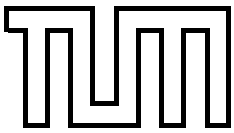
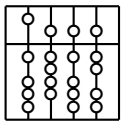


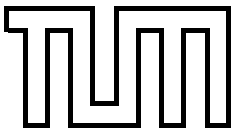
Technische Universität München
Fakultät für Informatik



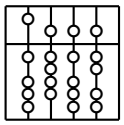
Bachelor Thesis

Analyzing and Monitoring Tracking Accuracy of an A.R.T. System

Christian Trübswetter



Technische Universität München
Fakultät für Informatik



Bachelor Thesis

Analyzing and Monitoring Tracking Accuracy of an A.R.T. System

Christian Trübswetter

Aufgabenstellerin:	Prof. Dr. Gudrun Klinker
Betreuer:	Dipl.-Inform. Hesam Najafi
Abgabedatum:	15. Oktober 2003

Ich versichere, dass ich diese Bachelor–Arbeit selbständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe.

München, den 15. Oktober 2003 Christian Trübswetter

Zusammenfassung

Ziel dieser Bachelor Thesis ist es zu untersuchen, welche Faktoren die Genauigkeit eines sogenannten infrarot–optischen Trackingsystems festlegen und wie eine gewisse Genauigkeit des Tracking sichergestellt werden kann. Darüber hinaus wird ein Tool entwickelt, das eine bestehende Trackingsoftware um die Möglichkeit erweitert, die Trackinggenauigkeit festzulegen und zu überwachen.

In der ersten Hälfte dieses Projektes ist das Ziel herauszufinden, welche Parameter welchen Einfluss auf die Trackinggenauigkeit eines infrarot–optischen Trackingsystems haben. Zuerst werden alle möglichen Faktoren, die das Tracking beeinflussen können, gesammelt und kategorisiert und eine vorläufige Einschätzung ihrer Relevanz abgegeben. Die verschiedenen Parameter werden dann mit dem infrarot–optischen Trackingsystem des Lehrstuhls, an dem diese Thesis erstellt wird, untersucht und anschließend ausgewertet. Dafür müssen zunächst Methoden entwickelt werden, die dafür benötigten Daten zu gewinnen und weiterzuverarbeiten. Um zum Beispiel den Datenaustausch zwischen dem Trackingsystem und einem Visualisierungstool zu ermöglichen wurden Scripts und Schnittstellen geschaffen.

In der zweiten Hälfte des Projektes wird untersucht, ob und wie das aus der vorangegangenen Studie gewonnene Wissen dazu verwendet werden kann, die Trackinggenauigkeit zu überwachen oder zu verbessern. Um dies zu zeigen, soll ein Tool für die A.R.T. GmbH, dem Hersteller des oben genannten infrarot–optischen Trackingsystems, entwickelt werden, dass die Genauigkeit des Tracking überwacht.

Abstract

The aim of this bachelor thesis is to explore which criteria limit the precision of an infrared optical tracking system, and how a certain degree of precision can be guaranteed. Furthermore a tool is developed that enhances an existing tracking system software with the capability of setting and monitoring the tracking accuracy.

The main task in the first half of this project is to find out which parameters determine and influence infrared optical tracking. After an intense brainstorming on parameters that are relevant for tracking accuracy, all these items are categorized and a preliminary estimation of the relevance of each of them is made. Next, some of these criteria are extensively assessed using the infrared tracking system of the chair this thesis is done at. Therefore procedures for measuring and evaluating scientific data have to be established first. For example, scripts and interfaces have to be created in order to establish data exchange between the tracking system and visualization tools.

In the second half of this project it will be evaluated, if and how the knowledge and results of the previous study can be used for monitoring or optimization of tracking accuracy. To demonstrate this, a precision monitoring tool shall be developed for A.R.T. GmbH, manufacturer of the before mentioned infrared optical tracking system.

Preface

About This work was done as a bachelor thesis in computer sciences at the Technische Universität München (TUM). My supervisor was Prof. Gudrun Klinker, my advisor was Hesam Najafi, Ph.D. student at the TUM, from the chair of applied software engineering at the TUM.

Acknowledgements There are quite a few people I want to thank. Prof. Klinker for contacting me and offering this work to me, Hesam Najafi, my internal advisor, for motivating me and giving me much freedom with my ideas, and my two external advisors from the A.R.T. GmbH, Armin Weiss for helping me in finding appropriate solutions and Kurt Achatz for his quick and helpful answers due to questions on implementation.

Finally I want to thank my family, my girlfriend, and all my friends for having had so much patience with me over the last three months, when I always rejected their invitations. They haven't seen me too often ultimately, but I promise I will make up to it.

Contents

List of Figures

1 Basics of Augmented Reality and the A.R.T. Tracking System

1.1 Introduction to this work

1.1.1 Scope

This work explores the precision of an infrared optical tracking system and how the results of the exploration confirmed the idea of developing a precision monitoring tool. Also the process of this development is described within this work. Here is a small list of items that delimit the coverage of this work:

- The exploration and research as well as the developed software are specific to the A.R.T. infrared optical tracking system. However, some of the procedures established for this study on precision may be applicable also for other infrared optical tracking systems, but in general the results may not be the same for other systems.
- A short introduction on how an A.R.T. system works will be given. A deeper insight into the A.R.T. system be found at [1], [2], and [3]. Also some basics on how optical tracking works will be given. Profound information on the tracking theory, such as the reconstruction of 3D points out images or image recognition, can be found at [8] for example.
- The development of the monitoring tool is described down to a certain depth. Requirements elicitation and analysis, system design and component design are described. The details on the implementation, however, are left out with a few exceptions. A description of the source code developed with this work can not be given within this document because of its extent. The reader is required to dig into the commented source code. However, the API of the reusable components is extensively described in appendix B.
- Users of the monitoring tool will find a short manual in the appendix A.

1.1.2 Goals

The goals of this work are not completely defined in the beginning, instead they evolve with this work. The initial goal is to use the ifrared optical tracking system of the A.R.T. GmbH, in the following called "*A.R.T. system*", in order to find out how the tracking precision of this system might be influenced and then to think about a possibility to measure, analyze, monitor, or improve the system's quality regarding tracking precision.

The goal of the second part of this work will be to demonstrate that developing a monitoring tool, based on the previously gained knowledge, makes sense. Another goal in this context is to record and describe the rationale behind the planning and evolution of the monitoring tool.

1.1.3 Structure

This document consists of four parts.

In the first part, basic terms in the context of augmented reality and tracking are explained and the A.R.T. system is described.

The second part the focuses on exploring all possible criteria that might influence the precision of infrared optical tracking. The next step was to measure, visualize and evaluate these criteria by using an A.R.T. system. Features that prove to be critical for the tracking accuracy of the A.R.T system are explored in more detail.

The third part describes the development of a precision monitoring tool based on the results of the tests described in part two. Its focus lies on describing the tool's architectural and detailed software design as well as the rationale behind implementation decisions.

In the last part, to–dos, testing results and future tests at the client side are covered. It will be illustrated, how the monitoring tool could be improved and enhanced. Finally, an short outlook on the future of optical tracking will be given.

1.2 Introduction to augmented reality and tracking

The tracking system this work deals with is mainly used for augmented reality applications. Therfore, a short introduction to augmented reality and the A.R.T. system will be given.

1.2.1 Definitions of terms

This subsection explains a few essential terms that have to be understood in order to read the text. Some more terms can be found in the glossary at the end of this work. Some of these terms have a special meaning in the context of this work. This is due to the fact that this work was done with a single tracking system, where these terms have a sharp and dedicated meaning.

Augmented reality: A definition of "*Augmented Reality*" is given by Professor Gudrun Klinker, who is Professor with focus on augmented reality at the TUM and supervised this work[1]:

> Augmented Reality (AR) is a newly emerging technology by which a user's view of the real world is augmented with additional information from a computer model. With mobile, wearable computers, users can access information

[1] Definition taken from [10].

without having to leave their work place. They can manipulate and examine real objects and simultaneously receive additional information about them or the task at hand. Using Augmented Reality technology, the information is presented three-dimensionally integrated into the real world. Exploiting people's visual and spatial skills to navigate in a three-dimensional world, Augmented Reality thus constitutes a particularly promising new user interface paradigm.

Research in Augmented Reality and wearable computing is beginning to receive more and more attention due to striking progress in many subfields and fascinating live demonstrations (due to advances in computer miniaturization, mobile networking, and sensing technology). Augmented Reality, by its very nature, is a highly inter-disciplinary field, and Augmented Reality researchers work in areas such as signal processing, computer vision, graphics, user interfaces, wearable computing, mobile computing, computer networks, distributed computing, information access, information visualization, software engineering, and the design of new displays.

The first picture in figure 1.1[2] shows real world objects, the second picture shows a virtual reconstruction of these objects plus additional objects (yellow object). In AR applications, the real world is enriched witch such additional objects or information.

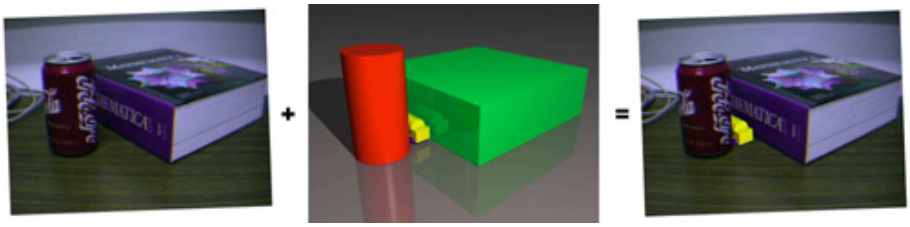


Figure 1.1: Illustration of augmented reality

It can be seen that augmented reality enables to embed parts of virtual realities into the real word. Therefore, real objects have to be tracked (which means their position and orientation has to be determined, usually in real–time). Based on this information and on additional information such as the shape of the single objects, a virtual model can be created. Additional components, e.g. textual information, graphical user interfaces, or additional objects, can be integrated into this model. Their appearance from the user's point of view can be computed, possibly with realistic lightning and overlapping. Thereafter these components are displayed on the user's view.

Optical tracking: Tracking in the context of augmented reality means detecting objects in space and determining their position, possibly also their orientation. In optical tracking light is used for getting information about an objects position. In most cases cameras are used for receiving this light, though also other light–sensitive receivers could be used, principally. In short, four elements are needed: so–called markers (see the definition below), light, one or more cameras, and a computer. The cameras take pictures from the scene. Then markers

[2]Image taken from [10].

have to be identified on the pictures using image processing techniques. Two techniques have found broad application:

Type 1: If one camera is used, at least three distinguishable points on an object forming a triangle have to be identified in order to find out the object's position and orientation relative to the camera (see figure 1.2). The form and size of the triangle must be known. A method called triangulation that is explained later is used for computing the triangle's vertices. Alternatively to distinguishable points, more than three indistinguishable points can be recognized if the pattern they form enables to identify these points. This is usually guaranteed by forms that have no symmetries.

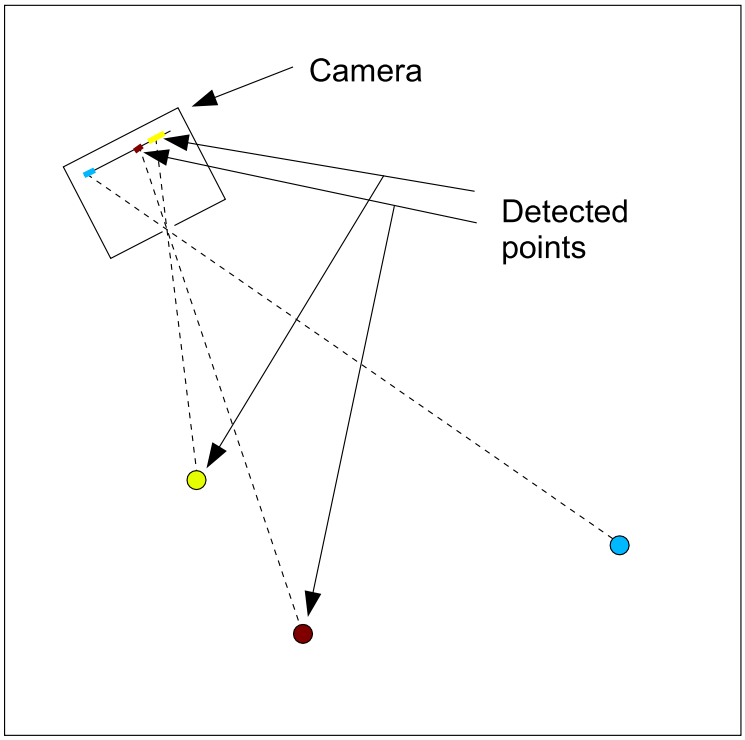


Figure 1.2: Optical tracking using one camera

The recognized points can be arbitrary distributed over an object. But they can also be on one physical marker (see figure 1.4). The only condition is that the points do not move relative to each other.

Type 2: A different approach uses more than one camera (see figure 1.3). These cameras have a fixed position relative to each other. The markers can be small, without any geometric structure. The same marker has to be identified by at least two cameras. As in the first case, the triangulation can identify a point. The difference is that only the position of the marker can be computed. If orientation is needed, a group of at least three markers is necessary.

The A.R.T system uses the second type for tracking markers.

Triangulation: According to *"hyperdictionary"* (see [21]) the word triangulation has the following two meanings:

> *1 a method of surveying; the area is divided into triangles and the length of one side and its angles with the other two are measured, then the lengths of the other sides can be calculated*
>
> *2 a trigonometric method of determining the position of a fixed point from the angles to it from two fixed points a known distance apart; useful in navigation (from hyperdictionary)*

Although these definitions are closely related in a mathematical sense, optical tracking generally uses the second form of triangulation. In case of the A.R.T system the triangulation works as illustrated in figure 1.3 (To simplify matters the two cameras are drawn as pinhole cameras).

