3D Interactive and View Dependent Stereo Rendering

Per Slotsbo

Kgs. Lyngby 2004
IMM-THESIS-2004-20

Technical University of Denmark
Informatics and Mathematical Modelling
Building 321, DK-2800 Lyngby, Denmark
Phone +45 45253351, Fax +45 45882673
reception@imm.dtu.dk
www.imm.dtu.dk

IMM-THESIS: ISSN 1601-233X

Preface

This M.Sc. thesis is the final requirement for obtaining the degree: Master of Science in Engineering. This work has been carried out at the Image Group at Informatics and Mathematical Modelling, IMM, at the Technical University of Denmark, DTU, and supervised by associate professor Niels Jørgen Christensen and M.Sc., Ph.D. Andreas Bærentzen.

It is assumed that the reader is familiar with basic computer graphics. It is an advantage to have knowledge in the areas of virtual reality, photogrammetry and image analysis, but it is not a requirement.

Acknowledgements

I would like to thank Niels Jørgen Christensen and Andreas Bærentzen, for believing in the project at its beginning, for giving me free hands within reasonable limits and giving helpful feedback and advice during the project. I would also like to thank Mikkel Gjøl, Thomas Krog, Kristian Kjems and Bjarne K. Ersbøll for good inspiring dialogs on various subjects related to the project. These people and Birgitte Maribo Larsen should also be thanked for response on text and language in the report. Thanks to Henrik Aanæs, Mikkel B. Stegman, Keld Dueholm and Allan Aasbjerg Nielsen who through literature and dialogs has provided ideas, knowledge and software used in the project. I would furthermore thank Thomas Rued and Søren B. Svendsen for the dialogs regarding Color Code.

Kgs. Lyngby 2004
IMM-THESIS-2004-20

Per Slotsbo

Resumé

Udviklingen inden for hardware og software er accelererende. Web-kamera er blevet almindelige og billige, algoritmer inden for billedanalyse bliver mere avancerede og computere er blevet så kraftfulde, at mere avancerede billedanalyse algoritmer kan køre i realtid.
Denne udvikling åbner op for nye muligheder indenfor interaktion vha. web-kamera og er grundlaget for konceptet, som foreslås og implementeres i dette thesis.
Konceptet er baseret på realtids computer grafik, anaglyph stereoskopi, objekt detektion vha. billede analyse, 2D til 3D rekonstruktion, kamera kalibrering og standard hardware i form af almindelige web-kameraer og en personlig computer.
Gennem implementeringen er det vist at det med disse elementer er muligt at bygge systemet på en personlig computer for prisen af to web-kameraer.
Implementeringen kan skabe en synspunktsafhængig 2D stereo projektion af 3D-data på skærmen med korrekt bevægelses parallakse, idet brugeren følges optisk. Det er muligt at interagere med disse 3D-data (fx flytte, rotere, editere eller tegne grove skitser i 3D) på en intuitiv måde vha. et pegeredskab, som systemet følger optisk.

Nøgleord: (3D-)interaktion, synspunktsafhængig, web-kamera, realtid, meanshift, farve-tracking, kamerakalibrering, tekstur volumen.

Abstract

The development of hardware and software is accelerating. Web cameras have become common and cheap, algorithms in image analysis are getting more advanced, and computer's processing power has increased so that more advanced image analyses can run real-time.
These advances open new possibilities of interaction via web cameras, and are the foundation for a concept which is proposed and implemented in this thesis.
The concept is based on real-time computer graphics, anaglyph stereoscopics, image analysis object tracking, 2D to 3D reconstruction, camera calibration and off-the-shelf hardware in form of webcams and a desktop computer.
It is shown through the implementation that it is possible, with these components, to build the system on a desktop computer setup for the cost of two off-the-shelf webcams.
The implementation can produce a view dependent 2D stereo projection of the 3D-data onto the screen enabling true moving parallax using viewer tracking. It is possible to interact with the 3D-data (E.g. moving, rotating, editing or drawing coarse sketches in 3D.) in an intuitive way using a tracked pointing device.

Keywords: (3D-)interaction, view-dependent, web-camera, real-time, meanshift, color-tracking, camera-calibration, texture-volume.

Table of Contents

Chapter 1 Introduction

1.1 Thesis Overview

This report starts with the background of the project explaining why it is relevant. This is followed by a walkthrough of some related and relevant literature. We will then make a theory and a hypothesis we are about to prove in the implementation and then analyze how to implement the theory including some considerations of the user interface. Finally in the last three chapters, results will be presented and discussed before the conclusion. In the discussion we will furthermore look at the future work still to be done.

1.2 Nomenclature

Scalars

Scalars are typeset in italic and non-boldface: s

Vectors & Matrixes

Vectors and Matrixes are typeset in non-italic and boldface. Matrixes are always upper-case, and vectors are not case sensitive:

$$\mathbf{v} = \begin{bmatrix} a & b & c \end{bmatrix}$$

$$\mathbf{M} = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$

2D-point

By 2D-point is meant a point in a given 2D plane:

$$\mathbf{x} = \begin{bmatrix} x & y \end{bmatrix}^{\mathrm{T}}$$

3D-point

By a 3D-point is meant the spatial location in a given 3D space:

$$\mathbf{x} = \begin{bmatrix} x & y & z \end{bmatrix}^{\mathrm{T}}$$

Direction

By direction is meant the way to go from one 3D-point, to hit another given 3D-point. The direction can explicitly be given e.g. by a single 3D-vector or two rotational angels.

Rotational Orientation

A rotational orientation is similar to a direction but additionally the rotation around the axis of direction is known. It can explicitly be given by e.g. three rotation angels, a quaternion or a rotation matrix.

Orientation

By orientation is meant a position and a rotational orientation. The transformation of an object from one orientation to another can be given explicitly by e.g. a 3D-point and a rotational orientation or a single transformation matrix.

Camera Orientation

Camera orientation is divided into two orientations:

- **Outer orientation:** Represent the spatial orientation of the camera relative to a given coordinate system (i.e. the same as the orientation described above)
- **Inner orientation:** Represents all manufacturing attributes (usually considered constant) which usually are: the camera constant, the principal point, the lens distortion and the affine deformation [8].

DOF

The number of *degrees of freedom* (DOF) is given by the minimum number of scalars that is needed to explicitly determine a given representation.

1.3 Background

The World Is 3D

Although we are living in a 3D world, much information is better presented in 2D (like maps and diagrams) due to the ease with which we can make an overview in 2D. This is mostly because of the nature of the human eye, which works by making a very precise perspective projection of the real 3D world into a 2D image. In spite of of this fact some types of information is still better presented in 3D because of their 3D nature.

Depth Cues

The human brain is able to perceive 3D information via the eyes using a range of visual depth cues such as: perspective, lighting, occlusion, moving parallax, stereoscopic parallax, depth focus etc. [1]. All these blend in such a way that they individually add more depth information to the perception

filling out each others holes, and the stronger ones ruling out the weaker ones when ambiguous. (E.g. surface lighting can give a perception of detailed continuously depth variation, but if some part of an object is in front of another part occlusion is stronger (but much less detailed) and you are sure that the occluded part is behind the other, independent of what the weaker lighting is telling you.)
An example of depth cues is normal photos, where we usually get a good impression of the depth although important depth cues like moving parallax and stereo parallax are missing. In Figure 1 we see how the depth cues even can be used to create an illusion of depth that is not there.

Figure 1 : Optical Illusion in Street Painting. Does his foot get wet? The illusion of a pool filled with water in the middle of the street is strong; although we are aware it is not the truth. It is well suited for pictures, because cameras have a single lens and photographs do of course not move. If we were on the street, our stereo vision and possible movement would dissolve the illusion immediately. I.e. the painting can be viewed only from a single position with one eye closed to be as convincing as at the snap shot. Augmented Reality seeks to let such an illusion be intact by producing the proper stereo image according to the user's position.

Hardware Accelerated 3D

As we have just seen, a lot of the depth cues can be utilized through different visualization techniques. Especially computers have made it possible to make real time 3D graphics using hardware acceleration (which produces perspective projected 2D images including lighting) and is one of the most used today in science, games and movies. The fact that it is real-time enables the moving parallax depth cue resulting in good depth perception, when for instance rotating an object, which often gives real-time graphics an extra feel of reality compared to the same images in still version. But as soon as the motion stops, the effect is gone. Stereoscopic parallax is very similar to motion parallax, because they both work by comparing the parallax between images seen from slightly moved positions. (see Figure 2 and Figure 3)

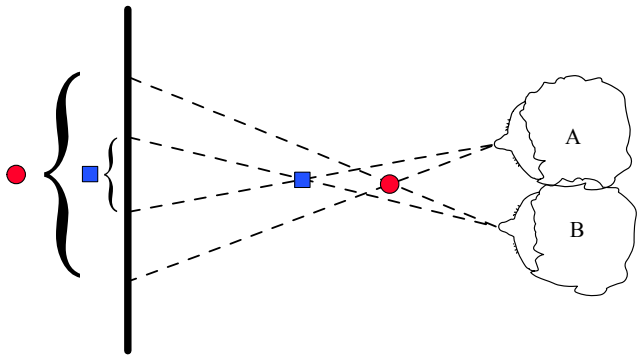


Figure 2 : Moving parallax. The depth cue, moving parallax, enables the brain to extract depth information from the movement of an object in relation to the viewer's movement from A to B. Rotation of an object utilizes this and enhances the effect on the rotating object. The brackets to the left indicate the amount of parallax for two objects in a projected image as a result of the movement.

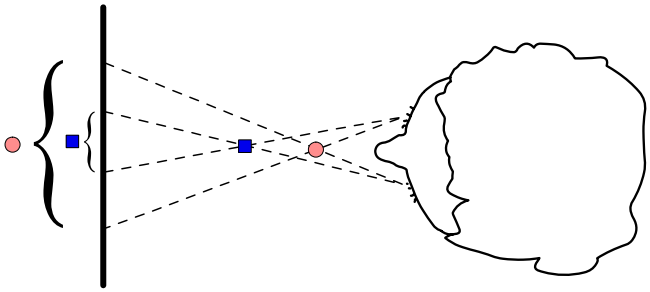


Figure 3 : Stereoscopic parallax. The depth cue stereoscopic parallax is very similar to moving parallax enabling the brain to extract depth information from the parallax of an object due to the eyes' separation. This also works in still images or with moving objects contrary to moving parallax.

Stereo vs. Mono

Systems that display the depth cue stereoscopic parallax, has not yet been developed to a level where they are commonly used every day, though they are widely spread. This might be caused by the ratio between how useful it is and what kind of effort the user has to put into it to make it work (including price). The precision of depth in stereo vision falls with distance [8]. Therefore, it is obviously most useful in close range, like hand-eye 3D coordination. With mono vision it can be difficult to poor water into a glass without spilling or to thread a needle, but daily life orientation and tasks are done with no big difference by people with mono or stereo vision respectively, because of other depth cues presented to a person. The unique quality of stereo parallax is that the depth information can be given in a still image from a single point. Other cues rely on other visual information like shadows, relative brightness, light reflections, perspective of shape, occlusion etc. The depth being independent of the rest of the image also means that stereo is good for presenting complex structures in still images.

"But why isn't it popular then?" one might ask. Glasses etc. that have been necessary to enable stereo for a long time, are just not as easy as rotating a 3D model.

Stereo without Moving Parallax

There is another drawback using stereo. When viewing a still stereo image the mind has got the depth information from stereo parallax. When the user moves his or her head relatively to the virtual object, there are no moving parallax, resulting in one perceptional solution in the mind: the object is rotated and skewed, to face the user the same way as before the movement [1] (see Figure 4).

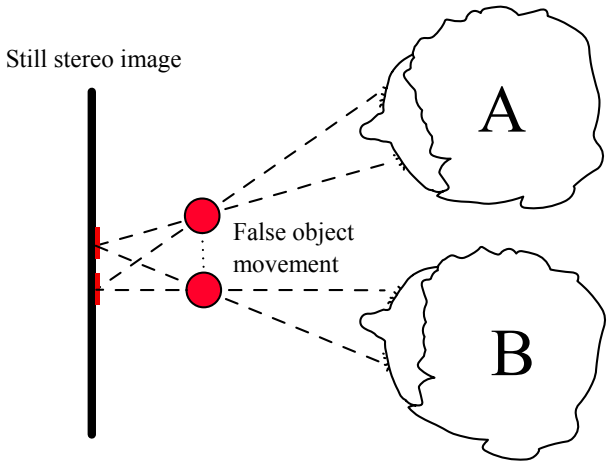


Figure 4 : Trouble with stereo without moving parallax. Moving your head from A to B watching a still stereo image gives a false (usually unwanted) impression that the object moves, because this is what has happened according to what is seen and moving parallax. The effect is proportional with the object-image distance, which also results in skewing of the object. (Solving this problem by changing the stereo image depending on the view, is in this thesis referred to as viewer alignment)

Automatic Stereo

During the past decades a lot of effort has been made to develope an auto-stereoscopic screen. Now there seems to be a break through, and the first commercial productions are starting up [9][10][11][12][13].
Systems for one or few viewers often rely on the viewer's eye position to continuously project the correct picture to each eye [11][14]. Some applications for these systems use the eye position to align the projection of the virtual 3D data with the viewer, enabling correct moving parallax [21]. This means that the viewer feels that the virtual objects are in the same world as himself, his hands, the screen etc, and could (if forgetting that the objects are in fact virtual) try to grab them in thin air. At this point, a 3D

pointing device seems to be an obvious tool for interacting with the 3D data, which we will look into later. But why do we need interaction?

The Need for Interaction

With the computers and their format of information, the need for an easy way of interaction with this information exists. Today the keyboard and the mouse are the most common interfaces to computers. The mouse working in pseudo analog 2D is an ideal interface for interacting in the 2D world of today's computer monitors. But when it comes to 3D, it has got its limitations.

Mouse vs. 3D Interaction

Although there are many techniques for interacting with a 3D world using the 2D mouse, a simple task like moving or rotating an object in 3D to a given orientation is never as trivial or fast as for a human moving a handheld object in the real world.
This is the reason why many new suggestions for 3D interaction are continuously developed.

One Ideal Goal

Ivan Sutherland proposed in 1965 the "*ultimate display*" as one that produce images and other sensory input with such a fidelity that the observer could not tell the simulated objects from real ones. He showed a prototype of such a virtual reality display in 1968 [1].

Virtual Reality

Virtual Reality pursue the idea of the ultimate display, by artificially produce sufficiently 3D cues of a virtual world to convince a viewer, that what is presented is real, enabling him or her to be submerged into this world (seeing nothing but this world), and to interact with it. Usually this is done by using movement tracking real-time computer graphics and view dependent stereoscopics. This has fascinated many people and a lot of money has been invested in creating hardware and software to make the feeling of realism most natural [1].

1.4 Objective

The main goal of this thesis is to build a prototype of a relatively inexpensive augmented reality system including a 3D pointing device based on a common desktop computer. It should enable fast and intuitive interaction with 3D data. The degree of intuitivity should be strengthened by viewer's and tool's alignment with the virtual world's positions. It is furthermore a goal along with the work to produce a demo application, which utilizes and demonstrates some of the abilities and benefits of the system. In Chapter 2 a hypothesis is setup describing the concept, and claiming that the system can work. This report is build up with the goal of proving the hypothesis by developing and implementing the concept.

1.4.1 Applications

In the following a few examples will be given of where the system might be applied with advantage.

Examples of use

Currently much work is in progress to automate registration and visualization of medical 3D-data (e.g. obtained via MR or CT scanning), to be used for instance in diagnostics, and surgery planning (see Figure 5). In this case augmented reality provides an interface, which makes interaction faster and more intuitive. The user can move around the data. A 3D pointing device in augmented reality can be used as input for direct orientation or editing of the data. Surgery planning and surgery training are also subjects of development.

Most industries working with 3D-data would benefit from a desktop 3D interface, as alternative to the mouse. Especially in 3D modeling used in game and movie industry more and more people work with 3D models every day moving and editing objects and points in the 3D space. Using augmented reality would make the tasks faster and make it easier to interact with the model directly.

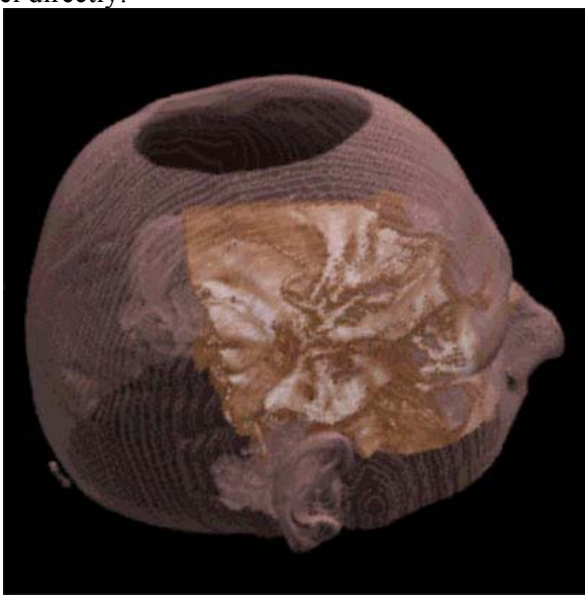


Figure 5: 3D visualization of MR and CT scanned medical data.

Other Applications

The anaglyph stereo technique by the Danish company ColorCode[15] allows for affordable color stereo. The remote viewer alignment system could be used i combination with their so called phantom model.

Furthermore, the remote sensing part of the project is related to eye-gaze-controlled applications for use by disabled people.

Should future systems developed from or inspired by this or other projects, become as effortless to use as purchasing and using a mouse, it would be beneficial in many areas: (education), medicine (MR scanning, CT-scanning, ultrasound scanning, surgery planning and surgery training tools), oil industry (underground analysis), city planning (coarse element placement), mechanical design

Furthermore in research areas like chemistry (macro molecule structures), medical (cognitive understanding and segmentation)

In principle it is also applicable in all software where real-time 3D graphics are present today enhancing the feeling of 3D and entertainment value.

Novel applications, not considered today because of limitations in today's monitors and interaction, might also evolve.

Chapter 2 Hypothesis

Introduction

In this chapter we will setup a hypothesis suggesting that it is possible to make a 3D system following a concept that fulfills specific properties. In the thesis an implementation is produced to prove this concept.

2.1 The Hypothesis

The development of hardware and software is accelerating.
Web cameras have become common and cheap, opening for the possibilities of using it for more than its original communication purpose. Algorithms in image analysis are getting more and more advanced, and computers processing power have increased so that some image analyses can run real time.
These advances opens for new possibilities of interaction via web cameras, and is the foundation for the hypothesis that now will be presented, and is the focus of this thesis.

The hypothesis claims:

It is possible to make a 3D pointing device and viewer alignment augmented reality system using cheap off-the-shelf web cameras fulfilling the following abilities:

Abilities:

The system should be able to:

- Run real-time on a desktop computer.
- Use cheap off-the-shelf webcam hardware.
- Track a pointing device in 3D (e.g. hands, fingers, colored spheres, a pair of tweezers etc.).
- Track a viewer's position in 3D.
- Produce a 2D projection of 3D data on a computer monitor.
- Align the virtual 3D objects with the viewer so that the projected image is similar to what the viewer would see if the objects were real, although the viewer is moving.
- Let the user interact with the 3D objects via the pointing device. (For instance drawing, moving, rotating, editing etc.).
- The user is able to make rough drawings in 3D.

With the goal setup, we are ready to go on with the thesis.

Chapter 3 Literature

3.1 Theory

3.1.1 Perspective

Our eyes see the world through perspective projection (i.e. the world within the line-of-sight in front of our eyes is projected to 2D images on our retinas). It has the special property of scaling objects with distance, so the projected scale of the object x_p becomes:

$$x_p = c\frac{x_w}{z} \tag{1}$$

, where c is a constant (e.g. the camera constant), x_w is the scale of the object in the world, z is the distance to the object. A consequence of the feature is, that if we would like to see an object in greater detail we just have to get closer.

Homogeneous Coordinates

Homogeneous coordinates allow us to make a linear perspective projection. The nonlinearity of equation (1) is performed in the transformation from homogeneous coordinates into image-coordinates. In homogeneous coordinates a point in the projected image is represented as:

$$\mathbf{x} = \begin{bmatrix} \omega x \\ \omega y \\ \omega \end{bmatrix} \tag{2}$$

, where ω (equal to $\frac{z}{c}$ in (1)) is the factor with which ωx and ωy should be divided according to perspective.
In computer graphics the perspective projection is usually given by the formulas seen in appendix 10.1 including the total perspective projection that, given a set of coordinates $\mathbf{X}$, a scale matrix $\mathbf{S}$, translation matrix $\mathbf{T}$, rotation matrix $\mathbf{R}$ and projection matrix $\mathbf{P}$, is:

$$\mathbf{x} = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{T} \cdot \mathbf{S} \cdot \mathbf{X} \tag{3}$$

The transformation matrixes, $\mathbf{S}$, $\mathbf{T}$ and $\mathbf{R}$, should be multiplied in order and number according to the set of wished transformations starting from the right with the first transformation.

