MEng Computing
Individual Project
Accurate Real-time Hand Tracking for Manipulation of Virtual Objects
Outsourcing Report

Thibaut Weise

January 31, 2005
## Contents

1 Introduction .................. 3

2 Background ................. 5
   2.1 Tracking Methods .......... 5
   2.2 Vision-based Hand Tracking and Recognition .......... 6
      2.2.1 Types of Gestures .......... 6
      2.2.2 Current Tracking Systems .......... 7
      2.2.3 Gesture Models .......... 8
      2.2.4 Hand localisation .......... 9
      2.2.5 Feature extraction .......... 10
      2.2.6 Tracking .......... 10
      2.2.7 Recognition .......... 10
   2.3 Colour Spaces .......... 11
   2.4 Calibration .......... 12
      2.4.1 Pinhole Model .......... 12
      2.4.2 Direct Linear Transformation Model .......... 13
      2.4.3 Photogrammetry Model .......... 14
      2.4.4 Types of Calibration .......... 14
      2.4.5 Image-World Coordinate Mapping .......... 15
   2.5 Triangulation and Epipolar Geometry .......... 15
      2.5.1 Triangulation .......... 16
      2.5.2 Epipolar Geometry .......... 17
   2.6 Virtual Environments .......... 19

3 Specification ................. 21
   3.1 Hand Tracking .......... 21
      3.1.1 Tracking Setup .......... 21
      3.1.2 Calibration .......... 21
      3.1.3 Tracking .......... 22
      3.1.4 Extensions .......... 22
   3.2 Virtual Object Interaction .......... 23
      3.2.1 Virtual Environment .......... 23
      3.2.2 Interaction with Virtual Objects .......... 23
      3.2.3 The Puzzle .......... 23
   3.3 Written work .......... 24
## 4 Evaluation and Testing

4.1 System Walkthrough ............................................. 25
4.2 Accuracy Testing ................................................ 25
4.3 Performance Testing ............................................ 26
4.4 Robustness Testing .............................................. 26
Chapter 1

Introduction

Mouse, keyboard and joystick have been the traditional user interface (UI) devices for human computer interaction (HCI) for the last decades. These devices are appropriate for text based and GUI software. Software that displays 3D data became very popular in CAD, research, animation and games in the last decade. Although mouse and keyboard are still appropriate for manipulating and navigating through 3D data on a 2D screen, the situation changes when the information is displayed in true 3D surrounding the user. Virtual and augmented environments (VE/AE) are becoming more widespread, but user interaction in those environments is still not mature. Wand and joystick are used but not very natural. Would it not be more natural to use the hands to directly manipulate virtual objects?

The major motivation for this project was to implement the "DaVinci Surgical System" [DaV] using stereoscopic vision instead of the current electro-magnetic approach. The "DaVinci Surgical System" consists of a console and a robotic apparatus that is used by the surgeon to do open surgery on human beings (e.g. heart surgery) using only tiny access ports. Robotic arms with different instruments and a stereoscopic camera access the body. The surgeon sees the 3D operational field at the console and controls the robotic arms using his hands. Currently these are tracked using an accurate yet obtrusive and expensive electro-magnetic approach. A stereo vision based approach would benefit from lower cost and less obtrusion on the surgeon.

Teleoperating robotic arms in general might greatly benefit from an accurate vision based tracking. This is certainly true for robots that operate in locations that are dangerous to humans or difficult to reach (e.g. outer space, deep water, radioactively contaminated areas and minefields). Manipulation of virtual objects can be seen as a very similar problem to those related to controlling a robotic arm that grasps and moves objects. In case of surgery, virtual objects could augment the operational field showing where the malicious tumor is located. Natural interaction with these objects is necessary. In augmented reality applications the objects also react on the physical environment.

The next section will present all the background and references that are necessary to understand what the current state of the art in 3D hand tracking and object interaction is. The background is necessary to implement the project which is clearly specified in the specification section. The evaluation section will state what the evaluation of the project is based.

It is important to keep in mind the project description when reading the background: The
project aims to track hands accurately (at least the thumb and index) in 3D space using two low cost cameras. The user can use his hands to grab and manipulate virtual objects like in reality. The user is assumed to explore the virtual world using 3D glasses and sees his hands as virtual representation. The vision environment is not a general purpose hand tracker, but is restricted to a relatively small and controlled working volume.
Chapter 2

Background

2.1 Tracking Methods

Hand tracking and gesture recognition has been a very active research area in recent years because of the many possibilities it offers for natural human computer interaction. There are different ways on how to track a hand, but the most dominant are magnetic, acoustic, optical and data gloves tracking [SZ94].

**Magnetic tracking**  Magnetic tracking is based on having a base station that emits a magnetic field. Sensors that consist of three orthogonal wire coils are used to calculate the position and orientation of the sensor within the magnetic field. To track a hand accurately a sensor has to be put on every finger as well as the palm to calculate the hand posture. Magnetic tracking is very accurate but also very sensible to magnetic objects in the environment that will distort the measurements. A second disadvantage is that the system is very obtrusive as sensors and wires are attached to the hand.

**Acoustic tracking**  Acoustic tracking is similar to magnetic tracking but a sound emitting device is used instead of the wire coils. At least three microphones are used to triangulate the position of the sound emitting device. It is not possible to measure orientation on a sensor. Problems with acoustic tracking are the obtrusive sensors similar as for magnetic tracking and the sensibility to acoustic reflections.

