the linear system could be solved using the least squares method.

The disadvantage of this approach is that it cannot be ensured that the resulting matrix $\mathbf{R}$ is rotation matrix. By solving the nonlinear optimization problem (5.5) we force that $\mathbf{R}$ is a rotation matrix.

$$||p_i^w - \mathbf{R}p_i^l - \mathbf{T}||^2 + \alpha||\mathbf{R}^T\mathbf{R} - \mathbf{I}||^2 \tag{5.5}$$

5.5 Stereovision Camera Calibration

In our scenario Gerhard's hands are supposed to be tracked by a static wall-mounted stereovision camera. Tracking means estimating the three dimensional coordinates and orientation of an object in space. A stereovision camera offers two images of the scene from slightly different perspectives. In these two images known landmarks are searched and rays on which these landmarks are aligned are obtained. Finding the intersection of the two rays for each landmark obtained from the two images yields the tree dimensional points of the landmarks. This process is called triangulation and is described in detail elsewhere. The orientation can then be calculated by knowledge of the spatial relation between the landmarks provided that sufficiently many landmarks are found.

5.5.1 Extrinsic and Intrinsic Camera Parameters

In preparation to apply triangulation to camera images several camera specific parameter have to be known. Parameters taken directly from the camera specification proved to be too imprecise for usage in AR. [4] Therefore these parameters have to be estimated via calibration. Assumed all intrinsic parameters are known triangulation could be applied and an objects pose (position and orientation) obtained.

The calculated pose is relative to the camera's coordinate system (CCS). But since Gerhards hands should interact with the virtual objects on the table we need their poses relative to the world coordinate system (WCS). Thus we also have to estimate the camera's pose relatively to the WCS which is also done via calibration.

5.5.2 The Camera Model

Basic Pinhole Camera

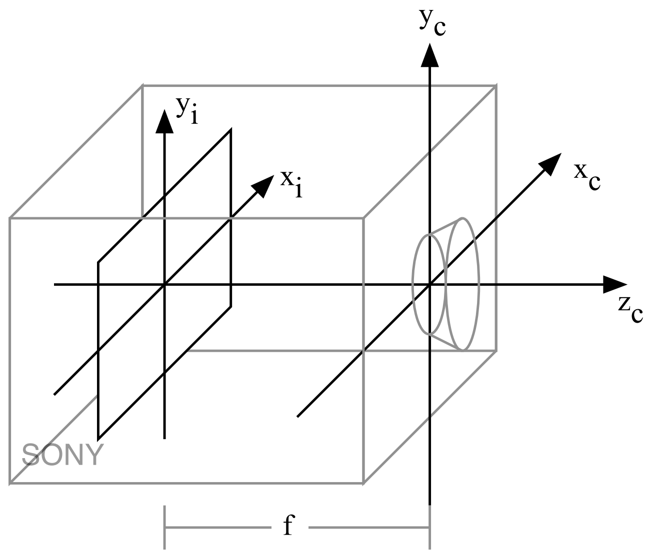

Figure 5.4: Basic Camera Model - Pinhole Camera

As described in Figure 5.4 the basic camera model is a perfect pinhole camera. This camera model requires the focal length f to be estimated as only intrinsic parameter.

Camera's Pose in WCS

As described beforehand to get the objects pose relatively to the WCS the spatial relationship between WCS and CCS has to be estimated. This relationship is shown in Figure 5.5.

This spatial relationship can be mathematically described as combination 3 x 3 transformation matrix R and three dimensional translation vector T. R forms a rotation matrix and thus can be described by the Euler angles yaw θ, pitch ϕ and tilt ψ leading to a matrix of three degrees of freedom.

$$R = \begin{pmatrix} \cos\psi\cos\theta & \sin\psi\cos\theta & -\sin\theta \\ -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi & \cos\psi\cos\phi + \sin\psi\sin\theta\sin\phi & \cos\theta\sin\phi \\ \sin\psi\sin\phi + \cos\psi\sin\theta\cos\phi & -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi & \cos\theta\cos\phi \end{pmatrix} \tag{5.6}$$

Thus we have two extrinsic Parameters of all in all 6 degrees of freedom.

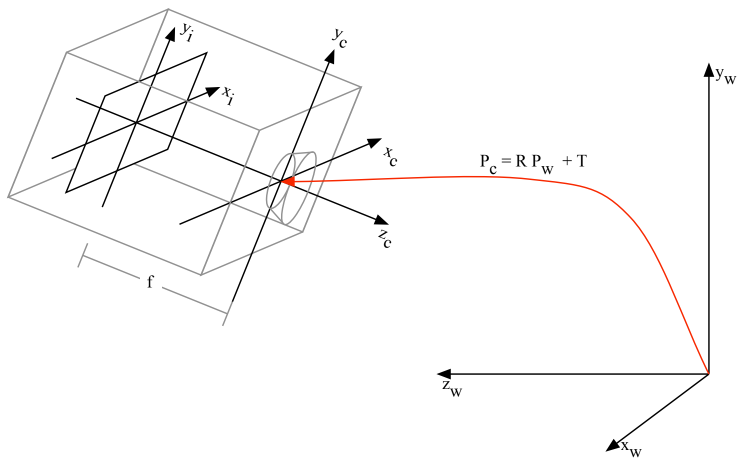

Figure 5.5: Spatial relationship between WCS and CCS

The Relation Between 3D WCS Points and their 2D Image Counterparts

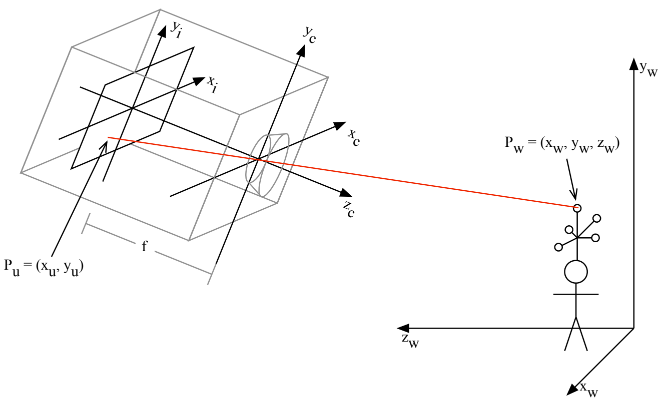

Figure 5.6: Relation between 3D and 2D points

Calculating the image coordinates of a 3D point in WCS follows standard pinhole camera geometric as shown in Figure 5.6.

Given a point $p_w = (x_w, y_w, z_w)$ in WCS we can calculate this point's representation p_c in CCS with calculating

$$p_c = Rp_w + T \tag{5.7}$$

Applying perspective projection with pinhole camera geometry yields

$$x_u = f\frac{x_c}{z_c} \tag{5.8}$$

$$y_u = f\frac{y_c}{z_c} \tag{5.9}$$

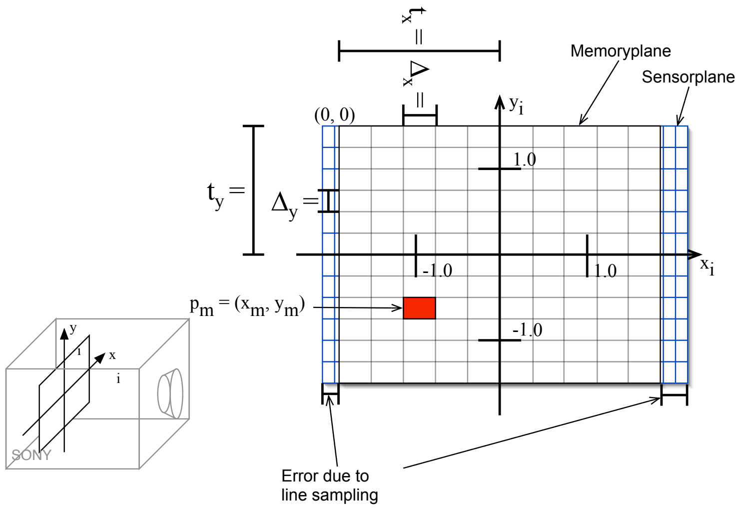

Figure 5.7: CCD related Intrinsic Parameters

CCD Related Intrinsic Parameters

The common usage of CCD cameras for optical tracking introduces two additional intrinsic parameters which both have to be estimated via calibration.

Firstly common image memories' coordinate origin is not in the center of the memory chip but at one the corners usually the upper left. Thus the memory pixels are shifted in x- and y-direction relatively to the CCS by a two dimensional translation vector $S = (t_x, t_y)$.

