University of Twente
Computer Science
Human Media Interaction
Enschede, Overijssel, The Netherlands

University of Otago
Department of Computer Science
Graphics and Vision Research Laboratory
Dunedin, Otago, New Zealand

Body pose tracking in the Watching Window.

A system that tracks both hands in a virtual reality environment.

K. G. F. Herms

January 2007

Examination committee:
dr. D.K.J. Heylen
ir. R. Poppe
prof. dr. ir. A. Nijholt
dr. B. McCane

Abstract

This thesis is written for the closure of the master course Human Media Interaction at the University of Twente. This thesis describes a research project related to the Watching Window (WW) project of the University of Otago.

The WW is a Virtual Reality (VR) environment in which the user can interact with a VR application without the use of body suits, markers or gloves. To achieve this interaction, the user's head is tracked by several cameras and these head coordination's are triangulated to estimate the coordination in 3D world space. With this 3D coordination, the right perspective of a VR application is drawn by a projector on a screen. This enables the user to see the 3D scenery in the right perspective continually, giving the user the right motion parallax while the user moves around. With the help of some stereo glasses, the user can even experience a 3D effect. The current applications only use the motion parallax and 3D effect while more interesting and challenging applications are possible if the user can physically interact with the VR world. This thesis describes a system that tracks both hands of the user making more interaction with the VR world possible.

A hand-tracking module is presented that uses all available cameras to estimate the current upper body pose of the user. Both hands are derived from the estimated pose and used in the WW applications. The body pose estimation technique used in the module is based on a model based pose estimation technique and a particle filter is used to track the body pose of the user. A 3D model is used that can be projected on every camera observation making a generic evaluation possible. This enables the use of all possible cameras for the evaluation which makes more information available and solves camera observation ambiguity. The dimensionality of the model is reduced by decomposing the model into separate body-part estimation problems. Each of these body part estimation problems is solved by a model based particle filter.

The hand-tracking module is evaluated by 2 evaluation tests namely an accuracy and a performance evaluation test. From these evaluation tests it can be concluded that the hand-tracking module proposed in this thesis, works well enough for pointing and manipulating objects. However, for detailed hand tracking and small movement the method described in this thesis is not sufficient. Some future work proposes the redesign of the image feature, the projection of the particles and an addition to the model to enable more detailed tracking.

Acknowledgements

During my Human Media Interaction study at the University of Twente I came in contact with the University of Otago, in Dunedin New Zealand. For the closure of HMI a few interesting research projects at the University of Otago where presented. From all of these assignments the hand tracking system in the Watching Window was the most motivating and appealing.

I want to thank the University of Otago for the opportunity to be part in such an interesting and motivating project. I would like to thank my supervisor from Otago University, Brendan Mccane, and from the Netherlands Dirk Heylen, Anton Nijholt and Ronald Poppe in particular. I also want to thank Geoff Wyvill for his help and fascinating ideas and Damon Simpson, Phil Mccleod and Sui-Ling Ming Wong for their help and support.

I also want to thank the other exchange students that made Dunedin a perfect place to study and of course my family and friends for their support and understanding for my seven months of absence.

Koen Herms,

Huizen, January-2007

Table of Contents

List of Figures

Part I Introduction

Part I gives an introduction to the thesis presented. First a global introduction to the thesis subject is given followed by the problem statement with all requirements and research questions. Finally an overview of the rest of the thesis is presented.

Chapter 1 Introduction

1.1 The Watching window

The Watching Window (WW) is an ongoing project at the Computer Graphics and Vision laboratory at the University of Otago. In time the WW project has evolved from a single computer application to a complete networked system. The WW is an environment that enables a user to interact with Virtual Reality (VR) applications without the use of any physical hardware (e.g. markers, gloves, HMD, etc.). To achieve this, 3 cameras are directed at the user and Computer Vision (CV) techniques are used to determine the coordination's and the movement of the user's body-parts. These coordination's and movements are used as the input of VR applications which is drawn on a big screen in the center of the cameras, see Figure 1. Currently, the environment is situated in a booth to get the best CV results and a long-term goal of the WW is to get the WW out of the controlled environment making it everywhere available with a few computers, a projector and cameras. Appendix A, gives a specification of the booth and the computers used in the WW.

Figure 1 - Watching Window Application

The WW environment is named after the main application for which it is build, the Watching Window. This application defines the framework for other applications created for this environment. The WW application is a simulation of a window with an interesting background. In this virtual window, the background is drawn in the right perspective according to the coordination of the user's head, which is determined by analyzing the camera observations. Because the head coordination is updated in "real-time" and the perspective is redrawn in "real-time" the user experiences the right motion parallax[1]. This enables the user to walk around and look through the virtual window as if it where a real window with a real view behind it.

Besides the parallax view, the WW environment has the possibility of a creating a 3D experience. This is achieved by drawing the right perspective for each individual eye in a different color, red and blue, and overlaying these views to define the final perspective view. When the user wears red-blue filtering glasses, the right view is filtered for each eye and each eye sees a different view specifically projected for this eye. This trick enables the user to experiences real depth in the VR world. In combination with the motion parallax view, any application created for this environment can give the user a real VR experience without the use of any hardware (except for the glasses).

1.2 Problem statement

The subject of this thesis is about the extension and improvement of the current WW environment. When I started this project the WW environment was only capable to track the head of the user and use this information in the 3D and motion parallax view. These applications give the user a great VR experience which becomes even more realistic when physical interaction with the virtual world is possible. For the user,

[1] *Motion Parallax is the change of angular position of two stationary points relative to each other as seen by an observer, due to the motion of an observer. The objects that are far in the background move slower when the user moves his head then the objects that are close to the virtual window.*

the most natural interaction with the WW environment would be the use his/her hands. When the user can use both hands it is possible to create more interesting applications and games, e.g. the virtual clay ball created by D. Simpson. As the name suggests, a virtual ball of clay is projected in the WW and the user can shape this ball in any form. This implies that the WW environment needs to know the coordination's of the user's hands. In several previous version of the WW environment, several hand tracking algorithms [33, 34, and 35] have been tried. However, none of these hand tracking algorithms are currently working and a complete new solution needs to be found.

1.2.1 Goal

The goal of the project is to track the hands of the user in order to enable more interesting and challenging applications. This means that a system needs to be created that fits into the current WW system and tracks both hands of the user. The coordination's of both hands, need to be transmitted to the VR application that uses this information. In short:

Track both hands of the user in world space using the current available setup of the WW.

1.2.2 Requirements

There are several requirements which define the boundaries of this research project.

1. *Good accuracy and speed.*
   The tracking of the coordination of both hands should be accurate enough to manipulate and move objects in the Watching Window environment and the system needs to operate in "real-time" in order to facilitate smooth interaction with the VR application.
2. *No physical changes to the user.*
   The current WW environment does not use any equipment and this property has to be maintained. This means that the user should be able to walk in and use the system without putting on some hardware.
3. *User independent.*
   It should not matter how the user looks. This means that size, clothes and skin-color of the user makes no difference in the tracking.
4. *Intuitive to use.*
   The user should be able to come into the booth and use the system without any knowledge on how to use the system.

1.2.3 Research questions

To achieve the main goal according to the requirements some research questions are defined which are divided in two main questions namely a research and evaluation question. The research question defines the system that can track both hands and the evaluation question determines how good the system works.

How can both hands of the user be tracked in the WW?

1. What projects are proposed in literature and are any of these projects useful?
   a. Which techniques do they use and which of these techniques are useful?
   b. Which of these techniques can be combined to form a new solution?
2. What does the current WW look like?
   a. What does the current WW architecture look like?
   b. What are the current CV techniques used in the WW?

When the system is defined, what is the performance of the system?

3. How accurate does the system track the hands of the user?
   a. How accurate are the estimated hands in relation with the real coordination?
4. What is the operating speed of the system?
   a. Is the operation speed sufficient for the user interaction?

1.3 Thesis outline

The rest of this thesis is subdivided into 3 parts namely the research, design and evaluation part. The research part consists of a literature study, elaborated in the next section, and gives an overview of different hand tracking and body pose tracking techniques. In addition some interesting useful techniques are explained in detail.

The design part starts with an overview and motivation of the methods used. First an evaluation of the WW and the currently used computer vision techniques are explained. From the literature study and the WW

information the hand tracking system is defined globally. The design part finishes with a detailed description of each of the components of the system.

In the last part, the module is evaluated in relation with the requirements stated in the previous section. First the evaluation setups, techniques and results are presented. Then the goal and requirements, described in the previous section, are related to the results to give a conclusion on the performance of the system.

Part II Research

This part describes the research of several Computer Vision techniques. First an introduction into the field of Virtual Reality and Computer Vision is given. From this introduction the field of pose estimation and tracking is explained in detail.

Chapter 2 Literature overview

2.1 Introduction

2.1.1 Virtual Reality

Ivan Edward Sutherland is the inventor of Sketchpad, an innovative program that influenced alternative forms of interaction with computers. Sketchpad was the first program that interacts in the way we currently interact with the computer. Later in 1968, Sutherland created what is widely considered to be the first VR and Augmented Reality (AR) Head Mounted Display (HMD) system. It was primitive both in terms of user interface and realism. The HMD worn by the user was so heavy it had to be suspended from the ceiling and the graphics of the virtual environment were simple wire frame rooms.

Since then, VR has matured and new systems and programs are created that aid us in many ways. The army uses VR to train its pilots and soldiers for warfare. NASA uses the same techniques to train astronauts for every possible situation before they even leave earth. Even the industry uses VR to design and test new products for there ergonomics. For example, in the airplane industry the designers can virtually sit and fly the plane before it is even build.

Still the majority of the VR applications are made for the amusements industry. Here the main goal of VR is to entertain users in the form of games and movies. These games and movies become more real when new VR techniques are designed. One technique that enables more realistic and natural interaction comes from the combination of VR and Computer Vision.

2.1.2 Computer Vision

Computer vision is the study and application of algorithms that allow computers to "understand" images. This understanding of images differs from the task that it is developed for. This understanding can be very superficial, e.g. color information of the image, and be very specific, e.g. recognizing objects. Many of the methods and applications are still in the state of basic research. However, more and more methods have found their way into commercial products. Here they are part of a larger system that can solve complex tasks (e.g., in the area of medical images and quality control or measurements in industrial processes).

One way to describe computer vision is by describing some of the application areas in which computer vision is used. One of the most common and prominent application fields is medical computer vision. In medical applications, computer vision techniques are used to analyze noisy image data (e.g. reconstructing 3D models of a series of noisy microscopic images [14]). A second big field in which computer vision is used is the industry. In the industry some computer vision techniques are used to control specific process in a factory (e.g. quality check on a factory belt or robot arm coordination and orientation). However, maybe most applications are used in the military. Even though most of the work done in the military is not open for public, one can guess where computer vision might get used, e.g. missile guidance, soldier detection from satellite images, enemy missile tracking etc.

Another big CV research field is focused on detecting and tracking humans and their poses. Applications made for this field include surveillance systems, advanced user interfaces, motion analysis and VR [5, 32]. If computer vision is used in these applications, instead of other tracking devices, people and their poses are estimated from the camera observation. Some of these applications only need to know if there is a human present (surveillance systems) while others need to know the complete human body configuration (motion analysis).

In VR and advanced user interfaces, body pose estimation is used to remove the cumbersome hardware that is currently needed. As explained in the introduction this thesis is about a system that enables the user to have a whole VR experience without the help of suit, gloves, HMD or any other hardware, instead computer vision is used.

2.1.3 Watching Window

The WW is a VR environment that uses CV techniques to estimate the head coordination of the user and use this information in VR applications. This thesis is about a system that uses CV techniques to track both hands of the user in several camera observations. This problem of estimating both hands is described by the difficulty in observing the hand. Firstly a hand may be observed in an infinite number of shapes and sizes.

This has several reasons, first of all the camera observation is ambiguous. This means that one hand pose can be represented by several camera observations. On the other hand multiple hand poses can result in the same camera observation. The other reason for the amount of different observations comes from the high number of body-parts in the hand. All these body-parts can move in more directions, each occluding other body-parts, making up an infinite number of possible shapes creating a huge dimensionality in the search space.

Another problem in observing the hand is the size of the hands in the camera observation. The hand is only a small percentage of the complete human body. Since the hand tracking system must be able to observe all the possible coordination's that the hands can be in, the hands only make up a small part of the camera observation. This creates a huge search space in which the hands can be anywhere in the observation. Because the hand is hard to detect and the literature about hand pose estimation only address the estimation of the hand pose instead of finding the hand in and image, the field of body pose estimation is investigated. The pose of the user can determine the coordination's of the hands of the user and thus solve create a system that tracks both hands. The next section addresses the art of body pose estimation while the section that follows addresses the tracking problem.

2.2 Body pose estimation

In the art of body pose estimation, the pose of the user is estimated according to one or more camera observations directed at the user. The detail of the estimated body depends on the application in which the pose estimation is used. One can imagine that for motion analysis, the amount the detail must give the best possible appearance of the human body. While for surveillance cameras a determination of people and global movement is sufficient. The detail is described by the number of body-parts, each of which is connected to a joint. Each joint can rotate in zero or more directions and each of these directions is called a Degree of Freedom (DOF). The amount of DOF determines the detail of the body pose.

Like the hand, also body pose estimation suffers from the dimensionality of the body. The humanoid representation of the H-Anim standard [29] has 94 joints in total. This representation includes a representation of the head, the hand, the vertebrate column and all the limbs of the user. Each of these joints can rotate in one or more directions creating at least 94 DOF. Gavrila and Davis [32] and Sidenbladh et al [18], each use 22 and 25 DOF respectively for their model estimation. However pose estimation can go so far as 54 DOF [31].

There are a lot of example algorithms that successfully track complete body poses of users in video sequences. All these of body pose estimation systems need to find a relationship between the camera observation and the pose of the user. Therefore the analysis of the image is an important part of the body pose estimation and this is described first in the next section. The field of body pose estimation can be divided in two main methods namely model-based and model-free pose estimation. In model-based approaches, a model of the user is used estimate the pose of the user. The model-free approach uses the camera observation to directly infer the user's pose. Sections 2.2.2 and 2.2.3 give an overview of each of these methods.

2.2.1 Image features

When analyzing the images the decision needs to be made which image features describe the estimated object best. Image features, also called background subtractions, are useful abstractions from the camera observation. In background subtractions all irrelevant information is removed from the camera observation, only the subject of the body pose estimation algorithm remains. The problem in the image feature analysis can be described by the appearance of a user. The appearance of the user differs in each camera observation because of lighting changes, different camera angles, user clothes etc. For this reason, image features need to be extracted from the camera observation, which are useful abstractions of the estimated object. In the next paragraphs an overview of several of the most used image features is presented.

The most common image features are the motion, edge, color and shape feature. When a feature needs to be extracted from a video sequence, the easiest usable feature would be the motion feature. The motion feature assumes that the object that needs to be estimated is the only continually moving object in the scene. When this assumption holds, the motion feature is an efficient, simple and computable feature. The motion feature determines the difference between two consecutive frames. The resulting image feature only represents pixels that were not corresponding in the previous frame. This comes from the assumption that the intensity of one pixel in the object does not change, it only moves. Therefore, the image feature only represents the moving objects and (if the assumption holds) the Object of Interest (OI). One problem, besides the movement assumption, is the fact that light changes and shadows give noise in the motion feature. This noise needs to be filtered out to get a clear image feature of the OI.

The edge feature uses the edges between high and low intensity to remove the background. The edges determine the outline of an object which is useful in the shape feature. The major advantage of the edge

feature is that the edge feature is invariant to light changes. On the other hand, one problem from which the edge feature suffers are high noisy texture backgrounds. When the background texture is very noisy, a lot of intensity edges are represented in the resulting edge feature from which an object is hard to determine.

The color feature is used in a lot of applications in a variety of ways. One method used in many applications uses the color distribution of the OI. This color distribution is compared with the camera observation and all pixels that do not fit this color distribution are filtered out [3, 35]. The remaining pixels are determined to be the OI. The main problem from which the color distribution method suffers is the lighting condition, e.g. normal lamp, tube light or the sun. The problem occurs if the lighting condition changes, then the color distribution of the OI also changes. Yet another problem is that man objects are hard to generalize e.g. skin color.

Another method, described by Horprasert et al [8], uses color to remove noise, shadows and highlights in the motion feature. Their algorithm uses a Background Model (BM) that describes the irrelevant background information. Their method filters out moving objects by comparing a new frame with the BM. The algorithm defines a pixel in the new frame to be background if it has both brightness and chromaticity similar to the BM pixel. It defines a pixel a shadow if it has similar chromaticity but lower brightness than the BM pixel, a highlighted background if it has similar chromaticity but higher brightness and a moving object if it is chromaticity different from the background model pixel. In addition the BM model is updated every time step to handle lighting condition changes. With this model they successfully perform motion filtering, removing all shadows or highlights in indoor and outdoor scenes even when the global lighting changes.

In many projects a combination of image features is used to extract high level information from the camera observation. For example the shape feature is mostly used in combination with other image features to extract high level information. In [13] lines are detected from an edge feature with a Hough transformation. From these lines only the parallel lines are selected and shapes are created from these parallel lines. In contrast, Yao et al [3] propose a system that uses a combination of image features to detect a raised arm in a classroom. Their system filters the camera observation with a motion filter and filters the resulting motion feature with an erosion and dilation filter to remove noise. They combine this motion feature with an edge feature to get good edges around the moving objects in the motion feature. Finally a skin-color check is performed to define which objects are arms. Haritaoglu et al [9] propose a general-purpose surveillance system that determines human motion and body pose. They use shape analysis to define points of interest and use corresponding points in stereo camera observation to determine the depth of the points. From these points of interest the human pose is successfully determined in outdoor scenes with multiple people.

2.2.2 Model-based body pose estimation

In model-based pose estimation a virtual model, that represents the user, is used to estimate the users pose. his model is described by its appearance, state representation, kinetic constraints and evaluation function. The appearance of a model can differ from a simple 2D plane representing with specific image features to a complete 3D surface model. Each of these appearances has its own advantages and disadvantages and the application for which it is designed determines the appearance. E.g. a 3D surface model of the user would represent the user in a perfect way and detailed tracking is possible with this model. However a 3D surface has 2 major problems. Firstly the model is very computational to manipulate and secondly the surface model is user (or clothes) dependant. This means that for each user (or the clothes that the user is wearing) a new surface model needs to be designed. In the case of motion analysis this would not be a problem because speed is not an issue and a detailed model can be created for each test person. However, when the application needs real-time estimation of completely different users, this model would be impractical. In [32, 16 super quadrics, e.g. cylinders, spheres, ellipsoids, cones and hyper rectangles, are used which are connected to each other to form a simple generic but realistic appearance of the user.

In literature there are two significant model-based body-pose estimation techniques namely top-down and bottom-up. In the top down approach the model is matched with the image features to estimate the pose while in the bottom up approach the individual body-parts are found and reassembled together to form the estimated pose. Currently some systems use a combination of both techniques to overcome the disadvantages of both of them. Each of these techniques is described in the next sections.

2.2.2.1 Top down

The top down approach matches a model of the user with the camera observation. The model parameters are changed until a pose is found that matches the camera observation best. Because this brute force method is not applicable in real-time, most top-down approaches use an initial pose to estimate the next pose. When the system needs to track the user over time this naturally implies an initialization procedure that determines the user's initial pose.

Gavrila and Davis [32] use a top down approach with a 3D appearance created by connected super quadrics. They use a 3D appearance because 2D appearances use the assumption that the user needs to

move parallel to the image plane while they want to be able to track more unconstraint human movement. Also the ability of projecting the same model onto multiple cameras makes it possible to track the body pose more stable because poses that are ambiguous in one view might not be ambiguous for another view. They use search space decomposition to overcome the dimensionality of the search space and first search for the head and torso and then estimate the other body-parts. Sidenbladh et al [18] use a top down method with a set of models because they assume that an analytical expression of the likelihood value can not be determined. This implies that one model can not give a clear answer if it is the right representation and therefore they use a set of models and let the best model be the estimated pose of the user. With this method they successfully use a set 10.000 models to estimate the user's pose at 5 min per frame.

Mikić et al [7] use Voxel data to estimate the user's body pose. In this approach multiple cameras are used to determine the movement surface in 3D space. For each of the cameras the movement surface is computed by a motion feature. Each motion pixel is triangulated to a 3D coordination (voxel) to form the 3D surface of the user's movement. From this 3D surface model the user's pose is inferred by a top down modeling approach in 3D space. A model of the user is matched in this 3D space to estimate the pose of the user that creates this movement surface. The big advantage of this method is that the model matching method is done in 3D which enables the use of a good likelihood function. A big problem with this approach is that the estimation of the 3D movement surface is very computational. Mikić et al resolve this issue by pre-calculating all the possible combinations of pixels offline and transforming the triangulation calculation into a database search.