**Data Gloves**  Data gloves are gloves that contain different sensors which measure the finger and wrist bending. These measurements can be used to calculate the 3D model of the hand. Additional position and orientation tracking (for example using magnetic tracking) allows the user to have a fully tracked 3D hand. However, data gloves are obtrusive and expensive. A nice survey about data gloves can be found in [SZ94].

**Optical tracking**  Optical tracking is based on tracking the users’ hands using one or more cameras. Computer vision techniques are used to analyse the video sequence and reconstruct the 3D pose and motion of the hand. Advantages are that the user does not have to wear any special equipment and that the equipment is relatively cheap. Problems
are that fingers are difficult to track because of occlusion and fast motion. We use a vision based approach in our project as it has the greatest potential for widespread use because of the price and easiness for the user.

2.2 Vision-based Hand Tracking and Recognition

Vision-based hand tracking and gesture recognition is currently a very active research area because of the promising possibilities in human computer interaction through the advances in computer vision techniques. Several good review papers state the current state of the art [Der04, PSH97, WH99, MN03].

The following section will first describe the different types of gestures that exist and their relevance for our project. Current systems will then be described, what they achieve and what this project wants to deliver. The sections following will describe the components that most tracking systems consist of. We will adopt the description of Moeslund [MN03] that segments the recognition process of hand tracking systems into three components: image processing, tracking and recognition.

- **Image processing** reduces the noise in the images, finds the position of the hand and extracts feature points.
- **Tracking** is the process that adjusts the parameters of the gesture model for every frame based on the extracted feature points.
- **Recognition** uses the estimated parameters from the tracking process to decide whether the user made a useful spatial and temporal gesture.

2.2.1 Types of Gestures

Human beings naturally use hand gestures to interact and manipulate objects, communicate with other people and express their feelings. The distinction of gestures has been an important part in the research for hand tracking as this affects the method of tracking [PSH97]. Wu et al. [WH99] distinguish between four types of gestures: communicative, controlling, manipulative and conversational. The best example for communicative gestures are sign languages such as for example the American Sign Language. A lot of research has been done in this context as it is very suitable as test bed for gesture recognition algorithm and its potential benefit for communication for impaired people [ILI98]. Pointing, selecting and navigating are part of controlling gestures which are important aspects in many virtual environments. Manipulative gestures are the ones that are used to act and modify objects in the environment. This is a very natural way to interact with objects which should make it suitable for direct manipulation of virtual objects in a virtual environment.

In the context of human computer interaction this project will focus on manipulative gestures. Controlling a robotic arm to grab objects as well as the virtual counterpart (grabbing virtual objects with a virtual hand) are most intuitively handled in a way as users do it in the real world. An object is grabbed when the user encloses the object with his thumb and index. Moving the hand will move the virtual object and hand rotation will rotate the object. This intuitive way would be especially interesting when working with soft virtual objects, that is, objects deform when manipulated. To our knowledge no current vision-based tracking system explores deformation. Controlling gestures such as navigating and
pointing will not be used in the project. Selection is automatically handled by grabbing an object.

2.2.2 Current Tracking Systems

Most current tracking applications are used to manipulate objects in a virtual or augmented environment. This would mean that these tracking systems use manipulative gestures as this project proposes. However, most systems rely on general position tracking and communicative gestures to give commands and manipulate objects [SK98, UO99]. Moving the hand to the virtual object and making a specific gesture will select the object. Another second gesture will tell the system to rotate the object. This is unintuitive, unnatural and cumbersome for users to learn. However, directly manipulating the objects using a virtual hand requires a very accurate 3D tracking of the hand. Current full markerless 3D model-based systems are generally not accurate and fast enough.

[DUS01] uses an approach where the user selects the object by moving his finger into the object and grabs it by bending his finger. Rotation and translation are then based on the movement of the finger. Their system is relatively robust because it tracks (only) the index using reflective markers. The problem that remains is that of an unrelated gesture: the finger has to be bend to grab the object.

Figure 2.1: The user can grab the building with his thumb and index [BVBC04].

[OZ00] extend their model to use the hands to grab objects in natural way. Verma et al. [VKW04] created an arm and hand tracking system for telerobot control. They track index and thumb, but only track the open and close gesture very coarsely to control the robot. Buchmann et al. [BVBC04] have created an augmented reality urban planning system where users can grab buildings using their thumb and index and position the buildings on a map (see Fig. 2.1). They use markers to track the glove and give haptic feedback through
buzzers. Their interface is restricted to rotation in z axis and has currently an unreliable tracking system which can be explained because they only have a single camera. Their work describes that users felt that interaction was easy and intuitive, but also tiring. Users have to move the hand slowly and keep the hand oriented to the camera. The tiredness was identified to be mainly caused by tracking problems.

This project will follow Buchmann et al.’s [BVBC04] approach for manipulating objects in a virtual environment. We try to create a more robust tracking system that allows rotation around any axis and the use of two hands.

2.2.3 Gesture Models

Research on gesture and hand tracking is broadly divided into model-based and view-based (or appearance-based) approaches which are used for the tracking and recognition stages.

