

Masterthesis Medientechnologie

RHOT

A real time hand and object tracking system with
low cost consumer grade hardware

vorgelegt von

Oliver Kalbfleisch

vorgelegt an der

TECHNISCHEN HOCHSCHULE KÖLN

CAMPUS DEUTZ

FAKULTÄT FÜR INFORMATIONEN-,
MEDIEN- UND ELEKTROTECHNIK

im Studiengang

MEDIENTECHNOLOGIE

Erster Prüfer: Prof. Dr. Peter Silie

Technische Hochschule Köln

Zweiter Prüfer: Prof. Dr. Maria Musterprof

Technische Hochschule Köln

Köln, im August 2018

Masterarbeit

Titel: Titel der Arbeit

Gutachter:

Prof. Dr. rer. nat. Adam Mustermann (TH Köln)

Dipl.-Ing. Eva Musterfrau (Media AG, Frankfurt)

Zusammenfassung: Was ein Abstract ist wird in der DIN Norm 1426 festgelegt: es ist ein Kurzreferat zur Inhaltsangabe. Die Definition des American National Standards Institute (ANSI) lautet: „An abstract is defined as an abbreviated accurate representation of the contents of a document”.

Es sollten 8-10 Zeilen Text folgen.

Stichwörter: Stichwort, Stichwort (hier sollten maximal 5 Stichwörter folgen)

Sperrvermerk: (optional) Die Einsicht in diese Arbeit ist bis zum TT. Monat JJJJ gesperrt.

Datum: TT. Monat JJJJ

Masters Thesis

Title: Title of the work

Reviewers:

Prof. Dr. rer. nat. Adam Mustermann (TH Köln)

Dipl.-Ing. Eva Musterfrau (Media AG, Frankfurt)

Abstract: Hier sollte eine Übersetzung des obigen Abstracts auf Englisch erfolgen (und kein komplett neuer Text). Auch hier bitte die Begrenzung auf maximal 10 Zeilen Text ein-halten.

Keywords: (hier die Übersetzung der obigen Stichworte)

Restriction notice: (optional, falls Sperrvermerk oben ebenfalls existiert)

The access to this thesis is restricted until jdd month YYYY_i.

Date: dd month YYYY

Contents

1	Introduction	1
2	Description of hand in digital space	2
2.1	Physiological structure of the human hand	2
2.1.1	Constraints in Hand Motion	3
2.2	Kinematics	5
2.2.1	Forward Kinematics	6
2.2.2	Inverse Kinematics	6
2.2.3	Jacobian Inverse	7
2.2.4	Damped Least Squares	10
2.2.5	Newton Methods	10
2.2.6	Sequential Monte Carlo Method	11
2.2.7	FABRIK Algorithm	12
3	Tracking in real space	14
3.1	General tracking technologies	14
3.1.1	Optical tracking	14
3.1.2	Magnetic tracking	15
3.1.3	Accoustic tracking	16
3.2	Hand tracking Systems	16
3.2.1	Sensor based	17
3.2.2	Appearance Based - Optical marker	18
3.2.3	Model Based - Image analysis	20
4	Digital hand models	22
5	System prerequisites	24
5.1	Display	24
5.2	Tracking Hardware	25
5.3	image analysis	26
5.4	inverse kinematics	27
5.5	handmodel	27
6	System conception	28
6.1	Technical conception	28

Contents

6.2	Image Analysis with OpenCV 3 and Python on the Raspberry	28
6.2.1	Image acquisition from camera	30
6.2.2	Image conversion to HSV colorspace	30
6.2.3	Mask construction and filtering	31
6.2.4	Contour finding and position calculation	31
6.2.5	Sending of positional data via network as UDP package	32
7	System Evaluation	33
8	Resume	34
	Abbildungsverzeichnis	i
	Tabellenverzeichnis	ii
	Literaturverzeichnis	vii

1 Introduction

The standard interface between human and computer has for long years been mouse and keyboard. But with the advance of technology, new interfacing methods were developed in the last few years.

Touch technology for interfacing with mobile devices and desktop computers has become a reliable technology and has been integrated into our everyday lives. advances in capabilities of CPU as well as GPU hardware has build a foundation for the use of advanced AR and VR technology. 3D and stereoscopic rendering can now be accomplished even by mobile hardware (with some limitations) without the need of specifci hardware. For an intently immersive experience VR googles are used to explore digitally created worlds. With this level of immersiveness, a touch device or just a mouse and keyboard setup is rather hindering the user experience and the logical and more natural way of interfacing with our digital technology would be by utilizing the build in human control elements that we carry around with us every day. The human hands offer a build-in control element that provides a large number of Ddegrees-of-Freedom (DOF's) and furthermore adds the posiibility of combine these witch a tactile component.

The first more humble attemps for the solving of this problem already began in the early 1980's where a lack of computational power and low-res imaging hardware made more complex systems impossible. These systems were mostly aimed at registering simple hand gerstures like pointing as an intefrace method an were not able to reconstruct full hand positions in realtime[4].

This thesis aims to display a method of combining the tracking of a physical object and the human hand(s) and merging this data into the digital space. The system should be composed of mostly consumer grade technology and easily accesible hardware to provide a low cost solution. The resulting experience for the user should be as natural as possible therefore the hindering factor of the tracking solution in terms of motion capability for the hands should be kept as and as comfortable as possible.

To achieve such a solution,different systems for tracking of objects and bodyparts as well as the display of these in digital space will be presented and assessed. The results from the assessment will be incorporated into a system prototype.

more
context
for in-
troduc-
tion?

2 Description of hand in digital space

Tracking of the human hand has always been a challenging problem. In comparison to other larger bodyparts like the arm or the head, the human hand itself contains a large variety of smaller parts, namely bones and muscles. These components have to be taken into account when trying to replicate the natural motion of the hand in digital space.

2.1 Physiological structure of the human hand

Lee and Kuni [25] describe the human hand as "an articulated structure with about 30 degrees of freedom [which] changes shape in various ways by its joint movements."

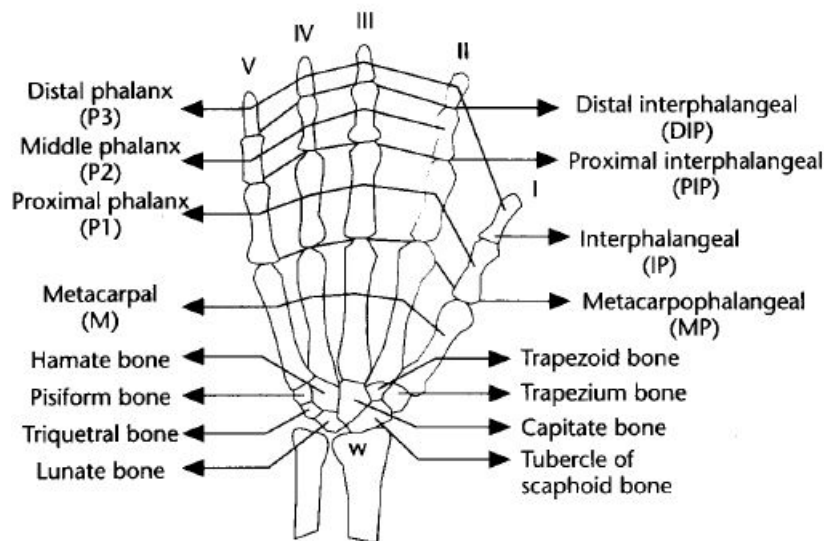


Figure 2.1: Bone structure of the human left hand ([25])

All of the hand components are connected to at least one neighboring component via a joint. The joints affect the position of the connected components. To describe the movement of the hand components, we can use the rotation angles of the joints to correlate to a specific position.

To do so, we define a local coordinate system for each of the existing hand joints. By doing so, we achieve a sequence of rotations in the local coordinate systems of the joints. Such a sequence can then be used to describe a specific movement and/or position of

a component. Not all of the joints in the human hand have equal degrees of freedom. Their functionality can be classified in the amount of DOFs (Degrees of freedom)[20]

- 1 DOF
 - A joint movement that can perform a **flexion** or **twist** in one direction
- 2 DOF
 - A joint movement that can perform **flexion** in more than one direction (**directive**)
- 3 DOF
 - A joint movement that permits simultaneous **directive** and **twist** movements.(**spherical**)

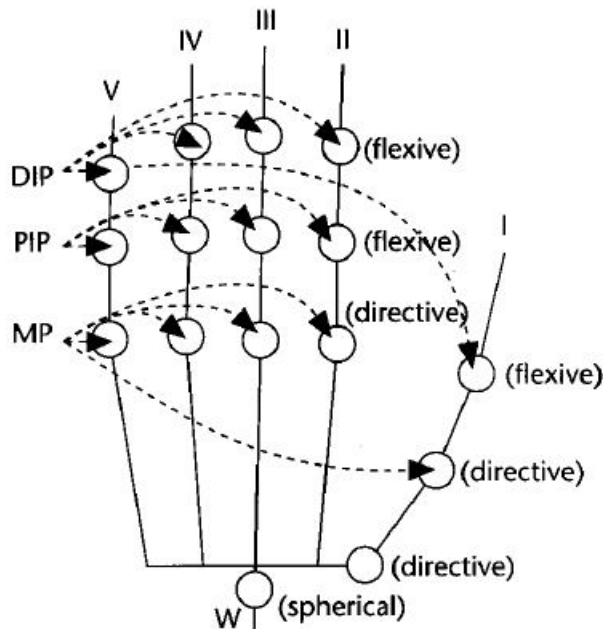


Figure 2.2: Representation of the DOFs of the human hand

When looking at the DOFs displayed in Figure 2.2, each finger (II-V) sums up to 4 DOFs and the thumb to 5 DOFs. Also considering 6 DOFs for the rotation and position of the whole and itself, the result gets us to 27 DOFs for the human hand.

2.1.1 Constraints in Hand Motion

A full usage of all the declared DOFs would lead to a large amount of possible combination. Since the hand is not only made up of bones but also Muscles and the skin, we can impose some constraints [3, 34] to the movement of the joints. Ling, Wu and Huang([26]) proposed following classification for the constraints:

- **Type I constraints**

- A constraint that limits the range of finger motions based on hand anatomy

- **Type II constraints**

- A constraint that the position of the joints during finger movement

- **Type III constraints**

- A constraint that limits position based on natural hand motions

The **Type I** and **Type II** constraints rely on the physiological and mechanical properties of the hand. **Type III** constraints are results of common and natural movements and can be differing from person to person. As these movements are to some degree similar for everyone, a broad grouping can be applied. The curling of the fingers at the same time when forming a fist is way more natural than curling each finger by itself. Here the motion of the hand is quite similar between different persons, but the constraints cannot be described in a mathematical form.