3.1.2 Stereo Measurement

As in the human stereo vision, one can reconstruct depth from two or more images. This is a well known method in photogrammetry e.g. using air-photos [4]. In the reconstruction one can use either a direct linear transformation model (DLT), ignoring lens distortions, or a more precise nonlinear model [3]. We will now go through the linear method.

3.1.3 Direct Linear Transformation (DLT)

All transformations in the perspective projection (3) can be represented in a single projection matrix, $\mathbf{A}$, which is the product of any matrices that form the transformation and projection:

$$\mathbf{x} = \overbrace{\mathbf{P} \cdot \mathbf{R} \cdot \mathbf{T} \cdot \mathbf{S}}^{\mathbf{A}} \cdot \mathbf{X}$$

$$\mathbf{A} = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{T} \cdot \mathbf{S} \tag{4}$$

If $\mathbf{A}$ is known for a camera, the image point $\mathbf{x}$ projected from a given 3D point $\mathbf{X}$, can be found by a single multiplication with this matrix:

$$\begin{bmatrix} \omega x \\ \omega y \\ \omega \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

$$\mathbf{x} = \mathbf{A} \cdot \mathbf{X} \tag{5}$$

This is exploited in all modern graphics hardware.

3.1.4 3D Reconstruction

When an objects position in 2D is known in images from more than one camera, these positions can be used to reconstruct the objects 3D position, just like the human stereo vision provides depth perception.
2D positions from two or more cameras provide four or more equations to solve for the three unknowns of the 3D position. This indicates redundancy. Redundancy is in some cases exploited by reducing the system and makes the 2D search algorithms faster by first finding the object or feature in one image and then search for it only on one line in the other, the so called epipolar line. The redundancy can also be used to give us a better estimate of the point. Then the residuals tell us something about the reliability of the tracked 3D position, which could be used to determine whether we want to accept the given result. Here we will only look into one method using over determination.

The equations

Let p be the number of cameras. Let the first camera have the projection matrix $\mathbf{A}_1$ and image point $\mathbf{x}_1$, then its projection of the unknown $\mathbf{X}$ is:

$$\mathbf{x}_1 = \mathbf{A}_1 \cdot \mathbf{X} \tag{6}$$

, which according to (5) leads to the two following equations:

$$x_1 = \frac{a_{1_{11}} X + a_{1_{12}} Y + a_{1_{13}} Z + a_{1_{14}}}{a_{1_{31}} X + a_{1_{32}} Y + a_{1_{33}} Z + a_{1_{34}}} \tag{7}$$

$$y_1 = \frac{a_{1_{21}} X + a_{1_{22}} Y + a_{1_{23}} Z + a_{1_{24}}}{a_{1_{31}} X + a_{1_{32}} Y + a_{1_{33}} Z + a_{1_{34}}} \tag{8}$$

By isolating X, Y and Z we obtain:

$$0 = \left(a_{1_{11}} - x_1 a_{1_{31}}\right) X + \left(a_{1_{12}} - x_1 a_{1_{32}}\right) Y + \left(a_{1_{13}} - x_1 a_{1_{33}}\right) Z + \left(a_{1_{14}} - x_1 a_{1_{34}}\right)$$
$$0 = \left(a_{1_{21}} - y_1 a_{1_{31}}\right) X + \left(a_{1_{22}} - y_1 a_{1_{32}}\right) Y + \left(a_{1_{23}} - y_1 a_{1_{33}}\right) Z + \left(a_{1_{24}} - x_1 a_{1_{34}}\right)$$

, forming the equation system:

$$\begin{bmatrix} \left(a_{1_{11}} - x_1 a_{1_{31}}\right) & \left(a_{1_{12}} - x_1 a_{1_{32}}\right) & \left(a_{1_{13}} - x_1 a_{1_{33}}\right) & \left(a_{1_{14}} - x_1 a_{1_{34}}\right) \\ \left(a_{1_{21}} - y_1 a_{1_{31}}\right) & \left(a_{1_{22}} - y_1 a_{1_{32}}\right) & \left(a_{1_{23}} - y_1 a_{1_{33}}\right) & \left(a_{1_{24}} - y_1 a_{1_{34}}\right) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = -\begin{bmatrix} a_{1_{14}} - x_1 a_{1_{34}} \\ a_{1_{24}} - y_1 a_{1_{34}} \end{bmatrix} \tag{9}$$

Same principle applies for camera p for x_p and y_p using $\mathbf{A}_p$ leading us to a system of $2p$ equations and 3 unknowns.

$$\begin{bmatrix} \left(a_{1_{11}} - x_1 a_{1_{31}}\right) & \left(a_{1_{12}} - x_1 a_{1_{32}}\right) & \left(a_{1_{13}} - x_1 a_{1_{33}}\right) & \left(a_{1_{14}} - x_1 a_{1_{34}}\right) \\ \left(a_{1_{21}} - y_1 a_{1_{31}}\right) & \left(a_{1_{22}} - y_1 a_{1_{32}}\right) & \left(a_{1_{23}} - y_1 a_{1_{33}}\right) & \left(a_{1_{24}} - y_1 a_{1_{34}}\right) \\ \vdots & \vdots & \vdots & \vdots \\ \left(a_{p_{11}} - x_p a_{p_{31}}\right) & \left(a_{p_{12}} - x_p a_{p_{32}}\right) & \left(a_{1_{13}} - x_p a_{p_{33}}\right) & \left(a_{p_{14}} - x_p a_{p_{34}}\right) \\ \left(a_{p_{21}} - y_p a_{p_{31}}\right) & \left(a_{p_{22}} - y_p a_{p_{32}}\right) & \left(a_{p_{23}} - y_p a_{p_{33}}\right) & \left(a_{p_{24}} - y_p a_{p_{34}}\right) \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = -\begin{bmatrix} a_{1_{14}} - x_1 a_{1_{34}} \\ a_{1_{24}} - y_1 a_{1_{34}} \\ \vdots \\ a_{p_{14}} - x_p a_{p_{34}} \\ a_{p_{24}} - y_p a_{p_{34}} \end{bmatrix} \tag{10}$$

, or:

$$\mathbf{B} \cdot \mathbf{X} = \mathbf{k} \tag{11}$$

At least 2 cameras are needed to estimate $\mathbf{X}$. An estimate of $\mathbf{X}$ is found using Least Squares Adjustment (see section 3.1.6 after the following section).

3.1.5 Linear Camera Calibration

To avoid unnecessary confusion we will now recall the camera orientation mentioned in section 1.2.
By camera calibration is meant finding the camera orientation which is divided into two orientations:

- **Outer orientation:** Represent the spatial orientation of the camera relative to a given coordinate system
- **Inner orientation:** Represents all manufacturing attributes (usually considered constant) which usually are: the camera constant, the principal point, the lens distortion and the affine deformation [8].

In the linear case we only deal with the camera constant and the outer orientation, ignoring lens distortions. The orientation can be found directly in means of the perspective projection matrix (4), from which representation in rotation angels and translations can be found directly. The projection matrix is sufficient for reconstructing 3D points, like we saw in the previous section.

Finding the perspective projection

Let $\mathbf{A}$ be the unknown matrix we wish to find:

$$\mathbf{A} = \begin{bmatrix} \mathbf{a}_1 \\ \mathbf{a}_2 \\ \mathbf{a}_3 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \tag{12}$$

The linear calibration is carried out by measuring a number of point pairs ($\mathbf{x}$, $\mathbf{X}$) in camera and world coordinate systems respectively.

Known point $\mathbf{X}$ in the world:

$$\mathbf{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^{\mathbf{T}} \tag{13}$$

In the image only x and y are known.

$$\mathbf{x} = \begin{bmatrix} \omega x & \omega y & \omega \end{bmatrix}^{\mathbf{T}} \tag{14}$$

As in the 3D reconstruction we use the projection equation (5) (written using one row vector at a time):

$$\begin{bmatrix} \omega x \\ \omega y \\ \omega \end{bmatrix} = \begin{bmatrix} \mathbf{a}_1 \cdot \mathbf{X} \\ \mathbf{a}_2 \cdot \mathbf{X} \\ \mathbf{a}_3 \cdot \mathbf{X} \end{bmatrix} \tag{15}$$

, which again leads to the equations (7) and (8), or shorter:

$$x = \frac{\mathbf{a}_1 \cdot \mathbf{X}}{\mathbf{a}_3 \cdot \mathbf{X}} \tag{16}$$

$$y = \frac{\mathbf{a}_2 \cdot \mathbf{X}}{\mathbf{a}_3 \cdot \mathbf{X}} \tag{17}$$

$\Rightarrow$

$$0 = (\mathbf{a}_1 - x\,\mathbf{a}_3) \cdot \mathbf{X} \tag{18}$$

$$0 = (\mathbf{a}_2 - y\,\mathbf{a}_3) \cdot \mathbf{X} \tag{19}$$

Equations (18) and (19) forms a homogeneous system, but fortunately we can scale the matrix setting $a_{34} = 1$ which leaves us with eleven unknown a's . We now reshape $\mathbf{A}$ into the vector $\hat{\mathbf{A}}$:

$$\hat{\mathbf{A}} = \begin{bmatrix} \hat{a}_1 \\ \hat{a}_2 \\ \vdots \\ \hat{a}_{10} \\ \hat{a}_{11} \end{bmatrix}, \text{ where } \begin{matrix} \hat{a}_1 = a_{11} \\ \hat{a}_2 = a_{12} \\ \vdots \\ \hat{a}_9 = a_{31} \\ \hat{a}_{10} = a_{32} \\ \hat{a}_{11} = a_{33} \end{matrix} \text{ or } \hat{\mathbf{A}} = \begin{bmatrix} \mathbf{a}_1^{\mathrm{T}} \\ \mathbf{a}_2^{\mathrm{T}} \\ a_{31} \\ a_{32} \\ a_{33} \end{bmatrix}, \; a_{34} = 1 \tag{20}$$

Using (18) and (19) we can now setup the following inhomogeneous system for the point pair ($\mathbf{x}$, $\mathbf{X}$):

$$\begin{bmatrix} X & Y & Z & 1 & 0 & 0 & 0 & 0 & -xX & -xY & -xZ \\ 0 & 0 & 0 & 0 & X & Y & Z & 1 & -yX & -yY & -yZ \end{bmatrix} \hat{\mathbf{A}} = \begin{bmatrix} x \\ y \end{bmatrix} \tag{21}$$

$$\mathbf{B} \cdot \hat{\mathbf{A}} = \mathbf{c} \tag{22}$$

, where $a_{34} = 1 \Rightarrow$ the right hand side: $\begin{bmatrix} x \\ y \end{bmatrix}$

The linear system to be solved

Every point pair registered, adds two equations to the system. Thus for 6 point pairs $\mathbf{B}$ becomes a 12x11 matrix and for p points $(2p) \times 11$:

$$2p\left\{\overbrace{\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -x_1 X_1 & -x_1 Y_1 & -x_1 Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -y_1 X & -y_1 Y & -y_1 Z \\ & \vdots & & & & \vdots & & & & \vdots & \\ X_p & Y_p & Z_p & 1 & 0 & 0 & 0 & 0 & -x_p X_p & -x_p Y_p & -x_p Z_p \\ 0 & 0 & 0 & 0 & X_p & Y_p & Z_p & 1 & -y_p X_p & -y_p Y_p & -y_p Z_p \end{bmatrix}}^{11}\right\} \hat{\mathbf{A}} = \left.\begin{bmatrix} x_1 \\ y_1 \\ \vdots \\ x_p \\ y_p \end{bmatrix}\right\} 2p \tag{23}$$

, or shorter:

$$\mathbf{B} \cdot \hat{\mathbf{A}} = \mathbf{c} \tag{24}$$

We would now like to estimate $\hat{\mathbf{A}}$, which requires $(2p) > 11$ - fulfilling this requirement an estimate can be found linearly using Least Squares Adjustment (see following section 3.1.6)

3.1.6 Linear Least Squares Adjustment

Least squares adjustment is a method that is widely used where a result is to be found from observations with redundancies, i.e. the number of observations n is larger than the number of unknown variables p [16]. In this thesis it is used in cameras calibration and 3D reconstruction.

Given some measurements without error, a 3D point can be understood as the intersection of lines each going through a center of a camera and the object positions on its image plane respectively. If error is introduced the lines might not intersect, and the 3D point can be chosen to be the point with the smallest sum of squared distances to the lines i.e. exactly between the lines (see Figure 6).

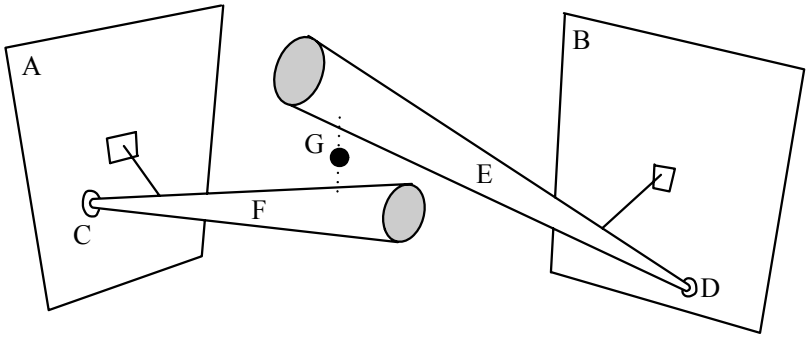


Figure 6: 3D Reconstruction: Given two image planes A and B and two image points C and D, an estimate of the 3D position, G, can be defined to be the place where the squared sum of distances to the lines F and E through the image points are smallest.

This is what is done when solving equation (11) in section 3.1.4. The squared error to be minimized is:

$$E = \sum_n \left\| x - \hat{x} \right\|^2 \tag{25}$$

, where $x - \hat{x}$ is the error between the observation and the result predicted by the system.
The system is redundant and linear of the form:

$$\mathbf{B} \cdot \hat{\mathbf{A}} = \mathbf{c} \tag{26}$$

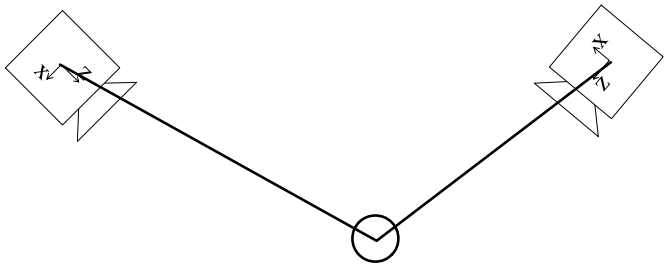
, where $\mathbf{B}$ is an $n \times p$ matrix, $\hat{\mathbf{A}}$ is an $p \times 1$ vector to be found, and $\mathbf{c}$ is an $n \times 1$ vector.
The solution $\hat{\mathbf{A}}$ (e.g. the point) can be found directly (i.e. not iteratively) by applying least squares minimization:

$$\hat{\mathbf{A}} = \left(\mathbf{B}^{\mathbf{T}} \mathbf{B} \right)^{-1} \mathbf{B}^{\mathbf{T}} \mathbf{b} \tag{27}$$

, which should not be solved by inverting $\mathbf{B}^{\mathbf{T}}\mathbf{B}$ but rather by means of SVD (single value decomposition), QR or Cholesky decomposition [16]. These routines are standard in most linear algebra packages.

3.1.7 Non-linear Relative Camera Calibration

In relative camera calibration, one cameras outer orientation (i.e. spatial position and rotation) in relation to a second camera is determined only using observation of 5 or more points of unknown 3D coordinates.
The relative camera calibration is non-linear because the transformation depends non-linearly on the camera rotation angels.



The relative orientation can be described by the transformation matrix $\mathbf{B}$:

$$\mathbf{B} = \mathbf{R} \cdot \mathbf{T} \tag{28}$$

, where $\mathbf{R}$ and $\mathbf{T}$ are the rotation and translation matrix respectively. The relation between the cameras projection matrixes is then given by:

$$\mathbf{A_2} = \mathbf{A_1} \cdot \mathbf{B} \tag{29}$$

Because the system is nonlinear, solving $\mathbf{A}_1$ and $\mathbf{A}_2$ involves rewriting it to an appropriate system that can be solved by iterative nonlinear least squares methods. Finding the solution to this system, has shown to be out of the scope of this thesis, restricting us to the linear case.

3.1.8 2D Tracking

Finding and tracking an object in 2D is done seamlessly by our brain constantly, but is in fact a rather complicated process. Tracking is a very broad research-area which continuously advances in complexity and quality. In this subsection two simple algorithms are mentioned.

3.1.8.1 Mean Shift Algorithm

The mean shift uses a search window, which limits the search to a small rectangular portion of the image. The algorithm is used to find local maxima iteratively, by finding a mean value in the search window and shifting the window to this position.
By using a search window, the mean shift algorithm limits its computational load compared to making a search in the total image.

Given a discrete 2D probability distribution and a search window, the mean location of the window can be found using the zeroth moment (or summed probability):

$$M_{00} = \sum_{x,y} w(x,y) \tag{30}$$

, and first moment for x and y :

$$M_{10} = \sum_{x,y} x\, w(x,y) \tag{31}$$

$$M_{01} = \sum_{x,y} y\, w(x,y) \tag{32}$$

, where $w(x,y)$ is the probability at a point (x,y).

The mean location of the window (or center of gravity) is then:

$$x_x = \frac{M_{10}}{M_{00}} \quad \text{and} \quad y_x = \frac{M_{01}}{M_{00}} \tag{33}$$

By iteratively shifting the window to the mean location, and recalculating, the mean shift algorithm climbs the distribution gradient through the image to a local maximum as seen in Figure 7.

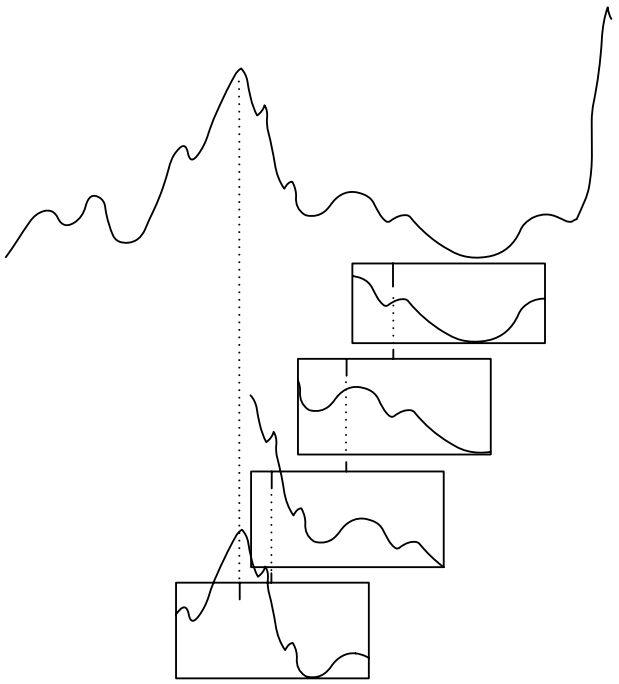


Figure 7: Iterations of the mean shift algorithm in 1D (e.g. finding maximum value on a line). It is seen how the center of the window is shifted to the mean location of the probability within it. Note also that only a local maximum is found.

The mean shift algorithm is fast and seems to stay in local maxima (which is good in some cases), but it has problems with scaling of the searched peak in the distribution e.g. if the peak gets too narrow, it has too little weight at the mean value calculation. This problem is addressed in the following section.

3.1.8.2 CAMSHIFT

A team at Intel [7] has introduced a fast algorithm called CAMSHIFT that, based on the mean shift algorithm, can track a colored object in a live image, especially with face tracking in mind.

Mean Shift Based

The CAMSHIFT model works like the mean shift algorithm by climbing a distribution gradient through the image to a local maximum, and furthermore adapts the size of the search window using the zero momentum in the window to overcome problems of perspective size change in a dynamic tracking environment.

Histogram Table

CAMSHIFT calculates the probability via a lookup table that is a histogram for the given color object. That way all colors represented in the object will attribute to the weight according to their value in the object histogram [7].

The basic steps

The basic steps of the CAMSHIFT algorithm is in short terms: defining a search window within the picture, finding the mean position intensity of the summed pixels weighted using the color probability. This process is repeated till stabilization and position and intensity is stored for starting conditions in the next frame.