![Figure 2.2: A 27 DoF hand model and its generated contour [SMC01].](image)

**Model-based approaches**

Model-based approaches try to infer the 3D position and orientation of the hand as well as the joint angles which is ideal for realistic interactions in virtual environments. An articulated 3D model consists of connected geometric volumes such as general cylinders, deformable polygon meshes and super-quadrics that approximate the shape and bone structure of the human hand (see Fig. 2.2). The parameters of the model are the joint angles and the position and orientation of the hand. Thus, the degrees of freedom (DoF) of the hand model is roughly between 20 and 30. The first articulated model of that dimension was proposed by Rehg and Kanade [RK93]. The problem with this model is that the parameter space that needs to be estimated is very high. The parameter space is even more complex if the colour and size of the model parts also have to be estimated. Tracking is achieved by projecting the 3D model and comparing the current image feature points with those of the projection. The model that minimizes the difference is assumed as current model.

The number of parameter updates can be reduced by using the estimation of the previous frame and motion prediction using Kalman and/or Bayesian filters. More information can be found in Stenger’s thesis [Ste04].

Techniques to reduce the parameter space based on natural constraints of the human hand (joint angle limits and correlation of joint movement) speed up the search [Wu01]. Initialisation of the model and reinitialisation once the model loses track are the major problems
besides the complexity. Stenger[Ste04] uses a large database of templates to find the best match and improves efficiency using temporal information.

Dorfmueller et al. [DUS01] track only one finger and reduce the parameter space to nine DoF. They also use markers which can be tracked directly in 3D space which removes the task of minimizing the difference to match the markers against the model. This project will rely on tracking the hand in 3D space to interact with virtual objects which means that it has to use a (partial) 3D model of the hand.

**View-based approaches**

View-based (also called appearance-based) approaches directly link the appearance of the hand in the image to a specific static gesture or a sequence of images to a gesture action. View-based approaches basically omit the hand pose reconstruction part of the model-based approach. This means that less 3D information about the fingers can be extracted. Most systems extract the position and orientation of the hand. This project requires accurate finger tracking which makes the appearance-based approach unsuitable. More details and references about view-based approaches can be found in [Der04, PSH97, WH99, MN03].

### 2.2.4 Hand localisation

Tracking systems have to localize the hand in the image before feature extraction can be done. This is true for both model- and view-based approaches. The task of localizing the hand is still problematic in respect to illumination changes and background clutter. Currently localisation can be differentiated into roughly two categories: constraining the fore- or the background. What localisation is used greatly depends on where the system is used.

- Systems that constrain the background assume that the background is either uniform or static. The hand can then be localised by uniformity, background subtraction [SK98] or segmentation by motion cues. Uniform background can only achieved in very constrained lab settings. Temporal static background could be assumed depending on the tracking volume of the cameras, but this will still fail when illumination changes.

- Foreground constraint systems work by detecting skin colour [TCS04, ZYW00, Ahm95, SSK01] or detecting special markers or gloves that are attached to the user’s hand [DS94]. Colour detection can be done relatively reliably in the hue-saturation colour space using histograms matching or table-lookup, yet illumination changes and skin-coloured background degrade the detection. Infrared camera systems that improve detection of the human hand can be used for localisation [OSK02].

It is possible that certain regions in the image have falsely been detected as hand regions. These regions can normally be removed by restricting the number and the minimum size of the regions. Prediction of the location of the hand will further improve the localisation of the hand especially when several regions are falsely detected. Large undetected hand regions are problematic. Tiny pixel regions which are within the detected hand region that are due to noise can normally be filled. Combining the different methods to create a robust localisation method seems to be promising [ADYS03].
Hand localisation will also play an important role in this project. We can constraint the system to use a temporarily static background because both cameras only see a viewing volume containing the arms and hands of the user.

2.2.5 Feature extraction

Feature extraction uses the output from the hand localisation step. A description of different feature detection techniques for use with model-based approaches is presented here. More information can be found in [PSH97] that also discusses feature extraction techniques for appearance-based methods.

- **Silhouettes** are one of the simpler features to extract and can be easily matched against the projection of the 3D model [KH95]. This technique depends highly on the successful localisation of the hand.

- **Contours and Edges** are an extension of the silhouettes scheme as they use more information in the image. More notably the edges of fingers can be used in the matching as they will have parallel segments.

- **Fingertip** locations are used frequently as feature points as they enhance the 3D model matching. Fingertips have been extracted either based on markers, template matching or curvature pattern matching. The major problem of fingertip detection is occlusion.

- **Markers** are used to extract fingertips, but they are also used for the hand palm region. Markers seem to be the easiest approach to extract because the characteristics of the marker are known. Again occlusion is one major problem.

As stated before this project will use color and/or pattern marked gloves to detect the hand gesture.

2.2.6 Tracking

The tracking step is responsible for using the extracted features to estimate the hand parameters of the model. Marker-(and to some extend fingertip-) based approaches can directly use the position of the markers to estimate the hand pose if multiple cameras are used. Contour, silhouette and single camera approaches need to minimize the difference between the projected model and the image features.

Error minimization between model and image as well as occlusion problems can be improved using parameter prediction algorithms. These are modeled using uncertainty frameworks such as Kalman filter or Bayesian networks [Der04, Ste04]. These frameworks incorporate system dynamics and prior estimations to create a probability distribution over the parameters. The estimations guide the search in the next frame to adjust the model against the observation.

2.2.7 Recognition

The gesture recognition component uses the estimated parameters from the tracking stage and decides whether the user made a useful gesture. Both temporal and spatial information
is used for this task. In model-based approaches the recognition stage uses hand position and joint angles. In view-based approaches different parameters are used (see [PSH97]).