A **Type I** constraint example would be that the position of the fingertip is limited by the length of the other finger segments and thereby can only reach as far as the combined length.

An example for **Type II** constraints would be that, for your fingertip to touch your hand palm, all joints in the finger have to be bent to achieve this position. The following inequalities can be used to describe these constraints:

Type I:

$$\begin{aligned}
 0^\circ &\leq \Theta_{MP_flex} \leq 90^\circ \\
 0^\circ &\leq \Theta_{PIP_flex} \leq 110^\circ \\
 0^\circ &\leq \Theta_{DIP_flex} \leq 90^\circ \\
 -15^\circ &\leq \Theta_{MP_abduct/adduct} \leq 15^\circ
 \end{aligned} \tag{2.1}$$

A further constraint that is specific to the middle finger is, that this finger's MP normally does not abduct and adduct much. Therefore we can infer an approximation and thereby remove 1DOF from the model:

$$\Theta_{MP_abduct/adduct} = 0^\circ \tag{2.2}$$

The same behavior can be seen in the combination of hand parts labeled W (the connection point between hand and lower arm). This approximation also eliminates one DOF on the connected thumb:

$$\Theta_{W_abduct/adduct} = 0^\circ \tag{2.3}$$

Since the DIP, PIP and MP joints of our index, middle, ring, and little fingers only have 1DOF for flexion, we can further assume that their motion is limited to movement in one plane.

Type II:

The **Type II** constraints can be split into interfinger and intrafinger constraints. Regarding intrafinger constraints between the joints of the same finger, human hand anatomy implies that to bend the DIP joints on either the index, middle, ring or little fingers, the corresponding PIP joints of that finger must also be bent. The approximation for this relation[36] can be described as :

$$\Theta_{DIP} = \frac{2}{3}\Theta_{PIP} \quad (2.4)$$

Interfinger constraints can be imposed between joints of adjacent fingers. Interfinger constraints describe that the bending of an MP joint in the index finger forces the MP joint in the middle finger to bend as well.

When combining the constraints described in the above equations, the starting number 21 DOF's of the human hand can be reduced to 15. Inequalities for these cases, obtained through empiric studies, can be found in [25].

2.2 Kinematics

The preceding sections gave an overview of how we can describe a model of the human hand and introduced some limiting constraints. With the model and the constraints, we can now start to build a kinematic system for the animation of the model.

Kinematic systems contain so called *kinematic chains*, which consist of a *starting point* or *root*, kinematic elements like *joints*, *links* and an *endpoint*, also called *end effector*. Applied to the human hand, the whole hand model represents the kinematic system. This system contains several *kinematic chains*, namely the fingers of the hand with the fingertips being the *end effectors* of each of these chains.

Each of these components has its set of DOF's which can be described mathematically. The established convention for describing this kind of linked systems is the *Denavit-Hartenberg* convention (D-H)[9]. It was actually introduced for the order to standardize the coordinate frames for spatial linkages in the field of robotics, but has since then been applied in other fields as well.[38]

As we begin to move our hands, the states of the kinematic chains begin to change. Joint angles and end effector positions are modified until the end position is reached. To represent the new position and angle dataset of our physical hand with our kinematic system, two major paths for achieving a solution can be taken.

maybe
move
hand
model
section
to here?

Denavit
Harten-
berg
formu-
mula-
tion for
coordi-
nate
System
defi-
nition
and

2.2.1 Forward Kinematics

Forward Kinematics (FK) uses the knowledge of the new angles and positions after the application of known transformations to the kinematic chain. The data of the *joints* and *links* between the *root* and the *end effector* is then used to solve the problem of finding the *end effector's* position.

We can denote the existing end effectors relative position to an origin as s_1, \dots, s_k . The s_i position is the result of the combination of all the joint angles in the corresponding kinematic chain. Respectively, we define the target position of the end effectors as t_1, \dots, t_i , with t_i being the target position for the end effector s_i . The required positional change for the end effector can now be described as $e_i = t_i - s_i$. In systems with more than one end effector, like our hand system, the components can be written as vectors.

$$\begin{aligned}\vec{s} &= (s_1, \dots, s_n)^T \\ \vec{t} &= (t_1, \dots, t_n)^T \\ \vec{e} &= \vec{t} - \vec{s}\end{aligned}\tag{2.5}$$

As the vector components of \vec{s} are results of the chain joint angles $\theta_1, \dots, \theta_n$ and therefore are effected by them, we define

$$\begin{aligned}\vec{s}_i &= f_i(\theta) \\ \vec{s} &= f(\theta)\end{aligned}\tag{2.6}$$

With θ being the column vector $\theta = (\theta_1, \dots, \theta_n)^T$. The second vector equation displayed in (2.6) is also called the *Forward Kinematics*(FK) solution.

The advantage of an FK solution is that there is always an unique solution to the problem. In consequence, this approach is commonly used in the field of robotics, where the information on the chain elements is easily available.

The tracking of the human hand and all of its chain components is rather complicated. Therefore a solution which takes a known position of the *end effektor* and calculates the parameters for the rest of the chain would be more desirable.

2.2.2 Inverse Kinematics

Inverse Kinematics (IK) is a method for computing the posture via estimating each individual degree of freedom in order to satisfy a given task [1]

The concept of *Inverse Kinematics* (IK) already describes its principle in its name. It takes the reversed approach in comparison to the FK principle in chapter 2.2.1. Instead of knowing the states of the chain elements and calculating the resulting position of the

end effector, we take the position of the end effector and try to retrieve the possible states of the other chain elements.

$$\theta = f^{-1}(\vec{s}_d) \quad (2.7)$$

The result of this equation is the vector θ for which the values of \vec{s} coincide to the desired configuration \vec{s}_d . In the case of an optimal result, this configuration would have the same position values as the target positions.

The main problem with this method occurs in the calculation of the f^{-1} function, due to it being a highly non linear operator which is not easily invertible. The approaches that [1] describes to counter this problem will be displayed later on in this chapter.

In contrary to having a unique solution with the FK approach, the IK approach can end at the point of not finding a suitable solution. Figure 2.3 displays three possible outcomes for the IK approach.

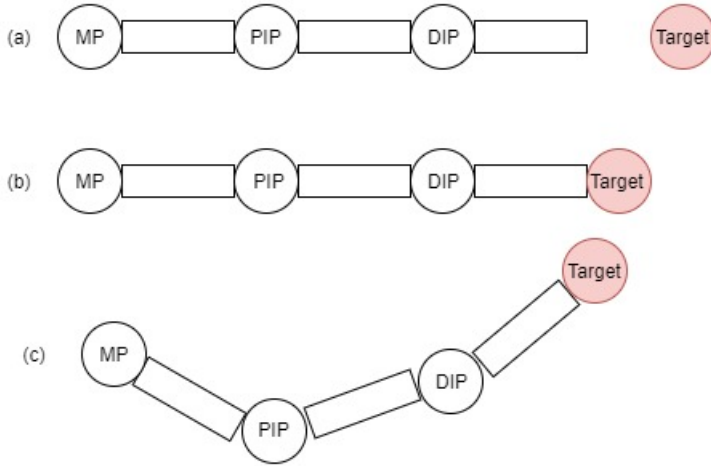


Figure 2.3: Possible solution for an IK problem of a human finger:

- (a) The given target position of the end effector can not be reached. (b) The given target can only be reached by one solution. (c) The target position can be reached with multiple different solutions.

2.2.3 Jacobian Inverse

One common approach to solve the IK problem is the utilization of a Jacobian Matrix and an iterative calculation process. This matrix contains the partial derivatives of the chain systems relative to the end effector \mathbf{s} . When using the Jacobian, a linear approximation of the IK problem will be applied for solving. The approximation's components model the end effector's relative movement to changes in transitions of the systems link translations and joint angles. Therefore, the resulting function is dependant on the joint angles θ values and can be defined as

$$J(\theta)_{ij} = \left(\frac{\partial \mathbf{s}_i}{\partial \theta_j} \right)_{ij} \quad (2.8)$$

with $i=1,\dots,k$ and $j=1,\dots,n$. Further readings on methods for the calculation of Jacobian matrices can be found in [33]. Based on definition (2.8), an entry for the j -th rotational joint would be calculated as follows:

$$\frac{\partial \mathbf{s}_i}{\partial \theta_j} = \mathbf{v}_j \times (\mathbf{s}_i - \mathbf{p}_j) \quad (2.9)$$

where \mathbf{p}_j is the position of the joint, and \mathbf{v}_j is the unit vector pointing along the current axis of rotation for the joint. Taking the derivate of definition (2.6) with respect to time gives the basic equation for forward kinematics that describes the velocities of the end effectors:

$$\dot{\mathbf{s}} = \mathbf{J}(\theta)\dot{\theta} \quad (2.10)$$

Now having all the values for the angles, the *end effector* position and the target positions, we can compute the resulting Jacobian matrix. Thereafter we seek an update value $\Delta\theta$ for incrementing the current joint values:

$$\theta_{new} = \theta_{curr} + \Delta\theta \quad (2.11)$$

The idea here is that the chosen value for $\Delta\theta$ should lead to the resulting $\Delta\vec{s}$ being approximately equal to \vec{e} from (2.5). The \vec{e} can be approximated by:

$$\Delta\vec{s} \approx \mathbf{J}(\theta_{curr})\Delta\theta \quad (2.12)$$

Using this approximation we can reformulate the FK problem as $\vec{e} = \mathbf{J}\Delta\theta$ and therefore our inverse kinematics problem from 2.7 can be expressed as $\Delta\theta = \mathbf{J}^{-1}\vec{e}$.

The problem we run into with this solution is the construction of the inverse Jacobian matrix. The Jacobian \mathbf{J} may not be square or invertible. In the case of it being invertible, the result may only work inferiorly because of it being nearly singular. Being singular means that no change in joint angle values may achieve the desired end effector position as an outcome.

