Realistic Hands: Multi-Modal 3D Hand Reconstruction using Deep Neural Networks and Differential Rendering

Michael Seeber

Master Thesis
May 2021

Prof. Dr. Marc Pollefeys

Advisors:
Dr. Martin R. Oswald
Dr. Roi Poranne

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Computer Vision and Geometry Lab

CRL

Abstract

One of the primary objectives for Virtual Reality (VR) systems is to immerse the user in an artificial environment and cause the perception of actually being there. However, this full immersion is weakened by the need to use unnatural devices like controllers to interact with the environment. As a result, hand tracking has become a prominent interaction method for VR recently. While this has made interactions less artificial, current systems represent the hands in very generic ways and therefore still lack the personal feel. We propose to introduce a new system which relies on head mounted cameras to generate realistic hands through a novel hybrid architecture utilizing deep neural network and differential rendering based optimization. Our approach outperforms other methods in the task of monocular 3D hand pose and shape estimation by fusing input modalities. We also demonstrate state of the art performance in the task of hand segmentation and elaborate how to further improve robustness by extending towards a stereo setting, which is prevalent in VR headsets. Additionally, we are one of the first to explore the accompanying generation of realistic textures, which comprises personal skin characteristics for a truly personalized virtual hand.

Acknowledgments

I want to thank my advisors Dr. Martin Oswald and Dr. Roi Poranne for their enormous support, advice and valuable input during all the different stages of this thesis. Without their outstanding assistance and dedicated involvement, this thesis would never have been accomplished. From the first meeting with Roi, that initiated this thesis, to the many following, I was always able to rely on his and Martin's guidance and experience. Moreover, I want to further thank Roi for lending me a GPU and especially highlight the weekly meetings with Martin, where he constantly provided helpful feedback and challenged me on my ideas. I am also grateful to Prof. Dr. Marc Pollefeys for giving me the opportunity to work on this project and providing me with computing resources to execute it.

Contents

List of Figures

List of Tables

1

Introduction

1.1. Motivation

Parallel to the constant increase in computational power, the interfaces between humans and machines have evolved continuously as well. The first computers used punch cards as the primary medium for input, which was followed by continuously more easily manageable command line interfaces. Subsequently, the first graphical user interfaces started to appear, making computing accessible to the general public for the first time. Similarly, multi-touch interfaces played a crucial role in the rise of mobile computing, which further expanded the user base by reducing entry barriers. What all these revolutions have in common is that every new generation of interaction was less artificial than the previous and therefore enabled a larger number of users to access more functionalities. With the current emergence of new computing platforms such as Virtual Reality (VR) and Augmented Reality (AR) we are standing on the verge of another potential evolution in human-computer interaction.

As VR/AR headsets get more sophisticated and the boundaries between reality and the virtual world start to vanish, interaction methods move into focus as unnatural input devices, such as controllers, diminish the full immersion. Due to their dexterous functionality, hands are the most effective general purpose interaction tool available to humans and are therefore a prime candidate for replacing those devices. By using hands directly, humans' natural communication and manipulation skills can be employed for intuitive user interfaces while gaining access to a very expressive input method with multiple degrees of freedom. For example, sculpturing 3D objects with hands becomes as natural as working with clay and removes the limitations posed by mouse and keyboard.

In contrast to VR, AR headsets started to opt for hand tracking as the primary input method very early on. This is encouraged by the already available cameras used to keep track of the environment. By reusing them for vision based hand tracking, no additional devices are required.

The enhanced mobility gained by the expendability of controllers further removes friction and is vital to in the field operation of such devices by surgeons or engineers. Additionally, as AR is merely blending real world with digital content, there is also no need to replicate the hands in digital space since the real hands are visible and only the intended actions need to be understood.

Lately there has also been a shift to inside-out tracking for VR headsets, which means that the tracking hardware is included in the headset in the form of multiple cameras similar to AR devices. This change fueled a new interest in turning hand tracking into the principal interaction method for VR as well. Compared to AR headsets, VR completely replaces what is seen and experienced, so there is the additional need to also represent hands in the virtual world. A drawback of current systems is that they represent hands in very generic ways and therefore lack the personalized feel. The feeling of self-presence, however, is an especially important prerequisite for an immersive experience.

Surprisingly, very few works [1] go beyond hand pose estimation on to realistic hand reconstruction where the appearance and shape of hands is also taken into account. With this thesis we intend to provide a first glimpse at how such a problem could be tackled.

1.2. Background

The most effective tool for hand motion capturing are electro-mechanical or magnetic sensing devices which come in the form of gloves. While they are able to deliver accurate real time measurements, they come with several drawbacks. Not only are they expensive and require complex calibration, but they also hinder the naturalness of hand motion and thus make the whole experience artificial.

As modern VR/AR headsets already include multiple cameras, vision-based hand pose estimation represents a promising alternative in order to provide more natural and unencumbered interactions without the need for additional hardware. Both pose estimation and also 3D reconstruction are long standing computer vision problems which have been studied for decades. The main challenges are accuracy and processing speed, which often conflict. Pioneering work relied on low level visual cues, such as edges, skin color, and similar features. In general, these are prone to tracking errors and not resilient to diverse backgrounds found in the wild. The improved availability of depth sensors revolutionized the field of body pose estimation and also impacted the research of hand pose estimation as they can yield more robust features. However, due to self-occlusion, finger self similarity, and higher requirements on depth noise levels, it is a more difficult task to solve. For such optimization-based methods a parametric model of the human hand is fitted to match specific features. Typical 2D joint locations best explain the observation and are often used as an optimization target.

With the emergence of deep learning, the dilemma between optimization-based approaches and regression-based methods became more relevant than ever for many computer vision tasks. While the results of optimization-based methods are promising, the methods tend to be slow and also sensitive to the chosen initialization. On the other hand, deep learning approaches can directly infer the parameters of a model. In theory, this approach is more powerful as it can take

all pixel values into consideration instead of relying on a sparse set of features. In practice, this one-shot prediction often leads to mediocre image-model alignment.

1.3. Problem statement

As previously mentioned, realistic modeling and rendering of human hands can be very valuable for virtual reality environments, as humans are extremely sensitive towards the physical appearance of their own hands. Due to complex poses and large shape variability exhibited by hands, this is a rather difficult task. Given a monocular or stereo camera stream, as found on many VR devices, we want to infer a 3D mesh that resembles the appearance of the user's hand in real time. This includes not only pose, size and shape, but also the generation of a matching realistic texture which comprises personal skin characteristics including, for example, possible tattoos.

1.4. Challenges

The challenges faced by 3D hand pose estimation are very similar to closely related tasks, such as body pose estimation [2] or facial landmark localization [3]. Unfortunately, most of the challenges arise in more complex configurations for hand pose estimation. Additionally, reconstructing individual hand meshes with the appropriated personal textures introduces further difficulties.

High Degrees of Freedom

Hands are highly articulated and come with multiple degrees of freedom due to the many joints this body part exhibits. Although these are not independent of each other and there exist kinematic constraints that restrict the pose space, this is challenging for learning-based methods like ours. To successfully learn meaningful statistical priors, large amounts of representative training data, ideally covering the full pose space, are required .

Self similarity

A difficulty in body pose estimation is the similarity between the left and right limbs, which can lead to erroneous detections. When estimating hand posture, the ambiguities are even more severe since all fingers have a similar appearance. Due to the exhibited self-similarity it is difficult to distinguish between different fingers and global information has to be taken into account to draw meaningful conclusions.

Occlusion

Occlusions pose another significant problem. Whereas key point locations of rigid objects can be deterministically derived from non-occluded parts, this is only possible to a limited extent for articulated objects. Even with cues from the visible parts, occluded joint locations can only be estimated using kinematic constraints or statistical priors of the pose space. In an egocentric viewer setting such as ours, this is exceptionally difficult: Self occlusion is very common since the palm or the forearm can completely conceal the fingers. In such cases, where the exact locations are not recoverable, we have to rely on predicting likely distributions of possible locations.

Fast motion

Humans are capable of moving their arms and hands very rapidly. For example, closing and opening the hand at a natural speed only takes 60ms [4]. These fast movements can lead to problems on different levels. For example, a well known side-effect is motion blur that deteriorates the quality of the input image and can cause dilution of the important finger contours. Additionally, depending on the frames per second (FPS) rate of the camera, there can be large differences between two consecutive frames, which makes it problematic to use previous frames as initialization to constrain the pose space. Furthermore, it makes performance optimization such as relying on interpolation between predicted frames less resilient. As a consequence, low latency is essential to enable adequate user interactions.

Various hand sizes and shapes

Human hands show high variability in their size and shape, depending on factors such as body height, gender and others. For robust tracking, it is important that the used hand model supports different sizes and shapes. While size is crucial and commonly integrated, shape is more subtle and related works have so far spent little effort on dealing with these variants.

Limited & noisy data

A reason why hand shapes have not been extensively explored is the fact that there is not enough data available to model them. Most datasets include only few participants and therefore only capture a tiny part of the hand shape variance. The lack of appropriate training datasets is also noticeable for a problem such as ours, where we would ideally have a stereo captured egocentric dataset with corresponding annotations available. Although it is possible to resort to synthetically generated data, there is a large domain gap given the significant discrepancies between real and synthetic images. In addition, the available large scale dataset annotations have been created using automatic annotation procedures, which means there is no guarantee for correctness. For example, the previously mentioned occlusion problem can also negatively influence the accuracy during dataset generation and lead to noisy data.

Segmentation

In the hand pose estimation literature, segmentations are often assumed to be a prerequisite. While in lab environments a simple approach such as color thresholding can be used, this strategy does not work in real-world environments and makes segmentation a non-trivial task in itself.

1.5. Contributions

Our key contributions are as follows:

1. We predict 3D mesh parameters with high accuracy by fusing input modalities.
2. We improve the resilience of our method by additionally exploiting stereo views.
3. We predict a hand segmentation and optimize the fine grain fit of the predicted MANO mesh through differential rendering.
4. We output a personalized hand mesh and therefore go beyond simple skeleton rigging.
5. We compute textures for a truly realistic hand mesh based on camera captured data.

To summarize, our hybrid approach is more expressive and personalized than work relying on the MANO hand model, but also more efficient and robust than methods that directly try to infer a mesh. Additionally, we are one of the first to explore texturing based on camera input, yielding a truly unique and personalized hand mesh for use in VR.

1.6. Thesis Organization

This thesis is structured into a total of six chapters, including this introduction. In Chapter 2 we examine existing works related to hand pose estimation and hand reconstruction. It is followed by Chapter 3, in which we review theoretical concepts associated with this work. In Chapter 4, we give insights into our method by providing an overview of the proposed architecture before discussing the implementation details of every component. In the sixth and last chapter, we conclude our findings and provide an outlook for future work.

2

Related Works

This thesis has been inspired by several other papers in the research areas of hand tracking, pose estimation and hand reconstruction. All fields have been thoroughly studied and provide valuable concepts that have been crucial while exploring our novel approach. In the following sections, we review the most relevant works related to this thesis.

2.1. Hand Detection and Tracking

Hand detection and tracking is one of the fundamental techniques involved in any form of camera based hand analysis. Whether one is interested in pose estimation, hand reconstruction or grasp analysis, identifying and tracking the hand position in video frames is inherent to all of them, as they all require hand crops as input.

Early methods tried to detect hands through skin color models, which only lead to limited success on unconstrained images. To overcome this, Mittal et al. [5] proposed a method that combines three independent detectors (hand-shape, context-based and skin-based) for robustness and precision. With the rise of data driven methods, the work from Bambach et al. [6] was one of the first to use convolutional neural nets instead of relying on strong prior assumption, such as that every skin pixel belongs to a hand. Very recent methods [7] are even able to infer richer hand representations which include not only the location and side of the hand, but also extend to contact states of interacting objects. One drawback of such sophisticated solutions is that they come along with increased inference times, which makes them often unusable for real time applications. Therefore, another branch of research is focused on optimizing the methods for fast processing times to enable real time and mobile applications. One such hand detection method with fast processing times is MediaPipe Hands [8]. They deployed a two-stage approach consisting of a palm detector locating the approximate hand location whenever tracking is lost. Once a palm is identified a more fine grained hand landmark detector continues with local hand tracking.

2.2. 3D Hand Pose Estimation

3D hand pose estimation is the process of attributing hand joints with their respective alignment so that a 3D hand skeleton can be recovered. Early methods [9] used low level visual cues such as edge maps and Chamfer matching [10] to rank the likelihood of possible predefined poses. The advent of low cost depth cameras paved the way for unconstrained 3D hand pose estimation and research focused on using depth data as input. Oikonomidis et al. [11] proposed a method based on Particle Swarm Optimization (PSO). Sharp et al. [12] made the approach more robust and flexible through adding a re-initialization strategy. Likewise, Tagliasacchi et al. [13] used depth images as input, but adopted Iterative Closest Point (ICP) optimization. Although these methods perform well, the application range is limited by the need for depth sensors. Therefore, Zhang et al. [14] replaced the active depth sensors with stereo cameras and enhanced the disparity maps with prior hand knowledge. Panteleris et al. [15] also used stereo cameras, but they optimized for color consistency between the two views.