1. Defining a search window.
2. Finding the mean location and size through histogram lookup.
3. Move search windows position and size according to 2.
4. Repeat step 2 and 3 till convergence.
5. Store location and size, and go to next frame.

Properties

The algorithm is fast, noise robust and easy to implement.

Now we want to remember, that we might want to track more objects with the same color, and as we just saw in the outlining, the algorithm is not able to do this directly, although it provides separation or abstraction from objects that get outside the tracked objects' search area.

3.1.9 Anaglyph Stereo

The principle of stereo in general is that our two eyes have slightly different positions and therefore they see slightly different images. Creating artificial stereo is done presenting two different images for the eyes.
Through history this has been done in countless ways, but here we will now look at the anaglyph approach.
Placing different color filters in a pair of glasses, has the effect of only letting a specific spectrum of light pass through to each eye. By encoding our stereo image so each image has the color that is passed by one filter, and blocked by the other, it is possible to let the eyes receive two different images. Stereo is finally achieved by letting these images be slightly different according to the stereo parallax of the eyes.

Computer Graphics

In real-time computer graphics the two images required are usually easy to produce. To encode these in the right colors can be done in several ways in hardware. Here two ways will shortly be outlined.

The first step is very simple and can be implement even on elderly graphics hardware. It is outlined in the following steps:

1. Render the scene into two different texture maps (or pixel-buffers) only storing the intensity of each pixel.
2. Enable the texturing mode: "Modulate".
3. Draw the first texture while modulating with the first filter color (e.g. red).

4. Enable the blend mode and set the blend function to mix 1:1.
5. Set the color to the second filter color (e.g. blue).
6. Draw the second texture.

The other method make use of a shader-program, that can lookup a pixel value in a texture using the pixel colors from the to textures as input.

1. Generate the constant lookup texture.
2. Render the scene into two different texture maps (or pixel-buffers) only storing the full color at each pixel.
3. Render to the screen making a per pixel lookup in the predefined texture map, using the two rendered textures values (found also by texture lookup) as argument.

Using a lookup table allows for more advanced color coding enabling more colors than the two used in the filters. This is done in the patented solution by [15] called “ColorCode” that gives a nearly full color picture in one eye and an intensity picture in the other. The brain uses only intensity to resolve the depth information, and is able to extract the color from the one eye watching the stereo.

3.1.10 Texture Volume

A texture volume enables visualization of a volumetric intensity field.
Often this is done holding the data in a discreet voxel representation, where a voxel is the analogous to a pixel in 2D.
By having a stack of 2D textures (or even one 3D texture) the volume can be rendered by drawing the stack back to front. Letting the alpha channel hold the intensity it can be compared to a threshold determining whether a voxel is to be drawn or not. Transparency can be obtained by using the alpha channel as a blend factor [23].

3.2 Existing concepts

In this section we will look at existing techniques and hardware in a row of applications related to the concept proposed in this thesis. This way we try to draw a picture of what is the foundation and inspiration of the proposed concept, and in what way it can offer an alternative to the galleria of concepts already being used.
Whether one technique is better than another depends on many factors beside the concept itself. Price, precision, speed is a few to mention. However in this section we will try to focus on presenting the concept itself and what it is capable of as concept rather than comparing all of them.

3.2.1 Virtual Reality

As mentioned a lot of effort is put into development of systems capable of virtual reality and augmented reality, including human tracking, device tracking and 3D interaction.
Virtual reality (VR) was introduced in the 60s, and the concept of the first prototype is well known due to a growing market of applications, and several science fiction novels and movies on the subject.

Traditionally the equipment of VR includes some sort of head-mounted display (see Figure 8) and also motion tracking equipment like gloves.
The users are tracked without noticeable lack through magnetic, inertial or optical technology or combinations of these.
The user sees the virtual world in stereo through the display in the proper view to his or her head posture and movement.

Figure 8: Head-mounted display for virtual reality.

Displays exist at low resolution for a reasonable price, but also at higher resolutions.

Large Room VR

VR that uses magnetic field orientation techniques are limited in space. Large room VR uses optical markers placed in the ceiling and a head-mounted camera to track a user's placement in a large room. Combined with a wearable computer, initial trackers and the standard VR equipment VR in large rooms are enabled [26].

Caves

By projecting the virtual world on walls surrounding the viewer an illusion of being in this world can also be achieved. Unless the head position is tracked perspective distortion will be present when the users head gets too close to a wall. If the head is tracked there can be only one user at a time.

3.2.2 Augmented Reality

The goal of augmented reality is to blend the virtual and real world in such a way that the virtual world is optically percepted as a part of the real world. It is usually done by recording a view of the real world, finding its orientation via image analysis and add the virtual world in correct orientation and occlusions [30].
Coded markers or known objects can be used to represent different virtual objects. By moving the marker or known object in the real world the virtual object is moved in the augmented scene.

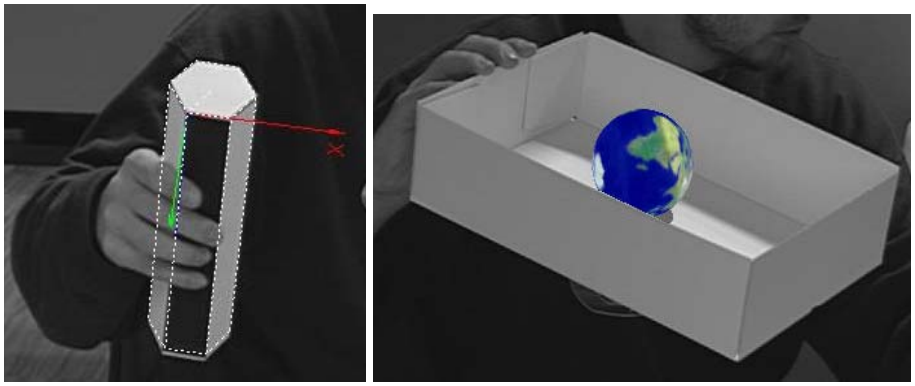


Figure 9 : Augmented reality. After registering a known objects orientation in the image, 3D graphics of virtual objects including some occlusions can be overlaid.

3.2.3 3D Interaction

Here we see a few devices addressing the problem of interaction indirectly or directly with virtual 3D-data.

Stationary devices

The SpaceBall, seen in Figure 10, is a typical stationary desktop device handling 6 DOF input.

Figure 10: The stationary SpaceBall from Logitech has 6 DOF.

VR-Gloves

VR-Gloves have been used in VR since its beginning, measuring the position and gesture of hands. In some cases it even can provide force feedback, when the user touches a virtual object.

Figure 11: Virtual Reality-Glove by InterSense.

Workbench

A system using a more or less horizontal large screen capable of stereo are sometimes referred to as a virtual workbench. Via tracked devices, model creation and painting directly in 3D is enabled. The tracking method is typically magnetic.

Reach-In

One setup that in its properties is very similar to the proposed concept (although much more precise) is the setup adopted by Reach-In. Stereo is produced via flicker glasses and time interleaved stereo images shown on a standard monitor. The monitor is placed so that the screen is best seen reflected in a semi transparent mirror (the image shown have to be upside down) producing the impression of the virtual world being behind the mirror (with nearly correct accommodation). A force feedback mechanical arm is placed behind the mirror to interact with the virtual world (see Figure 12).

Figure 12: 3D interactive stereo rendered setup by Reach-In.

3.2.4 View dependent rendering

To obtain viewer alignment the view of the viewer (eye positions of the viewer in relation to the screen and the virtual world) has to be known. In standard VR the eyes' positions are constant in relation to the display, so only the position in the virtual world has to be taken care of.
To produce view dependent rendering at a monitor the relation between the eyes and the monitor has to be considered as well.
This means that head-tracking is needed.

Head-Tracking

Tests have been made to find the effect of viewer alignment at a desktop computer (also called virtual-holography or light-field rendering) with special focus on the involved latency. In this case a mechanical tracking arm was used for the head-tracking (see Figure 13), but only horizontal movement was compensated for in the rendering.

Figure 13: Head-tracking in view dependent rendering or virtual holography.

The limit for noticeable latency was found to be 15 ms, with standard deviation of 3.1ms (ca 55-85 fps). It is worth mentioning that latency threshold was increased during repeated sessions meaning that the test person was getting more tolerant [24].

3.2.5 3D Screens

The 3D screens or auto stereoscopic screens have good potential as future 3D representation media. They use many different techniques with different properties. Here a few of them will be mentioned.

Viewing zone

The auto stereoscopic screen from sharp is integrated in a laptop computer as principally depict in Figure 14. The stereo effect can be switched off, so the screen can function as a standard 2D screen. The head has to be placed in a viewing zone in front of the screen to make the stereo effect work; otherwise the pictures to each eye will be mixed [13].

Figure 14: Principal illustration of Sharp's laptop with integrated auto stereoscopic screen.

Multiple views

To enlarge the viewing zone some screens uses multiple views. Images of more than two views are sent in multiple directions, so that the eye sees two proper images where ever they are in the enlarged viewing zone. The zone is usually repeated to cover all the views around the screen. However there will still be a mix of images when going from one image group to the next. In these screens the horizontal moving parallax is working correct, but only one of the image groups has the correct viewer alignment.
These screens have the unique auto stereoscopic ability to be viewed by multiple viewers at a time, but also need a constant generation of typically eight images (one for each view).

Eye-tracking

Alternatively to the multiple view systems some screens use the approach of directing the two images towards the eyes although moving. This obviously requires the eyes position to be known. In the monitor of SeeReal, hardware is implemented to continuously track the eyes of the viewer. The beam splitter, that splits the images to the eyes, is then moved according to the eye positions as seen in Figure 15. The eye position can be read back from the monitor to be used in the generation of the proper views, so that viewer alignment and correct moving parallax is achieved.

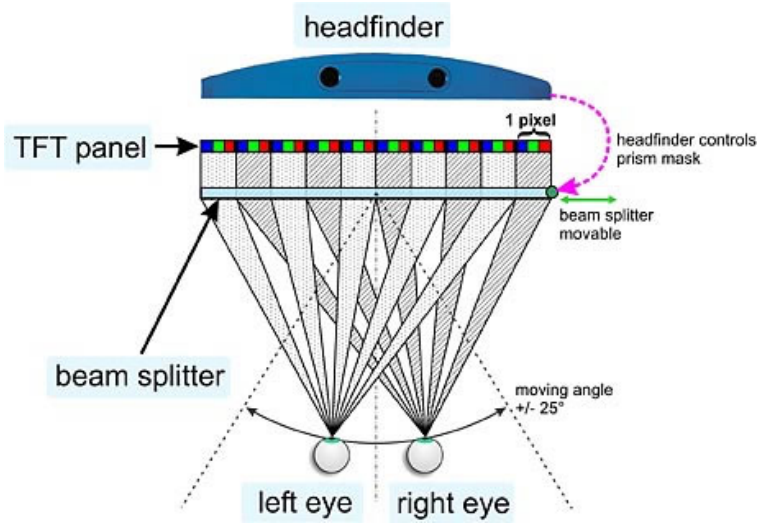


Figure 15: Hardware head-tracking and a movable beam splitter enables stereo at a large view zone in SeeReal's auto stereoscopic screen.

Hand tracking

At the Fraunhofer Institute they have extended the concept of the screen with hand tracking and eye-gaze tracking [14]. Two cameras under the screen tracks the hands of the user, and thus direct interaction with the 3D-data is possible as seen in the principle illustration in Figure 16.

Figure 16: Hand tracking makes direct interaction with the 3D-data possible.

Chapter 4 Analysis

Intro

In this analysis the goal is to sort out how a system can be made fulfilling the given requirements (see Objective in section 1.4 and Hypothesis in Chapter 2).

To do this, we first make an overview of the dataflow, allowing us to consider the system as a group of smaller segments, with specific input and output. Secondly we will analyze each segment separately.

4.1 Dataflow

The system can be divided into segments that in theory work independently of each other. Each segment takes a specific type of input and produces a specific type of output. Even now before the details of the approach are clear, we can make an estimate of what the main segments will be.

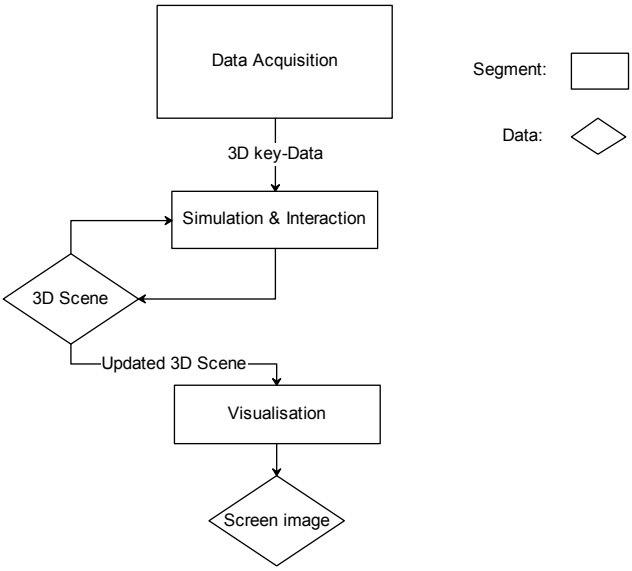


Figure 17: Outline of the systems segments and dataflow. Each segment can be analyzed separately when the input and output format is known.

As seen in Figure 17 the system can be divided into the following segments:

- Data Acquisition
- Simulation and Interaction
- Visualization

In the following sections we will look into the details of each of these segments.

4.2 Data Acquisition

In order to have a functioning pointing device and to produce viewer alignment, a few key-data (e.g. pointing device orientation and eye positions) from the real world have to be acquired in real-time. It can involve acquiring of great amounts of data, which is processed to extract the key data. A predicted process is outlined in Figure 18.

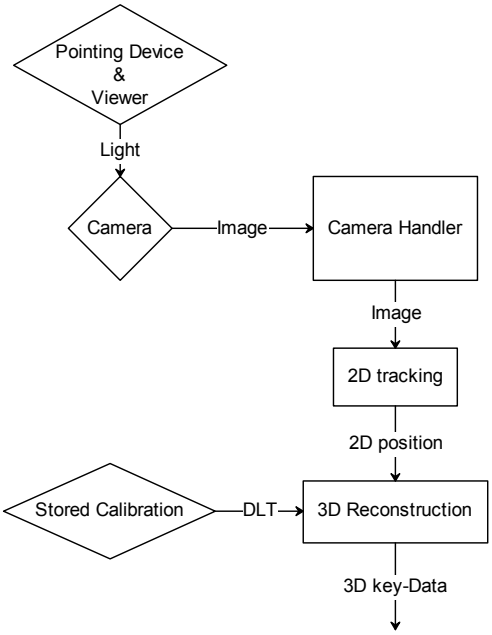


Figure 18: The sub-segments and dataflow of data acquisition segment.

Key-Data

The key-data should be thought of as the output from a given tracking algorithm. In our case, the key-data needed depends on how many degrees of freedom our pointing device is to have, and which kinds of viewer movements we should be able to handle (i.e. the viewer DOF).
By assuming that the key-data is a number of 3D points, we can determine the exact number needed for a given purpose. In Table 1 the DOF and the number of 3D-points needed for the same representation, is given for basic spatial representations.

Spartial representation	DOF	3D-Points
position on a line (1D)	1	1
position in a plane (2D)	2	
3D-point	3	
direction *	2	2
position and direction	5	
position, direction and scale	6	
rotational orientation **	3	3
rotation orientation and scale	4	
orientation	6	
orientation and scale	7	

Table 1: DOF and the number of 3D points needed to determine different spatial representations.
*** The direction can be found using only one 3D-point, but usually it is needed to be independent of the origin of the coordinate system, so a base point has to be provided.**
**** Rotational orientation is also added an extra 3D-point as base to be independent of system origin.**

Example Setups

To get an idea of the representation needed, two short examples of imagined setups are here given:

In the simplest setup only two 3D-points are needed; one for the viewer alignment and one for the 3D pointing device, each of which have 3 DOF.

In a second setup the pointing device is also meant to give the orientation of the object (6 DOF). In addition the viewer can tilt the head, which there has to be compensated for (2 DOF found using two 3D-points), ending up with the need of five 3D-points (8 DOF).

4.2.1 Choosing an Approach

So we need a number of 3D points (2-6 presumably) extracted from the real world, and the question is how to get them. This is a question of what kind of properties we want the system to have. A lot of methods have already been produced and many can be assumed yet to be developed [14][17]. To limit the analysis the decision of an approach can be done in a long row of steps, where each step involves an analysis. What to analyze depends the prior choice. Each analysis and choice narrows down the specifics of the method and gets us closer to a final approach. This also indicates that lots of deeper branches are not considered. To keep track of the most important choices made through this project a “map of choices” has been made (see appendix 10.1. A single choice is shown in Figure 19.

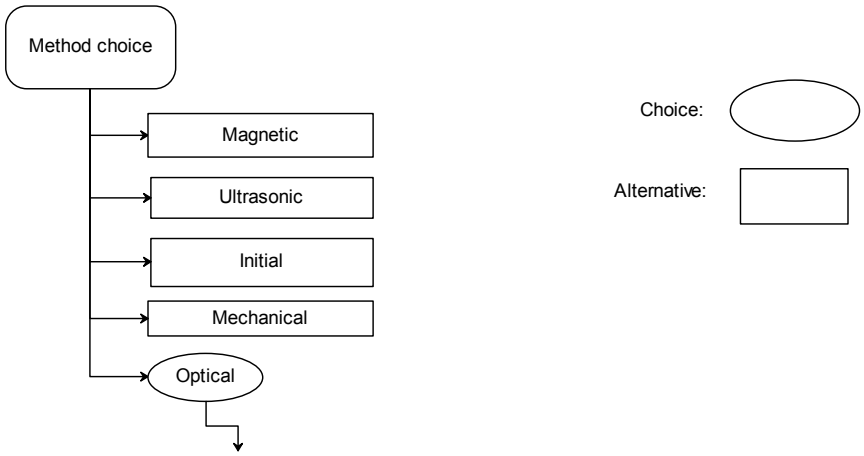


Figure 19: Sub Decision path. An optical position aqusition method among 4 alternatives was chosen.

The first choice to make is the main method. It has been indicated, that a webcam based method is chosen, but to understand what special properties and consequences this implies, we will compare this approach to other alternatives.
As a little reminder to the objective of the object tracking, two lists about virtual reality components according to [6] are presented:

Tracking devices:

- Tracking devices are usually used to measure the motion of the user's head, hands and eyes
- 6 degrees of freedom are required to describe the orientation of the object in 3-D space
- Tracking are currently: electromagnetic, mechanical, optical, acoustic (ultrasound), or inertial
- Tracking device quality depends on resolution, accuracy, and system responsiveness (sample rate, data rate, update rate)

Virtual displays:

- A VR visual display must provide a stereoscopic view of the virtual environment
- Head-tracking must enable continuous updating of the stereoscopic view
- The factors that affect the quality of a display are:
    - resolution
    - color vs. black and white
    - brightness
    - motion representation
    - ergonomic and health concerns

Each type of tracking device has its pros and cons (see Table 2). For VR, the magnetic type has been the most popular for a long time. For desktop use mechanical robot arms with force feedback are very popular. Optical

solutions are the most popular character motion tracking in the movie and game industry (sometimes in combination with mechanical cloves), due to the possibility of recording very large numbers of points using post calculation if not real-time.

	Magnetic	Ultrasound	Inertial	Mechanical	Optical	Webcam -based
Pros:						
Low latency	+	+	+	+	+	
High update rate	+	+	+	+	+	
Precision	+		+	+	+	
6DOF	+	+	+	+	+	+
Force feedback				+		
Relative Price			+			+
Cons:						
Wear equipment		!	!	!	(!)	(!)
Drift			!			
Wired	(!)	!	(!)	!		
Range limited	(!)	(!)		!		
line-of-sight requirements		(!)			!	!
Light sensitive					(!)	!
Metal sensitive	!					
Sound sensitive		!				
difficult calibration	!				(!)	(!)
Price	!	!		!	!	

Table 2 : Superficial table of tracking methods pros and cons. Here "+" markes the pros and "!" marks the cons. "(!)" marks when a con is partly or can be avoided in special cases. Many pros and cons also vary within the same product group.

Magnetic Field Orientation Registration

A very common way of tracking in 3D is magnetic field registration. Since 1975 AC and DC active source systems have been used [6]. It is especially popular because no line-of-sight is required.

Ultrasound

Ultrasound tracking works by measuring the time-of-flight of sound from typically three transmitters to three microphones. Knowing the position of

the transmitters, the speakers' spatial location and orientation can be found using triangulation.

