Technology
Arts Sciences
TH Köln

Masterthesis Medientechnolgie

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 Informations-,
Medien- und Elektrotechnik

im Studiengang
Medientechnologie

Erster Prüfer: Prof. Dr. Arnulph Fuhrmann
Technische Hochschule Köln

Zweiter Prüfer: Prof. Dr. Stefan Michael Grünvogel
Technische Hochschule Köln

Köln, im August 2018

Masterarbeit

Titel: RHOT A real time hand and object tracking system with low cost consumer grade

Gutachter:

Prof. Dr. Arnulph Fuhrmann (TH Köln)

Prof. Dr. Stefan Michael Grünvogel (TH Köln)

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 con-tents 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: RHOT A real time hand and object tracking system with low cost consumer grade

Reviewers:

Prof. Dr. Arnulph Fuhrmann (TH Köln)

Prof. Dr. Stefan Michael Grünvogel (TH Köln)

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 ¡dd month YYYY¿.

Date: dd month YYYY

Contents

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 on-board hardware. For an intensely immersive experience VR goggles are used to explore digitally created worlds.

With this level of immersivenes, 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 "*Degrees-of-Freedom*" (DOF's) and furthermore adds the posibility of combining these witch a tactile component.The first more humble attemps ofe solving 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[**?**].Modern day elaborate system for hand tracking use infrared or stereoscopic Cameras to achieve a performant tracking result, utilizing the calculation capability of standard consumer computers. These systems are rather specialized for the task they are doing and are therefore rather expensive to attain.

This thesis assesses if it is possible to create a real-time hand and object racking system based on easily accessible consumer grade hardware and open source Programs as well as displaying the tracking result on a head mounted display. The system should provide an experience for the user that is as natural as possible in terms of grabbing precision and the components should not hinder the motion capability of the hand. furthermore the system should aim at being comfortable in terms of design as a cumbersome system would not be used in everyday life. Before evaluating a prototype system, an overview of existing technologies in terms of tracking and display of the human hand will be given. A small overview over the anatomical properties of the human hand will be given as the understanding of these plays a major role int he setup for the hand simulation.

2 Description of the hand in digital space

Tracking of the human hand has always been a challenging problem. In comparison to other larger body parts 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 [**?**] describe the human hand as "an articulated structure with about 30 degrees of freedom [which] changes shape in various ways by its joint movements."

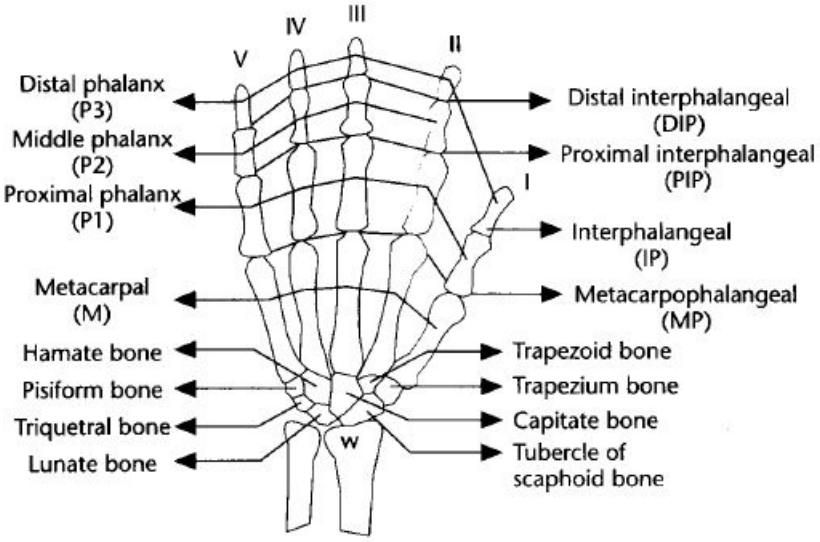


Figure 2.1: Bone structure of the human left hand ([**?**])

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 exiting 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)[**?**]

- 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**)

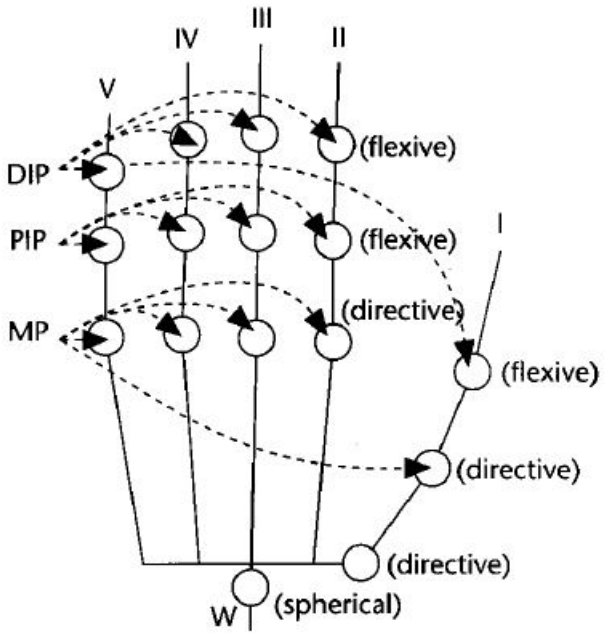


Figure 2.2: Representation of the DOFs of the human hand

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

2.1.1 Constraints in hand motion

A full usage of all the declared DOFs would lead to al large amount of possible combinations. Since the hand is not only made up of bones but also muscles and the skin, we can impose some constraints [**?** **?**] to the movement of the joints. Ling, Wu and Huang([**?**]) 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 human hand.**Type III** constraints are results of common and natural movements and can be differing form 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 sane 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 bend 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 fingers MP normally does not abduct and adduct much. Therefore we can infer an approximation and thereby remove 1 DOF 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 1 DOF 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 thath finger must also be bent. The approximation for this relation[**?**] can be described as :

$$\Theta_{DIP} = \frac{2}{3}\Theta_{PIP} \tag{2.4}$$

Inter-finger constraints can be imposed between joints of adjacent fingers. Inter-finger 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 [**?**].

maybe move hand model section to here?

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 it's 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)[**?**]. It was actually introduced in 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.[**?**] 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 data set of our physical hand with our kinematic system, two major paths for achieving a solution can be taken.

Denavit Hartenberg formmulation for coordinate System definition and transformation

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 $s_1, ..., s_k$. The s_i position is the result of th combination of all the joint angles in the corresponding kinematic chain. Respectively, we define the target position

of the end effectors as $t_1, ..., t_i$, 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.

$$\begin{aligned} \vec{\mathbf{s}} &= (\mathbf{s}_1, ..., \mathbf{s}_n)^T \\ \vec{\mathbf{t}} &= (\mathbf{t}_1, ..., \mathbf{t}_n)^T \\ \vec{\mathbf{e}} &= \vec{\mathbf{t}} - \vec{\mathbf{s}} \end{aligned} \tag{2.5}$$

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

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

With $\boldsymbol{\theta}$ being the column vector $\boldsymbol{\theta} = (\theta_1, ..., \theta_n)^T$.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 if its chain components is rather complicated.Therefore a solution which takes a known position of the *end effector* 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" [**?**]

The concept of *Inverse Kinematics* (IK) already describes it's principle in it's 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.

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

The result of this equation is the vector $\boldsymbol{\theta}$ for which the values of $\vec{\mathbf{s}}$ coincide to the desired configuration $\vec{\mathbf{s}_d}$. 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 calulaton of the f^{-1} function, due to it beeing a highly non linear operator which is not easily invertible.The approaches that [**?**

] 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.

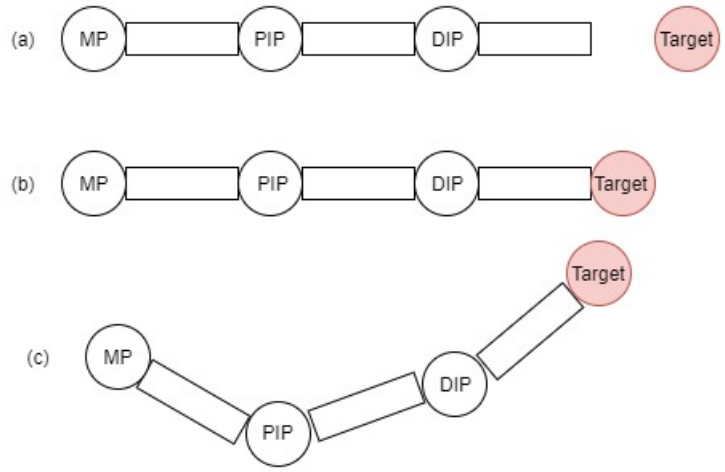


Figure 2.3: Possible solution for an IK problem of a human finger:

(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-based methods

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 **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 dependent on the joint angles $\boldsymbol{\theta}$ values and can be defined as

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

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

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

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 derivative of definition (2.6) with respect to time gives the basic equation for forward kinematics that describes the velocities of the

end effectors:

$$\dot{\vec{s}} = \boldsymbol{J}(\boldsymbol{\theta})\dot{\boldsymbol{\theta}} \tag{2.10}$$

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 $\Delta\boldsymbol{\theta}$ for incrementing the current joint values:

$$\boldsymbol{\theta}_{new} = \boldsymbol{\theta}_{curr} + \Delta\boldsymbol{\theta} \tag{2.11}$$

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

$$\Delta\vec{\boldsymbol{s}} \approx \boldsymbol{J}(\boldsymbol{\theta_{curr}})\Delta\boldsymbol{\theta} \tag{2.12}$$

Using this approximation we can reformulate the FK problem as $\vec{\boldsymbol{e}} = J\Delta\boldsymbol{\theta}$ and therefore our inverse kinematics problem form 2.7 can be expressed as $\Delta\boldsymbol{\theta} = J^{-1}\vec{\boldsymbol{e}}$.
The problem we run into with this solution is the construction of the inverse Jacobian matrix. The Jacobian J may not be square or invertible. In the case of it beeing 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 $\Delta\boldsymbol{\theta}$ without having to calculate the inverse of **J** is done by replacing the inverse with the transpose of **J**.