Secondly a horizontal scale factor s_x has to be estimated which is introduced by slight timing impreciseness during the process of line sampling. In difference to the y-axis pixels on the x-axis are usually resampled instead of directly copied from the sensor-plane to the memory plane, i.e. one line of pixels is taken and then sampled to fit the memory plane which usually has a different number of pixels than the sensor plane. This is done mainly for optic-psychological reasons but introduces impreciseness since the timings in sampling are usually not totally exact. To cope with this impreciseness an additional intrinsic parameter, the horizontal scale factor s_x has to be introduced which scales the x-coordinates to their real position in CCS as shown in Figure 5.7.

This leads to a mathematical relation between a 2D image point $p_u = (x_u, y_u)$ and its 2D counterpart in memory $p_m = (x_m, y_m)$ described by

$$x_m = s_x \frac{x_u}{\triangle_x} \frac{\sharp xMem}{\sharp xCCD} + t_x \tag{5.10}$$

$$y_m = \frac{y_u}{\triangle_y} + t_y \tag{5.11}$$

Intrinsic Parameters Describing Radial Lens Distortion

Real life cameras used for optical tracking cannot be modeled sufficiently by a perfect pinhole camera since lens distortion proved to be influencing the results too much for feasible usage in AR.[4] Thus lens distortion has to be included in our camera model as shown in 5.8.

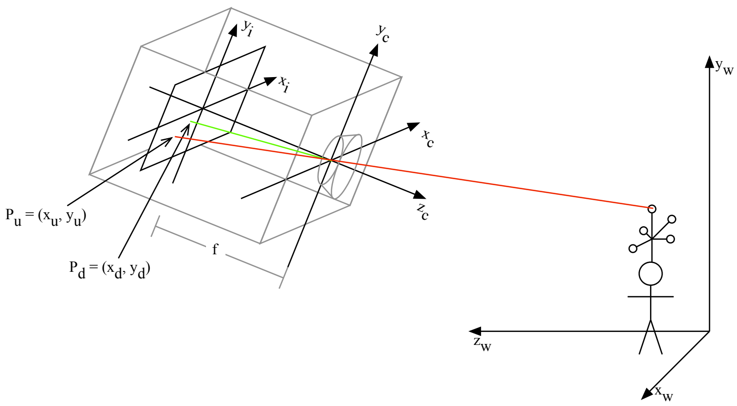

Figure 5.8: Radial Lens Distortion

Further experiments have shown that only radial lens distortion needs to be considered, tangential lens distortion not only can be neglected due to minimal effect but also should be let out to keep the number of possible numerical errors down.[4]

Radial lens distortion can be mathematically modeled with infinite series while using just the first two coefficients proved to be sufficient for optical tracking. Let $p_u = (x_u, y_u)$ be the undistorted 2D image of a 3D point p_c in CCS. Then the distorted 2D image point is modeled by

$$x_u = x_d(1 + k_1 r^2 + k_2 r^4) \tag{5.12}$$

$$y_u = y_d(1 + k_1 r^2 + k_2 r^4) \tag{5.13}$$

with

$$r = \sqrt{x_d^2 + y_d^2} \tag{5.14}$$

Thus considering radial lens distortion introduces two more intrinsic parameters, namely the coefficients k_1 and k_2.

Concluding: From WCS to Memory

- WCS $\rightsquigarrow$ CCS

  $p_w = (x_w, y_w, z_w)$ point in WCS $\rightsquigarrow$ $p_c = (x_c, y_c, z_c)$ point in CCS

  with $R = \begin{pmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{pmatrix}$

$$x_c = r_1 x_w + r_2 y_w + r_3 z_w + T_x \tag{5.15}$$

$$y_c = r_4 x_w + r_5 y_w + r_6 z_w + T_y \tag{5.16}$$

$$z_c = r_7 x_w + r_8 y_w + r_9 z_w + T_z \tag{5.17}$$

  Extrinsic Parameter R [3DOF], T [3DOF]

- CCS $\rightsquigarrow$ Undistorted Image Point

  $p_c = (x_c, y_c, z_c)$ point in CCS $\rightsquigarrow$ $p_u = (x_u, y_u)$ undistorted image point

$$x_u = f\frac{x_c}{z_c} \quad (5.18)$$
$$y_u = f\frac{y_c}{z_c} \quad (5.19)$$

Intrinsic Parameter f [1DOF]

- Undistorted Image Point $\leadsto$ Distorted Image Point

  $p_u = (x_u, y_u)$ undistorted image point $\leadsto$ $p_d = (x_d, y_d)$ distorted image point

  with $r = \sqrt{x_d^2 + y_d^2}$

$$x_u = x_d(1 + k_1 r^2 + k_2 r^4) \quad (5.20)$$
$$y_u = y_d(1 + k_1 r^2 + k_2 r^4) \quad (5.21)$$

Intrinsic Parameter k_1 [1DOF], k_2 [1DOF]

- Distorted Image Point $\leadsto$ Memory Pixel

  $p_d = (x_d, y_d)$ distorted image point $\leadsto$ $p_m = (x_m, y_m)$ memory pixel

$$x_m = s_x \frac{x_u}{\triangle_x} \frac{\sharp xMem}{\sharp xCCD} + t_x \quad (5.22)$$
$$y_m = \frac{y_u}{\triangle_y} + t_y \quad (5.23)$$

Intrinsic Parameter S [2DOF], s_x [1DOF]

5.6 Tsai's Monoview Camera Calibration Method

Roger Y. Tsai proposed in [4] a method for calibrating static monoview CCD cameras of the shelf. This method takes a set of known non-coplanar calibration points as input and estimates autonomously and efficiently both extrinsic and intrinsic parameters of provable accuracy. [4]

Tsai's method attracted wide attention and thus is of high relevance because it considers lens distortion and none the less is efficient and autonomous.

5.6.1 Overview

Tsai's calibration method works in two stages:

- Prerequisites
  - $\sharp xMem$, $\sharp xCCD$, $\sharp yRows$, $\triangle_x$, $\triangle_y$ have to be known from device specification
  - S is set as sufficient heuristic to $S = (t_x, t_y) = (\frac{\sharp xMem}{2}, \frac{\sharp yRows}{2})$
  - Non-coplanar calibration points $p_{wi} = (x_{wi}, y_{wi}, z_{wi})$ to be measured in WCS

- Calibration memory image points $P_{mi} = (x_{mi}, y_{mi})$ to be found in an taken image of the world

- Stage One computes
  - Transformation matrix R
  - x- and y-components T_x, T_y of translation vector T
  - The horizontal scale factor s_x

- Stage Two computes
  - Effective focal length f
  - Radial lens distortion coefficients k_1 and k_2
  - z-component T_z of translation vector T

5.6.2 Stage One

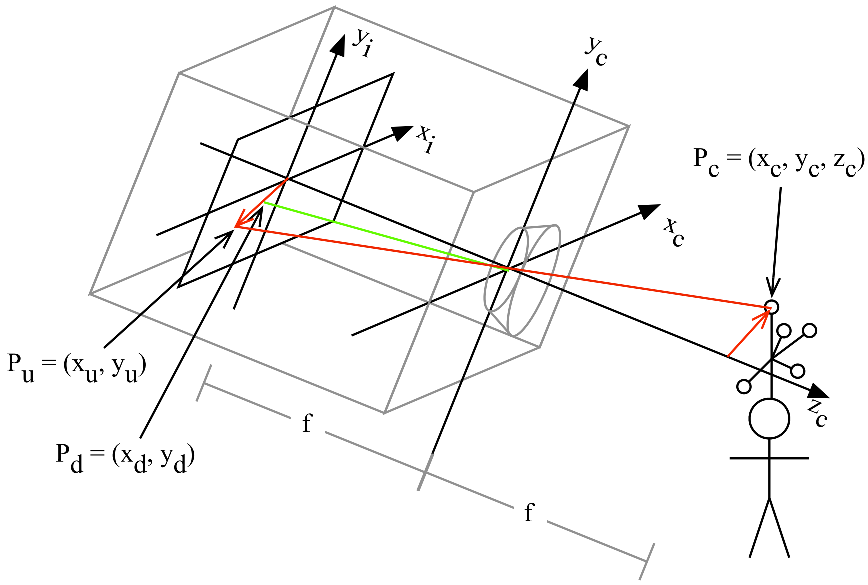

Figure 5.9: Parallelism Constraint

The first stage of Tsai's calibration method is based on the observation that the vector from the image origin $(0, 0, f)$ to the undistorted image point (x_u, y_u) is parallel to the vector extending from the point $(0, 0, z_c)$ on the optical axis to the point $p = (x_c, y_c, z_c)$. Additionally can be observed that radial lens distortion does not alter the direction of the vector as shown in Figure 5.9.