With the success of Convolution Neural Nets (CNN) in other research areas, it did not take long before they were applied in the context of 3D hand pose estimation. In contrast to optimization based methods, the focus moved from estimating angles of hand joints to predicting the keypoint joint locations directly. Initial methods [16] directly took the depth image as input and processed them with 2D CNNs. Other methods argue that 2D CNNs are not directly suited for such a task, due to the lack of 3D spatial information. However, because hands feature a large scope of viewpoints for a given articulation, it is also hard to define a canonical viewpoint. To alleviate this problem, different intermediate 3D representations have been studied. Ge et al. [17] projected an obtained pointcloud onto multiple views to better utilize depth cues. Further refined methods work with D-TSDF [18], voxels [19] or pointsets [20].

More recent work started to rely on only single RGB images as input. This is a sophisticated challenge due to being an ill posed problem, given only 2D vision of the hand and no depth data whatsoever. This change in input modality further fueled the switch from model to data driven approaches, as the methods became applicable for in-the-wild images. The first work that tackled the problem of 3D hand pose estimation from a single RGB image with a learning based formulation was created by Zimmermann et al. [21]. One of the limiting factors of such methods is the amount of annotated training data, which is crucial to learning how to resolve the inevitable depth ambiguities. Mueller et al. [22] used generative methods to translate synthetic training data into more realistic looking training images. Because there is a large discrepancy between factors of variation ranging from image background and pose to camera viewpoint, Yang et al. [23] introduced disentangled variational autoencoders to allow specific sampling and inference of these factors. In contrast to the mentioned generative approaches, other methods [24] make up for the lack of annotated training data by utilizing a depth camera for weak supervision of the training procedure.

2.3. 3D Hand Reconstruction

The general problem of 3D reconstruction has a long history in the field of computer vision. It is the process of capturing the shape and appearance of real world objects and replicating

a digital representation. The previously discussed 3D pose estimation methods yield skeletal representations of the hand. While this is expressive enough to describe all different kinds of poses, they are not concerned with reconstructing the full hand in form of a 3D representation such as a mesh. Although it is possible to align a generic hand mesh to the skeleton, this is fundamentally different from 3D hand reconstruction, as the visual appearance determined by fast variation of hand shapes is not considered at all.

In the related field of body pose reconstruction, multiple approaches have been explored. This includes, on the one hand, completely 3D supervised methods [25] which require vast amounts of annotated training data, but also more constrained parametric deformable surface models, like SMPL [26], were exploited by various methods. Examples for SMPL based full body 3D reconstruction includes the work by Kanazawa et al. [27], where an end-to-end CNN with an additional adversarial is used to decide whether a given prediction is realistic or not. Kolotouros et al. [28] tried to improve the alignment of predicted SMPL meshes by performing iterative optimization to fit the regressed shape on 2D joints.

Inspired by the success of SMPL, a similar parameterized model for hands was proposed with MANO [29]. This enabled shape aware hand mesh reconstruction methods, such as the work from Boukhayma et al. [30], which used single RGB inputs together with OpenPose [31, 32] hand keypoint detection to regress MANO parameters. The method proposed by Zhang et al. [33] follows a similar approach, but the keypoint heatmap is regressed implicitly. A more robust approach that deals with severe occlusion caused by hands that manipulate objects is introduced by Hasson et al. [34].

Although hand models like MANO are able to represent a large amount of hand shapes, they are still fundamentally limited in their expressiveness. Therefore, Ge et al. [35] proposed a model-free method that uses a graph convolutional network to directly regress vertex positions. Another alternative approach to obtain vertices directly is explored by Moon et al. [36], where they introduce a novel image-to-lixel (line + pixel) prediction network which retains the spatial relationship between pixels in the input image. Smith et al. [37] take 3D hand reconstruction to the extreme, by utilizing 124 cameras to capture a high resolution footage from different perspective and processing times of more than 20 minutes for a single frame. However, not only are they able to replicate the hand pose and shape with impressive accuracy, but they also capture details such as elastic skin deformations that occur when a finger is pressed against the palm.

3

Theory

In this chapter we review preliminary theory and concepts required to understand the methods introduced in this thesis. We do expect the reader to have a basic understanding of neural networks and accompanying concepts.

3.1. Camera Model

For most computer vision tasks, one requires an understanding of how an image is created in the first place to recover knowledge about the captured 3D scene. In the real world, a camera acts as projector of 3D space to a 2D image. Therefore, modeling such a camera to replicate this projection is one of the most essential tools in computer vision.

3.1.1. Pinhole Camera

In a perspective setting the transformation from 3D camera coordinates to 2D image coordinates is performed by the intrinsic camera matrix K, which models an ideal pinhole camera. More specifically, a point $\boldsymbol{X} = (x, y, z)^T$ is projected onto the intersection of the image plane with the line between $\boldsymbol{X}$ and the camera center $\boldsymbol{C}$, as illustrated in Figure 3.1.

Intrinsic Parameters

The mentioned intrinsic camera matrix K controlling the projection is defined as follows:

$$K = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix}$$

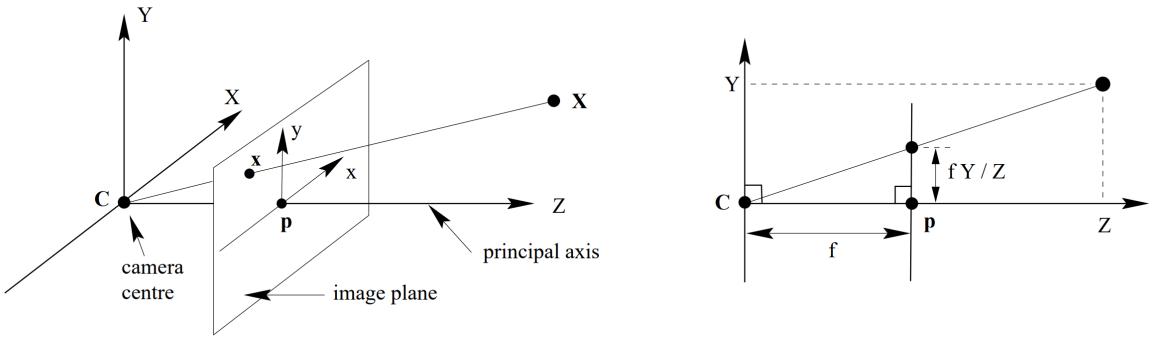


Figure 3.1.: Illustration of the pinhole camera geometry. *Points from the 3D camera coordinate system originating at the camera center C are mapped onto the image plane where the line $X \leftrightarrow C$ intersects the image plane. Taken from [38].*

The focal length f corresponds to the distance between the camera center $\boldsymbol{C}$ and the image plane. The camera's principal axis is perpendicular to the image plane and the intersection point is referred to as principal point $p = (p_x, p_y)$.

This yields the following mapping of 3D points to the 2D image plane:

$$\boldsymbol{X}_{camera} = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} fx/z + p_x \\ fy/z + p_y \end{pmatrix} =: \boldsymbol{x}_{screen}$$

Extrinsic Parameters

With only intrinsic parameters, we assume the camera is located at the origin of the world coordinate system. However, this is usually not the case which means we need to take the position of the camera in world space into account as well. The extrinsic parameters are also essential in a stereo camera setting where we need to model a small offset between the two cameras. We denote the transformation from world coordinates to camera coordinates corresponding to the extrinsic parameters as $[R|t]$, where $R \in \mathbb{R}^{3\times3}$ represents a rotation and $t \in \mathbb{R}^{3\times1}$ a translation. Combining everything we get:

$$\boldsymbol{x}_{screen} = K\boldsymbol{X}_{camera} = K[R|t]\boldsymbol{X}_{world}$$

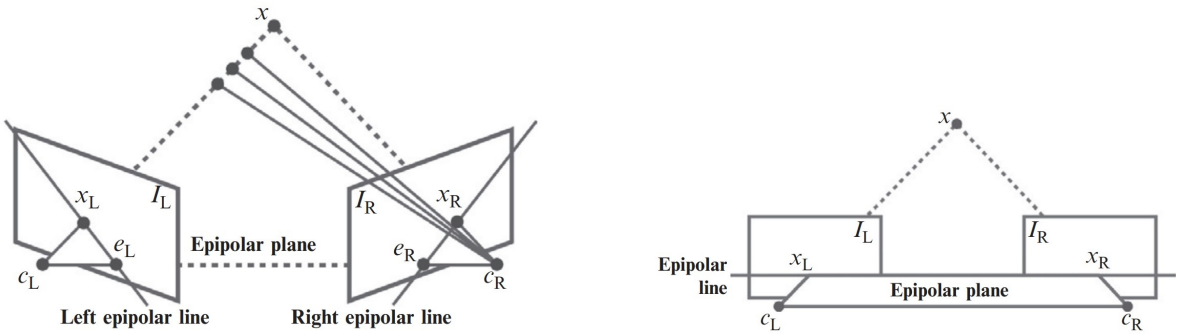
3.1.2. Stereo Cameras

In a stereo camera setting we have two camera views with image planes I_L (left) and I_R (right) which stand in a geometric relation to each other. This is similar to the human eyes, which see the world from slightly shifted vantage points. Compared to monocular vision, stereo vision has the ability to resolve projection ambiguities through imposed constraints, which enables the

extraction of information such as depth. The relations between the linked views are described by epipolar geometry, which we will quickly review.

Figure 3.2(a) depicts two pinhole cameras looking at the same point x with c_L and c_R representing the respective optical centers. The points x_L and x_R denote the projections of point x on the image planes. Since the optical centers of the cameras are distinct, each center projects onto a distinct point in the other image plane, which are called epipolar points and denoted by e_L and e_R. The line $c_L \leftrightarrow x$ is seen by the left camera as a single point due to being directly in line with the optical center. However, the right camera sees this line as line ($x_r \leftrightarrow e_R$) in its image plane, which is called epipolar line. Symmetrically, $c_R \leftrightarrow x$ is seen by the right camera as a point and as an epipolar line by the left camera. One important observation is that if the relative positions of the two cameras as well as the points x_L and x_R are known, we are also aware of their respective projection lines. Therefore, if two image points correspond to the same 3D point x, it implies that the projection lines must intersect precisely at x and the position can be calculated through triangulation.

If the image planes coincide, the epipolar geometry is simplified, as illustrated in Figure 3.2(b). In this case, the epipolar lines also coincide, i.e. $e_L \leftrightarrow x_L = e_R \leftrightarrow x_R$. Moreover, as the epipolar lines are parallel to the optical center, they can be aligned with the horizontal axis of the images. This is helpful as for each point in one image, the corresponding point in the other image can be found along a horizontal line. As stereo cameras are usually not perfectly aligned, the images can be transformed to emulate such a common image plane with a process called stereo rectification.



(a) Epipolar geometry of two pinhole cameras.

(b) Rectified setting with a common image plane and horizontal epipolar lines.

Figure 3.2.: Illustration of the stereo camera geometry.

3.2. Hand Pose Estimation

We adopt 21 pre-defined hand landmarks as our keypoints, which roughly resemble the joints of a hand with additional keypoints for the fingertips and palm. There exist small variations in literature, especially in regard to the order of the keypoints. For this thesis, we rely on the common mapping from OpenPose [31, 32], which is illustrated in Figure 3.3.

Therefore, a hand pose $P \in \mathbb{R}^{21}$ can be expressed as $P_k = (x_k, y_k) \in \mathbb{R}^2, k \in [0, ..., 20]$

in 2D. As the coordinates correlate with specific image features, convolutional neural networks are able to extract them. Moving to 3D space, a hand pose can be expressed in a similar manner by $P_k = (x_k, y_k, z_k) \in \mathbb{R}^3, k \in [0, ..., 20]$. If an additional depth sensor is at hand, inferring the z coordinate is trivial. By capturing a scene from two viewpoints, stereo cameras are capable of extracting depth knowledge. However, especially at close range, the depth resolution is not good enough to support a straightforward implementation. Thus, regressing to 3D coordinate keypoints from a single image resembles a more complicated task that requires prior knowledge about the camera model to understand it.