$$\Delta\boldsymbol{\theta} = \alpha\boldsymbol{J}^T\vec{\boldsymbol{e}} \tag{2.13}$$

Of course the transpose and the inverse of **J** are not the same thing. When using the theorems displayed in [**?** **?**] we can show that:

$$\langle JJ^T\vec{\boldsymbol{e}}, \vec{\boldsymbol{e}}\rangle = \langle J^T\vec{\boldsymbol{e}}, J^T\vec{\boldsymbol{e}}\rangle = \|J^T\vec{\boldsymbol{e}}\| \geq 0 \tag{2.14}$$

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

$$\alpha = \frac{\langle \vec{\boldsymbol{e}}, JJ^T\vec{\boldsymbol{e}}\rangle}{\langle JJ^T\vec{\boldsymbol{e}}, JJ^T\vec{\boldsymbol{e}}\rangle} \tag{2.15}$$

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*[**?**] for the calculation. The

pseudo inverse is defined for all matrices J, even ones which are not square or not of full row rank.

$$\Delta\boldsymbol{\theta} = J^{\dagger}\vec{\boldsymbol{e}} \tag{2.16}$$

The *pseudo inverse* represents the best possible solution for $J\Delta\boldsymbol{\theta} = \vec{\boldsymbol{e}}$ 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 null space of **J** . Therefore, for all vectors $\varphi, J(I - J^{\dagger}J)\varphi = 0$. Therefore $\Delta\boldsymbol{\theta}$ can be set by

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

for any vector φ and still obtain a value for $\Delta\theta$ which minimizes the value $J\Delta\boldsymbol{\theta} - \vec{\boldsymbol{e}}$.The pseudo inverse of **J** can be derived as follows:

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

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

2.2.4 Singular Value Decomposition

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

$$\boldsymbol{J} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^{\boldsymbol{T}} \tag{2.19}$$

For an *m*×*n* Jacobian matrix, **U** is *m*×*m* and the columns are the eigenvectors of JJ^{T}. **D** is *m*×*n*, and the entries are zero except along the diagonal where $d_{ii} = \sigma_i$ with $\boldsymbol{\sigma_i}$ beeing the singular value and $\sigma 1 \geq \sigma 2 \geq ... \geq \sigma m \geq 0$. Singular values are the non-negative square roots of the eigenvalues of JJ^{T} and $J^{T}J$.**V** is *n*×*n* and the columns are the eigenvectors of $J^{J}J$. The construction of the pseudo inverse is done as follows:

$$\boldsymbol{J}^{\dagger} = \boldsymbol{V}\boldsymbol{D}^{\boldsymbol{\dagger}}\boldsymbol{U}^{\boldsymbol{T}} \tag{2.20}$$

The pseudo-inverse of the diagonal matrix **D** is obtained by replacing each positive diagonal entry with it's reciprocal [**?**].

2.2.5 Damped Least Squares

The *Dampened Least Squares* method is 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 occurences for inverse kinematics can be found in [**?** **?**]. Instead of finding the minimum for $\Delta\theta$ as a best solution for the

equation displayed in 2.5, the goal of this method is to find the value that minimizes the following quantity with $\lambda \in \mathbb{R}$ as a dampening factor.

$$\| J\Delta\boldsymbol{\theta} - \vec{\boldsymbol{e}} \|^2 + \lambda^2 \| \Delta\boldsymbol{\theta} \|^2 \tag{2.21}$$

which can be rewritten to the corresponding

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

and therefore

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

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[**? ?**].

2.2.6 Newton Methods

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

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

where Hf(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 minimization 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.

2.2.7 FABRIK Algorithm

A more recent approach in the kinematics field is the **FABRIK** algorithm proposed by Aristidou and Lasenby[**?**].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 is solves for the position of a point on a line to retrieve the new joint positions. This is done in an 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 $\mathbf{d}_i$.

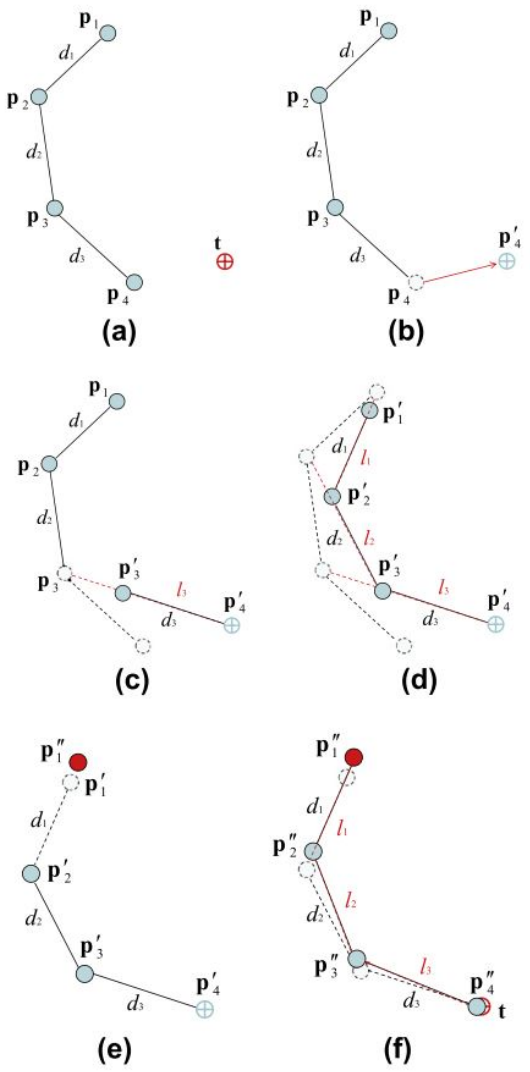


Figure 2.4: Forward and backward calculation steps for one iteration of the **FABRIK**algorithm.(a) initial position of the system.(b)End effector is moved to target position.(c) Deterimne position of next joint on constructed line.(d)repeat until root is reached. (e)move root joint to initial position.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

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

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 manged 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,...,1 is equal to **t**.

$$\mathbf{p}_n^{'} = \mathbf{t} \tag{2.26}$$

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

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

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)**).

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

Consecutively, this is done with the remaining joints until the root joint is reached(see **(d)**). This finishes the first halve 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 it's 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 algorithm

review of algorithm capabilities

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 developement since the 1980's [**?**] and have since then resulted in several solution attempts. Sturman and Zeltzer gave a survey on the exsisting tracking mehtods in their paper [**?**]. 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 [**?** **?**]. They account for solutions based in optical tracking based on marker detection, magnetic detection via measurements of an artificial magnetic field[**?**] 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 a complete coverage of the tracking space. This leads to a higher system complexity in terms of setup and calibration.

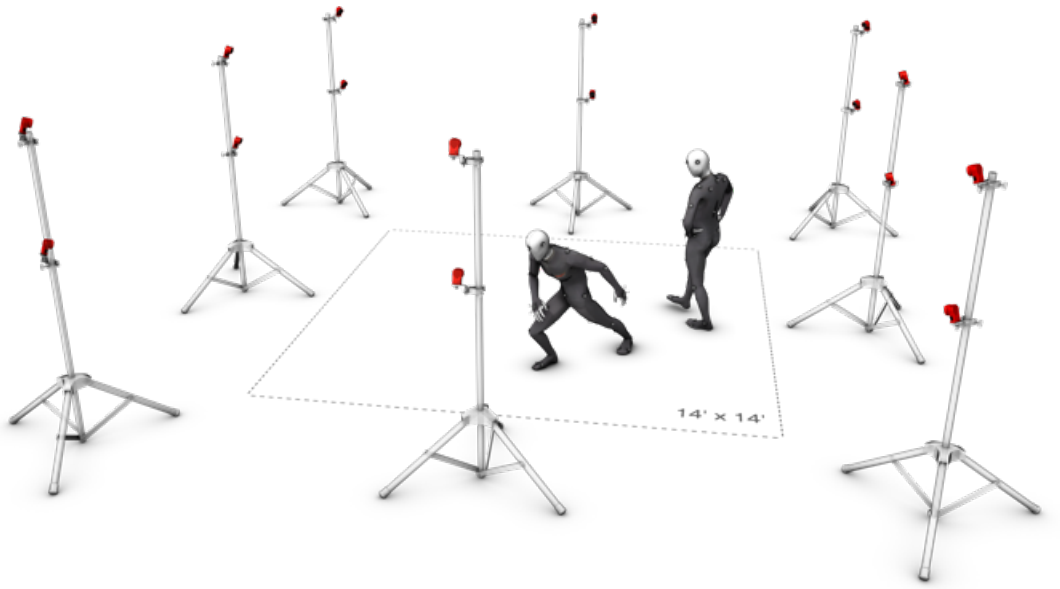


Figure 3.1: Example for an optical tracking system with passive infrared reflector markers [**?**]

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 orthogonal oriented axes to produce a 3D vector of the units orientation with respect to the excitation. As the earths magnetic field prone to changes based on geographical location, data corrections have to be applied to the measurements.Another solution is to generate an local magnetic field via a multi-coil 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 excitation's, you can estimate the position and orientation of the sensor unit with respect to the source unit.T he 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 sources 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 Accoustic 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.Sound waves travel at a much slower speed than light waves which brings benefits and downsides with it. Sound waves 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 sound waves travel through is also a limiting factor as humidity, air pressure and air currents can influence the traveling sound waves.In comparison to the optical systems, the acoustic systems are not as prone to occlusion errors since sound waves have a better ability to bend around obstacles than light waves. Most of these downside can be adressed with a combination of these systems with another form of tracking like in [**?**].

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 an 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 frame rate demand for the tracking cameras. The following section will describe different approaches to solve the aforementioned difficulties.