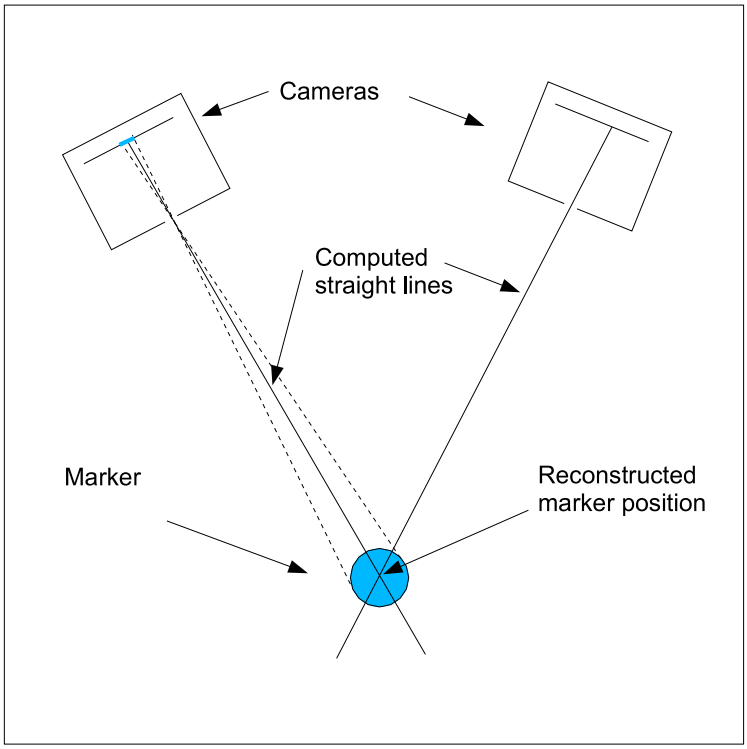


Figure 1.3: Triangulation

The two cameras take a picture of the scene, detect the projection of a spheric marker (definition see next paragraph), and compute its center. Then a straight line is constructed through this center and through the pinhole. The markers position is at the intersection of the two straight lines. Notice that the intersection point can only be computed if the position and orientation of the cameras in world coordinates are known. Also the size and the roundness of the marker's picture can be evaluated.

Marker: A marker in context of optical tracking is an object whose position in space can be determined by a tracking system. Additionally, some markers give information on their orientation. Objects that only provide position data are called objects with three **d**egrees **o**f **f**reedom, or in short 3DOF objects. Objects which give additional information on their orientation are called 6DOF objects.

6DOF markers are usually stickers that contain a certain number of asymmetrical points or a pattern. Two examples are shown in figure 1.4[3].

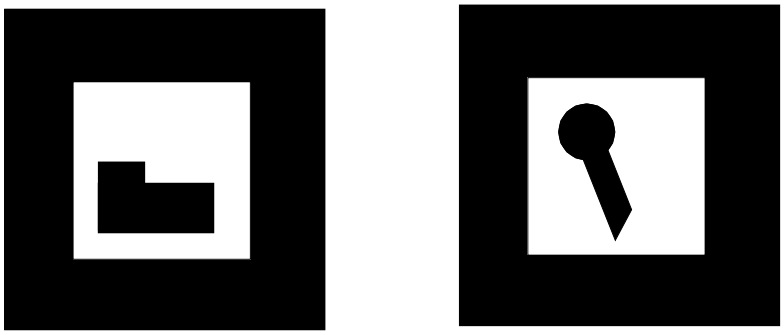


Figure 1.4: 6DOF marker for the ARToolkit

3DOF markers often have the form of discs or spheres. In order to be recognized in an image, the markers have to differ from the rest of the scene. The light sent from the markers has to have a different characteristic as the background. Either the colour, the saturation or the intensity, or each combination of them has to be unique in the image. This can be achieved by choosing unusual colours with high saturation (neon colours for example) or fluorescent materials. By combining at least three of such markers to so–called bodies, the orientation of the object can be computed. The result is a 6DOF body, sometimes called marker tree, when the structure looks like a tree.

The A.R.T. system uses markers in the form of small spheres that highly reflect infrared light (see figure 1.5 [4]). A special foil produced by 3M provides this effect. The spheres' diameters vary from $10mm$ up to $40mm$.

Residual: The term residual has various significations in science. Even in mathematics the term is not unambiguous. Usually the term means the result of solving circular integrals over poles of a complex function. In spacial geometry and consequently in optical tracking however, the residual is defined as the minimal distance between two skew straight lines.

[3]The ARToolkit is an optical tracking system that is open source and can also be used with an ordinary PC camera. It is available at [4].

[4]Image taken from the download section of the A.R.T. homepage (see [1]).

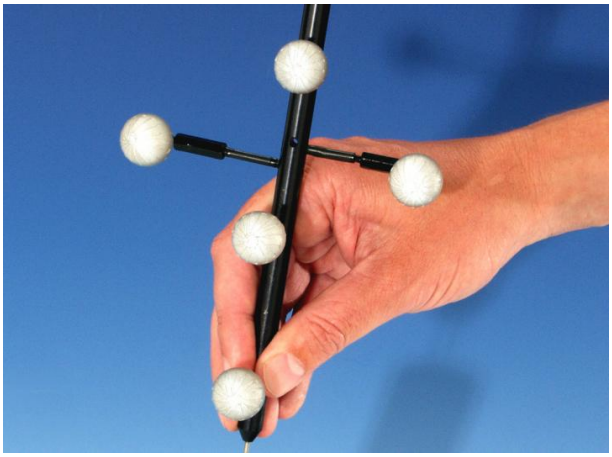


Figure 1.5: Marker from A.R.T. forming a body

This distance is a short line which is perpendicular to both straight lines. The straight lines in the A.R.T. system are the light beams that go from a marker through the cameras' lenses to a CCD chip. If the beams' residual is small, it is assumed that the light comes from the same marker and the markers position is evaluated. The detected marker is with a high probability somewhere near the short perpendicular line. Therefore, the residual gives information about the tracking precision.

When the A.R.T. system is operating with more than two cameras, there are also more than three lines. Then the minimal distances of each pair of lines is evaluated and a weighted average is computed. For each marker this residual is a value given in mm, that is usually less than half a millimeter in central tracking area with a typical setup and 2mm in the worst case. If the residual is worse, the marker is not detected.

In this work, the terms "*residual*", "*precision*" and "*accuracy*" are used synonymously and always mean a real number in millimeters that the A.R.T. system provides for every marker and that is computed as described here.

1.3 The A.R.T system

In order to understand how the A.R.T. system looks like and how it works, a short introduction to the system will be given. The full name of the system is "*ARTtrack1/DTrack system*", as it contains cameras of the type "*ARTtrack1*" and the software "*DTrack*". However, the term "*A.R.T. system*" is in common use. A detailed description of the A.R.T. system can be found at the A.R.T. homepage (see [1]). Only basic information and information relevant for this work will be presented here.

A typical setup consists of either 2, 4, or 6 cameras. The distance between two neighbouring cameras is usually between 1 and 5 meters. Figure 1.6[5] shows a setup used for this work. If only two cameras are needed, the others are simply turned off.

[5]Image taken from the A.R.T. system's manual (see [2]).

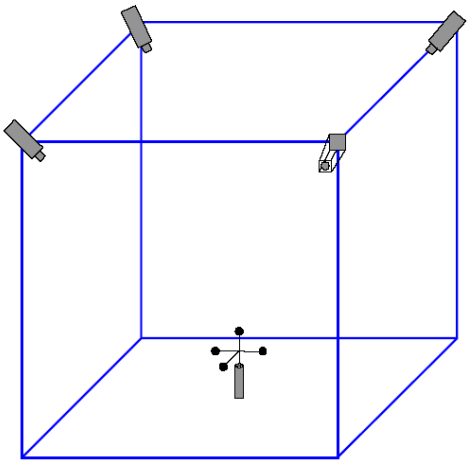


Figure 1.6: A typical A.R.T. setup with four cameras

1.3.1 The ARTtrack1 camera

The infrared cameras of type "*ARTtrack1*" (see figure 1.7[6]) are intelligent, equipped with an internal processing unit, a PC104 (an industry standard) with an AMD586 processor. A Realtime Linux runs on it. The cameras illuminate the measurement volume by an infrared flash and receive an infrared image using an infrared sensitive CCD chip. The information is transmitted via a PCI card developed by A.R.T. PCI card and processor filter and compute the 2D position and some parameters for each marker found in the image. These data are synchronously sent to the so–called DTrack PC via ethernet.

1.3.2 Interface to the DTrack PC

A sync slot card for the PC synchronizes the cameras with an internal signal, created by the sync card, or an external sync signal. The signal is used to apply delayed sync to up to three groups of cameras, in order to avoid mutual blinding of the cameras. Additionally the use of the tracking system together with IR controlled shutter glasses is made possible with the sync card signal.

1.4 The tracking software

The software delivered with the A.R.T. system is called "DTrack" (see figure 1.8). Version 1.18.2 was used for this work . It enables to set all necessary parameters both for the camera setup and for the network where the tracking data is passed to. Furthermore, all 2D marker

[6]Image taken from [1].

Figure 1.7: Camera ArtTrack1

information is received from the cameras. These data is processed so that the position of the markers and the position and orientation of the bodies can be retrieved. A complete description of the software is available at [1].

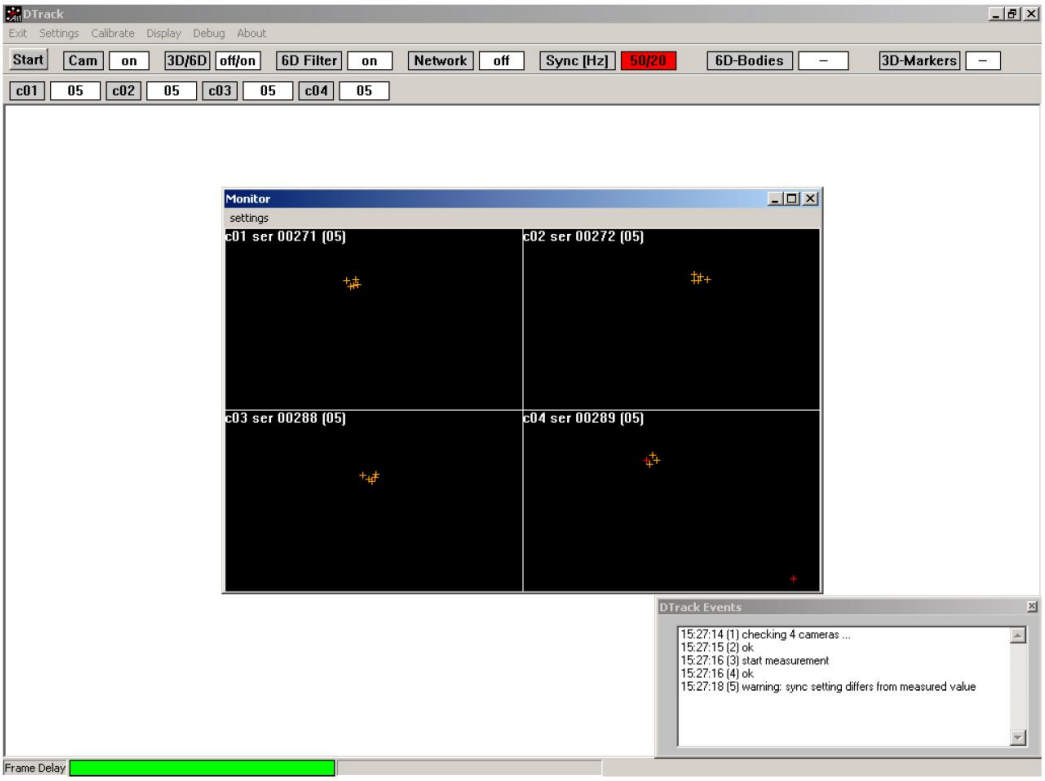


Figure 1.8: The DTrack software

1.4.1 DTrack

The software is installed on the central PC, where all data from the cameras run together and where the sync and ethernet cards are installed. It currently runs on Windows NT, 2000 and XP systems and a porting to Linux systems is planned. The DTrack software controls the tracking system, for example the intensity of the flash light or the sync rate, and samples the 2D data coming from the cameras. Next, 3DOF markers are computed out of these data and out of information on the cameras' position. If a group of markers can be matched to a body, the 6DOF position of the body is calculated. The standard deviation is also computed for each value. These data together with a time stamp and a frame number are sent to other computers in the network.

1.4.2 Connection to other applications

The software provides an interface to external applications via the UDP (User Datagram Protocol). The port where the data is sent to, as well as the IP addresses of one or more target computers, can be selected arbitrarily. Besides all the raw data from the cameras and some intermediate results of the 3DOF calculation, the very residuals of each marker for example, can be saved to disk as a log file or sent over UDP to a specified computer. This second interface, however, is not officially published by A.R.T. This undocumented interface had to be used, since the program developed within this thesis needs the residual information of the markers from DTrack.

2 Research on Accuracy of the A.R.T System

Now that the functionality of the A.R.T system has been cleared in principle, this chapter will demonstrate what degree of precision is possible to reach with this system, which parameters have influence on the tracking precision and to what extend, and how the information on precision was collected, processed and evaluated.

2.1 Motivation

For certain AR-applications high precision tracking is needed. Examples for such fields are measurement engineering, medical technology or applications with fast smooth movements. Whereas in the first two fields absolute information is important, if not even vital, for example in AR supported surgery, the latter one needs a high relative precision in order to achieve a certain smoothness, hence the absolute accuracy should be less important. It is also possible to smooth the position of an object in 3D space after tracking as a postprocessing, but smoothing conflicts with agile movements. The smoother a movement is made, the more time frames one needs to interpolate and the larger the time lag is.

Since there is not sufficient information on how the tracking precision of the A.R.T. system was influenced by different parameters, a study will be done in the following that analyzes tracking accuracy. Only when the main parameters are made out and they can be explained, it makes sense to create a tool that monitors tracking accuracy.

The main problem with monitoring precision is, that the accuracy of tracking is neither constant within the tracked area nor constant over time. That is why every single parameter can have different influence on different parts of the tracking area. Once there is a rough understanding of these dependences, proposals can be made about different monitoring algorithms, and about how certain quality states are defined. Furthermore, rules can be defined in which cases this quality is not sufficient anymore and a warning should be sent.

2.2 The exploration phase

The exploration phase mainly consists on collecting all kinds of possibly relevant tracking data and analysing them. As a first step, working through texts on tracking and doing brainstorming results in a list of criteria that should be tested. Next procedures are established for recording, analyzing and visualizing tracking data. Then measurements are performed and procedures defined before using them for processing the data.