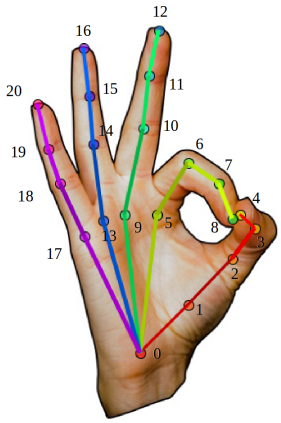


Figure 3.3.: Definition of hand landmarks as keypoint annotations. *The hand joints as well as the fingertips and palm serve as 21 keypoints. Taken from [31].*

3.3. Hand Model

As outlined in Section 3.2, recovering hand poses in 3D is a rather difficult task, due to their relatively small size, self-occlusion and other ambiguities. Although hands can come in many poses and shapes, there are fundamental physical properties that can be exploited to cope with the mentioned problems. A recently introduced statistical hand model is MANO [29] (hand **M**odel with **A**rticulated and **N**on-rigid def**O**rmations). One of the key contributions is that it factors geometric changes into those inherent to the identity of the subject and those caused by pose. Therefore, not only hand poses, but also different hand shapes can be represented with it. Before we can examine and discuss the model in more detail, we review some general related concepts.

3.3.1. Linear Blend Skinning

Linear Blend Skinning (LBS) is a widely used technique to deform meshes based on skeletal structures. The popularity of LBS mainly derives from the possibility to directly evaluate it, which makes it not computationally intensive.

To better understand LBS, we quickly review rigid skinning. In rigid skinning each vertex

is assigned to exactly one bone of the skeleton. After a transformation T_i of the ith bone the new position of a vertex v_j can be written as $v'_j = T_i v_j$. One problem of this method is that it leads to unwanted discontinuities in the boundary regions, where neighboring skin vertices are assigned to different bones.

To overcome this, LBS assigns each vertex to more than one bone and blends the bone transformations using skinning weights. The weights define the influence each bone has on mesh vertices. Hence the positions after a transformation can be computed as weighted average of the positions obtained from each bone via rigid skinning, i.e. $v'_j = \sum_i w_{ij} T_i v^i_j$

3.3.2. Blendshapes

Blendshapes are often used together with skeletal techniques like LBS. The idea behind the method is to have a source as well as one or multiple deformed target versions of the mesh. The vertices are interpolated in between, so that one can smoothly blend between the base shape and one or several morph targets. If the target is set to the posed mesh, with all unwanted discontinuities from LBS removed, this corrective blendshape can be blended in to obtain a smooth surface during the bending of joints.

3.3.3. MANO Hand Model

The MANO model is based on SMPL [26], a model for capturing body poses. The general formulation of the model M is as follows:

$$M(\beta, \theta) = W(T_P(\beta, \theta), J(\beta), \theta, w)$$
$$T_P(\beta, \gamma) = \overline{T} + B_S(\beta) + B_P(\theta)$$

where W: LBS function with blend weights w

T_P: personalized hand mesh

$\overline{T}$: template hand mesh

J: joint locations (kinematic tree)

θ: pose

β: shape

A hand mesh T_P together with joint locations J that form a kinematic tree, pose parameters γ, shape parameters β and blend weights w serve as input to a skinning function W that, in the case of MANO, corresponds to linear blend skinning (LBS). Unlike standard LBS models, the base hand mesh T_P is a function of the pose and shape of the hand itself. The blendshape function B_S allows the base hand shape to vary and the pose blendshape function B_P captures deformations of the mesh as a function of the bending of the joints. As mentioned, traditional

LBS suffers from collapse at the joints, which leads to unwanted discontinuities. Therefore, corrective blend shapes for the MANO model were learned from hand scans to correct these artifacts. Combined, this allows for natural-looking finger bending.

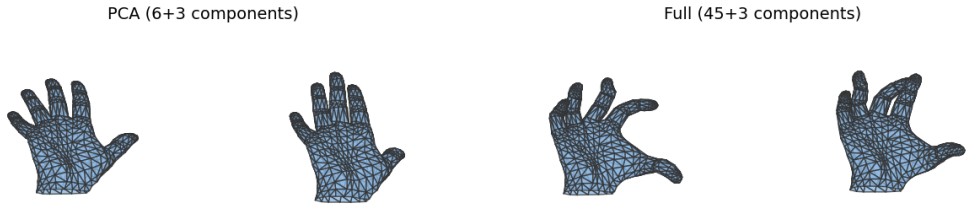


Figure 3.4.: Comparison of full MANO model to PCA version. *The first two visualizations are created by randomly choosing from the reduced PCA pose space. The last two visualization are created by randomly choosing from the more expressive full pose space.*

The model can be posed, either by specifying the full rotations for each joint or using parameters of a computed PCA space from training scans. A low dimensional PCA pose space makes it easier and more robust to fit, but also limits the expressiveness, as visualized in Figure 3.4. In our method, we utilize the full pose space and provide rotations for all 15 joints. In Figure 3.5 one can see how the selective joints can be rotated in the full model. The blendshapes that allow representation of different hand shapes are represented by a linear subspace of blendshapes that were computed with PCA. We visualized some samples in Figure 3.6.

Figure 3.5.: Visualization of the MANO pose space. *The respective joints are articulated in the visualizations .*

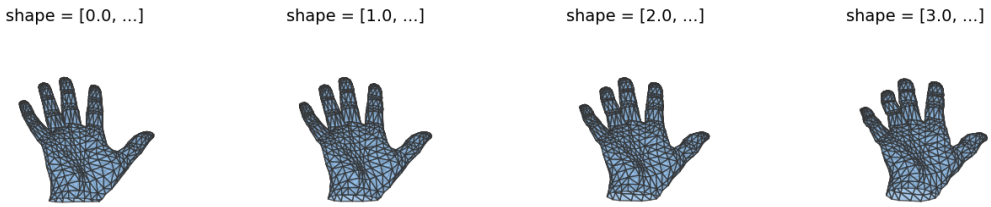


Figure 3.6.: Visualization of the MANO shape space. *The visualizations show how the shape parameters can modify the hand shape.*

4

Method

This chapter presents the methodological approach employed to solve the questions posed in this thesis. We start with a high-level overview on the architecture of a single camera and continue by providing more thorough descriptions of each component. Afterwards, we elaborate how the architecture can be extended towards a stereo setting in order to fully utilize the available sensors for robustness. This is followed by a comprehensive description of the training framework including the FreiHAND dataset and our own synthetic stereo dataset. Finally, we outline the complete inference pipeline from raw camera stream to rendering the textured mesh. To fully understand this chapter, basic knowledge of neuronal networks and deep learning as well as concepts explained in Chapter 3 are required.

4.1. Overview

Our method aims to estimate a textured 3D hand from monocular or stereo inputs. An important aspect is that we go beyond estimating only parameters of a hand model by having an additional differential rendering-based optimization step for fine grained adjustments. This makes our approach more expressive compared to solely relying on the MANO hand model. Additionally, this leads to increasingly accurate fits which enable us to rely on projections with basic filtering to yield an accompanying texture for a truly unique and personalized digital hand replication.

Our proposed framework for the monocular setting is illustrated in Figure 4.1. The inputs consist of an RGB hand crop and hand keypoints encoded in a 21 channel heatmap where each channel represents a hand joint. Details on how the heatmaps are obtained are described in subsequent Section 4.6.1 where we recount the whole inference pipeline. Both inputs are fed into our deep encoder-decoder network, which produces three outputs. These include a segmentation mask indicating the probability of a pixel belonging to a hand part. Further, we predict parameters $m = (pose, shape)$ of the MANO hand model, which consists of $3 + 45$ pose and

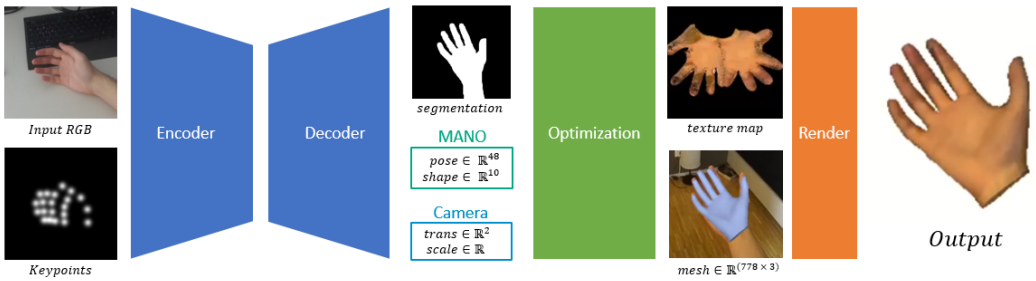


Figure 4.1.: High level overview of our method framework. *An encoder-decoder architecture is used to obtain MANO and camera parameters, followed by a differential rendering based optimization step.*

10 shape parameters. As introduced in Section 3.3.3, the pose parameters correspond to the global rotation as well as the 21 joint rotations in angle axis representation. The shape is determined through 10 PCA components. Lastly, we obtain weak-perspective camera parameters $c = (trans, scale)$ which allow to project a posed 3D hand model back onto the input image and obtain absolute camera coordinates. Here, $t \in \mathbb{R}^2$ is the 2D translation on the image plane and $s \in \mathbb{R}$ is a scale factor. Afterwards, the predicted segmentation mask is used to iteratively refine the vertex positions through silhouette-based differential rendering. As a last step, we project the input image onto a texture map and apply basic filtering to improve the visual quality. To obtain a final visual output the textured mesh is rendered with the help of generic rendering software.

4.2. Architecture

In this section, we briefly describe the overall topology of our proposed architecture and provide additional and more detailed insights into the constituting architectural components throughout the following subsections.

The overall network architecture with its components is illustrated in Figure 4.2. The starting point is the encoding stage that directly works on the inputs and serves several purposes. At the beginning, each separate encoding head extracts meaningful feature maps from the respective input. Afterwards, the feature maps get fused into a meaningful common representation, from which further features are extracted by the tail encoder. The obtained compressed representation gets passed to the segmentation decoder for predicting a segmentation mask as well as a fully connected regressor to retrieve MANO parameters. The output of the regressor is fed into a subsequent differentiable MANO layer outputting a mesh in form of vertices from the given parameters. Combined with the predicted camera parameters, a differential renderer is used to generate a silhouette image with gradients. The predicted segmentation from the segmentation decoder is used as an optimization target for the output of the differential renderer. With the help of the gradients we calculated vertex offsets that reduce the error between the two hand silhouettes. Afterwards, the refined hand mesh acts as projection target to update the texture map with the current appearance from the input image.

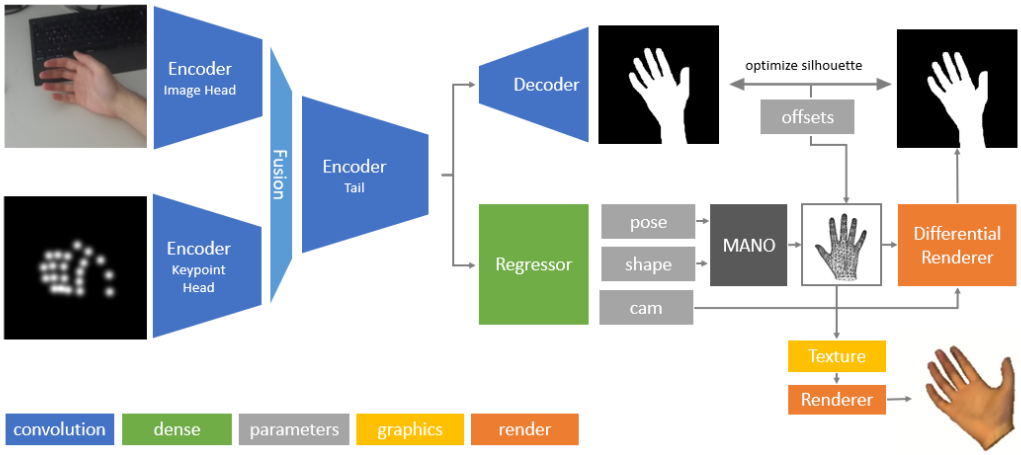


Figure 4.2.: Overview of the architecture for a monocular camera setting.

4.2.1. Encoders

Encoders are the foundation of fully convolutional neural network architectures and crucial to the success of the entire method. Every encoder architecture offers its own characteristics and is almost always accompanied by various trade-offs. We required an architecture with high representational ability as well as conformity with the computational budget and critical inference time requirements for our targeted application. Therefore, our encoder is based on the popular ResNet50 architecture proposed in [39]. The key idea behind residual networks is the introduction of shortcut connections, which improve the performance of deep nets and also mitigate vanishing gradients. For further details we refer the reader to [39].

As we not only have an RGB image as input, but also encoded keypoints in the form of heatmaps, we extended the ResNet architecture, so that it can receive two different inputs and learn an implicit modality fusion as shown in Figure 4.3. We achieved this by dividing the network at the third ResNet layer into two parts where we name the front section head and the back section tail. Afterwards, we duplicated the head section for both inputs and adapted them to the corresponding input channel sizes. To fuse the modalities in a sensible way, we added a fusion block that is based on the self-supervised model adaption (SSMA) block introduced in [40]. The idea behind the SSMA block is to adaptively recalibrate and fuse encoded feature maps according to spatial location and scene context. In other words, it learns to explicitly model the correlation between two feature maps, to selectively emphasize more informative features from a respective modality while suppressing the others. The fused modalities get fed into the tail of the ResNet50 architecture to obtain the base features, of which the decoding components of our method make use.