Jacobian Transpose

One approach to calculating the value of $\Delta\theta$ without having to calculate the inverse of \mathbf{J} is done by replacing the inverse with the transpose of \mathbf{J} .

$$\Delta\theta = \alpha \mathbf{J}^T \vec{e} \quad (2.13)$$

Of course the transpose and the inverse of \mathbf{J} are not the same thing. When using the theorems displayed in [8, 45] we can show that:

$$\langle \mathbf{J} \mathbf{J}^T \vec{e}, \vec{e} \rangle = \langle \mathbf{J}^T \vec{e}, \mathbf{J}^T \vec{e} \rangle = \|\mathbf{J}^T \vec{e}\|^2 \geq 0 \quad (2.14)$$

Under a sufficiently small $\alpha > 0$ the updated angles from 2.11 will change the end effector positions by approximately $\alpha J J^T \vec{e}$. They also state that the value of α can be calculated by minimising the new value of the error vector \vec{e} after each update.

$$\alpha = \frac{\langle \vec{e}, J J^T \vec{e} \rangle}{\langle J J^T \vec{e}, J J^T \vec{e} \rangle} \quad (2.15)$$

Jacobian Pseudo Inverse

Instead of calculating the normal inverse of the Jacoboian, which can lead to the problems described before, we can use the so called *pseudo-inverse*[7] for the calculation. The *pseudoinverse* is defined for all matrices J , even ones which are not square or not of full row rank.

$$\Delta \theta = J^\dagger \vec{e} \quad (2.16)$$

The *pseudoinverse* represents the best possible solution for $J \Delta \theta = \vec{e}$ in respect to least squares, but does suffer from instability near singularities. It has the property that the matrix $(I - J^\dagger J)$ performs a projection onto the nullspace of J . Therefore, for all vectors φ , $J(I - J^\dagger J)\varphi = 0$. Therefore $\Delta \theta$ can be set by

$$\Delta \theta = J^\dagger \vec{e} + (I - J^\dagger J)\varphi \quad (2.17)$$

for any vector φ and still obtain a value for $\Delta \theta$ which minimises the value $J \Delta \theta - \vec{e}$.

The pseudoinverse of J can be derived as follows:

$$\begin{aligned} J^\dagger &= (J^T J)^{-1} J^T \\ \Delta \theta &= (J^T J)^{-1} J^T \vec{e} \\ (J^T J) \Delta \theta &= J^T \vec{e} \end{aligned} \quad (2.18)$$

Using $\vec{y} = J^T \vec{e}$, we can now solve the equation $(J^T J) \Delta \theta = \vec{y}$

Singular Value Decomposition

The usage of a *singular value decomposition* is another method of calculating the *pseudoinverse* of a jacobian matrix. A singular value decomposition of J consists of expressing J in the form

$$J = U D V^T \quad (2.19)$$

For an $m \times n$ Jacobian matrix, U is $m \times m$ and the columns are the eigenvectors of $J J^T$. D is $m \times n$, and the entries are zero except along the diagonal where $d_{ii} = \sigma_i$ with σ_i beeing the singular value and $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_m \geq 0$. Singular values are the nonnegative square roots of the eigenvalues of $J J^T$ and $J^T J$. V is $n \times n$ and the columns are the eigenvectors of $J^T J$. The construction of the pseudonverse is done as follows:

$$J^\dagger = V D^\dagger U^T \quad (2.20)$$

The pseudo-inverse of the diagonal matrix \mathbf{D} is obtained by replacing each positive diagonal entry with it's reciprocal [15].

2.2.4 Damped Least Squares

The *Dampened Least Squares* method is applied for inverse kinematics as it avoids the singularity problems that can occur when using the pseudoinverse. It further provides a numerical stable method for selecting $\Delta\theta$. First occurrences for inverse kinematics can be found in [30, 42]. Instead of finding the minimum for $\Delta\theta$ as a best seolution for the equation displayed in 2.5, the goal of this methodd is to find the value that minimizes the following quantity with $\lambda \in \mathbb{R}$ as a dampening factor.

$$\| J\Delta\theta - \vec{e} \|^2 + \lambda^2 \| \Delta\theta \|^2 \quad (2.21)$$

which can be rewritten to the corresponding

$$(\mathbf{J}^T \mathbf{J} + \lambda^2 \mathbf{I}) \Delta\theta = \mathbf{J}^T \vec{e} \quad (2.22)$$

and therefore

$$\Delta\theta = (\mathbf{J}^T \mathbf{J} + \lambda^2 \mathbf{I})^{-1} \mathbf{J}^T \vec{e} \quad (2.23)$$

The selection of a correct λ value is essential for this method, small values of λ give accurate solutions but low robustness to the occurrence of singular and near-singular configurations, while large values of λ result in low tracking accuracy even when a feasible and accurate solution would be possible[1, 5].

2.2.5 Newton Methods

The Newton family of methods is based on a second order Taylor series expansion of the object function $f(x)$:

$$f(x + \sigma) = f(x) + [\nabla f(x)]^T \sigma + \frac{1}{2} \sigma^T H_f(x) \sigma \quad (2.24)$$

where $H_f(x)$ is the Hessian matrix. The downside of this approach is that the calculation of the Hessian matrix bares high computational cost for each iteration as a result of its complexity. Like the approximation of the Jacobian inverse, several attempts to lower the computational cost have benn made by utilizing an approximation of the Hessian matrix based on a function gradient value.

Since the Newton methods are posed as a minimisation problem, they return smooth motion without erratic discontinuities. The Newton methods also have the advantage that they do not suffer from singularity problems, such as that which occurs when finding the Jacobian Inverse.

2.2.6 Sequential Monte Carlo Method

Sequential Monte Carlo Methods (SMCM) Methods are a relative young approach in solving IK problems. Courty and Arnaud [6] proposed such a technique, whose main advantage is the usage of a sampling approach. When doing so, the IK problem can be solved through the usage of direct kinematics, avoiding the numerical inversion of the forward operator.

add
other
missing
meth-
ods,
maybe
only as
further
are

2.2.7 FABRIK Algorithm

A more recent approach in the kinematics field is the **FABRIK** algorithm proposed by Aristidou and Lasenby[2]. As shown in section 2.2.2, these approaches all depend on computational intensive matrix operation like calculating an inverse and may have problems with matrix singularity.

The **FABRIK** algorithm does not depend on these matrix operations as it solves for the position of a point on a line to retrieve the new joint positions. This is done in a forward and also inverse solving approach, iterating these steps until the calculated position converges towards the target position from the tracking data.

Figure 2.4 illustrates the steps that are contained in one iteration step. The chain joints are denoted as \mathbf{p}_i with the distance \mathbf{d}_i being $|\mathbf{p}_{i+1} - \mathbf{p}_i|$. The target point for the end effector is denoted as \mathbf{t} . Step (a) displays the starting point of the iteration. The joint positions are taken from either a previous iteration or from an initial calibration.

But before calculations can begin, the algorithm has to check whether the intended target point \mathbf{t} is reachable for the end effector. This is done by measuring the distance between the root of the kinematic chain and the target point \mathbf{t} . This value is then compared with the sum of the distances \mathbf{d}_i .

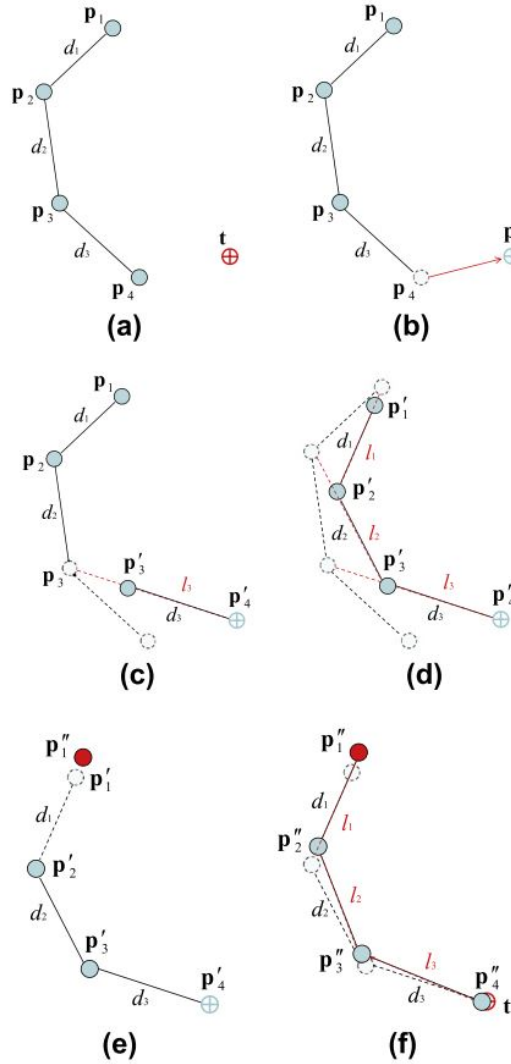


Figure 2.4: Forward and backward calculation steps for one iteration of the **FABRIK** algorithm. (a) initial position of the system. (b) End effector is moved to target position. (c) Determine position of next joint on constructed line. (d) repeat until root is reached. (e) move root joint to initial position. (f) repeat calculation in reverse direction [2]

$$dist_{t,d_1} < \sum_{k=1}^i d_i \quad (2.25)$$

The inverse calculation step is the first step, which is displayed in **(b)** and **(c)**. The calculation is started at the end effector, moving to the root of the chain. If the summed distance is greater, then the **t** is within the reach of the system and the calculation can continue, otherwise the calculation has to be aborted and the error has to be managed otherwise. Assuming this requirement to be met, we can now begin with the first calculation. Therefore we assume that the new position \mathbf{p}'_n with $n=4,\dots,1$ is equal to **t**.

$$\mathbf{p}'_n = \mathbf{t} \quad (2.26)$$

From this new point, we can construct a line that goes through \mathbf{p}'_n and \mathbf{p}_{n-1} .

$$\begin{aligned} A &= \mathbf{p}'_n \\ B &= \mathbf{p}_{n-1} \\ \mathbf{l}_{n-1} &= \overline{AB} \end{aligned} \quad (2.27)$$

The resulting position of the new \mathbf{p}'_{n-1} point is located in this line with the distance of \mathbf{d}_{n-1} from \mathbf{p}'_n (see **(c)**).

$$\mathbf{p}'_{n-1} = \mathbf{p}'_n + \left(\frac{\overline{AB}}{|\overline{AB}|} \cdot \mathbf{d}_{n-1} \right) \quad (2.28)$$

Consecutively, this is done with the remaining joints until the root joint is reached (see **(d)**). This finishes the first half of the iteration step. With the calculated positions, we now perform a forward calculation, starting from the root until we reach the end effector. Since the root of the system normally does not move from its initial position, we have to reset the root joint to this value before starting to calculate the new positions of the subsequent joints (see **(e)**).

