Drucksachenkategorie

Implementation of a 3D Hand Pose Estimation System

John-Certus Lack

DLR

Institut für Robotik und Mechatronik	BJ.: 2014
	IB.Nr.: 572-2014-22

BACHELORARBEIT

IMPLEMENTATION OF A 3D HAND POSE ESTIMATION SYSTEM

Freigabe: Der Bearbeiter:

John-Certus Lack _____

Betreuer:

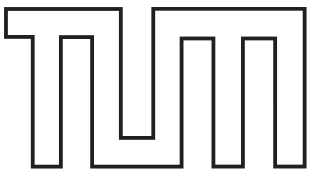
Dr.-Ing. Ulrike Thomas _____

Der Institutsdirektor

Dr. Alin Albu-Schäffer _____

Dieser Bericht enthält 65 Seiten, 41 Abbildungen und 7 Tabellen

Ort: Oberpfaffenhofen	Datum: 28.08.2014	Bearbeiter: John-Certus Lack	Zeichen: 74703



Technische Universität München
Institute for Media Technology
Prof. Dr.-Ing. Eckehard Steinbach

Bachelor Thesis

Implementation of a 3D Hand Pose Estimation System

Author:	John-Certus Lack
Matriculation Number:	03620284
Address:	Steinickeweg 7
	80798 Munich
Advisor:	M.Sc. Nicolas Alt (TUM)
	Dr.-Ing. Ulrike Thomas (DLR)
Begin:	01.04.2014
End:	11.09.2014

Abstract

Chair of Media Technology

Bachelor of Science

Implementation of a 3D Hand Pose Estimation System

by John-Certus Lack

The variety of means for human-computer interaction has experienced a surge in recent years and currently covers several methods for haptic communication and voice control. These technologies have proven to be a vital tool for information exchange and are already a fundamental part of daily human interaction. However, they are mainly restricted to a two dimensional input and the interpretation of formulated commands. To overcome these limitations, several approaches of 3D hand pose detection systems were made that often imply computation intensive models or large databases with synthetic training sets. Therefore, this thesis presents a 3D hand pose estimation system that provides real-time processing at low computational cost.

The first parts describes methods for the hand-background segmentation and the tracking of specific feature points, which include the center of the palm and the fingertips. In the second part, a novel approach is proposed for inferring the position of finger base joints and numbering the correlated fingertip-joint clusters. This method clearly distinguishes from other approaches, as it deduces the joint location from the relative configuration of finger clusters to the center of the palm, omitting the need for complex models or databases. Finally, all obtained information is combined to calculate the angles at each joint and to drive a 3D simulation model, which illustrates the imitated user hand pose.

Results presented in the evaluation show that this approach is well suited for hand pose estimation, as it achieves high accuracy in detection and additionally enables real-time interaction. Precision and stability can further be improved by upgrading the utilized camera to a system with higher resolution. This modification also allows considering multiple feature points for a more differentiated angle calculation. Overall, the user feedback was very positive and all test participants mentioned an incline of their success rate over time.

Acknowledgements

Foremost, I would like to express my sincere gratitude to my supervisor at the German Aerospace Center Dr.-Ing. Ulrike Thomas for her encouragement, patience and helpful advice. Furthermore I want to thank my supervisor at the TU Munich M.Sc. Nicolas Alt and professor Dr.-Ing. Eckehard Steinbach at the chair of media technology for their support and for the possibility of writing my thesis in cooperation with the TU Munich and the German Aerospace Center.

Especially I want to express deepest gratitude to Prof. Dr. Alin Albu-Schäffer, Dipl.-Ing.(FH) Peter Meusel and Dr.-Ing. Ulrike Thomas for enabling me to present my bachelor's thesis at the Automatica 2014, the world's largest fair for robotics, assembly systems and machine-vision solutions.

I thank my fellow student colleagues for their continuous support, creative input and for all the fun we could share together.

Finally, I thank my friends and family that supported me at all times with their encouragement and humor. Among these I especially want to highlight my brother John-Christopher Lack.

Contents

Chapter 1

Introduction

Any sufficiently advanced technology is equivalent to magic.

— Sir Arthur C. Clarke

In recent years and especially the past decade, technological developments have been thriving exponentially and throughout the globally connected world innovations have fundamentally influenced interacting with the environment. The human perception of this environment has experienced a shift from a purely object and structure based awareness - understanding the world as construct of buildings, machines and nature - to a more functional and interactive concept with greater focus on information exchange. The increasing integration of technical systems into every day life constitutes a major part of this concept. These systems have immensely emerged in the field of communication, where they progressively affect inter-human communication and their interaction with the environment. A major contribution to this development is the intuitive design of such interfaces, allowing a more natural operation and enhancing their integration.
The current landscape of novel communication systems includes two that stand out amongst others: multi-touch devices, which have been already completely integrated in daily usage, and voice control technology, which is successfully promoted in solutions such as *Google Voice, Siri* (*Apple*) or the *Connected Drive* system (*BMW*).
However, a new channel which to date has not yet been developed to a similar extent is 3D contactless motion control. This technology offers a promising solution where more natural forms of interaction are demanded. It comprises several advantages over other methods and is therefore seen to have great potential of improving current means of interaction. Current approaches to this field include solutions by *Leap Motion* [Lea], *Google-Flutter* [Goo] and *Intel-Omek* [Int].

Motivation

A major motivation for contactless motion control is the faster exchange of information. Therefore single process steps do not have to be elaborately described, but can be directly translated from the interpretation of movements and gestures. Additionally, this approach provides higher flexibility and more possibilities, as greater amounts of data can be transferred from actions performed in 3D space. Finally, it optimally empowers the user to interact with a technological system in the most natural way, without requiring him to learn a set of specific commands or the need to understand a complex system design.

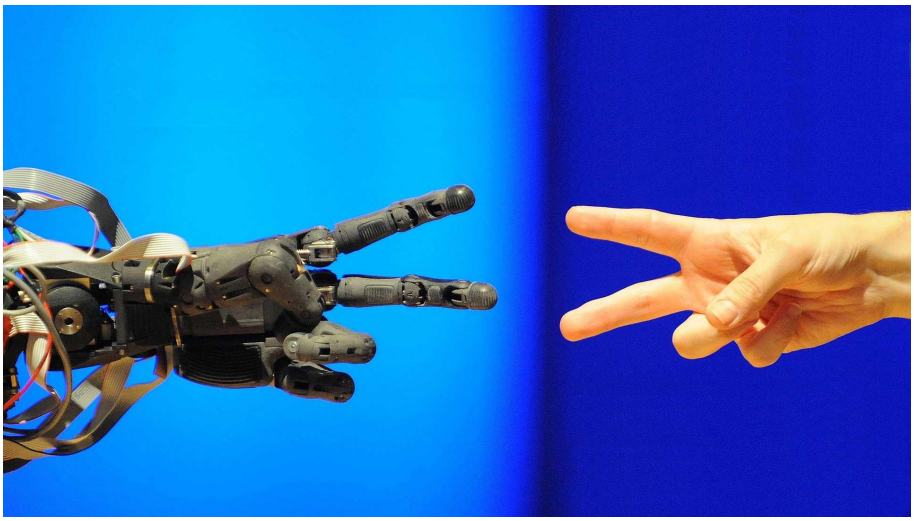
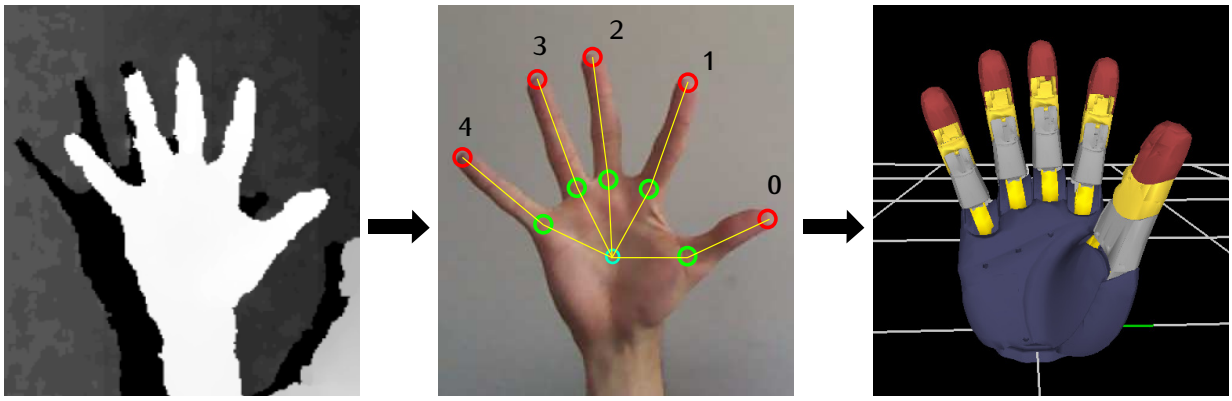


Figure 1.1: Gesture control of robots [GRC]

Such a 3D hand pose detection system can be utilized for several applications. A possible implementation could be within the field of robotics, i.e. in the process step of teaching robots certain gestures or movements (Fig. 1.1). Currently this stage is not very efficient and requires an elaborate definition of movements trajectories, however, it could be solved by inferring this learning process from the detected hand motions. Another utilization could be in the 3D manipulation of objects in the virtual space. If the detection of 3D coordinates of feature points, like fingertips or entire finger clusters, can be performed sufficiently accurately, the efficiency of modelling 3D structures could be boosted significantly. A third application could be integrated in the telepresence control of robots. Combining these technologies with the enhancement of augmented reality, robots can be deployed with remote control in scenarios of grasping and assembly tasks, thus allowing higher flexibility in solving complex mechanical situations.

Goal of this thesis

The work presented in this thesis proposes a 3D hand pose estimation system that translates the received data to a 3D simulation model, which in turn imitates the performed movements. The challenge of achieving a pose estimation lies in detecting 3D coordinates of certain feature points, such as fingertips and joints. Additionally angles at respective joints need to be accurately approximated to infer the according hand configuration. This process is illustrated by the images in figure 1.2. The processed information is entirely based on the color and depth output of a *Kinect* camera without applying any additional markers to the hand.



(a) Kinect: depth image (b) 3D feature points (c) 3D simulation

Figure 1.2: Development of the 3D hand pose estimation system

The thesis is organized as follows: In the following chapter 2, similar solutions and concepts are described and contrasted in terms of their respective approach of detection. Chapter 3 illustrates in detail the consecutive processing stages and explains essential algorithms that are utilized to achieve the implementation. As this chapter presents the major contribution, it is therefore divided in several sections, which are thoroughly elucidated. Chapter 4 gives an overview of the employed hardware setup. Chapter 5 gives an evaluation of the experiment. It covers the validation of the achieved accuracy in tracking and movement detection and provides the evaluation of user feedback after participating in the tests. Chapter 6 gives a discussion of all experimental results as a review of the accomplished outcome and compares it with the development of every processing step.

Chapter 2

Related Work

This chapter covers major common approaches to hand detection and analyses their current configuration regarding finger positions and angles between finger joints. Obtaining this information can be roughly divided into two parts. First, the 2D hand segmentation needs to be performed to locate the hand position and then the 3D hand pose estimation can be applied. The second step can be solved by a variety of different approaches and is therefore addressed in more detail.

2.1 2D Hand Detection

Attempts to locate the position of a hand in an image require either certain restrictions to the environment or knowledge of the prior frame. A common technique applied by [WP12] is to segment the hand by detecting skin color pixels within a certain color range. To receive more accurate results the color space is transformed from RGB to HSV, with H, the "Hue" value, representing an angle in the color wheel, S representing the "saturation" of this color and V corresponding to the "value" or brightness of the perceived color.

Van den Bergh et al. [VdBK09] enhance this procedure by integrating Gaussian mixture models and weighting each pixel according to its likelihood of resembling a skin color. Utilizing this correlation, threshold filters extract the relevant points and discard pixel exceeding defined values. This approach yields good results given conditioned environments with stable lighting and only minor influence of shadows covering parts of the hand. However, these systems are highly susceptible to unstable lighting conditions [SMHL00] and measures to consider these changes, like expanding the threshold boundaries, result in misinterpretation of other objects with similar color.
The approach implemented in this thesis is inferred from the depth-based hand detection method by Sung-il Joo et al. [JWC14]. This has the advantage of being independent of lighting conditions, therefore providing higher accuracy and more stable results.

2.2 3D Hand Pose Tracking

The goal of this step is to establish a logical connection of object positions between frames over time. Common tracking methods that utilize such temporal links can mainly be split into two categories: model-based and appearance-based tracking. The model-based approach relates the received image information to an underlying geometric model and matches the detected hand configuration with the best possible fit of the model, to varying angle parameters and considers kinematic boundary constraints. The appearance-based approach in contrast dispenses with a model and relies entirely on synthetic training data, which contains several hand postures, observed from different viewpoints to increase the accuracy of detection.

2.2.1 Model Based Approach

One example of model based tracking is described by Stenger et al. [SMC01]. Their system utilizes a highly accurate 3D hand model consisting of 39 truncated quadrics with 27 degrees of freedom. This enables generation of 2D profiles and provides a reliable method to handle self-occlusions. In order to minimize the error between computed profiles and observed edges from the original image, the hand pose is refined with an *Unscented Kalman Filter*. The modulation over time is then illustrated in a smooth simulation with approximately 3 frames per second. Ballan et al. present a multi-view approach [BTG$^+$12] that combines the input of eight cameras to a hand motion capture system. This allows detection of every DOF and enables capturing the interaction of two hands, while dealing with partial self-occlusions. In order to establish correspondence between the hand model and the input image salient points on fingers are highlighted via pre-trained classifiers and combined with edge detection and optical flow. Another approach utilizing the multi-view method is proposed by Ho M. Tseng in [HTLH11]. To overcome the self-occlusion problem, two cameras are set up respectively covering frontal and side view. As this generates a high dimensional parameter space of hand poses, a separable state based particle filter is implemented to reduce computational complexity. To summarize, the model based approach gives good results and high reliability, but also demands high computational resources due to the complexity of its design.

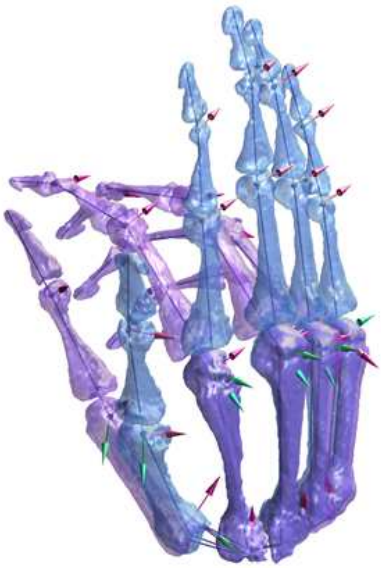


Figure 2.1: Kinematic hand model designed by DLR [Hana]

2.2.2 Appearance Based Approach

This approach is formulated by Doliotis et al. [DAKP12] as an image database retrieval problem. A large database of labelled synthetic training sets is utilized to infer the best match of poses to the input hand image. The ground truth labels of the identified matches serve as hand pose estimates from the input image. The database covers 20 hand shape prototypes, which are rendered from 4128 views, hence generating a total of 82560 images. To determine the best possible fit of synthetic data to input image, the chamfer distance and pixel-wise Euclidean distance are calculated and the image with lowest values chosen. However, this system performs not in real-time and accuracy is still low. A similar method addressing these issues is proposed by T.K Kim et al. [TYK13] as semi-supervised transductive regression forest. One of the main contributions of this work is the realistic-synthetic fusion. This combines the advantages of a synthetic database, to cover a wide range of poses by direct comparison, with a higher accuracy in pose detection, by processing realistic data. Another improvement is the reduction of labelling cost with a semi-supervised learning algorithm, which learns the relationship between sparsely labelled realistic data and a large synthetic dataset. Furthermore, transductive transfer learning is employed to preserve association of inter-domain data pairs, when labelling of realistic data is too costly to obtain. Another major contribution is the implementation of an efficient kinematic joint refinement algorithm to handle occlusions and input noise. In addition to the appearance database a greater joint database is generated, containing only its 3D positions, to obtain a maximum pose coverage. This allows recovering occluded joints and missing pixel information, due to noisy depth data. To summarize, appearance based tracking delivers results in real-time with a lower accuracy than model based methods.