Inertial

Inertial trackers continuously measure the acceleration, which is used in the calculation of the position.
Inertial trackers have become superior to other trackers in many areas, because of their low latency high update rates and precision and the fact that they are the only interference free tracking tool. Their big drawback is that they work by updating the position and the orientation by adding measured change to the old position (compensating for the gravitational field at all times). This means accumulation of errors and results in unacceptable drift in very short time (from a few seconds to 15 minutes depending on quality and price).
The inertial trackers have become of greater interest through combination with other trackers in hybrids, where the inertial tracker gives real-time position and orientation. The other tracker updates the inertial tracker as often as possible, so the drift does not reach a critical level.

Mechanical

By using some kind of exterior skeleton and measuring angels in every joint it is possible to track the position of every bone including the last endpoint, which usually is the point we are interested in. They possess the very exciting possibility of force-feedback, but are in some cases also tiring the user.

Optical

The growing market of fast or cheap optical sensors has increased the interest for equipment free remote sensing possible with optics [14][20], but still the typical way is tracking some sort of markers where high precision (0.2 mm [18]) in large area (400-500 square foot [18] [19]) and at high speeds (1500Hz [18]) has been reached.
In the movie and game industry optical tracking is used for character tracking. A person wear a suit with a number of reflectors, and several cameras records the motion in a studio with the proper lighting. In post processing the 3D position of all reflectors are reconstructed.
In virtual live TV-shows, infrared reflectors are placed on the cameras to register their position. The camera position is used in the generation of the virtual scene set.

Web Camera-Based

Naturally, web cameras of very low quality are the cheapest optical sensors on the market.

4.2.2 Discussion on the choice of Web Camera-Based Optical Tracking

As seen in Table 2 (p. 37), other methods are superior to the webcam-based tracker in many ways, speed being the most important. The web camera

wins on price though (especially since many people have one already) and good prospects of avoidance of troublesome worn equipment. As we will see next, it has got good future potential too.

Latency

When tracking a position and using it for view generation two latencies are involved. One is the time it takes to generate the image and present it for the viewer. Here the important latency is the time it takes from a position movement happens in the real world to it is tracked and accessible for the visualization part.
We have to remember the application, when discussing the choice. The speed is important to "keep the illusion". If the latency gets too high, the user notices that the world and the virtual world are not aligned during movements. In VR this can cause seasickness. At the desktop it seems like the virtual world is swimming around trying to keep up with the real world. The limit where the effect stops being detectable at all is 55-85 fps but can vary individually as much as from 30-140 fps [24].
For the pointing device, latency is not as serious. But it is a problem for the feel of real-time interaction and alignment of the tool in hand and the virtual tool seen on the screen.

Predictors

Tracking the head at a desktop does not necessarily mean a lot of fast head movement or big accelerations. With a predictor like a Kallman-filter, much of the swimming can be prevented at slow head accelerations. But overshoots will still appear at quick jags with the head, like head shaking.
A predictor for a pointing device though could turn out to be a problem because people tend to have very fast and abrupt movements, and the predictor would consequently cause annoying overshoots.

Line-of-sight

The line-of-sight requirement is not a big issue for head-tracking at the desktop. Unless the user waves the arms around a lot, the head will usually be visible at all times. But for tracking a pointing device, it could easily turn out to be a problem. The user's hands could occlude the pointing device, so the cameras have to be carefully placed with thought on right or left handed use.

Precision

Regarding precision it is again of much greater importance for the pointing device than for head-tracking. It is important that the movement is smooth more than precise in head-tracking, though the precision is responsible for the virtual tool alignment.

Quality vs. Price

We have accepted low quality in many areas in trade for low price. In this choice it is of importance that it is possible to restore much of the quality within the camera-based tracker (assuming the software problem is solved)

by using better, and more expensive, cameras. Better here means faster, more light sensitive or with higher resolution. Cameras come in frame rates of 24-1000 fps (webcams 12-60 fps) (2004) and with several mega pixels (webcams up to one) so we still have the possibility of making high end solutions with our choice. Additionally cameras would heighten precision and reduce the risk of pointing device occlusion.

The Ultimate Solution of Today's Technology

Considering what would be the best alternative solution if price didn't matter can also tell us a little about the quality of the choice. For very high precision on tracking, a hybrid of inertial trackers and optical or ultra sound trackers could be used. The inertial tracker would provide smooth real-time updates as fast as asked for, and the second tracker would prevent drift. Now we would like to have equipment free head-tracking, so we replace the head tracker with an infrared camera, which enables the tracker to work in almost any light condition. We would also like force feedback on the object interaction, so we choose a pen attached to a mechanical arm (phantom). Could it be better with an equipment free direct hand interaction? Possibly in some cases, but the fact is that you can turn and move a tool in your hand faster and more precise than your hand or fingers themselves, so let us stick to the force feedback tool.
Compared to our choice, the "ultimate" solution is not that far away, though the force-feedback part of course cannot be solved with the cameras.

Future Prospects

If the optical sensors keep getting less expensive, it is possible that webcams will soon be high quality cameras and maybe faster than now. That would solve the problems on precision and speed (assuming perfect tracking algorithms), leaving us with a high quality system at low price.

Light vs. Infrared

It could be considered to use infrared cameras that work in the dark as well, by applying some infrared diodes on the pointing device. Using webcams it is given that we have to stay in the visible range of light. Hence it is harder to segment out the tracked object from the surroundings than for instance is the case with infrared where only the infrared reflection of the tracker is visible. On the other hand visible light sensors might have an advantage in image tracking. Visible light sensors separate colors, which can be used to distinguish objects of different colors. In the infrared case all tracked objects are identical "white blobs" in the image, and some sort of arrangement has to be applied to separate each point unambiguously.

Next Step

Having chosen visible light image based tracking, we are still left with the choice of the tracking algorithms. We would like to track some points in space representing a tool and track the eyes of the user. We would like to be able to do it in real-time and with low latency.

4.2.3 Object Tracking

We are looking for a fast and somewhat precise method to track one or more points in a 2D image.

The Human Analog

When the human brain interprets an image pair it is done in a complex combination of several parts of the brain specialized in different low level and high level processing starting with the simplest processing already on the retina of the eye [22]. When one tries to make a tracking system (or image understanding in general) one usually starts out by imitating one or more of the low level techniques. A high level technique might then be applied also using some sort of prior knowledge.

Low Level Techniques

Here is a non exhaustive list of some of the low level abilities of the brain.

- Color field
- Edge detection
- Field of angular orientation
- Curvature field
- Corner detection
- Depth field
- Motion field (using background subtraction)
- Element separation

Some Higher Level Abilities

Among the higher abilities of the imaging system of the brain can be named:

- Object separation in 3D
- Object orientation and movement in 3D
- Object shape in 3D
- Object recognition
- Object categorization
- Face recognition
- Simple face expression interpretation

We don't need the unmatched abilities of the brain to track a known object, but we sure can be inspired by them in our search for general, robust and fast tracking algorithms.
A lot of development is going on at the moment in image analysis, and many very advanced methods have been presented in literature.
Because of its speed and ability to separate objects, a color tracking method is chosen. Here some of the alternatives will briefly be mentioned.

Feature Tracking

Feature tracking is based on prior knowledge of the shape of the tracked object and is very common in visual tracking. It can be done using outline

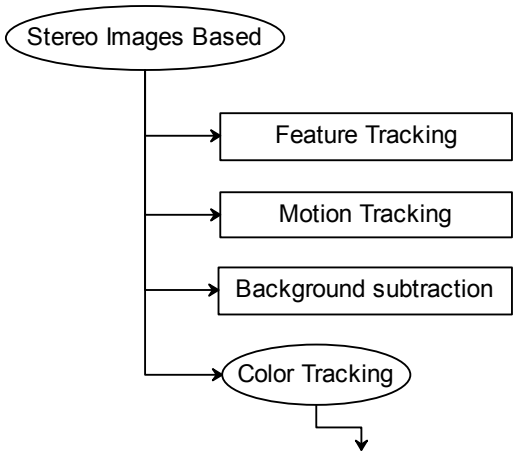
tracking, comparison with 3D models or be based on simpler shapes like circles and squares.

Motion Tracking

Motion Tracking was one of the early tracking methods used, because of it simplicity. An image sequence is analyzed for changes and their direction. This way the moving part in an image is easily identified. Some animals (like frogs) are said to have vision based purely on motion.

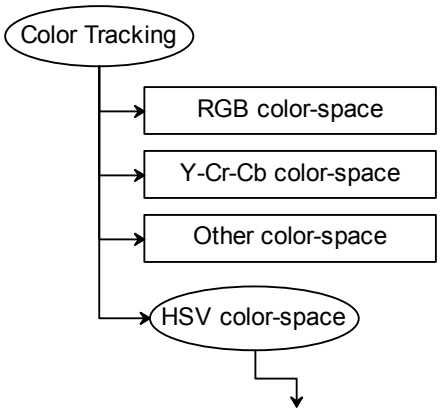
Background Subtraction

Background subtraction is similar to motion tracking. It works by first registering the background. Afterwards any object can easily be tracked by background subtraction, which only leaves the object that changed the image.



4.2.4 Color Tracking

Color tracking is simple and fast. To track colors one has to find a suitable color space to track in. A common color representation is RGB, but it is not easy to recognize a color only using its RGB values unless the color matches exactly. During tracking one would like to be independent of lighting conditions. Also one would like to tell, when testing a color, how far from the matched color it is. For this purpose the HSV (hue, saturation and value) color-space is well suited.



HSV Color Space

Hue is a periodic color representation, which through a rotation of 360° degrees goes through the visible light spectra and connects the highest frequencies (blue and violet) and the lowest frequencies (red) smoothly through magenta and purple (as seen in Figure 20). Two complement colors are shifted 180°. The higher saturation, the more precise the Hue can be calculated. Value or brightness is the component that we would like to be independent of (although it indirectly dependent on the saturation), and therefore we can ignore it.

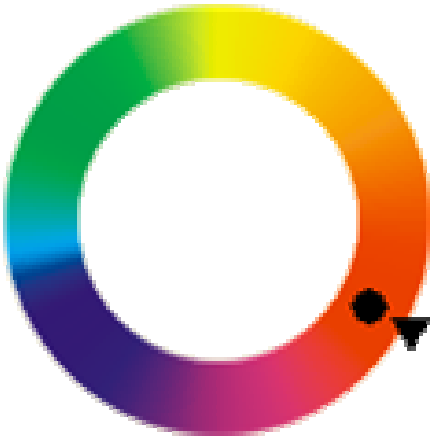


Figure 20: The hue, varied through all 360° degrees round the circle.

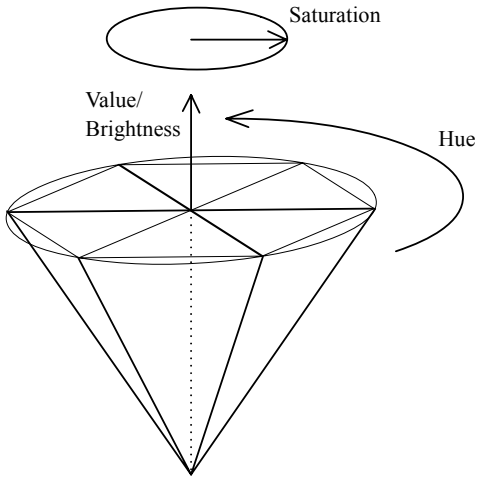


Figure 21: HSV Color space

Converting from RGB to HSV is not a cost free procedure, and it should be considered how to do it.

Converting from RGB to HSV

For each pixel to be color matched a RGB to HSV conversion has to take place. Some of the costs can be reduced by making criterions for early rejection (e.g. the color is too dark, or have too low saturation). This way none or only a part of the conversion has to find place.

Several different variations of HSV color spaces exist. Here we use a conversion to HSV as defined in (34) - (37):

$$\begin{bmatrix} I \\ v_1 \\ v_2 \end{bmatrix} = \begin{bmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ -\frac{1}{2} & -\frac{1}{2} & 1 \\ \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{34}$$

, where

$$S^2 = v_1^2 + v_2^2 \tag{35}$$

$$S = \sqrt{v_1^2 + v_2^2} \tag{36}$$

, and

$$H = \arctan\left(\frac{v_2}{v_1}\right) \tag{37}$$

Using saturation as a threshold implies that it should be calculated. The calculation of (36) is much more expensive than (35). By squaring the

saturation threshold and compare it to S^2 in (35) instead of S we reduce the computational load.

4.2.5 Adjusted CAMSHIFT

Making an exhaustive search through the total area of the image seems like a waste of time if the object only takes up a fraction of the image. To speed up things one can limit the search area, but one should be sure to have a way of fast reinitializing, if the object is lost (e.g. due to occlusion).

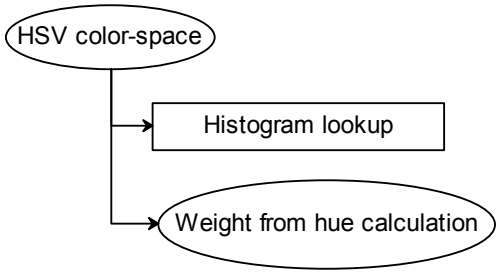
To limit the searched area, one can use a strategy where only a search window around the object's old position is used (see also Figure 7 in section 3.1.8.1). For this purpose, the face tracking algorithm CAMSHIFT, based on the mean shift algorithm, is well suited (see section 3.1.8.2).

It suffers from the limitation of not being able to track more objects with the same color, but by combining it with another algorithm (i.e. connected component analysis or feature recognition), could possibly enable tracking of multiple color objects. This would also reduce the tendency to get distracted by nearby objects with the same color.

Histogram vs. single color

Using a hue-histogram as lookup table is a good solution in face tracking, where the face can have colors in different parts of the spectra, but in single color tracking it is not a good idea, in spite of its speed.

One has to generate the histogram. This can be a clever way of selecting one's target object, but could also lead to errors or a non optimized result. Also if the histogram has - let us say - two evenly high peaks of different hue, the colors in each of these peaks will be weighted evenly. This is good when you want an even weight of the two parts of the object, but has a drawback. A single color surface of any of the peak colors would have at least the same weight as any part of the tracked object, which would lead to the tracking of the surface instead of the object. When tracking a single color we are better off with less distracters achieved by calculating the weight based on a single color.



Color weight

Calculating a **weight** $w(x,y)$ from a single color comparison $\Delta\hat{H}_{x,y}$ can be done in several ways. First of all providing a **width** H_{width} in hue-space within which the color should be to have a weight at all, simplifies things a lot.

$$w(x,y)=\begin{cases} k(\Delta\hat{H}_{x,y}) & for \quad -H_{width}<\Delta\hat{H}_{x,y}<H_{width} \\ 0 \ , else\, where \end{cases} \tag{38}$$

, where

$$\Delta\hat{H}_{x,y}=H_{x,y}-H_{object} \tag{39}$$

and $k(\Delta H)$ is some kernel function with the output interval zero to one.
The simplest way is now to linearly calculate the distance to the matched color in hue space $\Delta\hat{H}$, and see it relative to the width.

Figure 22 (a) shows the following formula:

$$w(\Delta\hat{H}_{x,y})=1-\frac{\left|\Delta\hat{H}_{x,y}\right|}{H_{width}}$$

For simpler notation in the following we define:

$$\Delta H^2=\frac{\Delta\hat{H}_{x,y}{}^2}{H_{width}{}^2} \tag{40}$$

, where

$$-H_{width}<\Delta\hat{H}_{x,y}<H_{width}$$
$$\Rightarrow$$
$$\Delta H^2\in[0,1] \tag{41}$$

Gaussian Distribution

A Gaussian distribution would on the other hand be what one would expect, but is more expensive, and might not necessarily produce the best results in all situations anyway. The Gaussian distribution also has the property of being non zero in the entire search spectra, which removes the

possibility of early rejection, unless one truncates it at the search width H_{width}.

$$w(x,y) = \begin{cases} e^{\left(\frac{-\Delta H^2}{a^2}\right)} & , \quad -H_{width} < \Delta H < H_{width} \\ 0 \ , else\, where \end{cases} \tag{42}$$

, where a is the width of the Gaussian distribution (shown in Figure 22 (c) for $a = 1,\ a = \frac{1}{2},\ a = \frac{1}{3}$).

The fast and Simple

Taylor expanding the Gaussian distribution to second order gives the following computationally fast weight distribution (see figure ? b)):

$$w(x,y) = 1 - \Delta H^2 \tag{43}$$

The interesting thing about (43), is that when raising it to a higher order, (44), it becomes very similar to the Gaussian distribution. Although it does not have all its properties, it has one essential advantage; it (like (43)) goes through $(\Delta H, w(\Delta H))$ = (-1,0) and (1,0) as seen in Figure 22 (d) for $a = 1,\ a = 4,\ a = 9$.

$$w(x,y) = \left(1 - \Delta H^2\right)^a \tag{44}$$

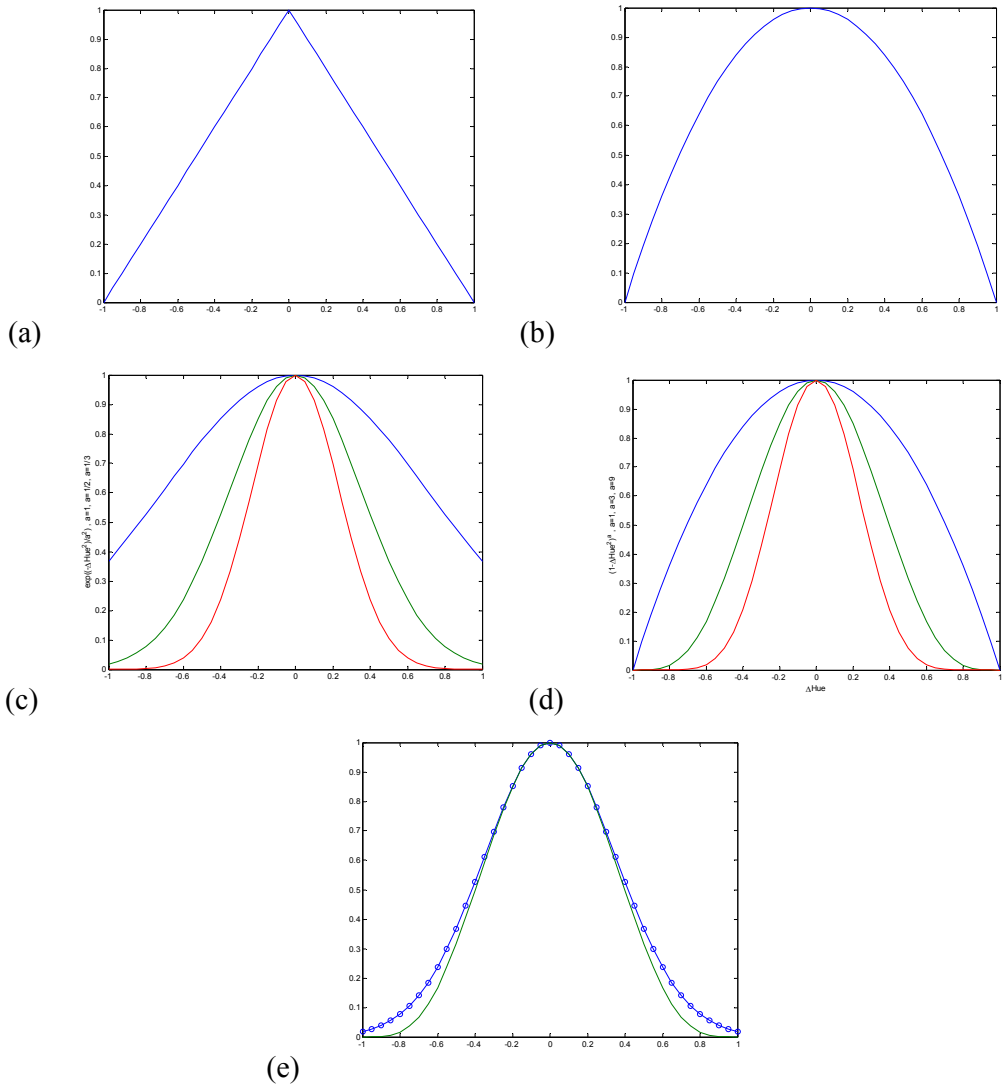


Figure 22: Weight distribution as function of ΔH (a) Liniar weight ($1-|\Delta Hue|$) is the fastest to calculate (b) Square version ($1-\Delta Hue^2$) is nearly as fast but nicer (c) Gausian weight distribution for $a=1, a=\frac{1}{2}, a=\frac{1}{3}$ (d) Taylor of Gausian (e) Comparison of Gausion $a=\frac{1}{2}$ (circles) and the faster (44) a=4.