Analogous to the procedure in the inverse step, we construct the lines between the points and determine the new position values of the joints. The end result of this step is shown in **(f)**. At this point, we can decide if the result position of the end effector is appropriate in comparison to the value of **t**. A simple threshold value for this case could be the position difference between these two points.

constraints
adding
for the
algo-
rithm

review
of algo-
rithm
capacili-
ties

3 Tracking in real space

The previous sections provided information on the structure of the human hand and how to represent it in the digital space. To apply the algorithms presented in Section 2.2 to our digital hand model, we need input data from our real world representative.

3.1 General tracking technologies

The methods for gaining positional data can be roughly categorized into two major groups, glove based methods and vision based methods. Glove based methods have already been in development since the 1980's [4] and have since then resulted in several solution attempts. Sturman and Zeltzer gave a survey on the existing tracking methods in their paper [40].

They distinguish between two areas of tracking, first the 3D positional tracking of the hand (and also other bodyparts) without regard to the hands shape and secondly the tracking of the hand shape with glove technologies.

These tracking technologies presented are still applied today in modern tracking solutions [37, 44]. They account for solutions based in optical tracking based on marker detection, magnetic detection via measurements of an artificial magnetic field[35] and acoustic measurements via triangulation of ultrasonic pings.

3.1.1 Optical tracking

The components for an optical tracking systems are several cameras for object detection and some kind of tracking characteristic of the object to be tracked. These characteristics can be either artificially applied ones like active flashing infrared LED on key tracking positions of the body or infrared reflective markers.

A series of cameras positioned around the tracking subject will then track these markers inside their visual fields. The second method uses a single camera to capture the silhouette image of the subject, which is analyzed to determine positions of the various parts of the body and user gestures.

The image data is supplied to special software which correlates the marker positions in the multiple viewpoints and uses the different lens perspectives to calculate a 3D coordinate for each marker. These image interpretation and correlation tasks require computationally costly operations. The marker tracking is also prone to errors through variation in lighting of the scene, material reflection properties and also marker occlusion as the trackers are moved. Also most of the systems rely on several tracking cameras for

3 Tracking in real space

a complete coverage of the tracking space. This leads to a higher system complexity in terms of setup and calibration.

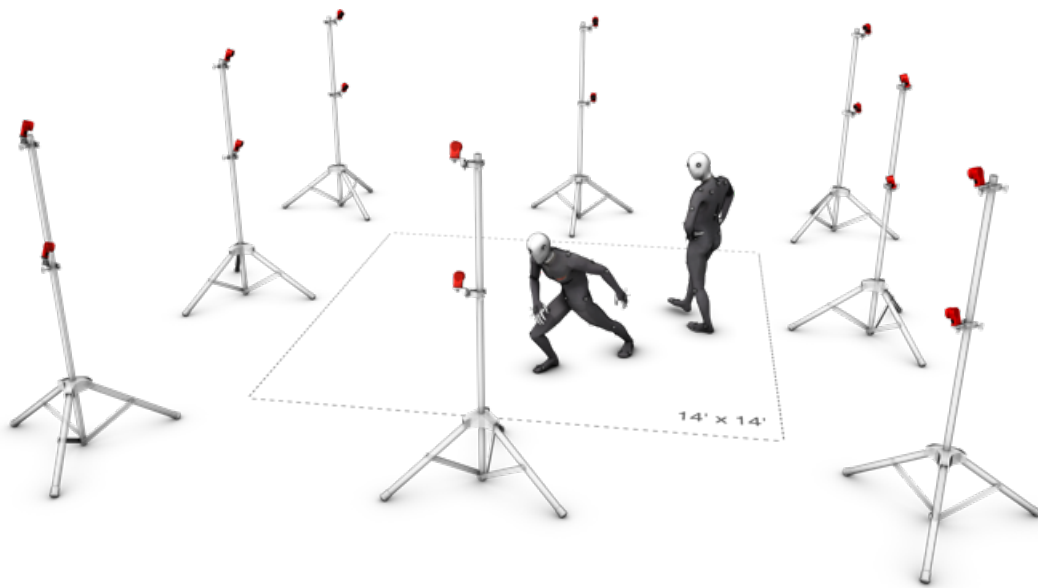


Figure 3.1: Example for an optical tracking system with passive infrared reflector markers [32]

3.1.2 Magnetic tracking

The usage of a magnetic field for position tracking is a relatively uncomplicated technique. The earth already provides us with a magnetic field to orient on. Similar to the optical trackers, magnet field sensors can be placed at key tracking positions. These sensors measure field strength in 3 orthogonally oriented axes to produce a 3D vector of the unit's orientation with respect to the excitation. As the earth's magnetic field is prone to changes based on geographical location, data corrections have to be applied to the measurements.

Another solution is to generate a local magnetic field via a multicoil source unit. The source unit coils are energized in sequence and the corresponding magnetic field vector is measured in the sensor unit. With three such excitations, you can estimate the position and orientation of the sensor unit with respect to the source unit.

The downside of this technology is that ferromagnetic and conductive material in the surrounding environment will have an effect on the generated magnetic field. These distortion effects will form small unwanted secondary source units through the induction of small Foucault currents by the main source's magnetic field. Magnetic fields have the property of having an inverse cubic falloff as a function of distance from the source, which limits the range of operation for the system. Position resolution in the radial direction from source to sensor depends on the gradient of the magnetic field strength, and thus the positional jitter grows as the fourth power of the separation distance.

In comparison to other tracking technologies, the magnetic field tracking solution has the convenience of not suffering from line of sight problems from tracker occlusion. The

magnetic fields are capable of passing through the human body. Also the sensor size for measuring magnetic fields is rather small, giving the trackers a small volume. Furthermore, only one source unit is needed for the tracking of multiple sensor units.

3.1.3 *Acoustic tracking*

The principles of acoustic tracking are very similar to those of the optic tracking technologies. Instead of using lightwaves, the systems utilize acoustic pulses of ultrasonic wavelengths to time the time of flight between emitter and sensor for range measurement. To get a good measuring result from the systems, the used acoustic transducers have to be as omnidirectional as possible, so that the signal can be detected no matter how the emitter is positioned or oriented in the tracking volume. For the speakers to achieve a wide beam width, their size has to be small.

To be able to build the microphones into the tracker, they can only have active surfaces a few millimeters in diameter. This leads to a reduction in range as the efficiency of acoustic transducers is proportional to the active surface area. Also acoustic systems can have problems with ambient noises occluding the signal. This becomes even more critical when using such a system outdoors.

Soundwaves travel at a much slower speed than lightwaves which brings benefits and downsides with it. Soundwaves can be reflected from objects, producing so echos which arrive at the receiving sensor at a later point in time. Here the slower speed can be beneficial as we can await the first sound occurrence to arrive at the sensor and filter out all later reflections from the data. The reflections of the previous pulse also have to be subsided before a new measurement can be made, lowering the update rates of the system. The air the soundwaves travel through is also a limiting factor as humidity, air pressure and air currents can influence the travelling soundwaves. In comparison to the optical systems, the acoustic systems are not as prone to occlusion errors since soundwaves have a better ability to bend around obstacles than lightwaves.

Most of these downsides can be addressed with a combination of these systems with another form of tracking like in [13].

3.2 Hand tracking Systems

In comparison to the tracking of the whole human body, the tracking of the human hands with their maximum of 27 DoF's each in a small space is rather difficult to handle.

The systems that try to achieve this have to encounter several difficulties. Self occlusion also plays a great role in the system design, especially when working with only one tracking optical system for reduced complexity. Also these systems need to maintain a certain amount of processing speed for achieving a fluent reproduction of the captured motion. This demands the capability of processing large amounts of data in very short intervals. Most prototype testing for the described systems will be done in a controlled

environment where the background is known. For a more widespread use, these systems need to be capable of registering the hand on an unrestricted background, which may have a wide range of patterns, color and lighting differences. Also human hand motion itself is quite fast, resulting in a higher framerate demand for the tracking cameras. The following section will describe different approaches to solve the aforementioned difficulties.

3.2.1 Sensor based

Tracking systems that make use of sensors mounted to the hand via straps or a glove have already been in use since the 1970's. First prototype glove systems included the Sayre Glove[41] and the Digital Entry Data Glove[16].

The Sayre glove is based off an idea of Richard Sayre. The system utilizes a light source and a photo cell which are connected via a flexible tube. These components are mounted along each finger of the hand. The light that passes through the tube is influenced by the bending angle of the tube which corresponds to the finger bending. A large bending angle of the finger induces a larger bending angle of the tube, reducing the intensity of the light that reaches the photodiode. The resulting voltage in the photocell can then be mapped to a specific bending angle of the finger.

The Digital Entry Data Glove was patented by Gary Grimes in 1983. Instead of using only one type of sensors for measurement, this glove based system used several sensor types for different measurements. Touch or proximity sensors were used for determining whether the user's thumb was touching another part of the hand or fingers. To measure the flexion of the joints in the thumb, index, and little finger four "knuckle-bend sensors" were used. To get measurements for the tilting of the hand in respect to the horizontal plane two tilt sensors were mounted. Finally two inertial sensors for measuring the twisting of the forearm and the flexing of the wrist were utilized. This glove was intended for creating "alphanumeric characters" from hand positions. Recognition of hand signs was performed via a hard-wired circuitry, which mapped 80 unique combinations of sensor readings to a subset of the 96 printable ASCII characters.



Figure 3.2: AcceleGlove system with the described sensor positions [18]

These early systems only provided a limited amount of sensors and were rather impractical in use. As they were developed to serve very specific applications, they were used only briefly, and never commercialized.

Newer developments in this sector include [18, 22, 27]. The *AcceleGlove* presented by [18] consists of six dual-axis accelerometers, mounted on the fingers and the back of the palm, reporting position with respect to the gravitational vector. Sensors are placed

on the back of the middle phalanges, on the back of the distal phalange of the thumb, and on the back of the palm.

Kuroda et al [22] introduced the *StrinGlov*, which obtains full degrees of freedom of human hand using 24 Inductcoders and 9 contact sensors, and encodes hand postures into posture codes on its own DSP. The bending angles of the fingers are measured with the inductcoders. These sensors relate the finger movement to a change in magnetic flux induced by the movement of the sensor parts that are attached to the finger. The sensor functionality is similar to the functionality of the light sensors from [41]. The 9 contact sensors, also based on magnetic fields, are put on the fingertips and on the inside of the hand to be able to measure contact between two fingertips or the fingertips and the hand. The system furthermore benefits from simple structure, which leads to low production cost. Majeau et al [27] proposed a system that uses optical flexion sensors to determine the bending angles of the fingers. The optical flexion sensors consist of a LED, a Photodetector and an optical fibre. The LED emits light which travels through the optical fibres and the intensity that results as the output from the fibre end is measured by the Photosensor. This measuring principle also corresponds to the principle of [41]. Furthermore the system is also capable of measuring the abduction of two fingers. Here the LED is mounted to one finger and the detector on the other. The optical fiber is run from the LED down the finger to the knucklebase and then up to the detector. When abducting the two fingers, the angle at the knucklebase changes, resulting in an intensity change at the detector through the loss of intensity at the curvature of the optic fibre. The change in intensity of both measuring techniques can be mapped to corresponding hand movements.