Thus following parallelism constraint holds

$$\overline{(0,0,f)(x_d,y_d,f)} \parallel \overline{(0,0,z_c)(x_c,y_c,z_c)} \tag{5.24}$$

This equation yields

$$\begin{pmatrix} x_d \\ y_d \end{pmatrix} = c \begin{pmatrix} x_c \\ y_c \end{pmatrix} \tag{5.25}$$

with c being a constant. This again leads to

$$x_d = cx_c, y_d = cy_c \Rightarrow x_d y_c = cx_c y_c = y_d x_c \tag{5.26}$$

Substituting x_c and y_c by their counterparts x_w and y_w in WCS transformed by R and translated by T yields

$$x_d = \frac{y_d x_w r_1 s_x + y_d y_w r_2 s_x + y_d z_w r_3 s_x + y_d T_x s_x - x_d x_w r_4 - x_d y_w r_5 - x_d z_w r_6}{T_y} \tag{5.27}$$

This equation is then used to obtain a over determined system of linear equations which can be solved. From this solution described parameters are extracted.

The different steps towards extracting parameters involved are as follows

- For each calibration memory point p_{mi} compute an interim distorted image point p'_{di} while setting the unknown horizontal scale factor to $s_x = 1$
- For each pair of points p'_{di} and p_{wi} formulate a linear equation $x_{di} = \ldots$ like in 5.27.
- In the resulting system of linear equations there are seven free terms, namely $\frac{r_1 s_x}{T_y}$, $\frac{r_2 s_x}{T_y}$, $\frac{r_3 s_x}{T_y}$, $\frac{s_x T_x}{T_y}$, $\frac{r_4}{T_y}$, $\frac{r_5}{T_y}$ and $\frac{r_6}{T_y}$.
- We choose the set of non-coplanar calibration points to have a cardinality larger than seven thus the system of linear equations is over determined and can be efficiently solved by application of the least square error method.

By applying basic geometric observations out of these seven terms R, T_x, T_y and s_x can be efficiently extracted which is described in detail in [4].

Thus at the end of stage two accurate values for R, T_x, T_y and s_x have been found efficiently.

5.6.3 Stage Two

In stage two the effective focal length f, the z-component T_z of the translation vector T as well as the two lens distortion coefficients k_1 and k_2 are to be found.

This is done in two steps

- **Step One** computes an approximation for f and T_z by ignoring lens distortion
- **Step Two** makes use of these approximations for f and T_z to compute exact solutions for f, T_z, k_1 and k_2

Step One

Ignoring radial lens distortion leads from

$$f\frac{y_c}{z_c} = y_u = y_d(1 + k_1 r^2 + k_2 r^4) \tag{5.28}$$

to

$$f\frac{y_c}{z_c} = y_u = y_d \tag{5.29}$$

By formulating the linear equation $f\frac{y_{ci}}{z_{ci}} = y_{di}$ linke in 5.29 for each calibration point i a system of linear equations is established.

Substituting y_{ci}, z_{ci} and y_{di} leads to

$$f\frac{r_4x_{wi}+r_5y_{wi}+r_6z_{wi}+T_y}{r_7x_{wi}+r_8y_{wi}+r_9z_{wi}+T_z}=\triangle_y(y_{mi}-t_y) \qquad (5.30)$$

The free variables f and T_z then are determined efficiently by solving this system of linear equations applying again a least square method.

Step Two

In step two exact solutions for f, T_z, k_1 and k_2 are to be found.

Therefor the approximation values of f and T_z are taken as initial guess for an algorithm solving the system of nonlinear equations determining the exact values of f, T_z, k_1 and k_2.

This initial guess is sufficient for the algorithm to work efficiently. Hence accurate values for f, T_z, k_1 and k_2 have been found autonomously and efficiently.

5.6.4 Tsai's Method for Stereovision Cameras

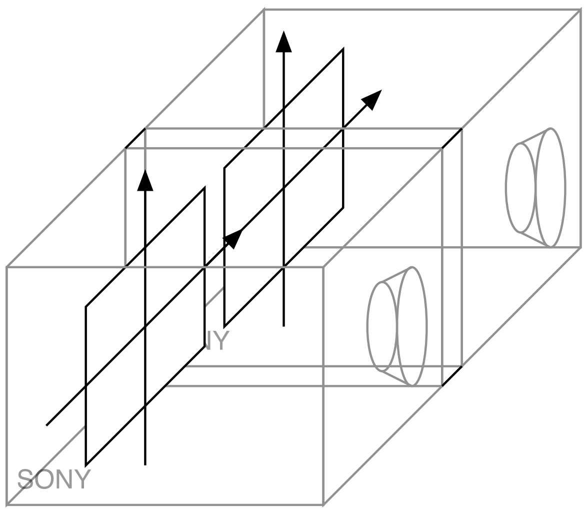

Figure 5.10: Stereovision Camera Model

Tsai's camera calibration method primarily is applicable to monoview cameras. But for optical tracking stereo vision is needed.

Stereoview camera constellations can be calibrated applying Tsai's method to each view resulting in a stereovision camera model as shown in Figure 5.10.

5.6.5 Conclusion and Variants

The Tsai calibration method for static monoview CCD cameras works efficiently and autonomously and hence is of great use in the field of AR.

- INPUT
  - Device specific data (resolution of CCD, image center in pixels, number of pixels scanned in a line)

    - Monoview image of non-coplanar calibration points of known coordinates in WCS

- OUTPUT
    - Extrinsic parameters
        * Camera pose relatively to WCS *[6DOF]*
    - Intrinsic parameters
        * Effective focal length *[1DOF]*
        * Horizontal scale factor *[1DOF]*
        * Radial lens distortion coefficients *[2DOF]*

Beside the presented calibration method Tsai proposes two other variants in [4]. The proposed variants are

- Single view with coplanar set of calibration points
    - Systems of linear equation with less free terms
    - Does not calculate the horizontal scale factor
    - Horizontal scale factor not necessary for some applications [4]
- Single view with non-coplanar set of calibration points
    - Presented variant
- Multiple view
    - Images are taken from different poses
    - Introduces new source of impreciseness

Because of its efficiency and accuracy together with the fact that it calculates the horizontal scale factor the presented variant seems to be the variant of highest relevance to the field of AR.

5.7 Calibration of an OST-HMD

In this section we want to describe the SPAAM (**S**ingle **P**oint **A**ctive **A**lignment **M**ethod) calibration procedure to calibrate Gerhards' optical see-through head-mounted display (OST-HMD) like it was proposed in [6]. Firstly a short description of the layout of the augmented reality setup used is given. Afterwards the calibration procedure is described in more detail.

The layout of the OST-HMD is shown in Figure 5.11. It consists of a scene generator generating the graphic images which are displayed on the monitor. The image of the monitor is reflected by an optical combiner to the user's eye. The optical combiner is quite sophisticated because it must enable the user to see the real world as well as the virtual objects. Moreover a six-degrees-of-freedom magnetic tracker is attached to the HMD. The magnetic tracker enables the calculation of the pose of the user respectively camera. With camera the virtual camera consisting of the goggles and the human visual system is meant.

In the last section Tsai's camera calibration algorithm was presented. The SPAAM is different because it doesn't compute the intrinsic parameters explicitly. Instead, a projective

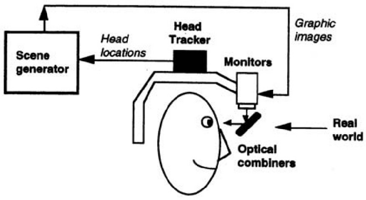

Figure 5.11: Schematic layout of a OST-HMD [3]

matrix transforming 3D real world coordinates to 2D image plane coordinates is used, where the camera distortion isn't taken into account. This leads to a simplified mathematical model and therefore to simplified calibration method. Another advantage of the SPAAM is that just one well-known point is necessary for calibration and that the camera mustn't stay fixed during calibration. As we use the SPAAM for the calibration of an OST-HMD the user hasn't to keep his head on the same position all the time, leading to a more convenient user interaction.