3.2.1 Sensor based

Tracking systems thath make use of sensors mounted to the hand via straps or a glove have alread been in use since the 1970's. First prototype glove systems included the Sayre Glove[**?**] and the Digital Entry Data Glove[**?**]. The Sayre glove is based off an idea of Richard Sayre. The system utilizes a light source and a photocell 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 photo diode. 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 systems 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. These early systems only provided a limited amount of sensors and were rather unpractical in use. As they were developed to serve very specific applications, they were used only briefly, and never commercialized. Newer developements in this sector include [**? ? ?**]. The *AcceleGlove* presented by [**?**] consists of six dual-axis acelerometers, 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 [**?**] 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 realte the finger movement to a change

Figure 3.2: Acceleglove system with the described sensor positions [**?**]

in magnetic flux induced by the movement of the sensor parts thath are attached to the finger. The sensor functionality is simular to the functionality of the light sensors from [**?**]. 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 [**?**] propsed a systems that uses optical flexion sensors to determine the bending angles of the fingers. The opticcal 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 [**?**]. Furthermore the system is also capable of measuring the abduction of two finger. 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 knuckle base and then up to the detector. When abducting the two fingers, the angle at the knuckle base changes, resulting in an intensity change at the detector through the loss of intensity at the curvature of the optic fiber. 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 [**?** **?**]

Find 1-2 newer systems for comparison

3.2.2 Appearance Based - Optical marker

Optical approaches use properties of the human hand to 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 section 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 [**?** **?** **?**]. In the method described by Wang and Popovic [**?**],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).

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

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. Algortihms that use other characteristics for tracking like texture or shading [**?**] rely on an accurate pose from a previous frame to constrain the posture 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 over complete 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 databse iamges. The comaprison is done via a nearest neighbor lookup using a Hausdorff-like distance [**?**] 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 [**?**] 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.

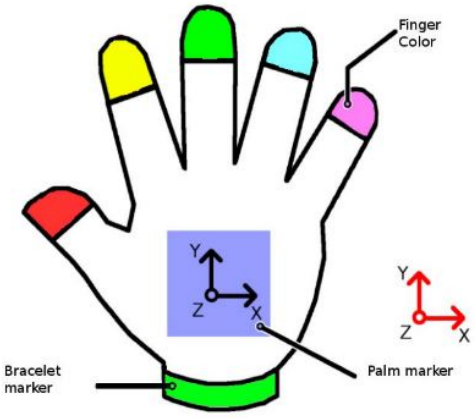


Figure 3.4: Color glove setup used by Fredriksson and Ryen [?]

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 the tracking area in a calibration process. Defining a bounding 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 an palm marker position, the system tracks the grip angle and the lateral tilt angle for each finger. The grip angle is defined as 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 estimations for the fingers Y position around an origin position at the knuckles 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 focuses on the 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 theses 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 have the drawback of not fitting every hand shape 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 artificial 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 [**?** **?**]. 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 [**?**]. It uses the idea of representing each pixel value as a mixture of Gaussian functions (*MoG*) and updating these pixels during run time 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 [**?**]:

$$if\ (|P_{new} - P^d_{mean}| \leq KP^d_{std}) \quad then \quad P_{new}\ \in\ Distribution\ d \tag{3.1}$$

where P_{new} is the new pixel value, P^d_{mean} and P^d_{std} 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 an background is achieved,the hand pose has to be retrieved from the left over image data. Solutions for retireving a full set of DOF as hand pose estimation data can be split into two main categories[**?**], *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 run time 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 assumptions on time coherence. At each iteration of the search algorithm the whole data set is searched for a match.This on one side introduces more computational work for achieving a match, on the other side lowers the amount of constraints that have to be put on the user when initializing the tracker system. Furthermore, the possibly rapid motion of the human hands and fingers acts as factor failure for time coherence dependent approaches, where a single frame approach is not as vulnerable.

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 photo realistic digital replicas of human skin wrapped onto anatomically precisely modeled body parts. These highly complex models acquire a lot of calculation resources and are mostly not appropriate for a mealtime 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.

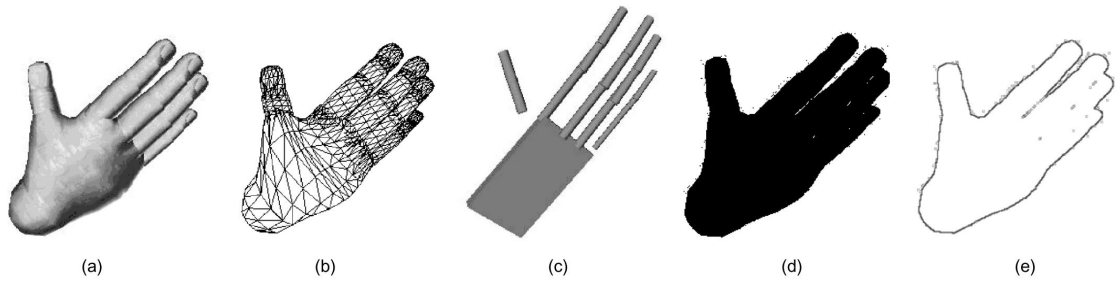


Figure 4.1: Pavlovic, Sharma et al: Hand models. Different hand models can be used to represent the same hand posture. (a) 3D Textured volumetric model. (b) 3D wire frame volumetric model. (c) 3D skeletal model. (d) Binary silhouette. (e) Contour.
[**?** , p. 682]

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 wire frame 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 real time if not supplying a large amount of computational power.
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 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 refereed to as *skeletal models* as they mostly try to reproduce the strucutre of the human hand as described in Section 2.1

5 System prequisites

Befor elaborating a final system conception, some perquisites have to be defined.
System components to be defined:

- Display hardware
- Tracking Technology
- Tracking Hardware
- Data cleaning
- Image analysis
- Inverse kinameatics algorithm
- Hand model

5.1 Display

The first definition that has to be done is actually the last component of the processing chain. The final display hardware properties have to be defined initially as these are the determining factor for all other components. Since the goal of hand and object tracking is the most 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 an 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 smart-phone. 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 an rotation integrated. The downside of this device class reveals 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 they are still no comparison for a desktop setup. Also these devices, as they are first of all mobile phones, show lacks for wire connections, making it necessary to mostly communicate over wireless protocols. This leads to a growth in infrastructure(server, wi-fi-hotspot, etc.) and every further node in the processing chain leads to a physical

get device specs for current phones

signal delay.
After these first preliminary steps, only pure VR-Headsets like the *HTC Vive* or *Occulus 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 high-power CPU capabilities of modern desktop computers. Furthermore, 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 frame rate needed. But with the latest generations of GPU's and CPU's, this category was opened towards the normal consumer.

get refs for devices

The *HTC Vive* as well as the *Occulus Rift* are able to display 90 mages per second on their displays with each having a resolution of 2160x1200 pixels. The duration of one frame is therefore around 11 ms. This is the time frame in which the rest of the components can theoretically 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.

Occulus specs

5.2 Tracking technology

Physical sensor placed on the hand tend to give more precise tracking results compared to optical methods. The cost for the data precision is that the sensors have to be mounted to the hand in some kind of fashion. Cables or fibers, depending on sensor type, can hinder the hand movement. Utilizing a glove based system brings the downside of the " one size fits all" problem. Human hands can differ largely in size, therefore on glove will not be able to be used for a wide range of users. Furthermore cloth gloves tend to get dirty while usage and are therefore unsuitable in therms of hygiene.

Color markers can be utilized very easily. The simplest form could be colored electrical tape wrapped around the fingertips. These markers can be made disposable after usage. The size of the marker can be adapted to be large enough to realize a robust tracking while being small enough to not constrain hand movement. They also provide the possibility to perform a person specific calibration of the hand model in an initialization step. The easiest way to detect color markers is to use an optical tracking system. Monocular optical tracking system are relatively easy to set up as they only need one camera, the complexity in the later processing steps of evaluating the marker positions from only a single frame rises dratically.
Stereoscopic system in comparison have a higher inital effort in setting up and calibrating the system, but the further processing steps are fairly easy.

5.3 Tracking Hardware

As explained int 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 Frame rates (60 fps and above), but most of the time store these high frame rate 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 frame rates yet. Also the hardware that is needed to record such specific high frame rates is relatively expensive. Consumer grade cameras like the GoPro that are affordable, can record at a Frame rate 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 frame rate to satisfy the time frame given by the display frame rate.
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 on-board hardware connector for a camera. The camera hardware 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 it's 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. But as hindering as these trackers are on the hand, they could be easily used to provide tracking data for used objects. the tracker could be positioned onto the object where it does not hinder the usage.
Since 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 informa-

tion 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 post processing.
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.

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 could 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.

5.4 Data cleaning

Every system that deals with the computation of live data is suspect to data fluctuation in input and output values. It is also to be assumed that the system will not be able to achieve an ideal synchronization between the cameras. This will probably introduce a degradation of tracking performance which has to be handled. It has to be evaluated at which point a data cleaning will lead to the best end results.

5.5 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 build in features for camera calibration, color and object detection as well as several image optimization features that might be helpful for the processing.

5.6 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 post-processing and calculates the resulting current hand model position. The applied algorithm for the IK solution should be able to keep up with the data-flow from the tracking algorithm and should produce a

fluid optical result on the display. As the calculation of the IK solution and the rendering of the scene with the models should be done on the same machine, computation power and memory consumption should be critical criteria. As the system will rely on optical tracking data, an algorithm that is capable of dealing with tracker occlusions and recovery from this data loss should be prioritized. ([**?**])

5.7 Hand model

For evaluation purposes, a rather simple model form should be used for the representation of the tracked hand in the digital space. The outcome results for tracking accuracy, update speed and the resulting match of physical and digital position of the hand are of more relevance than a highly realistic hand model.