Further readings on sensor based hand tracking systems can be found in [10, 40]

Find 1-2 newer systems for comparison

3.2.2 Appearance Based - Optical marker

Optical approaches use properties of the human hand to estimate the current position. The evaluation of the hand pose is usually split up into two steps. The first one is the registration of the "real" hand.

This is mostly done by at least one camera which is aimed at the hand, supplying a "continuous" data stream to the evaluation Software. In the second step, the evaluation software then tries to find key spots in the provided images. This data is then used for the search of the matching digital pose. The definition of these key spots can be achieved by several techniques. An example is the usage of some kind of marking for finger segments as displayed in



Figure 3.3: Color glove setup used by Wang and Popovic [43]

[11, 14, 43]. In the method described by Wang and Popovic [43], a specially colored glove is used. The glove is segmented into ten segments, colored randomly from a pool of ten distinct colors (see Figure 3.3).

The pattern is created by selecting twenty seed triangles on a 3-D hand model that are maximally distant from each other. The remaining triangles are assigned into patches based on their distance to each of these seeds. Each patch is assigned one of the ten colors at random. The jagged boundaries between the patches are artifacts from the low triangle count of the used hand model. Algorithms that use other characteristics for tracking like texture or shading [23] rely on an accurate pose from a previous frame to constrain the pose search in the current frame. When using bare hand pose estimation two different hand poses can map to similar images. This can lead to an inaccurate pose estimation and the breakdown of the algorithm if the wrong estimation is made. The benefit of the color glove is that the unique patch pattern, hand poses always map to distinguishable images, simplifying the search process. Therefore it can effectively recover itself at each frame without the need of a previous frame. With the colored glove as a tracking feature, the images from the camera are prepared for a database sampling step. The database that was used consisted of 18,000 finger configurations. A distance metric between two configurations was defined as the root mean square (RMS) error between the vertex positions of the corresponding skinned hand models. With this distance metric, a low dispersion low-dispersion sampling was used to draw a uniform set of samples from the overcomplete collection of finger configurations. The selected configurations are rendered at various 3-D orientations using a (synthetic) hand model at a fixed position from the (virtual) camera. The camera image was compared then compared to the database images. The comparison is done via a nearest neighbor lookup using a Hausdorff-like distance [19] and penalizing the distance to the closest pixel of the same color in the other image. The resulting pose from the database can then be applied to the digital hand model. The method presented by Fredriksson and Ryen [14] also uses a color coded glove. In contrast to the fully colored glove described before, the used glove only has colored fingertips, color markers for the palm and a colored band for the wrist.

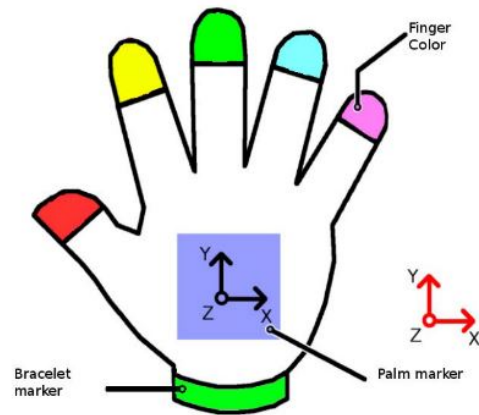


Figure 3.4: Color glove setup used by Fredriksson and Ryen

[14]

The palm markers and the wrist band are used to retrieve the 3D position of the hand. The wrist marker is furthermore used to define the bounding box of the tracking area in a calibration process. Defining a bounding box greatly simplifies further image analysis operations by taking away a large amount of unneeded image data. After determining position and orientation of the hand in 3D by utilizing wrist band and palm marker position,

the system tracks the grip angle and the lateral tilt angle for each finger. The grip angle is defined as the curling of the finger towards the palm. The second angle is defined as the lateral tilt angle which measures the spread of the fingers. The grip angle is calculated by comparing the density of finger pixels around an estimated knuckle line. From the density, an estimation for the finger's Y position around an origin position at the knuckle's base is made. This value is then transformed into an angle value using an inverse tangent operation.

One downside of this tracking method is that it does not yet incorporate the movement of the thumb. The thumb has other movement possibilities and therefore needs a different algorithmic approach. Also the approach only focusses on the tracking of the hand data and does not yet implement a digital counterpart like the previously described system.

3.2.3 Model Based - Image analysis

Hand tracking with image analysis of optical marker positions has the flaw that these markers have to be put onto the hand in some kind of fashion. Using a glove for this purpose imposes just a little bit of inconvenience, but has the drawback of not fitting every handshape equally and also having to deal with hygiene when switching between users. The most natural way of tracking the human hand would be by simply using the properties of the human hand like skin color and shape for tracking. This would remove the need for any artificially added tracking features and have the greatest amount of comfort for the human user. A full overview of the technological advances in this area can be found in [28, 29]. This paper will only highlight some of the main topics displayed. As already explained before, the human hand motion leads to a large variety of shapes with many self-occlusions, which becomes even more critical when trying to reproduce the human hand pose via Computer vision. Also the separation between background image information and the relevant hand information is a major topic, which is usually also the first step in the algorithmic process. An established method for the extraction of background data was introduced by Stauffer and Grimson [39]. It uses the idea of representing each pixel value as a mixture of Gaussian functions (*MoG*) and updating these pixels during runtime with new Gaussian functions. The algorithm stores mean, variance, and likelihood value, based on previous values, for each pixel location and distribution.

A new input is compared to the stored mean values for each distribution at this location. A result that is less than a constant times the variance declares that this pixel is part of that distribution and it is updated accordingly. For a higher result, a new distribution is created which replaces the least probable distribution. A new input belongs to the background if the distribution that it is part of is one of the most likely distributions. A foreground Pixel that keeps the same color over time will slowly be incorporated into the background. The decision rule, whether a new pixel belongs to an existing distribution

or not, can be written as [21]:

$$if (|P_{new} - P_{mean}^d| \leq KP_{std}^d) \quad then \quad P_{new} \in Distribution \, d \quad (3.1)$$

where P_{new} is the new pixel value, P_{mean}^d and P_{std}^d are the stored mean value and standard deviation of distribution d , and K is a constant. The convenience of this method is that it is also able to subtract background information from uncontrolled environments, making it possible for systems to work outside of lab conditions. After the segmentation of foreground and background is achieved, the hand pose has to be retrieved from the left over image data. Solutions for retrieving a full set of DOF as hand pose estimation data can be split into two main categories[12], *Model-based tracking* and *Single frame pose estimation*.

Model-based tracking refers to a top-down tracking method based on parametric models of the 3D hand shape and its kinematic structure. The root idea here is to execute a search at each frame of an image sequence to find the pose of the shape model that best matches the features extracted from the image(s). A prediction based on previous motion and dynamics data from the object is set up as a base for the search. Optimization features can be incorporated, like employing a local search around the calculated prediction to produce a single best estimate at each frame. The downside of this method represents itself in the weakness of accumulating errors over longer runtime periods due to occlusions and complexity of hand motion. These can be overcome by keeping track of multiple outcome hypotheses at each frame for error reduction.

*Single frame pose estimation*in contrast to the previous method, does not utilize assumption on time coherence. Another motivation for this approach is the rapid motion of the hand and fingers. Images of consecutive frames can be very different, making time coherence assumptions useless.

4 Digital hand models

After retrieving all needed data from the real world counterpart and calculation for pose estimation is finished, the returned solution has to be displayed in the digital world. Therefore a digital hand model is needed to represent the calculated data. Modern day computer animation techniques make it possible to create nearly photorealistic digital replicas of human skin wrapped onto anatomically precisely modeled bodyparts. These highly complex models acquire a lot of calculation resources and are mostly not appropriate for a realtime rendering approach.

When adapting calculation data to digital hand models, those of lesser complexity are preferred, sacrificing the quality of the resulting output for speed and more fluent motion results.

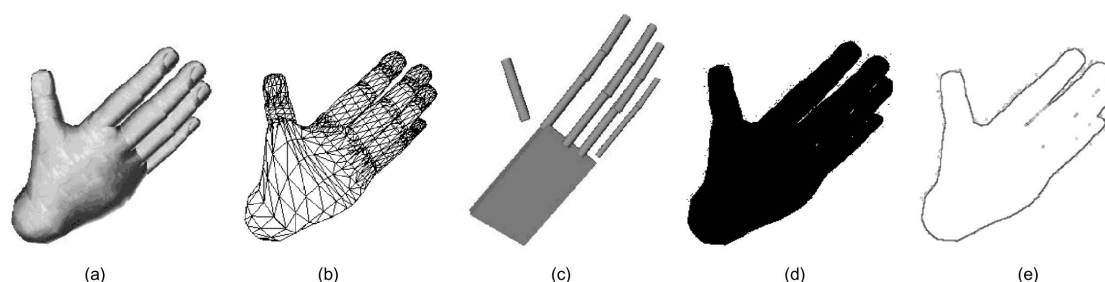


Figure 4.1: Pavlovic, Sharma et al: Hand models. Different hand models can be used to represent the same hand posture. (a) 3D Textured volumetric model. (b) 3D wireframe volumetric model. (c) 3D skeletal model. (d) Binary silhouette. (e) Contour.

[34, p. 682]

Figure 4.1 shows different model representations of the same hand posture. The models of **(a)** and **(b)** represent the aforementioned types of models that would be used in modern computer graphics for displaying hand models realistically.

These models are based on complex 3D surfaces like *NURBS* (nonuniform rational B-splines) which are wrapped around an underlying skeleton. The wireframe model of **(b)** give a good overview of how many vertex calculation have to be made for the surface in each frame, causing these models to hardly run in realtime.