Bregler et al [1] demonstrate a visual motion estimation technique that is able to recover human body configurations in video sequences. They use a single prior pose of the human body configuration and integrate a mathematical technique, the product of exponential maps and twist motions, into differential motion estimation. This results in solving simple linear systems, and enables them to recover the DOF in noisy and complex configurations.

2.2.2.2 Bottom up approach

In the bottom up approach the body model usually consists of 2D body-parts. Each limb is modeled by image features and in the estimation phase each limb model is matched with the camera observation. The great advantage of the bottom up approach is the fact that it does not need an initialization procedure. When the model is designed right the bottom up approach estimates the body pose fast and good. However, the design of the model turns out to be very difficult. The main problem is the fact that the users appearance changes because of lighting condition changes and rotation of the body-parts.

Ramanan and Forsyth [2, 12] propose a system that uses a bottom up approach with some assumptions. First they assume that all human body-parts are cylinder shaped and secondly that the body-parts in the video sequence do not change dramatically in appearance. The second assumption is in contrast with the problem that the body-parts do change due to rotations, but it works in small videos because they assume that the user does not change clothes over time. Their approach detects the user's body-parts with the first assumption, e.g. find all the cylinder shaped objects in the camera observation. These cylinder shaped objects are detected by convolving over the image with parallel lines of contrast. This is only done for a small part of the video sequence in which all the candidate body-parts are followed. The candidate body-parts are matched with a kinematics structure and movement model to form a complete body. The candidate body parts that do not fit these models are discarded. The color distributions and textures of the human model's limbs are learned and used to track the body of the rest of the video sequence. With this method Ramanan and Forsyth successfully track user body pose in several video sequences.

Sigal et al [19] propose a new method that uses a loosely-connected model. They use a graph model in which each node represents a limb model, each described by the appearance, likelihood function and current state. Each node is connected to the other nodes by a conditional probability. Together with a temporal evaluation probability of each limb the pose estimation is solved with an adjusted particle filter.

2.2.2.3 Middle way

The middle way uses the advantages of the bottom-up and top-down approaches described in the previous sections to overcome the disadvantages. Navaratnam et al [4] use a model that has a collection of separate body-parts linked according to a kinematic model. Each body-part is represented by a set of 2D templates created from a 3D model which encodes the 3D joint angles. In the first step all body-parts are searched in the camera observation with the 2D templates (bottom-up approach). Then the most reliable detected part is chosen to be the anchor for the rest of the estimation process. The rest of the parts are searched based on the detected location of this anchor in a kinematical order (top-down). The big advantage of this approach is that it does not need an initialization procedure and the 3D rotations of the users can be tracked.

2.2.3 Model-free body pose estimation

In model-free body-pose estimation the body pose is directly inferred from the camera observation. This means there is a need for a direct relation between the pose and camera observation. This direct mapping is not always obvious and easy to design because an analytical function that represents the likelihood of the user is impossible to design.

Argiwald and Trigs [31] use a learning based approach to directly infer a pose from a monocular camera observation. They choose for a learning based method to avoid the use of explicit initialization and 3D modeling which needs computational expensive rendering. They use a set of motion captured data to learn a reconstruction function. The motion capture data images are reduced to 100D image feature vector's which represent the human movement. Given this set of vectors a non linear regression algorithm learns a smooth reconstruction function which is used to determine the pose given a set of input images. However, monocular camera observations are ambiguous. This means that multiple hypothesis are possible give the camera observation. This problem is solved by either a regression on the image and the previous pose or the use a mixture of regressions in a multiple hypothesis tracking scheme. With this method they successfully recover a 54 DOF model in a monocular test sequence with real users. The advantage of this method is that it does not need an initialization procedure and is not computational. However, only users that are similar to the user in the motion capture data can be used in this system.

2.3 Tracking

Most pose estimation systems are designed for tracking the users pose over time and use a prior state to estimate the posterior state. In order to track the body pose of the user several methods are possible. Some methods use a single hypothesis while other methods use multiple hypotheses to determine the state. In this section the most common tracking techniques are discussed in detail.

2.3.1 Kalman filter

The Kalman filter [24, 25] is a generic estimation technique that estimates the new state of a system given the current state and noisy sensor data. The Kalman filter is proven to work and applied in many applications used today. The most common application is airplane tracking on radars. In this application the sensor data is the noisy radar data and the state of the system are the airplanes position and velocity. The Kalmann filter estimates the posterior state of the system in 2 steps, the predictor and corrector step. In the predictor step the next state of the system is estimated and in the correction step the predicted state is compared with the sensor data to refine the estimated state.

The prediction step depends on a linear transition function and a transition noise measurement described by a white Gaussian distribution. The corrector step depends on the noisy sensor data, a sensor noise described by a white Gaussian distribution and the Kalman gain. The Kalman gain is determined by the noise of the sensor data and the estimated error covariance from the prediction step estimation. The Kalman gain's goal is to minimize the estimates error covariance after the corrector step and thus get the most probable state of the system.

The Kalman algorithm begins with an update of the state of the system which is determined by the linear transition function and the previous state of the system. The prior error covariance is estimated according to the posterior error covariance of the previous state and the transition noise distribution. In the corrector step the Kalman gain is estimated according to the prior error covariance and the sensor noise. The estimated state is updated according the sensor measurement and the Kalman gain. Finally the posterior error covariance is calculated for the next filter cycle.

The Kalman filter assumes that the transition function is a linear function in order to predict the posterior state of the system. If this assumptions hold up there is no better solution then the Kalman filter. However, human body pose movement is far from linear because of joint acceleration. If the transition function is non-linear then the Extended Kalman Filter (EKF) can be used to estimate the state of the system. The EKF estimates the linear representation of the non-linear function every time step by using the calculating the Jacobian matrix of the non linear function and use the resulting matrix in the Kalman filter equations. An alternative to the EKF is particle filtering which has the advantage that it can represent the Probability Density Function (PDF) when enough particles are used. The main advantage of the particle filter is that it contains multiple hypothesis, when only a single hypothesis is maintained, e.g. Kalman filter and the EKF, the estimation can get stuck or drift in the wrong direction.

2.3.2 Particle filter

A particle filter is a sequential Monte Carlo method [23, 24], used in many applications and estimates the values of hidden variables according to noisy sensor data. In order to explain particle filtering the notation needs to be elaborated. A particle filter has a population of N samples (particles), each of which resembles a state of the system that needs to be estimated. Let $X_k=\{x_k^i,\ i=0\ldots N\}$ denote the set of N particles at time k

and Z_k denote the noisy sensor data. The goal of the particle filter is to estimate the posterior PDF denoted in Equation 1.

$$P(X_k \mid Z_k) \qquad \text{Equation 1}$$

2.3.3 SIS particle filter

The standard particle filter is called the Sequential Importance Sampling (SIS) particle filter. The SIS particle filter works according to two steps namely the prediction and evaluation step. The first step estimates the next state of each particle while the second step evaluates each particle according to the sensor data.

In the first step all the particles are propagated forward according to the current state of the particle and a transition function. The transition function can be a non-linear function which determines the behavior of the system, see Equation 2.

$$\widehat{x}_k^i = f(x_{k-1}^i) \qquad \text{Equation 2}$$

In the second step a weight is assigned to each particle, denoted by $W_k=\{w_k^i, i=0....N\}$, which is determined by the likelihood function, see Equation 3. The weight population is normalized according to Equation 4.

$$w_k^i = P(Z_k \mid \widehat{x}_k^i) \qquad \text{Equation 3}$$

$$\sum_{i=1}^{Ns} w_k^i = 1 \qquad \text{Equation 4}$$

If the number of particles comes close to infinity the complete population, particle with its weight, resembles the PDF and Equation 5 determines the state of the system at time k.

$$P(X_k \mid Z_k) \approx \sum_{i=0}^{N} w_i^k x_i^k \qquad \text{Equation 5}$$

However, when the number of particles does not come close to infinity the population is a poor resemblance of the PDF and the state estimation is difficult. Often there exist many local maxima in the likelihood surface and when for example 2 local maxima exist, each tracked by a separate group of particles, the state calculated by Equation 5 is be placed in between of the local maxima. A solution to this problem divides each particle group into different local maxima groups. The local maximum that best resembles the state of the system is chosen to be the state of the system.

Another problem from which particle filter suffers is the degeneracy problem. This problem occurs when after a few iterations all but one particle has a negligible weight. This implies that a lot of particles use a lot of computing time while there contribution to the PDF is almost zero.

2.3.4 SISR particle filter

The most common method of eliminating the degeneracy is called resampling. In resampling a new set of particles, X'_k, is constructed from the old particle set. The idea behind resampling is to eliminate the particles with low weights and concentrate on the particles with high weights while the new particle weight set approximates the old particle weight set as good as possible. The particle filter population is resampled when the variance of the weights is high and therefore degeneracy is taken place. However for implementation issues, the Sampling Importance Sampling Resampling (SISR) particle filter is often used [24]. In the SISR particle filter the resampling step is added to the particle filter.

While resampling is the frequently used, it suffers from the sample impoverishment problem. Sample impoverishment occurs when all particles follow the same trajectory and there is no diversity in the particle population. There are several resampling schemes e.g. multinomial, stratified, residual and systematic resampling [30]. Residual, stratified and systematic resampling give comparable results but the most common resampling algorithms systematic resampling for its simplicity in implementation. In systematic resampling the new particle population X'^i_k is determined by drawing N samples from a uniform distribution according to Equation 6 in which U is defined by a uniform distribution on the interval [0, 1/N].

$$U^i = (i-1)/N + U \qquad \text{Equation 6}$$

The new particle set is determined according to the Cumulative Density Function (CDF) which value for each particle is calculated according to Equation 7.

$$c_j = c_{j-1} + w_k^j \qquad \text{Equation 7}$$

The particle X_k^i in the new particle set is chosen to be the particle X'^j_k from the old particle set in which $c_{j-1} <= U^i <= c_j$. The resulting particle set focuses on the particles that have a higher weight by copying the particles with the high weight multiple times in the new particle set and removing the particles that have a low

weight. In order to maintain diversity and remove sample impoverishment in the population some particles with a low weight are copied in the new particle set.

2.3.5 Transition function

The transition function describes the behavior of the system and determines the most likely new state of each particle given the prior state of the particle. In the case of body pose estimation, the transition function is an estimate of the next probable pose given the current pose. However, the human body can perform a massive array of movement. Therefore a transition function that gives the next likely pose for every possible pose becomes infeasible. However, in most cases the human movement is restricted to some class of movement, e.g. walking, grabbing, etc. In this case, a probabilistic model can be created from the training data of this class of movement. This probabilistic model can then determine the next probable pose given a window of previous poses of the system.

Sidenbladh et al [6] propose a probabilistic motion model for pose synthesis and tracking. Their view on the problem changes from learning a probabilistic model to a searching problem. They propose a system that uses a large set of human motion. With a time window of *d* frames, this motion set is divided in a number of different movement samples, each of *d* frames. The dimensionality of the sample space is reduced according to principal component analysis (PCA) and each sample is projected on this low dimensional sub space. The resulting sample vectors are put in a binary tree for low cost searching. The input of the system, also of length d, is then projected on the same low dimension space. The resulting vector is compared with the database which returns the most likely sample given the input sample. From this sample, the next pose in the normal set of human motion is chosen to be the next probable pose of the system. The assumption made is that the high order statistics are implicitly represented in the database which removes the need for learning a motion model.

Part III Design

In this part the current system and some previous hand tracking attempts are evaluated. Together with the information from the literature study a new hand tracking approach is introduced. Chapter 3 gives a global overview of the approach and Chapter 4 explains the hand tracking system in detail.

Chapter 3 Main Approach

3.1 Current state

In order to create a system that tracks the hands of the user, the current WW system needs to be investigated. The architecture and computer vision algorithms that are already tried previously need to be explored. Even the current refining of the head tracking component is interesting in designing the hand tracking algorithm.

3.1.1 Current Architecture

The current WW architecture uses 3 different modules that work in a chain, across several computers, to get an end result. These are the computer vision, 3D server and the application modules. Figure 2 gives an overview of the modules. Each camera is connected to a computer vision module which performs the computer vision tasks. The results from the computer vision modules are transmitted to the 3D server module which estimates the 3D coordinates of these body-parts. These results are then transmitted to the application module which draws a VR application on the screen according to the information of the user. The WW architecture is setup in such a way that new cameras can always be added to improve the tracking. Because each camera has it own computer vision module which tracks the user's body-parts, more camera's results in more 2D coordination's of the user's body-parts which in turn results in a more detailed 3D coordinate estimation. In the current architecture the computer vision modules only execute computer vision calculations. The 3D server only handles the 3D data and the application module only draws the output on the screen.

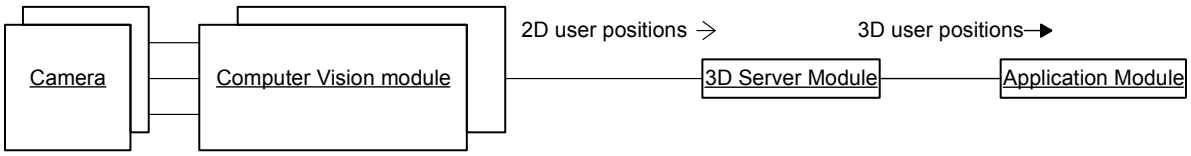


Figure 2 - System architecture

3.1.1.1 Computer vision module

The computer vision module's task is to track the coordination's of the user's body-parts. Currently the only body-part tracked by the computer vision module, is the head. Figure 3 shows the images from all 3 cameras currently used by the system. The cameras are installed in the booth in such a way that the complete booth is visible for each camera. Because of this installation, the camera observations are rotated.

Figure 3 - Images from the three different cameras

The computer vision module's tracking algorithm is made as general as possible so the computer vision module becomes camera independent. When the computer vision module is camera independent more cameras can be used which results in more information. Because the top camera has a completely different camera observation, in relation to the side cameras, a camera independent tracking algorithm is impossible. Therefore in the current head tracking scheme, only the side cameras are used. The hand tracking system however, needs to be as general as possible in order to use all possible cameras and thus use all available information to determine the coordination's of the hands.

3.1.1.2 3D server

The task of the 3D server is to estimate the 3D coordination of the user's body-parts given the 2D coordinates obtained from the computer vision module(s). This is only possible when there are multiple

cameras from which the intrinsic and extrinsic camera parameters are known. Figure 4 shows how the 3D coordinate of the point *X* is obtained from two observations. To determine the 3D point, a ray is traced for each camera, from the centre point of the camera through the point on the image plane, U_1 and U_2, into world space. The point where both rays intersect denotes the point in 3D space. See appendix B for a more detailed description of the 3D point estimation.

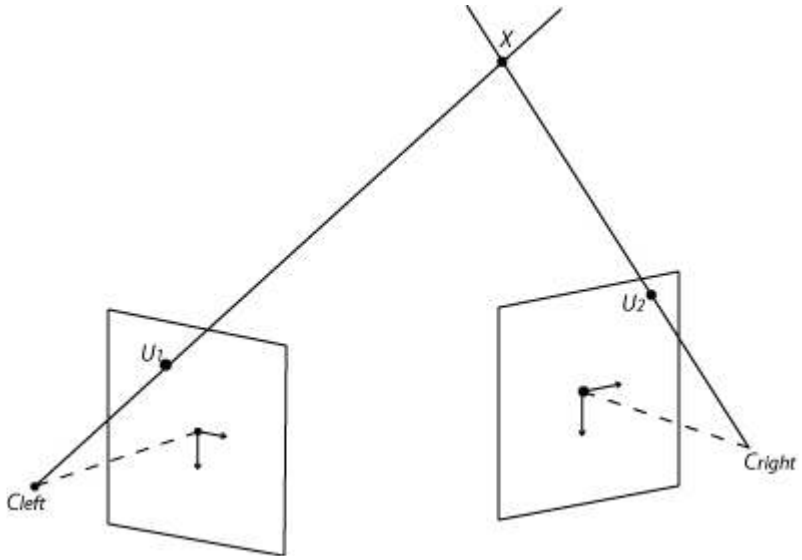


Figure 4 - 3D coordination estimation

3.1.1.3 Application

The application module is the VR application that uses the WW environment. This application can use the parallax and 3D visualization. Examples of applications are interactive games such as a maze, Rubric cube, virtual clay ball etc.

3.1.1.4 Networking

The protocol that connects all the modules to each other is the XML Remote Procedure Call (RPC) protocol. This protocol is good in sending complex structured data and works with remote method invocation on the server. The data is transmitted in XML code in order to make it readable for the user and easy in use. This communication protocol works well but is not applicable for fast communication of a massive amount of data. A problem in the current communication setup is that the 3D server acts as a RPC server. This means that the other modules can only ask for methods from the 3D server. The 3D server has no possibility to send requests to the other modules.

3.1.1.5 Conclusion

From the current architecture it can be concluded that the hand tracking system needs to present the coordinates of the hands to the 3D server, which communicates them to the application module. The hand tracking system can be part of the computer vision module and send the 2D coordinates to the 3D server. However, a general hand tracking algorithm that can be used in any camera observation is hard to design and a new algorithm have to be designed for each camera observation.

3.1.2 Current Computer Vision Algorithms

3.1.2.1 Head tracking

Currently only the head of the user is tracked by the computer vision module. In time, the head tracking algorithm has evolved from skin-color filtering to a particle filter that uses Eigenfaces as a likelihood function. The head tracking algorithm used in the beginning of the project used a particle filter method in which the state is the location and size of the user's head. Each computer vision module tracks the user's head with this particle filter. The appearance of a head is chosen to be an ellipse [35] and the likelihood function of the particle filter then becomes an ellipse shape matching procedure. Each computer vision module communicates the tracked coordination of the head to the 3D server which transforms them into 3D coordination's. The shape matching procedure from this particle filter is revisited into a likelihood function that uses Eigenfaces.

The Eigenface likelihood function [35] uses an average face created by averaging over a database of faces. This average face is subtracted from the other faces in the database and PCA is used to define a set of eigenvectors that best define the database set. In the tracking mode the average face is subtracted from the camera observation and the resulting image is compared with the eigenvector. The distance between the eigenvectors is the likelihood of observing a face. This new likelihood function enables the computer vision

module to become camera independent, nevertheless an Eigenface database needs to be created for each camera.

3.1.2.2 Hand tracking algorithms

Previous hand tracking attempts used the same tracking scheme as the head tracking algorithm. Namely, the hands are tracked in each computer vision module separately and the 3D server combines these coordinates into 3D world coordinates. All of these hand tracking algorithms assume that the user would only use one hand at the time and that the hand is close to the screen while interacting with the VR application. A good example that makes use of these assumptions is the earliest hand tracking algorithm. This algorithm denotes the skin-color pixel or movement pixel[2] that is closest to the screen as the hand coordination. This can only be done in the side cameras because only in the side cameras the closest pixel to the screen can be annotated see Figure 3.

The main problem with this approach is that the skin-color feature is not reliable. If the user wears an orange shirt the system can not distinguish between the shirt and the hands or face of the user. Another big unresolved skin-color feature problem is the lighting condition. A small change in the lighting condition, e.g. an object occluding the light source or different light sources (tube light, sun, etc), has a great influence in the skin-color. For each of these different lighting conditions a new skin-color filter needs to be created. In the current WW booth this is not a problem but the skin-color feature is impossible to generalize. Because of the bad characteristics of the skin-color feature it was decided to replace the skin-color feature for other algorithms. These algorithms used a motion feature to determine the hand of the user that was closest to the screen.

3.1.2.3 Conclusion

The hand tracking schemes described in the previous paragraphs are not satisfactory because they cannot be generalized. The skin-color feature depends too much on the lighting conditions. The other methods use the assumption that the user points to the screen and only track one hand in the side cameras. For these schemes it is impossible to create one generalized method that tracks a body-part from every possible camera observation. While these approaches cannot be used, the results obtained with the particle filter that uses a shape matching likelihood function, is promising.

In both the head and hand tracking there is another major problem. The computer vision modules are not synchronized. This means that each computer vision module tracks the body-parts for themselves. A problem might occur when one computer vision module tracks a shadow and another one tracks the right body-part. When this happens, the 3D point estimation fails or the projection gives the wrong representation of the body-part.

3.2 Main approach

3.2.1 Choices

Recall that the main goal is to track both hands of the user in “real-time” in order to make interesting interaction with the VR world possible. There are two main choices that can be explored. Option one has the same scheme that is tried before and has proven to work. In this scheme the tracking is done on each computer vision module independently and the results are transmitted to the 3D server which converts them to 3D coordinates. This option has the problem that it is hard, if not impossible, to find a generic method that can track the hands in several different camera observations and the tracking is done separately. Another option would be to design a new module that fits into the current architecture and uses services of the other modules. This option has the advantage that one single module uses all the available information from all the camera observations to determine the coordination of the hands.