As uncertainty plays an important role, classic artificial intelligence algorithms such as Hidden Markov Models and Neural Networks are used for classification. Learning-through-examples technique is normally used to train these algorithms.

This project will not be concerned with gesture recognition. The 3D model data from the tracking stage is sufficient as the position, orientation and pose of the hand are directly used in the virtual environment.

2.3 Colour Spaces

As previously discussed in section 2.2.4, colour detection is a powerful approach to find the regions of interest in an image. This is true for both glove and skin colour based approaches. Video input sequences are generally in the 24bit RGB colour space which is not very suitable for colour detection because intensity and colour tone are not separated. Colour detection is generally done in a colour space that is invariant to illuminance or brightness. This helps to reduce the effect of shadows in the hand detection phase.

Two common approaches in the computer vision community are the HSV and normalized RGB colour space. HSV stands for hue, saturation and value [Wik] (see Fig. 2.3):

- **Hue** represents the colour type and ranges from 0 to 360. 0 represents red, 120 green and 240 is blue.
- **Saturation** is the purity of the colour and is defined from 0 to 100%. The less saturated the colour is the grayer it appears.
- **Value** is the brightness of the colour, again defined from 0 to 100%. A value of 0 represents black.

![Figure 2.3: HSV colour space [Wik]](image-url)

The normalized RGB colour space is the brightness invariant modification of the RGB colour space. Colours are calculated using the following formula:
\[ R_n = \frac{R}{R+G+B} \]
\[ G_n = \frac{G}{R+G+B} \]
\[ R_n + G_n + B_n = 1 \]

\( B_n \) can be calculated from the other two normalized values but is not necessary for detection. \( R_n \) and \( G_n \) represent a two dimensional triangle which can be used for classification.

Several systems using these invariant color models (or some variation) have been proposed to automatically detect skin colour using neural nets, histogram matching [Ahm95] and Bayesian classifiers.

## 2.4 Calibration

Camera calibration is the process that calculates the intrinsic and extrinsic parameters of a camera that relates 2D image coordinates to 3D world coordinate direction vectors. Without calibration it is not possible to infer metric information from 2D images.

This project uses two cameras to track the user’s hand. If features can be identified and matched in the images of both cameras, the mapping of 2D images coordinates to 3D world coordinate direction vectors can be used to calculate the 3D position of the feature points (see section 2.5) using triangulation.

To understand calibration it is important to know how images are formed. Camera models are mathematical descriptions that describe how images are formed. According to [Kov02] cameras models can be devised into three categories:

- Pinhole Model
- Direct Linear Transformation Model
- Photogrammetry Model

In general each model represents a refinement of the previous model. The photogrammetry model is the most complex but also most accurate. The calibration system that this project uses lies close to the photogrammetry model.

### 2.4.1 Pinhole Model

The pinhole model is the most basic camera model as it does not consider lens distortions, scaling and skew. See Fig. 2.4 for a picture. The point \( \mathbf{M} \) in the world coordinate system is projected onto the image plane \( \mathbf{R} \) at the intersection of the plane with the line \( \mathbf{CM} \). The optical axis passes orthogonal through the image plane \( \mathbf{R} \) at the ideal located principal point \( \mathbf{c} \). The distance \( \mathbf{Cc} \) is the focal length \( f \). Assuming that \( \mathbf{C} \) is the center of the world coordinate system, and \( \mathbf{c} \) lies at point \( (0 \ 0 \ 1)^T \) then the intersection of \( \mathbf{CM} \) with the plane \( \mathbf{R} \) can be calculated as

\[
\begin{pmatrix}
M_x \\
M_y \\
M_z
\end{pmatrix} =
\begin{pmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
m_x \\
m_y \\
1
\end{pmatrix}
\]
\[ m_x = \frac{M_x}{M_z} \quad m_y = \frac{M_y}{M_z} \]

Figure 2.4: Simplified pinhole model [Pol00]

However, \((m_x \ m_y \ 1)^T\) are not image pixel coordinates. The focal length \(f\) of the camera will not be 1 (in world coordinate distance). Thus, the coordinates have to be scaled accordingly and the ideally located principal point \(cp\) added to the location.

Hence image pixel point \(p\) is defined as

\[
(p_x \ p_y \ 1)^T = \begin{pmatrix}
  f & 0 & cp_x \\
  0 & f & cp_y \\
  0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
  m_x \\
  m_y \\
  1
\end{pmatrix}
\]

Note that none of the intrinsic parameters above have to be estimated as we assume the camera is perfect. The only calibration that needs to be done with this model are the extrinsic parameters. Note that we assumed above that the camera and world coordinate system are identical. However, in reality the world coordinate system is defined by the calibration plane and the rotation and translation between these coordinate systems has to be calibrated. Thus the first formula changes to

\[
\begin{pmatrix}
  m_x \\
  m_y \\
  1
\end{pmatrix} = \begin{pmatrix}
  r_{x1} & r_{x2} & r_{x3} & t_x \\
  r_{y1} & r_{y2} & r_{y3} & t_y \\
  r_{z1} & r_{z2} & r_{z3} & t_z
\end{pmatrix}
\begin{pmatrix}
  M_x \\
  M_y \\
  M_z \\
  1
\end{pmatrix} = RM + T
\]