2.2.1 Parameters influencing tracking quality

There is a variety of possible influences on the precision of an optical tracking system. In the following paragraphs all found parameters are shortly described regardless of their assumed quantitative importance.

First some physical parameters that are likely to influence tracking are discussed.

Movements: The quality of tracking is influenced by small movements of the cameras. This movements may be triggered by external forces, for example an unintended kick against the system setup, or simply by natural deformations of the system due to gravity and material properties. Especially where high leverage takes effect on soft materials over time, deformation can lead to changes of the camera position. According to A.R.T., shifting a camera of only a tenth of a degree makes tracking impossible and requires a recalibration of the system set up. This is a consequence of the high tracking precision. This system allows a preciseness of less than half a millimeter in parts of the tracked area. On the other side, if you turn a camera 0.1 degrees around its axis, it makes up 3.5 millimeters at a distance of 2 meters. In order not to allow a too big error the maximum shift of a tracked object should be limited to 2 or 3 millimeters.

Heat: Heat is another physical value with influence on tracking. The overall temperature of the tracking system and the tracked area, thus the room temperature, might have effects on tracking. But also the cameras themselves produce heat that slightly changes the form of its components and therefore has a strong influence on tracking.

Vibrations: A.R.T systems are often used in rooms with mechanical machines, such as motors or medical devices, that contain rotating parts. These parts often cause vibrations of the floor, that is also transferred to the camera system. As discussed before, even small angular movements due to vibrations lead to unsatisfying tracking results. In addition to that, these vibrations also quicken permanent shifting of parts of the camera system.

External light: The AR-system works with infrared light sensitive cameras, so all objects emitting or reflecting infrared light might disturb the system. Especially sunlight contains infrared light that could disturb the A.R.T. system, whereas artificial neon glow lamps shouldn't have influence on it. Reflection might be a problem as it is often unknown which materials reflect infrared light.

Smoke: Particles in the air like smoke or dust have two effects. First, they absorb a part of the light, decreasing the intensity of the infrared light. The cameras' visibility of the markers might be pejorated. Second the light is diffracted by particles, leading to light dispersion. The image of a marker becomes washy and diffuse. Both effects have influence on tracking precision.

Most relevant physical parameters have been discussed now. Next, items will be discussed that depend on the system set–up:

Position of cameras / size of tracked area: It is obvious that the position of the tracking cameras to each other influences tracking. A camera can track better, the nearer a marker is and the closer it is to the camera's optical axis.

Number of cameras: Most A.R.T. systems are delivered with two or four cameras, very few with six cameras. The more cameras there are, the larger the tracked area can be. A further advantage of more cameras is that big objects can be placed in the room, for example a car which is tracked from all sides. Even when the sight of a camera towards a marker is interrupted by other objects or people working with the system, two other cameras will probably track this marker.

Camera quality: The quality of the cameras' lenses and the resolution and light sensitivity of the optical chip determine which accuracy can be achieved with the A.R.T system. Whereas the resolution of the optical chip is only 640×480 pixels, the sensitivity is relatively high, so that a markers position can be determined to $1/50$ of a pixel (see paragraph "*Tracking Algorithm*" below). Unter optimal conditions a 3D marker can be tracked witch a precision of less than 0.3 millimeters. As each camera is assembled manually and as each lens is not equal, there are slight deviations of each camera parameter. Therfore, the quality of the single cameras is not the same. Some have a higher aberration on the boarders of the lens, and altough most aberration effects are minimized by a camera internal correction, asymmetric or local deviations of tracking accuracy will always remain.

Room calibration: In order to receive a high absolute accuracy a room calibration is necessary. The calibration finds out the position and orientation of the cameras in a coordinate system that is determined by the calibration angle seen in figure 2.1. In addition to that there is a wand with markers that is moved around the area to be calibrated. The system calculates a room calibration that is optimal for all points received from the wand. In an A.R.T. system that is used for tracking objects on a desktop for example, the room calibration may only be done on a limited space of the whole tracking area: the space around the desktop's working area. The question is, how precise the tracking system works in other regions of the tracking area where no calibration data was taken. The other question is what happens if calibration data from all over the tracking area is received, but only a small area of it is used. The tracking would probably be more exact in that regions, if calibration data of only these parts were taken. The data from other, maybe less central regions, could pejorate the calibration for the working regions.

Intensity of the infrared light: The intensity of each camera's flash light can be modified. If the tracked area is large and therefore the distances between the camera and the tracked markers are long, a higher intensity may be required, whereas intensity should be reduced if unwanted reflections from non-markers occur.

Size of markers: The bigger a marker is, the more pixels its image contains. And the more pixels are available, the more precisely the position of a marker can be interpolated. On the other side, the bigger the marker is, the more difficult it is to guarantee the same kind of

Figure 2.1: The A.R.T. calibration angle

absolute precision: if there is an oval deformation of one percent, the possible impreciseness for a $30mm$ marker is three time higher in comparison to a $10mm$ marker.

Frequency of tracked data: The accuracy of tracking is certainly not directly depending on the frequency of the tracking. But in case of fast marker movements and a comparably slow tracking frequency, the position of a marker can not be determined between two timestamps. This results in relatively high uncertainness of a markers current position. On the other side, if the tracking frequency is higher than needed and the markers' position is noisy, several frames can be interpolated, resulting in a higher accuracy.

Tracking algorithm: The algorithm that does the image recognition determines all pixels of one marker and calculates the 2D position of the markers center on the image by integrating over the intensity of the pixels. Depending on how this calculation is done, a different maximal accuracy is possible. Currently an algorithm is used that determines a marker's 2D position with the precision of 150 of a pixel. There also exist algorithms that enable to rise the accuracy to 180 of a pixel, though that algorithm is more costly regarding complexity and time.

2.2.2 Measuring and illustrating the influence of certain features on tracking accuracy

Before doing quantitative analysis of these criteria, it is necessary to establish methods for measuring, visualizing and comparing tracking data. The methodology is described here.

As listed in the following paragraphs it exist a large variety of parameters that affect the accuracy of tracking. Its number is too big to analyse interdependences of any combinations

of changing parameters. That is why in all measurements only one parameter is variable, while all other parameters have a value that is typical for most applications. The only exception is the parameter for the number of cameras. Changing the number of cameras has serious impact on almost all parameters. However, the vast majority of set–ups contain either two or four cameras. For this reason, most measurements are taken for two–camera systems and for four–camera systems.

The data that is necessary for the later analysis is received by a hidden log data export from A.R.T.'s DTrack program. It records information on position, orientation, and residual of each 3d and 6d marker for each time frame. For this research position and residual data are required. It should be examined how the residual changes at a given position, when the parameters are changing. A Perl Script extracts necessary data out of the log file.

The amount of received data is by far too high for recognizing certain effects by looking at the numbers. The data has to be processed and visualized. Visualization eases intuitive comprehension of huge data. For creating 2–dimensional or 3–dimensional images of the tracking area AVS/Express is used, whereas simple graphs that show dependences of two parameters are created with a typical spread sheet application.

AVS/Express requires to specify input modules, so that the data file can be read out and loaded into the program. Before the data can be visualized, in most cases further processing is necessary. The positions and residuals of points are arbitrarily distributed over the room. These are interpolated to a grid. In figure 2.2 slices though the tracking area can be seen on which the residual data is interpolated.

In order to be able to see the visualizations without AVS/Express, export modules are needed. This study uses Jpeg images for 2D data and the VRML 2.0 format for 3D representation.

Visualizing complex 3D content is sometimes confusing, since data of the interior of the tracking area are often partially covered by external data. To avoid this, movies can be created, where one slice moves through the tracking area. Two images can easily be compared and show how the spacial distribution of the residual is changing when changing parameters. Figures 2.7, 2.8, 2.9 are such slices of which movies were created.

2.3 Evaluation of criteria

As soon as ways for measuring and visualization are found, evaluation can begin. Regarding the many parameters, it was important to find out which parameters do not have any or only marginal or irrelevant influence. Those parameters can then be ignored for the later analysis. Thus, they are identified and examined at first. Other parameters are excluded from further analysis, either because it is not possible to change this parameter or because its influence can be predicted. To the first group belong camera specific items. The general precision of the camera, that is determined by its construction, its optical properties, the kind and resolution of the CCD chip used and the built in tracking algorithm, can not be modified. To the second group belongs the frequency the tracking is done. Smoothing of tracking data by interpolating time frames is not performed. With the maximum frequency of 60 Hertz it is always possible to track objects in real–time, so that changes in position are not too high between two time frames, at least for human interaction with the system.

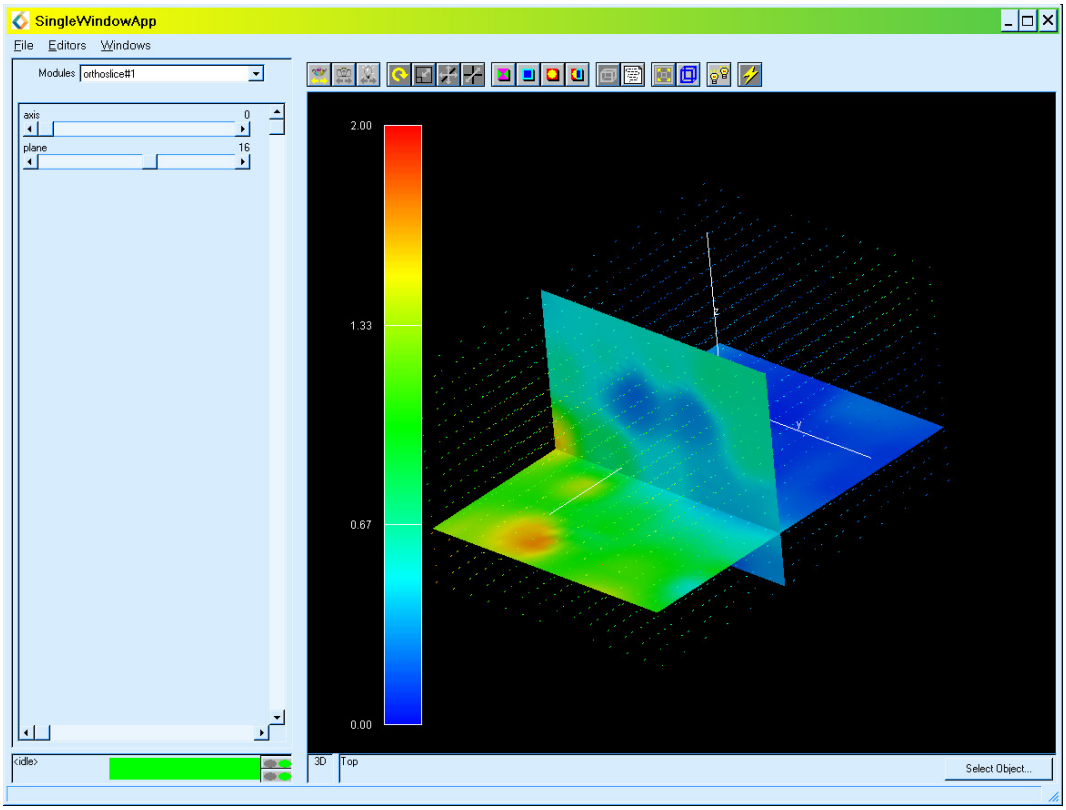


Figure 2.2: AVS/Express visualization: two slices illustrating the tracking precision

The process of separating critical from non–critical parameters is quite time consuming, as it is not clear in the beginning which parameter have an influence on which others and to what degree. That is why a much more measuring has to be done compared to what can be used finally. A lot of measurements turn out to be of no value, because the parameter to be measured is superimposed by other stronger effects.

2.3.1 Parameters with few effect on tracking accuracy

Room temperature: Whereas the heat produced internally by each camera shows some effect (described later), the differences in the room temperature do not. This can be explained by a comparatively small variance in temperature in the used AR–lab. The room temperature is always between 15 and 30 degrees centigrade. These differences seem to be too little to cause any thermal dilatation effects within the cameras or of the whole system set up.

External light: Direct sunlight can be a problem for the A.R.T. system, because it contains infrared light. Usually, the systems are set up, so that no sunlight can directly hit the cameras. But sometimes reflections can not be avoided. If only a small punctual face is reflecting sunlight, a marker could be detected by a camera even though there is not any. Furthermore, the marker position can only be computed if at least two cameras detect that point. In case

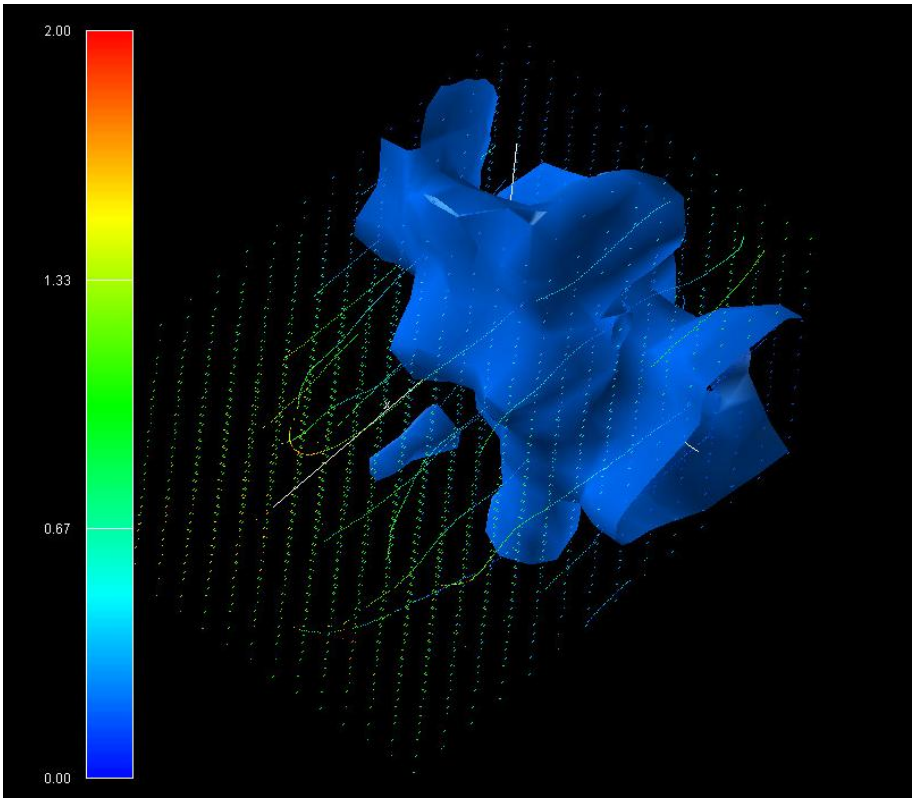


Figure 2.3: AVS/Express VRML export showing an isosurface for a residual of 0.2mm

a larger face reflects the sunlight, the algorithm in the camera detects lots of markers all over the illuminated surface. When the number of detected marker reaches its limit (this is usually between 50 and 100), the real markers could not be detected anymore. Another effect of sunlight is, that the image the camera takes becomes overexposed. Due to light dispersion not only the areas that are hit directly by sunlight are overexposed, but the whole image. The marker are not recognized any more. To sum up, a quantitative measurement of the influence of direct sunlight on tracking precision is not possible, as the cameras usually fail to work appropriately. The same effects can be found, when not sunlight but the cameras themselves dazzle each other. Usually the system set–up and the camera synchronization should prevent that.