Recall from the literature introduction that the hand is very hard to detect in the camera observation because of the small size and unlimited shapes it can have. From this observation it can be concluded that first tracking the arms of the user and then derive the hand coordination from the arms, is a promising method. The arms are easier to track than the hands because more information is available. When the problem is analyzed in this way, the hand tracking problem becomes a body pose estimation problem. Another advantage of the body pose estimation is that in a later stage the pose of the user can be used to determine gestures or other high level information from the user.

3.2.1.1 Upper-body pose estimation

The literature study gives a lot of options how to estimate a body-pose. All body pose estimation algorithms need to solve the self-occlusion problem. The literature overview gives 2 main body pose estimation methods, namely the model-based and model-free method. The model-based method has 2 main advantages over the model-free method. First, if a 3D model is used in a separate module, one great

[2] *A skin-color pixel is a pixel from the skin-color feature and a motion pixel is a pixel from the motion feature.*

advantage is that model-based pose estimation can handle the ambiguity in the camera observation by projecting the self-occlusion. Secondly, a 3D model-based method is easy to generalize for each camera observation because a 3D model can be projected on each camera plane making pose estimation the same for each camera observation. This enables the use of multiple camera observations together to get a tighter pose estimation and removal of the ambiguity [32]. Recall that the model-free estimation techniques use the camera observation to directly infer the pose. This method implies that there exists a direct mapping between the image features and the 3D pose of the user. Because in the model-free method a generalized mapping is hard to solve, the 3D model-based method is used.

In the literature study 2 main model-based methods are discussed, top down and bottom up. Both of these methods have advantages and disadvantages. The main disadvantage of the top down method is the need for initialization. And the main disadvantage of the bottom up method is the design of the model. The middle way approach that uses the advantages of these methods is a promising method.

When designing the upper-body estimation algorithm the tracking component should always be taken into account. In the case where the transition function is linear the Kalman Filter would be the optimal solution. However, the human body's movement is far from linear. A particle filter would be the better solution for the "real-time" body pose estimation problem however, for a particle filter the dimensionality of the human body is huge. If a particle filter is going to be used a solution needs to be found that solves this dimensionality problem.

3.2.2 Hand-tracking module

From the pros and cons of the choices discussed in the previous paragraph, a system that tracks both hands of the user can be designed. In order to make the hand tracking more generalized for the cameras observations a module is created that fits into the current WW architecture. This module is placed between the computer vision and the 3D server and uses the computer vision modules to estimate the coordination of both hands, see Figure 5. When this module is between the computer vision modules and the 3D server a module also has to act as a RPC server like the 3D server because of the one way communication of XML RPC protocol.

The big advantage of this module is that it can use all the camera observations together to estimate the coordination of the hands. This means that there is much more information available and the module is able to solve the ambiguity problem and thus has more accurate tracking results. Another big advantage is that this module enables a generalized hand tracking scheme. This means that other computer vision modules can be added to improve tracking without additions to the hand tracking algorithm.

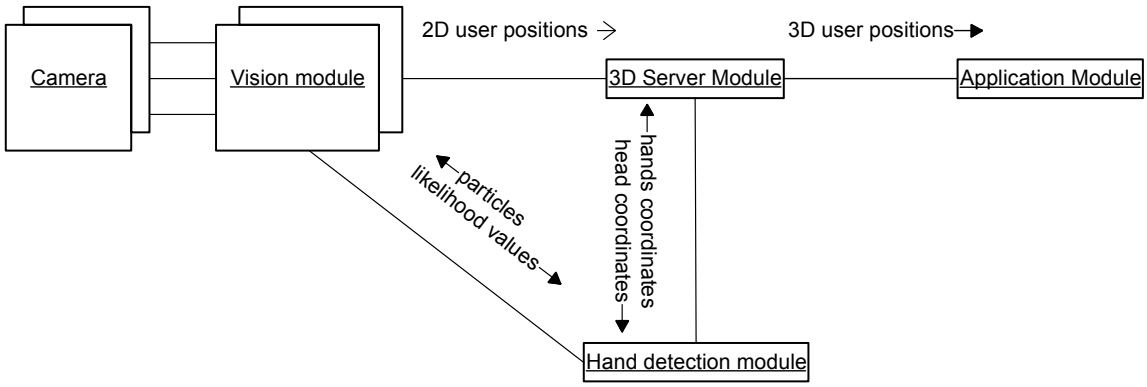


Figure 5 - New Watching Window architecture

The hand tracking algorithm behind the module is going to be a model-based body pose estimation technique. The upper-body pose is estimated because more information of the user's pose is available then information of the user's hands. The model used to estimate the pose is a 3D representation of the user. This 3D model consists of ellipsoids connected to each other and only represents the upper-body of the user. A 3D surface model would be more realistic then the ellipsoids but impossible to generalize and hard to project. The ellipsoids represent a more generalized shape of the user and are easy to project. The big advantage of using a 3D model is that it can be projected on each camera plane. Each computer vision module can then compare the projected 3D model with the camera observation. This implies that a generalized algorithm that tracks the user's hands is possible and all cameras can be used to determine the user's hands coordination.

The literature study tells us that particle filtering is a good way to estimate hidden variables of a system. However when the dimensionality of the state gets higher more particles are needed in order to determine the state. If a particle filter is going to be used a solution needs to be found that solves the dimensionality problem. Navaratnam et al [4] use image features of different body-parts to limit the search space. The best

reliable part is chosen to be an anchor and is used in the rest of the estimation process. Because the head coordination and orientation are known a similar method can be used which uses the head as an anchor. This already greatly reduce the search space however another big reduction in search space is possible if the search space is decomposed into several smaller search spaces. This is possible by breaking the model down and estimating each body-part independently. Each of these body-parts is estimated in a chain, first estimating the root joint of the model and then use the estimated state of this joint as a starting point for the child joints. When the state is estimated in this way, this approach becomes a mixture from top down and bottom up approach. Each body-part is estimated independently, bottom up, but the starting coordination of each body-part is dependant on the parent body-part in the chain, top down. However, this model decomposition has a drawback. When the body-parts are estimated independently after another, they are also projected independently after another. When each body-part is projected independently, self-occlusion can not be projected because the other body-part coordination's are not known. This is obvious because the lower arms are most likely to occlude another body-part but they are estimated last because they are lower in the chain. Nonetheless, the self-occlusion problem is solved partially if more then one computer vision module is used. When more computer vision modules are used the combined observations together give enough information to remove the self-occlusion problem.

Even though the head coordination already reduces the search space greatly both the upper-body method and the decomposed body-part method are tried in an evaluation test. The upper-body method has the advantage that it can project the self-occlusion. However the decomposed body-part scheme reduce the search space enough that with the same amount of particles, much more detailed estimation is possible.

Chapter 4 Design

In this chapter each part of the hand tracking system is explained in detail. The image feature that filters out the user from the camera observation is elaborated first. The following section describes the model in detail by the variables and the appearance. Finally the tracking method is illustrated in the particle filter section.

4.1 Image Feature

Recall that an image feature is a meaningful representation of the camera observation. This image feature needs to represents the user that is filtered out of the camera observation. Because the WW is an interactive system it can be assumed that the user is always moving while interacting with the WW. Another assumption is that the lighting conditions are steady because the whole WW is situated in a booth. According to these assumptions a simple motion-feature would be sufficient to describe the user.

4.1.1 Motion feature

The motion feature is a simple form of an image feature that removes the background in video sequences. The motion feature determines the motion that has occurred during small period (two consecutive frames). This is done by comparing to consecutive frames and filter out the difference between these frames. However there are some drawbacks to this image feature. First the shadows of the user are visible in the motion-feature because the shadows also move when the user is moving. Another problem is the multiple arms. When the time between two consecutive frames is too large, the user's body-parts can move too much. This leads to an image feature that has more then two arms. In Figure 6 the user seems to be having four arms and some interesting underarms. The blue arms are the arms of the user in the current frame, the red arms are the arms of the user in the previous frame and the yellow underarms resulted from shadows.

Figure 6 - Simple Motion Filter

4.1.2 Extended motion-feature

In order to get the right information from the motion-filter an edge filter is added to the motion feature. The edge filter needs to be fast and simple because all computing power needs to be available for other components. The edge filter algorithm compares each pixel with four of its neighboring pixels. The absolute difference between the pixel and its four neighboring pixels is compared with a threshold that determines if an edge is detected. Figure 7 a) represents the end result of the edge feature.

The resulting edge-feature and the motion-feature are combined with a logical AND operation. This states that a pixel needs to be 'on' in both images to be 'on' in the resulting image. The edges of the user are clearly visible and the two arms of the user from the previous frame are removed. If the edge-feature's threshold is determined right, the shadows are removed to because the shadows have a weaker contrast with the background of the booth. The result of the extended motion-feature is shown in Figure 7 b).

Figure 7 - Edge detection and resulting AND'ed image

4.1.3 Parallel lines

To get better results with the extended motion-feature another extension feature is investigated. Ramanan and Forsyth [12] assume that the user's body-parts resemble cylinder like objects in the projected camera observation. With this assumption they detect cylinder like objects by searching for parallel lines in the camera observation. According to this assumption the parallel lines in the extended motion-feature is a better representation of the user's body parts. For this an algorithm that filters out all parallel lines first needs to extract all lines from the extended motion-feature. From this set of lines, the parallel lines are filtered out to form a new image feature.

The lines are extracted with the Hough line detection algorithm. The underlying principle of the Hough algorithm is a transformation from image space to Hough space. Lines are estimated in Hough space and transformed back into image space. The fundamental theory behind the Hough space is that there are an infinite number of potential lines that can pass through any point, each line at a different orientation. A camera observation in Hough space is a summary of all the potential lines in the image.

In order to determine lines from the camera observation, it is necessary to create a representation of a line that is meaningful in this context. In the standard Hough transform, each line is represented by two parameters, commonly called r and θ. The r represents the length and θ represents the angle between a line that is perpendicular to the line in question and the x ax, see Figure 8.

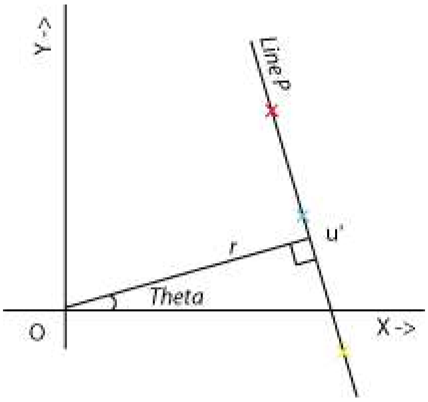


Figure 8 - Line Representation

By transforming all the possible lines that go through a point into this new line representation - i.e. calculating the value of r for every possible value of θ for a point in the image - a sinusoidal curve is created which is unique to that point. When all the points are converted into this sinusoidal Hough space, lines can be detected. Namely a set of points which form a straight line, produces Hough transformation in which the sinusoidal lines cross, see Figure 9. A line is defined a line if the number of pixels that have the same θ and r exceed a certain threshold. This means that there can be several lines with the same θ and a different r.

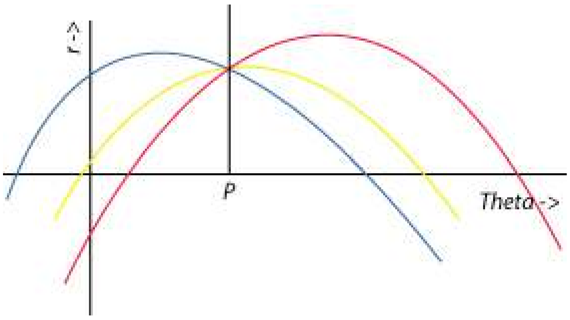


Figure 9 - Hough Space Representation of a Line

When the set of lines is derived from the motion feature, the lines that do not have a parallel equivalent need to be removed. This can easily be done in Hough space by removing all the lines that do not have other lines with the same θ. Figure 10 gives an example of the parallel lines feature.

Figure 10 - Hough Line detection results.
From left to right; the extended motion-feature, all lines and the parallel lines.

The main problem with this approach is the lack in speed. The calculation of the parallel lines costs a lot processing power. Remember that the hand tracking system needs to do this in real-time. In addition, the parallel lines are too unstable to get a reliable result. The Hough transform, depends too much on the threshold that determines a line. If the image intensity fluctuates, e.g. changes in lighting like occlusion of the

light source or shadows, the threshold needs to be adapted. Even a change in the amount of movement gives a completely different result, see Figure 11. Here the threshold figures are the same as in Figure 10.

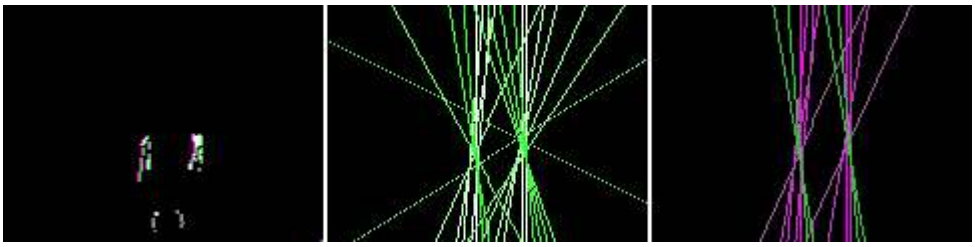


Figure 11 - Bad Hough Line detection

An adaptive threshold would be a solution for this problem, however this is hard to implement and costs even more processing power. For these reasons only the extended motion-feature described in the previous paragraph is used.

4.2 Modeling

If the estimation of the body pose is going to succeed a good representation of the user is needed. This representation of the user is a virtual 3D model which is described by the model parameters namely the variables and appearance parameter. The variables describe the state of the estimated pose of the user and the appearance is used by the particle filter in the likelihood function.

4.2.1 Variables

Recall that the model of the user only represents the upper-body. This upper-body consists of 6 body-parts specifically the head, shoulder, left and right upper and lower arm. Each of these body-parts is linked to another by a joint. For both body-parts and joints, variables need to be defined that define the pose of the model. The next paragraph discusses the body-part variables and paragraph 4.2.1.2 discusses the joint variables.

4.2.1.1 Size determination

One problem in the design of the upper-body model is the determination of the user's body-parts sizes. The pose estimation can only work, if the upper-body model is a good representation of the user and thus if the model matches the user in size. For the determination of the size of the body-parts there are three options. One option is to incorporate the size of each body-part as a hidden variable in the particle filter. This option has the disadvantage that the particle filter has more variables to estimate. This leads to a bigger dimensionality of the search space which in turn means that more particles are needed.

Another option is an initialization procedure in which the size of the user is annotated by an operator. This is obviously the best way of solving this problem. However, this solution needs an operator which is in conflict with one of the requirements. Another initialization procedure does not use an operator but rather a standard pose in which the user starts the interaction. From this standard pose the sizes of the body-parts can be determined. This is also in conflict with one of the requirements that state that the system needs to be intuitive to use. This initialization scheme requires that the user has knowledge about the starting procedure, which is not intuitive at all.

Another option is to obtain some standard proportions of the user and adjust the size of each body-part according to the height of the person. Leonardo da Vinci measured many men and came up with an average, or perfect proportion, between the height of a person and the sizes of the body-parts. According to the notebooks of Leonardo da Vinci [26], the outstretched arms, from finger to finger, equal the height of a person. The shoulders are ¼ of the height of a person which is the same for the underarms with hands stretched out. When both the upper arms are stretched out they occupy one eight of the total height of the person. Each hand is one tenth of the length of a person which is the same as the length of the head from the chin to the hairline. The total height of the head is one eight of the height of the person, see Appendix C. This approach is far from perfect especially because this approach assumes that all users are average, which is never the case. For the hand-tracking module, this is still the best size determination approach because all the requirements still hold.

4.2.1.2 Orientation variables

The joint variables are determined by the particle filter and define each joint's orientation. Each joint orientation is defined by all axes that this joint can rotate around, each representing a DOF. When the joint orientation is known, each body-part can be put in the right place and the hand coordination's can be determined. In addition, when all parameters are known the complete upper body can be projected on the camera plane for the likelihood function.

According to the H-Anim standard [29], all joints are put in a parent child structure. In this parent child structure each joint has a parent and 0 or more children and all joints are children of the human root joint, see Figure 12. Each joint has its own coordinate space which manipulates all children of this joint and the connected body-part(s). And consequently, each joint acts according to the coordinate space of the parent joint. In the H-Anim standard the acromioclavicular (AC) and the vc joints are connected to the vertebral column which is connected to the human root. The AC is the joint that enables the shoulders body-part, to move and the vc joints represent the vertebrates from the neck. The wrist is a child of the elbow which is a child of the shoulder that is connected to the AC.

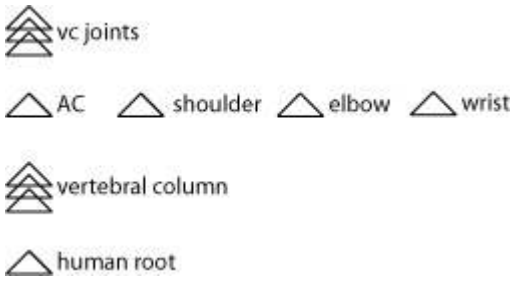


Figure 12 - H-Anim standard joint structure

However, the shoulder body-parts can not move very much and differences in the shoulder orientation are hard to detect in the camera observation. If it is assumed that the shoulder is primarily directed in the same direction as the users gaze while interacting with the WW, the AC can be deleted from the model. Because small differences in the vertebrates in the neck are also hard to detect in the camera observation, the vc-joints are replaced by one single joint, the neck joint. Because only the upper-body model needs to be represented, the vertebrate column is removed to. In this way the upper-body model becomes a simplified H-Anim model. The human root's coordination and orientation is replaced by the head coordination which is known. The upper-body model describes this human root in the neck joint which is connected to the shoulders body-part and the head body-part. In short there are 5 joints which connect the body-parts together. These are the neck, the left and right shoulder and the left and right elbow. The shoulder body-part and the head body-part are connected to the neck, the upper arms are connected to the shoulder joints and the lower arms are connected to the elbow joint.

Each shoulder joint can rotate around all 3 axes, with some constraints in the amount of rotation, and each shoulder joint thus has 3 DOF. Each elbow can rotated around only 2 axes, x- and z-ax, and each elbow has 2 DOF. This means that the model has a total of 10 DOF. The model can be simplified by observing that the rotation of a body-part around the x-ax has no influence on the visualization of this body-part. For this reason the x-ax rotation of the elbow can be deleted which effectively removes 2 degrees of freedom. While the rotation around the x-ax of the shoulder does not have any influence in the projection of the upper arm, the orientation and coordination of the lower arm is affected by this rotation. For this reason no further simplification of the upper-body model can be made and the model consists of 5 joints and 8 DOF. However the rotation around the shoulder x-ax is impossible to estimate because every rotation has the same camera observation. Therefore it is chosen that the shoulder x-ax rotation is determined by the elbow corresponding to the shoulder. In this way both the shoulders and the elbows each have 2 DOF. Appendix C gives a complete representation of the upper-body model.

4.2.1.3 Orientation representation

There are several methods that describe an orientation e.g. Euler, angles axis and quaternions. All these representations can be transformed to a Euclidian transformation matrix representation that is used to manipulate objects in 3D space. The Euler orientation is the simplest form of representing an orientation. Here the orientation is seen as a sequence of rotations around each DOF. The combination of these rotations is described as the complete orientation of the joint. However, the Euler representation suffers from one problem, the gimbal lock. A gimbal lock happens when a rotation over one axis 'overrides' a rotation over another axis, effectively losing 1 DOF.

The axis angle in contrast, represents an orientation by an axis and an angle. With this representation an object is rotated around this axis by the amount of the angle. This representation does not suffer from the gimbal lock but has an interpolation problem. Because of the representation of the axis, during interpolation the joint orientation velocity accelerates and decelerates while the interpolation steps are evenly divided.

The quaternion representation uses a quaternion as the representation of the orientation in 3D space. This quaternion describes the orientation by a 4 dimensional vector which is an extension of a complex number. In this extension all the numbers i, j and k are complex numbers. Equation 8 gives a description of a quaternion in which *w* is a real number, and *i, j*, and *k* are complex numbers [14].

$$Q = w, \{xi, yj, zk\}$$ Equation 8

In order to understand how a quaternion can represent an orientation it must be noted that only a unit quaternion represents an orientation in 3D space. This means that every quaternion needs to be normalized by Equation 9, to be used as an orientation.

$$Q_{unit} = \sqrt{(w^2 + x^2 + y^2 + z^2)}$$ Equation 9

The resulting 4D vector represents a 3D orientation. A big advantage of the quaternion representation is that it does not suffer from the gimbal lock and it does not suffer from the interpolation problem of the axis angle. Because unit quaternions lie on a unit sphere in 4D, the interpolation between two unit quaternions with even step size, result in a steady orientation interpolation. This also explains that the combination of two unit quaternions results in another unit quaternion that represents the combined orientation. In this way a simple multiplication between two quaternions can represent the difference between these two orientations and thus the distance between 2 orientations in a quaternion representation see Equation 10.

$$Q_3 = Q_1 Q_2^{conjugate}$$
$$Q_2 = Q_1 Q_3$$ Equation 10

In which $Q^{conjugate} = w, \{-x, -y, -z\}$ Equation 11

In conclusion the quaternion representation is used for their simplicity in adding, interpolating and calculation of the difference between two orientations.