6 System conception

6.1 Technical conception

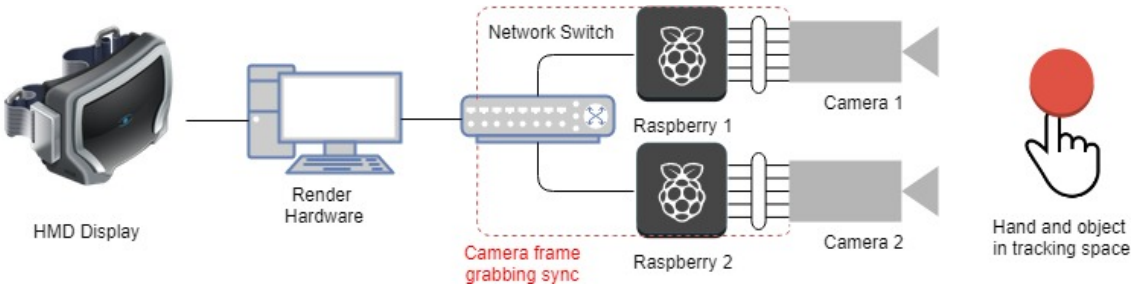


Figure 6.1: Technical conception of the single hardware parts

As depicted in Figure 6.1 the technical components of the setup are realtively 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 Pi's. The mage processing can be dne on the two Pi's in parallel, whch cuts down processing time. Furthermore the image data stay on the device and does not have to be sent to a min unit whch would introduce network and data readng latency.

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, but this might create latency problems and/or signal interference with other existing networks. Therefor 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 computational 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 it's *Python 3* bindings 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 [**?**]. 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.

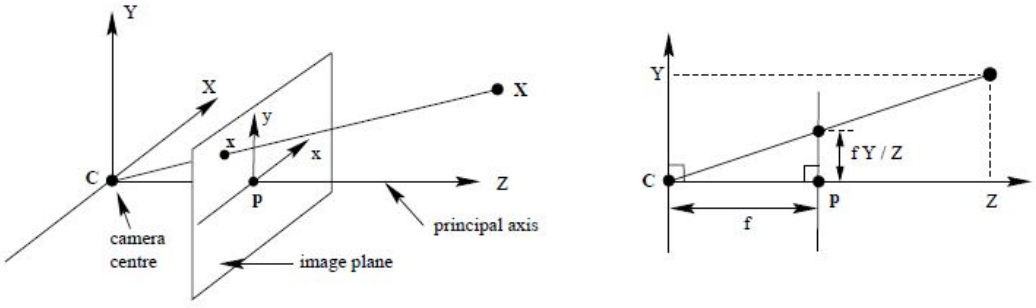


Figure 6.2: Image point projection for pinhole camera systems[**?**]

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 :

$$(X, Y, Z)^T \mapsto (fX/Z, fY/Z)^T \tag{6.1}$$

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:

elaborate

add Source code

$$\begin{aligned} x_{tanDistorted} &= x(1 + k_1 r^2 + k_2 r^4 + k_3 + r^6) \\ y_{tanDistorted} &= y(1 + k_1 r^2 + k_2 r^4 + k_3 + r^6) \end{aligned} \tag{6.2}$$

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

As can bee 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 prototype 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 an position calculation
- Sending of positional data via Network as UPD Package

6.2.1 System Parameters

6.2.2 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 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 can be done via triggering a PWM signal on one of the raspberry's and sending it to the other Pi. The synchronicity of the two frame reading threads is assumed for the first implementation and has to be tested on a finished setup.
The tread 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 hast 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.

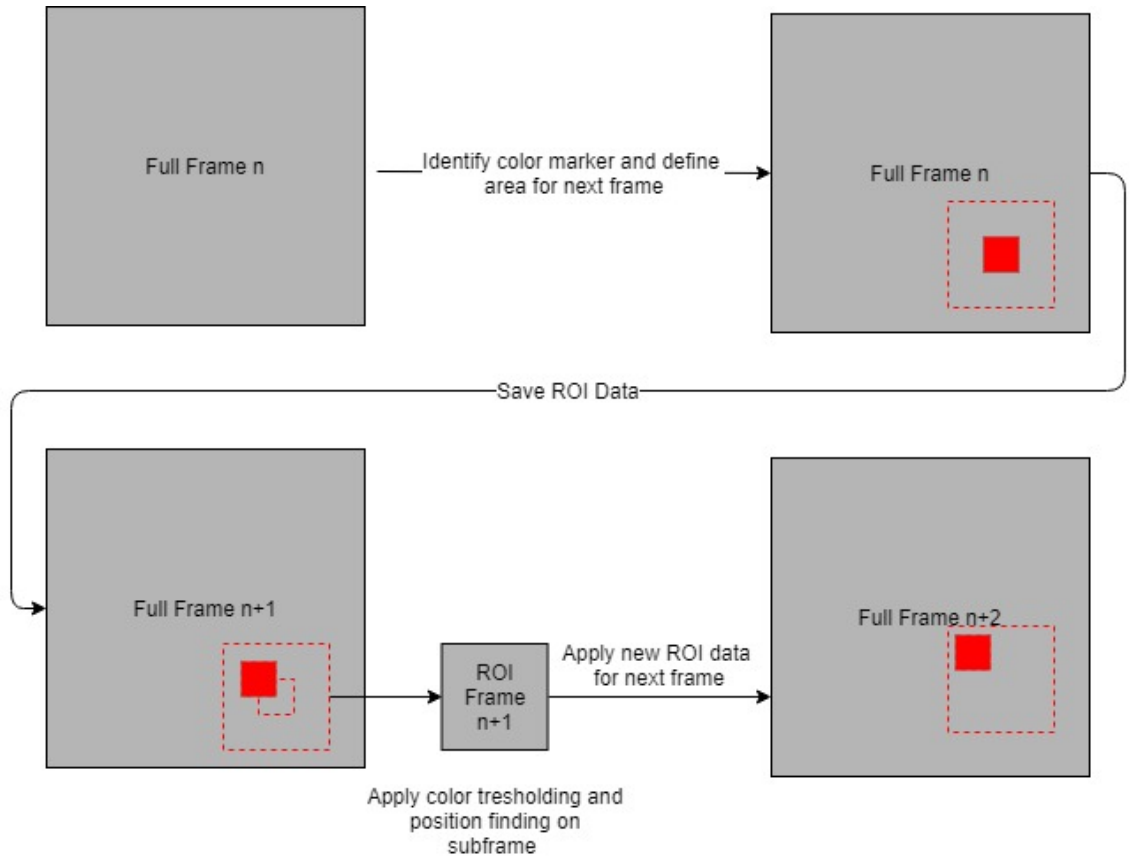


Figure 6.3: ROI calculation for consecutive frames

After a few frames, we get hold of the marker position and can define a reduced region of interest (ROI) on the image (Figure 6.3). The new ROI results from the positional data of the previous frame and has to be dimensioned correctly as to incorporate the possibility of large position differences between consecutive frames. Furthermore the offset position, preferably the top left corner of the region, has to be kept track of. By taking a sub-region of the frame as input for further processing, the resulting coordinates relate to the top left corner of this sub-region as origin. Consequently only the first set of coordinates has the correct origin as reference point for the next frame. Consecutive frames need this information to retrieve the correct sub-region from the full frame by adding the offset to the calculated positional data. First frame ROI calculation is based on a minimum axis-aligned bounding box(AABB) fitted onto the marker. The offset value for the ROI is added to the bounding box values resulting in a rectangle data-set containing the position of the offset top-left corner $\vec{\mathbf{tl}}$ and the offset **width** and **height** of the ROI rectangle:

$$\begin{aligned} \vec{\mathbf{tl_0}} &= (AABB_{x_0} - \text{offset}_x, AABB_{y_0} - \text{offset}_y) \\ \mathbf{width} &= AABBw_0 + (2 \cdot \text{offset}_x) \\ \mathbf{height} &= AABB_{h_0} + (2 \cdot \text{offset}_y) \end{aligned} \tag{6.3}$$

These values are taken to extract the sub-region of the consecutive at $\vec{\mathbf{tl}}$ with size of width and height. The $\vec{\mathbf{tl}}$ value is also saved as offset value $\vec{\mathbf{OffPos}}$. The position

calculation for the proximate frame will utilize the offset value as follows:

$$\begin{aligned} \vec{\mathbf{tl}_{1x}} &= x_1 + \vec{\mathbf{OffPos}}_x - \text{offset}_x \\ \vec{\mathbf{tl}_{1y}} &= y_1 + \vec{\mathbf{OffPos}}_y - \text{offset}_y \\ \mathbf{width} &= AABB_{w_1} + (2 \cdot \text{offset}_x) \\ \mathbf{height} &= AABB_{h_1} + (2 \cdot \text{offset}_y) \end{aligned} \tag{6.4}$$

This should result in the correct positional data for all frames following the initial frame. Finally,the resulting rectangle has to be fitted onto the size boundaries of the full image frame. When applying a fixed offset value, the resulting region that will be extracted from the following frame might reach outside the size boundaries of the full frame. This can happen when the markers are positioned near the edges of the frame. Trying to read out pixels outside the available size of the image will without doubt result into an error by reason of unavailable data at these points. Therefore the combination of position width, height and offset has to be clamped to fit into the available image data:

$$\begin{aligned} \textbf{width} &= (max(0, min(\mathbf{width}, image_{width})) \\ \textbf{height} &= (max(0, min(\mathbf{height}, image_{height})) \end{aligned} \tag{6.5}$$

Another option which could be evaluated instead of clamping the area would be shifting the offset point to a position where it does not violate the image boundaries and thereby preserving the size of the ROI. 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.