In applications where the 3D orientation of each individual finger is not necessary for processing and the general silhouette is sufficient (e.g general hand position tracking or sign language detection), even more reduced representation forms like **(d)** and **(e)** can be chosen. The model displayed in **(c)** is a simplification of the human hand structure with geometric primitives like cylinders and boxes. These geometric primitives have the virtue of being much easier to describe than a *NURBS* model. A cylinder for example

4 *Digital hand models*

is fully described by its height and its radius under the assumption that the position and orientation data is generally needed for all the described models. This type of model is also referred to as *skeletal models* as they mostly try to reproduce the structure of the human hand as described in Section 2.1

5 System prerequisites

Before elaborating a final system conception, some prerequisites have to be defined.

System components to be defined:

- Tracking Hardware
- Image analysis
- Inverse kinematics algorithm
- Handmodel
- Display hardware

5.1 Display

Before analyzing the other components of the system, the final display hardware properties have to be defined as these are the determining factor for all other components. Since the goal of hand and object tracking is mostly immersive when using a virtual reality headset rather than a normal display, these do not need to be taken into consideration.

Another device category which may be in the focus of interest are current mobile device Platforms like *Google cardboard* or the *Samsung Gear* platform. In terms of creating a low cost consumer grade solution, these devices should be the first category to look at, as they are more widely available and do not need extra hardware except the smartphone. Most modern devices have native support of WebGL in the Browser, making it fairly easy to display VR Content. They also come with the convenience of already having a set of integrated sensors for position and rotation integrated. The downside of this device class appears when taking a look at the hardware specifications of these devices. Although the newest high tier consumer grade devices do have a good performance output for their size, but still no comparison for a Desktop Setup. Also these devices, as they are first of all mobile phones, lack in terms of connection ports, for wire connections, making it necessary to mostly communicate over wireless protocols. This causes a growth in Infrastructure (Server, Hotspot, etc.) and also introduces physical delay through signal transmission.

After these first elementary steps, only pure VR-Headsets like the *HTC Vive* or *Oculus Rift* should be the main focus. These headsets also provide the capabilities of motion tracking while utilizing the processing capabilities of a dedicated GPU and highpower

get device specs for current phones

get refs for devices

CPU capabilities of modern desktop computers. Through these, they provide a variety of connection ports for wired connections, eliminating the need for wireless transmissions. In terms of cost, these devices do lie on the more costly side as you need the headset itself and a hardware setup which is capable of providing the output framerate needed. But with the latest generations of GPU's and CPU's, reaching this goal was made possible for the normal consumer.

The *HTC Vive* is able to display 90 Images per second on both of its displays with each having a resolution of 1080x1200 pixels. The duration of one frame is therefore around 10 ms. This is the timeframe in which the rest of the components have to supply the image data, do the image analysis to retrieve the tracking data, recalculate the IK model, calculate the object position and render the whole scene.

Oculus
specs

ref

5.2 Tracking Hardware

As explained in the previous chapters, the motion of the human hand can be complex and in some cases really fast. Therefore, the optical tracking hardware should ideally be able to record images with the same or higher frequency as the display medium that is to be used. Prosumer and professional grade cameras are theoretically able to produce these kind of Framerates (60 fps and above), but most of the time store these high framerate videos directly to a hard drive rather than broadcasting them somewhere as most of the current day devices are not able to display such high framerates yet. Also the hardware that is needed to record such specific high framerates is relatively expensive. Consumer grade cameras like the GoPro that are affordable, can record at a Framerate of 120 fps but the live transmission of this material has a time lack of several frames.

When trying to utilize low cost hardware the limitation of recording and sending a video signal from a camera to a processing hardware will therefore be limited to 60fps. This process would also introduce further latency on the transport and would force a further reduction of framerate to satisfy the timeframe given by the display framerate.

Another option that can be taken into consideration is to completely eliminate the need to transport the image data from camera to another device. This can be achieved when the camera and the computational hardware are located on the same hardware.

The *Raspberry Pi 3* microcomputer, which comes from the internet of things world provides this capability. It combines a for its size quite powerful computation capacity with an onboard hardware connector for a camera. The camera is able to record in a FullHD resolution at 60fps and at lower resolutions up to 90fps. The camera module is directly connected on the the chipboard, providing direct access for further processing.

For a complete 3 Dimension space tracking, a one camera setup is not sufficient as it lack the data for the depth position of the object. To get this data, further hardware is needed. One option would be the utilization of a time of light based depth sensing

system, but these system do come at a rather high entry level price.

prices
and lit

Another option could be the tracking pods that HTC offers as an expansion to its vive system. These trackers could provide 3d positional Data but with the downside of being relatively bulky. This would hinder the movement space of the hand and the tracker would have to be positioned on the arm as near as possible to the wrist. This would result in the position of only the wrist and not providing data on each individual finger of the hand.

As the hardware setup of the Raspberry Pi with the camera is relatively easy and cost efficient, the usage of a second Raspberry Pi with a camera is good leverage point. The two cameras can be setup to create a stereoscopic Camera setup which provides information about depth position through the disparity of the two camera images. Furthermore the image operations that need to be independently applied to each of the cameras frames can be done on the two separate device, eliminating the need for sending data traffic with image information through the network for postprocessing.

Downside of this method is that with all Stereoscopic camera Setups, the two cameras need to have some kind of frame synchronization to get the right two frames for evaluation.

As the setup is not only meant for tracking the hand, but also for tracking objects in the tracking space, the system needs to be capable to do this task as well. Since the computational power of the raspberry is limited at some point, it is to be evaluated if a separate hardware component is needed for the object tracking or if the tracking can be integrated into the tracking procedure of the hand markers. Another question that has to be investigated would be, how important the depth position for the to be tracked object really is as they will not be floating in space while being tracked. Maybe a combination of hand position data and object orientation can already function as a suitable tracking method.

5.3 image analysis

The images recorded by the camera system need to be processed further down the line to get the positional data of selected hand features. The most common library to do such kinds of image processing is the opencv library which is originally written in C and was ported into several other programming languages. It has built in features for camera calibration, color and object detection as well as several image optimization features that might be helpful for the processing.

5.4 inverse kinematics

The inverse kinematics model is the second essential part of the setup after the tracking Hardware. It receives the position data from the image postprocessing and calculates the resulting current hand model position. [24]

5.5 handmodel

6 System conception

6.1 Technical conception

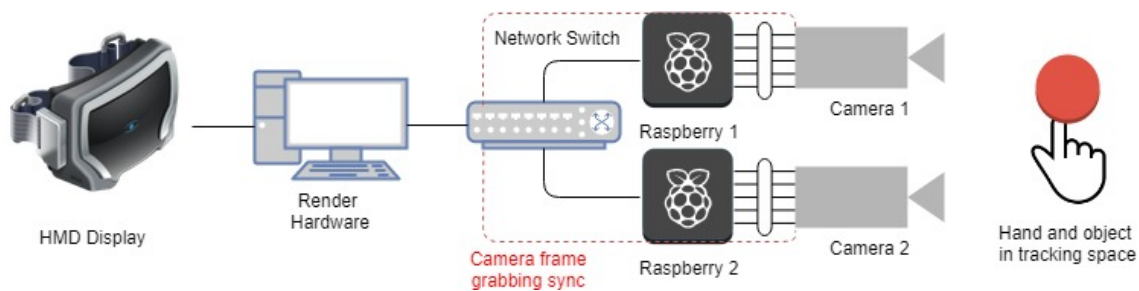


Figure 6.1: Technical conception of the single hardware parts

The technical setup of the whole system is relatively simple. Instead of using one large dimensioned Unit which takes care of all calculations, the image processing steps that have to be done on both stereo images anyway are outsourced onto the Raspberry Pis. As already mentioned in the conception Phase, the image processing directly on the Raspberry also reduces latency which would occur when sending image data over network channels to the main unit.

The two **Raspberry Pi 3 Model B**, which are used as the controllers for the **Raspberry Pi Camera** are connected via network cable to a network switch. A connection over wireless Network would be possible with the on-board capabilities of the Pi's, but here we might encounter latency problems and/or signal interference with other existing networks. Therefore a cable connection is the safer solution. The switch also connects to the "Master" PC to which the "Slave" Pi's communicate their data. The Master also takes care of the stereoscopic calculations, as it is dimensioned with far more computing power than the slaves. The 2D positional data from the slaves and the calculation results from the stereo image disparity is fed into the hand Model running on the "Master". The model solution is then applied to the digital hand model and rendered to the HMD.

6.2 Image Analysis with OpenCV 3 and Python on the Raspberry

For the Image analysis part, OpenCV3 with its Python 3 binding is chosen. The OpenCV package needs to be downloaded and compiled onto each device before it can be used. Before the actual image analysis part can take place, the used cameras need to be calibrated. As they are basically pinhole cameras, they introduce amounts of radial and

tangential distortion to the images. These distortions need to be compensated for, especially when using them as source for the stereo images. Image distortion in these pictures would lead to incorrect calculations for image depth. OpenCV supplies the tools to calculate the distortion parameters as well as the camera matrix [31]. The camera matrix is basically a description for the transformation of points in 3D object space to the 2D image space of the camera. For the used Cameras, we can consider a central projection of points in space onto our image plane. The center of the camera is considered the center of an euclidean coordinate system and the projection plane z is located at distance f equal to the focal length of the camera.