2.3 3D Hand Pose Estimation

The goal of this step is to estimate hand configurations, regarding finger positions and joint angles, as exactly as possible. Several approaches have been developed, with increasing interest in real-time compatibility. Unlike the before mentioned tracking methods, speed is increased by restricting the complexity of the hand model by reducing the degrees of freedom. According to Lin et al. [LWH00], models could have up to 30 degrees of freedom, but due to their infeasible computational demand, partial estimations have become more popular. Sridhar et al. [SOT13] propose a part-based pose retrieval method with a generative pose estimation method to increase efficiency. The generative pose estimator computes a possible configuration of the underlying hand skeleton from RGB images and the part-based pose estimator uses fingertip detection exclusively on depth data to estimate a certain pose. The detection of these feature points can be accomplished, however, in several ways. Bader et al. [BRB09] propose the extraction of fingertips based on processing hand contours. This includes the assumption that fingertips always touch the convex hull of the hand contour, which is a good approximation, but not always true. Another method, presented by Chaudhary et al. [CRDR11] analyses the density of skin color pixels and infers the position of fingertips from the histogram of the binary silhouette. First the wrist is identified as abrupt decline in the histogram, implying the hand orientation, and then the fingertips are derived as local minima in the histogram function.
In order to estimate the entire hand configuration the position of the palm also needs to be determined. This is accomplished by calculating the center of mass of the hand. According to Koiki et al. [KK01] this method is very volatile and will result in an unstable prediction. Therefore he suggests eroding the binary image to eliminate the fingers and to reduce the palm to a sufficiently small area to receive more stable results.
Although partial estimations are more efficient and require lower hardware performance, there are attempts to implement full hand pose estimation. Oikonomidis et al. [OKA11] present a system that recovers the full hand articulation, orientation and 3D position. First, the system acquires observation of the hand from a static camera network and extracts reference features from color and edge detection. Then, the observations are hypothesized, based on a human hand model, that is represented with 26 DOF. For each of these, color and edge feature maps are generated and compared with their reference parts. To minimize the discrepancy a *Particle Swarm Optimization* algorithm is applied. This approach is very accurate, but incorporates computational expensive processes, which either need to be outsourced to a GPU or parallelized to achieve adequate levels of performance.

Chapter 3

Approach

This chapter covers the methodology of the hand pose estimator and describes the development of each processing step. Every section represents a building block adding to the foundation of the basic hand structure and each one consecutively depending on the former operation stage. This concatenation of dependencies produces the logical processing chain, illustrated in figure 3.1.

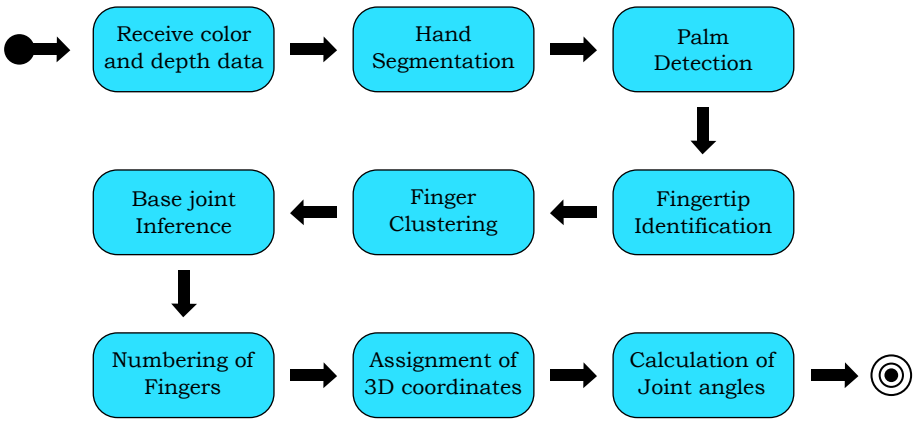


Figure 3.1: Processing chain

The first section presents the two hand models utilized in this thesis and lays the conceptual groundwork as a frame for all further considerations. Section 3.2 explains how the position of the hand is detected and how it is segmented from the background image. After extracting the hand from the background, the center of the palm needs to be estimated in order to establish a constant point of reference and later infer the finger base joint coordinates (Sec. 3.3). The next stage (Sec. 3.4) deals with information regarding finger detection and covers the identification of finger tips (Sec. 3.4.1), the clustering of these to logical units (Sec. 3.4.2), the inference of base joint positions (Sec. 3.4.3)

and the assignment of identification numbers to maintain a consistent mapping (Sec. 3.4.4). After completing all computations on the 2D layer, the processing is elevated to the third dimension (Sec. 3.5.1). Based on these results, 3D coordinates are assigned to further calculate the joint angles (Sec. 3.5.2). Finally, in section 3.6 some additional considerations are applied to stabilize the program and improve accuracy.

3.1 Hand Model

This section represents the framework of this project and describes building blocks, which are used in outlining the development of the program. Each model depicts the hand configuration for a different dimensionality, 2D and 3D respectively. They are both employed consecutively to accomplish the goal of a hand pose estimator.

3.1.1 2D Model for Calculations

As outlined in chapter 2, a complete model based approach for hand tracking is hardly feasible in real-time applications. Utilizing models, similar to the one illustrated in figure 2.1, requires too high computational resources. Therefore, a restricted definition is chosen for these computations. The proposed hand model consists of a palm and five fingers, which in turn are made up of one fingertip and one finger base joint. Without loss of generality, the palm is approximated as a circle around the palm center with a radius slightly larger than the minimal distance from the center to the closest background pixel. The single fingers each are clustered to logical units and contain the tips as furthest points from the palm center, within the cluster, as well as the base joints, which are inferred roughly at the intersection of this cluster and the palm contour. Figure 3.2 illustrates a geometrical representation of the hand model. This model comprises all major feature points, needed for an accurate hand pose estimation and enables real-time processing.

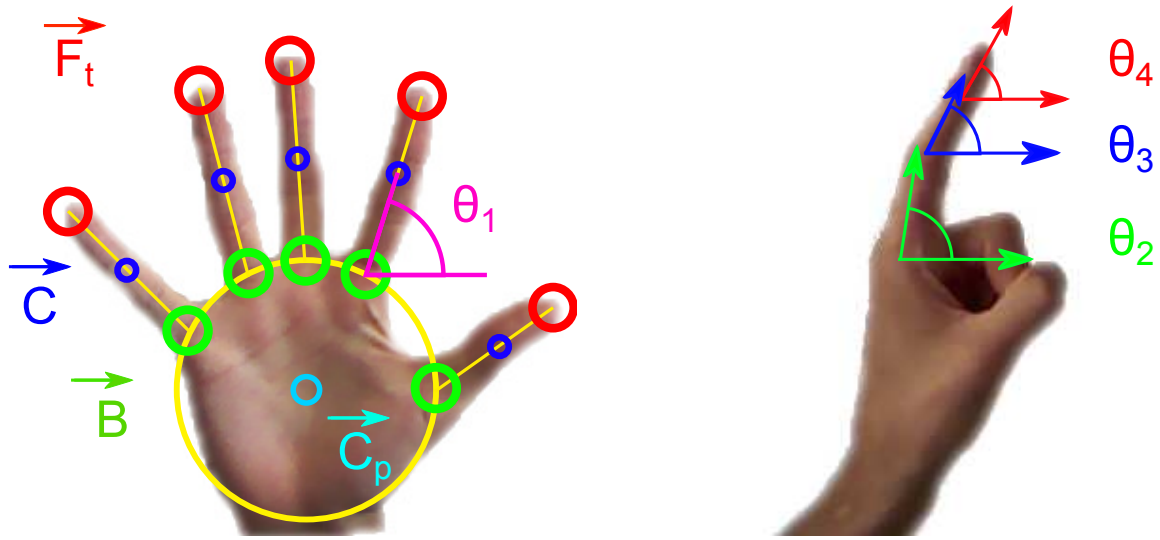
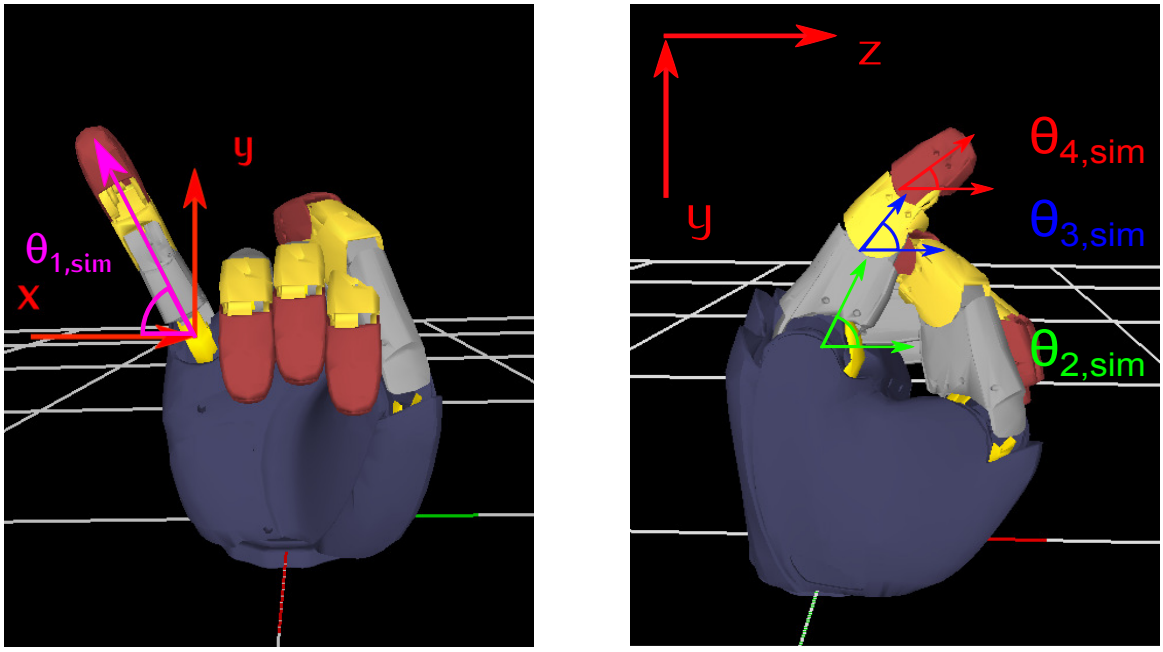


Figure 3.2: 2D hand model. Fingertips $\vec{F}_t$, centroids $\vec{C}$, finger base joints $\vec{B}$, center of palm $\vec{C}_p$.

The assumption is made, that all base joints lie on the circle around the palm center and as the hand is not rotated around the forearm axis, this approximation is an adequate simplification, due to the single perspective view.

3.1.2 3D Simulation

The 3D model deployed in the simulation allows movements with 19 degrees of freedom and hence provides high flexibility and agility. As in the 2D model, the center of the palm acts as main point of reference and resembles the origin of this coordinate system. Contrary to the latter model, here the position of the palm is fixed. All fingers, except the thumb, consist of three phalanges with two degrees of freedom in the base joint and one each in the upper joints. Every finger has three active degrees of freedom and one passive unit at the top joint, which is inferred from other calculations. The thumb, however, only consists of two phalanges with one degree of freedom in the middle joint and two in the base. The control of the simulated hand is performed by passing joint angles as function arguments to the simulation. These angles are then calculated, by computing the relative 3D position of adjacent joints of one finger to the palm center and then checked on boundary conditions. Due to the physiology of the hand, certain configurations cannot be performed and realistic constraints can be formulated to provide a more natural interaction. Fingers cannot be bent further backwards than a maximally stretched hand and the horizontal translation of a finger around its base joint cannot exceed certain values, as this would inevitably lead to the collision of fingers.



(a) Front view (b) Side view

Figure 3.3: 3D hand model of simulation

Figure 3.4 illustrates the coordinate systems utilized in the 3D simulation. The definition of transformations for each conversion is described in table 3.1, which translates the relative position of a point to the specific coordinate system.

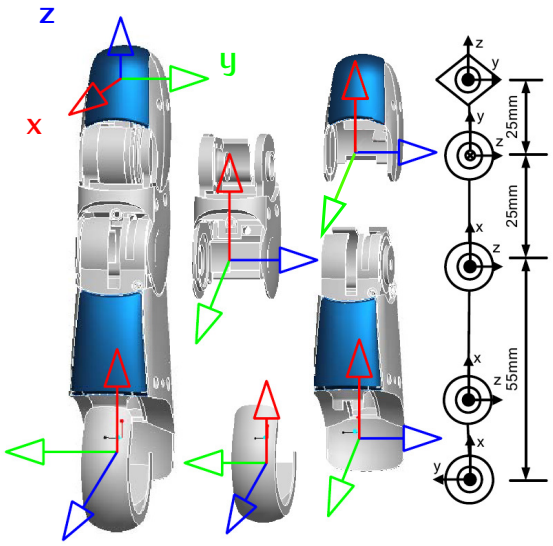


Figure 3.4: Finger model

joint θ_i	d	a	α
θ_1	0	0	0.5π
θ_2	0	55	0
θ_3	0	25	0
θ_4	0	25	0

Table 3.1: DH-parameters

3.2 Hand Segmentation

This procedure can be divided into two individual steps. First, the hand needs to be located within the image and in the second stage it has to be segmented from the background. The identification process is conducted with the support of the OpenNI framework and the functionality of its hand tracker middleware.

Detection of Position with OpenNI Hand Tracker

The OpenNI framework provides a range of useful features, including a recognition system for simple gestures, like *waving, raising the hand* or *clicking*, which is performed by quickly pushing with the flat hand towards the camera and returning to the initial position. To initialize the system two objects need to be created. The gesture generator is responsible for monitoring the image and detecting hand gestures. Then the hand generator invokes a function to start the tracking process and return 3D coordinates of the captured hand. In order to tell the tracker which gestures to look for, the user adds one of the predefined movements, in this case the *clicking* gesture, and the tracking node constantly refreshes to update the user's hand position. All functions relating to gestures or regarding hand positions are implemented as callbacks and form a closed system. After performing the *clicking* movement with a flat hand and stretched out fingers, the coordinates of an arbitrary point within the hand contours are stored and tracked. If the selected point is lost, due to abrupt movement, measures are taken to recover the current hand position. These are described in section 3.6.3.

Figure 3.5: OpenNI hand tracker

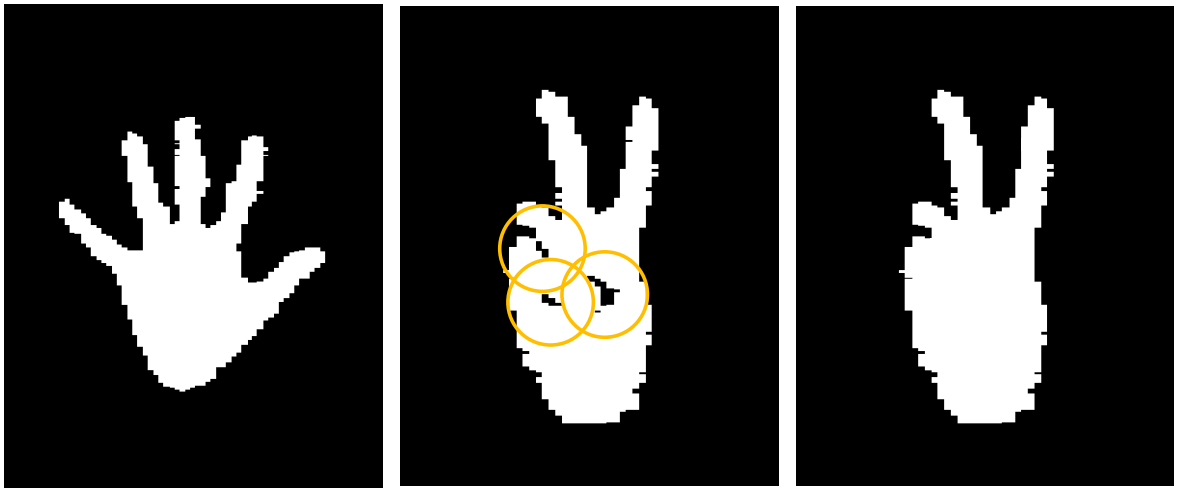
Background Segmentation

The next step is to segment the background and extract only pixels that are possible hand candidates. Therefore this approach utilizes the depth map, obtained from OpenNI functions, as indicator whether pixels should be considered. Points which are located not further than 55 pixel in the x/y plane from the palm center $\vec{C}_p$ and have a maximum depth difference of 10 cm meet these requirements. The average hand length is approximately 19 cm [**?**] resembling 80 pixels in a distance of 80 cm from the camera, which is the standard operating position. To detect movements performed closer to the camera, a radius of 55 pixels is chosen. Further, the maximum depth distance of 10 cm covers every rotation around the x-axis.