6.2.3 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 form 0 to 360 degrees. The Hue value (H)corresponds to the angle on the circle, where $0°$ corresponds to a reddish 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 leas 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 homogeneous 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:

$$\begin{aligned} hsv_{low} &= (hue_{targetcolor} - sensitivity, 100, 100) \\ hsv_{up} &= (hue_{targetcolor} - sensitivity, 100, 100) \end{aligned} \tag{6.6}$$

6.2.4 Mask construction and filtering

These values are the 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. The used cameras run on an automated white balance and exposure mode. The variation of these values can cause shifts in the appearance of the colored markers and therefore alter the generated masks. To compensate for white balance shifting, a non-white ideally diffuse reflecting background should be used for the tracking space. Also the auto modes should be turned off and appropriate values for white balance and exposure have to be determined at the initialization step of the system.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 should be filtered out before any further computation on the image data can be done. The first step in this progress is to use a Gaussian filter to blur the mask.

orig image and mask images for display

The Gaussian filter acts as a low pass filter for the image. The application of the filter cuts out high frequency noise generated by the sensor electronic and other factors. The downside of applying a Blur onto the generated image is a loss of detail. As the markers will not be utilizing complex geometric forms or fine patterns, the loss of details is acceptable.

After this step an erosion and a dilation is applied to the image tho further eliminate unwanted noise. A combination of these two morphological operation is used to further improve the data of the thresholded binary image[**?** , chapter 3.11-12]. Erosion and dilation operation on thresholded images perform the tasks of removing light spots on dark backgrounds as well as dark spots on light backgrounds. These spots are usually the result of the remaining image noise after the blurring or incorrect pixel values from the camera which case the pixel to be incorrectly thresholded. To apply the specific operators to the image a mask (often also called kernel) needs to be defined. These masks normally consist of an n x n shaped matrix where the values in the matrix rows and columns are either one or zero for binary images. An odd number of rows and columns is usually used

to be able to define a center pixel. The central entry of the matrix corresponds to the current pixel of the image for which the morphological operation is to be applied. As the data of the pixel should be preserved, the value is set to one.

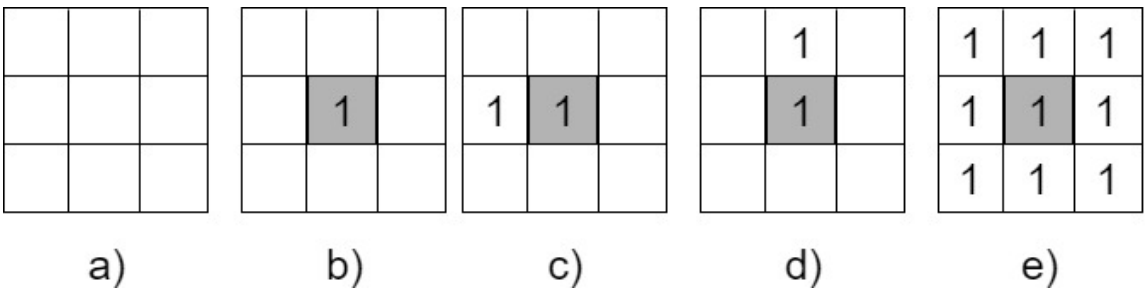


Figure 6.4: Example Mask of 3x3 size :**a)** empty mask **b)** Mask that keeps pixel value **c)** combination of b) with a left shift **d)** combination of b) with an upshift **e)** isotropic mask

The processing of these matrices is not very complex. The central pixel value and the values of the neighboring pixels are taken and summed up.The ones in the matrix outside of the center value define if the pixel is taken in for the operation. A threshold value is then defined. The result of the sum operation is compared to the threshold value. If above the threshold, the central pixel value is altered, depending on the image operation. If below, the current value is kept.

input: threshold $threshVal$ value for the calculation, image segment $imgSegment$ of size $n \times n$, mask $morphMask$ of size $n \times n$, operation value (0 or 1) $opVal$, value of central pixel $centVal$

Result: New value for pixel

for $i \leftarrow 0$ **to** *maskHeight* **do**
  for $j \leftarrow 0$ **to** *maskWidth* **do**
    if $morpMask[i, j] = 1$ **then**
      result ←result $+morpMask[i, j]$
    end
  end
end
if result $>=$ threshVal **then**
  centVal $=$ opVal
end

Algorithm 1: Pseudocode for the the pixel value calcuation

When using for example the isotropic matrix displayed in Figure 6.4 e) and setting a threshold of 8 on an dilation operation, the result would be the filling of a white pixel when surrounded by black pixel. The same operation on an erosion would lead to an removal of a black pixel surrounded by white pixels.

The combination of the morphological operator erode and dilate leads to two methods called *morphological opening* and *morphological closing*. The idea behind these two is to use the two basic operation in a certain order to enhance the binary image quality [**?** , chapter 3.12]. The formal definition for the operation:

$$\begin{aligned} O &= Original\ image \\ \oplus &= Dilation\ operation \\ E &= Erosion\ mask \\ \ominus &= Erosion\ operation \\ D &= Dilation\ mask \\ \circ &= Morphologic\ opening \\ \bullet &= Morphologic\ closing \end{aligned} \tag{6.7}$$

$$O \bullet D = (O \oplus D) \ominus E$$
$$O \circ D = (O \ominus E) \oplus D$$

The *morphological closing* operation is applied first to eliminate so called "salt noise" (white pixels in black area), narrow cracks, channels and small holes in the original image. The major part of the work for this operation is done on the areas where not tracking marker is found. It removes the remaining noise left over from the blur or signal distortion. The application of the *morphological opening* removes so called "pepper noise", fine hairs and small protrusions. The outcome of the practical application of the opening is the removal of wrong pixels inside the area of the color marker in the image. This helps with the later calculation of the bounding boxes as the algorithm searches for the largest closed area int the image. Generally the closing of an image enlarges it while filling bright defects inside of an black area and the opening makes it smaller while removing bright defects on a dark surface. As the system will not be handling large image data, the mask sizes can be kept low, utilizing the above described 3×3 mask size. The structure options provided by *OpenCV* contain a cross structure, an elliptical structure and a isotropic version. As it would be useful to get all surrounding image data and the tracking markers will not have complex geometric structures, the isotropic mask structure should be sufficient. The threshold limit should be set to an initially reasonable value with further testing on the running setup.

6.2.5 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

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

6.2.6 Sending of positional data via network as UPD package

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

6.2.7 object tracking

add object tracking method -¿ visual or HTC vive tracker

6.3 Stereoscopic calculations

On the parent device, the incoming positional data from both Raspberry's is collected and prepared for the stereoscopic calculation steps. The incoming data only consists of 2D position data of the tracked marker on the image plane. To retrieve depth information for correct spatial placement of the marker the image disparity of the two cameras can be utilized to calculate the object distance from the cameras.[**?**]
Before diving into the mathematical concepts, some essential terms have to be explained. These are:

- Interaxial distance
- Parallax
- Zero-point plane

Interaxial distance

A stereoscopic imaging system is based off of our own human visual system as it produces stereoscopic images. Therefore such a system always consists of at leas two cameras which are mounted in parallel. The distance difference between the two cameras is called the "inter axial distance" of the system. The usual distance for such systems is the mean distance of the human eyes, which corresponds to around mm. Shifting the cameras closer together or further apart will change the resulting stereoscopic effect, where shifting the cameras closer together will enlarge objects and shifting the cameras further apart will make objects seem much smaller than they are.

insert mm values

Parallax

Since the cameras are separated by the *interaxial distance*, a disparity in the generated images will occur. This means that the objects in the picture of the right camera will

have a different position in terms of perspective than the images that result from the left camera. This resulting disparity is used to generate the 3D effect when displaying the images. The term "*parallax*" is used to describe the comparison of the two images. The two images can have a negative, positive or zero *parallax*.

Zero-point plane

The *zero point plane* is a special theoretical plane in a stereoscopic system. The location of the *zero-point plane* defines the distance from the camera system at which the image points from both cameras match. Objects that lie before the *zero-point plane* will have a negative *parallax* and will appear closer to the viewer. Ton the opposite side, objects with positive *parallax* will lie behind the zero point plane and therefore will appear further away.