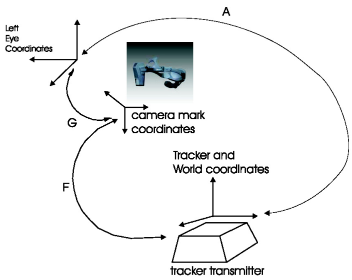

Figure 5.12: Coordinate systems of a monocular OST-HMD in an AR setup [6]

In a first step we will restrict the calibration to monocular optical see-through displays meaning displays just for one eye. In Figure 5.12 all the relevant local coordinate systems for a calibration of a OST-HMD are shown. The tracker coordinate system and the world coordinate system are the same to facilitate the calibration procedure. The goal of the whole calibration procedure is the determination of the 3 x 4 projection matrix **A** which maps 3D real points onto 2D image plane points. Like can be seen in Figure 5.12 the transformation **A** can also be described as follows:

$$\mathbf{A} = \mathbf{G}\mathbf{F} \tag{5.31}$$

where **F** is a 4 x 4 homogeneous transformation matrix that defines the relationship of camera mark coordinate system (the coordinate system defined by the tracking target attached to the HMD) and world coordinate system.

F is measured by the magnetic tracker system. **G** is the 3 x 4 projection matrix that defines the rigid transformation between camera mark coordinate system and left eye coordinate system. So a point in the world coordinate system p_w is projected on a point in the image plane of the virtual camera p_i according to:

$$\rho p_i = \mathbf{A} p_w = \mathbf{G}\mathbf{F} p_w \tag{5.32}$$

where ρ is a scalar. As $\mathbf{F}$ is measured, the only missing matrix is $\mathbf{G}$ which calculation is described in the following paragraph.

When users move their head freely within a scene, one specified point in the world coordinate system gets mapped to many different positions on the HMD screen - depending on the changing projection properties during a user's head motions. Tuceryan et al exploit this fact to find alignments between a single point in space and many projections on the HMD and into the user's eye. Let p_w be a point in the world coordinate system and p_m a point in the (head) marker coordinate system this is expressed by the following equation:

$$p_m = \mathbf{F} p_w \tag{5.33}$$

Therefore we know the point in the marker coordinate system and in the world coordinate system. If we would also know about the point in the left eye coordinate system we could calculate $\mathbf{G}$, because $p_i = \mathbf{G} p_m$. To get the coordinates of the point in the image plane a cross-hair is displayed to the user. This cross-hair has to be aligned to the real point. If it matches a button has to be pressed and the data is collected for the calibration. To reduce the systematic errors during the calibration process the user is encouraged to move his head as much around as it is possible by the tracker volume.

Now we will describe the details of the calculation of $\mathbf{G}$. It is a 3 x 4 matrix leading to 12 unknown parameters which have to be estimated. But because the matrix is a projection matrix just 11 parameters have to be calculated as it is defined up to a scale factor. Let $p_{m,j} = (x_{m,j}, y_{m,j}, z_{m,j})^T$ and $p_{i,j} = (x_j, y_j)^T$ be the coordinates of the point in marker coordinates and image coordinates at the j^{th} measurement. The following basic equation is fulfilled:

$$\begin{pmatrix} u_j \\ v_j \\ w_j \end{pmatrix} = G_{3x4} \begin{pmatrix} x_{m,j} \\ y_{m,j} \\ z_{m,j} \\ 1 \end{pmatrix} \tag{5.34}$$

where (u_j, v_j, w_j) are the homogeneous image coordinates of the projected point. The relation is:

$$\begin{aligned} x_j &= u_j / w_j \\ y_j &= v_j / w_j \end{aligned} \tag{5.35}$$

If we consider the layout of G

$$\begin{pmatrix} g_{11} & g_{12} & g_{13} & g_{14} \\ g_{21} & g_{22} & g_{23} & g_{24} \\ g_{31} & g_{32} & g_{33} & g_{34} \end{pmatrix} \tag{5.36}$$

and Equation 5.34 we get

$$\begin{aligned} u_j &= g_{11} x_{m,j} + g_{12} y_{m,j} + g_{13} z_{m,j} + g_{14} \\ v_j &= g_{21} x_{m,j} + g_{22} y_{m,j} + g_{23} z_{m,j} + g_{24} \\ w_j &= g_{31} x_{m,j} + g_{32} y_{m,j} + g_{33} z_{m,j} + g_{34} \end{aligned} \tag{5.37}$$

In consideration of Equation 5.35 and 5.37 we get

$$\begin{aligned} x_i(g_{31} x_{m,j} + g_{32} y_{m,j} + g_{33} z_{m,j} + g_{34}) &= g_{11} x_{m,j} + g_{12} y_{m,j} + g_{13} z_{m,j} + g_{14} \\ y_i(g_{31} x_{m,j} + g_{32} y_{m,j} + g_{33} z_{m,j} + g_{34}) &= g_{21} x_{m,j} + g_{22} y_{m,j} + g_{23} z_{m,j} + g_{24} \end{aligned} \tag{5.38}$$

If we put all the parameters of $\mathbf{G}$ into a column vector $\mathbf{p}$ we get the homogeneous equation

$$\mathbf{Bp} = \mathbf{0} \tag{5.39}$$

which has to be solved. Where $\mathbf{B}$ is defined as:

$$\mathbf{B} = \begin{pmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_{m,j} & y_{m,j} & z_{m,j} & 1 & 0 & 0 & 0 & 0 & -x_j x_{m,j} & -x_j y_{m,j} & -x_j z_{m,j} & -x_j \\ 0 & 0 & 0 & 0 & x_{m,j} & y_{m,j} & z_{m,j} & 1 & -y_j x_{m,j} & -y_j y_{m,j} & -y_j z_{m,j} & -y_j \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{pmatrix} \tag{5.40}$$

The matrix $\mathbf{B}$ has 2n rows, two for each data point. So at least six measurements have to be taken.

As mentioned above $\mathbf{G}$ has only 11 unknown parameters because the equation 5.34 is determined up to a scale factor. We try to get the vector $\mathbf{p}$ by minimizing $||\mathbf{Bp}||^2$ such that $||\mathbf{p}|| = 1$. This constraints the scale and therefore reduces the number of parameters to 11. The minimizing problem is solved by finding the eigenvector associated with the smallest eigenvalue.

The calibration procedure for stereoscopic displays is quite similar except that the transformations $\mathbf{A}$ and $\mathbf{G}$ have to be split up into A_L, A_R respectively G_L, G_R (see Figure 5.13) and that the alignment process has to be modified. Whereas in the monocular case just a cross-hair has to be aligned to the calibration point now a virtual three dimensional object has to be aligned. The objects for the two eyes are a little bit offset creating a disparity. The user's brain is creating a 3D object out of this. The calculation of the transformation matrices is quite similar to the monocular case.

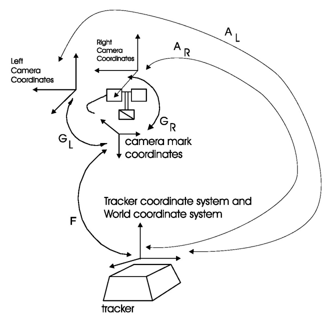

Figure 5.13: Coordinate systems of stereoscopic OST-HMD in an AR setup [6]

5.8 Image Calibration

In AR setups where no digital video but instead of a analog one is used a scan converter has to integrated into the system. The goal of the image calibration process is to estimate the distortion caused by scan converter and frame grabber. Reasons for misalignments of images are unequal delays in the horizontal and vertical sync processing of the analog video signals and that scan converter and frame grabber do not necessarily preserve the pixel aspect ratios of the images. The distortion introduced into the system can be seen as a linear transformation without rotation of the image points.

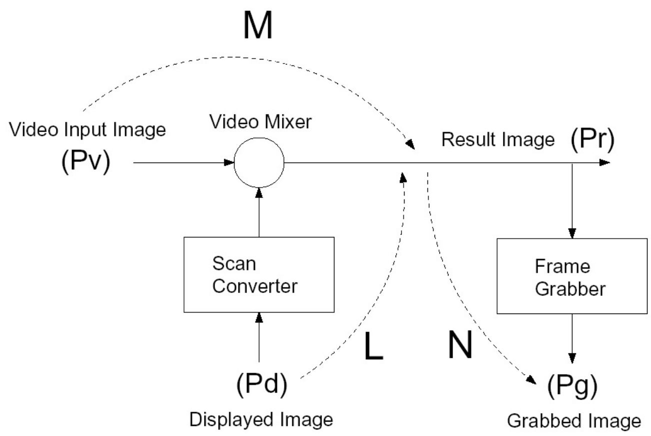

Figure 5.14: Image transformation [5]

In Figure 5.14 the schematic layout of the involved parts is shown. The distortion introduced by the video mixer to the video image is expressed by a matrix $\mathbf{M}$. The distortion caused by scan converter and frame grabber are considered by $\mathbf{L}$ respectively $\mathbf{N}$. If p_v is a point in the input image and p_d is a point in the computer generated graphics then the following equation has to be valid:

$$\mathbf{M}p_v = \mathbf{L}p_d \tag{5.41}$$

The calibration process bases on a physical pattern which is placed in front of the camera. We know about the positions of the points on the pattern in the real world and then measure the position of the points in the grabbed image p_g. The relationship is

$$\mathbf{NM}p_v = p_g \tag{5.42}$$

and we can calculate $\mathbf{NM}$.