Another question is, whether ambient sunlight has influence on tracking precision. Therefore the residuals of a few static markers were compared for the room with open and with closed roller blinds. No difference could be made out. This is presumably due to the fact that the image in infrared chip of a camera is nearly homogeneously superimposed by the sunlight, thus the differences in intensity of a marker remain nearly the same and the calculation of a markers position does not change.

Smoke: The influence of smoke on tracking accuracy could not be tested with the system. Up to now, no system has been required to work under the condition of smoke. It is expected, that the precision is not strongly influenced by light smoke. A camera's image of a marker

should not change considerably. The marker's contrast regarding the images background should be a bit lower. At the point where the marker almost disappears — because of a longer distance to the camera or a denser smoke — the tracking precision should rapidly get worse to the point where it can not be detected any more.

In case of very dense smoke too much infrared light is reflected to the camera. It may detect markers everywhere in the image and fail to operate.

Intensity of the infrared light: Each camera has six grades of intensity of its infrared light. The effect of decreasing light intensity is similar to the effect of increasing smoke density: if a marker is tracked, the tracking precision is relatively independent from the infrared light intensity. The tracking impreciseness increases only in case the illumination of a marker is close to the limit where tracking is possible. A quantitative measurement of the dependency between the light intensity and tracking preciseness is not possible, because the camera only has six built–in light intensities. For the most interesting parts of this dependency a further refinement would not be possible.

Size of markers: Different marker size is the first parameter to mention, on which quantitative measurement is done. Markers of a certain size are fixed on four different positions in the tracking area and their residual is logged. This is done for four different marker sizes. Only two cameras were used, so the 3D position of each markers is exactly determined by the two cameras. If four cameras are used, it would not be 100% sure if all cameras contributed to a marker detection or only a subset of it. As a result the value would not be comparable.

For each marker size the center of the markers has to be at exactly the same position, because the residual values might vary stronger on slightly different positions in the rooms as they do for varying marker sizes. That is why it is not possible to simply log some arbitrary points in the tracking area using one marker size, afterwards doing the same with a different marker size and finally comparing them. The residual fluctuation in space would be to high. The disadvantage of taking only a limited number of points is that the small regions in the tracking area with possibly high dependency of marker size and tracking precision, such as the borders of the trackable area, are not examined.

Figure 2.4 shows the results. The marker size is only of secondary importance regarding tracking accuracy and is superimposed by more relevant effects. The outcome is that bigger markers have a lower accuracy. This may seem a bit confusing. For bigger marker, the interpolation of the center of a marker's image should be more reliable. The problem with big markers may be that the conditions for absolute preciseness are more difficult to fulfill for a bigger marker: a slight percentual deviation from the perfect sphere leads to a higher absolute incorrectness in comparison to smaller markers. In contrast to that, the standard deviation of the residuals is higher for bigger marker, what was expected.

However, due to the few points that have been taken here, this result can be doubted. Under other circumstances, the result may be different.

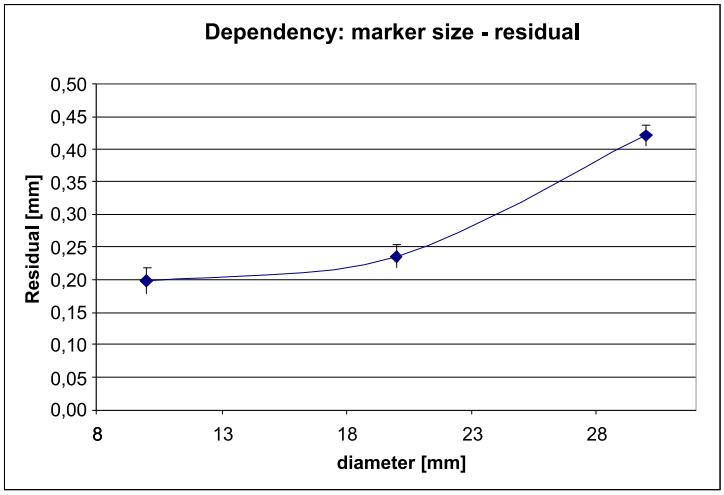


Figure 2.4: Dependency of marker size and accuracy

2.3.2 Parameters with high impact on tracking accuracy

Now that the less relevant parameters are identified, it is time to have a look at items that have influence on tracking of most typical system set-ups.

Movements: The first subject to discuss here are the little movements of components of the A.R.T system, actuated by vibrations, unintended hits against the setup or deformations of parts of the set–up due to their own gravity. Tests show that those movements are highly unpredictable. Sometimes it is possible to work with the same room calibration for three or more weeks without noticeable change of the tracking precision. Sometimes it is not even possible to work one day with one room calibration. A screw that is not fixed appropriately, a too strong touch when a camera is switched off, or a kick against the system makes the system incapable to operate. However, this is not the main problem since a new calibration has to be done anyway. The real problem is to detect, when tracking becomes so imprecise that a recalibration of the room is recommendable. This unpredictability gives a first argument for the necessity of building a system that is monitoring tracking accuracy.

Heating of cameras: That the room temperature has almost no influence on tracking precision – at least not for typical room temperatures between 15 and 30 degree centigrade – was explained before. However, the cameras themselves produce a lot of heat, because in every camera there is a PC processor and an infrared light source. When the camera is out, its temperature is that of the room. When it is switched on, the temperature of the processor and the lamp rises up to a certain level until a constant distribution of temperature is reached. The optical parts of the camera are deformed in that process. A.R.T. has already done research on that effect and has provided some data for this study (see figures 2.5 and 2.6). It was tested how many camera units the image of a marker shifts during the heating. One camera unit is about one hundredth of a pixel. The graphs show that every camera

behaves differently, when switched on. While one camera shifts about 14 units the other one shifts only 4 units maximum before almost going back to the initial value. If the camera with the highest deviation is taken, this effect is quite high: with a focal length of the camera of $4.5mm$ and the pixel distance of $7.4\mu m$, the shift of a marker's position makes up $0.7mm$ at a distance of three meters from the camera. Compared with an average tracking precision that can be less than $0.5mm$ in some parts of the tracking space, this effect surpasses others by far.

That is why cameras should only be used after at least half an hour of operation.

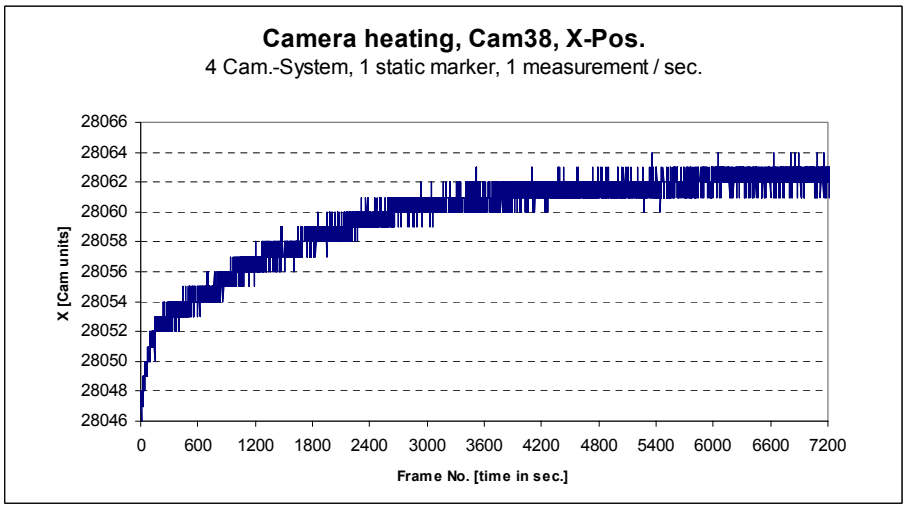


Figure 2.5: Strong effects of heat development in a camera

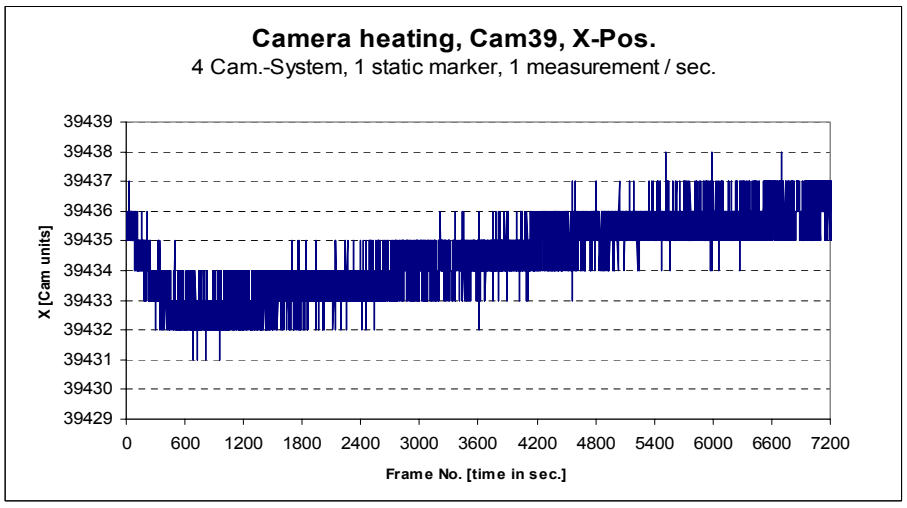


Figure 2.6: Weak effects of heat development in a camera

2.3.3 Critical parameters

Room calibration: The way the calibration of the tracking area is performed has considerable effects on the later operation, more than originally expected. In order to test this, the residual distribution of the room was measured for three different room calibrations. For the first calibration, points were collected all over the area where tracking is possible (see figure 2.7). For the second calibration (see figure 2.8), only points in the central tracking area, forming a cube of $2 \times 2 \times 2$ meters, are taken. Finally the points for the third room calibration are all in a cube of the size of one meter about one meter above the floor (see figure 2.9). Such a calibration covers the space that is typically used for desktop operations. It can be seen that the room calibration dramatically changes the residual distribution of the tracking space.

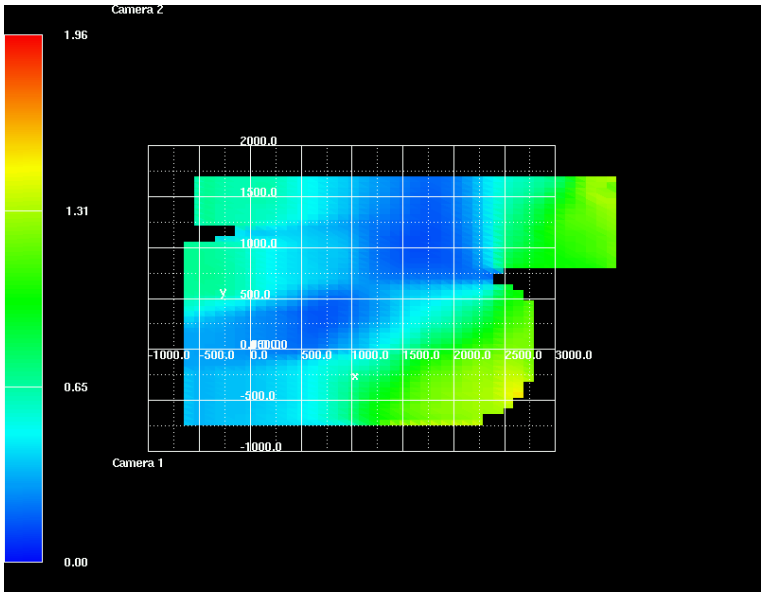


Figure 2.7: Room calibration using maximum space

For the calibration 1, the residuals do not reach optimal, but overall good values. Calibration 2 has optimal results for the parts it was calibrated for; the tracking space is shortened, and the results get worse when the distance to the central part becomes longer. When using third calibration only a limited part of the space can be used. The optimal results in the central area not better than those of the second calibration.

Number of cameras: The number of cameras used for tracking has a strong influence on the size of the tracking area as well as on the quality of tracking of some areas. Tests are done with two and four cameras. Figures 2.10 and 2.11 show the results. The set–up can remain the same all the time, the two additional cameras can simply be switched on when needed. The images compare the tracking precision in a slice through the tracking area at a height of $55cm$. For better comparison, not the whole tracking area is shown but only the parts where both configurations have detected markers.