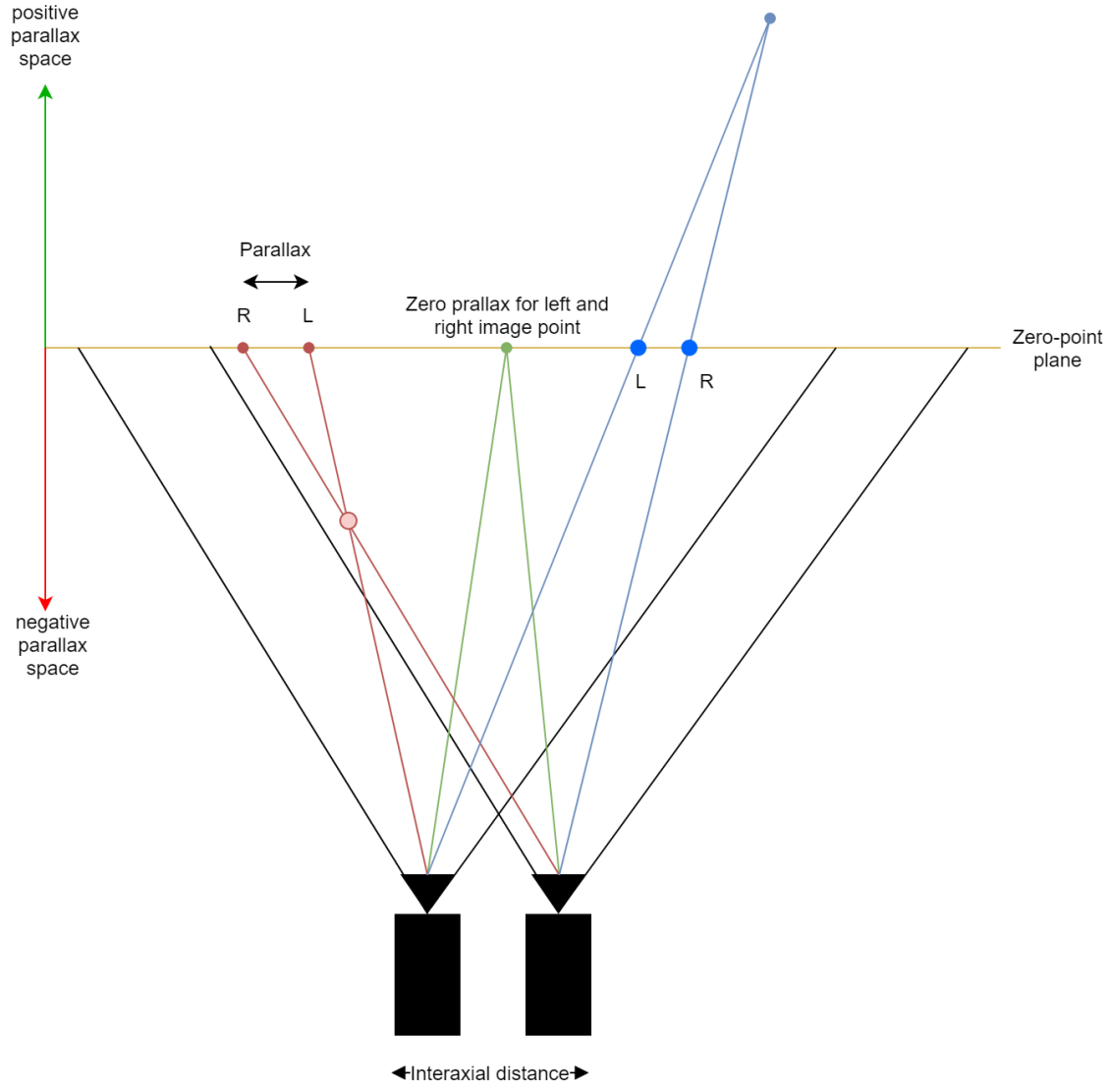


Figure 6.5: Stereoscopic areas

Additional attention needs to be put in the camera setup before calculating depth form disparity. The cameras need to be checked for horizontal alignment to achieve correct values for the calculation.

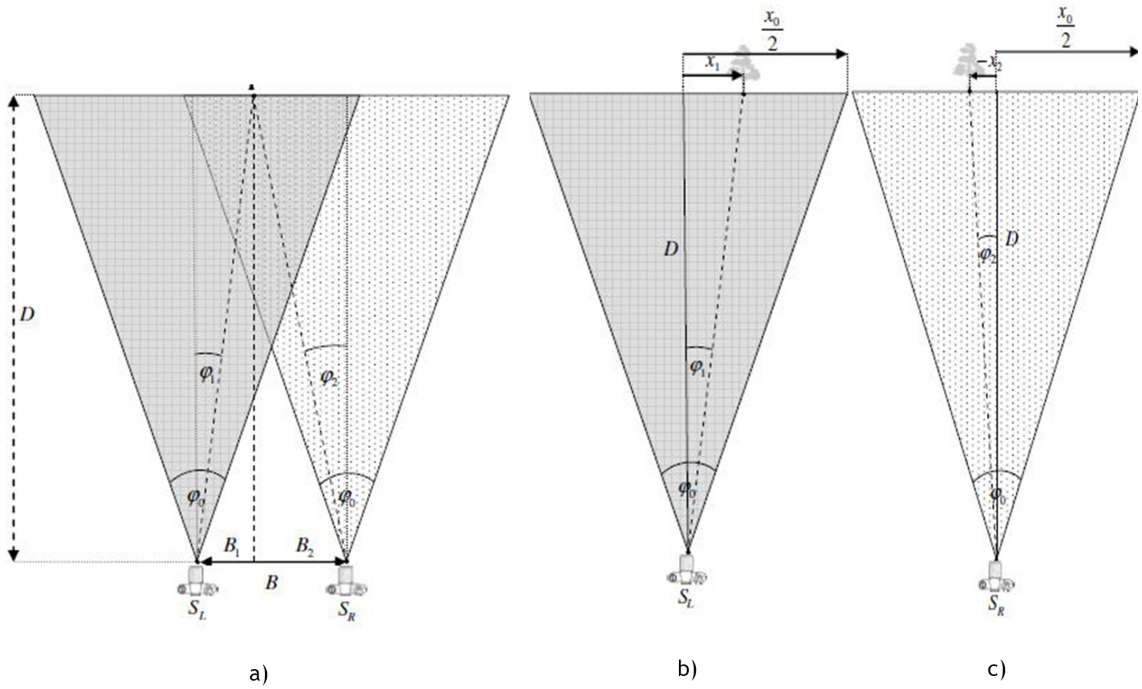
6.3.1 Depth calculation

With correctly aligned system, one can calculate the distance of an object from the cameras using common trigonometric fundamentals[**?** **?**].
The position for the right camera will be denoted as S_r and the position of the left camera will be denoted as S_l (see Figure 6.6a).The resulting distance of $S_l - S_r$ is the *interaxial distance* B of the system. The view angle θ of the Camera, which should be the same for both cameras, will respectively be θ_1 and $\theta 2$. From the camera positioning, the assumption of parallel optical axes is made. The angles of ϕ_l, ϕ_r describe the resulting angle between the optical axis of the camera and the object. By using geometric derivations we can express that:

$$B = B_1 + B_2 = D \tan \varphi_1 + D \tan \varphi_2 \tag{6.8}$$

Rearranging for D gives us:

$$D = \frac{B}{\tan \varphi_1 + \tan \varphi_2} \tag{6.9}$$



a) b) c)

Figure 6.6: Dimmensions for calculation.Images from:[**?**]

With Figure 6.6b-c one can find that:

$$\frac{x_1}{\frac{x_0}{2}} = \frac{tan\varphi_1}{\tan(\frac{\varphi_0}{2})} \tag{6.10}$$

$$\frac{-x_2}{\frac{x_0}{2}} = \frac{tan\varphi_2}{\tan(\frac{\varphi_0}{2})} \tag{6.11}$$

Resulting in the calculation of D to be expressible as:

$$D = \frac{Bx_0}{2\tan(\frac{\varphi_0}{2})(X_L - X_R)} \tag{6.12}$$

Where B is the distance between the cameras,x_0 being the horizontal width of the picture format in pixels, φ_0 representing the viewing angle of the camera and $(X_L - X_R)$ representing the disparity between the two images in Pixels.

7 System Evaluation

7.1 Image processing performance

The image processing of the camera frames on the raspberry can be a significant bottleneck for the system. Real time evaluation demands the processing time of a frame to be in the range of the time of the camera FPS. With the current system setup running at 30fps, this means that the image processing time, containing frame readout and downstream processing of the images, has to be done in about $1/30s$ or 33 ms to achieve "real time" processing.
The first prototype implementation with *OpenCV* in *Python* showed, that this speed was reachable when only tracking one color. The implementation reached around 30-40 ms processing time when scanning whole images every time. An implementation of an *"Region of interest"* featurebroke down the processing time to around 30 ms for a single color. The downside of the implementation was revealed when implementing the other 4 colors needed to do a full five finger tracking. This approach ramped-up up processing time to around 100 ms per frame, making this solution run at 10 FPS.
messwerttabelle
The sequential algorithm also showed another design flaw. When not utilizing the ROI approach, the algorithm would scan the whole grabbed camera frame for each color to create the corresponding threshold map. The *OpenCV* color threshold methods are designed to only search for one color threshold at the time, therefore needing this procedure. One solution option is the already mentioned ROI usage. This solution would furthermore benefit from a multi threaded implementation, as we are not manipulating the original image. Parallel memory readout is not a problem and the generated mask can be saved separately.

Another approach which could help speed up the process is implementing an own method for thresholding the grabbed frame with the possibility to do all the needed color thresholds in one image loop. With this solution, the overhead would be reduced by 4 image iterations, thresholding operations would stay the same as these still have to be run on each pixel. The output of this method would then return the 5 needed threshold mask for further processing.
Another performance issue that arose from the first prototype was that the input camera frames came RGB coded. For better options of color separations, the image was initially converted into HSV colorspace. This operation turned out to be second longest operation after frame grabbing. Testing of the cameras showed that the color segmentation in the

original RGB images of the cameras are good enough for the used test colors, causing this step to be omitted and shaving off several milliseconds of processing time. The rest of the processing steps were lying in the sub millisecond range and are therefore not as valuable for performance optimization. The results from this prototype showed, that the *OpenCV-Python* combination is not suitable for reaching the 30 ms target time. *Python* is not a very performant language in this area. The OpenCV library used by python is actually just the ported version of the C code library with bindings for python. As *C* and *C++* are more hardware near and therefore more performant, the second prototype implementation was done in *C++*. The usage of *C++* as the programming language provides more control over memory management and therefore reduces unnecessary memory data copies as these can be handled by pointers to memory locations.

insert time

The ported version of the Python code to *C++* initially showed similar performance measurements when using the sequential approach. This was to be expected, as it is the same code the python bindings are using. Further investigation into code timings showed, that another bottleneck was the image optimization feature which applied an erosion an dilation to remove high frequency noise in the image. This operation brought the time up to 100 ms processing time when processing the whole frame area, making the algorithm not usable for "real time" application. Removing this feature brought a major speed up in processing time. Furthermore, the implementation of a thread pool, which handles all the color detection jobs, brought down the calculation time for 5 colors to around 120-140 ms. Although these times are still not usable for real time, it brought a significant performance boost.

test image correction time

Test of using lower resolution than 640x480 showed that the calculation time reduced furthermore. This indicated that the ROI idea would be feasible for further performance optimization. A test at a frame resolution of 320x240 pixels reduced the processing time for all five colors to below 10 ms.