Moreover we can measure the relationship between the points in the displayed image and the grabbed image. We send some sample points through the scan converter and frame grabber. In the grabbed image we localize the points again and can afterwards calculate the transformation $\mathbf{NL}$ because

$$\mathbf{NL}p_d = p_g \tag{5.43}$$

So we know about the correspondence from p_v to p_g and p_d to p_g. When we combine Equation 5.42 and 5.43 we get the desired correspondence between p_v and p_d fulfilling Equation 5.41:

$$(\mathbf{NL})^{-1}\mathbf{NM}p_v = p_d \tag{5.44}$$

5.9 Auto-calibration

Auto-calibration of tracking devices means the calibration of the devices without the usage of a special calibration procedure or special user interaction. Instead the calibration should be done automatic during usage. In most AR scenarios exists a mobile unit which moves in an closed environment for example a lab. The correct and complete calibration of the fixed units ,e.g. ceiling- or wall-mounted components, is assumed. The calibration of these fixed units requires specialised methods. When AR systems should not just be restricted to research labs, user-friendly methods have to be developed.

5.9.1 Self-Surveying of location

Sighting data and known fixed unit locations are used to calculate the location of mobile units. So a number of unknown quantities is used to determine a few unknown quantities. In most sighting data there exists more information than is used for location calculation. This *surplus data* makes self-surveying possible. If there are constraints on the location of mobile units additional surplus data can be received. We get additional information because we reduce the number of unknowns thus more data can be used for determining the location of fixed units.

For gathering data there exist three different methods. The *people* method logs data during regular use. While mobile units move around sightings can be gathered. As the position of the mobile units is completely unconstrained, less surplus data is produced. The next method is the *floor* method where many mobile units are placed on the floor of a space for a period of time. As the mobile units are approximately coplanar and therefore more constraint more surplus data can be received. In the *frame* method mobile units are placed onto known points on a rigid frame. The location of the mobile units is more constrained than in the other methods and produces therefore most surplus data. Moreover as the position of the mobile units is exactly known the just the location to the fixed units has to be computed.

For processing self-survey data two different techniques are presented. The first one is *Simulated annealing* which is an iterative algorithm to find the best solution. It tries to find a best guess. Each possible solution has to be evaluated, therefore a scoring method is used. The second technique bases on inverting the location algorithm. In the frame method the location of the mobile units is already known. Therefore a algorithm similar to that one used to track mobile units can applied except that the role of mobile and fixed unit is swapped.

5.9.2 Auto-Calibration of Cameras

In the following we will take a look at the auto-calibration of cameras. The reason why auto-calibration of cameras is so important is that the camera parameters may change during usage. The change of the extrinsic parameters is already considered in the tracking procedure. But also the intrinsic parameters can change due to mechanical or thermal variations and focusing or zooming. As there is not always a calibration grid available on which the former presented methods are based on another calibration procedure has to be applied. The auto-calibration or self-calibration bases on a number of image correspondences. This calibration technique is highly flexible as no more special calibration marks in the real world are needed. According to [1] the methods used for auto-calibration fall into on of the following three cases:

- The Kruppa Equations proposed by Maybanck and Faugeras
- A linear constraint on the calibration matrix pioneered by Hartley
- An approach that find the explicit location of the absolute quadric, shown by Triggs

Circumstances like a restriction of the camera movement or already known parameters can be used to optimise the above listed methods. All this methods are based on the fact that Euclidean transformations leave the absolute conic unchanged. So for the absolute conic are just the intrinsic and not the extrinsic parameters relevant. Thus the problem of finding the intrinsic camera parameters is the same as finding the image of the absolute conic. If we have enough different views of a scene under certain kinds of motion it is possible exactly identify the absolute conic. Right now we have not yet explained what the absolute conic is. It can be seen as a calibration object which is naturally present in all the scenes. For a more precise definition look at the online Tutorial of Marc Pollefeys [2].

5.10 Conclusion

In this paper we have presented calibration methods for pointers, objects, cameras, head-mounted-displays and images. More precise information regarding the calibration procedures can be found in the referred papers. We have also stated a new calibration trend, the auto-calibration. As the number of tracking devices which have to calibrated will grow in the future and the usage of an AR system should be convenient to the user it is an very important topic.

Bibliography

[1] N. Azizi, *Camera Self-Calibration*, April 2003.

[2] M. Pollefeys, *Tutorial on 3D Modeling from Images.* ECCV 2000, June 2000.

[3] T. Suthau and et al., *Konzeption zum Einsatz von AR in der leberchirurgie*, DGPF, (2002).

[4] R. Y. Tsai, *Versatile Camera Calibration Technique for High Accuracy 3D Machine Vision Metrology using Off-the-Shelf TV Cameras and Lenses*, IBM Research Report, (1985).

[5] M. Tuceryan and et al., *Calibration requirements and procedures for a monitor-based augmented reality system*, IEEE, (1995).

[6] M. Tuceryan, Y. Genc, and N. Navab, *Single point active alignment method (SPAAM) for optical see-through HMD calibration for augmented reality*, Presence, (2002).

6 Foundations of Ubiquitous Tracking

— **Christian Wachinger, Benjamin Fingerle**

6.1 Introduction

This section deals with Ubiquitous Tracking in augmented reality systems. Since this is a topic of ongoing research there are still a lot of unsolved questions existent. The aim of this paper is to list and structure these questions and - in some cases - also present possible approaches leading to resolution. We will deduct these open questions from a hypothetic scenario of an exemplary augmented reality setup.

We will consider this scenario in different levels of detail. In a first approach the scenario will be described from the perspective of an outside observer. In a second stage then we will look behind the surface and see how this augmented reality environment could be modeled using the DWARF service concept. Finally we will go even deeper into detail and characterize how the spatial relationships of objects in the scenario could be represented based on the Ubiquitous Tracking Framework.

6.2 Scenario

6.2.1 Equipment of Test User Gerhard

In the scenario presented a mobile user, Gerhard, uses an augmented reality setup in his daily working life. Gerhard wears an optical see-through head-mounted display. Up on his HMD two 6DOF (6-degree-of-freedom) markers, one optical and one magnetic, are rigidly mounted. This enables an external optical or magnetic tracking system to locate the orientation and position of Gerhard's head. Moreover a stereo-vision camera is installed on top of his head. Using those cameras Gerhard also is able to track objects by his own. To enable an easy tracking of Gerhard's hands he is additionally equipped with special gloves with optical 6DOF markers attached.

6.2.2 A Day in Gerhard's Life

Imagine our well-equipped mobile user, Gerhard, who just strolls down the TUM hallway. By stopping in front of a colleague's of his closed door a little window crops up showing his colleague in the office. Gerhard now has the possibility to greet his colleague by waving his hand. After greeting several of his colleagues Gerhard finally reaches his own office and steps in. After sitting down in front of his desk he starts reading a virtual message of his Russian friend Vladimir from St. Petersburg. Vladimir asks him for a chess game. Gerhard agrees and the *RemoteChess* application gets started diplaying a virtual chess board aligned on his desk. The virtual counterpart of Vladimir takes a seat across of him.

After playing chess for a while, suddenly an emergency request for instant help by his befriended Spanish colleague José shows up. José asks Gerhard, who is a renowned surgeon, for advises regarding to a difficult surgery José is currently conducting. Gerhard immediately abandons the chess game which leads to a disappearance of Vladimir and the chess board. Just where a minute ago the chess board has been visible a 3D model showing José's patient crops up. Gerhard studies the patient while having a look from different perspectives and consulting computer tomography images registered on the patient on his demand. In vivid discussion with José a life is saved.

6.3 Modeling the Scenario Using the DWARF Framework

As explained in detail in chapter 1 when using the DWARF framework augmented reality setups are basically modeled as sets of *Services*. These services exchange data via *Connectors* which again are established by *Service Managers*.

Services in DWARF have *Needs* and offer *Abilities* - the former further refined by *Predicates*, the latter by *Attributes*.

Connectors offer interfaces for data exchange to services by different means of transport and access, e.g. push- and pull access. These connectors are created by Service Managers that automatically detect mutually satisfying services by the meaning of needs and abilities. For each network node - which again could represent a room or a section of a building - there is one service manager.