There seems to be no general improvement when four cameras perform the tracking. It

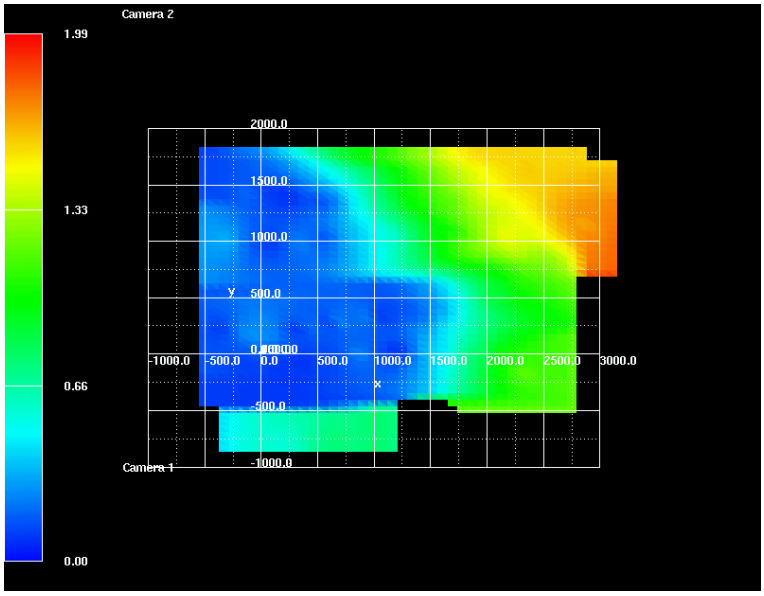


Figure 2.8: Room calibration using $2 \times 2 \times 2$ meters of space

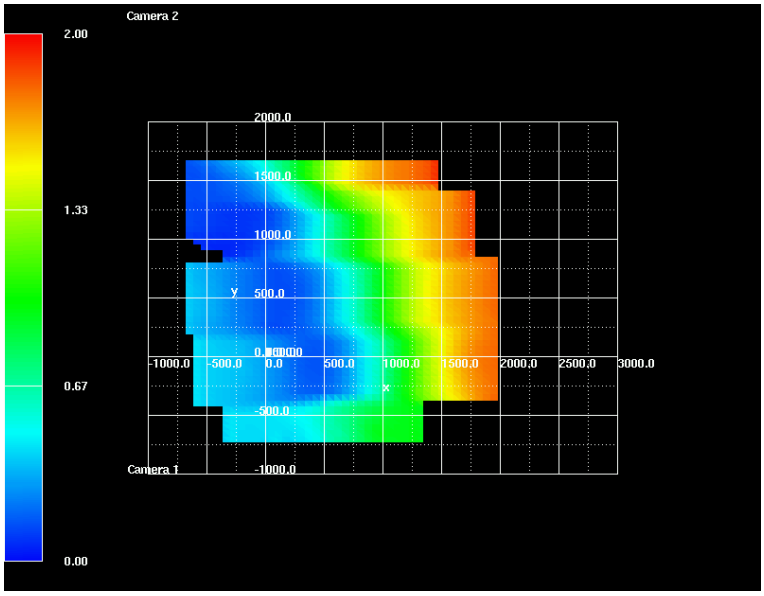


Figure 2.9: Room calibration using $1 \times 1 \times 1$ meters of space

even seems that in this measurement, the four cameras performed worse compared to the two–camera system. Maybe four cameras there are more differences in the distances between the cameras are possible. Therefore also the residuals between two cameras vary stronger, wheras perfectly calibrated two–camera systems do not face this problem. In contrast to that, the noise of the residual distribution is smoothed better in the four–camera system. This can be explained by considering that the residual is now calculated on the base

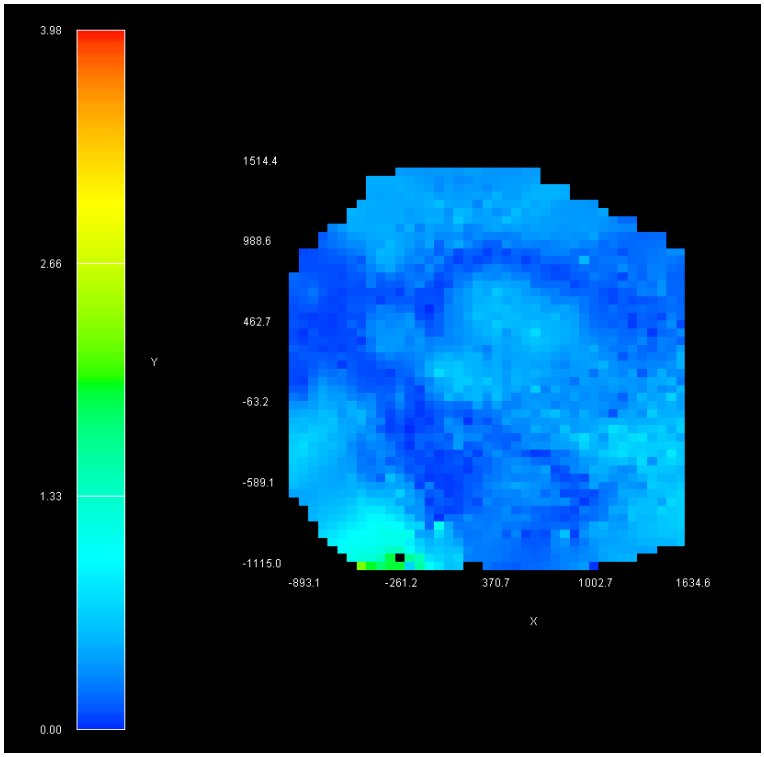


Figure 2.10: Residual distribution of a two–camera system

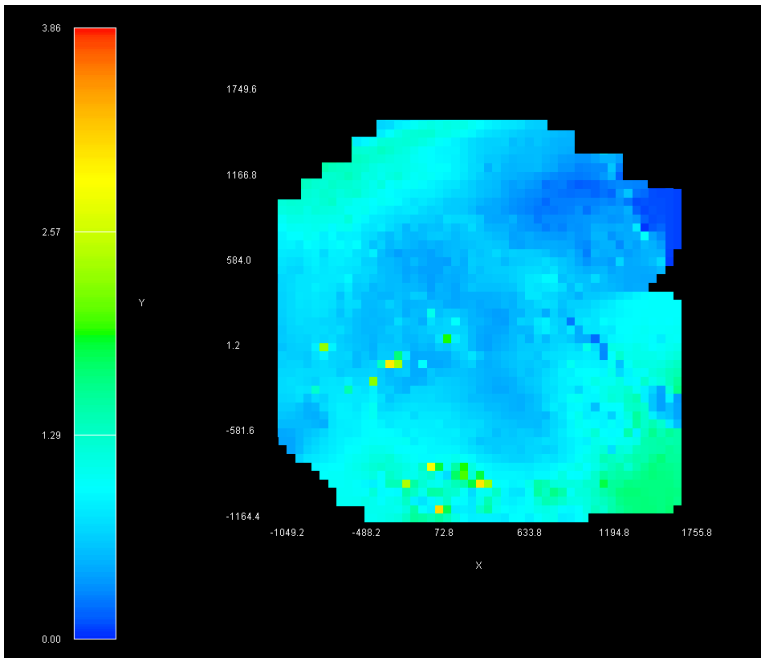


Figure 2.11: Residual distribution of a four–camera system

of four camera images instead of two.

A general difficulty in working with more than two cameras is the fact that it is not clear which cameras contributed how much to the calculation of the residual. Any subset of the set of cameras is possible. In a four–camera system, six possibilities exist in case a marker is detected by two cameras, four possibilities in case three cameras detect it, and one possibility in case all cameras track a marker. Altogether eleven possibilities for a marker to be tracked, and eleven possible residuals for that point in the room. For six cameras there are 51 possibilities. Even though DTrack logs which cameras participate for each residual value, it is unknown for how much residual deviation each camera is responsible.

In the four–camera graphic it was it was paid attention to collect only values for which the markers were recognized by all four cameras are interpolated. The few points where this did not work can be recognized well: the four–camera picture is much smoother in general, but some single points which have yellow or dark blue colour have been tracked with less than four cameras. It is clear that this distribution only shows the precision that is received under the condition that no camera fails to recognize markers. In practical use this is often not the case, since the person working with the system may block one or two cameras or a tracked object may be out of sight of a camera. The problem with measuring the residual distribution of a system with more than two camera is, that it can not easily be shown what minimal level of preciseness can be expected. This would require either that all measurements are performed for all combinations of cameras and the weakest residuals are taken or that the residuals are computed for each combination of involved cameras.

Two–camera systems do not have this problem, as always both cameras need to track a marker, otherwise its position and preciseness can not be computed.

Other effects: There are a few other things that do not influence the tracking precision of the system directly, but who are responsible for unsatisfactory precision or pseudomarkers. Sometimes it occurs that a marker constantly delivers bad residuals, whereas a different marker does show ordinary values, when placed at the same position. Reasons could be that either the marker is not round enough and shows some oval properties, or that the light reflecting material is abraded partially. The images of a marker in the cameras are not spheres any more and lead to the effect that the center of a marker is interpolated differently. So there is always an additional impreciseness.

The same effect is caused by markers that are partially covered by other objects. Then the markers image is deformed by truncation

Sometimes, pseudo–markers can be noticed. They can mostly be traced back to effects of sunlight or interfering cameras, as described under the topic "*External Light*". But sometimes it is also possible that the 3D pose estimation fails. Consider the following situation shown in figure 2.12. Two beams cross their way. Expecially if one of the markers can not be seen by the opposing camera, a marker will be detected wrongly.

2.3.4 Conclusion

There are a lot of parameters that could influence tracking precision. All parameters that were discovered have been analyzed, some of them were extensively assessed both qualita-

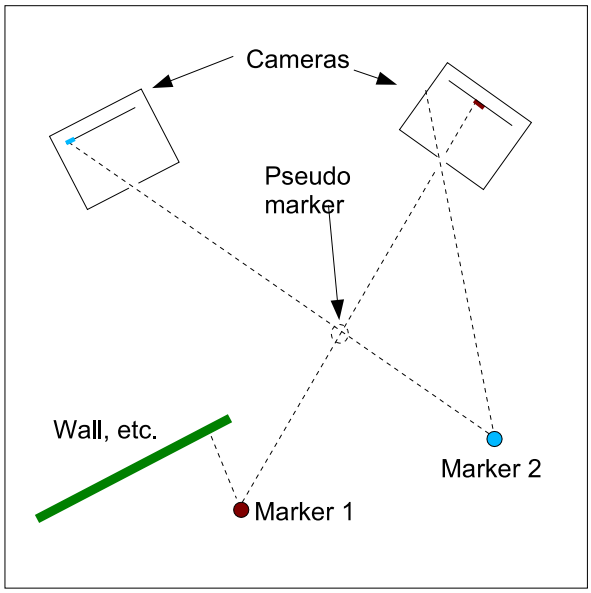


Figure 2.12: Appearance of pseudo markers

tively and quantitatively. Not all results proved to behave as expected. The size of a marker has relatively few importance regarding tracking precision, while the way of calibration has relatively high impact on it. An interesting result is the relatively complex determination of systems with more than two cameras, that closed this study. The author hopes that this knowledge can find application in other AR–projects. For the development of a monitoring software described in the next chapter it will definitely be of use.

3 Development of a Tool that Monitors Tracking Accuracy

For certain AR–applications high precision tracking is needed. Measurement engineering is an example. Medical technology is another example where the importance of augmented reality will grow. Especially in life critical areas, for example in surgery, it is not only necessary that a system provides a certain precision, but also that this precision is constantly monitored. A warning has to be given or the operation has to be stopped when certain restrictions of a tool's accuracy are violated.

Another field of application for a monitoring tool are systems which require a certain kind of smoothness. In virtual reality applications fluttering images are very disturbing. Smoothing has the disadvantage that it introduces a certain delay of time, that is unwanted especially for fast moving objects. A precision monitoring tool can not solve that problem, but it could find out where tracking impreciseness occurs that leads to that fluttering. An application could use this information and so decide in which areas tracking is smooth enough and in which additional smoothing should be done.

In the following a tool will be developed that enhances the A.R.T. system with the capability of monitoring the tracking precision (see figure 3.1). Therefore much of the knowledge obtained from the previous study can be used. One result is that the accuracy of tracking is neither constant over time nor within the tracked area. This affirms the need for an instrument that detects and monitors precision.

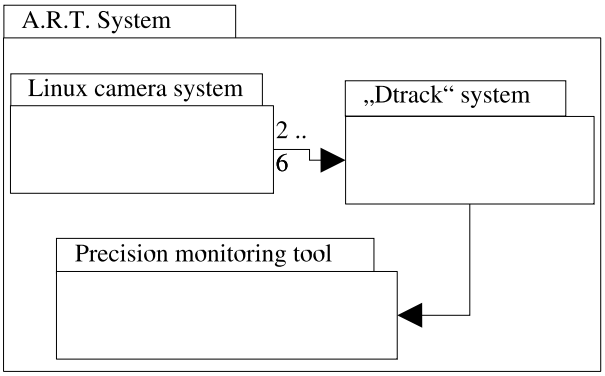


Figure 3.1: The monitoring tool as an enhancement to the A.R.T. system

3.1 Requirements analysis

As in most software projects nowadays, a structured approach is used to develop the monitoring tool. First it hast to be found out which requirements should be fulfilled by the tool.

3.1.1 Functional requirements

The requirements can be reduced to only one feature: the software should give a warning in case the tracking precision is not adequate any more. The user should specify a value that defines which degree of precision is adequate for his needs. Therefor one or two parameters can suffice.

For receiving data from DTrack a module has to be built that imports tracking data for the tool. The built-in export of tracking data can not be used, because the residual data is not included. Instead, a hidden function in DTrack that exports all raw and computed data is used. Therfore a filter is required that extracts the necessary data.

As the precision of tracking is not constant all over the tracking area, a scanning of the tracking area could be done. These values could be taken after a room calibration is done and seen as reference values. When the monitoring tool is activated it would compare those values with the current values and detect unwanted effects. A different solution to this problem is to precompute a tracking distribution solely by having some camera values. Then a mathematical model could precompute.

To compare the reference residual with a currently tracked marker's residual, the reference values have to be interpolated at the position of the marker. This interpolation could be done at runtime, or a precalculated interpolation on a grid could be used. A problem to solve is what should be done for regions in the tracking area where no reference data can be found. Sometimes it might be possible to interpolate or extrapolate those values from surrounding areas. Sometimes it might be better to not to do that, as the result will not be reliable. In this case marker's precision can not be evaluated.

It can be seen that the functional requirements are not completely fixed, and that there are several solutions that fulfill the precision monitoring requirement. This will be addressed in the next section (3.2).

3.1.2 Non–functional requirements

Firstly, the tool will be subjected to real–time constraints. It should immediately analyze all detected markers in real–time. Secondly a warning should be given only, when a certain impreciseness can be assured. A single marker that is imprecise or detected wrongly should not cause the system to give a warning. The same goes for short fluctuations of the tracking precision.

When it is necessary to do a scanning of the room, this procedure should not take more than five minutes. At first sight this seems a lot of time, but measurements from the study have shown, that it takes some time until the whole tracking area is scanned without too big voids.