2.4.2 Direct Linear Transformation Model

The direct linear transformation model (DLT) extends the ideal pinhole model by allowing a displacement of the optical center, scaling of the image axes and non-orthogonal image
axes (skew). Thus, the formula to calculate the pixel coordinate changes to

\[
(p_x \ p_y \ 1)^T = \begin{pmatrix}
    f_x & s & cpx
    \vspace{1pt}
    0 & f_y & cp_y
    \vspace{1pt}
    0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    m_x
    \vspace{1pt}
    m_y
    \vspace{1pt}
    1
\end{pmatrix}
\]

Note that all these parameters need to be estimated through calibration:

- \textbf{cp} - the true optical centre.
- \textbf{f}_x - the focal length divided by the pixel width (both in world coordinates). This means that \textbf{f}_x is the effective focal length in pixels.
- \textbf{f}_y - the focal length divided by the pixel height.
- \textbf{s} - the skewness factor which is based on the angle between x and y axes.

Note that there are slight differences between the calibration systems that are commonly used in the computer vision community. This description is based on the system used the Matlab Toolbox [MLT], Zhang [Zha00] and in [Pol00]. The OpenCV library [OpC], Tsai [Tsa87] and [HS97] do not use skew in their estimation. [HS97, Tsa87] also split \textbf{f}_x and \textbf{f}_y in individual components.

### 2.4.3 Photogrammetry Model

The pinhole model as well as the DLT model assume the ideal world where all light rays pass through a tiny pinhole. Cameras use lenses to focus more light rays onto the image plane. Lenses introduce radial and tangential distortion.

Distortion is due to the lens which means that the distortion calculation has to be applied before the intrinsic matrix in the above formula. Let \textbf{md} be the distorted coordinate and \( r^2 = m_x^2 + m_y^2 \):

\[
(md_x \ md_y \ 1)^T = \begin{pmatrix}
    (1 + k_1 r^2 + k_2 r^4)m_x
    \vspace{1pt}
    (1 + k_1 r^2 + k_2 r^4)m_y
    \vspace{1pt}
    1
\end{pmatrix}
+ \begin{pmatrix}
    2k_3 m_x m_y + k_4 (r^2 + 2x^2)
    \vspace{1pt}
    k_4 (r^2 + 2y^2) + 2k_4 m_x m_y
    \vspace{1pt}
    0
\end{pmatrix}
\]

\[
(p_x \ p_y \ 1)^T = \begin{pmatrix}
    f_x & s & cpx
    \vspace{1pt}
    0 & f_y & cp_y
    \vspace{1pt}
    0 & 0 & 1
\end{pmatrix}
\begin{pmatrix}
    md_x
    \vspace{1pt}
    md_y
    \vspace{1pt}
    1
\end{pmatrix}
\]

Again, different calibration approaches use distortion to different extend. Tsai [Tsa87] only uses 2nd-order radial distortion. [MLT, Zha00, OpC, HS97] use the distortion parameters presented here, but [MLT] also provides the possibility to add 6th-order radial distortion.

### 2.4.4 Types of Calibration

There are various types of calibration techniques that are used to find the parameters for a camera model discussed in the previous section. Roughly these can be divided into three categories [Zha00]:
Photogrammetric calibration - the calibration is performed by observation of a calibration object whose geometry is known with very high precision. Calibration can then be done using the known 3D feature coordinates and the identified image points. This technique is very efficient and accurate but needs a very precise calibration object which is difficult to create.

Coplanar calibration - calibration systems that use this technique require a planar image with known feature points and 2D dimensions. A chess board pattern is used in many cases. The advantage of coplanar calibration is that the calibration object can be created using a laser printer and the image is attached onto a planar surface such as a book. The object is seen from various viewing angles and the parameters can be calculated from these views.

Self-calibration - this technique does not require any calibration object. Images from different views of the same static scene are enough to find corresponding feature points and calculate intrinsic and extrinsic parameters. In recent years straight lines have been used for calibration with promising results. The advantage of these are that they are similar in accuracy to coplanar approaches but do not require the design of a pattern - straight lines are enough.

The project will probably use the coplanar approach as support software libraries are available [OpC, MLT] and the accuracy seems to be satisfying.

2.4.5 Image-World Coordinate Mapping

World to Image Coordinates
The intrinsic and extrinsic parameters of a camera can be used to calculate the projection of a 3D world coordinate point onto the image plane in pixel coordinates. First the normalized image coordinates are calculated by transforming the world coordinate into the camera reference frame using the extrinsic parameters. The resulting image points are then distorted using the distortion coefficients. The distorted image coordinate is then transformed into pixel coordinates using the camera parameters.

Image to World Coordinates
The inverse mapping of a pixel coordinate to the 3D world coordinate vector is more complicated. The distortion calculation cannot be inversed and has to be solved numerically. Once a camera is calibrated, a warping image can be defined that will undistort the image [OpC]. Once the image has been undistorted, the 3D world coordinate vector can be calculated by multiplication with the inverse of the intrinsic (excluding the distortion) and the extrinsic matrix.

2.5 Triangulation and Epipolar Geometry

Both triangulation and epipolar geometry can be found in Trucco [TV98] on which the explanation here is based on. Another good description can be found at [VLe]. For the following calculations we assume that radial distortion is removed using mapping and interpolation as described in section 2.4.5.
2.5.1 Triangulation

Notation (subscript \( L \) and \( R \) denote the notation for left and right camera respectively):

- \( M_{in}^L \) and \( M_{in}^R \) denote the intrinsic matrices that map camera coordinates to pixel coordinates.
- \( T_L \) and \( T_R \) denote the translation from world to camera reference frame.
- \( R_L \) and \( R_R \) denote the rotation from world to camera reference frame.