6.3.1 Specifying DWARF Services

Using this concept we can model all entities in an augmented reality setup based on their *context* which - according to [1] - can be defined as the set of *location, identity, activity* and *time*. In the DWARF framework the context will be modeled using predicates and attributes. Context based modeling assures the desired high scalability of the augmented reality setup.

As shown by in Figure 6.1 and 6.2 DWARF services and their needs and abilities, predicates and attributes are specified in a XML description. Figure 6.1 specifies a general Optical Tracker Service and Figure 6.2 describes an Optical Tracker Service provided by a camera attached to a Head Mounted Display.

```
<service name="OpticalTracker">
  <need name="video" type="VideoStream">
    <connector protocol="sharedMemory">
  </need>
  <need name="marker" type="MarkerData">
    <connector protocol="ObjectReference">
  </need>
  <ability name="markerPose" type="PoseData">
    <attribute name="location" value=$(marker.location)>
    <attribute name="identity" value=$(marker.identity)>
    <connector protocol="NotificationPush">
  </ability>
</service>
```

Figure 6.1: Optical Tracker modeled as DWARF Service

```
<service name="HMDOpticalTracker">
  <need name="
    ...
  </need>

  <ability name="HeadPose" type="PoseData">
    <attribute name="location" value=$(landmark.location)>
    <attribute name="identity" value="Gerhard">
    <connector protocol="NotificationPush">
  </ability>

  <ability name="markerPose" type="PoseData">
    ...
  </ability>
</service>
```

Figure 6.2: Optical Tracker Service of Head Mounted Display

6.3.2 The Scenario Modeled with Services

Table 6.1 shows that part of the scenario taking place in the university hallway modeled with DWARF services.

Service Name	Needs	Abilities
GerhardConfigData	–	landmarkDescription
HallwayConfigData	–	landmarkDescription
HMDCamera	–	videostream
HMDOpticalTracker	videostream	markerPose, HMDPose
HMDVideoShow	videostream	–
WhatsBehind	pose	videostream
ContextEstimator	landmark	context

Table 6.1: DWARF services for university hallway location.

The part of the scenario taking place in Gerhard's office is modeled by services described in table 6.2.

6.3.3 Matching Mutually Satisfying Services

As already mentioned in 6.3 there exists one *Service Manager* for each network node providing mutually satisfying services within his local domain with *connector*-objects for data exchange. Therefore the service manager permanently observes the needs and abilities of all those services belonging to his local scope. When a match is found the service manager provides all involved services with connector-objects according to their needs and abilities. Connectors offer e.g. *push-, pull, or shared memory* access to abilities via various interfaces. (See Figures 6.1 and 6.2 as examples where the needed connector-types are specified with the key "`connector protocol=...`").

The UML diagrams in Figure 6.3 show a matching found for the *HMD Optical Tracker*, *WhatIsBehind-Application*, *HMDVideoShow-Application*, *HMDCamera* and *HallwayConfig-*

Service Name	Needs	Abilities
GerhardConfigData	–	landmarkDescription
HMDCamera	–	videostream
HMDOpticalTracker	videostream	markerPose, HMDPose
HMDVideoShow	videostream	–
ContextEstimator	landmark	context
RoomConfigData	–	landmarkDescription
RoomCamera	–	videostream
roomMagneticTracker	landmark	markerPose
roomOpticalTracker	videostream	markerPose
Desk	3D-content	–
VirtualChess	chessPartner	3D-content
VirtualSurgery	handPose	3D-content
VirtualCommunication	communicationPartner	3D-content

Table 6.2: DWARF services for Gerhard's office location.

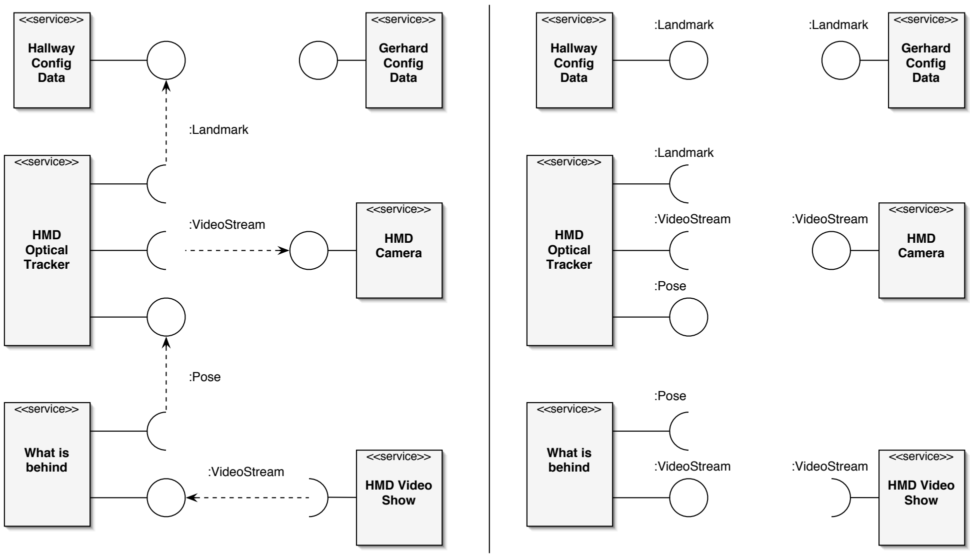

Figure 6.3: Matching of Mutually Satisfying Services

Data.

6.4 Ubiquitous Tracking

The crucial problem of current augmented reality systems is the correct tracking of objects. To improve the tracking procedure sensor fusion is used which tries to get better results due to the cooperation of different trackers. But there exists the problem that it is not so easy to integrate new tracking devices to the AR setup. One reason for this problem is the integration of the tracking task into the augmented reality application itself. It would be better to release the AR application from this task. Moreover because of the direct integration a lot of different special solutions for tracking problems exist.

Therefore it is desirable to introduce a new formal layer in between of the tracking process and the AR application (see Figure 6.4). So all requests for spatial relationships of objects are handled by the usage of the new layer. This should enable a seamless integration of new tracking devices and release the AR application from tracking details. The formal layer itself is formed by a formal framework called *Ubiquitous Tracking*.

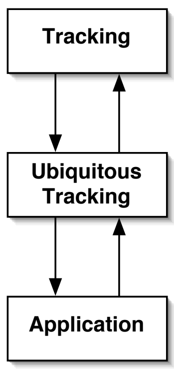

Figure 6.4: Introduction of a formal layer

Viewing at the framework as a black box, an AR application can just send requests for spatial relationships of various objects to it. The answer will be either the optimal available relationship between the objects or a message saying that no spatial relationship between these points is available. If a new tracker is available it has to register itself to the formal framework. The tracker device transmits information about the objects it is able to deliver spatial relationships for. When there is a request to the framework for a relationship involving this new tracker, the request will be passed to it. The tracker then responses with the desired current spatial relationship.

The Ubiquitous Tracking framework is internally based on a graph-model as can be seen in Figure 6.5. The nodes in the graph correspond to the objects in the real world. The edges in the graph represent the spatial relationships of objects. The edges contain information about the transformation and translation from the source coordinate system to the target coordinate system modeled as homogeneous 4 x 4 matrix. Moreover attributes characterising e.g. the quality of spatial relationships are stored with the edges. This could perhaps be the latency of the tracking device.

6.4.1 The Graph Model

There exist three different types of graphs which will we explained successively.

Real Relationship Graph

The real relationship graph delivers an idealized view of the world from the point of an omniscient observer. Meaning that there are spatial relationships between all available objects at every point of time. It exists a binary relation Ω on the object space $N = A, B, B, ...$ Each element (X, Y) of Ω is mapped to a function w_{XY} which describes the spatial relationship of the two objects over time. So we have

$$\mathbf{W} : (\Omega = N \times N) \rightarrow w, \quad \text{where} \quad w : D_t \rightarrow \mathbb{R}^{4\times4} \tag{6.1}$$

D_t is the source time domain which is in that case the whole time continuum. The directed graph G(Ω) describing the relation Ω is shown in figure 6.5. It is a complete graph as Ω is transitive, reflexive and symmetric.