4.2.2 Appearance

The appearance of each body-part determines the looks of the upper-body model which is used by the particle filter in the likelihood function. To cover the appearance of the model first the physical 3D appearance is discussed, followed by the algorithm that connects the body-parts together. Finally the projection algorithm and the likelihood function are discussed.

4.2.2.1 Physical appearance

The physical appearance of the upper-body model is described in 3D space. In order to use the upper-body model in the pose estimation it needs have some requirements. It needs to be easy to transform, have a simple projection algorithm and above all it needs to be a generic representation of the user. A quadric representation can handle all of these requirements [15].

A general quadric is a surface defined by a second order function which has second powers in its equation. A general form of this equation is:

$$Ax^2 + 2Bxy + 2Cxw + Dy^2 + 2yw + Fw^2 = 0$$ Equation 12

Each quadric *Q* can be represented as a 4x4 homogenous matrix ***Q*** from which the surface is represented as all the points ***X*** that satisfy Equation 13.

$$X^T Q X = 0$$ Equation 13

Different quadric functions represent different surfaces and each body-part is described by a quadric function that represents an ellipsoid. Figure 13 represents the geometry of an ellipsoid that is centered on the center point of the coordinate space. This ellipsoid has a, b and c as the length, height and width of the ellipse respectively.

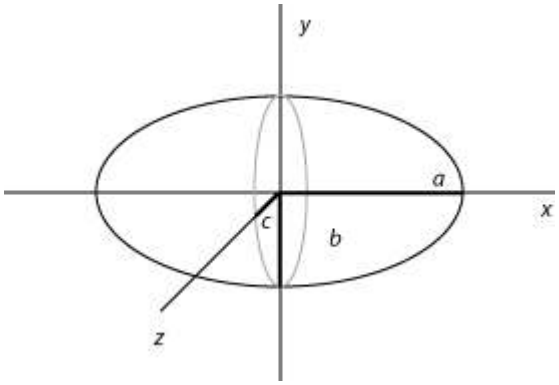


Figure 13 - Quadratic surface of an ellipsoid.

The corresponding quadratic function that represents the surface of this ellipsoid is given by Equation 14 and Equation 15 represents the corresponding homogenous matrix ***Q***.

$$\frac{x}{a^2}+\frac{y}{b^2}+\frac{z}{c^2}=1 \qquad \text{Equation 14}$$

$$Q=\begin{pmatrix} \frac{1}{a^2} & 0 & 0 & 0 \\ 0 & \frac{1}{b^2} & 0 & 0 \\ 0 & 0 & \frac{1}{c^2} & 0 \\ 0 & 0 & 0 & -1 \end{pmatrix} \qquad \text{Equation 15}$$

A quadric can transform under a Euclidian transformation. This means that the quadric's coordination and orientation can be changed while its shape is preserved. Assume that the matrix *T* is a Euclidian transformation matrix. Given a point *X* that lies on the surface of Q, the new transformed quadric Q', on which the transformed point $X'=TX$ lies, is derived as follows.

$$\begin{aligned} & X^T Q X = 0 \\ \Leftrightarrow\ & X^T (T^T T^{-T}) Q (T^{-1} T) X = 0 \\ \Leftrightarrow\ & (TX)^T (T^{-T} Q T^{-1})(TX) = 0 \\ \Leftrightarrow\ & X'^T Q' X' = 0 \end{aligned} \qquad \text{Equation 16}$$

$$\text{in which } T=\begin{bmatrix} R & t \\ 0^T & 1 \end{bmatrix} \text{ and } T^{-1}=\begin{bmatrix} R^T & -R^T t \\ 0^T & 1 \end{bmatrix} \qquad \text{Equation 17}$$

A transformation from quadric Q over a Euclidian transformation matrix *T* is thus defined by Equation 18.

$$Q' = T^{-T} Q T^{-1} \qquad \text{Equation 18}$$

With Equation 18 it is possible to translate and rotate every quadric at the right coordination in the 3D world. The only properties needed are the Euclidian transformation matrixes *T* that put the body-parts at the right orientation and coordination. The next paragraph explains how this transformation matrix *T* is derived.

4.2.2.2 Connecting the body-parts

When the right joint orientation is determined by the particle filter, this joint orientation needs to be used to put each joint in the right orientation. Then each body-part needs to be linked to each joint to connect the upper-body model together.

First the quaternion values need to be converted to a joint orientation matrix *R*. Then the joints are put in the right coordination and orientation according to the parent's joint coordination and orientation [14]. For example the neck joint is defined by the coordination of the head which is represented as a matrix called T_{neck} and the orientation of the neck which is represented by a matrix R_{neck}. The orientation and coordination of the neck joint is described by one matrix calculated by Equation 19.

$$\hat{T}_{neck} = T_{neck} R_{neck} \qquad \text{Equation 19}$$

If the shoulder joints need to be connected to the neck joint there is a difference in positions of both joints. This difference is described by the matrix t_{Lshld}. The final orientation and coordination of the shoulder joint is then determined by first rotating the joint according to its own orientation. Then translate over the physical difference, t_{Lshld}, and finally rotating and translating over the neck joints orientation and translation path, see Equation 20.

$$\hat{T}_{Lshld} = \hat{T}_{neck} t_{Lslhld} R_{Lshld} \qquad \text{Equation 20}$$

If the physical head is connected to the neck joint, there is a difference between the neck joint's coordination and the head coordination which is described by the matrix t_{head}. The final coordination and orientation of the head is determined by first translating head to according to t_{head} and then transform the head to the orientation and coordination of the neck joint. The transformation matrix that puts the head in the right coordination is described by Equation 21.

$$T_{head} = \hat{T}_{neck} t_{head} \qquad \text{Equation 21}$$

These calculations can be made more generalized and are described by Equation 22.

$$T_{bp} = \hat{T}_{joint} t_{bp} R_{joint}$$

where $\hat{T}_{joint} = \hat{T}_{joint\text{-}1} t_{joint}$ Equation 22

The head and shoulder's body-parts orientation point is in the center of the geometry and only needs to be translated to be in the right place. However, the orientation point of the arm body-parts is at the side of the geometry, see Figure 14. Each arm body-part should rotate around this point and the body-parts orientation matrix should be adapted to incorporate this rotation. Let point p in Figure 14 represents the point around which the body-part should rotate. If the body-part is only rotated according to the orientation of the connected joint, the resulting orientation of the body-part is denoted in Figure 14 b). In order to rotate the body-part around p such that p is in the center coordinate space, the body-part needs to be translated over the rotated point –p, see Figure 14 c).

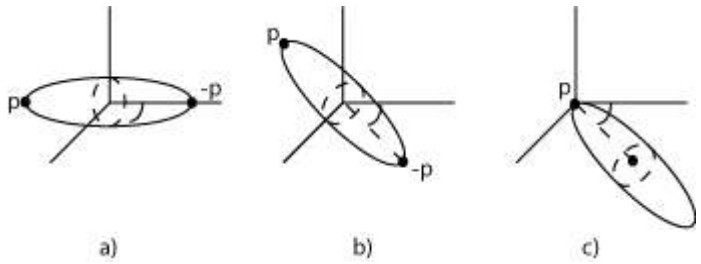


Figure 14 - Rotation of an Ellipsoid

If the transformation matrix z_{bp} is defined to be the point –p rotated by the rotation matrix R_{bp}, a simple additional translation together with the rotation puts the body part in the right orientation and coordination, see Equation 23. If the point –p equals zero, as in the case of the head and shoulder, the transformation matrix z_{bp} has no influence on the orientation and coordination of the body-part.

$$T_{bp} = \hat{T}_{joint} t_{bp} z_{bp} R_{joint}$$ Equation 23

4.2.2.3 Kinematics constraints

If the model is physically realistic and computation power is not going to be used unnecessary, the joints need to be restricted in their movement. A physical realistic representation of the user can not move its joints 360 degrees around. In order to restrict the joints in their orientation, each DOF needs to be restricted and a simple to validate representation of these restriction need to be found. A restriction for 1 DOF is defined by a simple threshold. However, if the number of DOF is more than one a more sophisticated description is needed.

The restriction of each joint is described by Appendix C in terms of radians for each DOF of each joint. These thresholds can not be checked with the orientation quaternion for each body-part. It is hard to convert a quaternion into Euler angles and even if the Euler angles are determined the constraints for these rotations are hard to validate. For example, a combination of 2 rotations can exceed a threshold while the 2 rotations separate do not exceed these thresholds. For these reasons another representation of the kinematics constraints need to be designed.

Besides the matrix representation, a quaternion can be characterized as a vector that points in the direction of the joint orientation. If the joint direction is visualized as a vector, there needs to be a representation that enables a constraint check. A simple constraint representation would be to determine a cone which point is directed to the origin of the coordinate space. If the joint orientation vector is exceeds the cone's surface, the joint constraints are violated and appropriate actions should be taken.

The simplest form of a cone would be a 4 sided pyramid which is constructed with 4 planes. Each plane is defined by one vector that is perpendicular to this plane. The constraint can then be checked by calculating the angle between the joint orientation vector and the plane vector. The joint orientation vector violates the joint constraints, if one angle is smaller then $\frac{\pi}{2}$. However, the joint orientation vector might exceed 1 plane while it does not violate the constraints, see Figure 15.

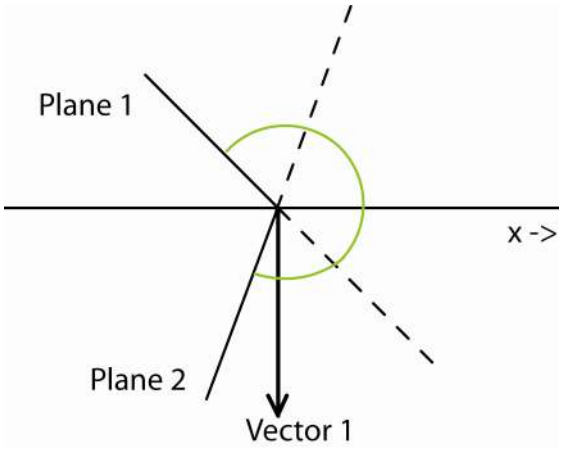


Figure 15 - Joint plane constraint problem

In order to solve this problem, several other planes need to be defined for each constraint plane. The combination of these planes defines an area for which the joint orientation vector should be checked. Each of these planes is again defined by a single vector that is perpendicular to this plane. In order to check if a joint orientation vector exceeds the constraints, the area in which the joint orientation vector finds itself in needs to be determined. When this area is determined the corresponding constraint can be checked. If the vector exceeds this plane the constraints are violated and an appropriate action should be taken. This action is part of the particle filter which is discussed in paragraph 4.3.

With this joint constraint representation, more constraint planes can be added to get more reliable joint constraints. However, this is a trade off between the speed of the algorithm and the accuracy of the joint constraint. When the number of constraint planes rises, more time is spend on the determination of the joint orientation vector's area. In the upper-body model the shoulder joints are checked according to 6 connecting planes each of which area is defined by 3 other planes.

4.2.2.4 Projection

When all body-parts are transposed into the right coordination in world space, they need to be projected on a camera plane in order to be matched with the camera observation. A great property of the quadric is the simplicity in projecting a quadric on a camera plane. The projection of a quadric can be seen as a transformation from world space into camera space and a perspective projection of the quadric Q, see appendix B. The same way that a quadric is translated to right coordination and orientation in the virtual world, a quadric can be translated to the specific camera space. However, a quadric can not be projected according to the normal pinhole projection method described in appendix B.

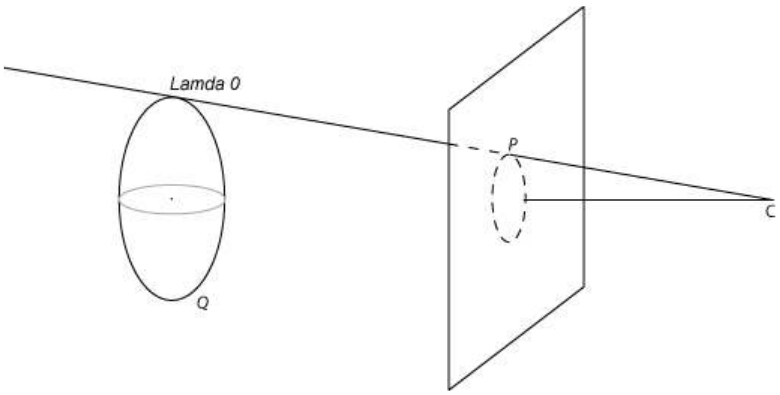


Figure 16 - Projection of a quadric

After the right transformation of a quadric Q into the camera space, a general matrix of a quadric is of the form of Equation 24.

$$Q = \begin{pmatrix} A & b \\ b^T & c \end{pmatrix} \qquad \text{Equation 24}$$

A random ray from the center point of the camera, C, can either go straight through, hit a side or miss the quadric Q. If the ray hits the side of the quadric, there is a certain point λ_0 where the ray is tangent to the edge of the quadric, see Figure 16. This is represented by Equation 25 and Equation 26.

$$X(\lambda_0)^T Q X(\lambda_0) = 0 \quad \text{Equation 25}$$

$$c\lambda_0^2 + 2b^T x\lambda_0 + x^T Ax = 0 \quad \text{Equation 26}$$

When the ray is tangent to the quadric the discriminant of the quadric equation is zero. This leads to the following equation.

$$\begin{gathered}(2b^T x.2b^T x) - 4cx^T Ax = 0 \\ x^T bb^T x - cx^T Ax = 0 \\ x^T bb^T - cAx = 0 \\ x^T (bb^T - cA)x = 0\end{gathered} \quad \text{Equation 27}$$

In other words all pixels *P* on the image plane which satisfy the Equation 28 are on the edge of the projected quadric [16].

$$P^T CP = 0 \quad \text{Equation 28}$$

$$\text{in which } C = bb^T - cA \quad \text{Equation 29}$$

This also means that if $P^T CP > 0$ the point *P* hits the quadric at more points, which means that the point *P* hits the body of the quadric. This also means that if $P^T CP < 0$ the point *P* is outside of the quadric. This new matrix C is the 2D representation of the ellipsoid projected on the image plane. When the upper-body model is drawn on the image plane, for each body-part every pixel *P* on the image plane, Equation 28 needs to be solved. This method is computational heavy. An alternative method first iteratively defines the outline of each body-part and then fills in the body-part according to the outline. If the center of each body-part is projected according to the pinhole projection this algorithm has a starting point which decreases the search space of the outline of the ellipse.

4.2.2.5 Self occlusion

When the upper-body is going to be projected, self occlusion can be projected. Self-occlusion happens when one body-part occludes a part of another body-part in the camera projection. In order to handle the self-occlusion the projection of the body-parts, should take the self-occlusion into account.

In order to project the self-occlusion each body-part first needs to be translated into camera space. Then the z-value of a body-part determines the distance from this body-part to the camera centre point C. This z-value is stored in a buffer for each body-part. When the upper-body is drawn onto the camera plane, the body-parts with the smallest z-value are drawn first. The first body-parts are the body-parts that are closest to the camera. Each overlapping pixel in a later drawn body-part is not drawn on the camera plane because this pixel is occluded by the previous drawn pixel.

A better solution would be to determine the z coordination of each overlapping pixel on the camera plane. This method costs heaps of computing time. While the method described before is not very accurate it is simple to compute and therefore the self-occlusion method used.

4.2.2.6 Likelihood function

The likelihood function describes the likelihood that the projected upper-body model is represented in the image feature. Because of the design of the extended motion-feature, it can be assumed that all pixels from the extended motion-feature describe the movement of the user. More specific the extended motion-feature describes the outline of the moving body-parts of the user. Therefore, the likelihood function needs to encourage the outline of the projected upper model in the image feature and in contrast, the likelihood function needs to discourage the body of the projected upper-body model in the image feature.

The first property of the likelihood property compares the edge of each projected body-part with the image feature. In this comparison an edge is determined to be the relation between the edge of the ellipse and the surrounding nothingness. This means that an edge is defined by 2 pixels the first being on the edge of the projected body-part and the second being outside the projected body-part. In order to calculate this likelihood property the amount of pixels that define the edge of the projected body-part is determined. Then each edge of the projected body-part is matched with the image feature and the first likelihood property, L_{edge}, is determined to be the number of matched edge pixels, divided by the total number of edge pixels. The edge pixels that are occluded by another body-part are excluded from this likelihood property.

The second likelihood property compares the body of each projected body-part with the image feature. Each pixel in the body of the projected body-part that compares to a pixel in the image feature needs to make this body-part less likely. Therefore, the second likelihood property, $L_{notinside}$, is determined by the number of pixels from the body of the projected body-part that do not compare with the image feature divided by the total number of pixels from the body of the projected body-part.

If the whole upper-body is projected, a third likelihood property can be defined. This property determines the likelihood that the whole upper-body is represented in the image feature while the latter two properties described the likelihood that a projected body-part is represented in the image feature. The third likelihood property discourages any pixels in the image feature that does not compare to the edge of projected upper-body. This likelihood property counts the number of pixels in the image feature that do not form an edge on the projected body-part. L_{body} is determined by the total number of pixels in the extended motion-feature divided by the number of matched edge pixels.

These three properties together can give the likelihood of the projected upper-body model. The complete likelihood function is used by the particle filter and elaborated for each specific particle filter in the next section.

4.3 Particle filter

The particle filter is the tracking algorithm behind the hand-tracking module. The goal of the particle filter is to estimate the current state of the user's upper-body pose, given the input from the computer vision modules and the model described in the previous sections. The state of the particle filter is determined by the 8 DOF of the model, described in quaternions. The specific particle filter algorithm that is used to determine the upper-body pose of the user is the SISR particle filter discussed in the literature study and summarized in algorithm 1.

Algorithm 1 – SISR particle filter

```
For each Particle in the population
    Propagate forward according to the transition model
    Assign a weight according to the likelihood
End for

State estimation

Resample
```

A particle filter's design depends on several properties namely the transition model, the likelihood function and the state estimation property. The first describing the transition from one state to another, the likelihood function determines the probability that the particle is represented in the camera observations and the state estimation determines the final state of the system. Because a SISR particle filter uses a resampling step the resampling step needs to be defined. The final property that needs to be defined is the initialization of the particle filter.

Because of the dimensionality of the upper-body model, 2 different tracking algorithms are investigated. The first particle filter scheme uses the upper-body as a state of the system. The second particle filter uses a decomposed upper-body model and estimates each joint separately to reduce the search space. Both tracking schemes use the SISR particle filter described above. The initialization and transition function are equivalent and are therefore described in the next paragraphs. The resampling step is also equivalent and is governed by the systematic resampling algorithm described in the literature study paragraph 2.3.4. However, the likelihood function and state estimation differs for each particle filter scheme and each is described separately for each particle filter scheme.

4.3.1 Initialization

A particle filter assumes that the initial state of the system is known and that the initial particles are drawn from a uniform distribution given the initial state. However, the requirements of the project state that the system must be intuitive to use which means that an automatic initialization scheme is needed.

The initialization scheme used for the particle filter starts with an adapted resampling algorithm. This resampling algorithm focuses on getting a population with a high likelihood. The difference between the systematic resampling and the adapted resampling algorithm is that the latter can end up with a different weight set.

The initialization of each particle filter starts with a random orientation for each joint in each particle. These random values are within the constraints of the joint to which these values are assigned. After the first particle filter steps, the adapted resampling algorithm determines the new particle population, X'_k, by removing all the particles from the current population, X_k, which have a weight difference bigger then 1/N in comparison with the best likely particle. For each particles removed from the population, a new random particle is chosen. After the first iteration of the particle filter, the adapted resampling algorithm removes all particles that do not influence the PDF. The particles that do influence the PDF stay in the population and in the same time the new population is spread out. When the best particle's likelihood estimation is bigger then

a threshold, the systematic resampling takes over. This threshold determines if the best particle is a good estimation of the camera observation.

This initialization scheme works because it keeps the particle with the highest weight while creating variance in the particle set for the particles that do not have any influence. Even if none of the particles resemble the camera observation, the complete population is spread out until the some particles have a good likelihood. Another advantage of this initialization scheme is that the adapted resampling algorithm can also be used when the particle filter looses track of the user. Then the particle with the highest weight falls below the threshold and the adapted resampling algorithm is used.