More specifically, as visualized in Figure 4.3, the "identity block" is the base building block that contains a shortcut for activations to directly propagate to deeper layers. Furthermore, it is based on a bottleneck design to reduce the number of parameters in order to save computation time. The "conv block" is composed in a very similar manner but yields a reduction in dimensionality and therefore requires a convolution in the shortcut path.

Similarly, the structure of the fusion block consists also solely of convolutional and merge

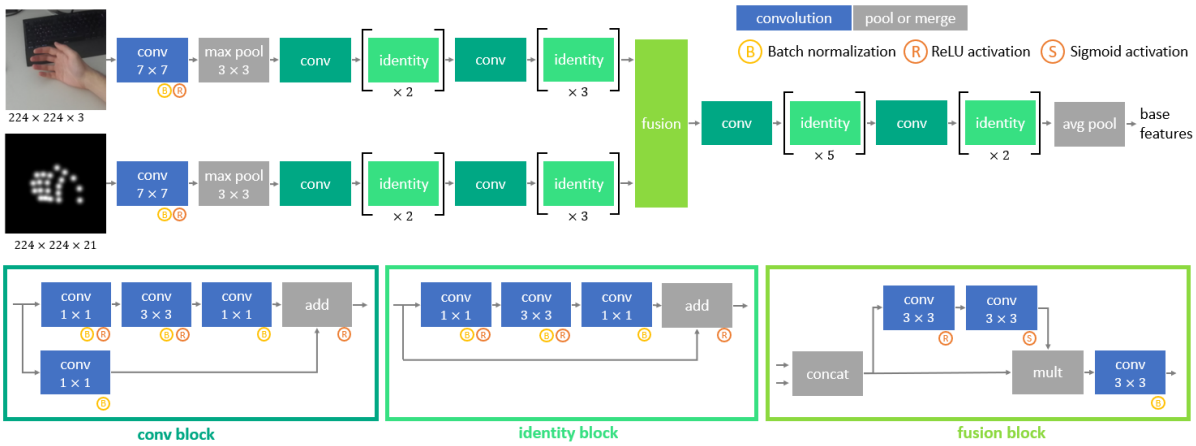


Figure 4.3.: Encoder Architecture. *Proposed ResNet50 based encoder with self-supervised model adaption block for modality fusion.*

operations. First, the feature maps of each modality $\boldsymbol{X}^{img} \in \mathbb{R}^{C\times H\times W}$ and $\boldsymbol{X}^{kp} \in \mathbb{R}^{C\times H\times W}$ get concatenated to $\boldsymbol{X}^{img|kp} \in \mathbb{R}^{2C\times H\times W}$. For our chosen fusion location inside the ResNet encoder, the dimensions correspond to $C = 512, H = 28$ and $W = 28$. Before the real fusion happens, a recalibration technique is applied in order to adapt the concatenated features to extract meaningful knowledge. For this $\boldsymbol{X}^{img|kp}$ is passed through a bottleneck, where the first ReLU activated convolution reduces the channel dimensionality by a ratio of 8 to $2C/8 = 128$. Subsequently, the dimensionality of the features is increased back to their original size by the second convolutional layer. This time, a sigmoid activation is applied to scale the values to the $[0, 1]$ range, so that they represent normalized weights. The resulting output w is used to emphasize and de-emphasize different regions in the originally concatenated features $\boldsymbol{X}^{img|kp}$. This is achieved by taking the Hadamard product, i.e. doing element wise multiplication of the obtained weights and the concatenated features, which corresponds to $\hat{\boldsymbol{X}}^{img|kp} = w \circ \boldsymbol{X}^{img|kp}$. As a last step, the adaptive recalibrated feature maps $\hat{\boldsymbol{X}}^{img|kp}$ are passed through a final convolution layer with batch normalization in order to reduce the channel depth and yield the fused output with the correct dimensionality, so that it can be further processed by the tail of the encoder.

4.2.2. Segmentation Decoder

Image segmentation is the process of attributing every pixel in an image to a certain category in order to create a simplified representation that is more meaningful to analyze. In the case of hand segmentation, each pixel of the input hand crop is assigned either 'hand' or 'background'. To obtain this segmentation mask we rely upon work from [41] which itself builds upon [42]. In this paper, the authors introduce so called "fully convolutional networks" with the idea to supplement a usual contracting encoding network by successive expanding convolutional layers. In these, the dimension reduction operations are replaced by upsampling operators, hence, the resolution of the output is increased. They also propose to add lateral connections that combine high resolution features from the contracting path with upsampled feature maps on the same level. This helps with localization and allows the successive convolution layer to assemble a

more precise output. In [41] the authors modified the previously mentioned architecture to have a larger number of channels in the upsampling part which, in turn, results in a more symmetric model architecture inspiring the name U-Net.

Because the U-Net architecture has proven very successful for various segmentation tasks, we adapted a modified architecture for our segmentation decoder, which is visualized in Figure 4.4. Instead of using a separate encoder for the segmentation, we reuse the already existing encoder described in Section 4.2.1. Because the input image contains more expressive segmentation features compared to the keypoint heatmap, the first skip connections utilize the image encoding branch. Nevertheless, the decoder can benefit from the fused hand keypoint modality through deeper level features of the encoder.

More specifically, in the "up block" of the architecture, we utilize transposed convolutions as upsampling operation, so that the dimensionality matches the size of the feature maps which get propagated over the skip connections. After concatenation, a series of two convolutions with batch normalization and ReLU activation is applied.

As a final step after the last "up block", the channels are reduced through a convolution to size one, as we only have one segmentation class. Furthermore, a sigmoid activation is applied in order obtain probabilities of a pixel belonging to a hand.

To drastically reduce the amount of parameters, we removed the convolutional bottleneck layers between the contracting and expanding part from the original U-Net architecture without significant impact on the performance.

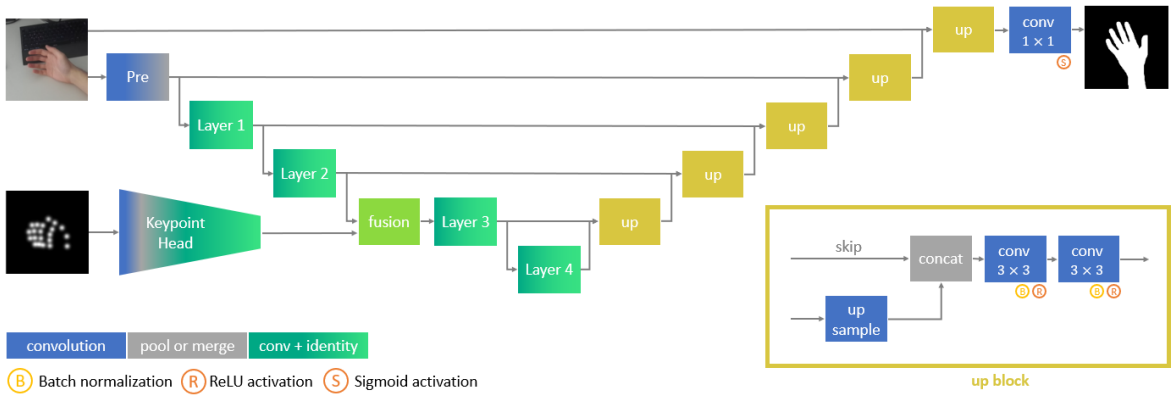


Figure 4.4.: Segmentation decoder architecture. *Proposed segmentation decoder architecture based on U-Net [41] that utilizes the encoder architecture introduced in Section 4.2.1.*

4.2.3. MANO Regressor

Besides the segmentation decoder also the MANO regressor makes use of the base features we obtain with our proposed encoder from Section 4.2.1. The goal of this architectural component is to predict the pose and shape parameters for the MANO model and also output camera parameters, so that obtained mesh prediction can be projected onto the input image.

The structure of the regressor is straightforward as visualized in Figure 4.5 and consists -

in contrast to all the other parts - of dense layers only. The obtained base features $\boldsymbol{X}^{base} \in \mathbb{R}^{2048\times1\times1}$ are flattened and processed by two fully connected layers with respective sizes 2048 and 512. This yields features of size 512 from which we regress our $pose \in \mathbb{R}^{48}$ and $shape \in \mathbb{R}^{10}$, as well as the weak perspective camera parameters $cam = (trans, scale) \in \mathbb{R}^3$.

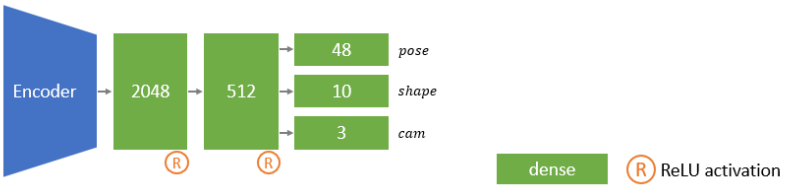


Figure 4.5.: MANO regressor architecture. *Proposed MANO regressor component of our architecture consisting of dense layers.*

4.2.4. Fine Grained Optimization

Research has shown that one-shot MANO parameter prediction methods like ours, although fast and robust, often come with mediocre image-mesh alignment. With the goal of mitigating this problem, we propose a test time optimization strategy. The idea behind the fine grained optimization section in our architecture is to iteratively refine the vertex positions to better align with the input image by adding an offset $\Delta\boldsymbol{H} \in \mathbb{R}^{778\times3}$ to the hand mesh $\boldsymbol{H} \in \mathbb{R}^{778\times3}$. The optimized mesh at iteration step $t+1$ therefore corresponds to:

$$\boldsymbol{H}(t+1) = \boldsymbol{H}(t) + \Delta\boldsymbol{H}(t).$$

The initial hand mesh $\boldsymbol{H}(0)$ is obtained from the MANO model by providing pose and shape parameters, i.e. $\boldsymbol{H}(0) = \text{MANO}(pose, shape)$.

The offset $\Delta\boldsymbol{H}(t)$ is obtained from the gradients provided by a differential renderer aiming to reduce the discrepancy between the predicted segmentation and the rendered silhouette of $\boldsymbol{H}(t)$. In other words, we learn offsets for each vertex, such that the predicted mesh silhouette S_{mesh} is more similar to the target silhouette image S_{target} from the previously introduced segmentation decoder at each optimization step.

In the subsequent paragraphs, we will describe in more detail how the offsets are calculated based on the differential rendering approach.

Differential Renderer

Given a 3D geometry and other scene properties, such as illumination, classical graphic renderers can produce realistic images. Differential renderers attempt to solve the inverse problem, which is to infer 3D geometry from image data. In other words, they try to model the relationship between parameter changes and image observations. A differential renderer takes

vertices and color as input to produce pixels in an image, similar to a classical renderer. However, it also retains derivatives so that it can determine which inputs contributed to the final pixel colors. Classical renderers include a fundamental discretization step, called rasterization, which assigns triangles to pixels. Due to discontinuities, this prevents the rendering process to be differential and made early differential renderers only approximate the gradients. In [43], a truly differentiable rendering framework was proposed which views rendering as an aggregation function that fuses probabilistic contributions of all mesh triangles with respect to the rendered pixels. This novel formulation allows to propagate gradients also to occluded and far-range vertices, which was not possible with previous methods.

For our implementation we used PyTorch3D [44], which is a library of reusable components for deep learning with 3D data and includes a differential renderer based on [43]. As we are only interested in obtaining a segmentation mask, we can make use of the soft silhouette shader that calculates the silhouette by blending the faces for each pixel, based on the 2D euclidean distance of the center of the pixel to the mesh face. We further initialized the renderer with the appropriate rasterization settings to be consistent with SoftRas [43].