A point \( P \) in world coordinates can be converted to camera coordinates by the following formula

\[
P_L = R_L P + T_L
\]

and

\[
P_R = R_R P + T_R
\]

Similar to convert from camera coordinates to world coordinates:

\[
P = R_L^T (P - T_L)
\]

This means that the center of projection \( O_L \) of the left image is

\[
O_L = R_L^T (0 - T_L) = -R_L^T T_L
\]

in world coordinates.

For corresponding feature points \( \text{im}_L \) and \( \text{im}_R \) that are the projections of \( P \) we have (\( px \) and \( py \) are the pixel coordinates of the image points):

\[
\text{im}_L = \begin{pmatrix} px_L \\ py_L \\ 1 \end{pmatrix}
\]

and

\[
\text{im}_R = \begin{pmatrix} px_R \\ py_L \\ 1 \end{pmatrix}
\]

The camera coordinates for these feature points are:

\[
p_L = M_{in}^{L^{-1}} \text{im}_L
\]

Similar for the right camera. We know that the point \( P_L \) has to lie on the line \( \alpha p_L \) for the left camera reference coordinate system, and \( P_R \) on \( \beta p_r \) for the right one. We can convert this to world coordinates:

\[
P = R_L^T (P_L - T_L) = R_L^T (\alpha p_L - T_L) = \alpha R_L^T p_L - R_L^T T_L = \alpha R_L^T M_{in}^{L^{-1}} \text{im}_L - R_L^T T_L = O_L + \alpha D_L
\]

and

\[
P = \beta R_R^T M_{in}^{R^{-1}} \text{im}_R - R_R^T T_R = O_R + \beta D_R
\]

One problem is that the two lines will not intersect because of measurement errors. This means we have to find the point that is closest to both lines. The point with minimum
distance to two lines lies in the middle of the line segment that connects and is perpendicular to both lines:

\[ O_L + \alpha D_L + \gamma (D_L \times D_R) = O_R + \beta D_R \]

We solve this linear equation for \( \alpha, \beta \) and \( \gamma \). The world coordinate of the \( P \) is then

\[ P = O_L + \alpha D_L + \frac{1}{2} \gamma (D_L \times D_R) \]

### 2.5.2 Epipolar Geometry

Epipolar geometry helps in finding corresponding feature points as it reduces the search space from two to one dimension. A point \( P \) will form a plane with \( O_R \) and \( O_L \). This plane intersects with both image planes of the left and right cameras (unless they are parallel - which cannot happen if it is a visible feature). The intersection of the plane with the image planes are the epipolar lines. Any point that is on the plane will be projected onto the epipolar lines. Using a projected feature point on the left image we can create the plane and intersect it with the second image plane. We only need to look on the epipolar line for the projected feature point. The advantage is that it is much easier to find correspondence between feature points as they have to lie on epipolar lines.

Let us consider the left camera reference frame. We have three vectors:

- \( P_L \) is the point we want to find.
- \( T \) is the center of projection of the right reference frame.
  \[
  T = R_L O_R + T_L = R_L (-R_L^T T_R) + T_L = -R T_R + T_L \\
  R = R_L R_R^T
  \]
- \( P_L - T \) is the difference vector between the point we want to find and the right center of projection.
  \[
  P_L - T = R_L R_R^T (P_R - T_R) + T_L - R_L (-R_L^T T_R) - T_L = R_L R_R^T P_R = R P_R
  \]

We can define the coplanarity condition using the three vectors:

\[
(P_L - T)^T (T \times P_L) = 0
\]

Thus,

\[
(RP_R)^T (T \times P_L) = 0
\]

Vector product can be rewritten in matrix form:

\[
T \times P_L = SP_L = \begin{pmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ T_y & T_x & 0 \end{pmatrix}
\]

\[
\Rightarrow (RP_R)^T SP_L = P_R^T R^T SP_L = P_R^T EP_L
\]
$E$ is called the essential matrix and can be calculated using the extrinsic parameters of the cameras. We know that

$$P_L = \alpha M_L^{in^{-1}} \mathbf{im}_L$$

and

$$P_R = \beta M_R^{in^{-1}} \mathbf{im}_R$$

Thus,

$$P_R^\top EP_L = (\beta M_R^{in^{-1}} \mathbf{im}_R)^\top E \alpha M_L^{in^{-1}} \mathbf{im}_L = 0$$

which means that

$$(M_R^{in^{-1}} \mathbf{im}_R)^\top EM_L^{in^{-1}} \mathbf{im}_L = \mathbf{im}_R^\top M_R^{in^{-1}} E M_L^{in^{-1}} \mathbf{im}_L = \mathbf{im}_R^\top F \mathbf{im}_L = 0$$

$F$ is called the fundamental matrix and can be calculated using the intrinsic and extrinsic parameters of the cameras. Every two corresponding feature points $\mathbf{im}_R$ and $\mathbf{im}_L$ have to satisfy the equation

$$\mathbf{im}_R^\top F \mathbf{im}_L = 0$$

The calibration from section 2.4 returns the intrinsic and extrinsic parameters which can be used to calculate $F$. As noted at the beginning of this section, we assume that distortion has been removed beforehand using a mapping and interpolation algorithm.