$\vec{H} = (x, y, z)$ is considered a hand pixel if:

$$\|\vec{H}(z) - \vec{C}_p(z)\| < 10\text{cm}$$
$$\|\vec{H}(x, y) - \vec{C}_p(x, y)\| < 55\text{pixel}$$

Limiting this volume also adds a beneficial factor regarding performance. Restricting this area reduces the matrix size, originally obtained with a resolution of 640x480, by a factor of 30 and drastically lowers the computational cost for all further operations. After roughly segmenting the background and converting the image to a black and white format (Fig. 3.6a), some correcting measures have to be applied. Shadows covering parts of the hand or interferences disturbing the exact measurement of the infra-red sensor lead to depth values far from acceptable deviations and cause black spots, scattered over the image (Fig. 3.6b).



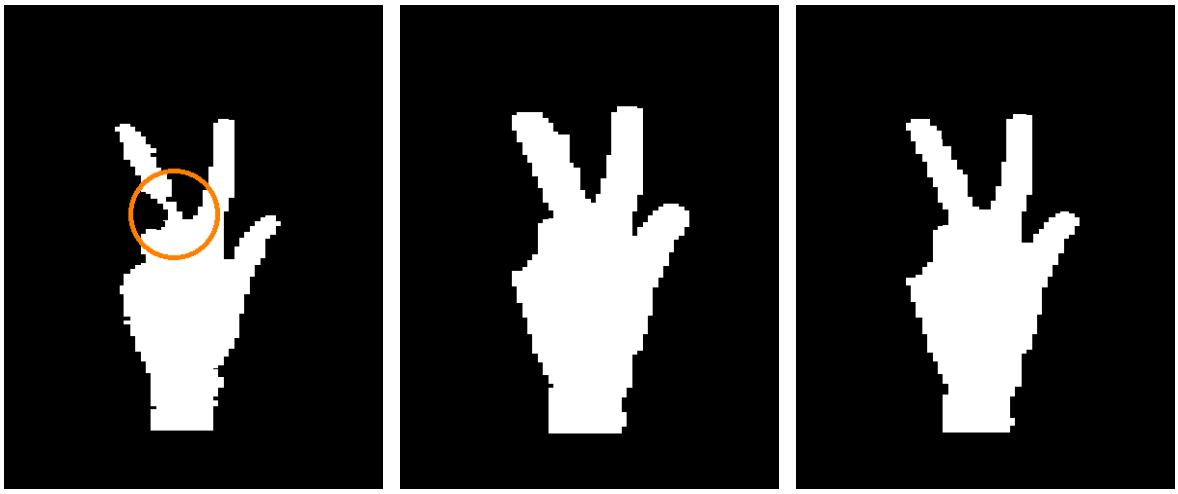
(a) After segmentation (b) Errors due to occlusions (c) Recovered hand

Figure 3.6: Background segmentation and correction of errors

Furthermore, a continuous plane of white pixels, representing the hand shape, is required for the palm detection. Therefore the contours are calculated and, in order to ensure a completely continuous area, then filled up by the *floodfill* algorithm [CSA12]. This uses the following arguments: input image, output image, seed point, new pixel value, threshold pixel value difference and fill type. In order to completely fill the image, the output matrix, described as the mask, needs to be one pixel wider and higher. The seed point is directly inferred from the 2D position of the hand tracker and then used as starting point for this algorithm.

3.3 Palm Detection

For the palm detection it is necessary to determine the center of the palm, represented by the point $\vec{C_p} = (x, y)$ and the radius r. In the previous step all errors and deviations from the depth data have been removed, making the detection of the palm more accurate and thus reducing the probability of sudden jumps of point $\vec{C_p}$. However, when performing certain hand postures some fingers may be recovered insufficiently and segments may be detected as too thin (Fig. 3.7a), which yields difficulties for the later clustering and might cause a single finger to be split into two parts. To avoid this situation, the morphological *Dilate* operation (Fig. 3.7b) expands clusters and merges them, if a sufficient size is reached. To reduce the size of the dilated image the morphological *Erosion* operation (Fig. 3.7c) is applied, still preserving the formerly gained merging of finger parts.



(a) Poor detection (b) After dilation (c) After erosion

Figure 3.7: Palm detection and recovery of segments which could cause false clustering.

As proposed by Abe et al. [ASO11] the center of the palm can be defined as the point inside the hand contour with the greatest distance to all boundary edges. OpenCV provides the *DistanceTransformation* function, which converts the original image $Palm$ from color information to a distance map $Palm'$ by applying the intensity coefficients I.

$$Palm' = Palm * I$$

$$B = \text{closest background pixel}$$

$$I \propto \|Palm_i - B\|$$

These coefficients I describe the distance of this pixel to the nearest boundary edge. The algorithm assigns greater values with increasing distance to a background pixel, resulting in ridge-like formations with the brightest spots in the palm center. Here, several points of high intensity can be observed (Fig. 3.9). To determine the correct position, the nearest candidate to the wrist is chosen and a Gaussian kernel $G(x, y)$ is applied to further smooth the process. Experiments show that a kernel size of 25x25 pixels yields the best results, in terms of minimizing the amount of possible candidates and simultaneously maintaining an adequate computational cost.

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

Figure 3.8: Transformation of distance to pixel intensity

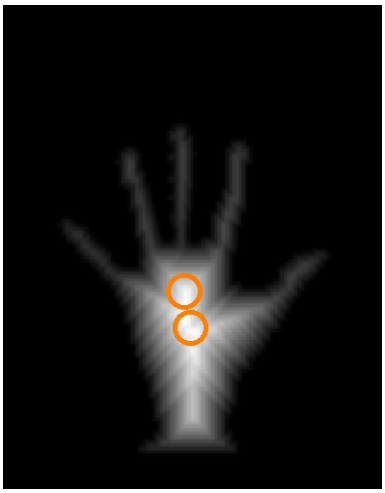


Figure 3.9: Competing palm center candidates

3.4 Finger Detection

The finger detection is a rather complex procedure and therefore is split up into four separate sections. First the fingertips need to be identified precisely, then the scattered pixel arrays are clustered to logical units and then both pieces of information have to be utilized to infer the position of all base joints. At the end of this section, all feature points are located and can be addressed with unique identification numbers.

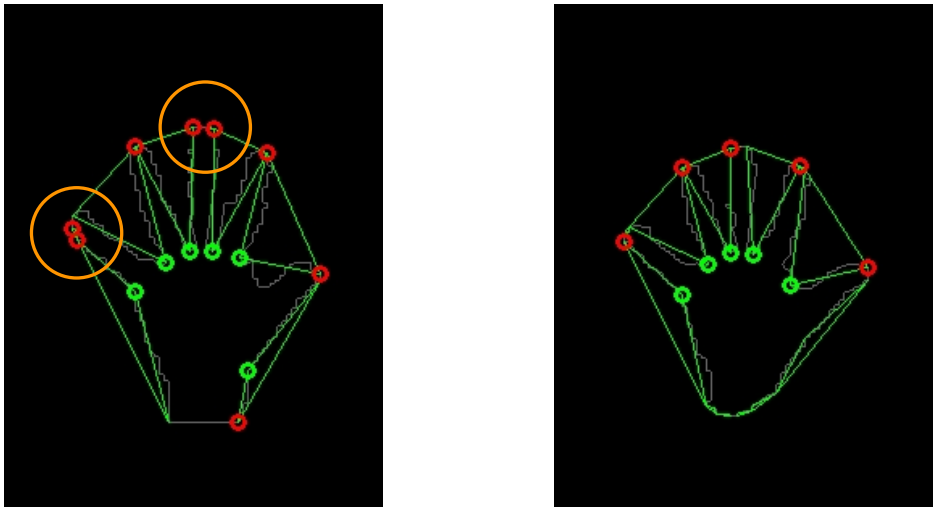
3.4.1 Fingertips

This section describes the procedure of receiving the 2D points of fingertips with the support of OpenCV methods. The first step is to calculate the convex hull, which is the smallest area hull that encloses the hand contour. The function *ConvexHull* computes a vector of points, resembling straight lines that connect all fingers with each other (Fig. 3.10b). In the second step, the exact positions of the fingertips are determined by another OpenCV method, *ConvexityDefects*. This returns convex points, marked red in figure 3.10a, as well as convexity defects, which are highlighted in green in the same image. As the convex hull may touch each fingertip more than once, several convex points per finger are calculated and a filter needs to be employed.

The filter discards points, that meet either of the two criteria:

$$\|\vec{F}_t - \vec{C}_p\| < r_{palm} \qquad \|\vec{F}_{t,i} - \vec{F}_{t,i+1}\|_{onConvexHull} < 20 \text{ pixels}$$

$$\vec{F}_t = \text{fingertip}, \ \vec{C}_p = \text{center of palm}$$



(a) Multiple candidates (b) Single choice

Figure 3.10: Detection of fingertip candidates and filtering to ensure unique assignment

This includes the case that neighbouring fingertips are located close to each other, which means one of them would be rejected following the before mentioned criteria. To avoid this, a third function needs to be taken into account.

$$\textbf{\textit{if}}\ \measuredangle(\vec{F}_{t,i}, \vec{F}_{t,i+1}) < 10^\circ\ \textbf{\textit{then}}\ \text{discardTip} = false$$

In section 3.3 the *Dilate* operation was applied to recover undetected pixel which led to a certain expansion of the palm and finger area, even though the *Erosion* method was also utilized. In a later chapter the mapping of 2D points to 3D coordinates is performed, which can only be accomplished if the 2D coordinates return a z-value, that is part of the hand's depth map. To ensure a correct mapping process the depth map is examined at these feature points and in the case of incorrect correlation a region query is conducted to find the best neighbouring fit.

3.4.2 Clustering with the DBSCAN Algorithm

This section describes the consolidation of the unlabelled pixel to coherent, logical units that represent the single fingers. This step is necessary to assign the before detected fingertips to a specific cluster and an essential step to infer the position of the base joint coordinates. The DBSCAN algorithm - density based spatial clustering of applications with noise - was proposed by Martin Ester et al. [EpKSX96] and describes a clustering algorithm that locates a number of clusters based on the estimated density distribution of corresponding nodes.
Before the clustering algorithm is applied, the fingers need to be optically separated from the palm area, thus resulting in single white areas that are illustrated in figure 3.11.

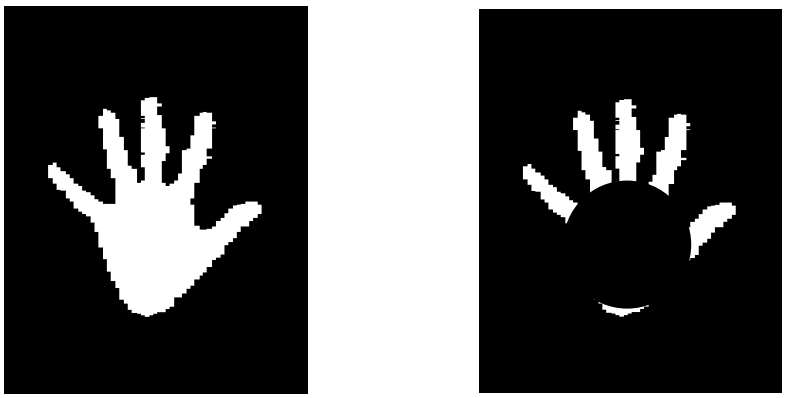


Figure 3.11: Cropping of image with black circle at center of palm

Algorithm 1 DBSCAN $(D, eps, MinPts)$

```
1: Input: D=Dataset, eps=Environment, MinPts=Critical number of neighbours
2: Output: Vector of clusters
3: C = 0
4: for each unvisited point P in dataset D do
5:    mark P as visited
6:    NeighbourPoints = regionQuery(P, eps)
7:    if sizeof(NeighbourPoints) < MinPts then
8:       mark P as NOISE
9:    else
10:      C = next cluster
11:      expandCluster(P, NeighbourPoints, C, eps, MinPts)
12:   end if
13: end for
```

Figure 3.12: Framework for clustering algorithm [EpKSX96]

Algorithm 2 expandCluster $(P, NeighbourPoints, C, eps, MinPts)$

```
1: Input: P=Observed point, NeighbourPoints=Adjacent points, C=Cluster,
   eps=Environment, MinPts=Critical number of neighbours
2: Output: Cluster of correlated points
3: add P to cluster C
4: for each point P' in NeighbourPoints do
5:    if P' not visited then
6:       mark P' as visited
7:       NeighbourPoints' = regionQuery(P', eps)
8:       if sizeof(NeighbourPoints') >= MinPts then
9:          NeighbourPoints = NeighbourPoints joined with NeighbourPoints'
10:      end if
11:      if P' not yet member of any cluster then
12:         add P' to cluster C
13:      end if
14:   end if
15: end for
```

Figure 3.13: Expansion of clusters [EpKSX96]

Algorithm 3 regionQuery (P, eps)

```
1: Input: P=Observed point, eps=Environment
2: Output: NeighbourPoints=Vector of adjacent points in area
3: for each point P' in eps with center P do
4:    if P' not visited then
5:       add P' to NeighbourPoints
6:    end if
7: end for
```

Figure 3.14: Neighbour search [EpKSX96]

The algorithm picks a random point in the dataset D and searches in a small region *eps* for neighbouring points. If a critical amount is present these points are added to the cluster C. Then again a region query is executed for all of them and at a significant neighbour size these clusters are merged. This procedure is illustrated in figure 3.15. The main advantage of this approach over other methods, like *k-means* [KMN+02], is the independence from cluster quantity, not having to specify the number of clusters a priori.