Utilization of the knowledge from these tests led to the implementation of the aforementioned "region of interest" feature (see Figure 7.1). The first frame is analyzed as a whole frame, optimally resulting in the detection of a marker. As the marker detection creates a rectangle around the tracked area, we already have the coordinates for the ROI and only have to add an offset to this area. This value of the offset has to be set high enough to ensure that in the following frame, the tracking marker is still contained in the ROI. It also has to be ensured, that the generated ROI is clamped to the frame dimensions. If not doing so, parts of the readout are can be located outside of the image plane, causing readout errors when used for the subsequent frame. The following frames will then be processed with the input of the ROI, selecting only the defined section of the image and updating the ROI with the new data for that frame.

As already mentioned in the conceptional phase to lighting of the tracking area is a major factor. Variations of lighting intensity and color temperature cause the color of

the trackers to shift their color. This can cause a fluctuation in the calculated tracker positions, as the defined color threshold ranges are kept as small as possible to achieve a clean separation of the tracker colors and reduce unwanted noise.

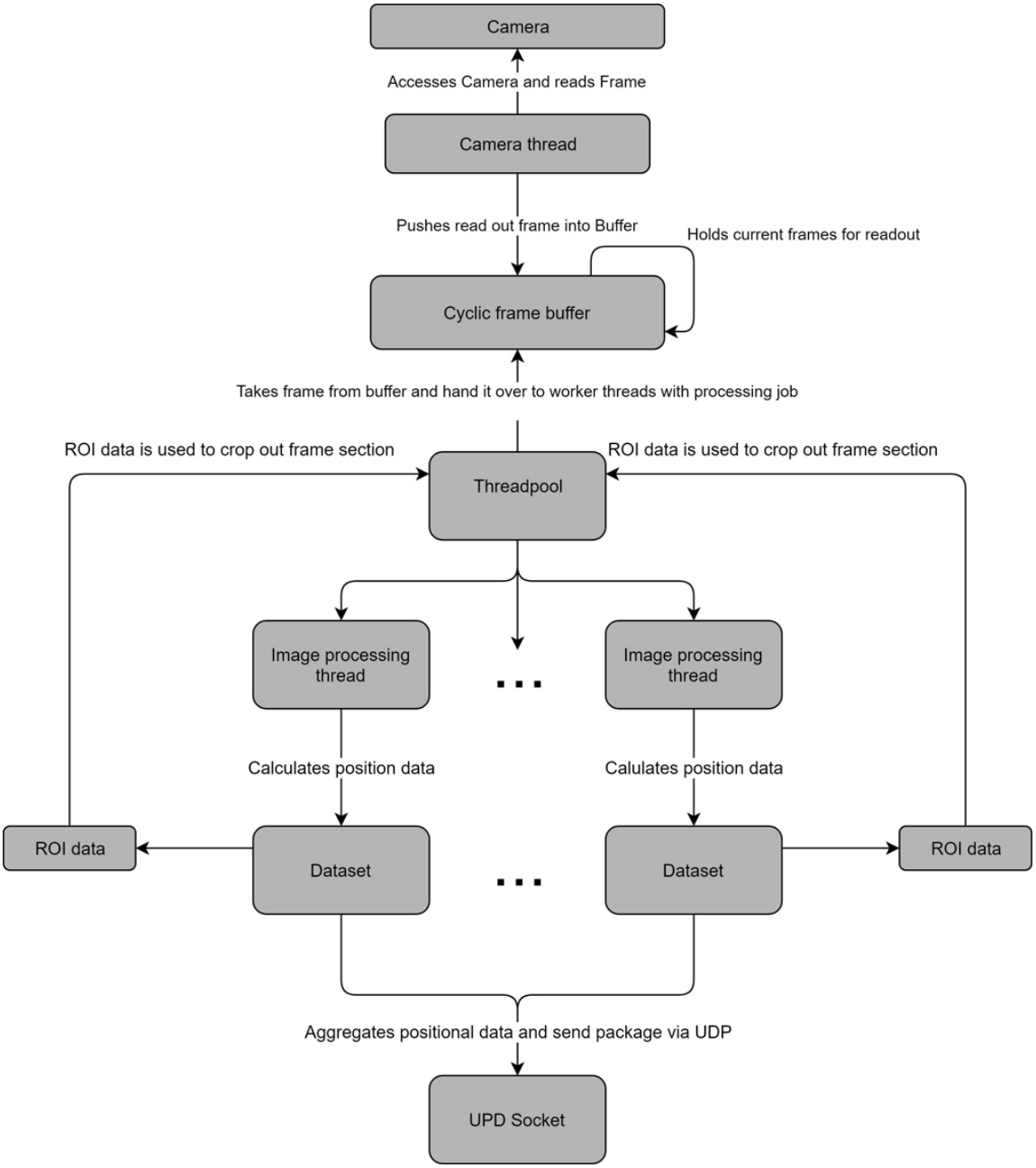


Figure 7.1: Work flow of the C++ implementation with thread pool solution and ROI

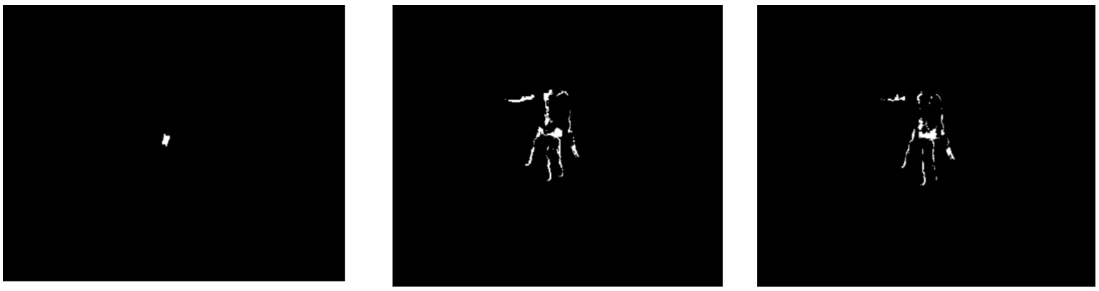
Table 7.1: Speed measurements for the used implementations

Timings in ms	Python Single Thread without ROI	C++ Single Thread without ROI	C++ Single Threadpool Thread with ROI
Frame Reading			
HSV Conversion	50.852	41.125	omitted
Image Blur		147.239	varies with ROI size
Mask erode/dilate		165.663	varies with ROI size
Color Threshold		14.686	
Position Calculation			
ROI Readout			

7.1.1 ROI size and color accuracy

System testing with the 5 different colored shrink tubing markers showed, that the color tracking for colors outside of the color range of the human skin tones (green,blue) is rather unproblematic. The color markers falling into the colorspace of posible light reflection from skin may cause probelms, dependng on the lighting situation of the setup.

elaborate



a) b) c)

Figure 7.2: Images showing reflection problems with orange color thresholding. a) Correct threshold with only the marker being thresholded b-c) Depending on the rotaton of the hand, skin reflecton color falls into tracking threshold.

Figure 7.2 displays such a case where depending on the orientation of the hand, refletion from the scenery lightng can fall int the color tone space of the tracked marker, causng large parts of the hand to light up in the image.Since the calculation of the trackng marker position relys on finding the largest closed "white" area in the thresholded image, this can cause a "jumping" of the calculated position.

The first solution for this problem is to narrow down the color threshold spans. By doing so, one can ensure that only the wanted colors are tracked. This method does have a downside effect. When the markers are moved in the tracking space, light variations and shadowing will cause the colors to shift their appearance slightly. With broader threshold ranges, this does not seem to be a large problem as these changes still lie inside

the ranges.Narrowed down color ranges might not include these color variations, creating cavities in the thresholded image.
The applied morphological operations on the image can fix these problems up to a certain point, but for larger areas this could vary position tracking results.The implemented ROI feature speed up the whole system calculations once the colors are found. Before this point, the system scans whole frames to find colors, which takes more tme than the much smaller ROI regions. Empirical evaluations showed, that for the used tracking space, a ROI region size offset of 40 px in x and y directions produces the best results in terms of consistency. Smaller region offsets caused the system to use tracking for faster hand motion, which causes system slow down until the tracking has recovered. Generally, larger ROI's produce more constent tracking results at the cost of hgher calculation time.

The initial color marker size was selecetd to be relatively large, spanning most of the last segemnt size of each finger. The used material of the colored shrink tubing was selected because it seemed to be relatively diffuse und therefore unprone to errors resulting from the reflection of the scene lightng. The material turned out to be mostly diffuse reflecting, but dependng on color and angle of light incidence reflections in the color of the scene light did occur.
A test set of smaller marker was also evaluated, showng similar tracking results to the larger markers and also the same color reflection problems. The benefit of a smaller marker size would be much lower restiction of the fnger movements and also unveiling the fingerstips of the fingers for better haptic interaction. As both markers showed the above described problem of mixing marker area with possble skin reflections, and improvement to the markers was applie. At both ends of the larger markers, a piece of gray colored tape was applied to create an artificial border for the tracking algortihm.