As optimizer we used stochastic gradient descent with the learning rate set to $\eta = 0.002$ and momentum $\alpha = 0.9$, which results in the offset at step t being calculated as follows:

$$\Delta \boldsymbol{H}(t) = \eta \nabla \mathcal{L}(\boldsymbol{H}(t)) + \alpha \Delta \boldsymbol{H}(t-1)$$

where η: step size (learning rate)

α: decay factor

Loss

The loss function $\mathcal{L}$ in (4.2.4) corresponds to a weighted sum of the image silhouette loss and regularization terms that include mesh normal consistency, laplacian mesh smoothing and mesh edge loss as denoted in (4.2.4). The hyperparameters which balance the loss are $\lambda_1 = 1$, $\lambda_2 = 1$ $\lambda_3 = 1$ and $\lambda_4 = 0.1$.

$$\mathcal{L} = \underbrace{\lambda_1 \mathcal{L}_{\text{silhouette}}}_{\text{image loss}} + \underbrace{\lambda_2 \mathcal{L}_{\text{normal}} + \lambda_3 \mathcal{L}_{\text{laplacian}} + \lambda_4 \mathcal{L}_{\text{edge}}}_{\text{smooth / regularize loss}}$$

The image silhouette loss is computed as the squared L2 distance between the predicted silhouette and the target. Furthermore, all of the regularization losses are essential to generate a quality appealing mesh model. More specifically, the mesh normal loss favors smooth surfaces, the edge loss encourages uniform distribution of the mesh vertices for high recall and the laplacian loss prevents mesh faces from intersecting each other. They are defined in the following equations, where v corresponds to a vertex and $\mathcal{N}(v)$ describes the edge connected neighboring vertices of v. Further n_0 and n_1 describe normals of neighboring faces.

$$\mathcal{L}_{\text{silhouette}} = (S_{\text{mesh}} - S_{\text{target}})^2$$
$$\mathcal{L}_{\text{edge}} = \sum_{v} \sum_{k \in \mathcal{N}(v)} \|v - k\|_2^2$$
$$\mathcal{L}_{laplacian} = \sum_{v} \|\delta_v\|_2^2 \text{ , where } \delta_v = v - \sum_{k \in \mathcal{N}(v)} \frac{1}{\|\mathcal{N}(v)\|} k$$
$$\mathcal{L}_{normal} = 1 - cos(n_0, n_1) \text{ , where } cos(n_0, n_1) = \frac{n_0 \cdot n_1}{||n_0||\,||n_1||}$$

4.2.5. Texturing

As we aim for a pixel perfect fit, our texturing method can rely on using the projection of the camera image onto the 3D mesh to obtain a texture map. To improve the quality and robustness against outliers, we apply some some basic filtering on top.

As a simple straight forward implementation in Python is too slow for real time use, we rely heavily on functions provided by the PyTorch3D rendering pipeline, which includes various optimized native implementations of useful functions. As a first step, the MANO hand mesh is unwrapped into a UV map to obtain a static UV mapping that provides UV map coordinates for every vertex. During test time, we utilize the PyTorch3D rasterizer to obtain fragments, from which the face attributes can be interpolated with the help of barycentric coordinates to yield UV map coordinates for pixels of the input image. We collect the corresponding RGB values from the input hand crop and update the texture map based on following strategy:

$$\gamma = 1 - |(\Pi_{new} - \Phi\Pi_{old})|$$
$$T = T + \Phi(\gamma\Pi_{new}) - \Phi(\gamma T)$$

where $\Phi \in \{0,1\}^{s \times s}$: mask , i.e. $\Phi_{i,j} = \begin{cases} 1 \text{ if } T_{new_{i,j}} \neq 0 \\ 0 \text{ else} \end{cases}$

$\Pi \in \mathbb{R}^{s \times s \times 3}$: projection of input image to UV map

$T \in \mathbb{R}^{s \times s \times 3}$: texture map

s : texture size

4.3. Stereo Extension

The estimation of a 3D hand pose from a single RGB image is an ill-posed problem due to depth and scale ambiguities. Additionally, the hand pose space is rather large, due to the many degrees of freedom. Self-occlusion, motion blur and diversity including jewelry, skin colors or tattoos contribute to the difficulty of the problem. While stereo has the power to overcome some

of the ambiguities, the lack of corresponding stereo training data requires a more sophisticated network architecture that allows the use of monocular datasets like FreiHAND to first learn the correlations between poses and real hand image data. Afterwards, limited stereo data such as our proposed synthetic hand dataset can be utilized to learn a sensible stereo fusion.

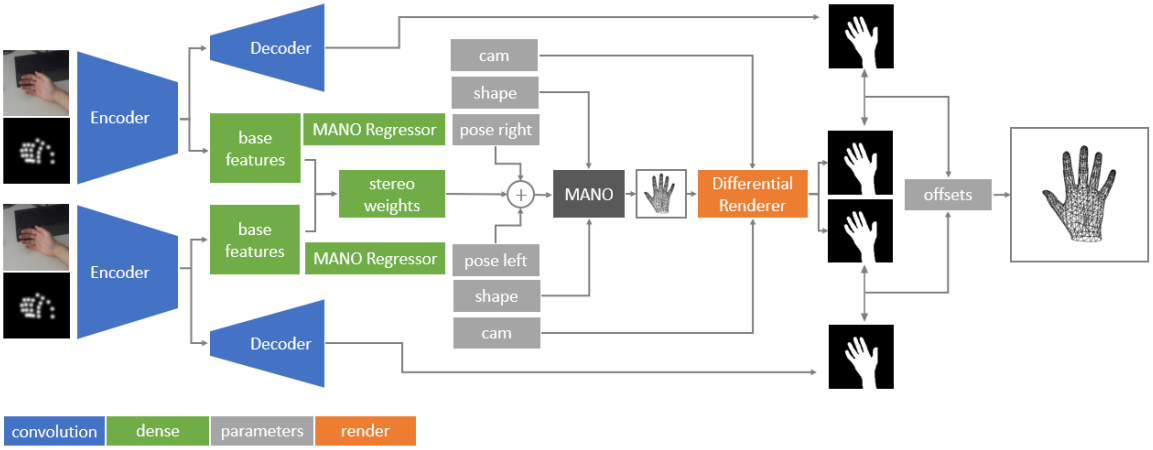


Figure 4.6.: Proposed stereo extension of our architecture. *We duplicate the monocular architecture and fuse the obtained predictions in a meaningful way. The second viewpoint is also helpful for the differential rendering based fine grained optimization.*

To take advantage of the stereo vision that most VR headsets provide, we extended our model to a stereo setting. We achieved this by duplicating most of the previously introduced monocular architecture, as illustrated in Figure 4.6. Besides regressing the pose p, shape s and camera parameters c for each view from the base features $\boldsymbol{X} \in \mathbb{R}^{512}$ of the MANO regressor, we also concatenate them from both views to a new stereo feature of size $\boldsymbol{X}_{stereo} \in \mathbb{R}^{1024} = \boldsymbol{X}_{right} | \boldsymbol{X}_{left}$. From this, we regress additional stereo weights $w \in \mathbb{R}^{48}$ with the help of a fully connected layer. The obtained weights are used to combine the predicted poses from the left and right view in a meaningful way. More specifically, we calculate the stereo fused right hand pose p_{right} as

$$p_{\text{right}} = w\, p_{\text{right}} + (1 - w)\, p_{\text{left}}.$$

In other words, we ideally want the network to be able to differentiate between visible and occluded joints of different views, so that they can be merged in a meaningful way. For example, if a joint is self-occluded by the palm in the right view, the exact location is not recoverable and we have to rely on predicting likely distributions of possible locations. However, if the specific joint is visible in the left view, the network now has the power to compensate for the lack of information by utilizing the left view joint predictions over the right view predictions.

The predicted shape parameters from both views are fused by averaging, the same way we refine the left and right camera parameters while taking their geometric constraints into account. With ξ transforming from left to right view, we obtain

$$s_{\text{stereo}} = \frac{s_{\text{right}} + s_{\text{left}}}{2}$$
$$c_{\text{right}} = \frac{c_{\text{right}} + \xi(c_{\text{left}})}{2}.$$

Moreover, the fine grained test time optimization introduced in Section 4.2.4 can additionally profit from the stereo setting. Instead of optimizing the mesh offsets with only one view, we can now compute the silhouette image loss as the average over the two views, i.e. for SL being the left silhouette image and SR being the right silhouette image. Our new loss term is:

$$\mathcal{L}_{\text{silhouette}} = \frac{(SL_{\text{mesh}} - SL_{\text{target}})^2 + (SR_{\text{mesh}} - SR_{\text{target}})^2}{2}$$

4.4. Datasets

In this section we briefly review the FreiHAND dataset and introduce our own synthetic stereo dataset.

4.4.1. FreiHAND Dataset

The FreiHAND [45] dataset consists of 32,560 unique RGB images of right hands with the corresponding MANO annotation. Additionally, intrinsic camera parameters, 2D joint locations and the scale of a specific reference bone are provided. All but the reference bone were used in our project. Furthermore, the creators of the dataset used a harmonization method [46] based on a deep network and the deep image colorization approach from [47] to augment the dataset to a total of 130,240 samples. There also exists an online competition based on a validation dataset containing 3960 samples, for which no groundtruth data was released.

What distinguishes the FreiHAND dataset from others is that the annotations include not only pose but also shape parameters for the MANO model. This makes it the first dataset which allows to also learn hand appearances directly from RGB image data. The hands in the dataset are captured from 32 people who were told to perform various actions in front of the camera setup. Additionally, the dataset contains samples that include hands that grasp objects, which range from workshop tools like drills, wrenches and screwdrivers to kitchen supplies.

4.4.2. Synthetic Stereo Hands Dataset

To overcome the lack of adequate training data for a stereo setting with focus on egocentric viewpoints, we generated a large-scale synthetic stereo image dataset of hands. The primary use was to learn the proposed stereo fusion, but the pixel-accurate segmentation masks were also utilized for training the segmentation decoder.

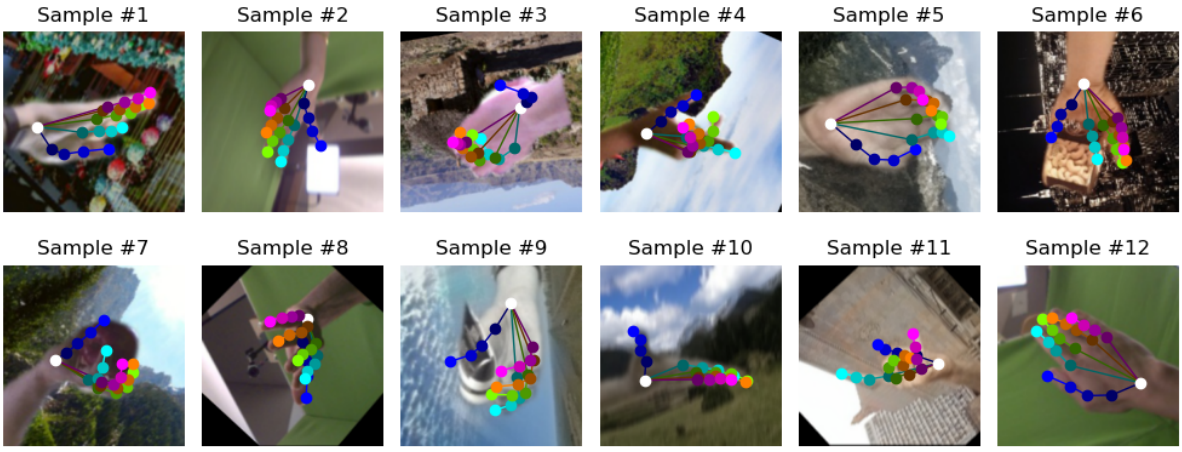


Figure 4.7.: Visualization of the FreiHand dataset with 2D keypoint annotations *Samples of the augmented FreiHAND dataset with groundtruth 2D keypoint annotations.*

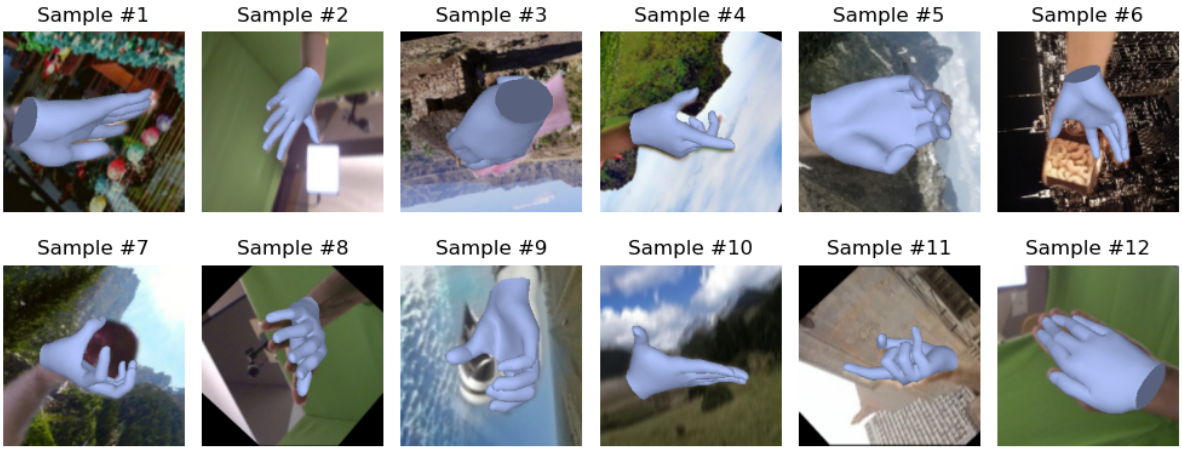


Figure 4.8.: Visualization of the FreiHand dataset with mesh annotations. *Samples of the augmented FreiHAND dataset with groundtruth mesh annotations.*

To render realistic images of embodied hands, we modified publicly available code for the generation of the synthetic ObMan [34] to our needs. The original ObMan dataset is concerned with modeling hands grasping objects so that hand-object interactions can be studied. Our additions include the introduction of a stereo camera resembling the properties of the ZED stereo camera[1] that was used for testing. This includes replicating the inter-ocular distance as well as sensor and lens properties. While ObMan uses a grasp database to obtain poses, we rely on random MANO parameters to cover a more general pose space. In order to only receive likely hand articulations, we sample randomly from the PCA space with six components. The shape parameters are also uniformly selected. For realism and so that boundary regions between hands and arms can be learned for segmentation, we render the full body. To achieve this, the random hand pose is converted to the SMPL+H [29] model which integrates MANO to the statistical body model SMPL [26]. Similar to ObMan, the body poses and shapes are varied

[1] `https://www.stereolabs.com/zed/`

by sampling poses from the CCMU MOCAP database [48] and shapes from CAESAR [49]. The body textures are obtained from the full body scans of SURREAL [50]. For egocentric viewpoints, the back of the hand is more often visible than the palm of the hand. To cope with this, we rotate the global orientation of the body, so that our camera captures relevant views. Furthermore, we translate the hand root joint to align with the camera's optical center. The images are rendered using Blender [51], with the background being sampled from the living room category of the LSUN dataset [52]. We output a left and right RGB image, a left and right segmentation binary mask as well as annotations that include hand vertices, 2D joints, 3D joints along with shape, pose and camera parameters.