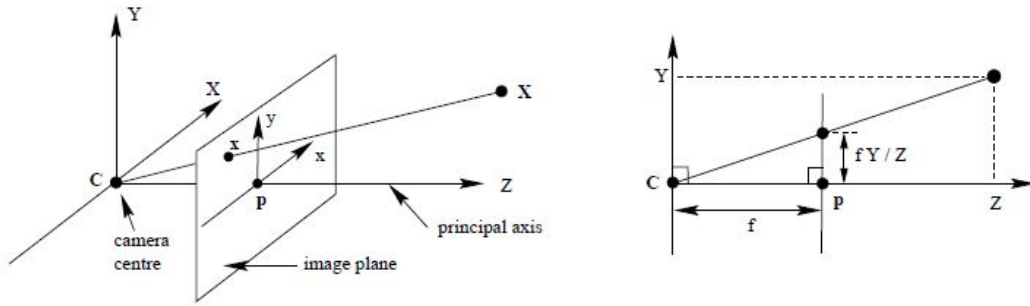


Figure 6.2: Image point projection for pinhole camera systems[17]

As shown in Figure 6.2, a point from the object space can be projected onto the image plane by drawing a ray from the object space point to the camera center. The intersection point of the ray with the image plane defines the projection point. With this knowledge the projection can be described as :

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T \quad (6.1)$$

elaborate

For the calculation of the image distortion , *OpenCv* uses a chessboard image for calibration. The printed image is held in front of the camera at a constant distance and rotated and positioned differently. After every position/rotation change an image of the pattern should be taken. These images are then used to find the corners at the intersection of the black and white squares. With the previous knowledge of the square sizes, the projection errors in radial and tangential directions can be calculated:

add
Source-
code

$$\begin{aligned} x_{tanDistorted} &= x(1 + k_1r^2 + k_2r^4 + k_3 + r^6) \\ y_{tanDistorted} &= y(1 + k_1r^2 + k_2r^4 + k_3 + r^6) \end{aligned} \quad (6.2)$$

$$\begin{aligned} x_{radDistorted} &= x + [2p_1xy + p_2(r^2 + 2x^2)] \\ y_{radDistorted} &= y + [p_1(r^2 + 2y^2) + 2p_2xy] \end{aligned}$$

As can be seen from the above equations, the parameters that need to be calculated are k_1, k_2, p_1, p_2, k_3 .

continue

The python program that is used for tracking the specified color markers is comprised of the following components:

- Image acquisition from camera
- Image conversion to HSV colorspace
- Mask construction and Filtering
- Contour finding and Position calculation
- Sending of positional data via Network as UDP Package

6.2.1 Image acquisition from camera

As the computation times for image filtering and mask generation may vary, it makes sense to separate the image acquisition from the computation part. Therefore the loading of the raw image data frame from the camera is outsourced into its own thread. Also this allows us to trigger the frame grabbing on both devices for synchronization. Frame grabbing synchronization is done via triggering a PWM signal on one of the raspberries and sending it to the other Pi. The synchronosity of the two frame reading threads is assumed for the first implementation and has to be tested on a finished setup.

The thread takes the data of the current image frame and hands it over to the main thread, where the image processing is handled. For the first frames, the full image data is needed as the position of the markers is not yet known. Performance-wise, this introduces a lot of overhead, since each mask generation step for the specific colors has to go through the complete image data, meaning every pixel has to be read out at least 5 times to get the tracking for all different markers. After a few frames, we get hold of the marker position and can define a reduced region of interest (ROI) on the image. The new ROI results from the positional data from the previous frame and has to be dimensioned correctly as to incorporate the possibility of large position differences between consecutive frames. Should no marker be found in the defined ROI or the certainty of the result is not high enough, the frame has to be dropped and the next frame should utilize the whole image data.

6.2.2 Image conversion to HSV colorspace

The image data that is supplied by the camera comes in an RGB data format, which we could already use for the further calculation. It does though make more sense to convert the input data into the HSV colorspace. Since we are not using high precision cameras, it is necessary to define a range of color values around the desired color which we want to

track. The HSV colorspace is displayed as a cone, in comparison to the RGB colorspace, which is displayed as a cube. The color values for the HSV space are all located on a circle spanning from 0 to 360 degrees. The Hue value (H) corresponds to the angle on the circle, where 0° corresponds to a redish color, 120° lies in the area of green and 240° and above correspond to blue. The saturation value (s) corresponds to the amount of white the color contains where 0 is pure white and 100 corresponds to the fully saturated color. For optimal results, only highly saturated colors should be used to ensure correct color detection. The last component is the value component (V) which describes the intensity of the color. Alike the value settings for the saturation, value ranges of at least 50 should be used for tracking precision.

For tracking the five fingers of the hand we need five distinguishable colors. Here the primary colors red, green and blue will be the choice for the first three colors. The other two selected colors are orange and yellow. To be able to clearly distinguish these colors in the video frame, a constant and homogenous lighting is needed as well as a color temperature of the lighting that is in the neutral area to not change the color of the markers.

check
test
results

the color conversion from the input RGB values to the desired HSV colorspace is done as follows:

$$\begin{aligned} hsv_{low} &= (hue_{targetcolor} - sensitivity, 100, 100) \\ hsv_{up} &= (hue_{targetcolor} + sensitivity, 100, 100) \end{aligned} \quad (6.3)$$

6.2.3 Mask construction and filtering

These values are used as the parameters for a mask generation which generates a binary mask for the current frame where all pixels whose values lie outside of the specified range are set to zero (black) and the remaining are set to 255 (white). For these masks to work properly, any other larger objects that might contain similar colors should be removed from the scene to eliminate wrong tracking data. As all digital cameras tend to have signal noise in the sensor data, high frequency noise in the color channels will be present. This noise needs to be filtered out before any further computation on the image data can be done.

orig im-
age and
mask
images
for dis-
play

The first step in this process is to use a gaussian filter to blur the mask. After this step an erosion and a dilation is applied to the image to further eliminate unwanted noise.

elaborate

6.2.4 Contour finding and position calculation

With the cleaned mask we can continue and search for the white areas in the mask which might represent our target. Under the assumption that we have removed all other parasitic objects from the image, the marker should be the largest area of positive pixels in the mask frame. Therefore only the largest area found in the search is taken as the

6 System conception

desirited tracking marker. For the selected area, a fitting bounding box is calculated and the center of this box is used as the posiitonal parameter of the tracking marker.

6.2.5 Sending of positional data via network as UPD package

The resulting data is then handed over to a seperate thread running a UDP server which sends the positional data to the parent device where the stereoscopic calulations as well as the hand model and rendering is done

7 System Evaluation

8 Resume

List of Figures

2.1	Bone structure of the human left hand ([25])	2
2.2	Representation of the DOFs of the human hand	3
2.3	Possible solution for an IK problem of a human finger: (a)The given target position of the end effector can not be reached. (b) The given target can only be reached by one solution.(c) The target position can be reached with multiple different solutions.	7
2.4	Forward and backward calculation steps for one iteration of the FAB-RIK algorithm.(a) initial position of the system.(b)End effector is moved to target position.(c) Deterimne position of next joint on constructed line.(d)repeat until root is reached. (e)move root joint to initial position. (f) repeat calculation in reverse direction [2]	12
3.1	Example for an optical tracking system with passive infrared reflector markers [32]	15
3.2	Acceleglove system with the described sensor positions [18]	17
3.3	Color glove setup used by Wang and Popovic [43]	18
3.4	Color glove setup used by Fredriksson and Ryen [14]	19
4.1	Pavlovic, Sharma et al: Hand models. Different hand models can be used to represent the same hand posture. (a) 3D Textured volumetric model. (b) 3D wireframe volumetric model. (c) 3D skeletal model. (d) Binary silhouette. (e) Contour.	22
6.1	Technical conseptiopn of the single hardware parts	28
6.2	Image point projection for pinhole camera systems[17]	29

List of Tables

Bibliography

- [1] ANDREAS ARISTIDOU AND JOAN LASENBY: Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver: University of Cambridge, Department of Engineering. (2009)
- [2] ARISTIDOU, Andreas ; LASENBY, Joan: FABRIK: A fast, iterative solver for the Inverse Kinematics problem. In: *Graphical Models* 73 (2011), Nr. 5, S. 243–260. <http://dx.doi.org/10.1016/j.gmod.2011.05.003>. – DOI 10.1016/j.gmod.2011.05.003. – ISSN 15240703
- [3] BADLER, Norman ; MANOOCHEHRI, Kamran ; WALTERS, Graham: Articulated Figure Positioning by Multiple Constraints. In: *IEEE Computer Graphics and Applications* 7 (1987), Nr. 6, S. 28–38. <http://dx.doi.org/10.1109/MCG.1987.276894>. – DOI 10.1109/MCG.1987.276894. – ISSN 02721716
- [4] BOLT, Richard A.: Put-that-there. In: THOMAS, James J. (Hrsg.): *Proceedings of the 7th annual conference on Computer graphics and interactive techniques*. New York, NY : ACM, 1980. – ISBN 0897910214, S. 262–270
- [5] CHIAVERINI, S. ; SICILIANO, B. ; EGELAND, O.: Review of the damped least-squares inverse kinematics with experiments on an industrial robot manipulator. In: *IEEE Transactions on Control Systems Technology* 2 (1994), Nr. 2, S. 123–134. <http://dx.doi.org/10.1109/87.294335>. – DOI 10.1109/87.294335. – ISSN 10636536
- [6] COURTY, Nicolas ; ARNAUD, Elise: Inverse Kinematics Using Sequential Monte Carlo Methods. In: HUTCHISON, David (Hrsg.) ; FISHER, Robert B. (Hrsg.) ; KANADE, Takeo (Hrsg.) ; KITTLER, Josef (Hrsg.) ; KLEINBERG, Jon M. (Hrsg.) ; MATTERN, Friedemann (Hrsg.) ; MITCHELL, John C. (Hrsg.) ; NAOR, Moni (Hrsg.) ; NIERSTRASZ, Oscar (Hrsg.) ; PANDU RANGAN, C. (Hrsg.) ; PERALES, Francisco J. (Hrsg.): *Articulated Motion and Deformable Objects*. Berlin and Heidelberg : Springer, 2008 (Lecture Notes in Computer Science). – ISBN 978–3–540–70517–8, S. 1–10
- [7] DAHMEN, Wolfgang ; REUSKEN, Arnold: *Numerik für Ingenieure und Naturwissenschaftler*. Zweite, korrigierte Auflage. Berlin Heidelberg : Springer-Verlag Berlin Heidelberg, 2008 (Springer-Lehrbuch). <http://dx.doi.org/10.1007/978-3-540-76493-9>. – ISBN 9783540764939