**How to use $F$**

When we find feature points in both images, then we need to check which feature points lie on the same epipolar line. Imagine we find a feature point in the left image

$$\mathbf{im}_L = \begin{pmatrix} px_L \\ py_L \\ 1 \end{pmatrix}$$

We know that the z coordinate of the corresponding point $\mathbf{im}_R$ is 1 (defined by calibration). Hence,

$$\mathbf{im}_R^\top F \mathbf{im}_L = (px_R py_R 1)F \begin{pmatrix} px_L \\ py_L \\ 1 \end{pmatrix} = 0$$

$px_R$ and $py_R$ are the only variables in the above equation. We can thus calculate whether a pixel coordinate is on the epipolar line. We can also create an equation for the epipolar line in pixel coordinates. This can then be used to calculate the distance of a point to the epipolar line. That may be important as a point will never be directly on the epipolar line because of measurement errors.

When using a feature point from the right camera to check for the epipolar line in the left image, we have to use the transpose of the fundamental matrix.

It is possible to calculate the intrinsic and extrinsic matrices from the fundamental matrix, but this is out of the scope of this project. Note that epipolar geometry only works with distortion free cameras or otherwise epipolar lines are curved.
2.6 Virtual Environments

The aim of this project is to track the hands accurately to interact with virtual objects in a virtual environment (VE). A VE is a computer generated world that user can immerse into and interact with [Sta]. In general VEs are displayed using head mounted displays or CAVE environments so that the user has the feeling of being in a different world. VEs are used for scientific visualisations (e.g. molecules or engines), architecture, training and entertainment. Most interaction tasks are currently done using wands, 3D mice, joystick or data gloves. Glove based systems use gestures of the hand to interact with the objects and some even provide haptic feedback when touching and deforming objects. Tzafestas [Tza03] uses a haptic glove with which people can grasp and deform objects in the virtual environment (see 2.5).

Figure 2.5: Users can grab virtual objects with their hand and get haptic feedback. [Tza03]

Most vision-based systems use only the 3D position of the hand and the gesture to execute some action [CFH97]. Huang et al. [HBTT95] is one of the few exceptions that use the actual grasping action to move objects. Buchmann et al. [BVBC04] use the actual finger location of index and thumb to grasp objects in an AE. [DUS01] uses the index and grasping is done by bending the finger.

Tzafestas [Tza03] glove-based system is ideal because it provides very precise input control and the user “feels” the objects as if they were real. The problem with these systems is that they are very obtrusive. A less obtrusive vision-based interface would be ideal. Haptic feedback gives the user a much more realistic feeling. Using haptic feedback with vision-based interfaces seems problematic as this would again introduce the obtrusion through tethered gloves.

Implementation

The project will implement a virtual game environment where nine multi-textured cubes are randomly scattered over the floor. Each side of the cubes is textured with a part of six different images so that the nine cubes joined together form one of the six images. The user
tries to figure out which sides belong together and assembles them in one wall so that the image is clearly visible. The user has to rotate and move the cubes to accomplish the task.

Collision detection between objects will have to be done in order to classify whether the hand grasps an object and to check whether cubes collide. Information to collision detection and a software library can be found at [Col].

Several possibilities exist to create virtual environments. OpenGL\(^1\) or DirectX\(^2\) can be used to display the 3D world. A virtual reality toolkit such as VR Juggler\(^3\) can speed up the application implementation.

\(^1\)http://www.opengl.org
\(^2\)Available at http://www.microsoft.com
\(^3\)http://www.vrjuggler.org
Chapter 3

Specification

The specification describes exactly the expected practical and written output of the project. There are two practical parts which have to be implemented: Hand tracking and virtual object interaction. Both components will be incorporated into one application. The implementation language is irrelevant as long as all the specifications are met.

3.1 Hand Tracking

This section specifies what the hand tracking system will be capable of and what assumptions it can make concerning the environment.

3.1.1 Tracking Setup

- Two cameras are used to track the user’s hands.
- The two cameras are attached to one PC that has to cope both with the tracking of the hands and the visualization of the virtual environment.
- The PC has one video input that contains both camera images horizontally aligned.
- The camera setup is not specified but both cameras will share at least a viewing volume of $50\text{cm}^3$ and see the lower arms and hands of the user. It cannot be assumed that both cameras are set up in parallel.
- The PC will have current standard specifications.

3.1.2 Calibration

- Both cameras will be easy to calibrate using a planar chess board pattern. Linear or self-calibration techniques are also possible. Calibration that requires a 3D object is not suitable. OpenCV [OpC] is recommended because it uses a chess board pattern and is accurate and fast.
• Interaction concerning calibration has to be kept to a minimum. Calibration should not take more than two minutes.

• Calibration accuracy of the system will be within 3mm and can be determined using an independent procedure where the object location and translation is known and compared to the output of the system.

3.1.3 Tracking

Assumptions

• The background of each camera is assumed to be temporarily static.

• Illumination of the scene only changes minimally.

Core requirement

• The system has to be able to track both hands.

• The 3D location of the joints and fingertips of both thumb and index are tracked.

• Tracking has to be accurate within 6mm.

• The tracking system can use gloves with suitable passive markers based on pattern and/or colour. No infrared system can be used. The gloves have to be lightweight, comfortable, cheap and easy to make. Initially three markers on thumb, index and hand are used.

• The minimal update rate is ten frames per second.

• The maximum latency should lie below 150ms.