4.3.2 Transition function

The transition function describes the transition probability from the state at time *t* to the state at time t_{+1}. There are many ways that determine this transition probability. The best transition function would be a linear function that can be solved however, in body pose estimation a linear function from one state to another is impossible to create because human body-part movement is far from linear. Another transition function uses a prediction model that predicts the next state from the previous state. This prediction model can be created from recorded data and probabilistic methods like an N-gram system can give the most probable state at time k_{+1} of a system given the previous state.

However, for the upper-body model, there was not enough time to capture the user data and free available data was not found. Therefore a transition function that uses the velocity of the joint orientations is used. This comes from the observation that a joint orientation direction does not completely change direction. A joint orientation is more likely to rotate in the same direction with some acceleration. From this observation, the transition function of each joint is determined by the estimated orientation's velocity. The distance between the estimated orientation at time k-1 and time k is calculated in quaternions, Equation 30. This can be seen in Figure 17 where the joint orientation difference between to consecutive particle filter estimates, represents the orientation velocity of this joint.

$$\mathbf{Q}_k^{bp} = Q_k^{bp} \overset{conjugate}{Q_{k-1}^{bp}} \qquad \text{Equation 30}$$

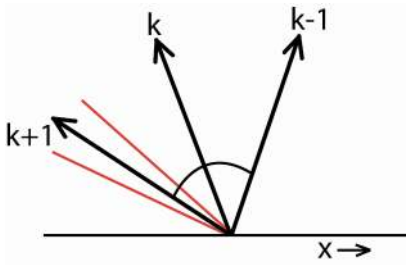


Figure 17 - Particle transition function

The velocity of the joint is added to the current orientation to get the most probable next state of the system. If the estimated state of the joint changes in direction the velocity also changes direction. This means that this function preserves the variety in the population and that the complete population set constantly moves according to the estimated state.

However, this means that the user is assumed to have a linear movement which is never the case. Therefore the acceleration of the joint is modeled in the transition function by adding a Gaussian noise which resembles the probability of the acceleration. This Gaussian noise is represented with a variance of 1 because this resembles a complete acceleration or deceleration of the joint velocity. This Gaussian noise is multiplied with the joint orientation velocity which is added to the current joint orientation to get the most probable joint orientation. The transition function can be calculated in quaternion space and is determined by Equation 31, in which *V* resembles the noise component.

$$\begin{aligned} \widehat{\mathbf{Q}}_k^{bp} &= \left(\mathbf{Q}_{k-1}^{bp} V\right) \\ Q_{k+1}^{bp} &= Q_k^{bp} \widehat{\mathbf{Q}}_k^{bp} \end{aligned} \qquad \text{Equation 31}$$

In this equation a value multiplied with a quaternion is a multiplication of this value with each of the components of the quaternion except the real value. When the quaternion is normalized, the new quaternion represents an in- or decrease in the angle while the orientation keeps its direction.

However, the joint orientation velocity has an influence on the child of this joint. This means that the child joint needs to rotate according to the parent joint. For example imagine the actions needed when grabbing an object from a table. Not only the lower arm moves in the right direction but also upper arm moves in the

right direction. This implies that each joint's movement is influence by the parent joint's movement. This influence can be modeled in the transition function by adding a part of the parent joint's transition. This parent orientation velocity is calculated according to a multiplication of the parent's velocity and a draw from a Gaussian distribution. The resulting quaternion calculation is shown in Equation 31 in which *H* represents the noise component from the parent joint's orientation velocity.

$$\hat{\mathbf{Q}}_k^{bp} = \left(\mathbf{Q}_{k-1}^{bp} V\right) \mathbf{Q}_{k-1}^{bp-1} H \qquad \text{Equation 32}$$

It must be noted that this parent joint orientation velocity can also be used in the decomposed particle filter. The decomposed particle filter estimates each joint after another in the parent child chain. This means that the orientation velocity of each parent joint is known.

4.3.2.1 Constraint handling

After each transition the joint constraints need to be validated. If a joint orientation violates the constraints an appropriate action should be taken. The maximum joint orientation in which the joint can move has to be calculated. This constraint is the orientation, in the direction of the joint orientation velocity, in which the joint can maximal move. In order to calculate this orientation, the angle, α, in which the joint orientation is exceeding the joint constraint is calculated. This is done by calculating the angle between the joint orientation vector and the constraint plane vector. This value only represents how close the joint orientation vector is to the constraint plane vector. In order to get the angle α, it needs to be subtracted from π. The angle β determines the angle of joint's orientation velocity. The angle between this quaternion orientation and the null orientation determines this angle β. The angle that the joint orientation can move maximal is thus defined by λ = β – α in the direction of the joint orientation velocity. The maximum joint orientation in quaternions is then calculated by Equation 33.

$$\hat{\mathbb{Q}}_k^{bp} = \frac{\lambda}{\beta} \mathbf{Q}_k^{bp} \qquad \text{Equation 33}$$

The new joint orientation can then be calculated in the same way the transition function is calculated. The difference now is that the draw from the uniform distribution is now multiplied with the maximum joint orientation see Equation 34.

$$\hat{\mathbf{Q}}_{k-1}^{bp} = \left(\hat{\mathbb{Q}}_{k-1}^{bp} V\right) \qquad \text{Equation 34}$$

4.3.3 Upper-body particle filter

The upper-body particle filter uses the complete upper-body model, described in section 4.2, as a particle. The state of each particle is described by the joint orientations of the shoulders and elbows. Globally the particle filter propagates all particles forward according to the transition function. All particles are transmitted to each computer vision module currently available, who determines the likelihood value for each particle. These values are transmitted back to the particle filter which uses these combined likelihood values of all computer vision modules to determine the pose of the upper-body. When the pose is determined, the coordination's of both hands are determined. Finally the current population of particles is resampled before a new iteration starts.

4.3.3.1 Likelihood function

The likelihood function determines the likelihood value of each particle according to the camera observation. In this specific case, the combined likelihood value of all computer vision modules is used as the likelihood value of this particle.

In order to explain the likelihood function the particle evaluation flow is discussed first. First every particle is put in the right place according to the joint orientations. Then the particles are projected for each computer vision module. This projection is the determination the C matrix described in paragraph 4.2.2.4. Each projected particle is transmitted to each corresponding computer vision module which draws it on the camera plane. From the drawn particle, several different likelihood values are calculated which are transmitted back to the particle filter. Because a particle in this particle filter scheme is a representation of the complete upper-body, all three likelihood categories discussed in paragraph 4.2.2.6 are used. The combined categories likelihood value determines the likelihood value for each computer vision module. When the likelihood values of each computer vision module is determined the final likelihood value is estimated according to the average value of all camera likelihood values, see Equation 35 in which *c* denotes the number computer vision cameras currently in use.

$$\mathbf{L}^i = \frac{\sum_{j=0}^{c} L_j^i}{c} \qquad \text{Equation 35}$$

in which $L_j^i = \left(\sum_{bp=0}^{3} \mathbb{L}_{bp_j}^{i\ edge} \mathbb{L}_{bp}^{i\ \neg body} \right) \mathbb{L}_j^{i\ body}$

4.3.3.2 State estimation

The estimation of the pose is hard because an infinite number of particles or a similar amount of particles is impossible to use due to the likelihood estimation, which requires a projection and comparison with the camera observation. Tests need to determine the right amount of particles which is a trade off between the speed of the module and the detail of the estimation. However, if it is assumed that the number of particles is somewhere between 50 and 500 particles it can be concluded that the normal state estimation, which uses an average in relation with weights, does not suffice.

The right option would be to single out all the different particle groups that follow different local maxima and choose the best likely local maxima to be resemble the state of the system. The estimation of the state then becomes a state estimation of this local maxima group. However the determination of each local maximum is very computational in a high dimensional search space. Therefore the average, in relation to the weight, particle is used as the final state estimation. Another method chooses the particle with the highest weight to represent the state of the system but this option is also not optimal. Therefore in an evaluation test both methods are tried out to see which method works best.

4.3.4 Decomposed upper-body particle filter

The decomposed upper-body particle filter exists of several particle filters that work in a chain to estimate the upper-body pose. Each particle filter estimates a joint's orientation according to the SISR particle filter algorithm. This particle filter scheme decomposes the upper-body tracking problem in several body-part tracking problems. This decomposition of the upper-body model reduces the number of particles needed dramatically or more detailed estimation is possible with the same amount of particles.

4.3.4.1 Likelihood function

The particle filters are executed in the parent child structure of the joints. Because the lower arms are most likely to occlude other body-parts and they are estimated as last, occlusion can not be projected on the camera plane. However occlusion is partially solved by the use of multiple cameras. Because more cameras give more information about the user, the combined camera observation gives a good representation of the occlusion. Because each joint is estimated separately the likelihood value of the complete upper-body can not be determined. This means that the likelihood value for each camera in Equation 35 becomes Equation 36 for the decomposed particle filter scheme.

$$L_{bp_j}^{i} = \mathbb{L}_{bp_j}^{i\ edge} \mathbb{L}_{bp_j}^{i\ \neg body} \qquad \text{Equation 36}$$

4.3.4.2 State estimation

In the case of the decomposed upper-body particle filter, it is possible to estimate the optimal state. Because each particle represents only one joint orientation each different local maximum can be determined. Each local maximum is then defined by a set of particles, which estimated state differs sufficiently from other particle set's estimated state. All local maxima can be found using an algorithm that iteratively searches the complete particle set. In this algorithm each set is expanded with a new particle from the original set. When all the local maxima are defined, the local maximum with the highest combined weight resembles the current state of the system. In contrast with the state estimation of the upper-body particle filter scheme, only this method is used because it is the optimal solution to estimate the state.

4.3.5 Tracking problems

4.3.5.1 Zero movement

The zero-movement problem occurs when the user, or one of the body-parts of the user, is not moving. When the user is not moving the image feature does not represent the user. Therefore the particle needs to detect if the user does not move and adjust the rest of the particle filter to this knowledge. Zero-movement can be detected by analyzing the weights of the particles in the population. If the most likely particle's weight becomes almost zero, the user, or the corresponding body-part, is not moving. This can be checked with a threshold that determines the probability of the particle being still. When the zero movement is recognized the estimated state of the system stays the same and no resampling is done. In the next particle filter cycle the transition function assumes that the user is not moving and the particles are not propagated forward.

4.3.5.2 Communication

In the current architecture the 3D server is in the centre the WW architecture. All the clients log in on the 3D server and ask for the services of the 3D server. The hand-tracking module is situated on the same level as the 3D server and needs the services of the computer vision modules while the XML-RPC protocol can only

handle one-way communication e.g. a computer vision module can only ask for a service of the XML-RPC server, not the other way around.

Another problem of the XML-RPC protocol is the speed of the protocol. The speed problem became obvious when a large number of particles were transmitted to a computer vision module. When 400 particles were transmitted, 100 for each body-part, it took ~40 seconds before the particles where communicated. The lack in speed of the XML-RPC protocol comes from 3 problems. First, all numbers are converted to text which is a costly process. Secondly, each piece of information is surrounded by two tags with the name of that piece of information. This results in lot of overhead in the message that needs to be transmitted. And finally, when a lot of data is transmitted, the algorithm converts all information to text iteratively which means that there is one buffer that keeps expanding when a piece of information is added. This result in 3 memory allocations per piece of information added to the XML message. When there are a lot of numbers involved this allocation costs a lot of time.

The speed was not really an issue when sending the 2D coordination's of the head e.g. the message only consisted of two integers and a procedure name. However, when the C matrix is transmitted together with two variables per body part, for 4 body-parts per particle and at least 100 particles, 4400 floats are needed. In order for the particle filter to work, the number of iterations needs to be "real-time", e.g. 20 fps which means that 88000 floats per second need to be transmitted every second. Because each float needs to be converted into text, each float is described in 8 Bytes surrounded with two tags each at least 8 Bytes. This results in sending 1375 KB per second to each camera. Compare this with sending only the numbers, which are 4 Bytes per float, resulting in only 343,75 KB to each camera per second.

The speed problem was partially resolved by a method that created 1 buffer with all the information. The size of the buffer was determined beforehand which only uses 1 memory allocation. This buffer was given to the XML-RPC protocol which only needed to put 2 tags around it and communicate it to the right client. The resulting speed of the communication protocol with 400 particles for 3 cameras was a satisfying 10 fps.

This still did not solve the second problem of the directionality of the protocol. The intermediate solution to this problem was a polling method on the computer vision modules side which is not preferred because of the extra network load. In the last month of the project the networking protocol was rewritten by another internship student. In this new implementation only the bytes of the floats where communicated over the network which removed 4 extra bytes of sending it in text data. This already gave a big speed improvement, e.g. ~15 fps, however the big change was that the server was able to send requests to a specific computer vision module which removed the polling of the computer vision module. The new complete algorithm is presented in Appendix E.

Part IV Conclusions

In this chapter the hand-tracking module is evaluated according to 2 evaluation tests. The evaluation methods are discussed first, followed by the results. A final conclusion is made by relating the results of the evaluation with the requirements. Finally some propositions on future work are made.

Chapter 5 Evaluation

In this chapter, the hand-tracking module described in the previous sections, is evaluated. The evaluation tests need to give the answer to the research question and check if the requirements are maintained. Therefore 2 evaluation tests are performed that combined give a complete overview of the hand tracking module. The main questions that this evaluation chapter tries to answer are:

- How accurate does the hand tracking system track both hands of the user?
- What is the speed performance of the system?
- Do the requirements still hold?

5.1 Accuracy

The accuracy of the system is divided into 2 parts:

1. How accurate is the hand tracking system?
    a. What is the distance between the estimated hand coordination and the real hand coordination?
2. How accurate is the upper-body pose tracking?
    a. What is the difference between the estimated pose of the user and the real pose of the user?
    b. How often does the module loose track of the upper-body?
    c. If the module looses track of the upper-body, why does is loose track?

The accuracy evaluation test is done offline because the module needs to save the results of the particle filter estimation in order to compare it with the camera observation. In addition several different setups need to be tried out with the same sequence in order to define the best particle filter scheme. For this offline accuracy evaluation 4 video sequences are recorded each with a different user. The video sequences are recorded at 18 fps to get real-time representation of the user's movement. The users from the video are asked to perform a few simple movements like move both arms up and down, forward and backward etc. Then the users are asked to visualize and manipulate a virtual ball of clay floating in front of them. The resulting movement is equivalent to the movement that the user makes while interacting with the virtual clay ball. This results in real movement to which the system can be compared.

5.1.1.1 Annotation Program

In order to compare the resulting estimates of the hand-tracking module with the camera observation a program is created which helps to annotate the real coordination's. This program enables a user to annotate the coordination's of the joints in the original data and the estimated data from the hand-tracking module. The joint coordination's need to be annotated in each camera observation, in order to triangulate the 3D coordination's of each joint. The poses are only annotated every ± 50 frames to make the annotation feasible. From the resulting 3D joints, the corresponding body-part vectors are calculated and the distance between the estimated hand and the real position of the hand is calculated by subtracting the real hand coordination from the estimated hand coordination. The size of the resulting vector represents the distance between the coordination of the estimated hand and the real hand. The differences between the estimated pose and the real pose is determined by the angle difference for each body-part. Therefore each body-part is converted to vectors and the angle between the estimated body-part and the real body-part is calculated.

In order to test how good both the measurements are, a null sequence is created. In this null sequence for both the real user pose and the estimation pose the same coordination's are annotated in the program. The goal of this null sequence is to define the standard error in manually annotating the body poses. This is used in the evaluation to relate the results to the real evaluation values. The average difference in the body pose is 5,077365^{o} and the average distance between the hands is 2,3831356 cm. The detailed graphs can be found in appendix F.

The annotation program is also capable of annotating the tracking property of the particle filter. This annotation is done for every 20 frames in the complete sequence and determines if the particle filter is still tracking. If the particle filter is tracking the complete particle population is surrounded around the real

position of the user's body part. In contrast, the particle filter is not tracking if the particle population is randomly scattered around. With these combined results each particle filter scheme is compared and the outcome determines the final setup of the hand tracking module.

5.1.2 Results

In the modules design 2 different particle filter schemes are described, each having its pros and cons and the best particle filter scheme needs to be determined with an accuracy test to define the final hand tracking method. In order to compare the different schemes, each scheme estimates the upper-body pose of the same camera observation sequence. The scheme with the best accuracy is chosen to be the final setup of the system and is used in the final accuracy evaluation.

5.1.2.1 Setups

Before the results are presented the differences between the different schemes need to be discussed. These are the upper-body particle filter scheme (A) and the decomposed upper-body particle filter scheme (B). For A there are 2 different setups each estimating the state of the system differently. One setup uses the average particle, with respect to the likelihood value, as the state of the system while the other uses the most likely particle as the state representation.

There is of course a trade off between the number of particles that are used and the speed of the complete system. During several tests with the module the best result (speed and accuracy) was noticed to be around 150 particles. The threshold value that determines if the particle filter is tracking the upper-body and thus determines the resampling algorithms set to 1,7 while the zero-movement threshold is set to 1,2.

5.1.2.2 Results

Figure 18 gives an example of the output for each of the camera observation. The top pictures denote the complete particle population while the bottom pictures resemble the estimated pose of the user. This picture is taken from the decomposed particle filter scheme. After the first upper-body particle filter scheme evaluations it became clear that the state estimation that uses the most likely particle as the state is really bouncy. The estimated state jumps from one place to another while the average value state estimation behaves smoothly. Because the bouncing state estimation is not desirable, only the average state estimations are considered in the rest of the comparison.

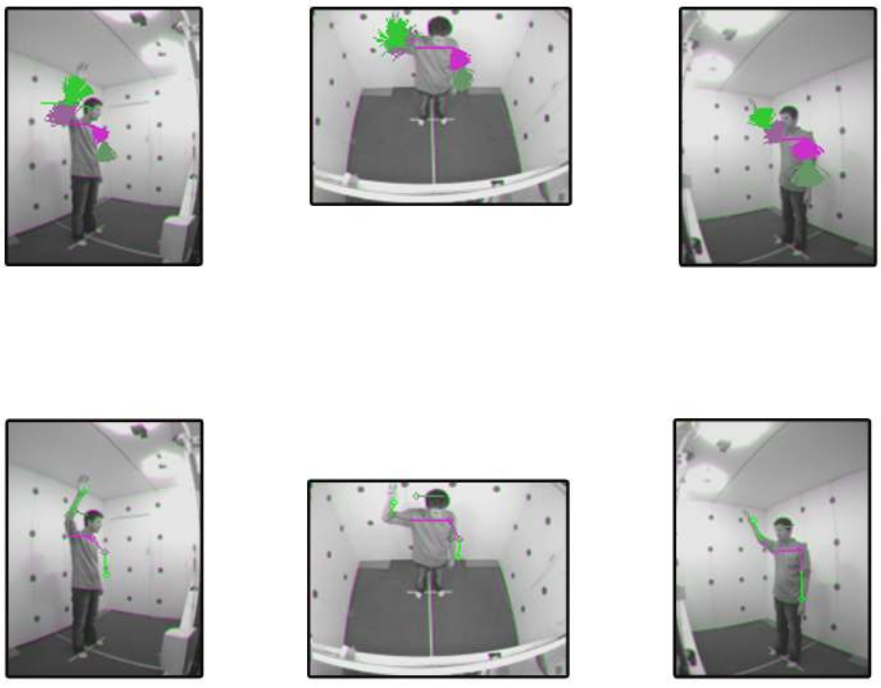


Figure 18 - Example body pose tracking

The hand tracking results are summarized in Table 1 and Table 2 and more detailed information of each particle filter scheme can be found in appendix F.

Table 1 – Average distance in CM

	Average distance in cm	Standard deviation
A	24,767186	25,585963
B	11,37441	9,117545

Table 2 – Distance in CM

Average distance in cm	0-5 cm	0-10 cm	0-15 cm	0-20 cm
A	14,70588%	29,41176%	55,882353%	58,82353%
B	30,55556%	58,333333%	77,77778%	86,111111%

In both tables a clear distinction between both particle filter schemes is visible and the decomposed upper-body particle filter is clearly better. In order to point out the difference between both particle filter schemes, the upper body estimation of each scheme is summarized in Table 3 to Table 5.