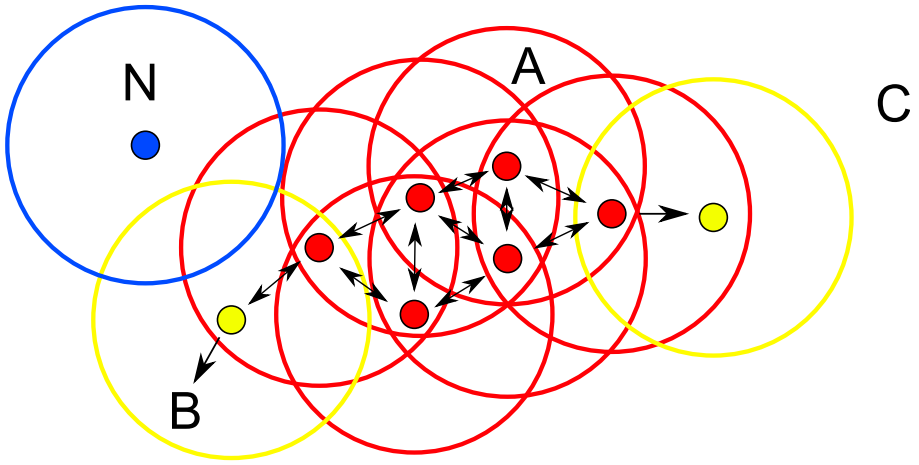
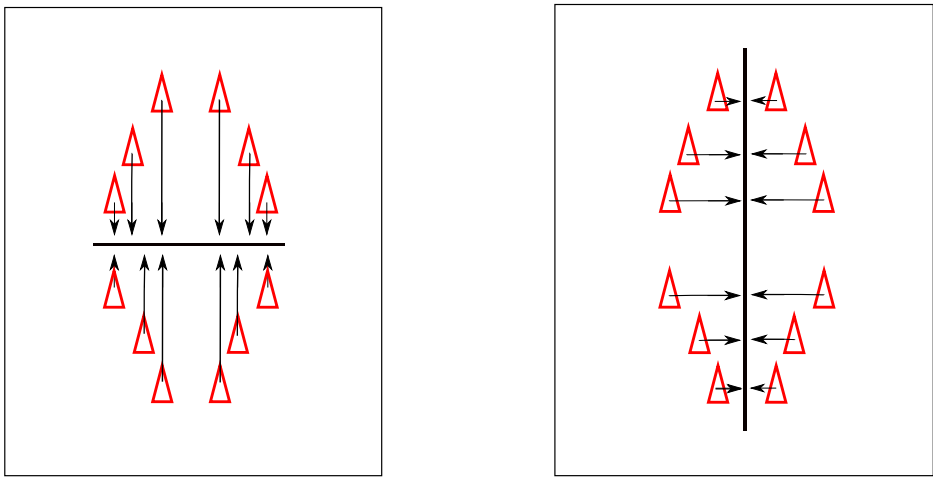


Figure 3.15: DBSCAN algorithm - core points A are highlighted in red. B and C are density-connected to A and thus incorporated in the same cluster. N resembles a noise point that cannot be density-reached nor is a core point [DBS].

3.4.3 Inference of Finger Base Joint Position

After clustering the dataset to finger structures and detecting the fingertips for each one of them, the according joint positions can be inferred and stored. One major assumption made in this thesis is that each root finger joint is located on a circle with a certain radius around the palm center. The radius is determined by calculating the minimal distance of the palm center to the foreground contour. To identify the exact position of each finger base joint on the circle, a straight line must be fitted through each finger, with low susceptibility to calculation errors and minimal processing time.
The method applied is the *Principal Component Analysis* (*PCA*). By utilizing this technique, the data is analysed on its variance and the distribution of deviation to a defined mean value.



(a) Small variance (b) Large variance

Figure 3.16: Principal component analysis results in two perpendicular major components

As seen in figure 3.16, the *PCA* iterates through several lines and calculates the according variance to distinguish between the different distributions. The result of this procedure is a vertical line through the data points of each finger. To determine the best fit *Eigenvectors* and corresponding *Eigenvalues* need to be computed. The *Eigenvectors* resemble a proposed line and the according *Eigenvalues* illustrate their level of variance, which can be measured and utilized to compare already calculated options. The principal component with the highest variance will be applied to identify the joint position.

Algorithm 4 Detect joint position $(\vec{C}_f, \vec{C}_p, \vec{e}, r)$

1: **Variables:** $\vec{C}_f$ = centroid, $\vec{C}_p$ = center of palm, $\vec{e}$ = eigenvector, r = palm radius
2: **for** j=0 **to** j=40 **do**
3: **if** $\|(\vec{C}_f - \vec{e} * j) - \vec{C}_p\| < r$ **then**
4: mark *this location* as finger joint base position
5: break
6: **end if**
7: **end for**

The joint positions are located at the intersection of the fitted line with the circle around the palm center and the algorithm 4 is deployed to obtain this position. It starts at the center of each finger and propagates in small iteration steps towards the palm center, until the distance between the current position and the palm center is smaller than the hand radius.

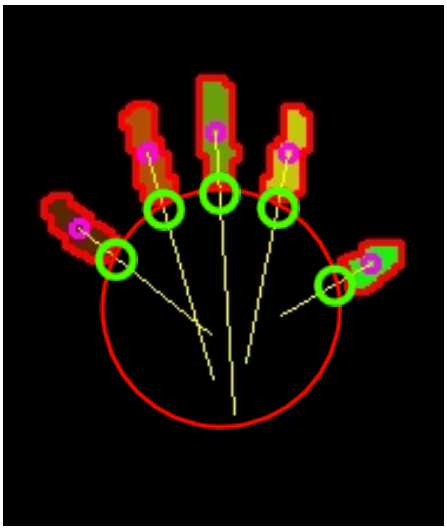


Figure 3.17: Base joint positions, marked green, at intersection of the first principal component and circle around palm center.

After calculating the position of all joints, the angles between them are stored in a joint map, which is utilized in a later processing step to estimate the joint positions. This is necessary, when assigning a specific number to fingers that were occluded and reappear in the image. The next section 3.4.4 describes this process in detail and gives an example of such a finger recovery.

3.4.4 Numbering of Fingers

After the cluster of every finger has been detected and the according joint positions have been inferred, the correct number corresponding to thumb, index, middle, ring and little finger can be assigned. The approach applied in this work does not utilize a hand model and thus cannot provide a single frame analysis regarding correct finger numbering. This means, decisions have to be made, based on knowledge of previous assignments and inference of the most likely relation. Therefore an initialization situation is defined, when all five fingers are detected and the correct numbering can be performed easily.
The next step is to ensure a consistent assignment and maintain the numbering for fingers that remain visible and are not bent out of sight. When calculating the *Principal Component Analysis* in section 3.4.3, centroids of each cluster are computed that serve as a good tool to match the detected fingers to the assigned numbering of the last iteration step. Depending on the distance d from the current centroid to the last position of each of them, the according finger is assigned and the centroid position is updated.

Algorithm 5 updateCentroidPosition $(\vec{C}_{lP}, \vec{C}_{cP}, \vec{F}_{t,cP})$

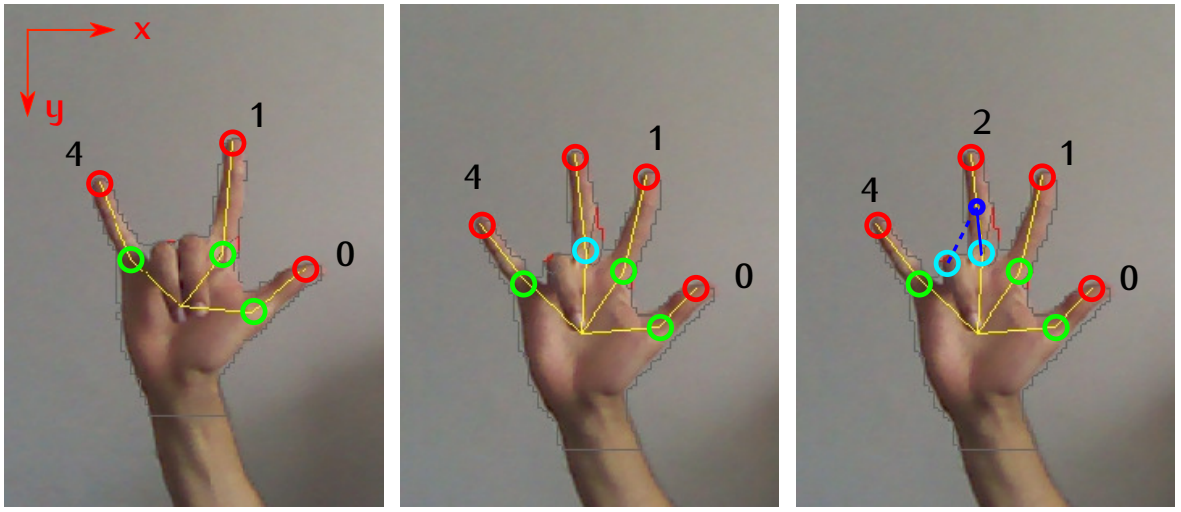
Variables: $\vec{C}$ = centroid, $\vec{F}_t$ = fingertip
Index: lP = last position, cP = current Position
for $\forall \vec{C}$ **do**
  $d = \|\vec{C}_{cP} - \vec{C}_{lP}\|$
  if $d < 15$ pixel **then**
    $\vec{C}_{lP} = \vec{C}_{cP}$
    $\vec{F}_{t,cP}(ID) = \vec{C}_{cP}(ID)$
  end if
end for

In order to identify whether a finger has moved out of sight a polygon test is performed that iterates through all clusters and checks, if the current centroid is located in one of them. The function requires a point and a contour as input parameter and returns *+1*, if the point is inside the contour, *-1* if it is outside and *0* if it lies on the edge. Then the vector *fingersHidden* is set to *1* for visible and to *0* for hidden fingers.

Algorithm 6 polygonTest ($\vec{C}_{cP}, Ct, \vec{F}_{hidden}$)

Variables: $\vec{C}$ = centroid, $\vec{F}_{hidden}$ = fingers hidden, Ct = contour
Index: cP = current Position
for $\forall Ct$ **do**
  compare $\vec{C}$ position with Ct
  if $\vec{C}$ within Ct **then**
    $\vec{F}_{hidden,cP} = 1$
    return +1
  else
    $\vec{F}_{hidden,cP} = 0$
    return -1
  end if
end for

The most complicated part of this section is to correctly recover the numbering of fingers, which were occluded and now reappear as new clusters. Therefore two different cases need to be considered. The easier situation is a single occluded finger, that reappears and consequently resembles the number of the position of the only hidden finger in vector $\vec{F}_{hidden}$. The second, more difficult case is illustrated in figure 3.18a, in which several fingers are hidden and the reassignment cannot be achieved directly, but has to be approximated, based on the relative distance to estimated joints and considering certain boundary constraints.
As described in section 3.4.3 the angle between every joint is stored at the time of initialization, enabling the estimation of joint positions in case of occlusion. Thus the distance of the centroid of the new detected finger cluster to every joint can be calculated and the match with the minimal euclidean norm is chosen as the according finger. For example, figure 3.18c demonstrates a situation with two occluded fingers and the recovery of the correct numbering.



(a) Middle and ring finger hidden.

(b) Direct assignment not possible, as new cluster can be interpreted as either finger.

(c) Correct finger assignment, as centroid (*blue*) closer to estimated joint of middle finger.

Figure 3.18: Finger numbering based on centroid method

However, the numbering sometimes returns wrong values, because the centroid of one finger is closer to the estimated joint position of an adjacent one or a movement is performed too quickly such that the correlation of the last centroid position to the current location is lost. To partially solve this problem and improve accuracy, the proposed numbering is checked after every iteration regarding order and sorted, if necessary. Both vectors, the position of all centroids and the vector list of fingertip and joint positions, are sorted accordingly to maintain the correct matching (Alg. 7).

Algorithm 7 sortNumbering $(\vec{C}, \vec{C}t)$

Variables: $\vec{C}$ = centroid, Ct = contour
Index: i = finger number
for i=0 **to** i=3 **do**
  compare centroid position with contour
  if $\vec{C}_{i+1}(x) < \vec{C}_(x)$ **then**
    swap centroid numbering to ensure correct order
  end if
  if $\vec{C}t_{i+1}(x) < \vec{C}t_(x)$ **then**
    swap contour numbering to ensure correct order
  end if
end for

3.5 Hand Pose Estimation

Sections 3.2 to 3.4 illustrate the entire procedure of receiving raw image data and processing it to perform a full 2D tracking system of fingertips and joints. This section adds the third dimension and elevates the present tracker to a 3D hand pose estimator, which is capable of translating 3D positions to angles that are then utilized by the simulation model to represent a computer-based imitation of the performed motion.

3.5.1 Assignment of 3D Coordinates

As mentioned in section 3.4.1 the mapping can only be performed, if the observed 2D coordinates represent an appropriate depth value, which is part of the hand. The solution for fingertips is already described in section 3.4.1 and now applied to retrieve the depth data from the depth map.
The next step is to determine the z-value of the finger joints. Therefore the assumption has to be made that all joints are located in the same plane with equal distance from the camera. This consideration is based on two main reasons.
Firstly, sometimes joints of visible fingers are occluded due to certain hand configurations when fingers are bent. This leads to smaller depth values, as the concealing finger covers this point in a closer distance to the camera. The result in the 3D image is an alteration of the joint angle, because the difference of joint and fingertip changes, as seen in figure 3.20. Furthermore the tracking will become unstable and show high deviations and jitter.

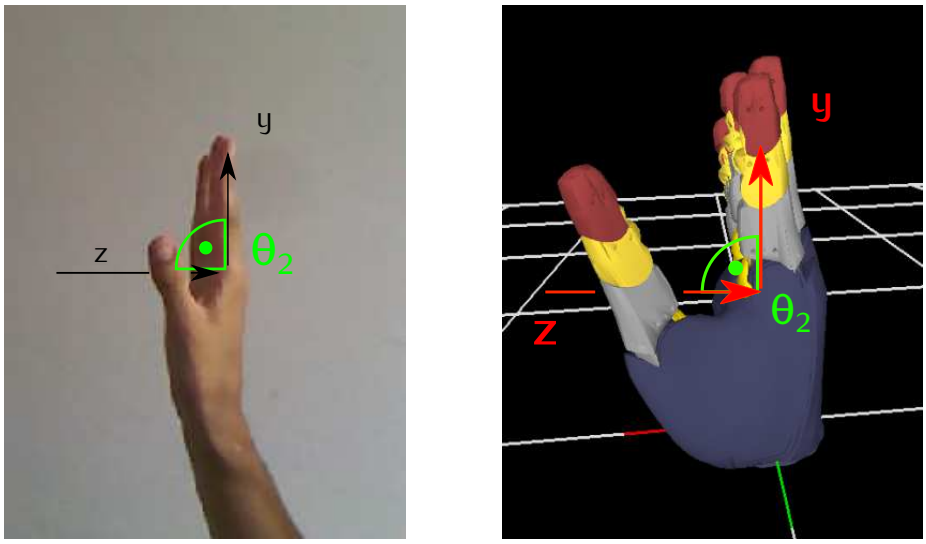


Figure 3.19: Thumb not obstructing direct view of finger base joints. Therefore angle θ_2 in simulation is equal to angle θ_2 in real hand pose.

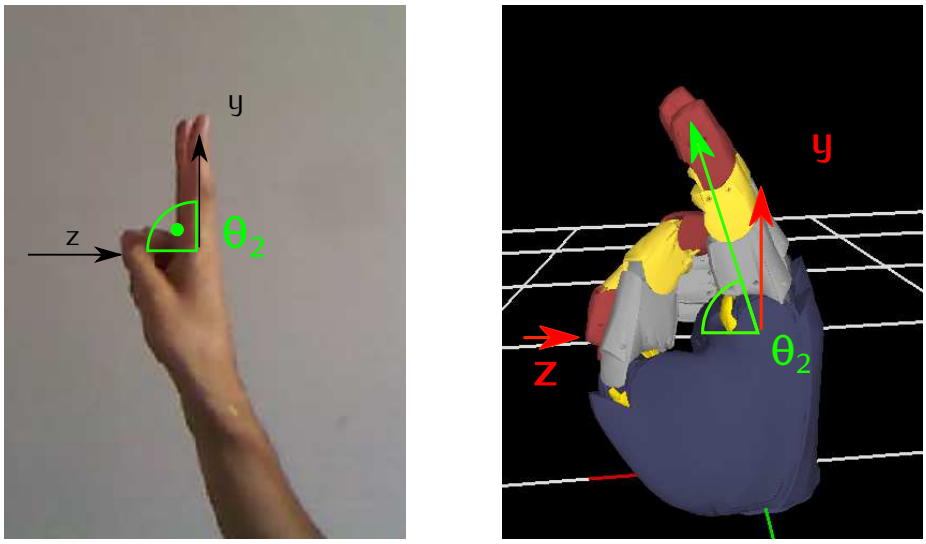


Figure 3.20: Base joints of index and middle finger are occluded. This leads to different depth values of joints and fingertips, thus reducing the calculated angle θ_2 in the simulation.