We rendered a total of 15082 samples, corresponding to 30164 RGB images, annotations and segmentation masks. In Figure 4.9 the full rectified image from the left and right camera is visualized and in Figure 4.10 hand crops with the corresponding annotations are shown.

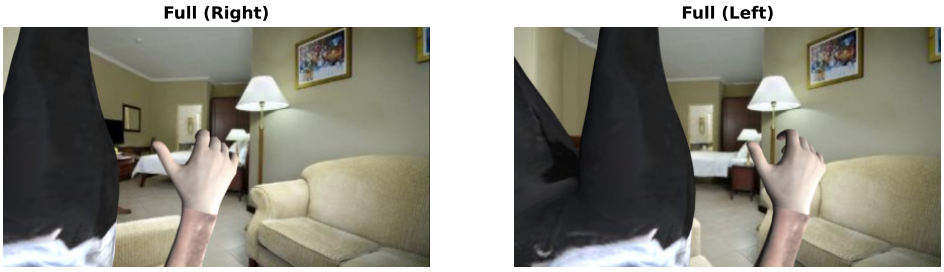


Figure 4.9.: Stereo image from our synthetic dataset. *An example of the rendered rectified images from our synthetic dataset, which models the camera properties of the ZED stereo camera.*

Figure 4.10.: Annotations of our synthetic dataset. *Extracted hand crops with groundtruth annotations consisting of 2D keypoints, segmentation mask and hand mesh vertices.*

4.5. Training Framework

We employ a multi-stage training procedure for our proposed architecture where we freeze different parts of the model and also selectively remove components from the information flow. Figure 4.11 visualizes a simplified representation of our architecture, which together with Table 4.1 and Table 4.2 helps to understand our training framework.

In addition, the datasets mentioned in Section 4.4 and compositions thereof are used to train the network's various stages. For the monocular model there exist five training stages as denoted in Table 4.1. For the 'Image', 'Keypoint', 'Fusion' and 'Finalize' stage, the FreiHAND dataset with data augmentation is used. For the 'Segmentation' stage, both our synthetic stereo hands dataset and FreiHAND are used in combination. The reason for this is that while the FreiHAND dataset captures real hands, the segmentation masks lack accuracy compared to the pixel-level accurate groundtruths from the synthetic data. By jointly training on both, we mitigate their respective drawbacks.

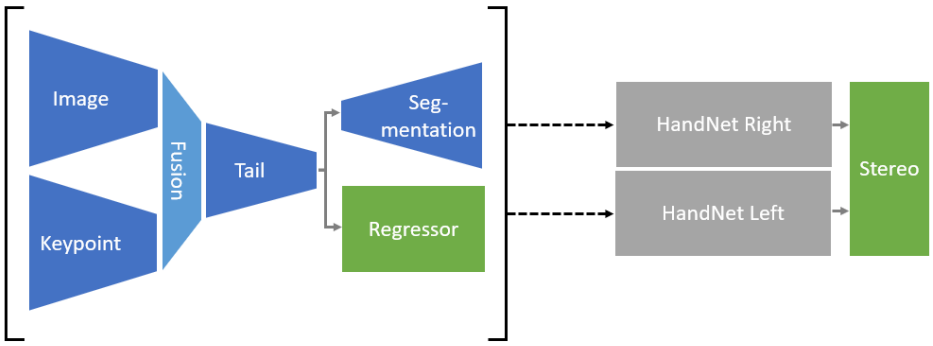


Figure 4.11.: Trainable components. *Simplified representation of all trainable architectural components.*

Stage	Information Flow	Trained Components
Image	Image → Tail → Regressor	Image, Tail, Regressor
Keypoint	Keypoint → Tail → Regressor	Keypoint
Fusion	(Image, Keypoint) → Fusion → Tail → Regressor	Fusion
Finalize	(Image, Keypoint) → Fusion → Tail → Regressor	Tail, Regressor
Segmentation	(Image, Keypoint) → Fusion → Tail → Segmentation	Segmentation

Table 4.1.: *Training stages for monocular setting.*

For the stereo setting, the previously trained 'HandNet' gets duplicated, as described in Section 4.3 to learn meaningful stereo fusion weights. Ideally, the network learns to distinguish visible and occluded joints from different views, so that the respective predictions from each view get merged in a meaningful way. The training is carried out with our synthetic stereo hand dataset which was mainly introduced for the purpose of learning this stereo fusion. As there

Stage	Information Flow	Trained Components
Synthetic adaption	(Image, Keypoint) → Fusion → Tail → Regressor	Image
Stereo	(HandNet Right, HandNet Left) → Stereo	Stereo

Table 4.2.: *Training stages for stereo setting. The adaption to synthetic data is only done to better learn the stereo fusion. During test time, the original monocular network is used.*

is a rather large domain gap, the trained HandNet has limited success in predicting poses for synthetic images, which makes it also hard to learn a sensible fusion. We propose to introduce an additional synthetic adaption training stage, in which only the image encoder is retrained for a small number of epochs to adapt towards the synthetic image data. Afterwards, we continue with training the 'Stereo' weights regressor, which concludes training the two stages for the stereo setting, as denoted in Table 4.2. It is important to note that during test time on a real stereo camera stream, the original non-adapted HandNet is used.

4.5.1. Losses

The previously described training stages use different loss functions, as specified in Table 4.3. In the following paragraphs, the respective loss functions are defined and discussed.

Stage	Mano Loss	Segmentation Loss	Stereo Loss	FreiHAND	Synthetic
Image	✗			✗	
Keypoint	✗			✗	
Fusion	✗			✗	
Finalize	✗			✗	
Segmentation		✗		✗	✗
Synthetic adaption			✗	✗	✗
Stereo			✗		✗

Table 4.3.: *Overview on what loss and dataset is used for each training stage.*

Mano Loss

The loss used to train the network which regressed the MANO parameters consists of a weighted sum that is composed as follows:

$$\mathcal{L}_{\text{mano}} = \underbrace{\lambda_1 \mathcal{L}_{\text{pose}} + \lambda_2 \mathcal{L}_{\text{shape}} + \lambda_3 \mathcal{L}_{\text{joints2D}}}_{\text{mano losses}} + \underbrace{\lambda_4 \mathcal{L}_{\text{shape\_reg}} + \lambda_5 \mathcal{L}_{\text{cam\_reg}}}_{\text{regularization}}$$

where $\mathcal{L}_{\text{pose}}$: MSE of pose parameters

$\mathcal{L}_{\text{shape}}$: MSE of shape parameters

$\mathcal{L}_{\text{joints2D}}$: MSE of 2D joints

$\mathcal{L}_{\text{shape\_reg}}$: MSE of shape with $[0, ..., 0] \in \mathbb{R}^{10}$

$\mathcal{L}_{\text{cam\_reg}}$: MSE of scale camera parameter

$MSE(\hat{y}_i, y_i) = \frac{1}{B} \sum_{i \in [0,B)} (\hat{y}_i - y_i)^2$ with batch size B

While $\mathcal{L}_{\text{pose}}$ and $\mathcal{L}_{\text{shape}}$ help to infer MANO parameters, $\mathcal{L}_{\text{joints2D}}$ is required to learn the weak camera parameters for the root joint, so that the mesh can be projected back onto the image. As the mean shape is encoded by the zero vector, we introduce an additional regularization term $\mathcal{L}_{\text{shape\_reg}}$ to protect against shape blow ups. Furthermore, as we rely on projection of the 3D joints to obtain the 2D joints, we were required to add a regularization term $\mathcal{L}_{\text{cam\_reg}}$ on the scale parameter as to not learn scale values that place the root joint behind the camera. During training, we used the following hyperparameters to balance the loss: $\lambda_1 = 10$, $\lambda_2 = 1$ $\lambda_3 = 100$, $\lambda_4 = 0.5$ and $\lambda_5 = 10^{-5}$.

Segmentation Loss

For training the segmentation decoder, we rely on the binary cross entropy.

$$\mathcal{L}_{\text{segmentation}} = BCE(\hat{y}, y) = -(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y}))$$

Stereo Loss

Similar to the mano loss, the stereo loss is based on a weighted sum. However, this time we directly supervise the vertices and the 3D joint locations.

$$\mathcal{L}_{\text{stereo}} = \lambda_1 \mathcal{L}_{\text{vertex}} + \lambda_2 \mathcal{L}_{\text{joints3D}}$$

where $\mathcal{L}_{\text{vertex}}$: MSE of vertices

$\mathcal{L}_{\text{joints3D}}$: MSE of 3D

$MSE(\hat{y}_i, y_i) = \frac{1}{B} \sum_{i \in [0,B)} (\hat{y}_i - y_i)^2$ with batch size B

4.5.2. Data Augmentation

We apply a variety of data augmentation techniques during training, more specifically:

- Rescale: The input image and the accompanying annotations get scaled correspondingly by a provided factor. In our experiments this was used to obtain the correct input sizes for the encoders.
- Rotate: The RGB image and the accompanying annotations get randomly rotated. For our experiments, we randomly selected one of 20 pre-computed linear spaced rotation matrices.
- Crop: A tight hand bounding box is calculated from the 2D joint annotation and scaled by a randomly selected factor. For our experiments, the scale factor was in the range $[1.5, 2.5]$.
- Blur: The RGB image is blurred with a given probability with a randomly selected blur kernel. In our experiments, we used a probability of 0.5 and pre-computed four blur kernels, with the kernel size being linearly spaced between $[3, 12]$.

Correctly augmenting the annotations is not always trivial by itself. For example, cropping leads to a change in the principal point which impacts the perspective projection of groundtruth mesh vertices. To align them with the cropped image again, the hand mesh must be rotated appropriately to cope with the transformation, as visualized in Figure 4.12.

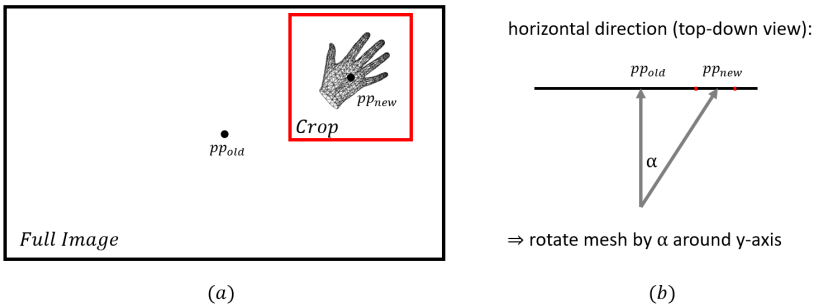


Figure 4.12.: Geometry of hand crop data augmentation. *When a hand crop is extracted (a), this also introduces a new principal point pp_{new}, which leads to changes in the perspective projection. To compensate for this, the groundtruth mesh vertices need to be rotated by some degree α, as illustrated in (b) for the horizontal direction. Similar for the vertical direction.*

4.6. Inference Pipeline

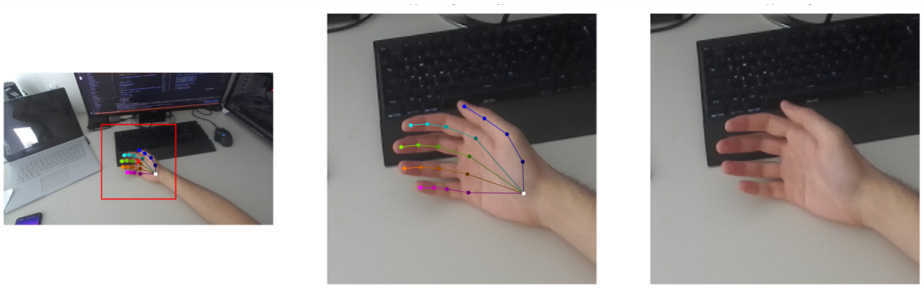
In the following sections we describe the inference pipeline in more detail. It consists of pre-processing, executing our method and rendering the obtained results.