Table 3 – Average distance in degrees

	Average distance in degrees	Standard deviation
A	41,25354	46,08328
B	17,07226	19,31595

Table 4 –Distance in degrees

Average distance in degrees	0-5^{o}	0-10^{o}	0-15^{o}	0-20^{o}
A	10,29412%	30,8824%	44,11765%	54,411765%
B	15,27778%	44,44444%	66,66667%	73,611111%

Table 5 - Tracking

	Tracking	Numbers of losses
A	75	3
B	83,33333%	2

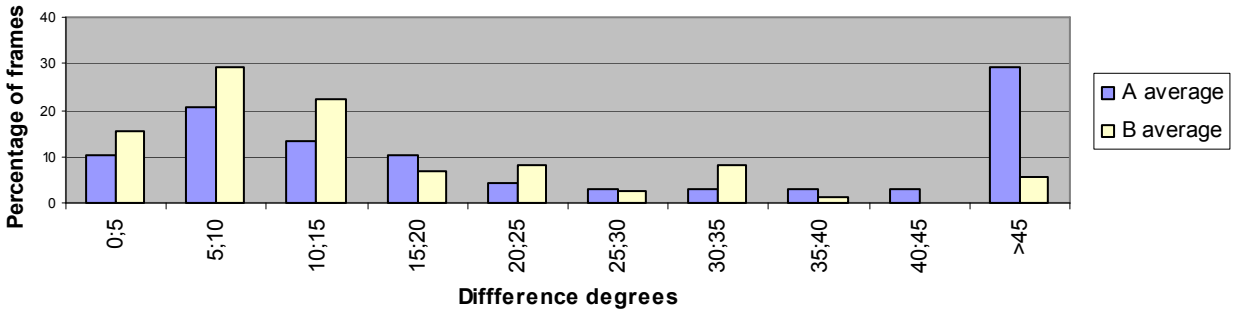
From these tables the conclusion can be made that the decomposed upper-body particle filter scheme is more accurate then the complete particle filter scheme, 10% against 15% between 0 and 5 degrees. It is also visible from Table 5 that the complete particle filter scheme looses track more often then the decomposed upper-body particle filter scheme.

The decomposed particle filter looses track when the shoulder looses track and when the shoulder particle filter looses track, the corresponding elbow particle filter looses track. The shoulder looses track because the upper arm is hard to observe, e.g. it has the same color as the torso of the user. This results in zero movement which can not be represented in the extended movement feature.

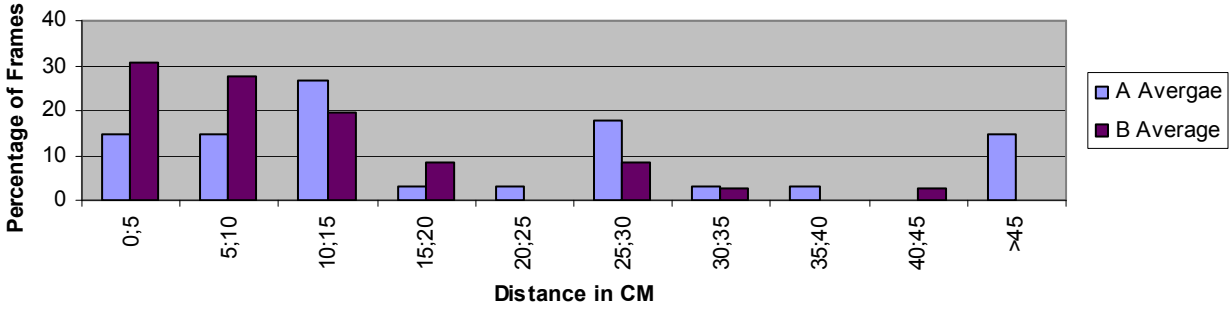
The upper-body particle filter looses track of the user when one arm of the user does not move. When the user does not move an arm the particle likelihood drops. The particle filter tries to maximize the likelihood which results in the estimated arm drifting off in the wrong direction. Because in the evaluation sequence in a few cases only one arm was moving the complete upper-body particle filter scheme continually lost track of the user.

Graph A and Graph B shows another view of the same evaluation. Here the view represents the percentage of frames of each setup that falls between certain degrees. From these views it is clear that the decomposed upper-body particle filter scheme tracks both hands more continuously.

Graph A – Difference in degrees



Graph B - Distance in cm



5.1.2.3 Overall results

The previous paragraph shows that the decomposed upper-body particle filter scheme works best therefore this particle filter scheme is used in the final accuracy test. For this test the other 3 video sequences are estimated and annotated and the results of each annotated video sequence can be found in appendix F. The annotated results are averaged in order to summarize the accuracy test and can be found in the following tables and graphs.

Table 6 –Average Distance

	Mean	Standard Deviation
Degrees	24,08291	24,30338
CM	14,22683	10,5657

Table 7 –Distance

	0-5	0-10	0-15	0-20
Degrees	9,210526%	28,15789%	48,42105%	59,21053%
CM	15,78947%	40%	66,31579%	76,84211%

If these results are compared with the null sequence it becomes clear that the average distance is not that far off. In relation to the null sequence, 40% of the estimated frames are between 8 and 12 cm of the real position of the hands. Recall that this is averaged for both hands over 4 video sequences of each consisting of ± 1150 frames each.

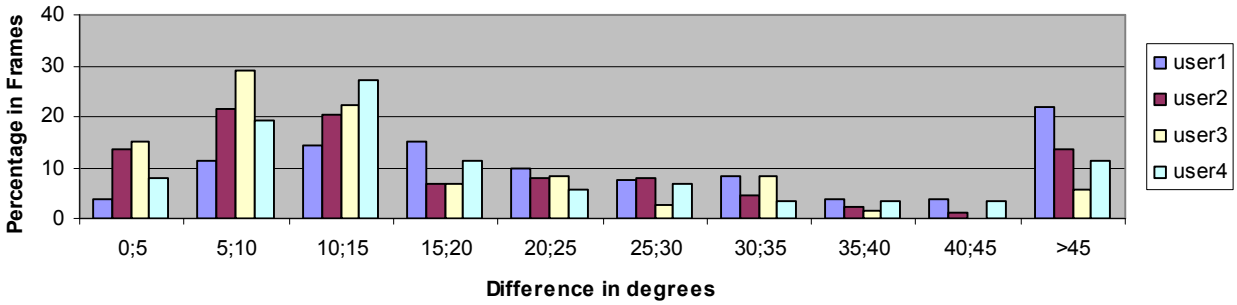
One thing that stands out is that the final accuracy test is not as good as the accuracy of the video sequence used in the previous paragraph. This comes from one video sequence in which the user performs a Tai Chi sequence. This Tai Chi sequence consists of very small movement which is not detected by the motion feature. Because the motion was not detected by the motion feature, the particle filter lost track continually which resulted in a bad estimated sequence.

Another thing that needs to be noted is that the performance of the gesture sequence, e.g. moving the arms up and down, waving, etc, outperforms the estimation of the clay deforming sequence. This also comes from the small movement when the user is interacting with the virtual clay ball. This small movement is not detected by the image feature while the movement of the arms to the starting position is detected by the

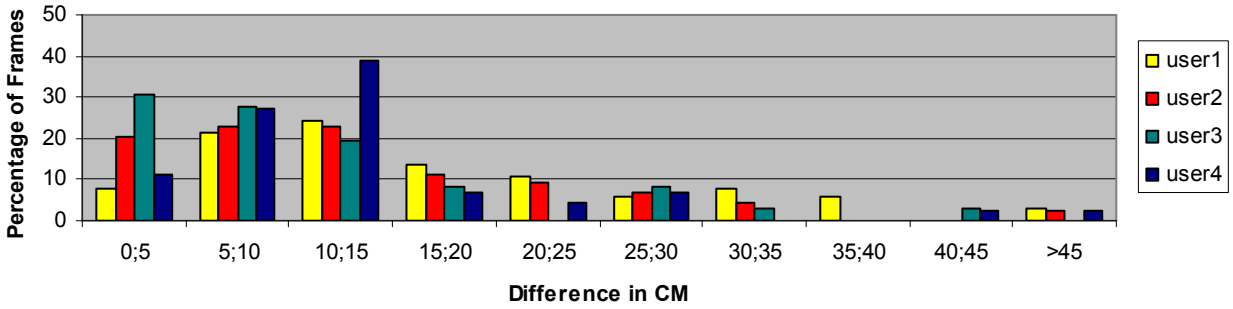
image feature and tracked by the particle filter. Because the starting coordination is known by the particle filter, the average distance does not become large and therefore the overall estimation is not punished. However, tracking in this small movement is not taking place.

Another thing that strikes is that the size estimation is not good enough because the estimated size of one user and the real size of the user do not match. This results in a fixed difference between the real coordination and the estimated coordination of the hands. This fixed difference can be seen in the next graphs of user 4. The difference in cm in the hands almost never comes close to zero while the estimated pose is always reasonable.

Graph C – Distance in degrees



Graph D - Distance in CM



Graph E - Average distance

5.1.2.4 Conclusion

In conclusion, from the accuracy evaluation it can be concluded that the main accuracy is reasonable. The tracking occurs steadily however for really detailed tracking more improvement is necessary. This means that for the clay deforming application, the hand tracking module is not accurate enough. This arises from the small movements that the user makes in detailed object handling. However, for button clicking, object handling and pointing applications the hand tracking module is accurate enough because the movement used in these applications is big enough.

5.2 Speed results

In order to answer the evaluation question regarding the speed of the hand tracking system, the following question needs to be answered:

1. Is the speed performance of the system fast enough for smooth user interaction?

This question is answered by running the system and track of the performance of the different particle filter parts to see where the bottlenecks of the system are. The results of this evaluation can be found in appendix D and is divided into 2 parts. The first part evaluates the operating speed of the hand tracking system in the WW architecture while the second part evaluates the operating speed at one computer. In the former evaluation 50 particles are used while in the latter 150 particles are used. In these speed performance tests, the decomposed particle filter scheme used in order to compare the speed performance with the accuracy results.

From these results it is visible that the repositioning and projecting of the particles takes more time with more computer vision modules. Therefore, the computer vision modules need to wait longer to get the particles (which takes almost 50% of their time). Another bottleneck in the system is the drawing and evaluation of the particles. In conclusion the complete projection of the particles forms the main bottleneck in the hand tracking module.

The overall speed of the hand-tracking module in the WW architecture is with 5,6 fps not fast but well enough for hand tracking. However, this speed performance only uses 50 particles and thus is not as accurate as in the evaluation of the previous sections. The performance of the hand tracking system with 150 particles on one computer is with 5,1 fps also good. The main difference between the performance differences is the fact that the particles do not physically have to be transmitted to other computers.

5.2.1.1 Conclusion

In the WW architecture the speed is sufficient but more particles are needed. However when more particles are needed the speed performance of the system goes down which in turn has a devastating effect on the accuracy of the hand tracking system. This arises from the fact that if the speed performance goes down, the time interval between 2 particle filter cycles becomes to high which in turn makes it impossible to predict the transition function. However, the speed performance of the hand tracking system can be improved by better hardware which is shown in the second speed performance evaluation.

Chapter 6 Conclusions

This chapter concludes to the work presented in this thesis. First a summary of the thesis is presented followed by the summarized conclusions of the system in relation with the requirements. Finally future work is proposed which can improve the hand tracking module described in this thesis.

6.1 Summary

This thesis is written for the closure of the Master degree Human Media Interactions and describes a hand-tracking module for the Watching Window of the University of Otago. The module is designed to track both hands of the user and enables the user to use his/her hands in VR applications. A module is proposed that fits into the current WW architecture and uses all available cameras to estimate the upper-body pose and the coordination of both hands are derived from this estimated upper-body pose. A 3D model-based pose estimation method is used which enables a projection on all camera planes. This enables a generic likelihood function and thus uses all available cameras to infer the pose of the user.

A particle filter is used to track the upper-body pose of the user over time. The module proposes 2 different particle filter schemes which are compared in an evaluation test. The first particle filter scheme uses the complete upper-body of the user as a state of the system while the other particle filter decomposes the search problem by searching for each individual body-part separately. The decomposed particle filter turned out to work best and this scheme is used in the hand tracking module.

From the evaluation of the system it is clear that the accuracy of the module is accurate enough for applications that only use pointing, gestures or object handling. If detailed tracking is needed, e.g. in the virtual clay application, this hand tracking module is not sufficient. The speed of the module is to slow in the current WW architecture but when hardware updates are preformed the hand tracking system works with acceptable speed.

6.2 Conclusions

In order to conclude if the hand tracking system presented in this thesis is succeeded the requirements of the project are discussed in relation with the evaluation results.

Good accuracy and speed.
The hand tracking system works at a reasonable accuracy which is good enough for pointing and manipulating objects. However when more detailed interaction is needed, e.g. drawing or deforming clay, the hand tracking system does not work well enough because the movement in such a detailed interaction is to small to be filtered by the extended motion feature. An improvement in the image feature may solve this problem.

The speed of the system is currently good enough when the hardware of the system is updated. The evaluation results show a 5,1 fps which is high enough for smooth interaction with objects in the virtual environment.

No physical changes to the user.
The property that the user should not need any physical changes is still maintained. This comes from the global 3D model that is used and the automatic size determination of the user. This automatic size determination does not work well enough. The assumption that all users are average is never right. However, even though the sizes do not always fit, they are not far off which means that the module still tracks the upper body but the estimated hand coordination is always at a fixed distance of the real hand coordination. This implies that for pointing and manipulating applications this size determination is sufficient.

User independent.
The hand tracking system is also user independent. All users can use the system because of the global model that is used. The module is skin color, size and clothes independent. However, this requirement also suffers from the size determination discussed in the previous paragraph.

Intuitive to use.
Because no initialization procedure is used, the module is intuitive to use e.g. the user can come in and the module starts tracking. However the application that the user is using needs to be intuitive to use to. A good feedback of the users hand coordination or upper-body pose can be used to make it clear that the user can use his/her hands.

From this section it can be concluded that both hands are tracked with the hand tracking module described in this thesis. Even though the system works there are still some improvements that can be done to get better accuracy and speed performance.

6.3 Future work

One great improvement lies in the speed performance of the system. When the speed of the system improves more particles can be used or better likelihood estimations can be used. The speed improvement can be gained in the projection algorithm because this is still the bottleneck of the whole module. One solution that improves the projection algorithm lets the video card project the particles because the video card is optimized for projecting and rendering 3D objects.

Another improvement lies in the image feature because the currently used extended motion-feature is not good enough when small movement is being made. One improvement is an increase in the image observation size when very detailed movement is happening. However when the size of the image observation is increased, the projection and evaluation of the particles will cost more time.

Another improvement makes the head tracking algorithm part of the hand-tracking module. Enabling the head tracking algorithm to benefit from the same advantages as the upper-body particle filter namely combined camera observation, a generic algorithm and a solution to ambiguity.

Part V Appendices

Appendix A *The Watching Window definition*

The mock-up of the WW

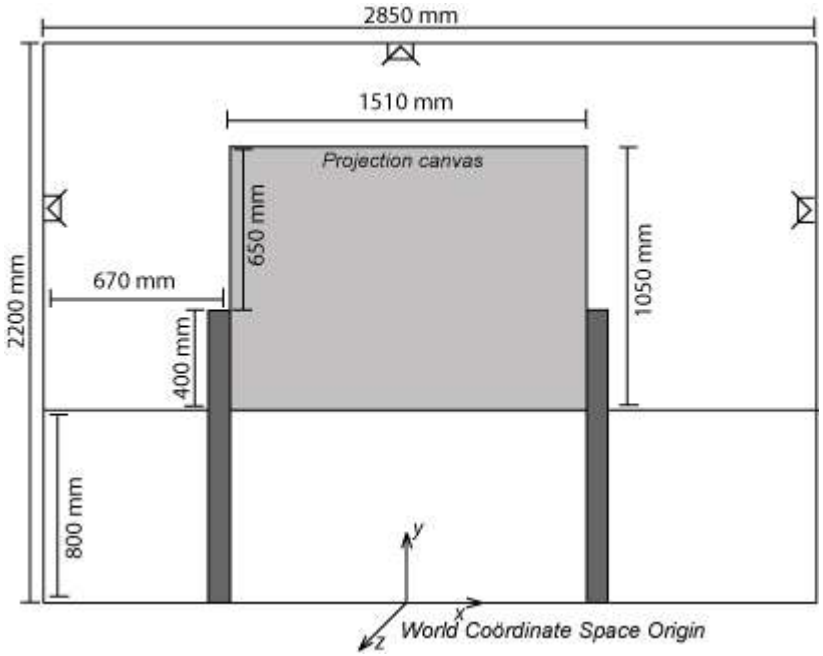


Figure 19 - Mock- up of the WW screen

Camera calibration files

Ceiling camera

This is the camera that hangs on the ceiling of the booth and looks down straight at the user.
Image Width = 640, Image Height = 480 [pixels]
Focal distance f = 444.273376 [mm]
The first Taylor coefficient for the distortion kappa1 = 2.187971e-006 [1/mm^2]
Translation vector (16.182921, -1218.654973, 2322.946311) ^{T} [mm]
Rotation values in degrees Rx = -56.219719, Ry = 2.455825, Rz = -1.335191 [deg]

Rotation matrix $\begin{pmatrix} 0.998810 & -0.022650 & 0.043186 \\ -0.023280 & 0.556689 & 0.830395 \\ -0.042849 & -0.830412 & 0.555499 \end{pmatrix}$

The horizontal scaling factor Sx = 1.037047
Principal point C = (333.459571, 240.954616)^T [pixels]
Camera Location in World Space O = (55.002249, 2607.781380, -279.127965) ^{T} [mm]

Right camera

This camera is the camera that hangs on wall with the door of the booth. For the user this is the left camera, however it is called the right camera from the perspective of the operator which is sitting behind the screen.
Image Width = 640, Image Height = 480 [pixels]
Focal distance f = 379.473768 [mm]
The first Taylor coefficient for the distortion kappa1 = 2.736282e-006 [1/mm^2]
Translation vector (1444.893629, 743.974924, 1037.472235) ^{T} [mm]
Rotation values in degrees Rx = 7.431532, Ry = -58.327393, Rz = 82.675733 [deg]

Rotation matrix $\begin{pmatrix} 0.066938 & -0.997542 & 0.020700 \\ 0.520781 & 0.017234 & -0.853517 \\ 0.851062 & 0.067913 & 0.520654 \end{pmatrix}$

The horizontal scaling factor Sx = 1.036270
Principal point C = (320.343403, 262.032805) ^{T} [pixels]
Camera Location in World Space O = (-1367.119028, 1358.063220, 64.921325) ^{T} [mm]

Left Camera

For the user this is the right camera, however it is called the left camera from the perspective of the operator which is sitting behind the screen.

Image Width = 640, Image Height = 480 [pixels]

Focal distance f = 380.345847 [mm]

The first Taylor coefficient for the distortion kappa1 = 3.203158e-006 [1/mm^2]

Translation vector $(-1350.195260, 824.911723, 1043.953462)^T$ [mm]

Rotation values in degrees Rx = 3.350987, Ry = 55.997712, Rz = -88.260526 [deg]

Rotation matrix $\begin{pmatrix} 0.016975 & 0.999301 & -0.033304 \\ -0.558968 & -0.018133 & -0.828991 \\ -0.829015 & 0.032688 & 0.558270 \end{pmatrix}$

The horizontal scaling factor Sx = 1.044189

Principal point $C = (329.840409, 229.709931)^T$ [pixels]

Tz / f = 2.744748

Camera Location in World Space $O = (1349.472737, 1330.084560, 56.069743)^T$ [mm]

Appendix B *Camera calibration*

There two sorts of camera parameter, the intrinsic and extrinsic parameters. The former includes the cameras internal parameters i.e. the principle point C in camera space, the scaling factor and the lens distortion which convert a point of the image into the real point on the camera plane and visa versa. And the latter includes the coordination and orientation of the cameras in world space which is needed for the conversion to the real world and visa versa.

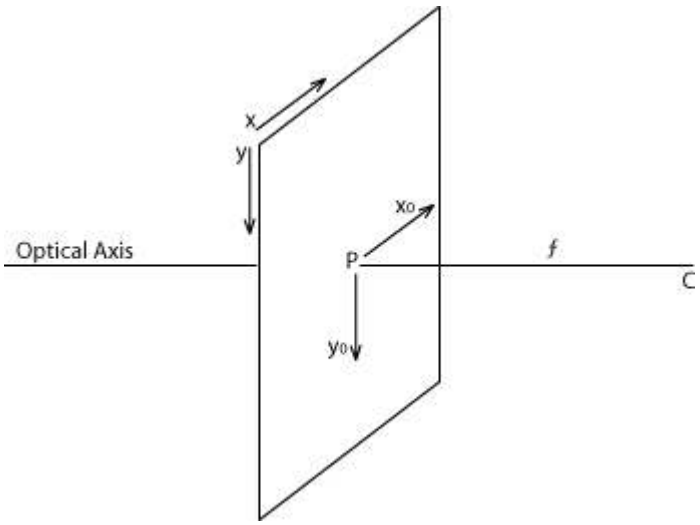


Figure 20 - Camera model

The camera calibration is done according to the well-known Tsai [27] method. The first parameters needed are the principal point P = (X_0, Y_0) and the camera point C that is defined by (X_0, Y_0, f) ^{T}. The camera point C lies on the optical z-axis of the camera, see Figure 20. The optical axis z is perpendicular to the plane on which the image is projected and goes trough the points x_0 and y_0. The principal point C lies on this line at the focal distance f.