Various forms are imaginable for the user interface. An expert user may want to enter details about the scanning process and about how the interpolation is done. He may want to have a visualization of the points used for the interpolation, maybe a precalculated interpolated grid. He would also specify how and when a warning should be given. The other extreme is a reduced user interface, which only offers two buttons, one for a new scanning of the room, one starting and stopping the monitoring process. In case the A.R.T. system is maintained by a service team, it would even be possible that the monitoring tool is started together with an AR application and that the user would not even notice the presence of the tool as long as everything works fine.

3.1.3 Pseudo requirements

There are also some additional requirements that mainly concern the implementation of the monitoring software. The tool's tracking data import is over a UDP port and therfore the software will also run on a different PC. So any form of implementation could be chosen. However, the following two reasons suggest platform–independent C/C++ code.

The A.R.T. system is delivered together with a PC with a Windows operating system, either Windows NT, or Windows 2000 Professional. The control and configuration of the system is done with A.R.T's DTrack software. In the medium–term a porting to linux is planned. The precision monitoring tool therefore should also work on these systems. To have a platform–independent graphical user interface would be another advantage.

Furthermore, the software has be easily integrateable with DTrack software. So if the tool's application proves to be useful, it can be delivered within DTrack. This decision also has influence on the programming language as DTrack is mainly written in C. This fact reduces the choice to C or C++.

3.2 Proposals and evaluation

The requirements could be fullfilled in different ways. For example, it is not defined if and how an initial scanning of the room is done. Such basic decisions have a considerable influence on the later architecture as well as the functionality of the tool. To find out which kind of program fulfills best the requirements, three proposal are worked out and assessed.

3.2.1 Proposal I: Mathematical or physical model of the tracking area

A mathematical model can be very reduced: a certain space could be defined, where a certain precision has to be fullfilled. If not, the system will give a warning. Also more complex model can be thought of: A camera's physical characteristics could be entered and a model of the aspected residual distribution be computed on that data. Then this is the basis from which deviation in residuals are computed.

3.2.2 Proposal II: Monitoring with reference data from a room scan

There is another possibility how the monitoring process could work: Before the tool can be used it is necessary to do a scan of the tracking room. Afterwards the scanned residuals are interpolated to a grid. When the monitoring tool is in operation, the grid points are used to calculate the expected residual of a currently tracked marker and compares it with its real residual. Here also a warning is given in case the residual exceeds a certain value dependent on the position in the room.

3.2.3 Proposal III: Monitoring with automatic recalibration

The last suggestion is the most sophisticated. The raw data provided by the cameras is used. When, in a three– or four–camera system, one camera is responsible for a relatively high residual, compared to the residual calculated without that cameras, it is possible that this camera has slightly moved. In theory it can be computed how a camera moves. In the cases where a movement of a camera's position and orientation can be detected and evaluated, a recalibration of the room could be computed. But if the a camera movement is not responsible for bad residuals, a warning could be given. In that case this proposal could also be combined with the proposals I and II: The residuals measured are compared with computed or scanned reference data. For proposal II however, a scanning of is required and therfore too big camera movements are not possible, because the residual distribution in the tracking area would change, whereas in proposal II a new distribution could be computed.

3.2.4 Evaluation

In order to find out which proposal should be realized the pros and cons for each one are named.

Simple mathematical constraints — as suggested in the first proposal — have the advantage to be easy to realize. However, also the use of such an implementation would not be very high: by setting a limit to the residual in a part of the tracking area only this upper limit can be monitored. The potential of areas with far higher accuracy can not be used. Furthermore, the tracking area can not be controlled completely. A more complex physical model of the cameras and the tracking area would provide an approximation of the residual distribution that the monitoring data is compared with. The difficulty is in building such a model. The research on tracking accuracy showed that this would be very difficult due to the many parameters which the system is dependent on. Local minima and maxima of the tracking accuracy can hardly be predicted. Also the high dependency of the residual distribution on the room calibration seems to be a problem.

The second proposal has the advantage that the software knows the tracking area's residual distribution, at least in the regions where the room was scanned. So the monitoring can be done differently for different spots of the tracking area. Each part of the room could be monitored according to its possibilities. The disadvantage is that a scanning of the room necessary. This has to be more exhausting compared to the room calibration, because data from every part of space that should be monitored later has has to be collected. No bigger free spaces should be left out. However, when the system set–up does not change, the scanned

data may be used even with a new room calibration, and so the scanning has to be done less frequently. This works only if the new room calibration covers about the same area that the old one does.

Automatic room recalibration of proposal three could make precision monitoring redundant, because bad accuracy can be traced back to camera movements. This proposal prevents a high percentage of cases where tracking becomes imprecise and a new room calibration is necessary. It would improve tracking accuracy for systems running a long time or for systems with high propability of camera movements. However there are also some concerns: first of all, at least three camera are needed to find out which camera has moved relative to two or more other cameras. When using two cameras, it can not be determined in general which camera has moved. Furthermore, it will be very complex to find out which camera could have moved in which direction. The straight lines for each camera through each marker has to be computed, as the Dtrack log data does not contain these. If it has been detected that one camera lacks accuracy, scenarios will have to be computed for the camera having moved in any direction or orientation. If a camera movement can not be detected, an approach mentioned in the other proposals needs to be used to decide about a warning. To sum up, this proposal does not really fulfill the requirements and can be more thought of an additional feature to the software. Another weakness to mention is that the absolute position of the markers can easily shift over time, because every recalibration is computed from the one before and every error sums up.

3.2.5 Decision

The decision is, to focus upon proposal II. Proposal I seems to be too imprecise in its easier variant and highly unsure to be realized in its more sophisticated form. Proposal III does not fulfill the requirements, though it can be thought of a further improvement. Proposal II seems to be a realizable and good idea and therefore it is chosen to be implemented.

3.3 Designing the monitoring tool

Now that the most important decisions on the tool's requirements have been taken, the software's architecture has to be defined by splitting it up into some logical components. Afterwards connections between the components — the components' interface — have to be specified. The last step before implementing will be defining each components' internal structure.

3.3.1 System design of the monitoring tool

The monitoring software should be an additional part of software for the A.R.T. system. Figure 3.1 shows the data flow between the camera system, the DTrack software, and the monitoring tool.

The monitoring tool itself is a component, containing five classes, that provide all necessary functions (see figure 3.2). In contrast to other software projects the tool's architecture

was strongly influenced by the pseudo requirements that were defined in the previous chapter. The fact that the tool should be integrateable with DTrack and its own graphical user interface enforced the view component to be separated from the rest of the tool. This also led to the decision to separate the UDP interface from the rest: if the tool is integrated into Dtrack, the residual data needs not to be sent over UDP and can be removed easily.

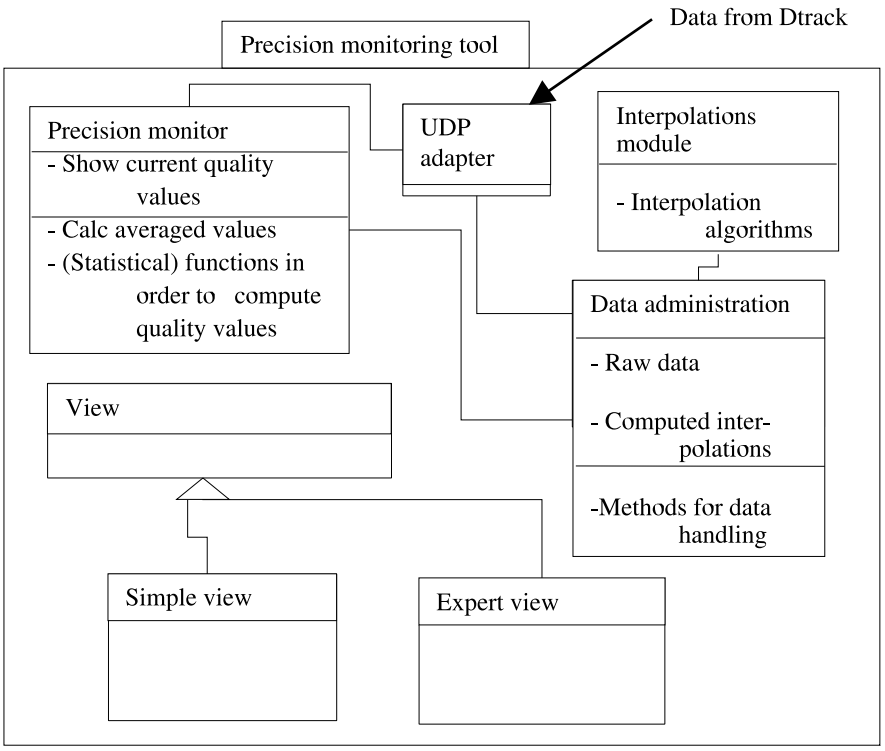


Figure 3.2: The architecture of the monitoring tool

The core part of the system is completely platform independent and results in three different modules. Here most changes and rework are done throughout the development process. The three parts are a data administration module, an interplation module and a precision monitoring module. When a new room scan is done, the data administration module receives the data from the UPD adapter and stores these as scan data that may also be visualized in the expert view. When the user presses an interpolation button the data module calls an interpolation method from the module which does its job according to predefined or customized settings. Afterwards the data administration module stores the interpolation. The interpolated data can also be visualized in the expert view. Once an interplation has been made, the system is ready for operating. The precision monitoring module is active now and gives information and warnings on tracking accuracy.

3.3.2 The components' interface design

Now that the task of each module has been defined, the interfaces have to be designed. Here a short description of the tasks that each class provides will be given. The technical

description of the classes' interface can be found in appendix B.

For the UDP adapter the interface is rather simple. It provides a function for receiving and filtering UDP data from Dtrack.

The data administration class saves the UDP module processed data as well as all some general setting, for example some filenames. It also provides methods for accessing data from the hard disk and saving data onto the hard disk. The current scan data, the current interpolation as well as status variables and information on what kind of scan and interpolation has been done and which values have been chosen for the parameters. Next to methods for accessing these data, methods for starting the room scan are provided.

The interpolation itself is done by the interpolation module. An interpolation method that can be called with various interpolation options it is accessed by the data storage component and returns the interpolated data to it.

The precision monitor's interface contains variables in which current and averaged residuals of markers are stored. Furthermore methods are provided that compare these variables with the interpolated reference data of the data administration component. The warning method of this class can be called when an insufficient accuracy is detected.

The views use the interfaces of all the described components. They display some of the information for the user and provide him with the user interface control. For the expert view a GUI should be chosen, which places all the operational controls around a central area where a visualization of tracking data is shown (see also figure A.1 in the appendix). User interface studies have shown that this arrangement is superior to others where all features are combined in one block.

3.4 Implementation

Now some important implementation decisions are mentioned that had major impact on the system. A complete description of the implementation can not be given within the scope of this work.

- The programming language C/C++ has been chosen, because DTrack is mainly written in native C which it should integrateable with. The modules are all implemented as C++ classes and sometimes contain native C code.
- For the user interface different alternatives can be chosen. The simple user view will be written in QT, and could be a prototype for a future application of a monitoring tool in the industry. The more sophisticated expert view requires OpenGl compliant hardware acceleration and enables full control over all functions that the tool provides. The GLUI (see [16]) library is used because of four reasons: it is free source code, it is platform independent, written in C++ and it enables 3D visualization as it is based on OpenGl (see [18]) and GLUT (see [18] and [17]).
- A task during implementation is to decide which parts should be written newly and where existing source code could be used. For the UDP adapter parts of a library from A.R.T. could be used that contain functions and data types that ease the access to the UDP interface of A.R.T. Some functions had to be modified or added though.

- For the data storage the usual C/C++ libraries are used to access and store files. The access to the files has to be coded by hand as GLUI does not provide graphical elements for file access.

- Most own code is produced for implementing the data administration, the interpolation and the precision monitoring tools. The grid interpolation of the `Interpolation` class will be described shortly.

  It was evaluated if interpolation methods from existing libraries could be used. But nothing that would fit to this special interpolation problem was found and therefore own algorithms were implemented. The interpolation grid forms a bounding box around the scanned data. This box is computed first. The grid divides this box into cells, which are numbered. Next the scanned data is sorted by ascending cell number which they belong to. This provides fast access when data of a specific region of the tracking area is requested, without searching in the list of scanned points each time. All these data is stored and accessed by pointers. The real–time requirements for the monitoring tool can be fulfilled this way. A grid point's residual is computed by interpolating all points which are found in the eight cells, the grid point is a vertex of. But only if more than four of these cells contain points. A Gaussian function with $\mathcal{O}(e^{-x^2})$ is used therefore, where x is the distance from a scanned point to the grid point. This method yields much better results than bilinear interpolation, which was also implemented, but then discarded.

4 The Future of this Work

4.1 To–dos and enhancements to the monitoring tool

Most things described in the development process could be implemented successfully. There are working demos for SUSE LINUX 8.1/8.2 and Windows 2000/XP. Some small bugs have to be fixed. Furthermore enhancements are planned. The following tasks are foreseen to be done for the near future:

Rename classes and variables: During this work some central terms came out to be misleading for others and were replaced. For example the term "*tracking quality*" was substituted by "*tracking precision*" or "*tracking accuracy*". The procedure of scanning the room and doing an interpolation was called "*calibration*" before. In the Dtrack software however, a room calibration exists. Those two things were often mixed up or unclear, when presented to others. The new names are both more precise in their meaning and nearer to the application domain. Throughout this document the "*new*" terms were used. In the source code however, the deprecated terms are still in use, due to some dependencies between the classes. This will be changed next.

Fix monitoring bug: The tool works well, but the values for the current and averaged precision as well as the precision limits on the tool's main panel are not updated (The main panel is described in the manual of the precision monitoring tool in appendix A). Tests show that the values are computed correctly but not displayed.

Update list of scan data files: When new scan data is saved with a name, the file list is not updated. Only when the tool is restarted, the list is correct. This is due to the fact, that the GLUI library does not provide a method for dynamically configurable menus. This problem will remain, or a different solution has to be found.

Delete redundant code: The source code contains a lot of unused codelines that was created during the development process. Some parts contain useful lines for fixing bugs and implementing improvements. The rest will be cleaned.

Create simple view: For now the GUI resembles to the expert view. Though this is more sophisticated compared to the simple view, it had to be implemented first, because it can be used better for testing. The reduced simple view, which provides almost no information to the user but works rather as a background job, will be developed once the first testing phase will have been finished.