4.6.1. Pre-processing

Pre-processing consists of detecting the hand in a video frame and extracting the appropriate inputs for our method.

Hand Detection

As hand crops are required for subsequent regression, we explored of the shelf hand detection solutions, namely 100DOH [7], OpenPose [31, 32], GANeratedHands [22] and MediaPipe Hands [8]. While the first offers superior accuracy, it does not offset the poor runtime performance of only several frames compared to MediaPipe Hands, which is lightweight and runs at 60+ FPS. Additionally, the latter provides 2D keypoint annotations, whereby we are not required to do an implicit heatmap regression similar to [33]. Therefore, we rely on it for detecting the hands in the input frames.



(a) (b) (c)

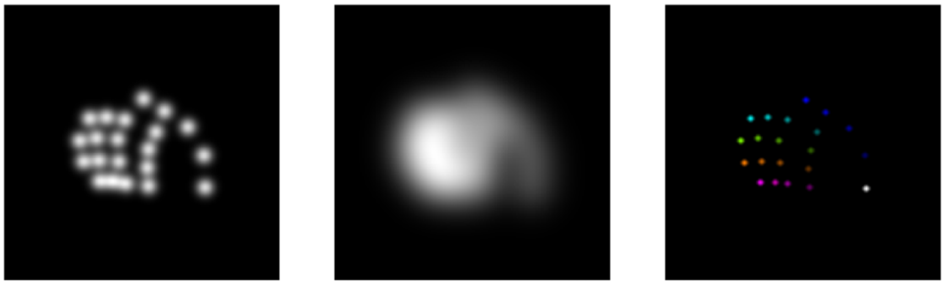
Figure 4.13.: Extraction of hand crops. *The bounding box (a) is calculated based on the 2D keypoints (b) to obtain the final hand crop (c).*

Extracting inputs

We use the obtained 2D keypoint annotations to calculate a tight bounding box which then gets scaled by a factor of 2. Afterwards, the obtained hand crop I_c is scaled to a spatial size of 224×224 and normalized to the range $I_c \in [-1, 1]$. Furthermore, the retrieved 2D keypoints are converted into a heatmap with the same spatial dimensions as the image and 21 channels. Each channel corresponds to a hand joint and, in order to provide informative gradients, we encode the locations with a 2D gaussian distribution H_k:

$$H_k(x, y) = A \exp\left(-\left(\frac{(x - x_c)^2}{2\sigma^2} + \frac{(y - y_c)^2}{2\sigma^2}\right)\right), \{x, y\} \in \Omega.$$

In the equation k corresponds to a joint and Ω is the set of pixels in I. For our experiments we used $\sigma = 5$ for a kernel size of 49 and $\sigma = 15$ for a kernel size of 127. A is a scale factor, that is chosen such that the sum over the whole kernel equals one. Similar to the RGB image, we normalize the heatmap afterwards to the range $H \in [-1, 1]$.



(a) (b) (c)

Figure 4.14.: Keypoint heatmaps. *Visualization of the 21 channel heatmaps with kernel size 49 (a) and 127 (b) for the keypoint input modality. The color-coded joints are visible in (c).*

4.6.2. Realistic Hands

The obtained RGB hand crop and the heatmap serve as input for our method. For the stereo setting, we have an input image and heatmap for each view. The output of our method consists of a textured mesh with weak camera parameters that specify the location of the hand root joint. Any standard renderer can be used to render the textured hand mesh. For visualization purposes, the obtained camera parameters can be used to overlay the hand mesh. In a real VR setting additional viewpoint correction can be incorporated on top, in order to cope with the camera-eye misalignment.

5

Results

In this chapter, we first provide details about the computational environment for the experiments and introduce the used evaluation metrics. Afterwards, we evaluate and discuss the results of the different architectural components. Furthermore, we demonstrate that our method outperforms other monocular 3D hand reconstruction and segmentation methods. Additionally, we provide many qualitative results to give detailed insights into our approach.

5.1. Computing Environment

Our implementation uses Python 3.7.9 and PyTorch 1.8.0, along with other common libraries such as PyTorch3D, Numpy and ZED SDK. The code of this project can be found on GitHub[1]. The training was conducted on the Leonhard cluster provided by the ETH Zurich using an Nvidia RTX 2080Ti with 11GB memory alongside 8 cores and 64 GB of RAM. The experiments were carried out on a personal workstation with an AMD Ryzen 9 5900x, 64 GB of RAM and an Nvidia GTX TITAN X with 12GB memory.

5.2. Metrics

The performance for the 3D reconstruction is evaluated using multiple metrics. One metric is the mean per vertex position error (MPVE), which measures the Euclidean distance in millimeters between the estimated and groundtruth vertex coordinates. Similarly, the mean per keypoint position error (MPJE) measures the Euclidean distance between the estimated and groundtruth joint coordinates, which correspond to our 21 defined hand keypoints. Following the 3D pose

[1] `https://github.com/michaelseeber/realistic-hands`

estimation literature, we also report the area under curve (AUC) of the percentage of correct keypoints (PCK) by computing the percentage of predicted vertices/joints lying within a spherical threshold ρ around the target vertex/joint position p, i.e.

$$PCK\left(\boldsymbol{x}, \hat{\boldsymbol{x}}, \rho\right) = \frac{1}{K} \sum_{k} \mathbb{I}\left[\left\|\boldsymbol{p}_k - \hat{\boldsymbol{p}}_k\right\|_2 \leq \rho\right]$$

where K is the amount of vertices/keypoints and k corresponds to a specific index. The function $\mathbb{I}[\cdot]$ returns 1 if its input is true and 0 otherwise.

We also report F-scores which - given a distance threshold - define the harmonic mean between recall and precision between two sets of points [53].

For evaluation of the segmentation performance, we use two metrics, namely pixel accuracy and mean intersection over union. The first corresponds to the percentage of correctly classified pixels. The second, which is also known as Jaccard Index, is defined as the area of overlap between predicted segmentation and groundtruth, divided by the area of the union of both. The mean IoU is calculated by taking the mean of both classes "hand" and "background". For TP = true positives, FN = false negatives and FP = false positives this yields:

$$IoU(\boldsymbol{x}, \hat{\boldsymbol{x}}) = \frac{TP}{TP + FN + FP}$$

5.3. MANO Regression

In this section, we report and discuss experiments related to the HandNet component of our method which regressed MANO parameters from monocular RGB images. The HandNet was trained for a total of 500 epochs that were distributed between the different training stages as listed in Table 5.1. Moreover, the table lists the amount of epochs used at the end of each stage for linearly decaying the learning rate towards 0. The Adam optimizer [54] was used with an initial learning rate of 0.0002 at each stage and $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

Stage	number of epochs	epochs used for decay
Image	120	20
Keypoint	100	20
Fusion	100	20
Finalize	180	100

Table 5.1.: *Number of epochs for each training stage and amount of epochs used at end of each stage for linear learning rate decay.*

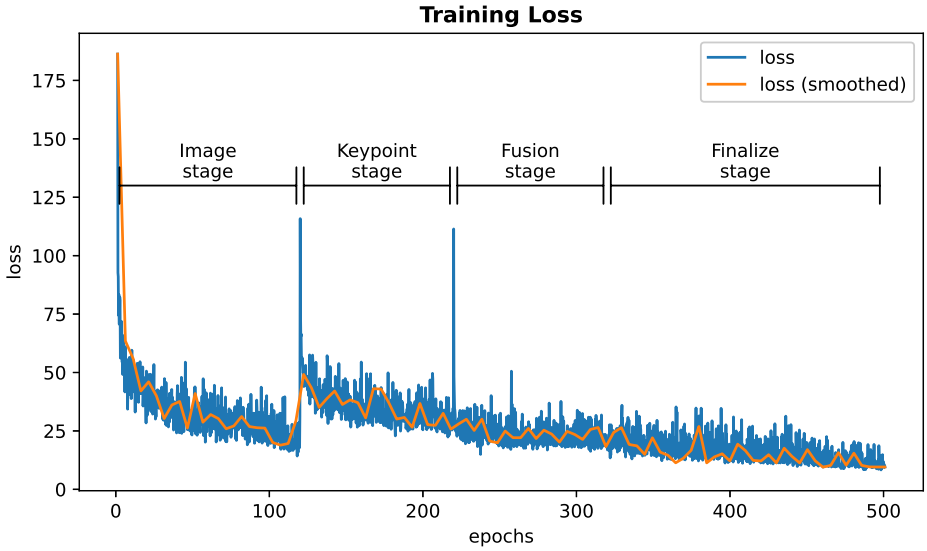


Figure 5.1.: *Training loss of the HandNet component.*

5.3.1. Comparison

We compared the monocular HandNet component to other related, state of the art work using the FreiHAND test set, for which no groundtruth was published and evaluation is carried out through an online challenge [2]. As some methods only predict relative coordinates, Zimmerman et al. [45] provide an "aligned" metric, which performs a 3D alignment of the predicted vertices using Procrustes analysis (PA). This also copes with unfair comparison between methods that utilize the reference bone length, which is provided at test time and defined as the phalangal proximal bone of the middle finger (i.e. distance between keypoints 9 and 10).

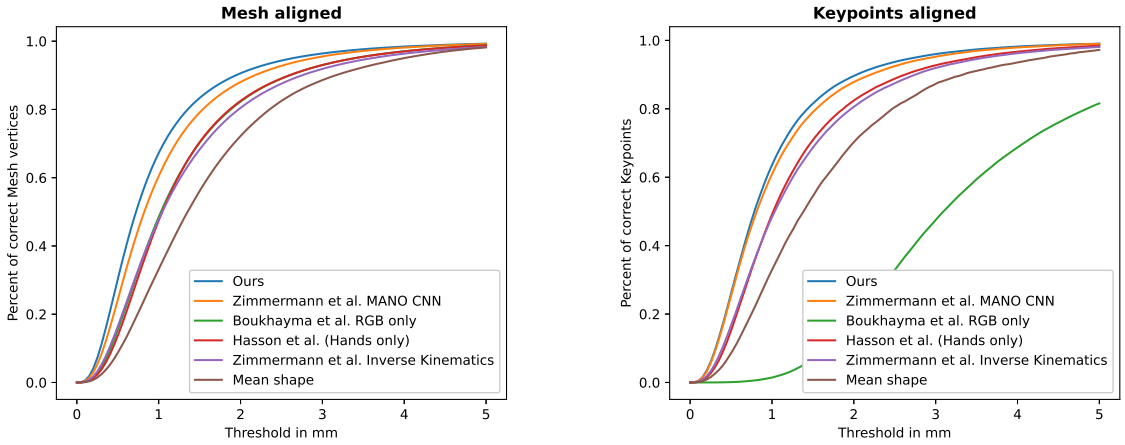
Methods	MPVE (PA)	F @ 5mm (PA)	F@15mm (PA)
Mean shape	1.64	0.336	0.837
Inverse Kinematics [45]	1.37	0.439	0.892
Hasson et al. [34]	1.33	0.429	0.907
Boukhayma et al. [30]	1.32	0.427	0.894
Mano CNN [45]	1.09	0.516	0.934
Ours	**0.97**	**0.568**	**0.946**

Table 5.2.: *Comparison of our approach with other methods on the task of monocular hand pose and shape estimation. Our method outperforms the others in all three evaluation metrics, where small values for MPVE (PA) and large values for the F-scores are better.*

[2] `https://competitions.codalab.org/competitions/21238`

Table 5.2 shows, that our approach outperforms baselines, such as the mean shape and inverse kinematic fits of the MANO model by quite a margin. Moreover, our method ranks higher than other learning based approaches, such as [34, 30, 45]. It is important to acknowledge that we rely on an extremely lightweight hand detector for real-time performance reasons with MediaPipe Hands [8] as opposed to other methods utilizing more sophisticated hand detectors, which could further boost the performance.