A perspective projection is ruled by the pinhole model. In relation to the pinhole model the point C is the hole through which the scene is projected on the image plane. This means that every point on the image goes through this point. With these points a ray can be created that goes into the real world until it hits the point in the scene that it represents on the image. A big complication with the camera calibration is the horizontal scaling factor, S_X. This factor needs to compensate the difference between the horizontal spacing of the pixels in the sampled image from the frame grabber and the horizontal spacing between the sensor points. There is a difference between these two because before the frame grabber grabs the image the signal that comes from the sensor goes through a low pass filter that hides the transition of the sensors. This means that the right coordination of the sensor points is not definitely known.

For a camera with a lens, the ideal pinhole projection does not hold. Some distortion happens when the point in the image plane is different from the point that is calculated by the pinhole model. It is possible that the real point is closer to the principal point (pillow distortion) or further away from the principal point (barrel distortion). The distortion is in a radial form increasing to the outside of the image plane. See Figure 21 for an example of this.

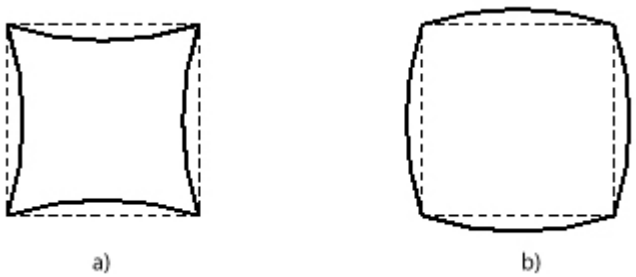


Figure 21 - Distortion a) pillow distortion b) barrel distortion

When it is assumed that the radial distortion comes from the principal point on the plane we can model this distortion according to the following Taylor expansions.

$$\delta x = x\left(k_1 r^2 + k_2 r^4 + ...\right) \text{ and } \delta y = y(k_1 r^2 + k_2 r^4 + ...) \qquad \text{Equation 37}$$

In which x and y are measured from the principal point and usually only the first two of the power series are maintained. In conclusion, the intrinsic camera parameters consists of the camera pinhole point C, the horizontal scaling factor S_X and *k*appa 1 and *k*appa 2 values.

The extrinsic parameters are the parameters that describe the relation between the camera space and the world space. The transformation from world to camera space consists of a translation and a rotation. If r_W is a point in world space and r_C is a point in camera space then each point can be converted from world to camera space by $r_c = R(r_w) + T$. This can be done in a homogenous matrix multiplication according to Equation 39.

$$\begin{pmatrix} x_c \\ y_c \\ z_z \\ 1 \end{pmatrix} = \begin{pmatrix} R_{00} & R_{01} & R_{02} & T_0 \\ R_{10} & R_{11} & R_{12} & T_1 \\ R_{20} & R_{20} & R_{22} & T_2 \\ 0 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \qquad \text{Equation 38}$$

To calibrate the cameras a geometry that is known is needed in the world. The object on the image plane can be compared with the real position the object to obtain the intrinsic and extrinsic camera parameters. In the WW there are fixed calibration objects in the booth. On all three of the visible sides of the booth there are several calibration dots. These calibration dots are red dots the size of CD's and the 3D coordination's of these dots in the booth are known by measurements. The measurements are from the dots to the corners of the screen and the real world coordination's are calculated by interpolating these measurements. See Appendix A for a description and measurement of the booth and see [35] for the calculations of the 3D coordination's from the measurements.

From and to camera space

If the camera parameters are known, these parameters are used to project a 3D object from the virtual world onto the camera plane. In addition, if for multiple cameras the camera parameters are known, a 3D point's coordination can be triangulated by the points from the camera planes that resemble this point. In order to estimate the 3D coordination of multiple 2D cameras coordination's, the camera plane vector $(X_C, Y_C)^T$ that corresponds to this point needs to be warped to the real camera space $r_c = (u, v, f)^T$ vector. This is done by first subtracting the principal point from the camera vector. Then the distortion and scaling factors are applied to the resulting vector. Finally the focal distance *f* is added as the z-value for the resulting camera space vector. When these coordinates are obtained for both the cameras, the 3D coordination is triangulated. In order to determine the 3D coordination from these coordinates, each vector is traced into world space. The point where both lines intersect is the 3D coordination of the annotated point. However, the lines never really intersect each other at a certain point because of small errors. The real point is then decided to be the point halfway in between the two points on the smallest line between each ray.

The projection of a point *U={X, Y, Z}* can be seen as a transformation from the world space into the camera space, by the inverse camera translation and orientation matrix T_C, followed by the projection to the camera plane. Suppose the point *u = {x, y}* is a 2D point on the camera plane then the perspective projection is done according to the following equation.

$$x = \frac{fX}{Z} \text{ and } y = \frac{fY}{Z} \qquad \text{Equation 39}$$

This is converted to a matrix calculation by multiplying the point *U* with the homogenous perspective matrix ***P*** see Equation 40.

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ 1 \end{pmatrix} = \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \qquad \text{Equation 40}$$

Now this coordination needs to be de-warped to the real image coordinates. This de-warping is done according to the principal point, scaling factor and radial distortion. Resolving the radial distortion and subtracting the principal point compute the final 2D coordination.

Appendix C *The user model*

Model

The model of the user consists of 6 body-parts. These are the Head (HEAD), Shoulder (SHLD), Right Upper Arm (RUA), Left Upper Arm (LUA), Right Lower Arm (RLA) and Left Lower Arm (LLA).

These body-parts are linked together with 5 joints. The first joint is the head coordination and orientation which is derived from the other components. The other joints are the Right Shoulder (Rshld), Left Shoulder (Lshld), Right Elbow (Relbow) and the Left Elbow (Lelbow).

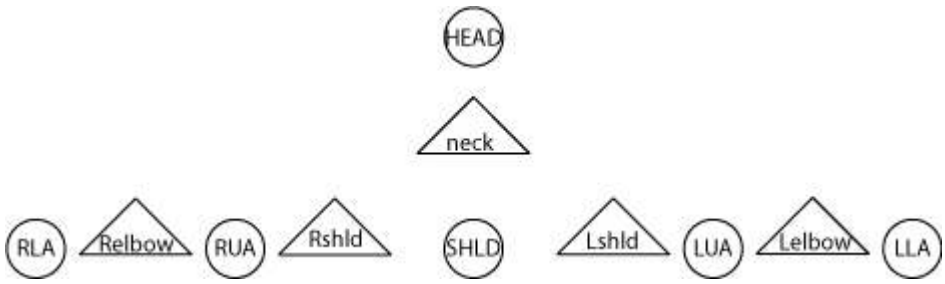


Figure 22 - Kinematics Constraint

Figure 22, shows the body-parts in relation to the joints and Figure 23 shows the sizes of the body-parts in relation to the height of the user.

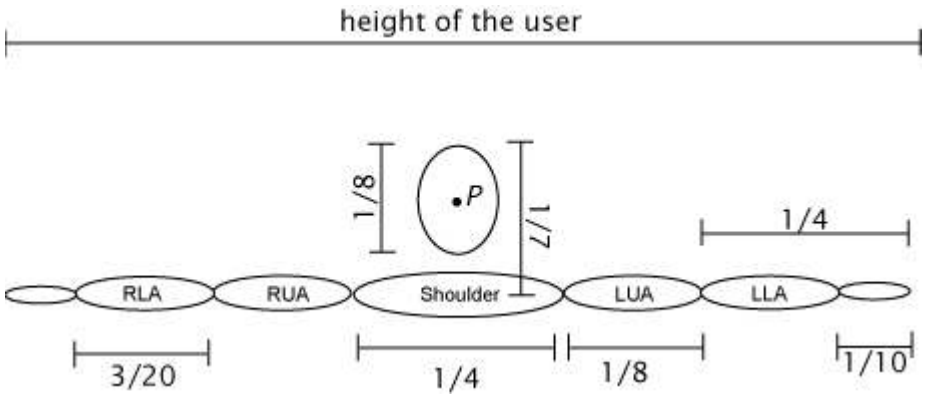


Figure 23 - Model of the user

Kinematics constraints

Table 8 describes the kinematic constraints for every body-part and is based on observations.

Table 8 - Kinematics constraints

	Y-ax	Z-ax
Left Shoulder	-2/3 π to 1/4 π	-1/2 π to 1/2 π
Right Shoulder	-1/4 π to 2/3 π	- 1/2 π to 1/2 π
Left Elbow	-π to 0	-π to 0
Right Elbow	0 to π	0 to π

Appendix D *Hardware Statistics*

WW architecture speed performance

Computer statistics

The visualization computer, called puck, has the following characteristics:

- Pentium III Xeon 930MHZ
- 512MB
- XP 2002 service pack 2

Al computer vision modules have the same characteristics namely:

- Pentium III 800MHz
- 256 MB
- XP 2002 service pack 2

Speed performance

The speed performance is evaluated in different setups to determine the bottlenecks in the hand-tracking module. For each setup the operating performance is given together with profiler information of each part of the system.

1 server and 1 computer vision module running on Puck at ~16.5 fps

Computer vision module:

Name	Time per iteration	Percentage
Image Filters	0.004344	4.64%
Waiting for the server	0.005757	36.92%
Drawing the best particle	0.000360	2.31%
Evaluation of the particles	0.008482	54.38%
Sending results	0.000273	1.75%

Particle Filter:

Name	Time per iteration	Percentage
Propagate particles forward	0.001046	25.28%
Reposition and project particles	0.002739	66.25%
Combine the evaluation	0.000016	0.39%
Normalize the evaluation	0.000129	3.13%
Estimate PDF	0.000157	3.80%
Resample the particles	0.000048	1.16%

1 server and 2 computer vision modules each running on different machines ~8 fps

Computer vision module:

Name	Time per iteration	Percentage
Image Filters	0.008063	5.18%
Waiting for the server	0.013136	50.63%
Drawing the best particle	0.000467	1.80%
Evaluation of the particles	0.010735	41.38%
Sending results	0.000264	1.02%

Particle Filter:

Name	Time per iteration	Percentage
Propagate particles forward	0.001227	26.25%
Reposition and project particles	0.003080	65.90%
Combine the evaluation	0.000027	0.58%
Normalize the evaluation	0.000060	1.27%
Estimate PDF	0.000110	2.36%
Resample the particles	0.000170	3.64%

1 server and 3 computer vision modules ~ 5.6 fps

Computer vision module:

Name	Time per iteration	Percentage
Image Filters	0.005846	1.86%
Waiting for the server	0.042510	81.05%
Drawing the best particle	0.002264	0.72%

Evaluation of the particles	0.008299	15.82%
Sending results	0.000289	0.55%

Particle Filter:

Name	Time per iteration	Percentage
Propagate particles forward	0.001794	13.76%
Reposition and project particles	0.010420	79.89%
Combine the evaluation	0.000070	0.53%
Normalize the evaluation	0.000082	0.63%
Estimate PDF	0.000158	1.21%
Resample the particles	0.000518	3.98%

Single computer speed performance

Computer statistics

- AMD Athlon 64 bit 2 GHz
- 512 MB
- XP professional service pack 2

Vision Speeds

In this evaluation test the hand tracking system is tested on a single computer with better hardware. The total operating speed of the module is 5,1 fps and the following tables show the results of particle filter algorithm and the computer vision modules algorithms.

Computer vision module:

Name	Time per iteration	Percentage
Image Filters	0.007333	7.64%
Waiting for the server	0.008790	54.95%
Drawing the estimated state	0.000100	0.43%
Evaluation of the particles	0.007072	30.70%
Sending results	0.001446	6.28%

Particle Filter:

Name	Time per iteration	Percentage
Propagate particles forward	0.003205	22.56%
Reposition and project particles	0.010218	71.95%
Combine the evaluation	0.000079	0.56%
Normalize the evaluation	0.000066	0.46%
Estimate PDF	0.000368	2.59%
Resample the particles	0.000268	1.88%

Appendix E *Architecture*

Hand-tracking module

A new module is going to solve the hand-tracking problem. This module is situated on the same level as the 3D server and uses the computer vision modules to get the likelihood value of each particle. In order to give a complete picture of the module an overview of the current modules and the changes is given. The new module is shown in Figure 24.

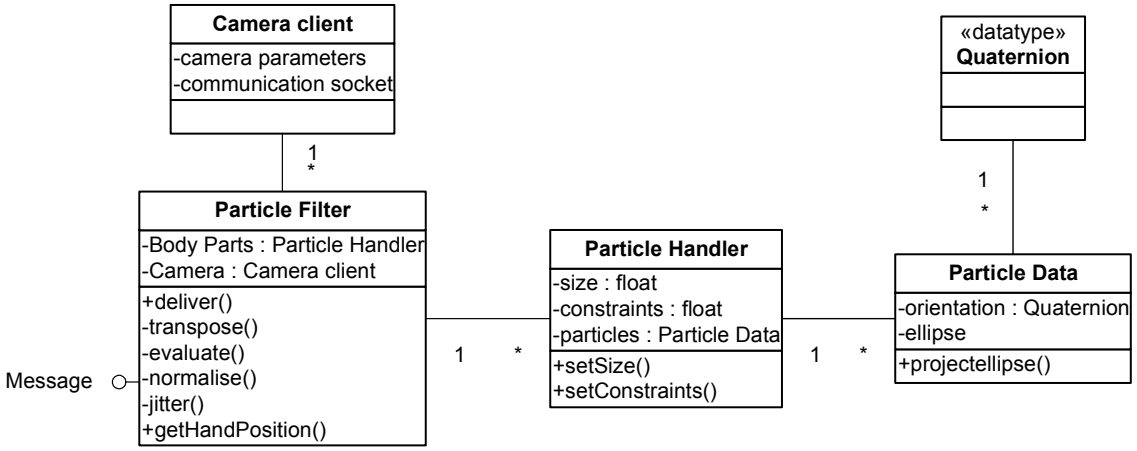


Figure 24 - Hand-tracking module

In short, the particle filter has several particle handlers. Each of the particle handlers represents a body-part. Each of these particle handlers has multiple particles for which the values are stored in the particle data class. The particle filter then executes the particle filter operations and sends the result to the 3D server. In addition the particle filter needs one or more camera clients in order to function. These camera clients are vital for the evaluation of the particles. These camera clients have an interface that connects them to the computer vision modules. This interface is the main XML RPC architecture, which was replaced by another communication protocol. This new communication protocol changed the communication model used in the hand tracking system, Figure 25 represent this new communication model.

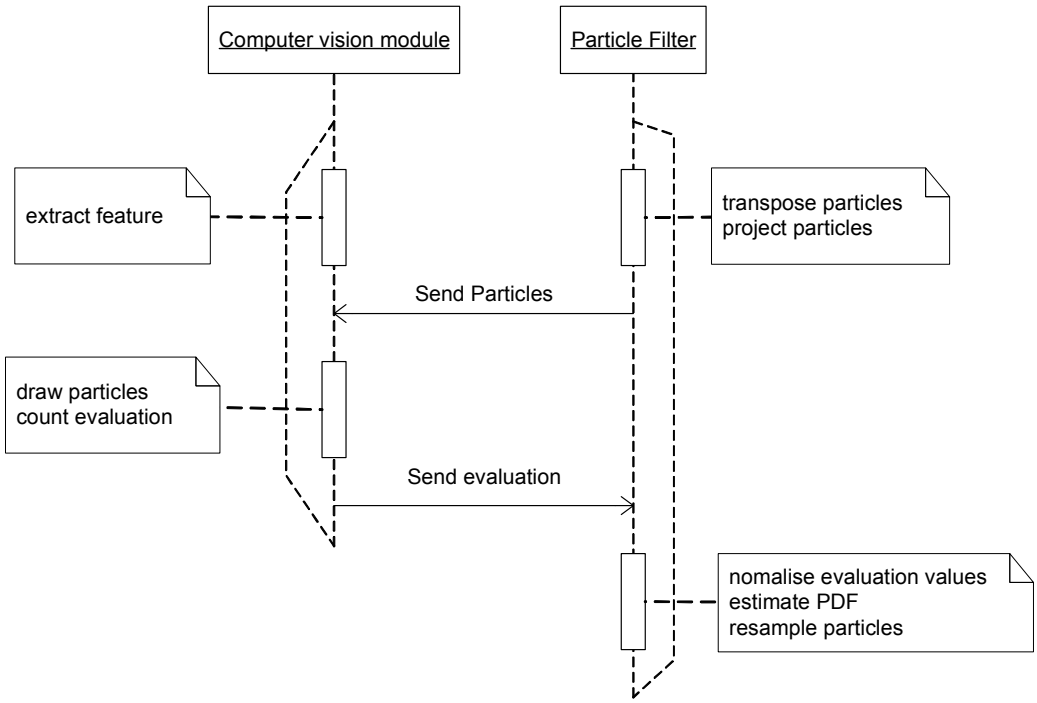


Figure 25 - Communication model

Computer vision module

The computer vision module is constructed around a frame grabber class and the virtual class “display-mode”. The frame grabber is the component that captures the frames from the camera. The module is setup in such a way that it is possible to try out multiple computer vision algorithms in an easy way. Each computer vision algorithm is situated in a separate class that is extension from the “display-mode” class. The GUI lets

the user choose the right display-mode so the user can choose the computer vision techniques used to track the user's body-parts. Every cycle a new frame is grabbed from the camera by the frame grabber. This frame is used by the current display-mode to estimate the body-parts of the user at this current time. The results from the display-mode are transmitted to the 3D server through the communication protocol.

Changes

In the new situation a new display-mode is added to the computer vision module. This display-mode filters the camera observation with the right feature and evaluates the particles according to the resulting feature. This new display-mode needs to work side by side with the display-mode that tracks the coordination of the head of the user. The coordination of the head is vital for the estimation of the rest of the upper-body. For this reason 2 new display-mode are introduced. The new cycle of the computer vision module adds a new display-mode action in each cycle after the 3D server communication.

3D Server

In the old situation the 3D server had one thread that handles the incoming communication. When a service was needed from the 3D server, the client asks the XML RPC interface for the function. The XML RPC interface server thread calls the right method from the 3D server class, the result of this method is transmitted back to the right client. In the new architecture the interface with the 3D server is still the same. The only difference is that the communication interface is changed into the Message interface and the hand-tracking module is a client of the 3D server class.

Differences for the computer vision modules

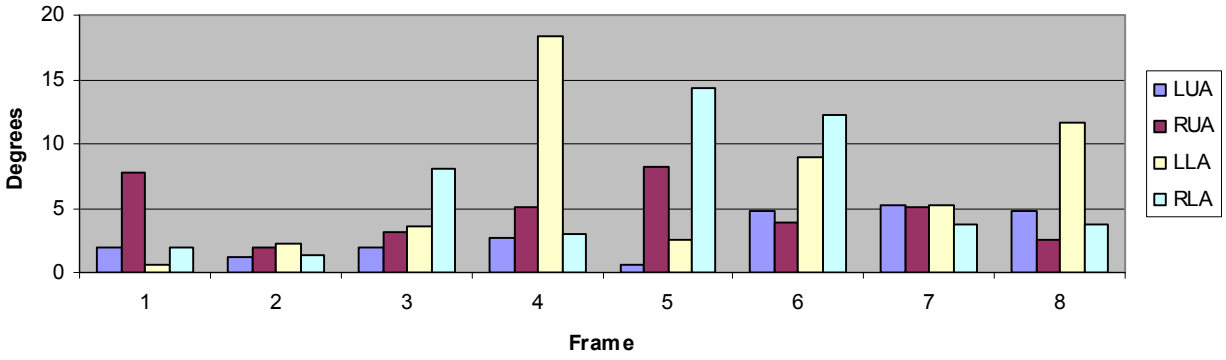
Besides the differences that influence the particle filter side there is another big difference for the computer vision module. The big difference for the computer vision module is that in the decomposed upper-body particle filter scheme there are 4 distinct evaluation phases while in the upper-body particle filter scheme there is only one. While both these methods are globally equivalent there are 2 other big differences that need to be resolved. These differences are already discussed in paragraph 4.2.2.6. The likelihood function has 2 differences namely the occlusion handling and the resulting pixels count in the upper-body particle filter scheme.

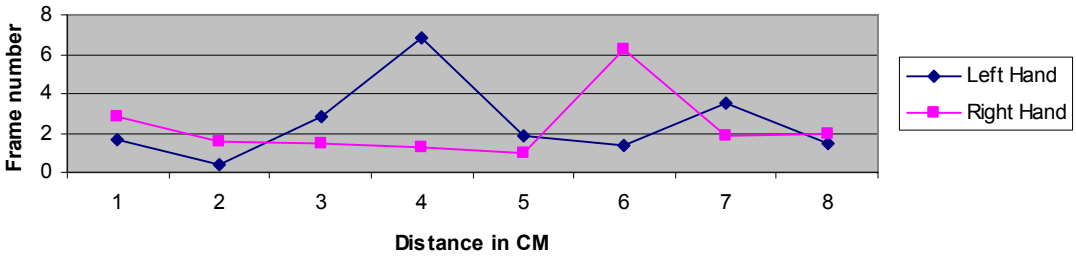
For these differences 2 different display-modes are created, each with a different evaluation algorithm. The algorithm of the body-part particle filter stays principally the same. The only difference is that there are 4 evaluation phases in which the display-mode needs to wait for new particles, evaluate them and send the likelihood value back.