The fast weight function (44) with $a=4$ seems to be well suited for the purpose.

In Figure 23 it is seen how the colors of an orange sphere is weighted using the chosen function. Any of the functions could easily create a lookup table to use for even faster processing like the histogram.

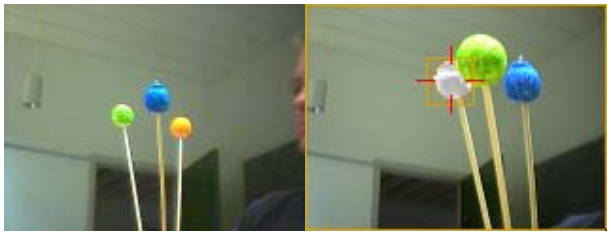


Figure 23: Color weight. The orange colors of a sphere are changed to gray scale representing its weighted value with light as highest weight and dark as low weight. The sphere is close to the camera, to make the weight distribution more clear.

Choose Saturated Colors

Although choosing a fast method for calculating the weight is important one should remember, that a far more important factor is avoiding the calculation by using the saturation threshold. This also means that the more saturated objects we track, the higher threshold we can use, resulting in more robust tracking and less computations.

Multiple Colors

Having found some alternatives to the original histogram of the original CAMSHIFT algorithm we now seem to have a single color tracking algorithm. We would like to apply it on several objects with different colors at the same time, where as the CAMSHIFT model was designed to track one person at a time.
We can do this remembering to save the found location and size of each color respectively. We don't have to take care of all colors at once when receiving an image, because each color's search window probably is located in different parts of the image.

4.2.6 3D Reconstruction

When the 2D positions of the object are found the reconstruction of the 3D position can begin. More than one method exists to do this.....

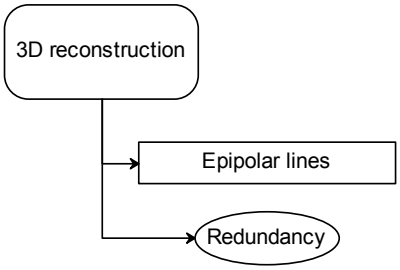
Reconstruction 3D data from obtained 2D positions, from 2 or more cameras, is a well known problem in photogrammetry [4]. 2D positions from 2 cameras provide 4 equations to solve getting the three unknowns of the 3D position. This indicates redundancy.

Redundancy

Redundancy is in some cases avoided by reducing the system and makes the search algorithms faster by first finding the object in one image and only search for it on one line in the other, the so called epipolar line. The redundancy can instead be used to tell us something about the reliability of the tracked 3D position, and determine whether we want to use the given point.

Using the adjusted CAMSHIFT algorithm (see Section 4.2.5) the search area is already reduced, and direct use of epipolar lines would not speed

things up much. Instead the residuals are nice to have, also because they apply to any number of cameras.



We choose to use the 3D reconstruction method described in section 3.1.4.

4.2.7 Camera Calibration

3D reconstruction heavily relies on camera calibration, according to (11).

Reference coordinate system

As we will look further into in section 4.3.1 on page 58, it is important to know the reconstructed 3D points in relation to the monitors position and orientation. We use a 3D-monitor-coordinatesystem for all real world positions. Hence we have to know the monitor orientation in the coordinate system of the cameras, or even better let the projection matrix be obtained via camera calibration be described in the coordinate system of the monitor.

Fixed cameras

If the system is factory produced, and the cameras are fixed on the monitor (see Figure 15) it simplifies the problem, since it can be calibrated from factory. And the user does not have to worry about calibration at all.

Non-fixed Cameras

Unfortunately it becomes a little inconvenient for the user if the cameras are not fixed. Then each time the cameras or the monitor are moved, he or she has to recalibrate. We see that it is important that the calibration is easy, fast and precise.

4.2.8 Camera placement

Using optical calibration makes the camera setup much easier than if their position and direction were to be measured. The cameras can be placed at any position that is sure to be fixed during calibration and use, or even on the monitor, which then is free to be moved around.

Creating a suitable measuring volume

This all sounds very good but we still have a factor to take care of; the placement of the measuring volume. The measuring volume we define to be where at least two cameras views overlap and 3D reconstruction can take place.

We should make sure that the following things are within the measuring volume:

1. The viewer or the object on the viewer that should be tracked.
2. The interaction tool in front of the screen.

Tracking the viewer could be done placing the cameras on top of the monitor because the measuring volume would then extend from the cameras toward the viewer as seen in Figure 24.

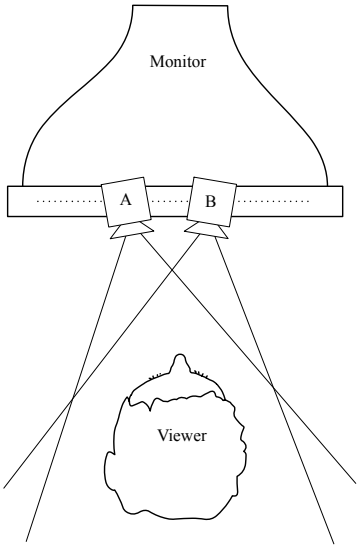


Figure 24: Camera setup with focus on viewer tracking. The overlapping views of cameras A and B define the measuring volume.

On the other hand - when tracking an interaction tool, it can be assumed that the user wants to interact as close to the screen as nearly touching it because the user interacts with a virtual world that extends through the screen. If this is made possible by placing the cameras on each side of the viewer, pointing towards the screen, it should be noted that it heightens the requirements to the tracking algorithm, because what is shown on the screen is seen by the cameras, and gives rise to major distraction.
Furthermore occlusion e.g. by user or the monitor has to be considered as well (especially when tracking colors).

Optimal Precision at Right Angels

It should also be taken into account that the precision of the 3D reconstruction increase when the lines going from the cameras and intersecting through the tracked object, are near orthogonal.

The choice

A solution is chosen, where the cameras are placed close to the plane of the screen making a nearly right angle towards the axis going through the

center of the screen and monitor. Depending on their field of views, the cameras should be placed in a distance great enough to place the viewer inside the viewing volume as depict in Figure 25. The setup additionally avoids problems of user occlusions by right handed users (of course, by mirroring it would suit a left handed person). A disadvantage is that placing the cameras fixed in these positions is more difficult than just placing them on the monitor or the office table. Furthermore the setup is not well suited for tracking algorithms that rely on seeing the user's face from the front.

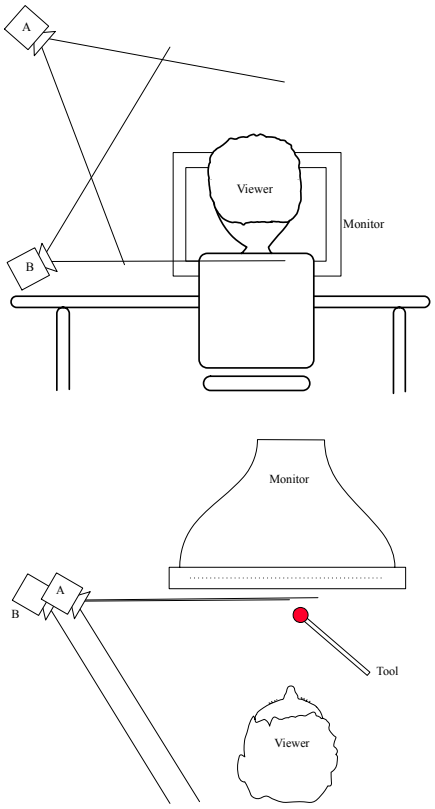


Figure 25: Front-view and top-view of the camera setup. The angle between the cameras gives good conditions for 3D reconstruction and both the viewer and the interaction tool is inside the measuring volume that even allows tracking objects very close to the screen of the monitor.

More Cameras

By adding more cameras, the task of setting up the system fulfilling the requirements, becomes less restricted, because each camera adds to the measuring volume.

4.2.9 Monitor and Viewer Calibration

Given calibrated cameras, we still need to know the position of the monitor. Furthermore, if the eyes of the viewer are not directly tracked, we need to

find the eyes' position relative to the coordinate system of the tracked viewer object.

The monitor placement could be found by hand in relation to the points used in the camera calibration, but it would be easier and probably more precise if it could be done using the already calibrated camera 3D position measuring system, or even better by combining it with the calibration itself. Here we will outline a few approaches dealing with monitor calibration, viewer calibration, or a combination of these and camera calibration.

Following approaches are outlined:

1. Simple viewer calibration
2. Monitor calibration in a calibrated camera system
3. Combined monitor and viewer calibration in a calibrated camera system
4. Combined camera and monitor calibration
5. Combined camera, monitor and viewer calibration

4.2.9.1 Approach 1: Simple Viewer Calibration

A very simple way of setting up the viewer-eye relation is by combining simple hand measurement and knowledge of human anatomy.
For example the tracking object registering the viewer can be a number of colored spheres on for example a hat worn by the viewer. The number of points used influences on how well head rotations are handled. With three points the full head orientation can be found, and therefore also the eye positions regardless of head rotation. The fewer tracking points used the relatively smaller will a hand measured error be.
If two points are used and placed on a line parallel to the line on which the eyes are placed then rotation around the vertical axis and head tilt will be found and used in calculation of the eye positions. Head nodding will not be registered and causes uncertainty in the eye position.
If only one point is used the rotation is not registered at all. This leads to errors in the plane of rotation proportional to the distance B seen in Figure 26 depending on the error in the angel, θ, of rotation:

$$\begin{bmatrix} x_{error} \\ y_{error} \end{bmatrix} = B \begin{bmatrix} \sin(\theta) \\ \cos(\theta) \end{bmatrix} \tag{45}$$

This implies that it is an advantage to place the center of the tracking object close to the midpoint between the eyes. It should also be noted that any error in the angles of tracking objects orientation gives rise to the same error (although θ here is much smaller).

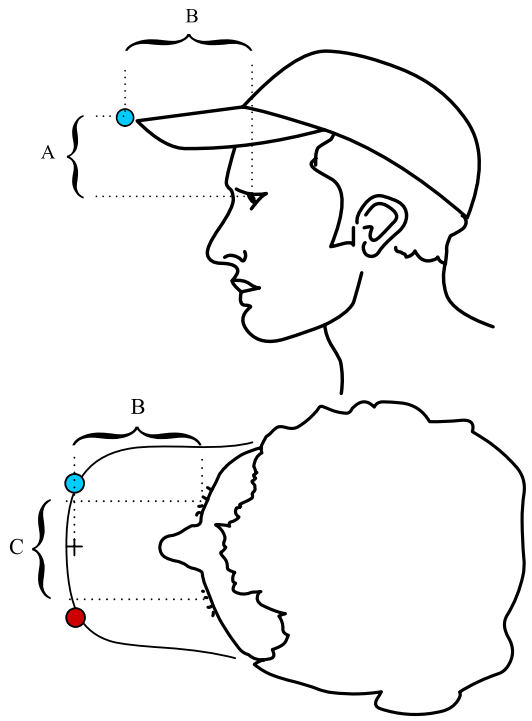


Figure 26: Viewer calibration; placing the eye relative to the tracked viewer object. The distances A, B and the eye separation C can be measured by hand.

4.2.9.2 Approach 2: Monitor calibration in a calibrated camera system

Given a calibrated measuring volume we in principle only need three points related to the screen to determine its position. But as mentioned in the previous section (section 4.2.8) we are not ensured that the screen is inside the measuring volume. This means that we somehow have to measure points remote from the screen and reconstruct the screens placement from known constraints. A stable but tedious solution would be to construct a frame fitting the screen and extending it into the measuring volume, and measure points on that. Instead it is proposed to use a simple stick including two tracking objects. By placing the stick in one corner of the screen at a time registering a few point pairs at each position, the position of the corner can be derived to the point where the lines of the point pairs intersect. If also the distances between the tracked objects to the end of the stick is known the position is even more constrained and can be found even more precisely.

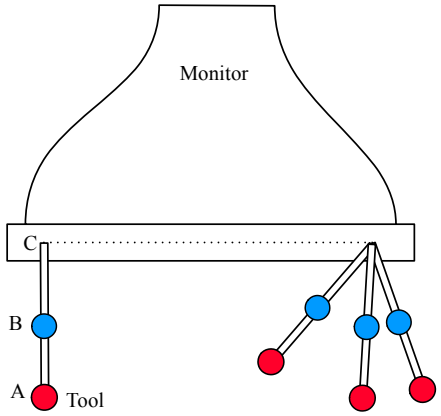


Figure 27: Top-view of monitor calibration. A tool or stick with two tracking objects, A and B, are placed at one corner, C, of the monitor at a time. To reconstruct the corner position, C, one can use the known distance from A to B to C. Furthermore when more positions are registered, as depict on the right hand corner, the intersection of lines going through the tracking objects gives the corner position if the stick distances between A, B and C are not known.

Usually monitors project the picture on the inside of the glass screen. The thickness of this glass has to be found in factory data, or simply by guessing! The image shift caused by the refraction of the glass screen is considered negligible.

Finding a good estimate

Finding a good estimate of the over determined corner positions can be done using least square adjustments (see section 3.1.6). Requirements on a rectangular screen can be included.

4.2.9.3 Approach 3: Combined monitor and viewer calibration in a calibrated camera system

This approach is in a way better than the two prior ones because it takes care of two calibrations at once, but also has high requirements, to the user.
Assuming we are capable of stereo vision, a set of stereo points can be shown on the screen. For each stereo point the user (wearing the viewer tracking object) points out the point in space with the interaction tool, keeping his or hers head still.
Choosing the stereo point distances to be respectively half the distance of the eye separation and the double of the eye separation makes it easy to calculate the eye point and the stereo midpoint. As seen in Figure 28 the distances called A, B and C in the figure are equal. In principle it now takes one set of such stereo point pairs to determine the eye position and three gives three points on the screen, which is enough to determine the screen position and orientation. More points could be measured to reduce errors.

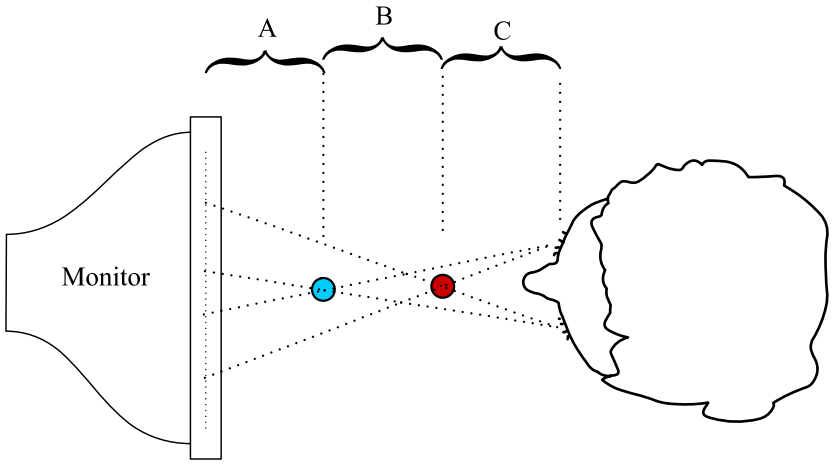


Figure 28: Calibrating monitor and viewer using stereo points. The stereo points can be arranged so that the distances A, B and C are equal.

As implied this approach heavily relies on the precision in the users' depth perception, which is also its disadvantage. Furthermore, the close points in the calibration process are so close to the user, that it is hard to focus on the object (the stereo image and interaction tool will have different focus lengths). The advantage is that for the user with good depth perception there is a good probability that the precision of the viewer calibration is so good that he is not able to see an error in alignment between the interaction tool and the virtual object on the screen (at least when having his head near the place it were under calibration).

4.2.9.4 Approach 4: Combined camera and monitor calibration

This approach is actually two different approaches, but for the user they are the same. It is based on approach 2, in that the procedure of finding the four corners of the screen is also nearly the same (see Figure 27).

It is based on three steps:

1. Find a rough estimate of camera calibration and monitor calibration.
2. Use the rough estimate as starting guess in a non linear iterative camera calibration finding a precise estimate.
3. Recalculate the monitor calibration.

The simple version is to stop after the first step.
To calibrate the cameras, a known calibration object with at least six non-coplanar points is needed. The idea here is to use the points obtained in the monitor calibration as calibration object. This is not a stable object, in that the stick is not surely perpendicular to the screen. Now we have a calibration that is only a rough estimate (which still is of same level of

quality as any other unstable calibration object), but a calibration of both the cameras and the screen at once.
A method of camera calibration called "Relative Camera Calibration" [8] depends only on at least 5 unknown 3D points coherently registered by both cameras. The relative camera calibration is non-linear because of transformations that depend non-linearly on rotation angels, as mentioned in section 3.1.7. It can therefore not be solved directly using least squares adjustments but in an iteratively process. Doing this is out of the scope of this thesis, but it can be done according to [8] and therefore this approach has good prospects in the future.

4.2.9.5 Approach 5: Combined camera, monitor and viewer calibration

By combining approach 3 and 4 it should be possible to make a combined calibration of camera, monitor and viewer at once using a calibration object defined using stereo points on the screen. As for the later half of approach 4 this approach is also out of the scope of this thesis.

4.2.9.6 The Choice of calibration approach

Approach 1 is chosen to be used, but also the simple combined camera-monitor calibration in approach 4 is used.

4.2.10 Summery on data acquisition

Through analysis we have chosen one of many ways to acquire the data needed in the system.
First optical data is recorded by at least two web-cameras. Colored objects are tracked in these images using an adjusted CAMSHIFT algorithm giving 2D positions. The 2D positions are reconstructed into 3D positions using camera data. The camera data is obtained prior to the operation in a camera calibration, which uses the monitor as a base calibration object.

Eye Tracking – A simple start

Here we have chosen eye tracking done by head mounted trackers. This doesn't limit later use of other methods, because it could be a good first guess to use in initial development of other methods like direct eye tracking. Tracking eyes are not a new problem and has been subject of interest not only in VR but also in gaze tracking for user interface analysis and interface alternative for disabled people. Real-time eye tracking is not an easy task though.

4.3 Visualization & 3D Interaction

In this section we will look into the visualization and 3D interaction within the system. The system is a prototype to prove a concept and therefore does not have a specific application oriented visualization purpose, but rather a demonstrational purpose.

Specifying

We will first look into the necessary abilities to produce the view dependent stereo rendering. Furthermore we will look into the visualizations that to some extend will be able to demonstrate what the use of the system in a final application could be like.
It is the power of viewer alignment and the 3D interaction which are the essential aspects. The 3D interaction enables editing methods not possible with e.g. the mouse.
The visualization of editable scene objects should also demonstrate and validate the functionality of the system. Among others the scene object types “Line Stroke” and “Texture Volume” are introduced to do this.

4.3.1 Viewer Alignment

Viewer alignment or view dependent rendering basically means to render the picture of the virtual scene, so that it forms the same image on the eyes retina as if the object were real. It is simply done by placing the virtual eye-point aligned with the viewer’s real eye position and render the scene by projecting it on to a virtual plane placed at the same position as the screen or monitor in the real world (see Figure 29).

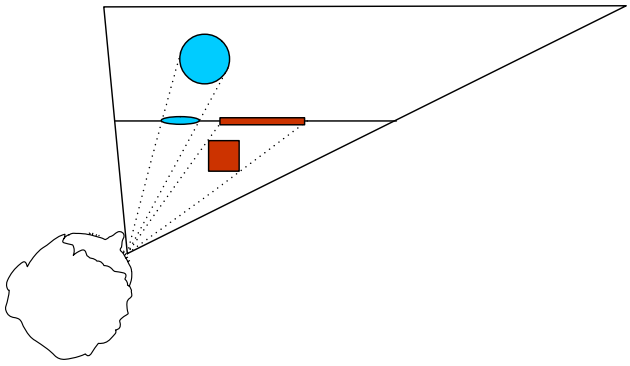


Figure 29: Top-view of viewer aligned projection. The projection of the virtual scene onto the virtual screen using a viewer aligned virtual eye position produces an image that seen from the viewer is identical to what would have been seen if the scene was real.

It should be noted that this is not the same result as if the projection had been to a plane perpendicular to the view-direction (e.g. if a virtual camera had been rotated to the viewers of-axis position) as seen in Figure 30, which in comparison would cause the virtual objects to change shape, size and place.