Implementation of the "`fillHoles()`" method: If a marker is monitored, usually it is within a cell of the interpolation grid. The marker is surrounded by the eight vertices of this cell. The reference value to the residuals of the marker is evaluated using the same Gaussian interpolation that is used for computing the interpolation grid from the scan data. In two cases a reference value can not be computed (see figure 4.1):

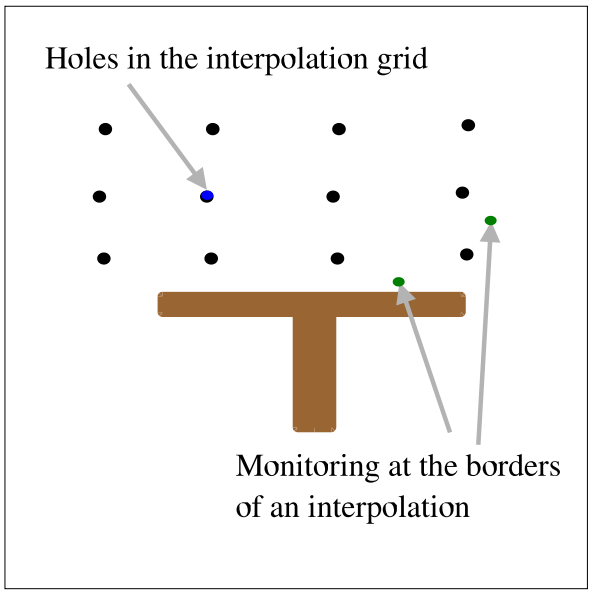


Figure 4.1: Holes and borders of an interpolation

In the first case some parts of the tracking area are left out unintentionally during the scanning. Holes in the grid are the consequence thereof. In the second case, an interpolation can not be calculated, because a bigger object, for example a table, can not be removed during the room scan. Then a marker that is close to that object can not be monitored, because the marker is outside of the interpolation grid. The problem here is that an extrapolation of the grid data has to be done in order to estimate a reference value. This is critical however, since the residual distribution is hardly predictable, as the previous study showed. For the first case instead, an interpolation from the surrounding areas can be done. The function "`fillHoles()`" should implement a solution to at least the first case. This will be one of the next tasks in improving the monitoring tool.

4.2 Testing the implementation

In the weeks after termination of this document the monitoring tool will be made ready for the first field tests. An A.R.T. system at the A.R.T. GmbH will be installed with a running monitoring tool. Some month of usage should bring enough data for deciding about a possible delivery of the tool together with the Dtrack software. The following questions have to be answered:

- Does the program run stable over some days?

- Does it detect poor tracking accuracy reliably?
- How frequent do false alarms occur?
- Can the real–time requirements be met also with many markers?
- Is the tool usable for inexperienced operators?
- How many days does it take on average before a new room scan is necessary?

4.3 Next steps

If the tests at A.R.T. GmbH are successful, the monitoring tool can be distributed together with Dtrack to a few selected clients, who can test and use the software. In case the clients are satisfied and other customers are interested, an integration directly into Dtrack can be thought of. A further advantage of an integration is that the room calibration of Dtrack and the data scan of the monitoring could be combined to one user task what makes the software more userfriendly.

Dynamic recalibration of the A.R.T. system, that has been described as "*Proposal II*" in chapter 3, could be realized as an enhancement to the existing software, but much rework would be required. Although discarded for beeing too difficult to realize and for not fulfilling the requirements, dynamic recalibration of tracking cameras has considerable benefits. Such a software would be useful in virtual reality. VR often faces the problem that objects are skipping arbitrarily. When the cameras of a VR system are not calibrated optimally the coordinates for the same object vary, depending on the camera. When such an object moves out of sight of a camera, or in the sight of a camera, the interpolated position makes a jump. By detecting even small camera shifts, the disturbing effect of skipping could be avoided.

4.4 Outlook

Augmented reality is a field that finds more and more use in laboratories as well as in the industry. Whereas initial AR applications are often used for prototyping, it is only a question of time until augmented reality conquers the everyday life of industry workers. When used in assembly lines, a monitoring tool like the one developed here could care that a predefined accuracy is maintained and therefore it would form a part of the quality assurance of a company. For the precision of a robot arm could be monitored. A different field of application for augmented reality is medical surgery. There is an increasing interest in this technology, because parts of the body that can not be seen, for example organs or arteries, can be visualized. Guaranteeing a high degree of precision is vital in the medical sector, so there may be some interest in precision monitoring. Even more, the study on tracking accuracy may help developers of medical software to better assess risks and chances of optical tracking.

If the monitoring tool was enhanced with the ability of automatic recalibration, there may be other fields, where the software could be deployed successfully. This feature could boost the number of possibilities, because it would also be of use for virtual reality applications.

The author hopes that at least one of the mentioned scenarios for a monitoring tool can find some usage in the future and that the study on tracking accuracy may help to produce new software for precision sensitive augmented reality applications.

Appendix

A Manual for the Precision Monitor

This manual gives a short introduction to the monitoring tool delivered together with A.R.T.'s DTrack program. The tool enables the user to monitor the tracking precision of his A.R.T. system. If the accuracy of the tracking gets worse over time, the tool gives a warning. Before the monitoring process can start, reference data has to be genereated. Therfore a room scan has to be done for the area that is intended to be monitored. After this, an interpolation grid has to be calculated, that provides the reference data. Once this is done, the monitoring can be started.

A.1 Installation of the program

The tool was tested to be running on SUSE LINUX 8.1 and SUSE LINUX 8.2, Windows 2000 Professional, Windows Server 2003 and Windows XP Professional. The directories containing the Linux or Windows program simply needs to be copied to the hard disk and can be started directly. OpenGl has to be supported by the graphic card and activated in the operating system. No further installation procedures are required. The program can be started directly by double clicking on the executable program in the folder icon or by executing the file on the command line.

Please make sure that port 5000 is opened for receiving data from a different PC where DTrack is running. Also make sure the log data from DTrack is activated and sent to the correct computer. The logging of data is not documented in the DTrack manual, so please contact A.R.T. for further information or use the copy of DTrack provided with the monitoring tool, where these options are activated for DTrack and the monitoring tool running on the same PC.

A.2 Description of the user interface

The monitoring tool window consists of a visualization field at the center with panels grouped around it (see figure A.1). With exception of the main panel all panels can be hidden. Then only the necessary data is displayed.

A.2.1 The visualization field

In the central there is a visualization field, showing information from the tracking area. Some test data are already preloaded so that some scanned room data and an interpolation grid can be seen.

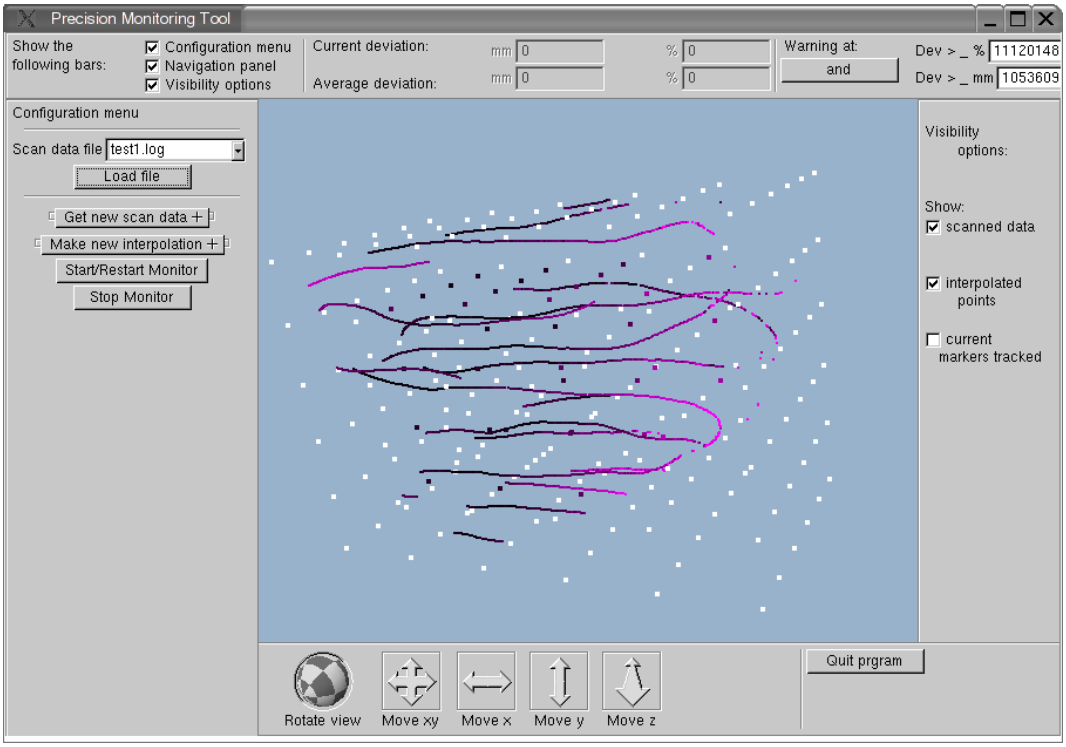


Figure A.1: User interface of the precision monitoring tool

A.2.2 The main panel

The main panel on the top is always visible. It has checkboxes for switching on and off the other panels. Furthermore data on the current state of accuracy is displayed. "*Current Deviation*" shows the average deviation of the residuals of the currently tracked markers in millimeters as well as in percentage, whereas "*Average Deviation*" indicates the state of accuracy averaged over some time and over some marker in different places of the tracking area. These values cause the system to raise an alarm, if they go over certain limit. A perceptual and an absolute limit of the allowed deviation can be entered in the fields on the right of the panel.

A.2.3 The scanning and interpolation panel

The configuration panel on the left is used for loading previous scans, making new scans, for calculating interpolations, and for starting and stopping the monitoring.

The pulldown list on the top contains up to five previous scans. One can be selected at a time. The data can be loaded with by pressing the "*Load file*" button. For making a new scan, the pulldown window "*Get new scan data*" has to be opened. This is done by clicking the '+' symbol on the right of the pulldown window (see figure A.2). In the following fields, some properties can be specified: The first one says on which of the five fields the scan should be saved, the second one contains the name the scan should have and the third field finally,

displays the time that is needed to make the scan. Previous scan data at the specified field will be deleted. Once the fields are filled appropriately, the start button can be pressed. After doing so, the scan starts. The GUI freezes during the scan and is reactivated once the scan is over.

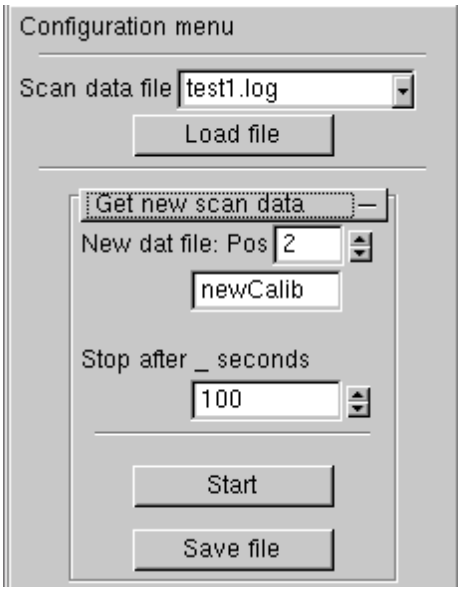


Figure A.2: The scanning window

For the parts of the tracking area that should be monitored, scan data has to be collected. This is best done with a self constructed bar with at least four markers. Something similar to figure A.3 which can be turned fast around its axis has proved to work well. Keep attention to cover every peace of the space intended to monitor and not to leave greater vacancies.

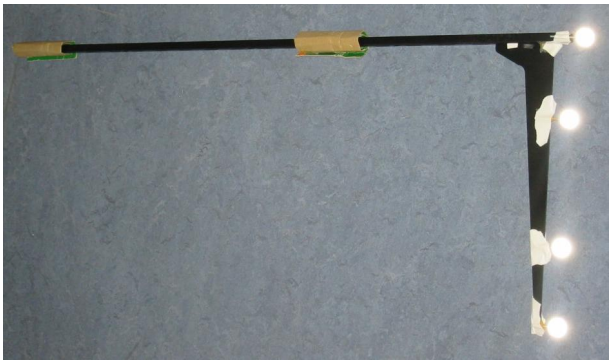


Figure A.3: A bar for scanning the tracking area fast and efficiently

Now a grid interpolation can be done. A further pulldown window can be opened therefore A.4). The position of the grid is defined by bounding box around all scanned points. The number of cell in each direction of the grid can be specified. For grid points, where no or not enough data is scanned to compute the residual value of the grid point, no value is

computed. For now other options are grey out. Options for filling grid points with missing data will be offered. Finally, by clicking at "*Calc interpolation*" the interpolation is computed.

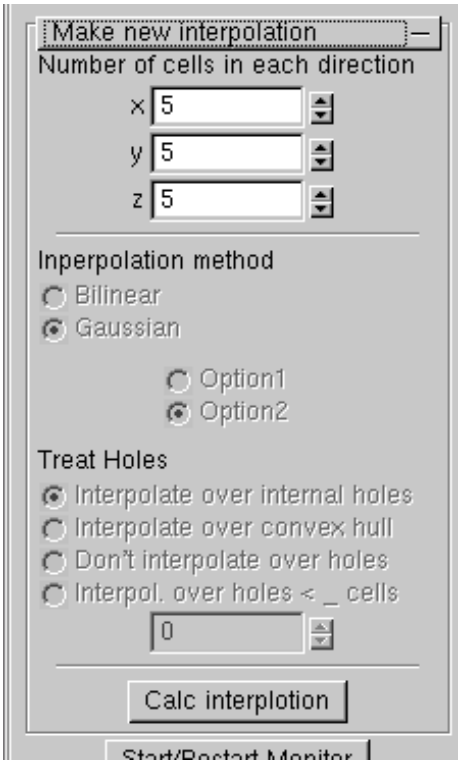


Figure A.4: The interpolation window

Now the system is ready for starting the monitor. When the start button is pressed, the current markers are displayed. The status values of the top panel will be updated continuously. The monitor can be stopped manually or it is stopped automatically in case the average residual deviation becomes too high.

A.2.4 The visibility panel

In this panel it can be chosen what is actually displayed in the visualization field. Three checkboxes enable to see the scanned data, the grid points and the currently tracked points. The colour of the points in the visualization field indicates the accuracy. Black means that there is a relatively high precision, whereas red means less precision. Where no interpolation was possible, grid points are white.