Secondly, the rotation around the forearm axis is not considered in this project. Due to the fixed wrist position in the 3D simulation a rotation of the hand cannot be transferred to the calculations and does not affect the relative angle of the finger to the palm plane. Additionally, this would easily lead to occlusions between fingers that cannot be resolved without utilizing a model-based tracking approach, which is not in the scope of this thesis. Therefore the depth coordinate assigned to all joints is chosen as the deepest point around the palm center. As the simulation assumes a fixed position for the entire palm this value is also adopted for the depth of the palm center.

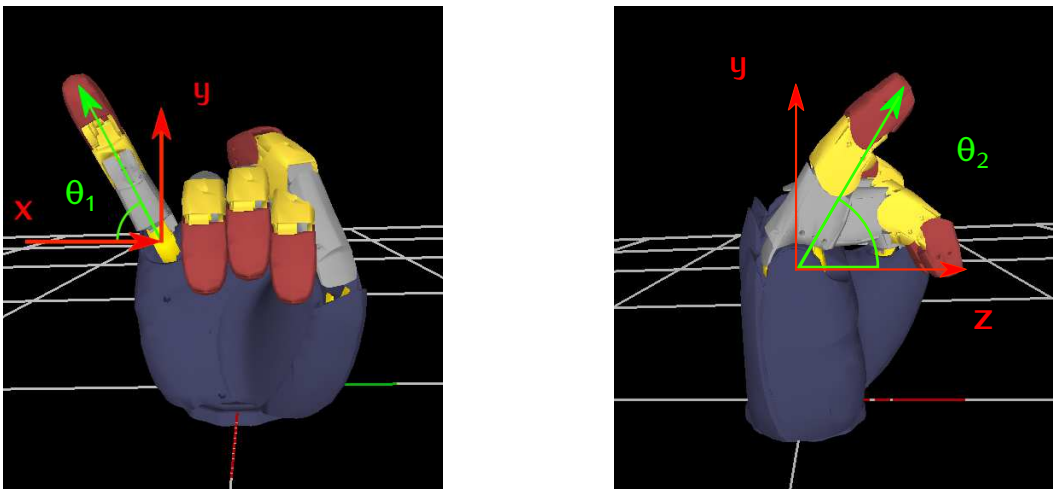
3.5.2 Angles of Finger Segments at Respective Joints

The 3D simulation model requires four input parameters for each finger and three for the thumb, that represent the angle at each joint in the x/y plane (Fig. 3.21a) and y/z plane (Fig. 3.21b). The angles θ_1 and θ_2 are calculated independently, whereas θ_3 and θ_4 are inferred from θ_2. For maintenance, one vector stores the 3D position of every joint and fingertip in a separate vector, which is later fed into the data package to communicate the information to the simulation. The perspective in the model is set to the palm center, which is why the current points first have to be converted to the new coordinate system.

$$\vec{F}_t, i(x,y,z) = \begin{pmatrix} \vec{F}_t, i(x) - \vec{C}_p(x) \\ \vec{F}_t, i(y) - \vec{C}_p(y) \\ \vec{F}_t, i(z) - \vec{C}_p(z) \end{pmatrix} \qquad \vec{B}_i(x,y,z) = \begin{pmatrix} \vec{J}_i(x) - \vec{C}_p(x) \\ \vec{J}_i(y) - \vec{C}_p(y) \\ \vec{C}_p(z) \end{pmatrix}$$

$$\vec{C}_p(x,y,z) = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \qquad \vec{C}_p = \text{center of palm}, \qquad \vec{F}_t = \text{fingertip}, \qquad \vec{B} = \text{joint}$$

The angles are calculated consecutively for each finger, beginning at the base joint in the x/y plane and the y/z plane to then infer the angle of the middle and top joints.



(a) x/y plane (b) y/z plane

Figure 3.21: Coordinate systems utilized in the 3D hand model for angle calculation.

Algorithm 8 Angle θ_1 of base joint in x/y plane($\vec{B}, \vec{F}_t$)

Variables: $\vec{B}$ = joint, $\vec{F}_t$ = fingertip
for each finger **do**
  if finger is hidden **then**
    $\theta_1 = 0°$
  else
    if finger != thumb **then**
      $\theta_1 = \angle(\vec{B}_i(x,y), \vec{F}_{t,i}(x,y))$
    else
      $\theta_1 = \angle(\vec{B}_i(x,y), \vec{F}_{t,i}(x,y)) + 35°$
    end if
  end if
end for

The base joint in the thumb is treated separately, because the simulation model already implies an offset, as the initial position of the thumb approximately lies in an angle of 35° to the plane of the joints. Thus it is balanced out by adding a correction factor of 35 degrees. When a finger is declared hidden the contraction needs to be performed smoothly and the final position should represent a natural posture. Hence, the angles ought not to exceed physiological limits and cover the specific range for each joint. Considering the angle θ_2 in the y/z plane of the base joint, this comprises the spectrum of 10 to 85 degrees for each finger and 30 to 45 degrees for the thumb. The greater value, likewise, resembling the contracted posture.

Algorithm 9 Angle θ_2 of base joint in y/z plane $(\vec{B}, \vec{F_t})$

Variables: $\vec{B}$ = joint, $\vec{F_t}$ = fingertip
for each finger **do**
  if finger is hidden **then**
    if finger != thumb **then**
      $\theta_2 = 85°$
    else
      $\theta_2 = 45°$
    end if
  else
    if finger != thumb **then**
      $\theta_2 = \angle(\vec{B}_i(y,z), \vec{F}_{t,i}(y,z))$
    else
      $\theta_2 = \angle(\vec{B}_i(y,z), \vec{F}_{t,i}(y,z))$
    end if
  end if
end for

The smooth movement of fingers and natural bending can be ensured by deducing the middle and top angle from the flection at the base joint. After experimenting with a set of multiplication factors, a percentage of 90% has proven to be the best fit.

Algorithm 10 Angle of middle joint in y/z plane (θ_2)

for each finger **do**
  if finger is hidden **then**
    $\theta_3 = 75°$
  else
    $\theta_3 = \theta_2 * 0.90$
  end if
end for

After all angles have been calculated, a median filter is applied to filter extreme deviations and smoothen the process. At this stage, the hand tracker and 3D pose estimator are fully functional and the data package can be sent to the simulation model.

3.6 Stabilization

Without this stabilization step, the tracking and pose estimation would already provide the complete functionality, but still be highly susceptible to minor deviations and incapable of dealing with relatively quick movements. This chapter deals with these issues and improves the stability of joint and palm detection, the accuracy of cluster computation and the overall robustness of the program.

3.6.1 Kalman Filter for Joints and Center of Palm

The accuracy of determining the exact joint position depends on the calculation of the intersection between the circle around the palm center and the line fitted through each finger cluster. The significance of this detection decreases with the cumulated errors of the circle and the fitted lines. To reduce the discrepancy in the computed palm center and the resulting jitter of the position of the joint, a Kalman filter is employed and implemented for all elements.
The concept of this filter is an iterative weighted average [Kal60].

$$\widehat{x}_k = K_k z_k + (1 - K_k)\widehat{x}_{k-1} \tag{3.1}$$

$\widehat{x}_k$: Current estimation

$\widehat{x}_{k-1}$: Previous estimation

z_k: Measured value

K_k: Kalman gain

The Kalman filter smoothens movements by reducing deviations and ensuring a continuous signal propagation. This is achieved by computing predictions for the next time step k and comparing them with the measured value Z_k. The advantage of this filter is that it constantly adapts the averaging coefficient K_k to find the optimum factor to correct its predictions.

The entire process of the filter can be generally divided into four steps:

- model design
- prediction
- correction
- iteration

Model design:

$$x_k = Ax_{k-1} + Bu_k + w_{k-1} \tag{3.2}$$
$$z_k = Hx_k + v_k \tag{3.3}$$

A: State transition model
B: Control input model
x_k: Current estimation
u_k: Control signal
w_{k-1}: Process noise
H: Observation model
z_k: Measured value
v_k: Measurement noise

The equation 3.2 describes the calculation of the signal value x_k by a linear stochastic equation. It consists of its previous value x_{k-1} multiplied with the state transition matrix A, that is filled with fixed numeric values, a control signal u_k multiplied with the control input matrix B and the process noise w_{k-1}.
The second equation 3.3 illustrates that the measured value z_k is expressed by a linear combination of the signal value x_k multiplied with the observation matrix H and the measurement noise v_k. It is important to note, that the process and measurement noise are statistically independent.

Prediction:

$$\widehat{x_k^-} = A\widehat{x}_{k-1} + Bu_k \tag{3.4}$$
$$P_k^- = AP_{k-1}A^T + Q \tag{3.5}$$

A: State transition model
B: Control input model
$\widehat{x_k^-}$: Predicted a priori state estimate
$\widehat{x_{k-1}^-}$: Previous estimate
u_k: Control signal
A: State transition model
P_k^-: Predicted a priori covariance
P_{k-1}: Previous covariance prediction
Q: Covariance of process noise

The prediction is constantly updated and adjusted by considering the covariance of the process noise and multiplying the previous predictions with the state transition model to refine the current outcome.

Correction:

$$K_k = P_k^- H^T (H P_k^- H^T + R)^{-1} \tag{3.6}$$
$$\widehat{x}_k = \widehat{x}_k^- + K_k(z_k - H\widehat{x}_k^-) \tag{3.7}$$
$$P_k = (I - K_k H) P_k^- \tag{3.8}$$

H: Observation model
P_k^-: Predicted a priori covariance
K_k: Kalman gain
R: Covariance of measurement noise
P_k: Updated a posteriori covariance
$\widehat{x}_k$: Current estimation
$\widehat{x}_k^-$: Predicted a priori state estimate
z_k: Measured value

To correct the predicted movement the Kalman Gain K_k needs to be updated, first. This considers the current observation H with regard to a certain covariance of measurement noise R and then revises the a priori prediction P_k^-. With this improved factor, the estimate $\widehat{x}_k$ can be updated more accurately when comparing the prediction with the measured value z_k. Finally the a posteriori estimate covariance P_k can also be updated and fed into prediction cycle $k+1$.

Iteration:

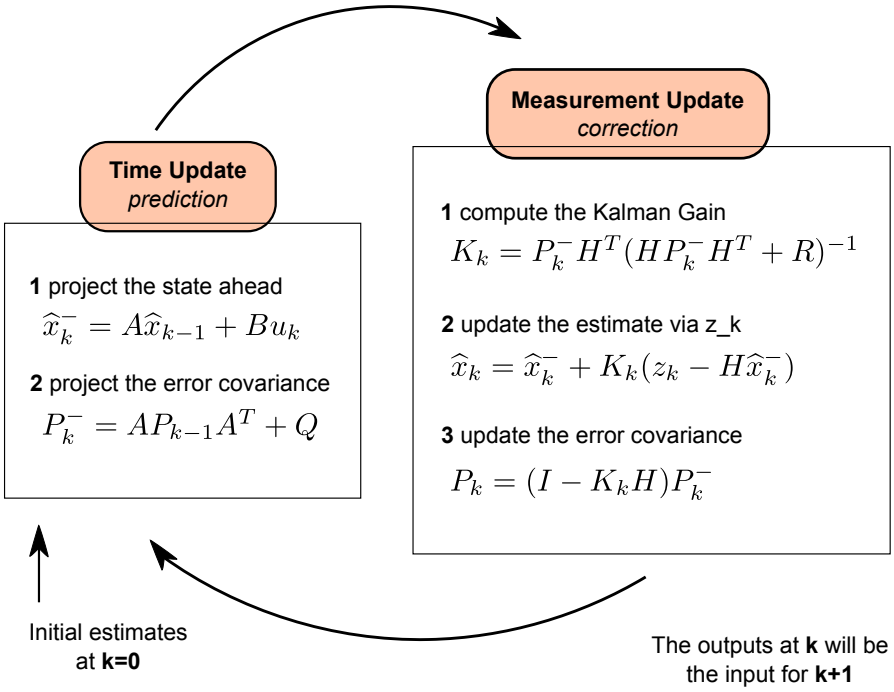


Figure 3.22: Iteration cycle of the Kalman filter [Kal]

After gathering all the required information, the actual process can be started and the estimates constantly refined by using the previous estimates as input for the current state. In the prediction step, $\widehat{x}_k^-$ describes the prior estimate, which is the rough estimate before the measurement update correction. P_k^- also represents the prior error covariance. Both of these prior values are utilized in the measurement update step.
The goal of the second step is to compute the best possible estimate $\widehat{x}_k$ of x at the time k. Therefore the Kalman Gain K_k is calculated first and later applied again to update the error covariance P_k, which is necessary for the future estimation at step $k+1$, together with $\widehat{x}_k$. The matrices applied to the hand tracker are listed in the following:

$$A_{palm} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad Q_{palm} = e^{-4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad R_{palm} = e^{-4} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

$$A_{joints} = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \quad Q_{joints} = e^{-2} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad R_{joints} = e^{-1} \begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix}$$

3.6.2 Reduction of Unwanted Dilations

During the process of palm detection certain algorithms are applied to extract the position of its center. The dilation algorithm is necessary to improve the stability of the palm center detection, as it fills gaps in the black and white image and thus steadies the basis for the distance transformation (Sec. 3.3). On the other hand, this method invokes the unwanted side effect that the entire hand area expands and the gap between fingers diminishes, as single clusters merge together (Fig. 3.23). The excessive use of the erosion algorithm might counteract the positive influence of the dilation process. However, in order to reverse the negative corollary effect, the incurred fusion of clusters can be removed, leaving only clearly separated finger clusters behind (Fig. 3.24).

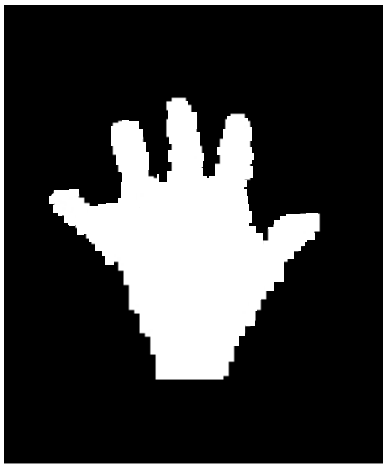


Figure 3.23: Segmented hand without erasing unwanted dilations

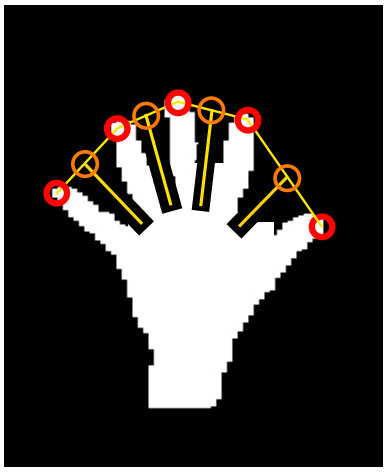


Figure 3.24: Separation of dilation from base joints to center between fingertips

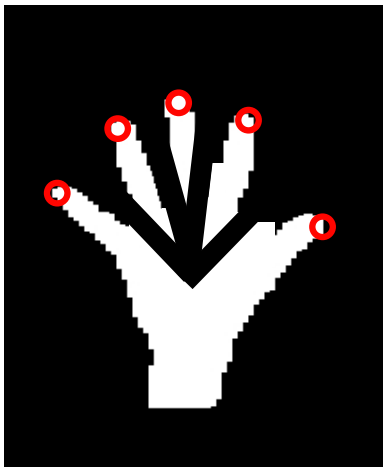


Figure 3.25: Erasing from palm center will lead to several palm center candidates

The first step is to find the center between two neighbouring fingertips. As all fingertips are stored in one common vector each center can easily be determined and temporarily saved for further calculations.

$$\vec{center}_{F_t}(x, y) = \vec{F}_{t,i} + (\vec{F}_{t,i+1} - \vec{F}_{t,i})/2$$

At the next step, the line between the center of the palm and the center between the two fingertips needs to be identified covering the merged area. The challenge is to propose a line separating the clusters, but originating between adjacent finger base joints in order not to falsely affect the distance transformation (Fig. 3.24). If all four lines were to originate at the palm center, the palm would be divided and no unique center could be determined (Fig. 3.25).