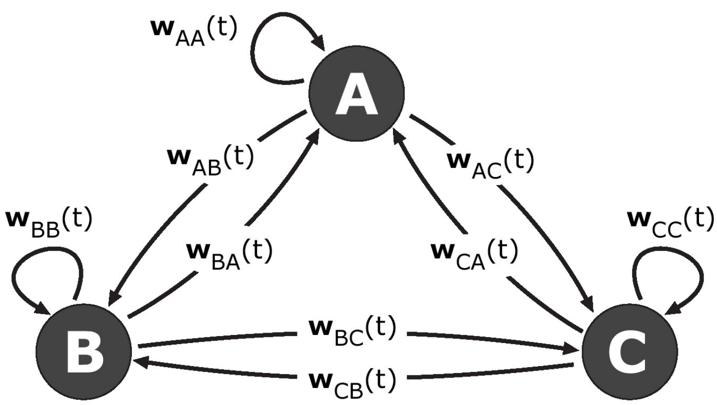

Figure 6.5: Real relationship graph [2]

Measured Relationship Graph

It is obvious that the real relationship graph is unrealistic because we will never know about all spatial relationships at every point in time. A more realistic graph which bases on the measurement of the spatial relationship will be presented now. As the relationships have to be measured there exist only such edges in the graph where we have tracking information about the nodes. Moreover the measurements are taken at discrete points in time yielding to a function p which is simply defined for certain times D_t. This function does not just deliver information about the transformation but also attributes $\mathcal{A}$ defining the quality of the taken measurements. Analogue to Ω a relation Φ is defined.

$$\mathbf{P} : (\Phi \subseteq N \times N) \rightarrow p, \quad \text{where} \quad p : D_t \rightarrow \mathbb{R}^{4\times4} \times \mathcal{A} \tag{6.2}$$

In figure 6.6 a example measured relationship graph is displayed. In this graph the object A knows about the spatial relationship to the objects B, C and B measures the spatial relationship to C.

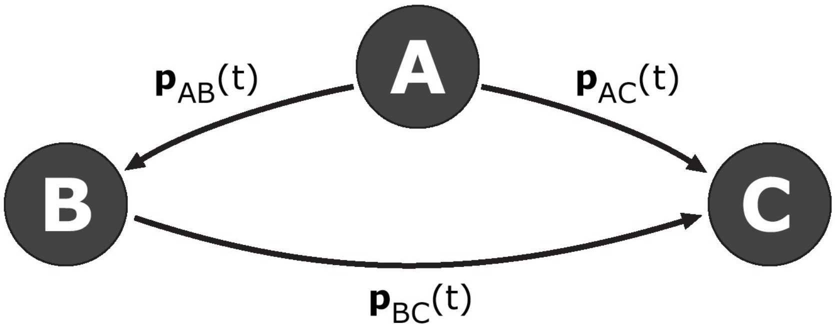

Figure 6.6: Measured relationship graph [2]

In this type of graph we already have attributes associated to the relationships. To evaluate the attributes we need an error function

$$\begin{aligned} e : \mathcal{A} &\rightarrow \mathbb{R} \\ \mathcal{A}_i &\mapsto e(\mathcal{A}_i) \end{aligned} \tag{6.3}$$

Inferred Relationship Graph

The disadvantage of the real relationship graph is that we only know about the relationship between objects for discrete points in time. Meaning that if we want to get the spatial relationship between two objects we just get an result when at this point of time a measurement was taken. As they are quite infrequent compared to the continuous time it is improbable to get any relationship.

So what we have to do is to continue the function p for the whole time interval in that measurements were taken. Therefore we infer knowledge about the spatial relationships of objects. Analogue to the other graphs we have a binary relation this time called Ψ which tries to approximate the world relation Ω:

$$\mathbf{Q} : (\Psi \subseteq N \times N) \rightarrow p, \quad \text{where} \quad p : D_t \rightarrow \mathbb{R}^{4\times4} \times \mathcal{A} \tag{6.4}$$

If we think about a function p_{AB} defined at the times $D_t = \{t_1, t_2\}$ a possible definition of a function q_{AB} could look like $q^m_{AB}(t) = p_{AB}(t)$ or

$$q^e_{AB}(t) = \begin{cases} p_{AB}(t_1) & \text{if} \quad |t - t_1| < |t - t_2| \\ p_{AB}(t_2) & \text{otherwise} \end{cases} \tag{6.5}$$

This is just a very simple possibility for inference. There also exist others like interpolation p^i_{AB}, Kalman filter p^k_{AB} or particle filter p^p_{AB}. When we have multiple inferences between objects A and B the graph could look like shown in Figure 6.7. So far we have extended the measured spatial relationship between objects for the whole measured time interval. But it is also possible to deduce spatial relationships between objects where no direct measurements exist. For example imagine three objects A, B and C. A can measure the spatial relationship to B but not to C. B can measure the relationship to C. So we have q_{AB} and q_{BC}. What we want is the spatial relationship from A to C. It is now possible to infer the relationship q_{AC} from q_{AB} and q_{BC} because the spatial relationships are transitive.

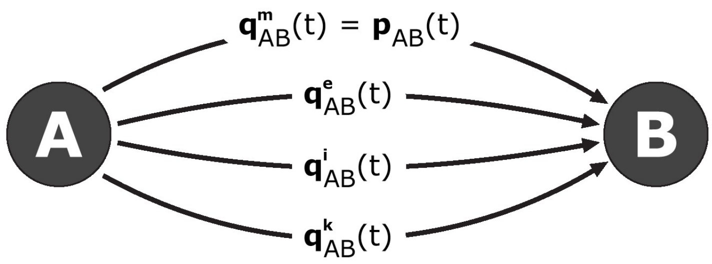

Figure 6.7: Multiple inferences between objects A and B [2]

Right now we add to our example that it is possible to measure the spatial relationship from A to C. Therefore there exist two different measurements for the spatial relationship between A and C. The first one is direct and the second one uses object B. Which one is the *better* spatial relationship? For deciding about the quality of the spatial relationship the error function is used. In Figure 6.8 a possible constellation is shown where boxes on the edges display the attribute.

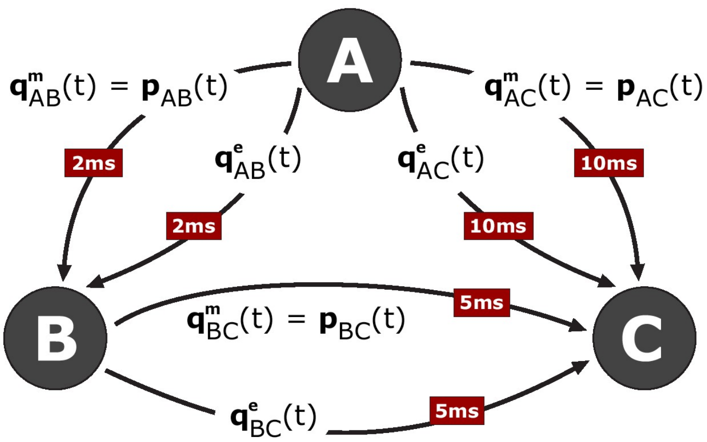

Figure 6.8: Inferred relationship graph [2]

6.4.2 Error Function

In the sections before we already mentioned the error function. Before we want to consider it in more detail we have to survey the attributes on which it is based on. The job of the attributes is to characterise the quality of a spatial relationship. Possible attributes are:

- Latency
- Update frequency
- Confidence value

- Pose accuracy
- Monetary cost
- Time to live

The attributes determine the properties of the tracker devices. It is not so easy to get the correct values for the attributes because it is hard to get correct parameters from manufacturers.

The error function has to evaluate the quality of a path in the graph. One simple error function regarding latency and update rate could be:

$$e^t := \sum_{q \in path} \text{lag}(q) + \frac{\lambda}{\text{rate}(q)} \tag{6.6}$$

There exist error functions which allow an edgewise evaluation of paths. For this functions it is easy to use well-known shortest path finding algorithms like Dijkstra's algorithm. But it is not always possible to make an edgewise calculation because sometimes it is necessary to apply the error function to the whole path.

6.4.3 Optimisation

There exist two possible optimisations that can reduce the computational complexity: Precomputation of data flow graphs and spatial hierarchies with supernodes.

Precomputing Data Flow Graphs

The data flow graph just depends on available spatial relationships between objects and their attributes $\mathcal{A}$. As they are both changing quite infrequently compared to the pose measurements it is possible to precompute the data flow graph. Let's have a look at the graph in Figure 6.8. The evaluation of the error function would lead to the result that the combination of q_{AB} and q_{BC} is better than q_{AC}. When there is a request for the spatial relationship between A and C the precomputed data flow graph can be used and just the current pose measurements have be taken into account.