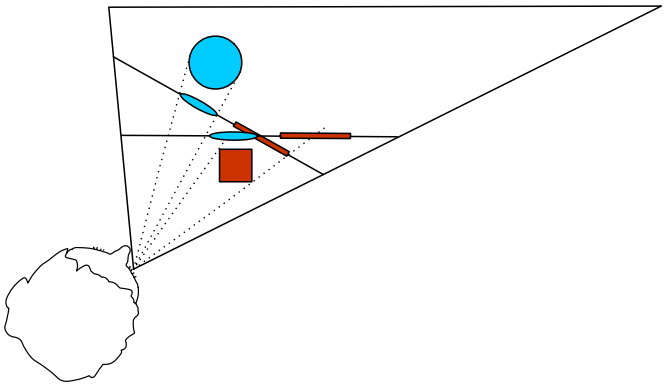


Figure 30: Top-view of traditional camera projection. A projection onto a plane perpendicular to the view direction is shown, and the projected image rotated to the plane of the screen. Note that it is not the same result as if projection directly to the plane of the screen. It is seen that objects changes in size and placement.

Frustum Volume

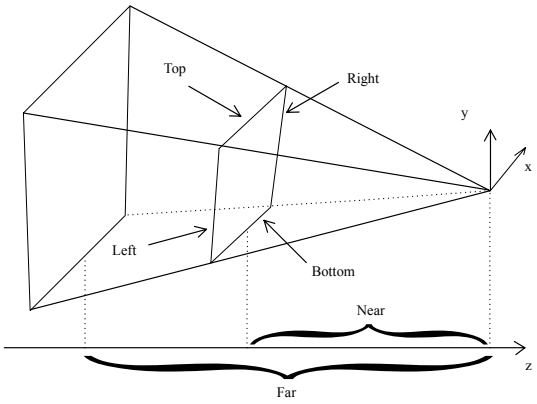


Figure 31: Frustum Volume

Most modern graphic cards use a frustum volume to determine what part of a scene is visible and should be rendered. Usually it is defined by a near and a far clipping plane and a x-y-axis aligned rectangle on the near clipping plane. The volume is the projective extrusion of the rectangle from the near plane to the far plane.
If using near and far clipping planes parallel to the virtual screen plane, the frustum is ideal for setting up viewer aligned projection. See appendix 10.1 for a formal definition of the associated projection matrix [2].

Setting up Viewer Alignment

If given the eye position in the virtual screen coordinate system, the setup can be done in a few simple steps:

1. Scale the rectangle of the virtual screen to the near clipping plane using the eye position as scaling center.
2. Transform to the coordinate system of the virtual screen.
3. Translate to the eye position
4. Render the scene.

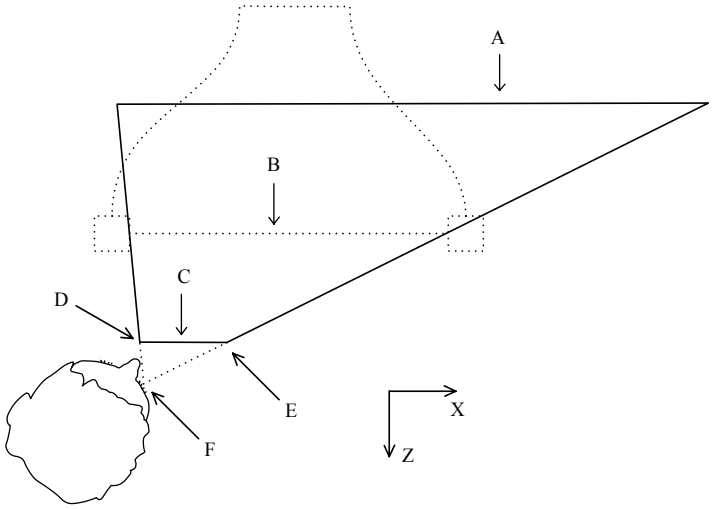


Figure 32: Top-view of viewer aligned frustum. The far cliping-plane, A, the virtual screen, B, which when scaled (with the virtual eye-point, F, as center) to the near clipping-plane, C, defines the corners, D and E.

Window in Screen

It should be noted, that if the rendered image is to be shown in a window on the screen, the corresponding positions of the window corners in the virtual world should be used instead of the screens corners. Consequently it is necessary to be able to tell were on the screen the window is placed.

Stereo

For stereo rendering or multi-viewer rendering the procedure is repeated for each eye.

Depth impression break down

Each image of stereo rendering obviously has different frustum volumes. For objects behind the image plane there will be a natural occlusion when leaving the volume. The object is visible for one eye a little longer than for the other as if looking through a window (see Figure 33).
Unfortunately when an object in front of the image plane leaves the frustum volume it also disappears which has no equivalent in the real world. Because the object is still visible for one eye longer than the other it also

seems like an occlusion behind the frame of the image plane (e.g. the monitor frame). This causes contradictions in the depth cues and as mentioned in section 1.3 occlusion is a very strong depth cue that overrules the stereo effect and moving parallax and therefore the depth impression breaks down and the objects is thought to be behind the image plane (The same effect is seen if you move your hand in front of a still stereo image behind the virtual object). This effect can not be avoided completely, but can be reduced, by removing the object earlier.

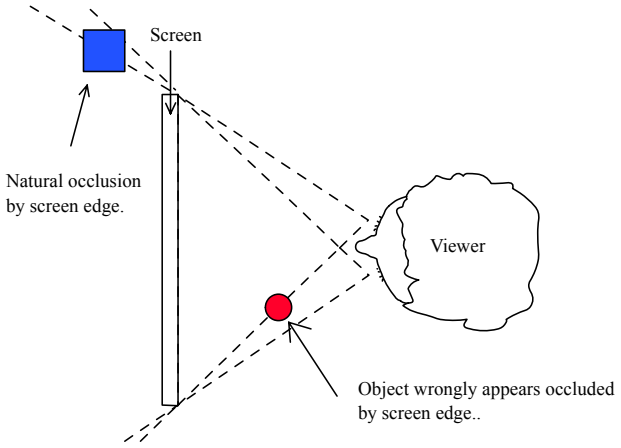


Figure 33: The frustum volumes of the stereo view causes natural occlusion behind the image plane, but also fatal unnatural occlusion in front of the image plane.

Inner Frustum

An inner frustum is produced by adding clipping planes, so that the frustum volumes in front of the image plane become identical in both views. Now objects disappear before hitting the frame of the image plane, one eye might even see the inside of the object as it is clipped.

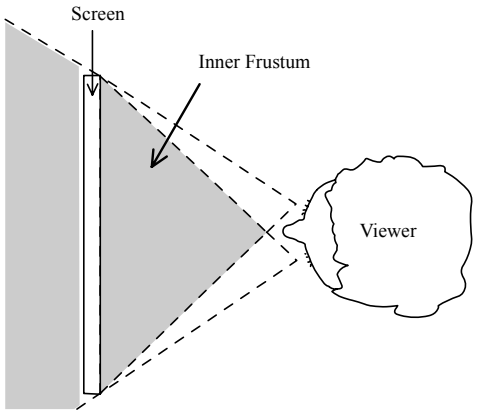


Figure 34: The inner frustum of the stereo view.

Using the inner frustum is not perfect but better than not to. To improve further one could make the frustum volume even smaller (e.g. converging to two thirds of the distance to the viewer), so it is clear that the objects

disappear rather than getting occluded, although this also seems strange compared to the real world.

4.3.2 Hierarchical Structure of Monitor, Camera and Eye

It is clear that viewer alignment requires that the eye position is known. It is also clear that it is the position relative to the monitor that is important. The absolute position of the system is irrelevant, although the individual absolute positions can be used to derive the relative position.

Move-ability

When working with 3D-data it is important to be able to move relative to the virtual world. To do this we have to find a way to move the eye, monitor and tool positions at the same time, so the relation between them is preserved. This is easily done by making a hierarchical system.

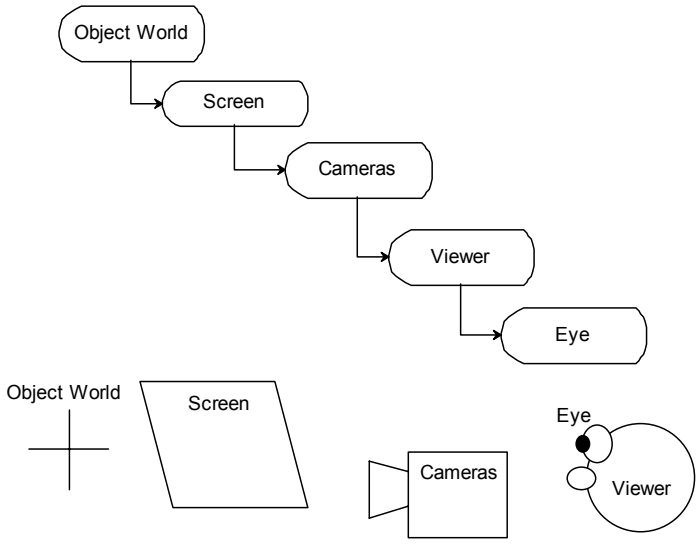


Figure 35: Hierarchical system of real world's positions in virtual setup. With the screen as base it is free to be moved around without other relative positions deeper within the tree change.

Screen as base object

Although the cameras are the base for all our position observations in the real world it is simpler to let it be a child to the screen in the virtual world. In that it is presumed, that the screen or monitor in the real world is stationary, and their positions does not have to be updated after calibration. Having this setup with the virtual screen as base in the virtual world the screen can be moved freely around with all its object children preserving their relative positions.

The Eye Position

The eye position is to be determined. As mentioned in previous sections the eye might not be tracked directly. It could be the users' head that is tracked, and the eye positions are calculated by adding its position relative to the

head. To freely setup any relative eye positions an individual viewer object is needed. If the eyes are tracked directly the viewer object can be left out.

The Cameras

Finally the camera's coordinate system in the real worlds relative to the monitor has to be determined. This can be done in several ways e.g. measuring by hand or as a part of the camera calibration. The relative position and orientation between the cameras and the monitor can be incorporated in the camera projection matrixes (see equation (6)), so that the positions found would be described directly in the screens coordinate system, so the camera object in the hierarchical tree can be left out.

4.3.3 Drawing and Point-Transformation in a Hierarchical Scene

Push-pop

Drawing objects in a hierarchical system is easy on modern graphic cards, because of the push-pop matrix system. A child object is simply drawn by first applying its own transformation to the transformation to the transformation matrix produced by its parent. After drawing, the changes can be removed by popping the changed matrix.
Transformation of points between objects coordinate system does not come automatically. The problem is that the object does not know where in the tree it is placed relative to another.

Arbitrary Transformations

It is necessary to transform a point from one coordinate system to another in the hierarchical tree that can be changed at any time. A simple way to do this is making a recursive function that be called on any object in the tree. The function calls its parent object until the root of the tree is reached. Transforming from one object to another can then be done by transforming to the root coordinate system and then back into the wished objects coordinate system. Although some unnecessary calculations will be made converting between objects in the same branch of the tree, the function is general and works in all cases. To transform to the root the transformation should be applied before calling the parent object. To transform back to the object the transformation should be called after the call to the parent object.

4.3.4 Stereo

To fulfill the requirement of stereo the simple anaglyph method is chosen (outlined in section 3.1.9).
Many alternatives stereo devices have been developed through time. Here we will mention a few real-time alternative, and compare their advantages and disadvantages.

Method name:	Technique:	Main advantage:	Main disadvantage:
Anaglyph stereo	color filters, color encoding	easy, cheap	no color, 1/3 light intensity, or crosstalk in the common collor
Color Code	color filters, advanced color encoding	easy, cheap for user, full color	patented, minor crosstalk
Shutter glasses	liquid crystal shuts each eye subsequently. Monitor frequency locked, show alternating left and right image	full color, relatively cheap	halves the refresh rate of monitor, require special hardware, minor crosstalk
Polarization glasses	polarize light from left and right image differently, polarized glasses	full color	Special screen, minor crosstalk
Auto stereoscopic display	lenticular, barrier, directional interleaved color filter	glasses free, full color	* very expensive

Table 3: Different stereo producing techniques and their main advantages and disadvantages. * 2004.

As seen in Table 3 the anaglyph method is one of the easiest and most affordable, but is restricted to monochromatic stereo imaging and only a third of the intensity of a non stereo image is achieved (because only one of three color channels passes through the filters). It is chosen because of ease and prize, but any other method could in principle have been chosen. The system of this thesis should work with any stereo method where the user is free to move the head (some methods though require fixed head position and can not be used).

4.3.5 Line Stroke

A line stroke is described by a row of 3D-points. Between each point a line is drawn.

To improve this, a mesh forming a tube can be drawn instead to add thickness to the line. By storing the lines as a midpoint, orientation and a length, the stroke can be drawn using simple transformation and scaling of a predefined tube that can be stored in a display-list.
To make the connection at the joints look nice (i.e. no overlapping parts), it is possible to store two vectors describing a plane at each point in which the points of the tube cross-section should be drawn. A custom build cross-section can be defined before drawing and its coordinates only have to be transformed by multiplying its 2D coordinates with the vectors defining the plane. The two vectors orientation at the point can be determined using the second derivative of the line at the point, which always is a normal to the line pointing towards the line's center of rotation at the point. For a straight line it is zero and a specific value has to be chosen (e.g. the same as the first point on the straight part).

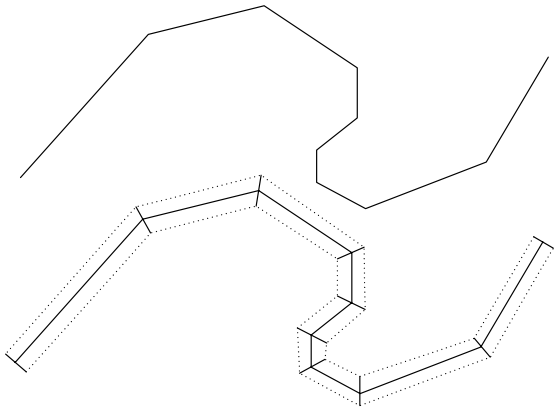


Figure 36: Simple and thick line.

4.3.6 Texture volume

Using a texture volume method as outlined in section 3.1.10, a volume representation can be achieved. Due to the bi-linear filtering of texture lookups implemented on most hardware, the edges of a texture volume are round and smooth. It is able to visualize transparent materials, and although no shading is applied, will the low values near the edge of a visible volume sum up to be dark, giving the volume a fuzzy sort of shaded look.

Keep it Simple

There has been much development in texture-volume techniques enabling very advanced visualizations in real-time. In this project we only need a simple method to demonstrate the concept. Therefore a simple visualization that is relatively easy to implement is chosen.
Although simple, it is possible to visualize the inverted volume or a semi-iso-surface (as seen in Figure 37), by changing the drawing mode for alpha threshold and blending (e.g. using the functions; glAphaFunc() and glBlendFunc() in OpenGL).

(a) (b)

(c)

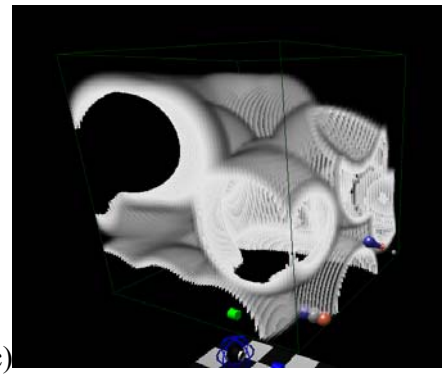


Figure 37: Three texture volume visualizations using three different drawing modes. In (a) alpha-blending and alpha-testing is used. In (b) there is no blending, and alpha-testing is inverted to accept values below the alpha threshold. In (c) a combination of (a) and (b) is seen, where alpha blending is used with the inverted alpha-test, so only a shell or semi iso-surface is seen.

Texture volumes are well suited to be edited using the enabled 3D interaction, because of its data format. Voxels near the curser can easily be accessed and changed.

The properties of the editing task can be thought of much like painting in 2D but now expanded to 3D. A voxel representation is the equivalent to pixel representation in 2D images. Most 2D image editing operations can be expanded to 3D, providing vast amount of possibly manipulation tools. Furthermore, a range of specific methods for editing voxels exists.

Chapter 5 Implementation

In this chapter the basics of an implementation of the system will be outlined starting with the external libraries, mentioning threads, dataflow and structure and finally looking at the hardware.

5.1 External Libraries

The implementation is written in the programming languish C++ using Visual Studio v.6.0 and is build upon following libraries and classes:

- **MFC:** Microsoft Foundation Classes
- **Vision SDK:** Microsoft Vision Software Development Kit
- **OpenGL:** Open Graphics Library by Silicon Graphics
- **LinAlg:** Linear algebra interface to LAPACK by Henrik Aanæs

Their main functions are to provide user interface, handling of image capturing, hardware accelerated 3D visualization and fast linear algebra computation.

5.1.1 MFC

MFC provides a programming frame work with standard interface functionalities such as: Menus, toolbars, dialogs and serialization. Furthermore it provides a tactic for separation of data classes and user interface classes, which makes the program more flexible.

Strategy

Using MFC it is typical to structure you program into a document containing data and views (including dialogs) to present data to the user and interact with the user. This strategy was followed in the implementation.

Not portable

MFC is not portable to non-windows platforms.

5.1.2 Vision SDK

The Vision SDK provides the functions to get contact to the image recording capable resources on the computer which in this case is the web cameras. Furthermore it provides the image formats used by the tracking algorithms in the implementation.

Camera Handler

In the implementation one class (CCameraHandler) is made to take care of all function calls of vision SDK regarding the connection to web-cameras.

Threads

When an image-grabbing process is initialized a separate thread is started by the Vision SDK library to perform the image grabbing continuously independent of the status of the rest of the program. This way there is always access to the newest image provided by the image source delay.

Alternative

As alternative for instance DirectX could be used in image grabbing as well as for the hardware accelerated 3D visualization. Probably it would make a better interface for the image grabbing, because it is better at handling and initializing more cameras at once. Using Vision SDK each camera after the first has to be selected manually making automatic initializing at program startup very difficult.

5.1.3 OpenGL

OpenGL is a well tested graphics library used in a vide range of 3D visualization programs. Most OpenGL calls that draws graphics are restricted to the classes “CScene”, “CSceneObjects”, and a class called “PersGF” only containing static functions for drawing 3D objects using OpenGL. The view-classes use OpenGL calls mainly to setup the content device and frustum of the view.

5.1.4 LinAlg

The LinAlg packet, written by Henrik Aanæs [25], supports vectors and matrices of all standard types via templates. It has special implementation of vectors of low dimensions (two and three), with better performance than using the general functions. It does not contain an exhaustive list of matrix and vector operations, but produce an interface to the fast LAPACK (Linear Algebra PACKage) by linking to the precompiled library “clapack.lib”.

Most important is the function “SVD” (Singular value decomposition), which is used to decompose the matrix in linear least squares adjustment described in section 3.1.6.

5.2 Threads and Dataflow

The implementation is multithreaded as mentioned in section 5.1.2. The main thread takes care of object-tracking, interaction (and simulation), and visualization. The secondary threads are run by vision SDK and take care of grabbing images from the webcams (see Figure 38).

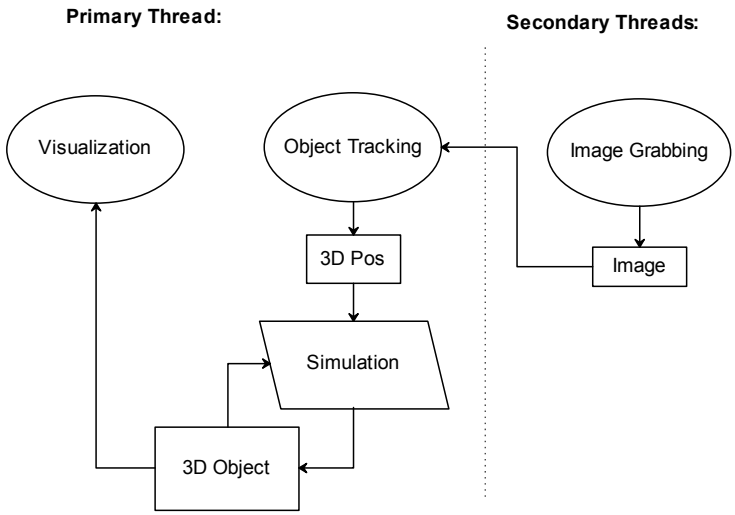


Figure 38: Outlining of threads in the implementation.

The rate of which the objects are tracked compared to the visualization can be varied but is set to every second frame as default.
No predictor, but a very simple smoothing algorithm is implemented in the simulation step (averaging the old position with the last tracked, evenly weighted). An advanced predictor (e.g. kallman filter) could be implemented to take its place without any further adjustments.

Alternative

An alternative threading strategy is to include the total 3D tracking in secondary threads, so that the simulation and the visualization could run smoothly independent of the rate of 3D-tracking.