A.2.5 The navigation panel

The navigation panel enables the user to move around in the virtual tracking area. There is a trackball which makes the tracking area rotate, as well as arrows for moving the tracking area into certain directions.

B API of Reusable Components

One of the requirements for the monitoring tool is, that it can easily be integrated with DTrack. That is why the architecture was split up in four parts — C++ classes — that can be reused and one part — the view — that is not likely to be reused, because the other parts might be integrated directly into DTrack. The interfaces of the four classes are listed and shortly described here. Private attributes and operations are left out. It is not necessary to know or change the classes' source code. The following provides merely technical information for the programmer. More information on the design process and the rationale can be found at the section 3.3 and the following.

Another note: the interface of the classes in the source code contains additional public variables and functions that are not mentioned here. In most cases this is deprecated legacy code that has not been deleted, because it can be used for testing and because it contains valuable source code that may be reused. Some functions that are used for testing only are also not treated here. Other functions not named have not or not fully been implemented yet or will be made private in the future.

The `main()` method of the `View` component of the tool's source code gives an example, how the four classes can be instantiated and how they work together. A look at the first few lines of this method should answer most questions on that.

Common for all four classes is that all pointers to float variables (`float*`) point at a series of markers. A marker is described by its position and its residual. The pointer itself points at the x–value of the first marker and all markers values follow the this scheme:

index	x value of marker 1
[0]	x value of marker 1
[1]	y value of marker 1
[2]	z value of marker 1
[3]	residual of marker 1
[4]	x value of marker 2
[...]	...

B.1 UPD module

The `UdpAdapter` is a class that receives data from the UDP port. After initializing a new `UdpAdapter` with `initAdapter(int port)`, where `port` is the port the data is received from, the data that is currently sent from DTrack can be received with the `getNextFrame()` method. Information on the time frame number (`framenr`), the number of currently received markers (`nmarkers`; not necessarily all currently tracked markers, as it is possible that the information of one time frame is split), and the `markers` are stored in the variables

of the class. `stopAdapter()` breaks the connection to the UDP port, `deleteAdapter` deletes the `UdpAdapter` object.

```
class UdpAdapter{

public:
    unsigned long framenr[1];
    int nmarkers[1];
    float* markers;

    UdpAdapter();
    void initAdapter(int port);
    void stopAdapter();
    void deleteAdapter();
    int getNextFrame();

};
```

B.2 Data administration module

The class `DataModule` administers and provides the data that is used by the other components. `fileList` contains filenames of previous room scans. These names are saved in the file `filenames` in the directory where the program binary is located. The filenames can be loaded into `fileList` with `loadFilenames()` and saved into a file with `saveFilenemes()`.

When a new scan is made `getRawData(*udp, seconds)` is called. Then the class recives data from the `UdpAdapter *udp` for the specified number of `seconds`. The scan data is stored in `rawData`, its size in `rawDataSize`. The data can be saved to a file with `saveRawData(*filename)`. `loadRawData(*filename)` tries to load `rawData` and `rawDataSize` from a file with a specified name. Note that the programmer has to care for consistency with the `fileList` variable.

Data describing the interpolated grid is stored in the following variables: `resolution`[3] contains the number of grid cells in each direction of the cube. The field indices 0, 1, and 2 correspond to the x, y, and z axes. `boundPoints`[6] contains defines the bounding box where the interpolation is made in coordinates of the tracking area. The order is: *minimal x, maximal x, minimal y, maximal y, minimal z, maximal z*. `interpolData` finally contains all grid points of the interpolation. The index starts with the minimal x, y, and z coordinates and increases first in x, then in y, and finally in z.

```
class DataModule{

public:
```

```
    char *fileName;
    float *rawData;
    int rawDataSize[1];
    char **fileList;

    float *interpolData;
    float boundPoints[6];
    int resolution[3];

    DataModule();

    void loadRawData(char *filename);
    void saveRawData(char *filename);
    void getRawData(UdpAdapter *udp, int seconds);
    void loadFilenames();
    void saveFilenames();

};
```

B.3 Interpolation module

The `Interpolation` class contains all parameters that are necessary for the interpolation as well as the interpolation method. The pointers and arrays `rawData` `rawDataSize`, `interpolData`, `interpolDataSize`, `borders` (corresponds with `boundPoints`), and `resolution` point at the same memory address as the corresponding variables in the `DataModule` class and are introduced for a more convenient notation. The constructor `Interpolation(float *data, int *size)` has to be initialized with corresponding pointers to the `rawData` in the `DataModule`.

After initialisation, `findBoundaries()` finds the bounding box explained in the `DataModule` class description above, and stores the values in `borders` and `DataModule->boundPoints` respectively. Next `makeSortedList()` has to be called. It sorts the `rawData` in order to optimize the later interpolation. This method will be moved or encapsuled in `makeVarInterpolation()` later and be made private. For now it has to be called before executing `makeVarInterpolation()`, which finally completes the interpolation and stores the data in `interpolData`. It is also planned to implement the `fillHoles()` method. This should interpolate over isle in the tracking area, where no data is available. The maximum size such an isle can have is specified in `holesFillSize`. `interpolData` will be updated with the newly interpolated data.

```
class Interpolation{
public:
    float *rawData;
    int rawDataSize[1];
    float *interpolData;
    float interpolDataSize[1];
```

```
    float borders[6];
    int resolution[3];

    int holesFillSize[1];

    Interpolation(float *data, int *size);
    void makeSortedList();
    void makeVarInterpolation();
    void fillHoles();

    void findBoundaries();

};
```

B.4 Precision monitoring module

Most time of the project the precision monitoring module was called quality monitoring module. As the word quality is quite general, misunderstandings may occur. Precision or accuracy seem to be more apposite. That is why it was decided to rename quality monitoring into precision monitoring or accuracy monitoring in this work. Because of some dependences within the software, the word quality is still used in some classes. This will be changed in the future.

The class `Quality` monitors the currently received marker data and compares them with the interpolation. `Quality` is instantiated with a `UdpAdapter` and an `Interpolation` object as argument. The system design proposes a connection to a `DataModule` object instead of an `Interpolation` object, because it is the data that is needed by `Quality`. However, a function out of the `Interpolation` class can be reused. This function interpolates the residual of a given grid point from of a list of points. In the `Quality` class the function is used to interpolate the residual of a currently tracked marker out of a list of grid points.

The values `limitmm` and `limitPerc` define the difference in millimeters and in percent that the average residual is not allowed to exceed compared to the interpolated residual from the room scan. Once these value have been specified, the `init()` function starts the precision monitoring process.

Every time the `getNextFrame()` method of the `UdpAdapter` is called, it returns the currently tracked markers. By calling `updateQuality(float *markers, int *nmarkers)`, the state of the following variables is updated: `currDevmm` and `currDevPerc` are the averaged deviations of the currently measured residuals from the room scan values. `avDevmm` and `avDevPerc` are the average residual differences additionally averaged over some time frames. They are permanently compared with `limitmm` and `limitPerc`.

The `warning` variable is set by an algorithm that is written specifically for that purpose. An exceeding of the limits needs to occur in different periods of time, and at different places in the tracking area. When the sum of such exceeding reaches a certain limit, the `warning` is set `true`. In this case, the function `giveWarning()` can be called. The function is specific

to an AR–application and the programmer should write code what to do in case a warning occurs.

```
class Quality{

public:
    float limitmm;
    float limitPerc;
    float currDevmm;
    float currDevPerc;
    float avDevmm;
    float avDevPerc;
    bool  warning;

    Quality(UdpAdapter *udpAdapter, Interpolation *interpolation);
    void init();
    void giveWarning();
    void updateQuality(float *markers, int *nmarkers);

};
```

C Glossary

Note: Words in SMALL CAPS style refer to other keywords in the glossary

3DOF Abbreviation for "*3 Degrees Of Freedom*"; *see* section 1.2, paragraph "*Markers*".

6DOF Abbreviation for "*6 Degrees Of Freedom*"; *see* section 1.2, paragraph "*Markers*".

API Abbreviation for "*Application Programmer Interface*". A well–defined standardized interface that can be used by application programmers.

A.R.T. Abbreviation for "*Advanced Realtime Tracking*"; short form for the A.R.T. GmbH.

A.R.T. system An infrared–optical tracking system by the A.R.T. GmbH.

ARTtrack1 Infrared–optical camera by A.R.T. used for tracking.

Attribute In the context of object oriented programming: Specifying the properties of an "OBJECT".

Augmented Reality (AR) A human's visual perception is augmented by virtual objects. *See* section 1.2 for more detailed information.

Body In an A.R.T. system: group of four or more 3DOF markers forming a 6DOF marker. *See* also section 1.2, paragraph "*Markers*".

C/C++ One of the most popular programming language families available for nearly all platforms; C is an imperative programming; C++ additionally provides a class concept.

CCD chip Light sensitive chip for taking photos. Used in digital cameras, telescopes, etc.

Class Concept of object oriented programming; an abstraction of a set of objects; a class contains "ATTRIBUTES" and "OPERATIONS".

DTrack Software developed by the A.R.T. GmbH and provided with an A.R.T. system.

GLUI A "*GLut-based C++ User Interface* " for facitlitating OPENGL programming; provides platform independent file-, control-, and window handling

GLUT An "*open**GL** **U**tility **T**oolkit*" for facitlitating OpenGl programming; provides platform independent file-, control-, and window handling.

GUI Abbreviation for "*Graphical User Interface*", the part of the program, that is visible by the user and that the user interacts witch.

Infrared–optical Tracking "OPTICAL TRACKING" using infrared light.

Jpeg A format for saving and compressing images. The compression can be chosen freely and leads to some loss of information, the higher the compression is.

Marker In optical tracking: Special object, whose position and/or orientation can be determined by a camera. *See* also section 1.2.

Method In the context of object oriented programming: synonym for "OPERATIONS".

Object Instantiation of a "CLASS" with "ATTRIBUTES" and "OPERATIONS".

OpenGl An "***Open G**raphics **L**ibrary*" that facilitates hardware accelerated graphic programming.

Operation In the context of object oriented programming: Specifying the behaviour of an "OBJECT".

Optical Tracking Method for recognizing and tracking objects, using a camera and image recognition techniques.

Private In the context of object oriented programming: "ATTRIBUTES" and "OPERATIONS" can be private, meaning that they can not be accessed by other objects.

Public In the context of object oriented programming: "ATTRIBUTES" and "OPERATIONS" can be public, meaning that every object can access them.

QT Platform independent windows toolkit that is free for Linux and commercial for Windows systems.

Residual Minimal distance between two skew straight lines. *See* also section 1.2.

SUSE Short for SUSE LINUX AG. Major linux distributor. The monitoring tool was tested on SUSE systems 8.1 and 8.2.

Triangulation Geometrical method for determining an object's position or orientation by computing the unknown edges or angles of a determined triangle. *See* also section 1.2.

Virtual Reality (VR) Artificial worlds generated by a computer, in which the user can interact.

UDP Abbreviation for "*User Datagram Protokol*".

VRML Virtual Reality Markup Language. A language for textually describing 3D content, that can be viewed with a browser. Used for visualisation in this work.

Bibliography

[1] A.R.T GMBH, *Homepage of A.R.T. http://www.ar–tracking.de*.

[2] A.R.T. GMBH, *ARTtrack1 Dtrack — Manual Version 1.18*, 2002. Available at *http://www.ar–tracking.de*.

[3] A.R.T. GMBH, *ARTtrack1 Dtrack für Windows NT 4.0 und Windows 2000 Professional — Handbuch Version 1.18.0*, 2002. Available at *http://www.ar–tracking.de*.

[4] ARTOOLKIT, *ARToolKit and Download and Page*.

[5] ARVIKA CONSORTIUM, *Homepage of the ARVIKA project*. *http://www.augmented–reality.de*.

[6] B. BRÜGGE and A. H. DUTOIT, *Object–Oriented Software Engineering. Conquering Complex and Changing Systems*, Prentice Hall, Upper Saddle River, NJ, 2000.

[7] DUDEN, *Duden Informatik — Ein Fachlexikon für Studium und Praxis*, Dudenverlag, Mannheim, 2001.

[8] O. FAUGERAS, *Three–Dimensional Computer Vision*, The MIT Press, Cambridge, Mass., 1993.

[9] KDE E.V, KDEVELOP PROJECT, *Homepage of KDevelop. http://www.kdevelop.org/*.

[10] G. KLINKER, *Augmented Reality Page of Prof. Grudrun Klinker, Ph.D.* *http://wwwbruegge.in.tum.de/projects/lehrstuhl/twiki/bin/view/DWARF/AugmentedReality*.

[11] LEO DICTIONARY TEAM AT THE DEPARTMENT OF COMP. SC., TECHNISCHE UNIVERSITÄT MÜNCHEN, *LEO — a german–english online dictionary. http://dict.leo.org*.

[12] MICROSOFT CORCOPRATION, *MSDN library — C++ Language Reference*. *http://msdn.microsoft.com/library/default.asp?url=/library/en–us/vccelng/htm/plintro.asp*.

[13] Y. OHTA and H. TAMURA, *Mixed Reality — Merging Real and Virtual Worlds*, Ohmsha, Ltd. Springer–Verlag, Tokyo, Japan, 1999.

[14] OPENGL ARCHITECTURE REVIEW BOARD, M. WOO, J. NEIDER, T. DAVIS, and D. SHREINER, *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*, Addison–Wesley, Reading, Massachusetts, 3rd ed., 1999.

[15] P. RADEMACHER, *The GLUI homepage. http://www.cs.unc.edu/∼rademach/glui/*.

[16] P. RADEMACHER, *GLUI — A GLUT–Based User Interface Library*, version 2.0 ed., June 1999. Available at *http://www.cs.unc.edu/∼rademach/glui/*.

[17] N. ROBINS, *Homepage of GLUT for Win32. http://www.xmission.com/~nate/glut.html*.

[18] SILICON GRAPHICS INC., *OpenGL hompage. http://www.opengl.org*.

[19] B. STROUSTRUP, *Die C++ Programmiersprache*, Addison–Wesley, Boston, Mass., 2000.

[20] SUSE LINUX AG, *Homepage of SUSE. http://www.suse.de*.

[21] WEBNOX CORPORATION, *hyperdictionary — an online dictionary*. *http://www.hyperdictionary.com*.