• The system will cope with shadows.

• The system will cope with temporal occlusion.

• Initialisation of the tracking will be automatic.

The hand tracking component outputs the tracked 3D locations to the virtual object interaction component. Both components have to agree on the 3D hand model that is used for the 3D locations.

3.1.4 Extensions

The extensions might be added once the core requirements are fulfilled.

1. Several fingers are tracked.

2. The constraints on background and illumination are relaxed.

3. The system can track the hands without any markers.
3.2 Virtual Object Interaction

The virtual object interaction component is responsible for creating and displaying the 3D virtual game environment containing a puzzle that the user has to assemble.

3.2.1 Virtual Environment

- A virtual 3D environment is created that consists of simple textured cubes.
- The environment is displayed using a common graphics library such as OpenGL or DirectX.
- The environment does not need to obey any laws of physics.
- The tracking output is displayed in the environment as a virtual hand. It may consist only of a thumb and index.
- Moving the real hand will move the virtual hand accordingly using the hand tracking. Important is that the user feels the control of the virtual hand.

3.2.2 Interaction with Virtual Objects

- The virtual hand can be used to push an object. The objects move in the direction the hand pushes and only as far as the hand moves.
- The virtual hand can grab objects by enclosing objects with thumb and index. The object can be rotated and moved as long as the hand grabs it. An object is grabbed if thumb and index touch opposite sides of the object.
- The hand may move into the object when the object is grabbed (the situation occurs when the object is grabbed and the user “closes” his hands).

3.2.3 The Puzzle

- The puzzle is used to demonstrate the capabilities of the system.
- The puzzle consists of nine multi-textured cubes scattered over the virtual floor.
- Each of the six sides of a cube is textured with one part of six different images.
- The user has to reassemble the cubes into a three by three wall structure such that one of the images is visible on the wall side. This means the user has to figure out which sides of the cubes belong together and which position each cube has within the wall structure.
- A button on the floor can be pressed to rescatter the cubes on the floor and possibly texture them with different pictures.
3.3 Written work

A project report has to be handed in with the practical work. The report has to conform to the guidelines set by the computing department\(^1\). Important is that the following is provided at the right places in the report:

- A detailed description of the tracking methods used and a comparison to approaches in existing published work.
- An exact description of the calibration technique that is used as well as the procedure to assess the accuracy.
- The hand tracking method which is used including mathematical foundations. Hand localisation, feature point extraction, 3D reconstruction and tracking method have to be described precisely.
- A short description of the virtual world and a thorough explanation of the object interaction.
- The evaluation will clearly state the accuracy of the hand tracking method, how well it copes with occlusions and tests with real users.
- A user guide for the application is necessary including a tutorial to easily start a virtual environment session.

\(^1\)http://www.doc.ic.ac.uk/~ajf/Teaching/Projects/Guidelines.html
Chapter 4

Evaluation and Testing

The following sections describe the necessary steps to evaluate the project. The project is successful if it fulfills the specification. Unfortunately several aspects of the system will be hard to evaluate because precise measurements are not possible.

4.1 System Walkthrough

This evaluation is concerned with the ease of setting up and using the system. Despite the fact that the system is designed to work in a restricted environment, it is important that it is easy to set up. Ideally a novice user should be able to read the user manual, position and calibrate the cameras and start exploring the virtual environment. Connecting the cameras to the PC does not have to be described in the manual.

A usability test with novice users will be used to check whether the application is well structured, the manual well written and the system robust. Especially the calibration has to be intuitive so that the user can successfully complete it.

4.2 Accuracy Testing

Accuracy is stated in the specification to be within 3mm for calibration and 6mm for hand tracking. Calibration will never be perfect because of inaccuracies in the measurements. Accuracy of calibration will be assessed using various methods:

- Each camera is calibrated and the location of the projected coordinate compared to the actual detected coordinate. This will give a first estimation of the error.
- Both cameras are calibrated at the same time to improve the calibration. As in the previous method the projected and the detected coordinates are compared.
- Both cameras track an object in 3D space where the exact location of the object is known within 0.2 mm of accuracy. The known locations of the object against the output of the tracking system will give an estimation of the error of the system.
Tracking inaccuracies are much harder to test as no rigid object can be used for testing. It is actually not even clear how important inaccuracies in measurements are. In augmented reality inaccuracies will come up because the user sees his own hand and the misalignment with the objects he interacts with. In virtual environments and robot arm control applications the user controls a virtual hand. Thus, small inaccuracies in 3D reconstruction because of calibration and measurement errors might not affect the user’s experience. A usability evaluation to explore this problem will consist of a simple marker on a pen that the user will use to interact with the virtual environment. Users will report on consistency, usability and control they experienced during the test.

4.3 Performance Testing

The application will be tested on a current standard PC to test whether it can operate in real-time with a minimum tracking update rate of 10 frames per second and a latency below 150ms. The application has to track the hands and visualize the virtual environment.

4.4 Robustness Testing

Two more informal tests will be performed to evaluate the robustness of the system.

- **Occlusion**: How well does the system cope with occlusion of fingers with respect to one or both cameras?
- **Speed**: How well does the system cope with the speed of the hand movement?

Comparison to other Systems

This system will be compared to current established state-of-the-art systems and the contribution of the work to the research community evaluated.

Written work

The written report will be examined whether it provides all the details that were described in the specification and the guidelines for the report.
Bibliography


[VL] Vision lectures.