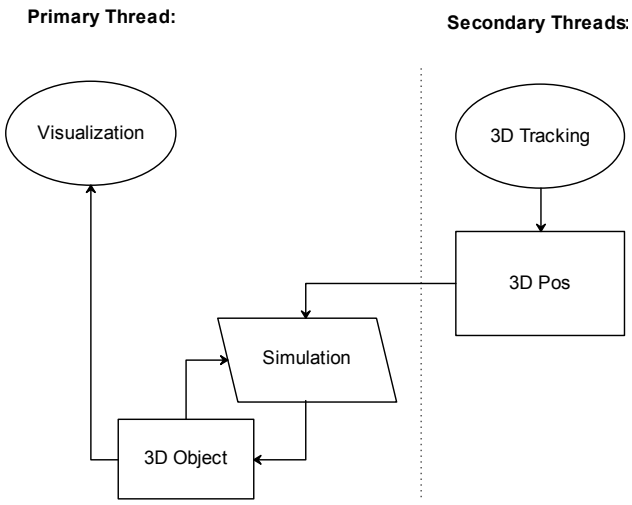


Figure 39: Alternative threading.

Dataflow

The entire dataflow of the implementation is outlined in Figure 40.

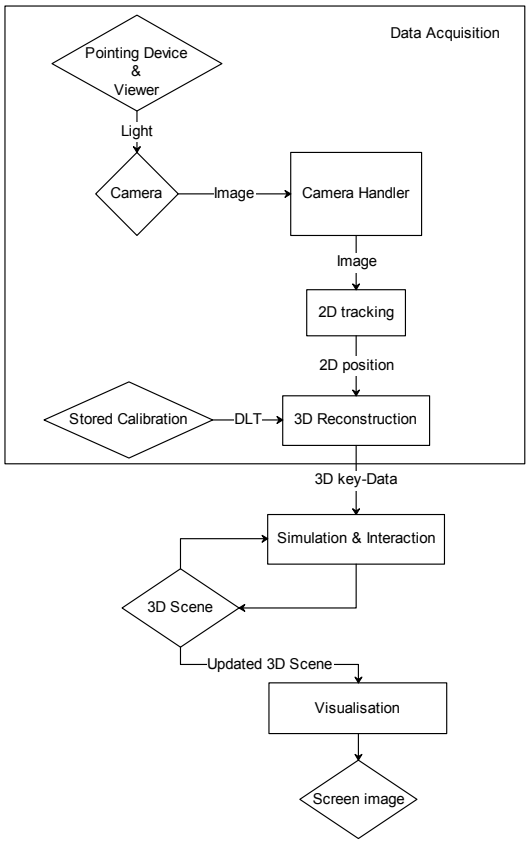


Figure 40: Dataflow.

Each segments functionality is in most cases spread over several classes, which will be outlined in following section.

5.3 Program structure – Classes

5.3.1 Overview

This overview is a listing of the classes categorized by purpose.

The Framework

The main classes of the implementation classes (inherit from MFC) are: CMainFrame, CSimple3DApp and CSimple3DDoc and the view classes COpenGLView and CAnaglyphView.

Image Aquisition

For image acquisition we have the essential classes CCameraHandler, CameraData and CCameraOrientation.

Tracking

The class CTracker is the main class of the object tracking providing functionality, whereas classes like CColor, CTraked2DSet, CTracked3DData and CMonitorCalibration contains the needed data.

3D Scene

The class CSceneObject is the root of all scene classes. The 3D Scene is build upon the classes CScene and CGLObject (see Figure 42). Additionally the classes CBall, CGLCheckerboard, CLineobject, CTestOriObject, CTextureObject, CViewTransformerObject and CVolumeTexture specifies the individuel object types.

Furthermore the class PersGF provide a few static drawing functions and the class CGLColor contains color suited for openGL format.

Orientation and Transformation

The class COrientation contains data of a spatial objects orientation, mass, speed and scale. Furthermore it provides functionality of 3D-point transformations and transformation matrix generation.

Dialogs

The modal and modeless dialogs based on the MFC dialog of the system are based on the MFC dialog implemented in the dialog classes CAboutDlg, CCameraModelessDialog, CModelessTreeDialog and CTrackObjPropDialog.

5.3.2 Inheritance

In the implementation, code is reused and through inheritance. The trees of inheritance are seen in Figure 41, Figure 42, Figure 43 and Figure 44.

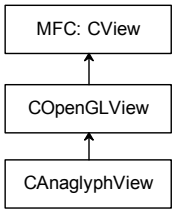


Figure 41: The view classes all inherit from the MFC class CView.

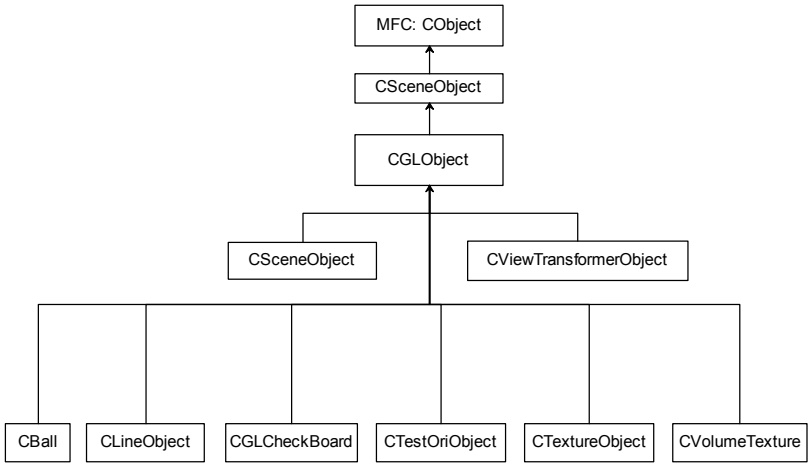


Figure 42: All Scene objects inherit from CSceneObject and CGLObject, which provide the hierarchical structure for drawing and transformation.

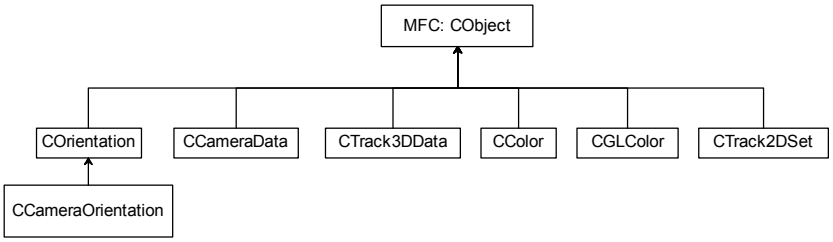


Figure 43: Objects that are stored or serialized at some point inherits from the MFC Class: CObject.

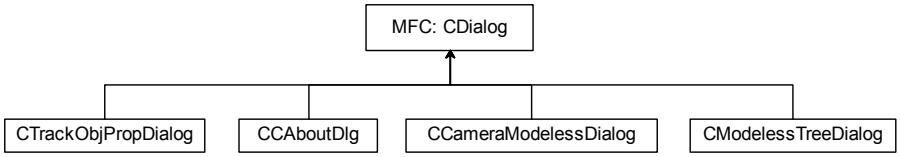


Figure 44: All Dialogs inherit from the MFC class CDialog.

The class CObject is base for many classes because of the serialization methods of MFC.

5.4 Hardware

The software developed is meant to be general. However it is out of the scope of this thesis to test it on a range of hardware. Therefore it has so far only been run on a few PC's with one kind of web camera.

5.4.1 System

The implementation was done using:

- **PC**: a Pentium III, 700 MHz
- **Graphics hardware**: G-Force 4
- **Web cameras**: Philips ToUCam Pro (PCVC740K) with a CCD sensor, capable of 60 fps at 320x240. Maximum video resolution: 640x480 (30fps)
- **Connection**: USB 1.0

The web cameras used are unusual fast though cheap capable of 60 fps compared to the normal 20-30 fps. Unfortunately only up to 30 fps are performed by Vision SDK, which implies a benefit of changing to another image grabber interface (e.g. DirectX as mentioned in section 5.1.2). The speed of a single USB 1 controller in the present system is not sufficient to transfer images from two cameras at full frame rate unless the resolution is reduced to 160x120. Systems with more than one USB controller (which are standard), can run with at least two cameras at higher resolution (e.g. 640x480).

5.4.2 Tracking-Objects

As tracking objects colored spheres are used as seen in Figure 45. They are made by painting cotton spheres with a felt-tip pen and finally put on wooden sticks. The production took 5 minutes and is very low in price. Only the most saturated colors are used.

Sphere Diameter: 15mm
Stick length: 240mm

Figure 45: Hand made tracking-objects.

5.4.3 Camera Setup

The cameras have been setup similar to the proposed setup in section 4.2.8. One camera was placed to the left of the monitor and one straight above the monitor in a distance of approximately one meter creating a measuring volume in front of the screen at approximately 50cm x 50cm x 50cm stopping few centimeters from the screen.

5.5 State of Implementation

The implementation in the scope of this thesis is not meant as a ready-for-use library, but rather a state of research program for demonstrational use, with many remaining performance and structural improvements to be done. However, the possibility of a transformation into useful libraries in the future was not ignored during implementation.

Performance and quality of the system depends on both the software and the hardware parts of the implementation. This has to be considered when evaluating, which we will look into in the following chapter.

Chapter 6 Results

The proposed concept has been implemented. Simple tests have been made to give a hint of the speed and the precision of the system. To get a precise knowledge of the speed and precision of the algorithms and system more thorough tests have to be made. In this chapter the results are presented. The results and the system will be discussed in the following chapter.

6.1.1 Image Grabbing

The speed test was done by registering the rates of the main loop of the system, while enabling and disabling different processes.

The image grabbing part of the program use vision SDK and runs in separate threads for each camera. The loop rate was measured for the system with most processes disabled (e.g. visualization). Starting the camera threads causes the loop rate of the main process to drop, which indicates the workload of the image grabbing threads.

The loop rate drops 27% from 2090 Hz to 1540 Hz on the 700 MHz processor for two live 160x120 pixel images. The first thread grabs 30 fps the second a little lower; 26 fps. Taking the workload into account, the total processor time to grab an image is about 4.7 ms.

6.1.2 Color Tracking, 3D-Reconstruction and Calibration

Color Tracking

The quality and the speed of the color tracker are dependent on many factors and should therefore be treated special. The tracker can track a color, by setting up the hue, the width and the threshold. These values depend on lighting conditions, the background colors and the color of the object to be tracked. In bad conditions it is not possible to track the color. The cameras used in the setup can automatically adjust for the shift in lighting from indoor to outdoor, including adjusting the shutter speed. However indoor lighting tends to flicker at 50 or 60 Hz. This is clearly seen on images taken at high shutter speeds causing higher requirements to the saturation contrast between the tracked object and the background.

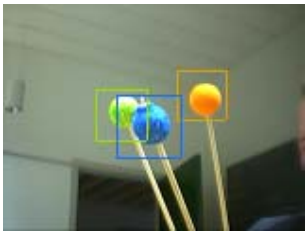


Figure 46: Color tracking of three spheres in close up.

Number of Colors

Through empirical tests it seems that three colors can be reliably separated and tracked simultaneously. By adding more colors the colors are easily mixed up or hard to separate from background colors. In perfect light conditions or at higher resolutions the number is expected to be higher.

Tracking Speed

Color tracking	processor time (ms)
3 objects in two images found	12-14
3 objects in two images not found	25-27

Table 4: Processor time for implemented color tracking on a 700 MHz Pentium III. Image size is 160x120. Object size in picture < 8x8 pixel.

The total processing time for tracking 3 objects in two images is 12 - 14 ms. This was measured when all three objects were found and did not move fast. When the objects are not found or have to be found again due to e.g. occlusion the whole image is scanned at first and the processing time increase to about the double; 25-27 ms. For larger image size the factor will be larger, due to the larger factor between the start search window and the whole window.

3D reconstruction

The timing for the reconstruction is quite small because only few points have to be reconstructed per frame, and only when sufficient 2D positions are found. Reconstruction of the three points takes 0.3-0.5 ms.
The quality of the reconstructed 3D-points depends on precision in the 2D color tracking and the camera calibration, which also depends on the tracking precision (because the color tracking is used in the calibration). In the reconstruction the remaining error can be calculated. The error E is defined by:

$$E = \sqrt{\frac{\sum_n e_n^2}{n}}$$

, where n is the total number of 2D position coordinates for all images used and e is the error in the coordinate between the pixel found by the 2D color tracker and the pixel found by projecting the estimated 3D-point.

Precision hint

One test on one calibration in two live 160x120 images has been performed, to give a hint about the precision. The remaining error associated with a reconstructed 3D-point was typically 0.0-3.0 pixels when the real points were found. Occasionally the error exceeded 20.0 pixels, which implies that two different objects were tracked as the same object in the two images.

Calibration

When calibrating, an error similar to that in 3D reconstruction can be calculated, to estimate the precision of the calibration. Unfortunately this has not been implemented, but the above error hint also gives a hint of the calibration error, because of the dependencies.

3D precision

No precise values of the systems 3D precision can be given due to lack of tests. However, a rough estimate of the current setup is a relative precision better than 1.5 cm and a 3D resolution around 0.5 cm

6.1.3 Visualization and interaction timing

Scene

The frame rate of the system is important for interaction with the virtual world. The timing for non stereo visualization depends heavily on what is drawn in the scene. The scene tested was rather simple, visualizing the cursor, a checkerboard, few registration points shown as spheres and a texture volume with a resolution of 64x64x64. The volume reach the fill rate limit of the graphics hardware, which is seen as a variation in frame rate depending on the screen area occupied by the volume (being inside it causing the lowest frame rates).

Interaction

Interaction involves calculations (e.g. transformation), that has not been optimized for speed. This causes the frame rates to drop when rotating, moving or editing an object. Editing large numbers of voxels in the texture volume simultaneously also has a high computational cost.

Stereo

Stereo is implemented, using the anaglyph technique. Due to multiple whole screen renderings (while blending), the stereo mode is much slower on this system than without stereo.

Processor time for the individual processes is seen in Table 5.

Visualization or action type:	**Processor time (ms):**
Non stereo visualization	11-22
Stereo visualization	30-35
Moving object	4-5
Rotating object	4.5-5.5
Editing object	5-15

Table 5: Processor time for visualization and interaction with a 3D scene.

These timings plus the rest of the system adds up and gives the frame rates measured on the fly as seen in Table 6.

Operation:	**Frame rate (fps):**
Passive stereo	25-32
Passive	40-100
Passive head tracking with stereo	12-15
Passive head tracking	32-37
Orientation tracking (object rotation)	26-29
Tracked interaction (volume editing)	29-31
Volume editing with large curser (radius = 15 voxels)	10-20

Table 6: Run time frame rates during different operations of the implementation.

Viewer alignment and interaction will be discussed in the following chapter.

Chapter 7 Discussion

In this chapter we will discuss the results of the thesis in relation to the concept setup in the hypothesis. Furthermore we will discuss what could be done in future work, and which alternatives could improve single parts or the concept as a whole.

7.1 The Individual Parts

7.1.1 Data Acquisition

Image Grabbing

The image grabbing is functioning as planned fulfilling its purpose, although the resolution is low (160x120). This is not the first limiting factor of the project in the used setup, but with faster processing hardware or speed optimizations on other parts of the system it could be a limiting factor.

To get image resolution and speed up several factors have to be considered. First of all the transfer rate have to be higher. That means that more than one USB bus, or a faster transfer method, has to be used. Higher speed first of all requires cameras capable of higher frame rates than normal web-cameras. The web-cameras used in the system are capable of higher frame rates (60 fps), but attempts to grab more then 30 fps using Vision SDK has failed, even for a single camera. Hence, higher image speed requires another grabbing interface. E.g. a DirectX grabber implementations has shown to run 60 fps with one of the system cameras, and seems to be an obvious alternative for future improvements of the system.

Color Tracking

To prove the concept a fast and simple tracking algorithm was needed. The color tracker chosen has proved to fulfill these requirements. It is able to track 3 colors in 2 images at real-time frame rates providing sufficient precision to test the concept, but with too high latency to generate seamless viewer alignment. Its speed could be improved by optimizing the per-pixel RGB to probability conversion e.g. by using lookup tables in one or more steps.

Although suited for testing the current color tracker seems much too unstable to use in an every day tool. The high dependency of lighting and background colors makes it hard to setup and can cause it to loose track of the objects during operation.

The color tracker in combination with calibration also seems to be the main error source in the 3D point registration.

Improving color tracking

A way of improving the robustness of the color tracker that would not involve large changes to the basic idea is to introduce a variable threshold. By analyzing the rate of changes in the color probability within the search window a better threshold might be found. This way the algorithm might be less sensitive to light conditions, where the contrast between the background and tracked object is very small. It has to be considered how to determine the size of the window since the sum of accepted pixels will vary.

As mentioned in the analysis in Chapter 4 an important step to improve the color tracker would be, to be able to separate different tracking candidates and identify the most likely by other means than color (e.g. shape or placement relative to other objects). To separate the candidates a first step could be to use connected component analysis directly on the image or on a down sampled version. The down sampling would remove noise, but also blend the color of an object with the background color at the edges. The blending has a crucial effect if the tracked object is very small (few pixels in image) and should be considered in a choice of implementation.

Alternative to color tracking

To track an object can be done other ways than color tracking. As an alternative feature tracking can be implemented searching for a given 2D or 3D shape. 2D shapes could be circles (suited for the spheres), ellipses, squares or other simple shapes that can be recognized fast and stable. A simple box could be the 3D shape recognition object, making determination of the orientation of the object possible [30]. A combination with the color tracker could possibly add speed or stability if e.g. a cube were used with different colors on each side a start guess of the cubes orientation could be determined by color tracking. A more precise orientation could be found using other algorithms.

Only one camera

The full orientation of a known 3D object can be found from a single 2D image [30]. By incorporating this into the system in future work would mean that only one camera is needed. If this combination could work at sufficient speed the costs and complexity of the system and its setup would be improved significantly.

3D-Reconstruction

As mentioned above there are alternatives to find the orientation using multi image 3D reconstruction. However in this implementation the 3D reconstruction we used is well suited, because it is very fast and estimates 3D positions using 2D positions from multiple images (which is what we have). If very precise 2D positions were available the 3D-reconstruction would also be more precise.
If the color tracker had sub-pixel precision the 2D position could be made more precise by including camera parameters like lens distortions in the model. This would furthermore require a more thorough camera calibration.

Camera Calibration

The camera calibration proposed and used in the system includes calibration of the monitor. This is done by using the monitor as a base calibration object, which insures that the relation between the cameras and the monitor is determined precisely. Only a linear model has been implemented which means that the precision of the calibration is dependent on the users ability to place the tracked pointing device in right angels to the screen of the monitor.

The calibration could be improved by implementing a nonlinear calibration model, which is independent of known coordinates. It could easily have a much higher number of calibration points providing higher precision by statistically canceling out errors in the 2D registration. Furthermore it would open up for the possibility of calibrating the cameras inner orientations (lens distortion parameters) as well.

Viewer alignment calibration

In the implementation, the viewer is tracked by tracking an object with a position known relative to the viewer's eye-position. This "known" position is actually not known but have to be found through a viewer alignment calibration.

No viewer alignment calibration is implemented. Instead the coordinates of the viewer's eye-positions relative to the tracked viewer object have to be measured by hand. This has simplified the implementation of the system, but is not a very precise method. An alternative method using stereo points has been proposed, but is not implemented or tested.

Eye tracking

The viewer calibration could be avoided by tracking the eyes of the viewer directly. Eye tracking algorithms exist in literature and applications, but none has been implemented in this system. If such an algorithm was implemented it would most likely be dependent of the viewer-camera angle, which would give an extra restriction to the camera placement. The used method, where the viewer position is found, could be used as a first guess in an eye-tracking algorithm to limit its search area, number of candidates and processor time. It is most likely that an eye-tracking algorithm would take up more computer resources than the implemented viewer tracker, and therefore it would only be suited for systems with more computer power, than the test system used here.

7.1.2 Visualization

Stereo

The real-time stereo rendering is important to perceive the depth when working with the 3D-data. It enables a higher level of coordination between the tool and the 3D-data in the viewer aligned setup than what can be

obtained without stereo. Thus it is a significant loss using mono mode, though not crucial because all operations are still possible.
Contrast, clarity and color resolution is still at a much higher level in mono mode and is crucial for other depth cues, so these factors should be considered as well as the speed, when choosing mono or stereo mode.