Grouping Nodes

Another possibly for optimisation is to represent several nodes in the graph through one single supernode. Imagine two users, each of them wearing trackers and marks which correspond to nodes in the graph. It is now possible to merge several nodes, for example the left and right eye cameras and the fiducial node of a person as the spatial relationships do not change. It is possible to build more and more supernodes as can be seen in Figure 6.9, ending up with just one supernode. That technique enables different levels of detail. This is important because if Gerhard uses the *WhatsBehind* Service it is not relevant where the people exactly are in the room - it is just important whether they are in the room. This concept facilitates the search algorithms as there are less nodes in the graph. An interesting question is that of the coordinate system of the new supernode? There exist different possibilities, for example it could simply be the coordinate system of one of the combined nodes or a newly defined coordinate system.

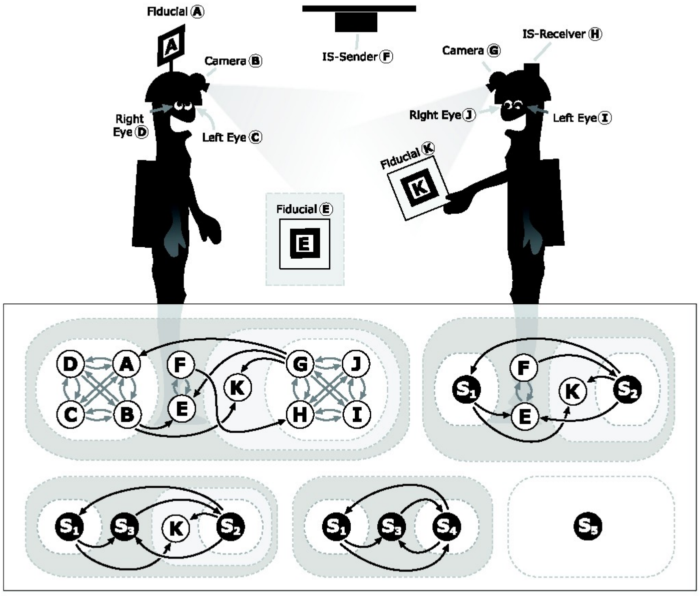

Figure 6.9: Usage of Supernodes [2]

6.5 Issues and Open Problems

6.5.1 Security and Safety issues

A topic which we have not considered so far is the security and safety of the augmented reality systems. Nevertheless it is a very important issue because of its direct influence on human's perception.

Every time when considering security topics, checking privacy, authenticity and integrity is a good starting point. If we think about *privacy* it is necessary to restrict the access to the mobile clients because a lot of private information may be saved there. Moreover it is also important to restrict the access rights of the mobile clients to offered services. For example it should not be possible to apply the *WhatsBehind* Service to every door. Another point concerning privacy is the stored tracking information. It should not be possible that a user knows about all the positions of other users. Otherwise users would be able to store all this position information leading to a violation of privacy rights of users.

Also very important is the *authentication* of users. Different users should have access to different services and data storage. Why should we know who wears the head mounted display? It is conceivable that a password has to be entered perhaps with the help of the pointing device or data glove. Concerning the auto-calibration aspect it could be possible to identify behaviour patterns of users with the help of trackers integrated into clothing.

The last point is the *integrity* of data meaning that no data is corrupted. How can we be sure that the distance to the truck is really 100 meters and we can risklessly cross the road? As you can imagine it is possible that man-in-the-middle attacks are possible and that they are extremely dangerous. But it is also possible to use current cryptography protocols to ensure the integrity.

So far we have described security issues concerning augmented reality systems but there are also safety reasons. Imagine a user wearing a head mounted display who tries to cross a street. What if suddenly a window pops up occluding all other road users? It must be ensured that augmented reality systems *augment* the real world with additional information but not mask it.

6.5.2 Issues regarding Service Manager Performance

Finding matchings, i.e. finding mutually satisfying services by the meaning of needs and abilities, forms a crucial performance issue. There is only one service manager per network node but the number of possible matchings grows exponentially with the number of services available within the service manager's local domain (take into account that all possible subsets of the set of services has to be checked for mutual satisfaction). This leads to limited scalability and has to be overcome.

Different strategies for coping with this issue could include

- Heuristics based on context information that are realized as graph searches on the spatial relationship graph. Only those sets of services are checked for mutual satisfaction that are locally close to each other. This approach reduces the number of sets of services dramatically but might not be feasible for all kinds of needs and abilities since there might be location independent (within the local domain) features.
- Interpretation of the matching problem as one huge predicate logic formula in disjunctive normal form likewise to the interpretation of a program's source code as a predicate logic formula. Time is discretised and each possible subset of services becomes an OR-clause which again describes all the matching pairs of needs and abilities in form of an AND-clause. On this huge predicate logic formula specialized algorithms known from theorem proving and model checking will be applied which might lead to drastic savings in terms of computational complexity.
- ...

6.5.3 Issues regarding Representation of Spatial Relationship Graph

The question how to digitally represent the spatial relationship graph basically forms a trade-off problem between the variety of applicable graph algorithms on the one hand and decentralization on the other hand.

Two possible ways of storing the spatial relationship graph are evident and forming the two extrema of above stated trade-off problem

- There is one graph-service holding the complete relationship graph. Certainly all graph algorithms needed are applicable but the distributed computing paradigm of the DWARF framework gets heavily violated.
- The relationship graph is stored implicitly in a distributed fashion by each service knowing its adjacency. This approach fully complies with the distributed computing paradigm but many graph algorithms cannot be applied directly like e.g. Dijkstra's algorithm for the shortest path problem.

6.5.4 Issues concerning Access to Information in Spatial Relationship Graph

Another question is how services could actually access the information about spatial relationships between objects.

Conceivable ways of accessing spatial relationships include

- Requests for spatial relationship information are formulated as a the requesting service's need for an ability *spatialRelationship* with predicates *source* and *target*. One possibility then is that the according abilities are offered by certain *GraphInformationServices*. This approach incorporates the advantage that the entire graph can be partitioned by several such GraphInformationServices. On the other hand different graph search strategies for the same relationship cannot coexist without some kind of workaround.

  One *relationshipInformationService* for each relationship between two objects could be an alternative to *GraphInformationServices*. These relationshipInformationServices then would be automatically instantiated and permanently updated by special *relationShipGeneratorServices*. As an advantage relationships would be instantly available but again different graph search strategies for the same relationship could not coexist without some kind of workaround. As an additional disadvantage of the latter approach the number of services would raise dramatically which according to 6.5.2 might lead to an infeasible high complexity.

- Services interested in a certain spatial relationship request a *graphInformerObject* providing an interface *getSpatialRelationShip(source, target)*. Following this approach several such graphInformerObjects can be instantiated and thus different graph search strategies can coexist. As a drawback this approach again leads to centralization violating the distributed computing paradigm.

6.5.5 Open Questions

Since this paper is meant to take over a summarizing function within the seminar *Ubiquitous Tracking for Augmented Reality* in the scope of the *Joint Advanced Student School 2004* we will end with an unordered and definitely in-exhaustive list of open questions that have come up during the seminar and did not find an immediate solution.

- How much and which information about the world has to be put into the error function?
- Which graph algorithms have to be applicable to the spatial relationship graph?
- Which representation lets theses algorithms become applicable?
- How to perform context changes of services?
- How to enable new users to enter an augmented reality environment?
- What has to be calibrated when a new user enters an augmented reality environment?

6.5.6 Outlook: Critical Success Factors

Certainly augmented reality is still rather a topic of ongoing research than of corporate research leading to concrete products in the near future.

Still we think its worth it to have an eye on those issues that have a critical impact on the success of future AR-solutions in the market. We found the following list of success factors to be the most eminent

- The number of compatible AR-ready buildings has to be reasonably high to convince customers of the solution's benefit.
- Therefor standards for AR-ready buildings and AR-user-equipment have to be developed and widely accepted.
- The convenience of an AR-solution as well as its price are certainly a major criterion for customers to buy a solution or not.
- Since head mounted displays distract its wearer and even might occlude the reality, legal issues definitely have to be taken into account - not only in road traffic.
- Finally the most compelling argument to buy a product is the product's use itself. Thus killer-applications urging the user to buy have to be found. This means beside the investigation of new techniques the research on possible AR applications exploiting already known technologies has to be seen as equally important.

Bibliography

[1] A. Dey and G. Abowd, *Towards a Better Understanding of Context and Context-Awareness*, in Proceedings of the CHI 2000 Workshop on The What, Who, Where, When, and How of ContextAwareness, 2000.

[2] J. Newman, M. Wagner, T. Pintaric, A. MacWilliams, M. Bauer, G. Klinker, and D. Schmalstieg, *Fundamentals of Ubiquitous Tracking for Augmented Reality*, Tech. Rep. TR-188-2-2003-34, Vienna University of Technology, 2003.