Therefore the starting point is chosen at the intersection of the circle around the palm, with radius equal to the center-contour distance, and the direct line from the middle of two fingertips to the center. This line then erases all unwanted dilations up to the center between both tips and hereby ensures a higher precision in palm center detection, a more accurate finger clustering and a higher overall performance.

3.6.3 Reinitialization of Program for Undefined Status

Certain situations induce disturbances of the program, which are rarely invoked, but have to be taken care of. Errors of sensor data, caused by certain lighting conditions or too rapid movements of the user can result in such situations. If a movement is performed quickly, the transition between the present and prior state cannot be conceived as such, but instead will be ignored as measurement deviation. To handle these situations, a reconfiguration is implemented that restores the initial situation, resetting all variables, deleting and reconstructing entities, and hence provides the user with a stable framework.
One of these situations is created, when the *FloodFill* algorithm delivers a wrong outcome and the further operations are performed on false data. When the seed point input parameter is altered, the calculated mask will be the negative image and subsequently entail a different distance transformation. As a consequence, the point with the highest value after the transformation usually is located at the coordinate system's origin.

Figure 3.26: Negative mask, due to wrong *FloodFill* seed point (*yellow*)

Another situation, invoked by similar miscalculations, shows the symptoms of the palm center coordinates exceeding the 640x480 limits of the matrix size. This circumstance can be detected by checking the proposed palm coordinates at every iteration.
When encountering one of these conditions, the program has to be reinitialized with all its original values, but should also continue the tracking process for a comfortable user interaction. Therefore, the seed point for the next *FloodFill* iteration is copied from the last determined, true position of the palm center and simultaneously used to update the *OpenNI* hand tracker. This allows a continuous usage of the program, without the user having to restart the program manually by executing the identification gesture.

Chapter 4

Hardware Setup

The utilized hardware setup is a *Dell Precision T3500* computer with an *Intel Xeon* 2.4 GHz dual core processor and 6GB RAM and the *Microsoft Kinect* camera. This camera combines several sensors for color and depth data processing (Fig. 4.2).
The technical specifications of the *Kinect* are:

1. RGB camera with the maximal resolution of 640x480 @ 30Hz
2. multi array microphone consisting of 4 microphones, 16 bit audio @ 16Hz
3. vertical tilt motor with range of ± 27°
4. infra red projector emits light patterns. As they hit the surface the pattern becomes distorted and the distortion is then red by the IR camera
5. infra red camera to analyse IR patterns and build a 3D depth map

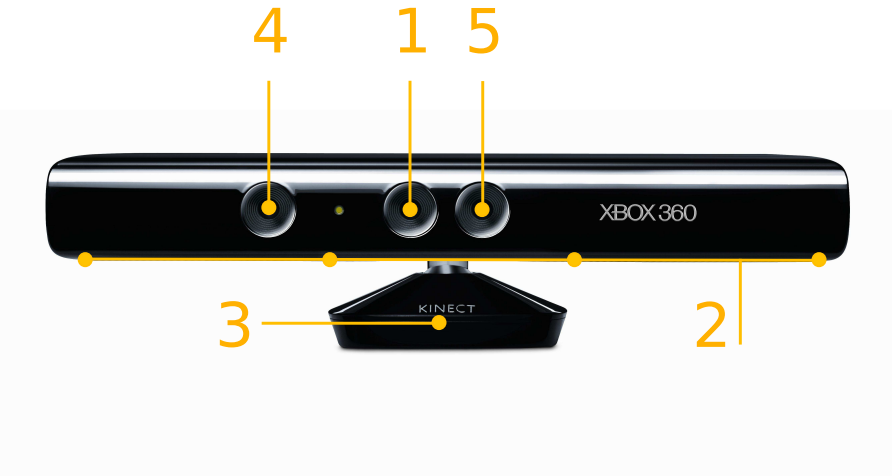


Figure 4.1: Kinect with color, depth and audio sensors [Kina]

The monochrome CMOS sensor captures light which has been projected by the infrared laser. The depth map is then constructed by comparing the known with the captured pattern, thus displaying an array of deviation values. The sensor provides a depth sensitivity of 2048 levels, as it streams 11-bit data and acts within a range of 0.6-3.5 meters. The vertical field of view comprises 43 degrees and the horizontal spectrum is limited to 57 degrees.

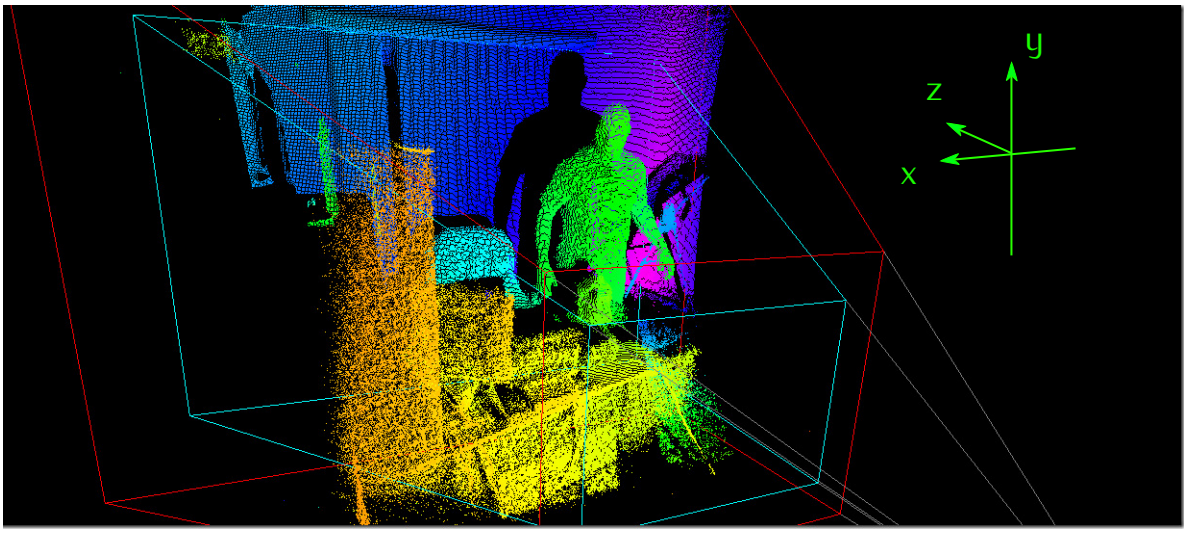


Figure 4.2: *Kinect* coordinate system [Kinb]

The coordinate system of the *Microsoft Kinect* camera represents a 2D matrix with its origin in the center of the image with the y-axis pointing up and the x-axis to the left. As the image is represented by pixels, the allocation in the 2D plane is not described in metric distances, but in single pixel incrementation steps. The 3D value, however, depicts the depth in physical units with a sensitivity of 2048 steps. When processing data, the transformation of coordinates has to be considered, as the *OpenCV* library exclusively utilizes a matrix like coordinate system with its origin in the upper left corner and axes pointing to the right and down. The data retrieval and processing of information acquired from the *Kinect* was performed with the support of the software framework provided by *OpenNI* and *OpenCV*. Apart from this configuration no other hardware modules or auxiliary devices like data gloves were utilized.

The test setup utilized in this thesis is illustrated in Figure 4.3. Depth and color information is provided by the *Kinect* camera and then forwarded to the program, which conducts the processing and feeds the simulation model with data of angles and joint positions.

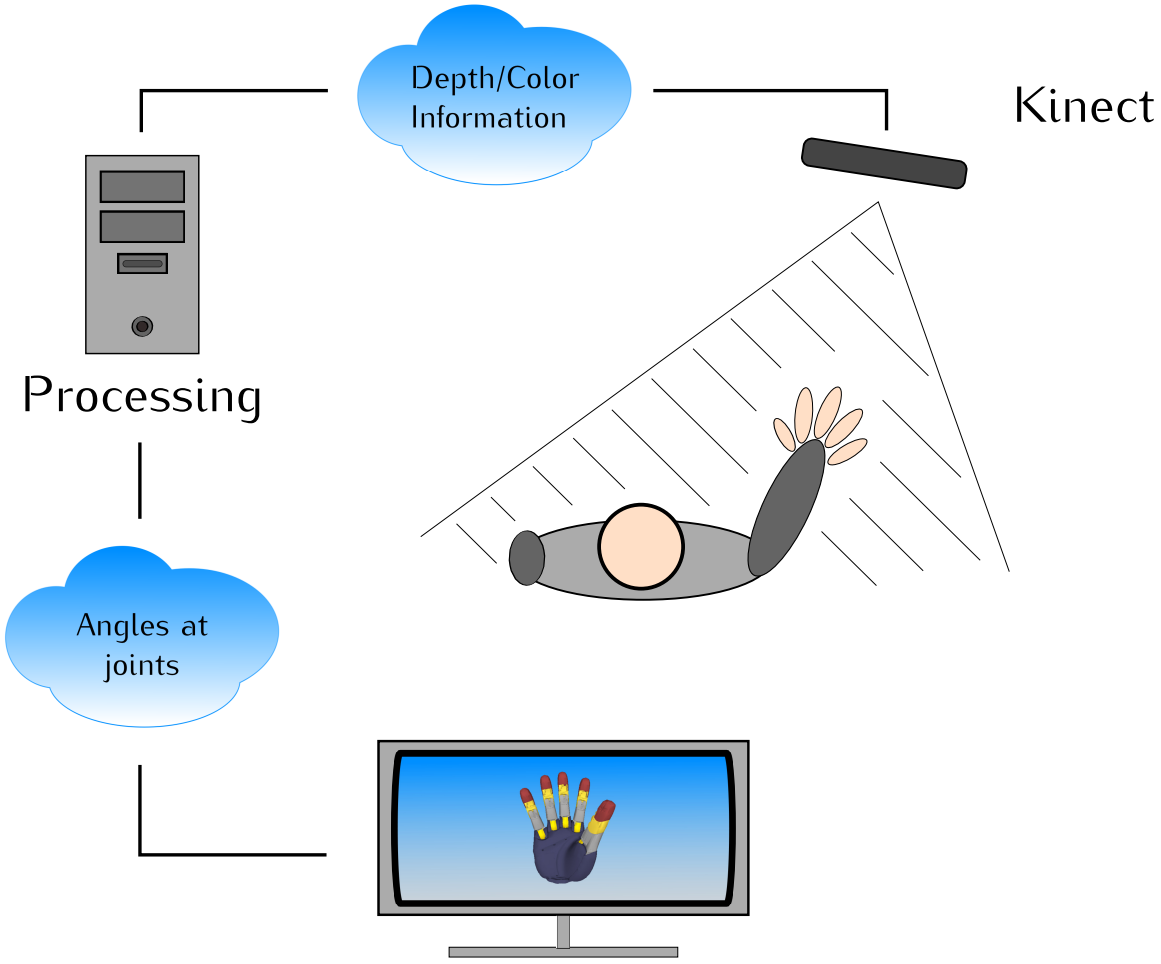


Figure 4.3: Setup

Chapter 5

Evaluation

The approach, described in chapter 3, presents a solution for a hand pose estimation system that incorporating several processing blocks, which are all evaluated in this chapter. The core factors of this system that define its quality include precision of movement perception (Sec. 5.1), accuracy of correct finger assignment (Sec. 5.2) and user experience when interacting with the interface (Sec. 5.3). These aspects are considered in the following evaluation and assessed by a group of independent users.

5.1 Lateral View Accuracy

A great challenge that has been tackled in this project, is to develop a high sensitivity for the modification of angles in finger joints and thus provide the user with a natural interaction experience. As described in section 3.5.2, the calculation of angles depends on the relative 3D position of the fingertip to the base joint, which is located in the same plane as the palm center. A big influence that might alter the computed joint position is a strong occlusion of the palm by contracted fingers. This case is not considered in the following test, without a loss of generality.
The test measures the angles at all joints in both images, the real hand posture and the 3D simulation, and compares them by calculating the difference. The resulting value represents the discrepancy of the real image to the simulation model.

$$\theta_{diff} = \theta_{real} - \theta_{sim}$$

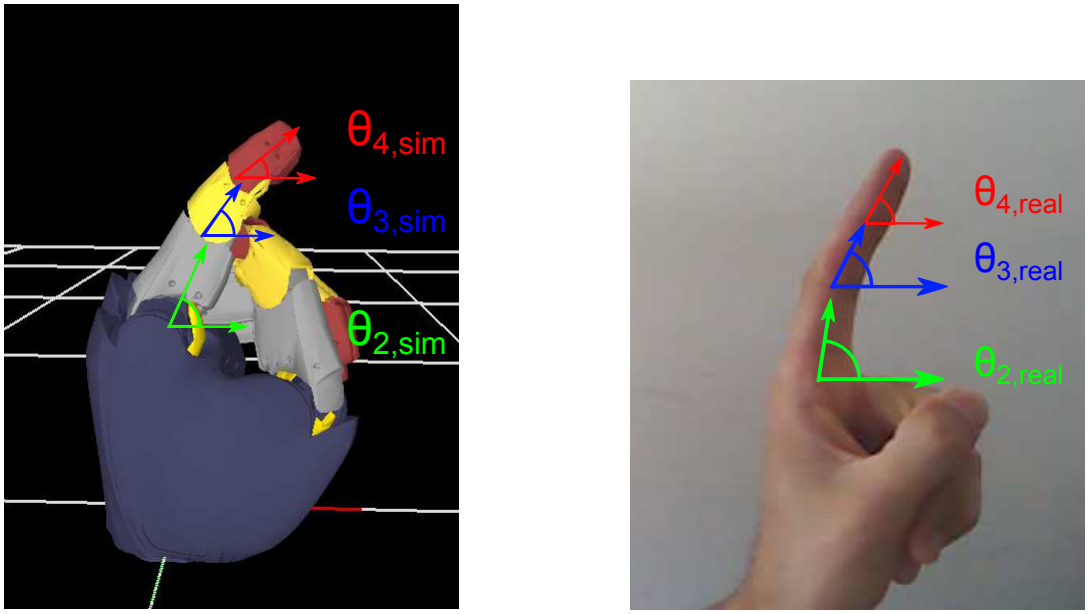


Figure 5.1: The denotation of angles in base, middle and top joint is defined likewise for the 3D simulation model and the real hand configuration. They are utilized to describe discrepancies θ_{diff} between θ_{real} and θ_{sim}.

At every iteration step the three angles in the y/z plane are measured for the real and the simulated hand in maximal resolution and subtracted to give a quantitative evaluation scheme. The results observed in figure 5.3 are described in the following table 5.1.