Faster stereo

Since the stereo rendering requires two images of the scene the frame rate is not expected to exceed half of the non-stereo frame rate. The speed of the stereo rendering can increased by several means. Naturally more computer power would speed up the rendering (e.g. faster graphics hardware), but also adjustments to the method can help. In the current implementation, the scene is rendered to the frame buffer and copied to a texture. A faster method would be to use an OpenGL extension enabling pixel-buffers, so that the scene can be rendered directly to the textures (or buffers). Also rendering the textures in a single pass using an extension enabling multi-texture rendering, would speed up the stereo rendering.

Viewer Alignment

Viewer alignment is implemented in the system, but its precision has not been tested. A test suited could be done in stereo mode by letting a test person point out stereo points without seeing the cursor on the screen. This way the difference in the perceived 3D position and the virtual 3D position is estimated.

Although no precision test has been carried out the viewer alignment can be subjectively estimated. The basic principle works. When the head is moved objects behind the screen moves and gets occluded by the frame of the screen like it would if it were real. The viewer can move around objects in front of the screen and inspect them from different views e.g. look through holes on the side or virtually stick the head inside the object and observe it from the inside.
By holding a tracked 3D tool in front of the head the tool can be seen on the screen approximately behind the real world tool. A virtual extension of the tool has been made, so that one can see the tool cursor without occluding the view.

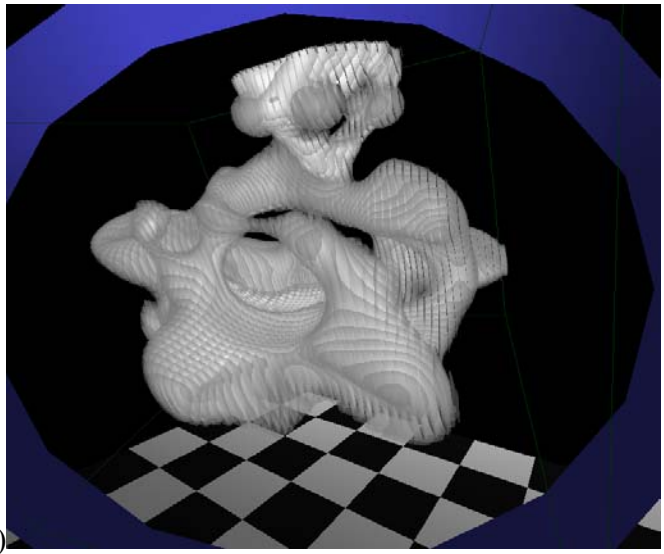
7.1.3 Interaction

Objects can be moved, orientated and edited with the 3D tool. Special objects have been implemented to demonstrate this.

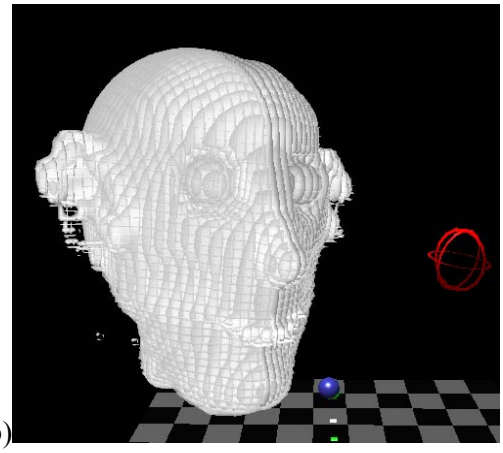
Modeling

A simple line object enables drawing making it possible to make coarse sketches in 3D.
A texture volume can be edited with the tool, adding or removing material. Especially organic shapes seem to be unusually fast and easy to sketch in few minutes like the heads seen in Figure 47.

(a)



(b)



(c)

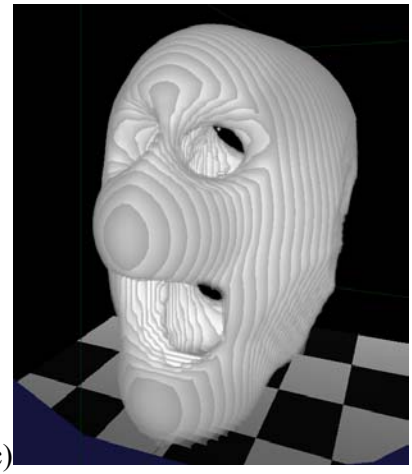


Figure 47: (a) Abstract organic cave-like shape and (b-c) sketches of a head and a mask, created in a few minutes during test of the implemented 3D tool on a texture volume with mirror.

Medical inspection

Volumetric medical data obtained through CT and MR scanning was inspected using the system. The data could be moved and rotated, and by moving the head inside the data, inner structures and details were revealed at close up. No preprocessing has been implemented.

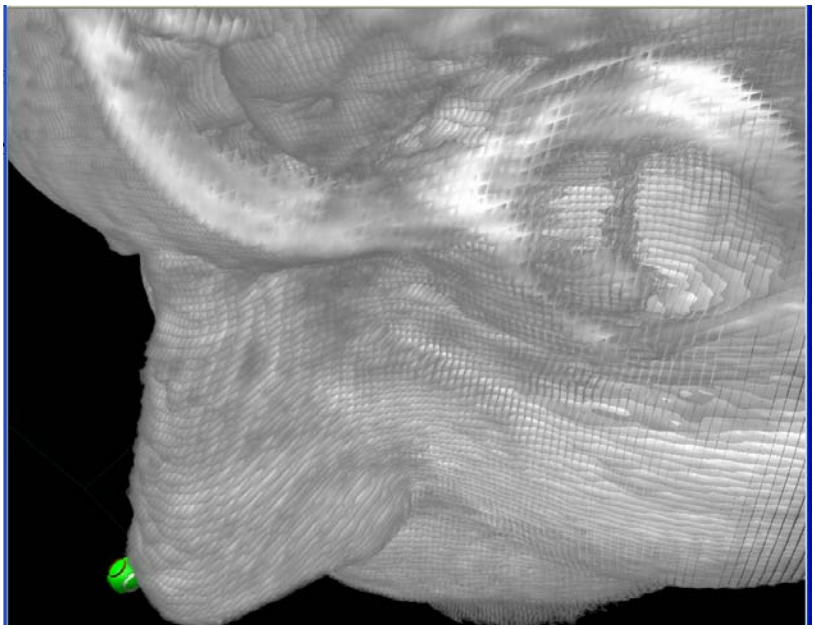


Figure 48: Close up on a MR-head-scan. The nose is seen from the outside in the bottom left corner. The near clipping plane allows us to look under the scull revealing the curly brain of the forehead (top-middle) and a cross section of the eye (top-right).

7.2 Future work

In this section we will present and discuss further improvements of the system that has yet not been implemented. Additionally improvements of more philosophical origin will be presented.

Object Picking

In common object based drawing programs it is standard to select an object by clicking on it with the mouse. No such selection method has been implemented, but it could easily be done in similar ways of the 2D mouse selection.
The user places the pointing device in the scene, and pushes the select button, causing the scene to be drawn at a resolution of a single pixel in index mode using a small frustum surrounding the cursor. The index of the drawn object are read back from the graphics hardware, and selected.
Similar many standard operations from 2D input drawing programs could be converted to be used in the 3D input system.

Tool Stability Improvements

Kalman filters are used in many applications also in object tracking to smooth or predict object's position at a given time. Such a filter can predict positions based on the old position of tracked objects, giving a good prediction of where the object is, when the next frame is shown on the screen. It obviously works best at low accelerations, because of overshooting at high accelerations. It also has the ability to smooth out jitter.
The implementation of such a filter in the system would provide obvious benefits to the system compensating for low update rates and latency.

Accessibility Improvements

If the system should be of any use beside inspiration, a range of practical improvements have to be done addressing accessibility.
As a standalone system it should be able to load and save files, in one or more standard formats (currently it is only possible to load and save texture volumes in a simple raw format).

If the concept should be more general providing a 3D interface for a range of applications, it would need a driver. For a simple 3D input standard driver, formats like mouse and joystick drivers, might be sufficient.
To provide functionality like viewer alignment or stereo, special software or plug-ins for the given application would most likely be necessary.

Flexibility Improvements

At its current state the implementation has a fixed configuration of the tracked viewer objects and the pointing device. Only calibration and colors tracked can be adjusted and saved.

To make the system useful it would be necessary to make this setup completely customizable.

The number of cameras can be chosen freely but in the current implementation only the positions found in the first two will be used in the 3D reconstruction. The system can function without cameras, but some functionality (and the whole point of the concept) will be missing.
It should be possible to custom choose the number of cameras. This extension of the implementation would be trivial due to the similarity to the current implementation.

Interface Improvements

To improve the 3D interactivity it could be considered to use virtual 3D buttons as interface on the screen. The buttons should be placed virtually outside the screen, so they were accessible with the pointing device. Menu systems like in 2D could be adapted.

Tool Improvements

Finally the interaction tools of the implementation could be improved in different ways. Custom shaped interaction shapes would be interesting. E.g. in the editing of the texture volume, shape of the cursor could be varied to form a box, a line, a plane, a scalpel or any desired form for a specific purpose.

Chapter 8 Conclusion

The goal of the thesis has been reached. The work has involved a number of parts that have to function individually and in combination with each other. All of these parts have been put together in the implementation.

The proposed concept is implemented on a system using two inexpensive off-the-shelf web-cameras and a low-end desktop computer (700MHz). The developed system shows the following abilities:

- Runs real-time (more than 25 fps) non stereo.
- Runs near real-time (more than 12 fps) in stereo.
- Track two colored spheres as pointing device in 3D (5 DOF).
- Track a viewer's position indirectly (3 DOF).
- Produce a viewer aligned 2D projection of the 3D-data on the screen enabling true moving parallax as if the objects were real.
- Enables the user to interact with the 3D-data using a pointing device, with a precision estimated to 1.5 cm and resolution of 0.5 cm. E.g. moving, rotating or editing objects. Even drawing coarse sketches in 3D is possible.

The main parts of the implementation are the data acquisition, visualization and interaction.

Data Acquisition

Data acquisition, in itself, is composed by several parts. First of all Image grabbing. Image grabbing functionality has been implemented capable of grabbing 25-30 fps from two web cameras simultaneously at 160x120 pixels of resolution.
Up to three colored spheres can be tracked, in each image with a common office environment as background, though it is not very robust to shifting lighting conditions.
The obtained stereo 2D coordinate pairs can efficiently (in less than 0.5 ms) be reconstructed into 3D coordinates in the coordinate system of the monitor. The camera data used has been obtained via camera calibration.
The camera calibration setup, proposed and applied in the system, relies on the 2D tracking and uses the monitor as a base object in calibration of the cameras.

Visualization

A visualization part has been implemented that handle and render a scene (virtual world). In stereo mode frame rates are low (12-15 fps). This is due to the lack of graphical processor power and lacking optimization on stereo

drawing methods. An optimization that solves this problem has been proposed.
A method for viewer alignment has been implemented. Viewer alignment sets up the appropriate frustum for one eye (or two eyes in stereo mode). This way the scene is rendered so the image on the screen gives the correct projection of the virtual world onto the retina of the viewer's eye (assuming the correct position of the viewer's eye is given).

Interactivity

A 3D pointing device has been implemented, with which it is possible to rotate and move objects and interact with them in an intuitive way compared to the use of mouse in traditional 3D applications.
Interactive objects have been implemented. E.g. a texture volume demonstrates how it is possible to model and draw freehand in 3D by adding or removing material. Especially organic shapes seem unusually fast and easy to create using this method. The method of 3D interaction could in principle be adapted to other 3D representations.

Final conclusion

Through the implementation it has been shown that it is possible to use the system as intended, proving the concept. Only the stereo visualization lacks in speed and clarity, which is temporary solvable problems.

The concept is proven by fulfilling the setup conditions. Many improvements can be done on each single part, leaving much future work still to be done. The system could be improved and specialized in a lot of different ways, using different approaches. In future development particularly stability, accessibility and flexibility would be important issues.

Chapter 9 Bibliography

[1] J. D. Foley, A. van Dam, A. K. Feiner, J. F. Hughes: ”Computer Graphics - Principles and Practice”, Addison-Wesley Publishing Company.

[2] M. Woo, J. Neider, T. Davis, D. Shreiner: ”OpenGL - Programming Guide”, Addison-Wesley.

[3] D. A. Forsyth, J. Ponce: ”Computer Vision. A Modern Approach”, Prentice Hall.

[4] B. Jähne, H. Haussecker: “Computer Vision and Application”, Academic Press.

[5] Kay Stanney: “Handbook of Virtual Reality”, Ed., Lawrence Erlbaum associates, 2002.

[6] Eric Foxlin: “Motion tracking requirements and Technologies”, chapter 8 in “Handbook of Virtual Reality”[5]

[7] Gary R. Bradski: ”Computer Vision Face Tracking For Use in a Perceptual User Interface”, Microcomputer Research Lab, Santa Clare, CA, Intel Corporation.

[8] J. M. Carstensen: “Image Analysis, Vision and Computer Graphics”, Technical University of Denmark, Lyngby 2001.

[9] 4D-vision, www.4d-vision.de (per 2004-04-01)

[10] Dimension Technolegies, www.dti3d.com (per 2004-04-01)

[11] SeeReal GmbH, www.seereal.com (per 2004-04-01)

[12] StereoGraphics, SynthaGram, www.stereographics.com (per 2004-04-01)

[13] Sharp, www.sharpsystems.com/products/pc_notebooks/actius/rd/3d/# (per 2004-04-01)

[14] Jin Liu, Siegmund Pastoor, Katharina Seifert, Jörn Hurtienne: “Three dimensional PC: toward novel forms of Human-Computer interaction”, Three Dimensional Video and Displays: Devices and Systems SPIE CR76, 5-8 Nov. 2000 Boston, MA USA. Three-Dimensional Video and Display: Devices and Systems SPIE CR76 (2000).

[15] ColorCode, www.colorcode3D.com

[16] Allan Aasbjerg Nielsen: “Least Squares Adjustment”, Technical University of Denmark.

[17] V. Lepetit, L Vacchetti, D. Thalmann, Pascal Fua: “Fully Automated and Stable Registration for Augmented Reality Applications” Swiss Federal Institute of Technolegy, Switzerland.

[18] Vicon, www.vicon.com (per 2004-04-01)

[19] UNC HiBall Tracker, www.worldviz.com (per 2004-04-01)

[20] Teófilo Emídio de Campos: “Hand Tracking for Intention Recognition”, Robot Research Group Department of Engineering Science University of Oxford (2003).

[21] N. A. Dodgson: “Autostereo Displays: 3D without glasses”, Computer Laboratory, University of Cambridge, UK.

[22] Rodney Cotterill: “Enchanted Looms – Consciuos Networks in Brains and Computers”, Cambridge University Press.

[23] C. Rezk.salama K. Engel et al. : “Interactive Volume Rendering on Standard PC Graphics Hardware Using Multi-Textures and Multi-Stage Rasterization”

[24] Matthew J. P. Regan, G. S. P. Miller et al. : ”A Real-Time Low-Latency Hardware Light-Field Renderer”, SIGGRAPH99, Interval Research Corporation (1999).

[25] **Henrik Aanæs IMM/DTU, www.imm.dtu.dk/~haa/ (per 2004-04-01).**

[26] **L. Naimark and E. Foxlin: "Circular Data Matrix Fiducial system and Robust Image Processing for Wearable Vision-Inertial Self Tracker", Intersense Inc., ISMAR2002.**

[27] **ReachIn, www.reachin.se (per 2004-04-01).**

[28] **SeeReal GmbH, www.seereal.com (per 2004-04-01).**

[29] **InterSense, www.isense.com (per 2004-04-01).**

[30] **Vincent Lepetit, Luca Vacchetti, Daniel Thalmann, Pascal Fua: "Fully Automated and Stable Registration for Augmented Reality Applications", Swiss Federal Institute of Technology, Lausanne, Switzerland, http://cvlab.epfl.ch/research/augm/augmented.html**

Chapter 10 Appendix

10.1 Transformation and projection

The translation, rotation and perspective projection in homogeneous coordinates for a given point **X** into **x** are:

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \omega x \\ \omega y \\ \omega \end{bmatrix}$$

, where $\frac{1}{\omega}$ is the factor with which ωx and ωy should be scaled according to perspective. Could also be understood as the depth coordinate z in the camera coordinate system.

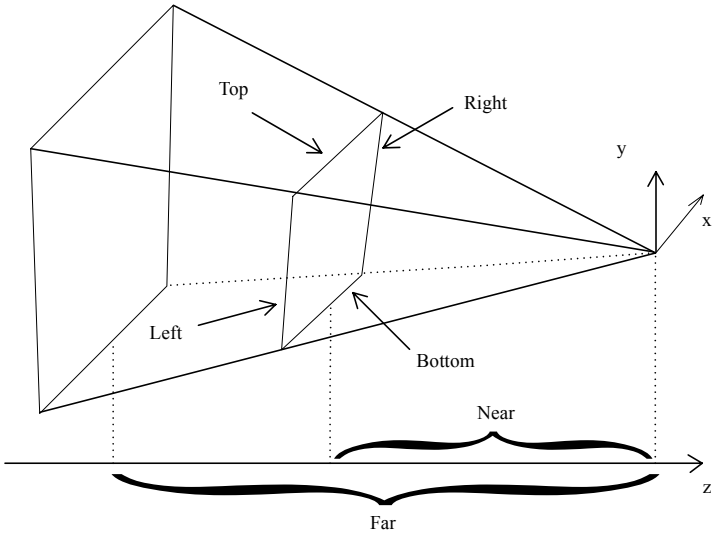
Perspective projection matrix

$$\begin{bmatrix} \omega x \\ \omega y \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & -1/c & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \quad \mathbf{x} = \mathbf{P} \cdot \mathbf{X}$$

, where c is the camera constant (i.e. distance from camera center to image plane).

Frustum

The perspective projection matrix can also be defined using the so called frustum:



$$\mathbf{P}(\text{Left, Right, Bottom, Top, Near, Far}) = \begin{bmatrix} \frac{2\,Near}{Right-Left} & 0 & \frac{Rigth+Left}{Right-Left} & 0 \\ 0 & \frac{2\,Near}{Top-Bottom} & \frac{Top+Bottom}{Top-Bottom} & 0 \\ 0 & 0 & \frac{-(Far+Near)}{Far-Near} & \frac{-2\,Far\,Near}{Far-Near} \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

The scale, translation and rotation matrix in homogeneous coordinates for a given point $\mathbf{X}$'s transformation into $\mathbf{X}_t$ are:

Scale

$$\begin{bmatrix} \omega X_t \\ \omega Y_t \\ \omega Z_t \\ \omega \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_x & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \ \mathbf{X_t} = \mathbf{S} \cdot \mathbf{X}$$

Translation:

$$\begin{bmatrix} \omega X_t \\ \omega Y_t \\ \omega Z_t \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -X_0 \\ 0 & 1 & 0 & -Y_0 \\ 0 & 0 & 1 & -Z_0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \ \mathbf{X_t} = \mathbf{T} \cdot \mathbf{X}$$

Rotation around X

$$\begin{bmatrix} \omega X_t \\ \omega Y_t \\ \omega Z_t \\ \omega \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & cos(\Omega) & \sin(\Omega) & 0 \\ 0 & -\sin(\Omega) & cos(\Omega) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \ \mathbf{X}_t = \mathbf{R}_\Omega \cdot \mathbf{X}$$

Rotation around Y

$$\begin{bmatrix} \omega X_t \\ \omega Y_t \\ \omega Z_t \\ \omega \end{bmatrix} = \begin{bmatrix} cos(\Phi) & 0 & -\sin(\Phi) & 0 \\ 0 & 1 & 0 & 0 \\ \sin(\Phi) & 0 & cos(\Phi) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \ \mathbf{X}_t = \mathbf{R}_\Phi \cdot \mathbf{X}$$

Rotation around Z

$$\begin{bmatrix} \omega X_t \\ \omega Y_t \\ \omega Z_t \\ \omega \end{bmatrix} = \begin{bmatrix} cos(K) & \sin(K) & 0 & 0 \\ -\sin(K) & cos(K) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \ \mathbf{X}_t = \mathbf{R}_K \cdot \mathbf{X}$$

Total rotation

$$\begin{bmatrix} \omega x \\ \omega y \\ \omega z \\ \omega \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & 0 \\ r_{12} & r_{22} & r_{32} & 0 \\ r_{13} & r_{23} & r_{33} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}; \ \mathbf{X}_t = \mathbf{R} \cdot \mathbf{X}$$

Total perspective projection

$$\mathbf{x} = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{T} \cdot \mathbf{X}$$
$$\mathbf{x} = \mathbf{P} \cdot \mathbf{R} \cdot \mathbf{T} \cdot \mathbf{S} \cdot \mathbf{X}$$

10.2 Map of choices