The display-mode for the upper-body particle filter has only 1 evaluation iteration. In this iteration the particles are transmitted in the right order, the closest particle first and the particle that is furthest away as last. The display-mode needs to keep track of the drawn body-parts and discard the pixels from other body-parts that are already drawn, from the likelihood estimation. In addition after each evaluation the resulting pixels need to be counted for the extra likelihood estimation variable.

Appendix F *Results*

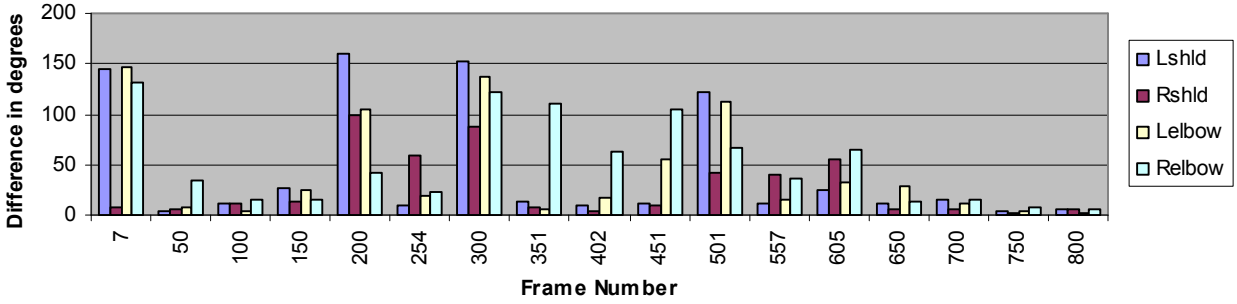
Null sequence

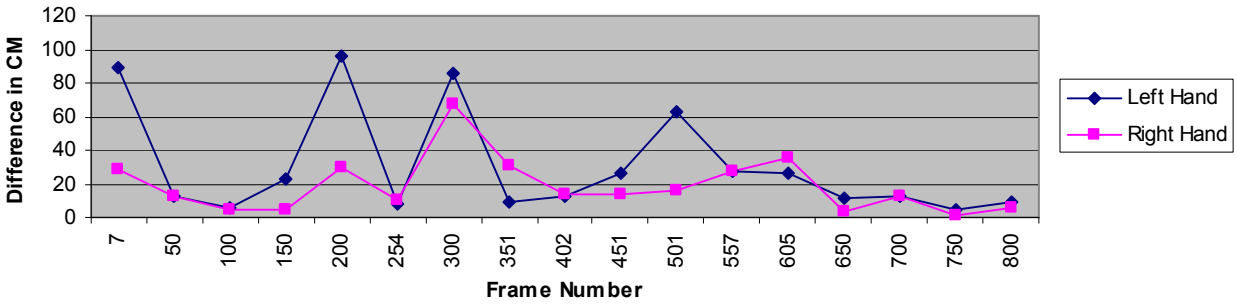




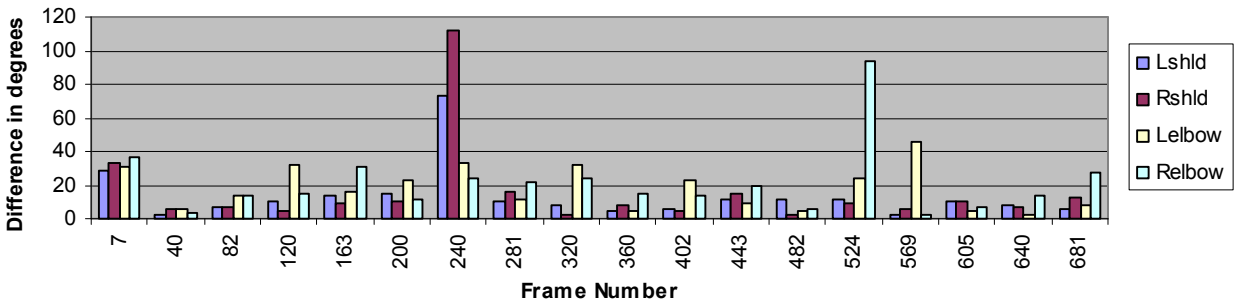
Different setup tests of user 1

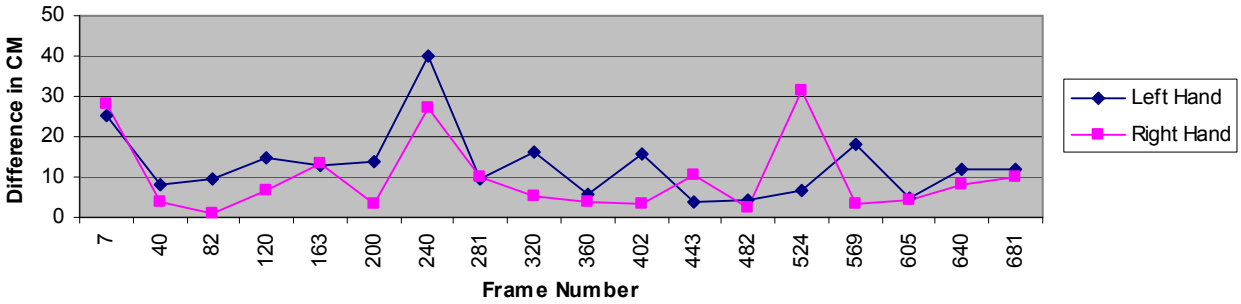
A Average





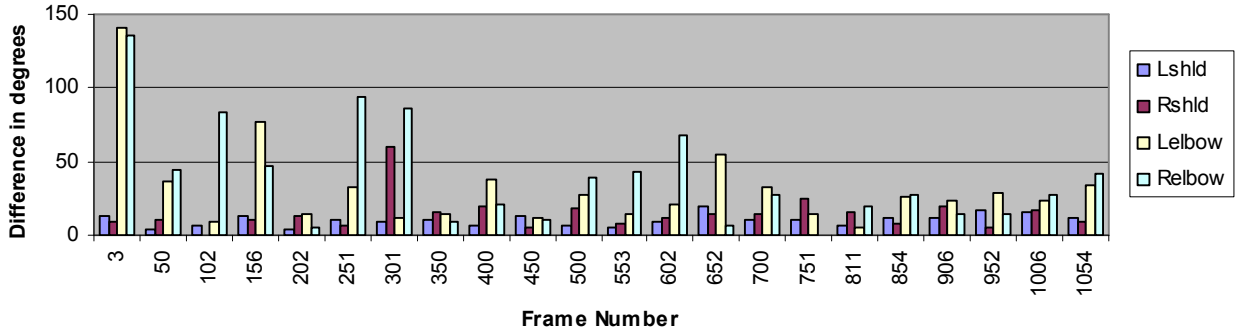
B Average

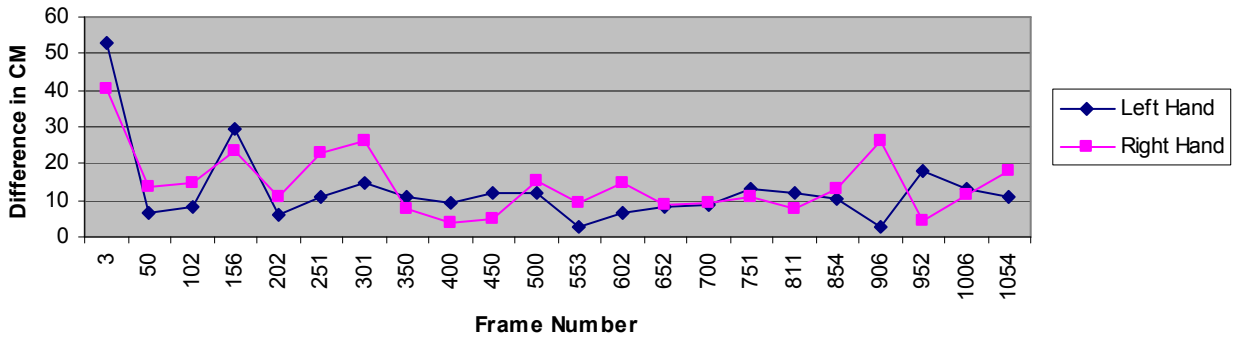




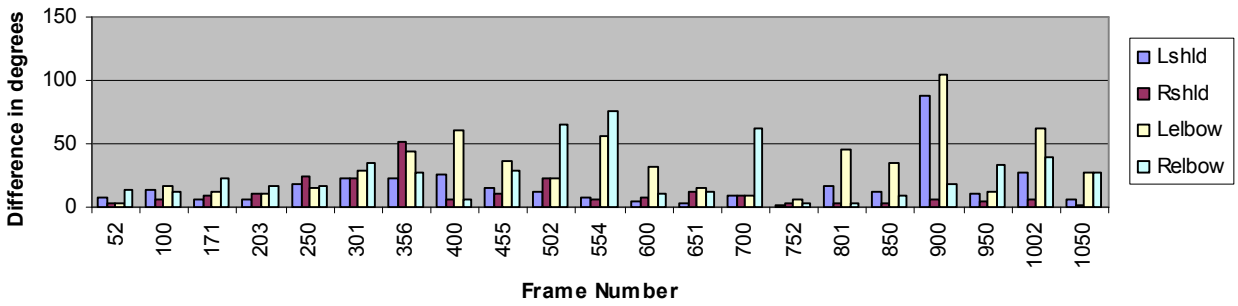
Final accuracy test

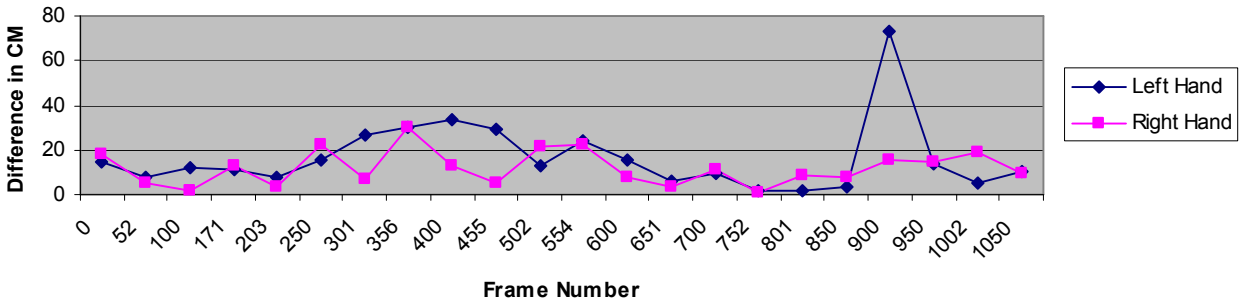
User 2



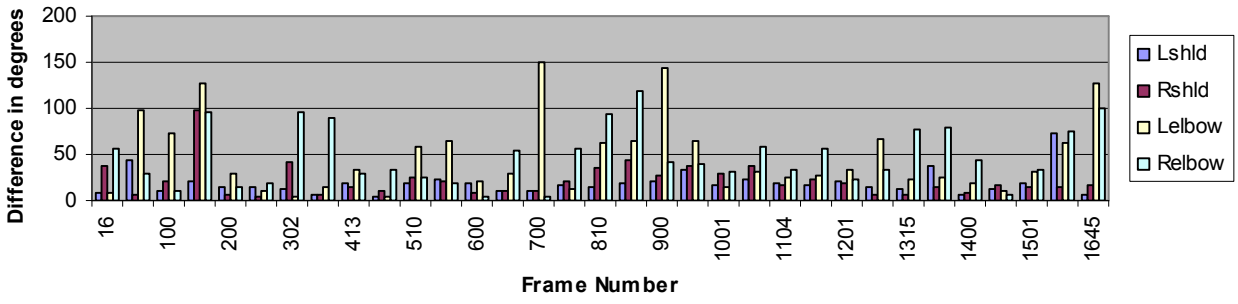


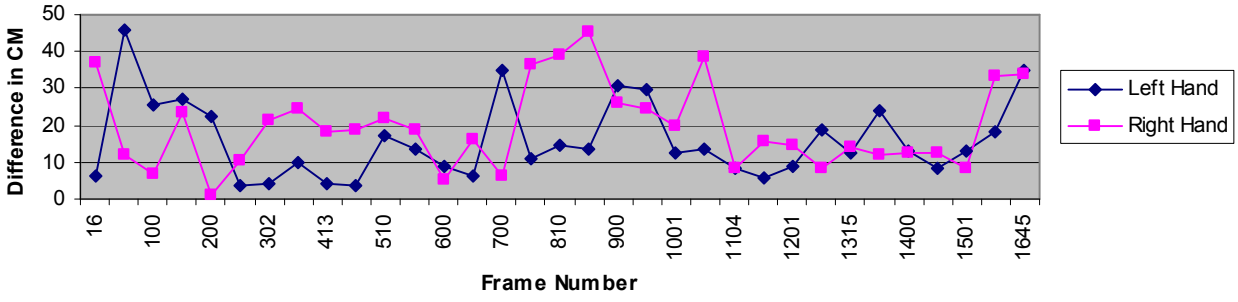
User 3


Difference in degrees
150
100
50
0
52
100
171
203
250
301
356
400
455
502
554
600
651
700
752
801
850
900
950
1002
1050
Frame Number
Lshld
Rshld
Lelbow
Relbow


Difference in CM
80
60
40
20
0
0
52
100
171
203
250
301
356
400
455
502
554
600
651
700
752
801
850
900
950
1002
1050
Frame Number
Left Hand
Right Hand

User 4


Difference in degrees
200
150
100
50
0
16
100
200
302
413
510
600
700
810
900
1001
1104
1201
1315
1400
1501
1645
Frame Number
Lshld
Rshld
Lelbow
Relbow


Difference in CM
50
40
30
20
10
0
16
100
200
302
413
510
600
700
810
900
1001
1104
1201
1315
1400
1501
1645
Frame Number
Left Hand
Right Hand

Part VI References

1. , Twist based acquisition and tracking of animal and human kinematics.
C. Bregler, J. Malik, K. Pullen.
In International Journal of Computer Vision 56, pp. 179–194, 3-2004

2. Strike a Pose: Tracking People by Finding Stylized Poses.
D. Ramanan, D. A. Forsyth, A. Zisserman.
In proceedings of IEEE conference in computer vision and pattern recognition, San Diego, 2005

3. Arm Gesture Detection in a Classroom Environment.
J. Yao and J. R. Cooperstock.
In proceedings 6th IEEE Workshop on applications of computer vision, pp. 153, 2002

4. Hierarchical part-based human body pose estimation.
R. Navaratnam, A. Thayananthan, P. Torr, R. Cipolla.
In Proceedings of the British Machine Vision Conference (BMVC), Oxford, United Kingdom, 2005

5. A model of attention-guided visual perception and recognition.
I.A. Rybak, V.I. Gusakova, A.V. Golovan, L.N. Podladchikova, N.A. Shevtsova.
Vision Research vol. 38 no 15-16 pg. 2387-2400, 1998

6. Implicit probabilistic models of human motion for synthesis and tracking.
H. Sidenbladh, M. J. Black, L. Sigal.
In proceedings of the European Conference on Computer Vision (ECCV) - volume 1, no. 2350, 2002
In Lecture Notes in Computer Science, Copenhagen, Denmark, pp. 784–800, 2002

7. Human body model acquisition and tracking using voxel data.
I. Miki´c, M. Trivedi, E. Hunter, P. Cosman.
In International Journal of Computer Vision 53 volume 3, pp. 199–223, 2003

8. A statistical Approach for Real-Time Robust Background Subtraction and Shadow Detection.
T. Horprasert, D. Harwood and L.S. Davis.
In IEEE ICCV frame rate workshop, 1999

9. W_4s: A real-time system detecting and tracking people in 2 1/2D.
I. Haritaoglu, D. Harwood, L. S. Davis.
In Proceedings of the European Conference on Computer Vision (ECCV'98) - volume 1, no. 1406
In Lecture Notes in Computer Science, Freiburg, Germany, pp. 877–892, 1998

10. Object Recognition from Local Scale-Invariant Features.
David G. Lowe.
In ICCV pp. 1150-1157, 1999

11. The visual analysis of human movement: A survey.
D. M. Gavrila.
Computer Vision and Image Understanding (CVIU) 73, pp 82–92, 1999.

12. Finding and tracking people from the bottom up.
D. Ramanan, D. A. Forsyth.
In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Vol. 2, pp. 467–474, 2003.

13. Perceptual Organization as a Method for Detection and Selection of Filamentous Structures in Highly Noisy Images Acquired by Cryo-electron Microscopy.
Y. Zhu, B. Carragher, D. Kriegman, and C. S. Potter.
Submitted ICCV July, 2001 Vancouver.

14. A Mathematical Introduction to Robotic Manipulation, chapter 2 – 3.

R. M. Murray, Z. Li, S. S. Sastry
ISBN: 0-8493-7981-4

15. The Algebraic Properties of Homogeneous Second Order Surfaces.
J.F Blinn.
SIGGRAPH'93 course notes 25: Modelling, Visualizing and animating implicit surfaces, August 1993

16. Model-Based Hand Tracking Using a Hierarchical Bayesian Filter, chapter 1 – 3.
R. Stenger.
PhD thesis, department of engineering, university of Cambridge, march 2004

17. Principals of Object perception.
E. S. Spelke.
Cognitive Science 14, 29-56 (1990)

18.Stochastic tracking of 3D human figures using 2D image motion.
H. Sidenbladh, M. J. Black, D. J. Fleet.
In Proceedings of the European Conference on Computer Vision (ECCV) volume 2, no. 1843
In Lecture Notes in Computer Science, Dublin, Ireland, pp. 702–718, 2000

19.Tracking loose-limbed people.
L. Sigal, S. Bhatia, S. Roth, M. J. Black, M. Isard.
In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 1, pp. 421–428
2004

20. Human Body Posture Inference for Immersive Interaction.
I. Cohen, H. Li and Mun Wai Lee.
International Workshop on Immersive Telepresence (ITP) 2002 in conjunction ACM Multimedia.

21. Markerless Kinematic Model and Motion Capture from Volume Sequences.
C. Chu, O. C. Jenkins, M. J. Mataric.
IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR) Volume 2, 2003

22. Pfinder: Realtime tracking of the human body.
C. R. Wren, A. J. Azarbayejani, T. Darrell, A. P. Pentland.
In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), pp. 780–785, 1997

23. Artificial Intelligence a modern approach, chapter 15 Probabilistic reasoning over time.
S. Russel, P. Norvig.
ISBN:0-13-080302-2 Second Edition, 2003

24. A Tutorial on particle filters for online, non linear, non Gaussian Bayesian tracking.
M. S. Arulampalam, S. Maskell, N. Gordon and T. Clapp
IEEE Transactions on signal processing, Vol. 50, No. 2, February 2002

25. An introduction to the Kalman filter.
G. Welch and G. Bishop.
ACM Siggraph 2001 course notes TR 95-041

26. The Notebooks of Leonardo Da Vinci. Volume: 1. Leonardo da Vinci Introduced by Edward MacCurdy.
Reynal & Hitchcock. 1954.

27.A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using off-the-shelf TV Cameras and Lenses.
Roger Y. Tsai.
In IEEE Journal of robotics and automation, vol. RA-3, no. 4 August 1987

28. Image Processing: Analysis and Machine Vision, chapter 9.
M. Sonka, V. Hlavac, R. Boyle
ISBN: 0534-95393-x

29. ISO/IEC FCD 19774:200x Humanoid animation (H-Anim)

Information technology, Computer graphics and image processing, Humanoid animation (H-Anim) standard, http://www.H-anim.org

30. Comparison of resampling schemes for particle filtering.
R. Douc, O. Cappé and E. Moulines.
Conference contribution of 4th International Symposium on Image and Signal Processing and Analysis (ISPA), 2005

31. Recovering 3D human pose from monocular images.
A. Agarwal, B. Triggs.
In IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 44–58, 2006

32. Tracking of humans in action: A 3D model-based approach.
D. M. Gavrila, L. S. Davis
In Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'96), pp. 73–80, 1996

WW Documentation

33. The watching window 2005: architecture and arm tracker changes.
Vincent van der Tuin, april 2005

34. A New Vision System for the 'Watching Window'
Wout Slakhorst, December 2004

35. The Watching Window
Jeroen Broekhuijsen, November 2004