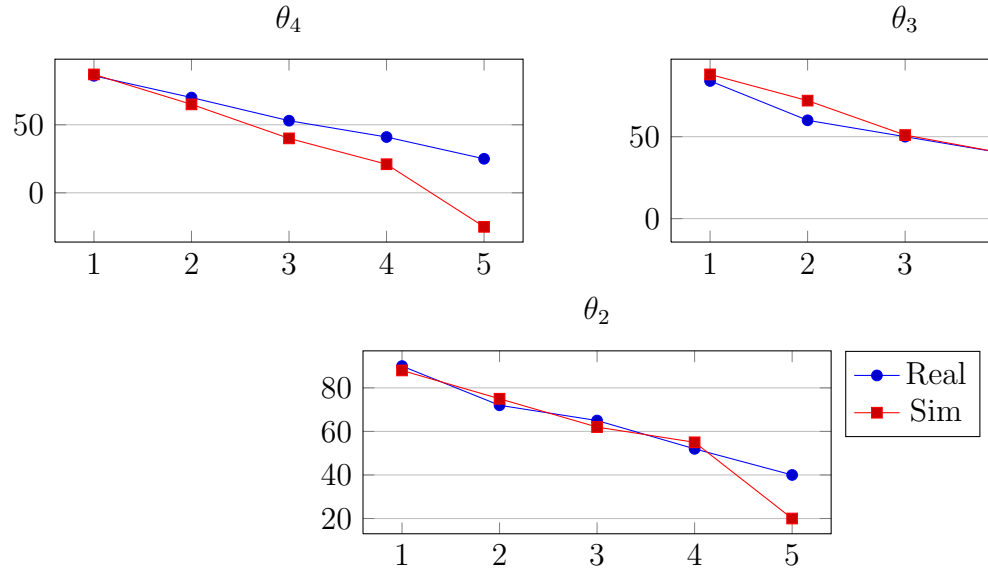
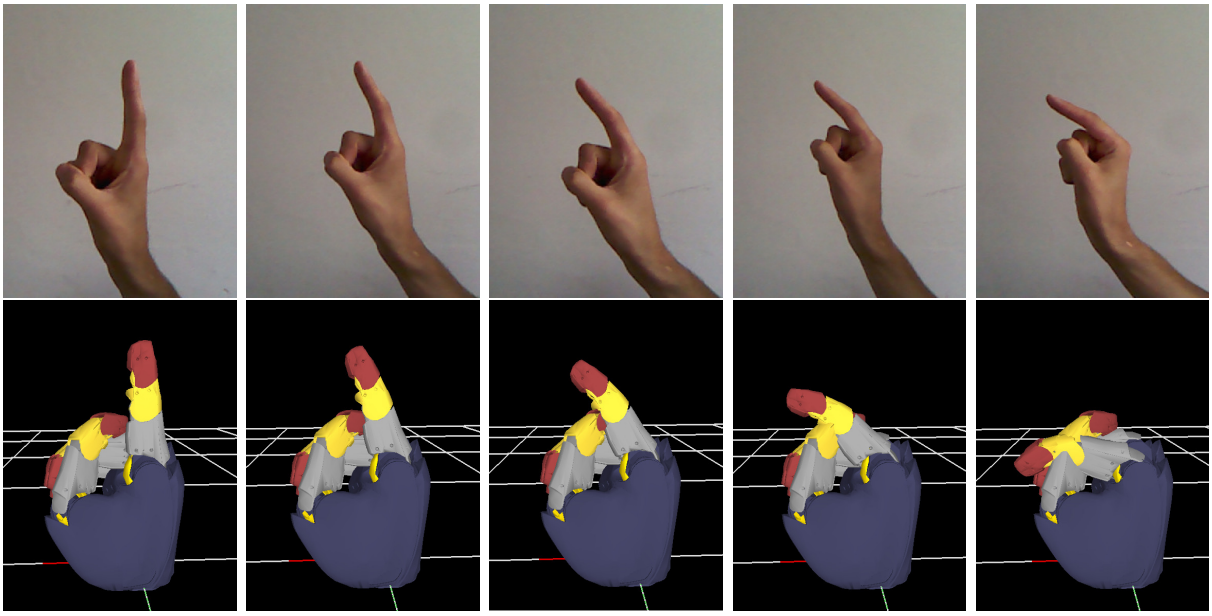


Figure 5.2: Progression of angles θ in real hand and simulation



(a) 1 (b) 2 (c) 3 (d) 4 (e) 5

Figure 5.3: Lateral view accuracy at iteration steps of bending motion

(a) Angle θ_4

θ_4	1	2	3	4	5
Real	86	70	53	41	25
Sim	87	65	40	21	-25
△	-1	5	13	20	50

(b) Angle θ_3

θ_3	1	2	3	4	5
Real	84	60	50	40	20
Sim	88	72	51	40	-5
△	-4	-12	-1	0	25

(c) Angle θ_2

θ_2	1	2	3	4	5
Real	90	72	65	52	40
Sim	88	75	62	55	20
△	2	-3	3	-3	20

(d) Deviation in %

Dev	1	2	3	4	5
θ_4	1.2	7.1	24.5	48.8	200.0
θ_3	4.8	20.0	2.0	0.0	125.0
θ_2	2.2	4.2	4.6	5.8	50.0

Table 5.1: Comparison of angles at every joint in real and simulated hand configuration

The results of this test, shown in table 5.1, display a varying deviation for different joints. It can be observed that the base joint imitates the actual movement most accurately with an average deviation of 4.2% for the first four iteration steps. Also the angle at the middle joint is computed quite accurately with a maximum discrepancy of 20% in the first four steps and a peak in the last configuration. Whereas the top joint shows a high variance in steps three to five. A probable cause is the inference of this angle from the middle joint, which cumulates the computed error and results in greater deviations.

In this project, the angles of finger joints have been mainly inferred from the relative position of the fingertip to its respective base joint. To ensure maximum smoothness of each movement, the angles of consecutive joints were derived from the base value and adapted to cover the majority of natural hand motions. As a consequence, movements as illustrated in figure 5.4 cannot be imitated exactly and will entail strong deviations at all three joints. However, this thesis focusses on providing an accurate translation of common gestures and interaction with natural movements.

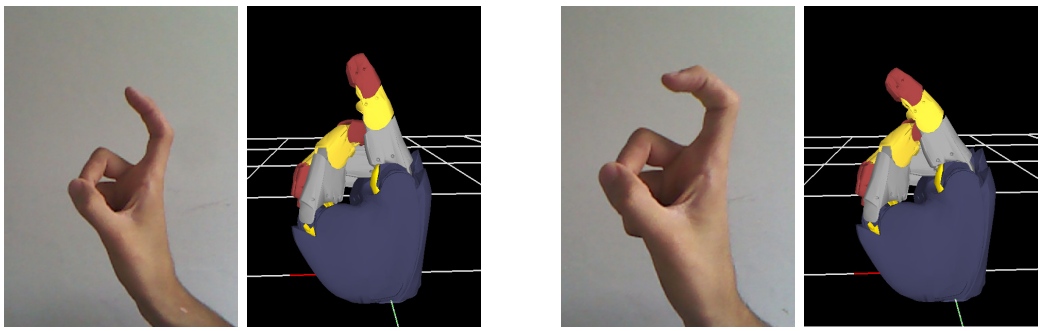


Figure 5.4: Accuracy issues at strong bending motion

5.2 Accuracy of Reassigning Fingers after Occlusion

One major factor contributing to the quality of the program is the accuracy with which fingers are numbered, as it strongly influences the pose estimation and likewise acceptability experience. In section 3.4.4 it was described that there is a difference in the level of difficulty for the reassignment of fingers for various scenarios. In the case of only one finger occluded, the recovery of its ID is fairly trivial. But when several fingers are hidden, the identification is more elaborate and assumptions like base joint positions need to be made in order to infer a possible match.

The test utilized to measure the level of accuracy analyses three different sequences of defined gestures, which were performed respectively by five independent participants. Each person was seated at the same desk with the *Kinect* camera mounted over the monitor in a fixed distance of approximately 80 cm to the participant. The setup for each candidate remained unchanged for all tests.
To evaluate the influence of prior training and therefore the level of adaptation to the

system, two candidates with no experience, one with medium practice level and two advanced users were asked to contribute to the assessment. Every participant was given five minutes to get used to the system and test the interpretation of performed hand poses. Then the order of proposed gestures was explained and the rating system illustrated.
The test is rated successful, if every single finger reappearing after occlusion is properly assigned the correct numbering and this is maintained until full recovery is accomplished. If a single finger is falsely assigned during the procedure, the test is declared as having failed. Every sequence has to be performed ten consecutive times and each performance will directly be rated.

Sequence #1

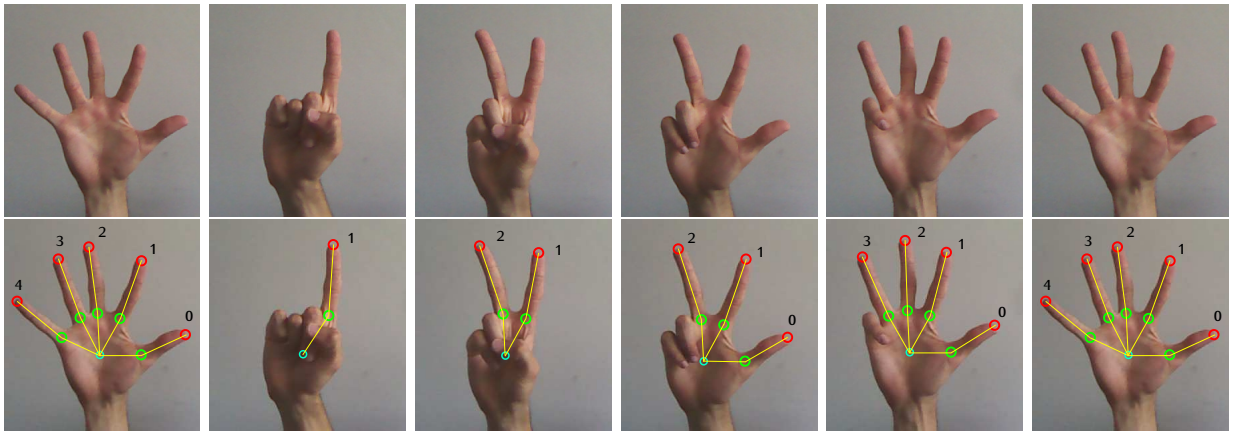


Figure 5.5: The first test sequence

Person	1	2	3	4	5	6	7	8	9	10	%
# 1	–	–	–	–	✓	–	✓	–	✓	✓	40
# 2	–	✓	✓	–	–	✓	✓	✓	✓	✓	70
# 3	✓	✓	–	✓	–	✓	–	✓	✓	✓	70
# 4	–	–	✓	✓	✓	✓	✓	✓	✓	✓	80
# 5	✓	✓	–	✓	✓	✓	✓	✓	✓	✓	90
Overall											70

Table 5.2: Participants #1 and #2 are beginner, #3 is intermediate, #4 and #5 advanced.

Sequence #2

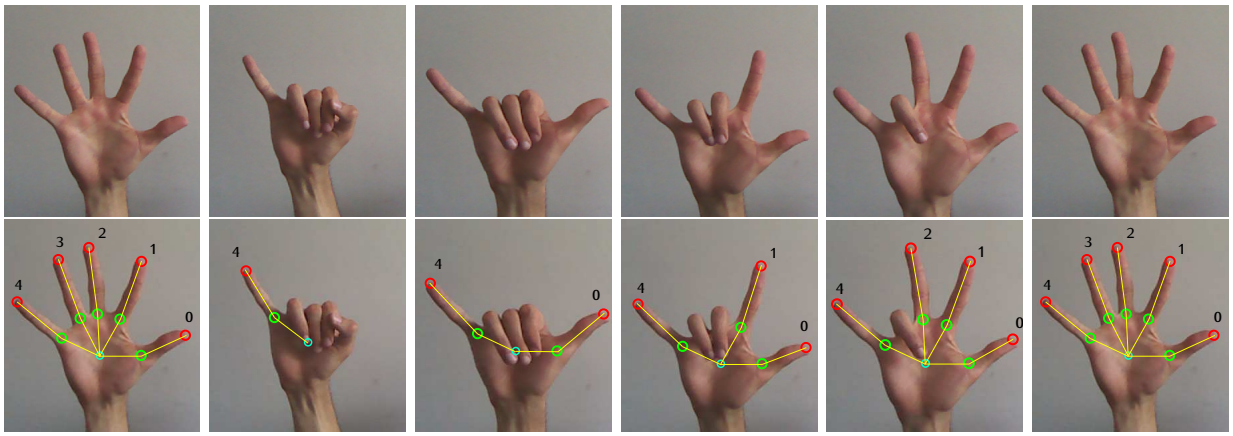


Figure 5.6: The second test sequence

Person	1	2	3	4	5	6	7	8	9	10	%
# 1	–	✓	✓	✓	–	–	✓	–	✓	–	50
# 2	✓	✓	✓	✓	–	–	✓	✓	✓	–	70
# 3	✓	✓	✓	–	✓	✓	✓	✓	–	–	80
# 4	✓	–	–	✓	✓	✓	✓	✓	✓	–	70
# 5	–	✓	✓	✓	✓	✓	–	✓	✓	✓	80
Overall											70

Table 5.3: Participants #1 and #2 are beginner, #3 is intermediate, #4 and #5 advanced.

Sequence #3

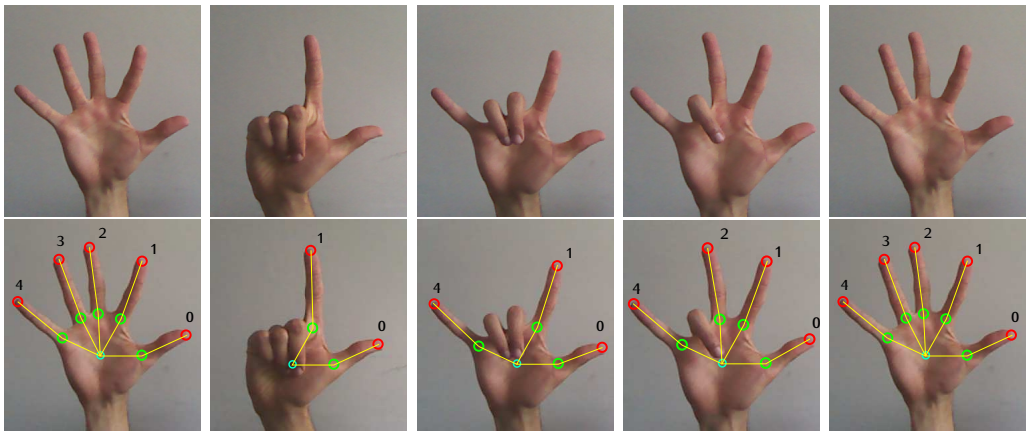


Figure 5.7: The third test sequence

Person	1	2	3	4	5	6	7	8	9	10	%
# 1	–	–	–	–	–	–	–	–	–	–	0
# 2	–	✓	✓	–	–	✓	–	–	✓	–	40
# 3	✓	–	–	–	✓	✓	–	–	–	✓	40
# 4	✓	✓	–	✓	✓	–	✓	✓	✓	–	70
# 5	✓	✓	✓	✓	✓	✓	–	✓	✓	✓	90
Overall											48

Table 5.4: Participants #1 and #2 are beginner, #3 is intermediate, #4 and #5 advanced.

Sequence #4

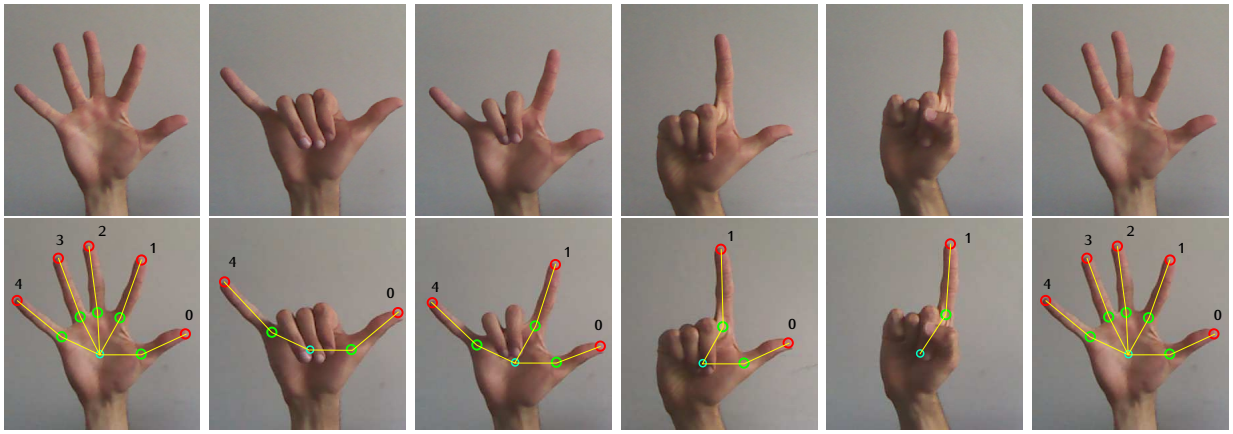


Figure 5.8: The forth test sequence

Person	1	2	3	4	5	6	7	8	9	10	%
# 1	–	✓	–	✓	–	–	–	✓	✓	–	40
# 2	–	✓	✓	–	✓	✓	–	–	✓	✓	60
# 3	✓	✓	✓	–	✓	✓	–	–	–	✓	60
# 4	✓	–	–	✓	✓	✓	✓	✓	✓	–	70
# 5	✓	✓	✓	✓	✓	✓	–	✓	✓	–	80
Overall											62

Table 5.5: Participants #1 and #2 are beginner, #3 is intermediate, #4 and #5 advanced.