- [8] DAVID E. ORIN ; WILLIAM W. SCHRADER: Efficient Computation of the Jacobian for Robot Manipulators. In: *The International Journal of Robotics Research* 3 (1984), Nr. 4, S. 66–75. – ISSN 0278–3649
- [9] DENAVIT, Jacques: A kinematic notation for lower-pair mechanisms based on matrices. In: *ASME J. Appl. Mech.* (1955), S. 215–221
- [10] DIPIETRO, L. ; SABATINI, A. M. ; DARIO, P.: A Survey of Glove-Based Systems and Their Applications. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38 (2008), Nr. 4, S. 461–482. <http://dx.doi.org/10.1109/TSMCC.2008.923862>. – DOI 10.1109/TSMCC.2008.923862. – ISSN 1094–6977
- [11] DUCA, Florin ; FREDRIKSSON, Jonas ; FJELD, Morten: Real-time 3d hand interaction: Single webcam low-cost approach. In: *Proceedings of the Workshop at the IEEE Virtual Reality 2007 Conference; Trends and Issues in Tracking for Virtual Environments*, 2007, S. 1–5
- [12] EROL, Ali ; BEBIS, George ; NICOLESCU, Mircea ; BOYLE, Richard D. ; TWOMBLY, Xander: Vision-based hand pose estimation: A review. In: *Computer vision and image understanding* 108 (2007), Nr. 1-2, S. 52–73. <http://dx.doi.org/10.1016/j.cviu.2006.10.012>. – DOI 10.1016/j.cviu.2006.10.012
- [13] FOXLIN, Eric ; HARRINGTON, Michael ; PFEIFER, George: Constellation. In: CUNNINGHAM, Steve (Hrsg.): *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*. New York, NY : ACM, 1998. – ISBN 0897919998, S. 371–378
- [14] FREDRIKSSON, Jonas ; RYEN, Sven B. ; FJELD, Morten: Real-time 3D hand-computer interaction. In: TOLLMAR, Konrad (Hrsg.): *Proceedings of the 5th Nordic conference on Human-computer interaction building bridges*. New York, NY : ACM, 2008. – ISBN 9781595937049, S. 133
- [15] GOLUB, Gene ; KAHAN, William: Calculating the singular values and pseudo-inverse of a matrix. In: *ISAM* 2 (1965), Nr. 2, 205–224. <http://statistics.uchicago.edu/~lekheng/courses/302/classics/golub-kahan.pdf>
- [16] GRIMES, Gary J.: *Digital data entry glove interface device*. 1983
- [17] HARTLEY, Richard ; ZISSERMAN, Andrew: Multiple View Geometry in Computer Vision Second Edition. In: *Cambridge University Press* (2000)
- [18] HERNANDEZ-REBOLLAR, Jose L. ; KYRIAKOPOULOS, Nicholas ; LINDEMAN, Robert W.: The AcceleGlove. In: APPOLLONI, Tom (Hrsg.): *ACM SIGGRAPH*

- 2002 conference abstracts and applications. New York, NY : ACM, 2002. – ISBN 1581135254, S. 259
- [19] HUTTENLOCHER, D. P. ; KLANDERMAN, G. A. ; RUCKLIDGE, W. J.: Comparing images using the Hausdorff distance. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 15 (1993), Nr. 9, S. 850–863. <http://dx.doi.org/10.1109/34.232073>. – DOI 10.1109/34.232073. – ISSN 01628828
 - [20] KOREIN, James U.: *A geometric investigation of reach: Zugl.: Pennsylvania Univ., Diss. : 1984*. Cambridge, Mass. : MIT Press, 1985 (ACM distinguished dissertations). – ISBN 0262111047
 - [21] KRISTENSEN, Fredrik ; NILSSON, Peter ; ÖWALL, Viktor: Background Segmentation Beyond RGB. In: NARAYANAN, P. J. (Hrsg.) ; NAYAR, Shree K. (Hrsg.) ; SHUM, Heung-Yeung (Hrsg.): *Computer Vision - ACCV 2006*. Guildford : Springer London, Feb. 2007. – ISBN 978–3–540–32432–4, S. 602–612
 - [22] KURODA, T. ; TABATA, Y. ; GOTO, A. ; IKUTA, H. ; MURAKAMI, M. u.a.: Consumer price data-glove for sign language recognition. In: *Proc. of 5th Intl Conf. Disability, Virtual Reality Assoc. Tech., Oxford, UK*, 2004, S. 253–258
 - [23] LA GORCE, Martin de ; PARAGIOS, Nikos ; FLEET, David J.: Model-based hand tracking with texture, shading and self-occlusions. In: *IEEE Conference on Computer Vision and Pattern Recognition, 2008*. Piscataway, NJ : IEEE Service Center, 2008. – ISBN 978–1–4244–2242–5, S. 1–8
 - [24] LANSLEY, Alastair ; VAMPLEW, Peter ; SMITH, Philip ; FOALE, Cameron: Caliko: An Inverse Kinematics Software Library Implementation of the FABRIK Algorithm. In: *Journal of Open Research Software* 4 (2016), Nr. 1. <http://dx.doi.org/10.5334/jors.116>. – DOI 10.5334/jors.116. – ISSN 2049–9647
 - [25] LEE, Jintae ; KUNII, T. L.: Model-based analysis of hand posture. In: *IEEE Computer Graphics and Applications* 15 (1995), Nr. 5, S. 77–86. <http://dx.doi.org/10.1109/38.403831>. – DOI 10.1109/38.403831. – ISSN 02721716
 - [26] LIN, John ; WU, Ying ; HUANG, T. S.: Modeling the constraints of human hand motion. In: *Proceedings, Workshop on Human Motion*. Los Alamitos, Calif : IEEE Computer Society, 2000. – ISBN 0–7695–0939–8, S. 121–126
 - [27] MAJEAU, L. ; BORDUAS, J. ; LORANGER, S. ; EL-IRAKI, Y. ; LAVOIE, J. ; BANVILLE, D. ; LATENDRESSE, V. ; BELAND, V. ; DANIEL-RIVEST, J. ; THIAW, A. ; BAMBARA, H. ; BEAUSOLEIL, T. P. ; TROTTIER-LAPOINTE, W. ; LAPOINTE, J.: Dataglove for consumer applications: Low cost dataglove using optical fiber sensor. In: *7th International Workshop on Fibre and Optical Passive*

- Components (WFOPC)*, 2011. Piscataway, NJ : IEEE, 2012. – ISBN 978–1–4577–0861–9, S. 1–4
- [28] MOESLUND, Thomas B. ; GRANUM, Erik: A Survey of Computer Vision-Based Human Motion Capture. In: *Computer vision and image understanding* 81 (2001), Nr. 3, S. 231–268. <http://dx.doi.org/10.1006/cviu.2000.0897>. – DOI 10.1006/cviu.2000.0897
 - [29] MOESLUND, Thomas B. ; HILTON, Adrian ; KRÜGER, Volker: A survey of advances in vision-based human motion capture and analysis. In: *Computer vision and image understanding* 104 (2006), Nr. 2-3, S. 90–126. <http://dx.doi.org/10.1016/j.cviu.2006.08.002>. – DOI 10.1016/j.cviu.2006.08.002
 - [30] NAKAMURA, Yoshihiko ; HANAFUSA, Hideo: Inverse Kinematic Solutions With Singularity Robustness for Robot Manipulator Control. In: *Journal of Dynamic Systems, Measurement, and Control* 108 (1986), Nr. 3, S. 163. <http://dx.doi.org/10.1115/1.3143764>. – DOI 10.1115/1.3143764. – ISSN 00220434
 - [31] OPENCV: *Camera Calibration*. https://docs.opencv.org/3.1.0/dc/dbb/tutorial_py_calibration.html. Version:2018
 - [32] OPTITRACK: *flex13MocapVolume.png (680×375)*. <http://www.optitrack.com/public/images/flex13MocapVolume.png>. Version:2017
 - [33] ORIN, David E. ; SCHRADER, William W.: Efficient Computation of the Jacobian for Robot Manipulators. In: *The International Journal of Robotics Research* Vol.3 (1984), Nr. No.4. – ISSN 0278–3649
 - [34] PAVLOVIC, V. I. ; SHARMA, R. ; HUANG, T. S.: Visual interpretation of hand gestures for human-computer interaction: A review. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 19 (1997), Nr. 7, S. 677–695. <http://dx.doi.org/10.1109/34.598226>. – DOI 10.1109/34.598226. – ISSN 01628828
 - [35] RAAB, Frederick ; BLOOD, Ernest ; STEINER, Terry ; JONES, Herbert: Magnetic Position and Orientation Tracking System. In: *IEEE Transactions on Aerospace and Electronic Systems* AES-15 (1979), Nr. 5, S. 709–718. <http://dx.doi.org/10.1109/TAES.1979.308860>. – DOI 10.1109/TAES.1979.308860. – ISSN 0018–9251
 - [36] RIJPKEMA, Hans ; GIRARD, Michael: Computer animation of knowledge-based human grasping. In: THOMAS, James J. (Hrsg.): *Proceedings of the 18th annual conference on Computer graphics and interactive techniques*. New York, NY : ACM, 1991. – ISBN 0897914368, S. 339–348

- [37] ROLLAND, Jannick P. ; BAILLOT, Yohan ; GOON, Alexei A.: A SURVEY OF TRACKING TECHNOLOGY FOR VIRTUAL ENVIRONMENTS: Center for Research and Education in Optics and Lasers (CREOL) University of Central Florida, Orlando FL 32816. In: *(Fundamentals of wearable computers and augmented reality* (2001)
- [38] SPONG, Mark W. ; VIDYASAGAR, Mathukumalli: *Robot dynamics and control*. John Wiley & Sons, 2008
- [39] STAUFFER, C. ; GRIMSON, W.E.L.: Adaptive background mixture models for real-time tracking. In: *1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. Los Alamitos, Ca : IEEE Computer Society, 1999. – ISBN 0-7695-0149-4, S. 246–252
- [40] STURMAN, D. J. ; ZELTZER, D.: A survey of glove-based input. In: *IEEE Computer Graphics and Applications* 14 (1994), Nr. 1, S. 30–39. <http://dx.doi.org/10.1109/38.250916>. – DOI 10.1109/38.250916. – ISSN 02721716
- [41] THOMAS A. DEFANTI ; DANIEL J. SANDIN: *Final Report to the National Endowment of the Arts*
- [42] WAMPLER, Charles: Manipulator Inverse Kinematic Solutions Based on Vector Formulations and Damped Least-Squares Methods. In: *IEEE Transactions on Systems, Man, and Cybernetics* 16 (1986), Nr. 1, S. 93–101. <http://dx.doi.org/10.1109/TSMC.1986.289285>. – DOI 10.1109/TSMC.1986.289285. – ISSN 0018-9472
- [43] WANG, Robert Y. ; POPOVIĆ, Jovan: Real-time hand-tracking with a color glove. In: *ACM transactions on graphics (TOG)* Bd. 28, 2009, S. 63
- [44] WELCH, G. ; FOXLIN, E.: Motion tracking: No silver bullet, but a respectable arsenal. In: *IEEE Computer Graphics and Applications* 22 (2002), Nr. 6, S. 24–38. <http://dx.doi.org/10.1109/MCG.2002.1046626>. – DOI 10.1109/MCG.2002.1046626. – ISSN 02721716
- [45] WOLOVICH, W. ; ELLIOTT, H.: A computational technique for inverse kinematics. In: *The 23rd IEEE Conference on Decision and Control*, IEEE, 1984, S. 1359–1363

Eidesstattliche Erklärung

Ich versichere, die von mir vorgelegte Arbeit selbständig verfasst zu haben.

Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die ich für die Arbeit benutzt habe, sind angegeben.

Die Arbeit hat mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegen.

Köln, xx. August 2016

Oliver Kalbfleisch