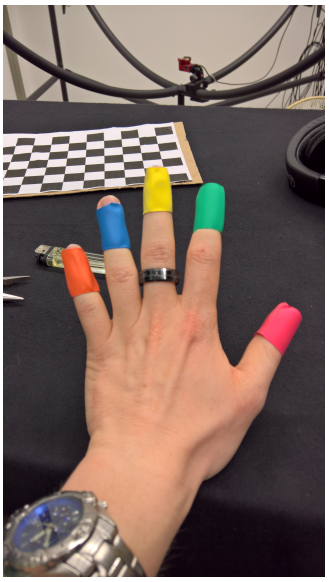


Figure 7.3: First set of shrink tube color markers

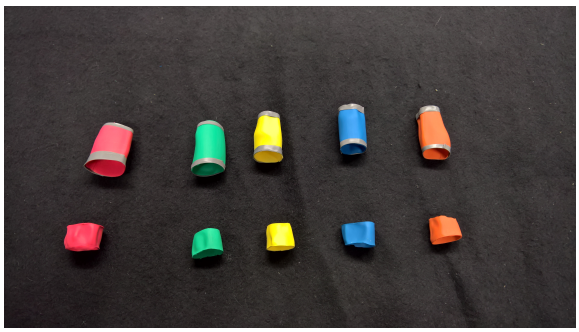


Figure 7.4: First set of shrink tube color markers

The idea behind this solution was to create a fixed border between the color area of the tracker and the color area of the skin tones. The grey color of the border ensures that this area will not be tracked. The usage of black instead of gray would evenn furthermore ensure this.
Since the tresholding algorithm searches for the largest closed color area in the bnary thresholded image, this border should cause a seperation of unwanted skin reflecton area and actual tracker color area. The problem that the resultng color marker area might not be the largest area still presists after this improvement but this case has to be handeled seperately.
Optical results showed that for the colors lying outside of the red to orange color spectrum that skin reflections produce, the artificial border stabilized the color detection.
The idea of usng shrink tubing as marker material showed to be suitable except in the cases described above. One downside of the material is that the displayed colors are already the highest variety of available colors on the market. Other colors from the green and blue color areas like purple are simply not available or have to be ordered as a special request in larger amounts.
Also the original diameter is much larger than a human finger. The used shrnk tube was able tu be shrunk to one thrd of its size. The reason for this was to be able to adapt to the variyng dameters of the human hand. The shrnking procedure requires a constan amount of heat to be applied to the material to activate the shrinking. The amount of heat that is needed surpasses the heat production of a standard hair dryer since the original application of shrink tubng is on heat resistant cables. The heat that needs to be applied can easily be more than 100 gegree Celsius, which makes a direct application on the users hand highly insecure.
For the prototype, the shrink tube parts were shrunk with a standard cigarette lighter and continously fited onto the user finger until the desired form was reached. This method showed to be suboptimal, since the flame of the lighter produces only a puntual heat source. Ths caused uneven shrinkage of the heated parts. This can lead to cavities or bumps in the material and change the refelection properties at these points. A regulated heat gun would probaly generate better results. test
To get smoother shrinking results, parts of pvc tubng with the correct diameter could be used as dummy parts for the man part of the shrinking.

7.1.2 Depth measurement accuracy

To determine the accuracy of the system for it's depth measurement values a simple test bench setup was used. The camera rig was aligned horizontally and fixed to the testbench. A large sized colored markerwas used as target for detection. The marker marker was positioned at altering distances from the camera rig along the central axis of the camera rig. The height at which the camera rig is positioned in the experiment setup will be around 100 cm, so the measured distances started at 100 cm from the camera and were decremented in steps of 5 cm until 20 cm in front of the Camera Rig(see Appendix Table 9.1 and Figure 7.5).

measure size

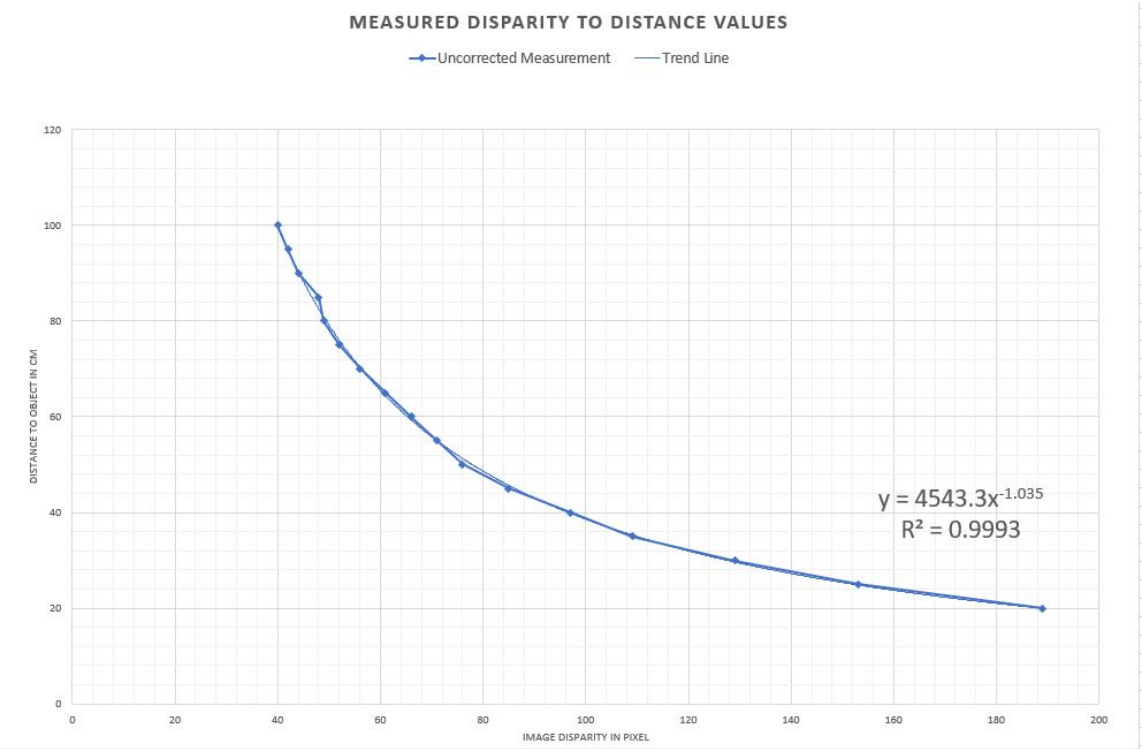


Figure 7.5: Chart showing the results of the disparity measurement with a plotted trend line

The accuracy of the camera system is limited by the number of pixels in relation to the camera view angle:

$$\Delta\varphi = \frac{\varphi_0}{x_0} \tag{7.1}$$

With the system set up at 640 pixels of image width and a horizontal viewing angle of $62.5°$, $\Delta\varphi$ is equal to $0.0977\frac{1°}{pixel}$. The resulting error would be:

raspi docu reference

$$\frac{\tan\varphi}{\tan(\varphi - \Delta\varphi)} = \frac{\Delta D + D}{D} \tag{7.2}$$

$$\Delta D = \frac{D^2}{B} tan(\Delta\varphi) \tag{7.3}$$

The system error for a $D = 100cm$ would therefore result in about $2cm$ of possible error. As the measured values show deviations of up to 5% from the correct value, the function from the trend line displayed in Figure 7.5 was taken as a correcting factor for the depth calculation.

Equation 7.5 uses the uncorrected values for the calculation. Values lying on the trend line should correspond more exactly to the correct distance values [**?**]. The equation can be rewritten to:

$$D = k * x^z \tag{7.4}$$

with K being:

$$k = \frac{Bx_0}{2\tan(\frac{\varphi_0}{2} + \phi)} \tag{7.5}$$

and x the disparity in pixels. The ϕ term in the equation above is used as a compensation for possible alignment errors. The trend line, which is fit to the measurements in Figure 7.5 represents the the function needed to fulfill Equation 7.4.The calculated values are $k = 4543.3$ and $z = -1.035$. With the utilization of the corrected value formula, the accuracy of the depth measurement results is acceptable for the prototype applicaton. For future work, the distance measuremetn procedure could be applied with more measurement points to refine the resulting function for more precision.

add chart for corected values

8 Resume

Future work:
-improve depth tracking value preciion through a more elaborate calibration seesion where the depth correcton algorith is further optimized
-improve image reading source code to accept hgher framerates. The hardware capacbilities provide image reading frequences of up to 90fps, current bottleneck is the c++ implementaion which only provides 30 fps max.
-When utilizng higher framerates, the multiproceesng approach will be viable to gain performance. Optimizaton work on the implementaion can be done.
-Image manipulaton on the cpu is rather costly, even at small image sizes. The raspberry Pi has a "small" gpu unit onboard, whch is in dle state when the device is run in headless mode. The gpu could then be utilzed to do heavy weight image calculation such as stereo rectification.
-Optimization pont would also be creating a mor optimized verson of the color thresholding algorithm where the threshoding for all of the thracking colors is done in one run on the current frame. It is also to be evaluated if ths problem falls can be expressed as a SIMD functon and would benefit from the processng on the gpu.
The current prototype is still missing an graphcal user interface on the display side. Further communication of states on the slave Pi's to the master unit would also be beneficial. The habd data that is used for prototyping is still hardcoded and only manually configurable. A calibration procedure for the system as well as a suitable data format for representing and translating these values for the IK algortihm s still to be done.
-Shrnk tubing showed to be a utilzable material for the usecase. Downsinde of the material is that the colors used are the largest spectrum of colors available on the market and geting outer colors is relatively hard. Other materal shuold be evaluated.

List of Figures

List of Tables

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

9 Appendix

Table 9.1: Depth measurement Values

Real distance from camera (cm)	Left camera x-value	Right camera x-value	Disparity in pixels	Calculated distance in cm	Deviation in %(abs.)	Corrected Values	Deviation in %(abs.)
100	302	342	40	99.463	0.54	99.670	0.33
95	301	343	42	94.727	0.29	94.760	0.25
90	300	344	44	90.421	0.47	90.304	0.34
85	298	346	48	82.886	2.49	82.523	-2.91
80	296	345	49	81.194	1.49	80.780	0.98
75	296	348	52	76.510	2.01	75.960	1.28
70	293	349	56	71.045	1.49	70.349	0.5
65	289	350	61	65.222	0.34	64.388	0.94
60	289	355	66	60.281	0.47	59.344	1.09
55	286	357	71	56.036	1.88	55.022	0.04
50	285	361	76	52.349	4.7	51.279	2.56
45	280	365	85	46.806	4.01	45.668	1.48
40	265	362	97	41.016	2.54	39.831	0.42
35	258	367	109	36.5	4.29	35.300	0.86
30	244	373	129	30.841	2.80	29.650	1.17
25	288	381	153	26.003	4.01	24.848	0.61
20	206	395	189	21.050	5.25	19.965	0.17