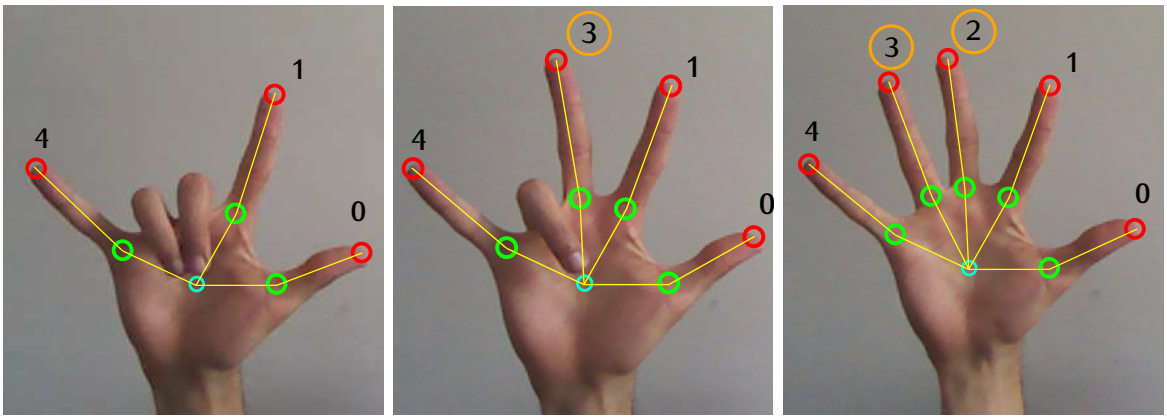


Figure 5.9: Example of wrong assignment and correction after sorting

The evaluation of all test sequences shows that there is considerable discrepancy in the performance of participants with prior training and those without practical experience. Although overall performance does not differ a lot, it can be clearly stated, that certain transitions from one hand configuration to another seem to be more difficult for some candidates. Sequence number three appears to pose problems for the first participant, who explains that stretching the little finger feels unnatural and cannot be bend further up than approximately 30 degrees. As the flexibility of the middle and top joint is connected with the agility of the base joint, the entire finger cannot be lifted sufficiently from the palm area and the resulting cluster will be too small. Therefore this posture cannot be detected and hence leads to the low score of the first person. An example of a wrong assignment and the recovery of the correct numbering is illustrated in figure 5.9.

5.3 User Feedback

A qualitative evaluation is available through analysis of user feedback and observing their interaction experience during the test. All candidates participating in the tests of sections 5.2 and 5.1 were interviewed after completing the experiments. They all gave similar comments.

Steep learning curve:
In the beginning most participants had difficulties initializing the program and after mastering this step, occasional problems placing their hand in the best possible position and angle for detection. As a consequence, this lead to only a few postures getting recognized and the program having to restart, as the tracking process lost information of feature points. However, after a short time of adaptation they reported increasing success rates up to 80%, yielding a steep learning curve when considering certain measures of adaptation.

Measures of adaptation:
In order to achieve high success rates and control the simulation model most effectively the user has to adjust his execution of gestures and movements slightly to the detection system. A common mistake of participants was to point the palm of their hand towards the simulation on the screen, thus altering the interpretation of this gesture by the camera. Additionally this accompanied a reduction of the detected cluster size, as the surface of the 2D projection of the hand was decreased. After identifying this effect, candidates adjusted the angle between the hand palm and camera center to a perpendicular posture. The second adaptation they described is exaggerating finger spread and clear execution when performing a movement. This proved to be helpful for motions with just a few occlusions and for likewise configurations with multiple hidden fingers. The third mentioned consideration is to initialize the program in a distance of about 80 cm and maintain the control in a similar distance that varies up to 10 cm closer to the camera.

Anatomy and agility:
A difference in physiological characteristics of the hand structure among all participants was noted and indicated as a relevant factor for user interaction. The hand shape and size varies with each candidate and therefore defines the maximally detectable number of pixels, which fundamentally influences the image processing accuracy. Hence, people with long fingers will experience an easier interaction and a more precise feedback even without having to adapt their movements a lot. Two participants also mentioned issues performing certain motions with their fingers independently, which could lead to a false numbering. As some fingers such as the little and the ring finger or the ring and middle finger are more closely linked by tendons, it is difficult to move these autonomously. This may invoke a brief reappearance of an adjacent limb and thus result in a wrong classification. The algorithm would falsely assign a number, sort this numbering and change the formerly detected order, due to this wrong detection.

Speed:
The translation of motions to the simulation was experienced as highly accurate and moreover completely fluent. This is clearly based on the quick real-time processing of each development step. The program runs with a rate of 20 fps and requires a complete processing time of 60 ms on a *Dell Precision T3500* with 6GB RAM and an *Intel Xeon* 2.4 GHz dual core processor. It should be noted that no optimization by multi-threading has been applied and that the computation runs without GPU support. The times for each processing step are listed in the table 5.6 below.

Processing step	Average computing time
Hand segmentation (Sec. 3.2)	$< 3ms$
Palm detection (Sec. 3.3)	$20ms$
Fingertips (Sec. 3.4.1)	$5ms$
Clustering (Sec. 3.4.2)	$10ms$
Inference of joints (Sec. 3.4.3)	$25ms$
Numbering of fingers (Sec. 3.4.4)	$< 3ms$
Calculation of angles (Sec. 3.5.2)	$< 1ms$
Total	$67ms$

Table 5.6: Average computing time for each processing step

Chapter 6

Conclusion

The goal of this thesis was the implementation of a 3D hand pose estimation system that interprets the user input and translates this information to a 3D simulation model, which in turn imitates the performed posture. The development of the proposed approach is summarized in the order of processing steps. Major challenges and respective solutions are emphasized. This is followed by an evaluation of the experiments' results and an outlook on future suggested work.

The first step of this approach was identification of the 2D position of the hand and its segmentation from the background image. Therefore the *OpenNI* hand tracking tool was utilized and a depth based filter applied to extract all relevant feature points. To recover feature points, which were discarded due to occlusions or camera measurement errors, the *FloodFill* algorithm was employed. In section 3.3, the center of the palm was detected and stabilized for a continuous, smooth rendering and further computations. Before calculating the center, erased segments were restored by merging neighbouring clusters with the *Dilate* operation and subsequently these expanded areas were reduced in size with the *Erosion* method. Furthermore, certain areas were erased manually (Sec. 3.6.2) to crop unwanted dilations and provide optimum conditions for the next step. After that, the *Distance Transformation* method was executed and the correct palm center candidate elected. The stabilization of the palm center was achieved by applying a *Kalman Filter*. In section 3.4.1, the location of all fingertips was computed by identifying the intersection of the hand's convex hull with its contour. As the convex hull touches the contour more than once, the best fit was extracted by considering various factors, such as the distance between adjacent fingertips, the distance to the center of the palm and additionally the angle between two points. In the next step, the fingers are optically separated from the palm area and then clustered utilizing the *DBSCAN* algorithm to single, logical units. Afterwards the finger base joint positions were inferred by applying the *Principal Component Analysis (PCA)*, as its first principal component gives the fitted line with the highest variance through the cluster and is less susceptible to minor

deviations. The corresponding base joint locations were again stabilized with a *Kalman Filter* and the angle between them saved in a joint map for later use. Section 3.4.4 covers the assignment of identification numbers to each finger and maintenance of its consistency. Maintenance was ensured by measuring the distance of each former centroid position to the current location. If this value exceeded a certain level, the former assignment was discarded and the reappearance of another finger had to be assumed. To infer the correct numbering for fingers detected after reappearing from occlusion, the centroid of the new cluster was compared to each joint position saved in the joint map and the finger assigned to the nearest match. In section 3.5, the 3D coordinates were assigned to each feature point and the angles calculated for the respective finger segments at each joint.

The evaluation performed in chapter 5 includes tests measuring the precision of imitating angles at the three joint locations and the accuracy of reassigning fingers after occlusion. Additionally the test participants were interviewed afterwards and their feedback noted. The analysis of the first test revealed that the detection of the angles θ_2 at the base joint and θ_3 in the middle joint are performed highly accurate with an average deviation of approximately 4.2% and 6.7%. Although the calculation of the top angle stays within boundaries during the first two iteration steps, these values progressively exceed limits due to incrementally cumulating deviations. This development is based on the joint angle correlation and can be alleviated by considering more feature points. The second test evaluated accuracy of finger reassignment and showed good results with an overall success rate of approximately 80% for advanced users. However, this outcome differs from the performance of beginners, which achieved a rate of about 50%. This discrepancy is explained in section 5.3, in which the user feedback is examined to propose measures for accomplishing better results. One discussed method of adaptation is to overemphasize the performance of movements in order to ease the assignment of fingers and accordingly the process of reassignment. Another aspect, emerged from user feedback, is the impact of physiological characteristics of the hand structure. Test participants with smaller hands reported a difference in detection, which they experienced as correlated to the hand and finger size. This factor mainly depends on the camera resolution, which is 640x480 on the current *Kinect*, making the detection susceptible for deviations in finger cluster size. Overall, the user feedback was very positive and every participant mentioned an increase in their learning curve, achieving higher results and a more accurate control, when considering certain adaptations.

For future work, even better results are expected, especially with the integration of the new *Kinect* model. This would provide a higher resolution and the possibility of receiving larger amounts of pixels per cluster, thus enabling a more detailed detection. As a consequence, the process can become increasingly independent of hand shape and the manner in which movements are performed. Another improvement could be achieved by considering multiple feature points per finger for a more accurate calculation of angles θ_3 and θ_4 at the middle and top joint. However, this refinement is almost certainly only possible for input images with higher resolution levels.

List of Figures

List of Tables

Bibliography

[ASO11] K. Abe, H. Saito, and S. Ozawa. 3-D drawing system via hand motion recognition from two cameras. In *Systems, Man, and Cybernetics, 2000 IEEE International Conference on*, pages 840–845 vol.2, 2000.

[BRB09] T. Bader, R. Räpple, and J. Beyerer. Fast Invariant Contour-Based Classification of Hand Symbols for HCI. *Lecture Notes in Computer Science*, 5702:689, 2009.

[BTG+12] Luca Ballan, Aparna Taneja, Jürgen Gall, Luc Van Gool, and Marc Pollefeys. Motion capture of hands in action using discriminative salient points. In *Proceedings of the 12th European Conference on Computer Vision - Volume Part VI*, ECCV'12, pages 640–653, Berlin, Heidelberg, 2012. Springer-Verlag.

[CRDR11] Ankit Chaudhary, J. L. Raheja, Karen Das, and Sonia Raheja. Intelligent approaches to interact with machines using hand gesture recognition in natural way: A survey. *International Journal of Computer Sciences and Engineering Systems (IJCSES)*, 2(1): pages 122–133, 2011.

[DAKP12] Paul Doliotis, Vassilis Athitsos, Dimitrios Kosmopoulos, and Stavros Perantonis. Hand shape and 3d pose estimation using depth data from a single cluttered frame. In *Advances in Visual Computing*, volume 7431 of *Lecture Notes in Computer Science*, pages 148–158. Springer Berlin Heidelberg, 2012.

[DBS] DBSCAN algorithm `http://en.wikipedia.org/wiki/DBSCAN#mediaviewer/File:DBSCAN-Illustration.svg`. [Online; accessed 05-August-2014].

[EpKSX96] Martin Ester, Hans peter Kriegel, Jörg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Journal of Data Mining and Knowledge Discovery*, 2(1): pages 226–231. Kluwer Academic Publishers, 1998.

[Goo] Flutter gesture recognition system `https://flutterapp.com/`. [Online; accessed 05-August-2014].

[GRC] Gesture control of robots
http://robotechlabs.com/wp-content/uploads/2013/11/gesture-control-robotics-workshop.jpg. [Online; accessed 05-August-2014].

[Hana] Hand kinematics
https://rmintra01.robotic.dlr.de/rmwiki/images/e/e3/HandKinematics.png. [Online; accessed 05-August-2014].

[HTLH11] Meng-Fen Ho, Chuan-Yu Tseng, Cheng-Chang Lien, and Chung-Lin Huang. A multi-view vision-based hand motion capturing system. *Pattern Recognition*, 44(2): pages 443–453, 2011.

[Int] Intel Omek gesture recognition system
http://www.intel.com/content/www/us/en/architecture-and-technology/realsense-overview.html. [Online; accessed 05-August-2014].

[JWC14] Sung-Il Joo, Sun-Hee Weon, and Hyung-Il Choi. Real-time depth-based hand detection and tracking. *The Scientific World Journal*, 2014, 2014.

[Kal] Kalman iteration cycle
http://bilgin.esme.org/portals/0/images/kalman/iteration_steps.gif. [Online; accessed 05-August-2014].

[Kal60] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D): pages 35–45, 1960.

[Kina] Kinect hardware
http://www.microsoft.com/global/en-us/news/publishingimages/Kinect_Sensor_Print.jpg. [Online; accessed 05-August-2014].

[Kinb] Kinect coordinate system
https://dgoins.files.wordpress.com/2014/03/image_thumb1.png%3Fw%3D1427%26h%3D545. [Online; accessed 05-August-2014].

[KK01] Sato Y. Koike, H. and K Kobayashi. Integrating paper and digital information on enhanceddesk: A method for realtime finger tracking on an augmented desk system. *ACM Trans. Comput.-Hum. Interact.*, 8(4): pages 307–322, December 2001.

[KMN+02] Tapas Kanungo, David M. Mount, Nathan S. Netanyahu, Christine D. Piatko, Ruth Silverman, and Angela Y. Wu. An efficient k-means clustering algorithm: Analysis and implementation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7): pages 881–892, 2002.

[Lea] Leap motion gesture recognition system `https://www.leapmotion.com/`. [Online; accessed 05-August-2014].

[LWH00] J. Lin, Ying Wu, and T.S. Huang. Modeling the constraints of human hand motion. In *Proceedings of Workshop on Human Motion, 2000.*, pages 121–126, 2000.

[OKA11] I. Oikonomidis, N. Kyriazis, and A. Argyros. Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, pages 2088–2095, Nov 2011.

[CSA12] S. Chandran and A. Sawa. Real-Time Detection and Understanding of Isolated Protruded Fingers. In *Computer Vision and Pattern Recognition (CVPR), 2004. Proceedings of the 2012 IEEE Computer Society Conference on*, volume 10, pages 152 vol.10, 2012.

[SMC01] B. Stenger, P.R.S. Mendonca, and R. Cipolla. Model-based 3d tracking of an articulated hand. In *Computer Vision and Pattern Recognition (CVPR), 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 2, pages 310–315 vol.2, 2001.

[SMHL00] M. Soriano, B. Martinkauppi, S. Huovinen, and M. Laaksonen. Skin detection in video under changing illumination conditions. In *Proceedings of Pattern Recognition (ICPR), 2000. 15th International Conference on*, volume 1, pages 839–842 vol.1, 2000.

[SOT13] S. Sridhar, A Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using rgb and depth data. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 2456–2463, Dec 2013.

[TYK13] Danhang Tang, Tsz-Ho Yu, and Tae-Kyun Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Computer Vision (ICCV), 2013 IEEE International Conference on*, pages 3224–3231, Dec 2013.

[VdBK09] M. Van den Bergh, E. Koller-Meier, F. Bosche, and L. Van Gool. Haarlet-based hand gesture recognition for 3d interaction. In *Applications of Computer Vision (WACV), 2009 Workshop on*, pages 1–8, Dec 2009.

[WP12] Wei Wang and Jing Pan. Hand segmentation using skin color and background information. In *Machine Learning and Cybernetics (ICMLC), 2012 International Conference on*, volume 4, pages 1487–1492, July 2012.