The PCK curve based on vertices is visible in Figure 5.2(a) and the PCK curve based on the 21 joint keypoints is shown in Figure 5.2(b)



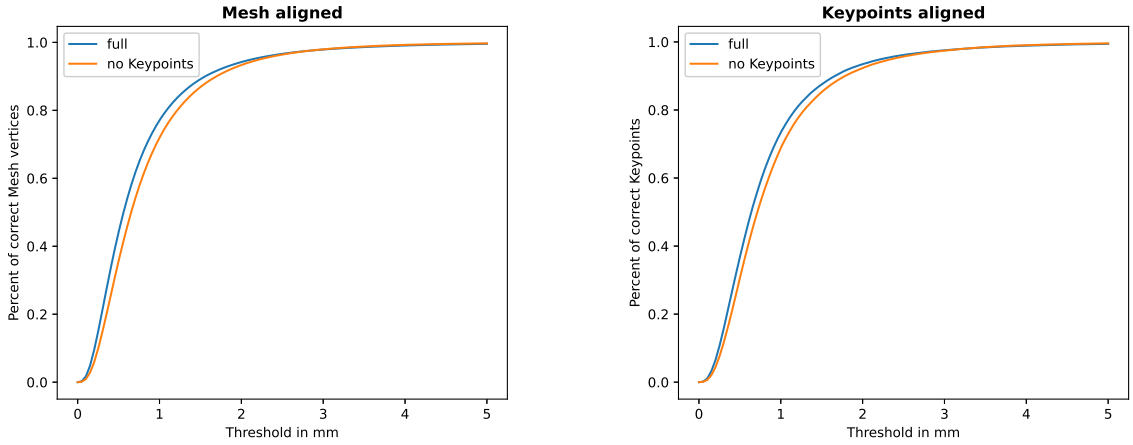
(a) PCK plot based on vertex positions using PA. *(b) PCK plot based on 3D joint position using PA.*

Figure 5.2.: *PCK plots comparing various state of the art methods and baselines on the test set of FreiHAND.*

5.3.2. Modality Fusion

For the following experiments on the modality fusion, we use a local validation set that was created by randomly splitting 30% of the FreiHand samples into a validation dataset and the remaining 70% are used for training the HandNet component. As the FreiHAND dataset comes with only 32560 unique samples that get quadrupled through different color-hallucination-techniques, we made sure to base the training and test splits on unique samples and include the augmented versions in the respective sets.

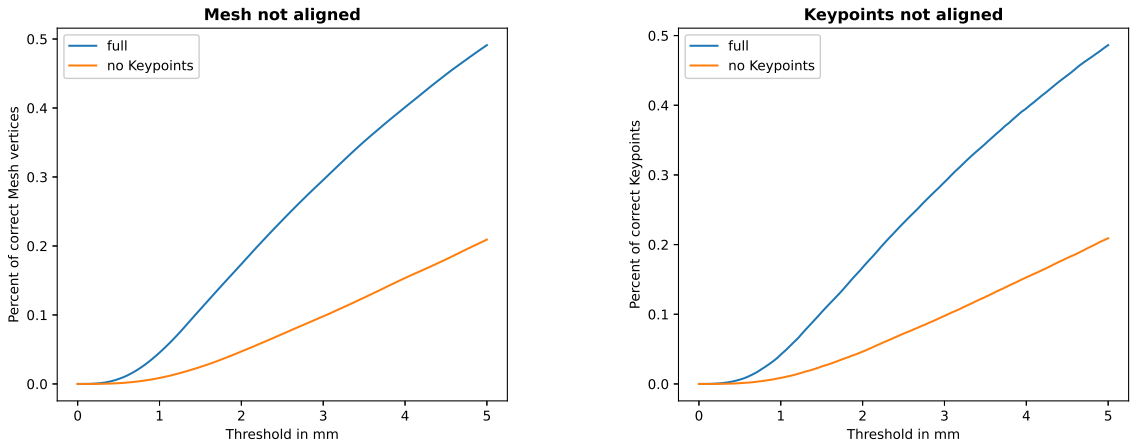
To explore the impact of the additional keypoint heatmap modality we replaced the keypoint input with a zero tensor. It should be noted that during training we feed a zero tensor as heatmap with a probability of 30%, in order to make our method more robust and resilient during test time, in case the 2D keypoint detector is failing to provide any predictions. The similar results in Figure 5.3(a) and Figure 5.3(b) show that our SSMA network component successfully learned to suppress the keypoint modality when no meaningful information is provided by this input branch while using PA.



(a) PCK plot based on vertex positions. *(b) PCK plot based on 3D joint position.*

Figure 5.3.: *PCK plots on the validation set with and without keypoint heatmaps provided using PA.*

In contrast, when we inspect the non-aligned results in Figure 5.4(a) and Figure 5.4(b), it becomes apparent that the keypoint heatmap provides valuable and meaningful information for localizing the exact hand position and alignment. This proofs that the modality fusion is indeed helpful for our method, especially because good initial alignments are crucial for the fine grained optimization to succeed.



(a) PCK plot based on vertex positions without PA. *(b) PCK plot based on 3D joint positions without PA.*

Figure 5.4.: *PCK plots on the validation set with and without keypoint heatmaps provided.*

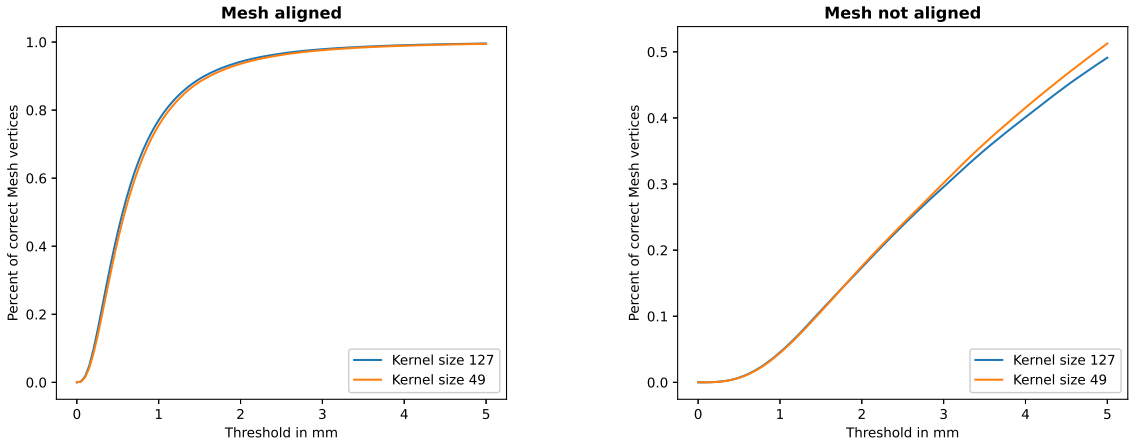
5.3.3. Keypoint Encoding

Furthermore, we explored different heatmap encodings of the 2D keypoints, namely various kernel sizes as visualized in previous section's Figure 4.14. Table 5.3 as well as Figure 5.5(a) and Figure 5.5(b) show that our method performs very similarly for both kernel sizes $k = 49$

and $k = 127$, with only a slight edge for $k = 127$. This is consistent with qualitative real world tests.

Kernel size	MVE (A)	F @ 5mm (A)	F @ 15mm (A)	MKE	F @ 5mm	F@15mm
K = 49	0.81	0.656	0.964	6.60	0.123	0.408
K = 127	0.77	0.673	0.967	7.33	0.119	0.393

Table 5.3.: *Results of different heatmap kernel sizes on the FreiHAND validation set.*



(a) PCK plot based on vertex positions with PA. *(b) PCK plot based on 3D joint positions without PA.*

Figure 5.5.: *PCK plots based on vertex positions comparing different kernel sizes on the validation set.*

5.3.4. Qualitative results

Qualitative results on the test dataset, for which no groundtruth information is available are visible in Figure 5.6. The two last examples in the third column show failure cases. In the first failure case, the position is wrongly predicted, due to the hand being cut off. In the second example, the scale is wrongly predicted, due to severe occlusion. Noteworthy is that the hand pose itself resembles a very likely pose, also in these failure cases, which explains the discrepancies in accuracy when using PA.

5.4. Segmentation

In this section, we report and discuss experiments related to the segmentation decoder of our proposed method, which provides the segmentation masks that are used as optimization targets by the differential renderer. The segmentation decoder was trained for 10 epochs, with linearly

Figure 5.6.: Qualitative results on the FreiHAND test set. *Visualization of predictions on the FreiHAND test set, where no groundtruth is available. The last two examples in the third column show failure cases, due to cutoff and occlusion, however the overall hand pose still correctly predicted.*

decaying the learning rate during the last 6 epochs. Furthermore, the Adam optimizer [54] was used with an initial learning rate of 0.0002 and $\beta_1 = 0.9$ and $\beta_2 = 0.99$.

For the segmentation experiments both FreiHAND and our synthetic dataset were used. More specifically, we use 70% of the data from each dataset for training and 30% for validation. Due to the chosen size of the synthetic dataset, the corresponding amounts of samples from the two datasets are also balanced as listed in Table 5.4.

Splits	Total	FreiHAND	Synth
Training	43911	22795	21116
Validation	18813	9765	9048

Table 5.4.: *Number of samples in the training and validation set for the segmentation experiments.*

We compared the segmentation performance to a trivial color thresholding baseline (HSV) and a recently introduced method [55] for egocentric hand segmentation (CNN and UMA). For the color thresholding baseline, we converted the image to the HSV color space and applied color thresholding based on sensible values for skin color [56]. Cai et al. [55] proposed in their work a bayesian CNN that is based on RefineNet [57]. As segmentations tend to not generalize well to other domains, they further introduced an iterative self training method, that adapts the pretrained model to a new domain. The process is self-supervised and based on the uncertainty estimates of the bayesian CNN. We retrained the bayesian CNN using our training dataset and for the UMA baseline we further ran the uncertainty-guided model adaption on the validation dataset. The performance of the CNN and the UMA is almost similar, which stems from the fact that training and validation dataset are from the same data distribution and the domain adaption is, therefore, fundamentally limited. The results are recorded in Table 5.5.

	Combined		FreiHAND		Synth	
	Mean IoU	Accuracy	Mean IoU	Accuracy	Mean IoU	Accuracy
HSV	0.26	0.81	0.33	0.90	0.18	0.72
CNN [55]	0.85	0.98	0.77	0.97	0.93	0.99
UMA [55]	0.85	0.98	0.77	0.97	0.93	0.99
Ours	**0.88**	**0.99**	**0.80**	**0.98**	**0.95**	**0.99**

Table 5.5.: *Segmentation Results*

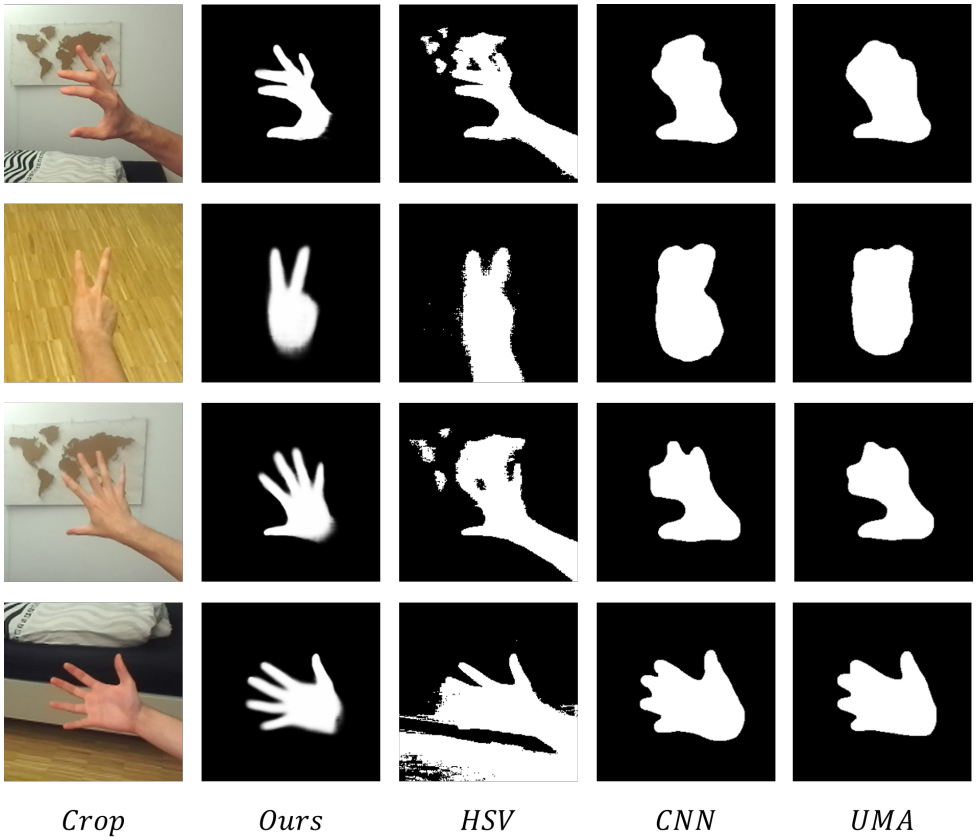


Figure 5.7.: Qualitative segmentation comparison. *Visualization of obtained segmentation masks by different methods. HSV corresponds to thresholding on skin color in the HSV color space. CNN corresponds to the bayesian CNN that serves as starting point for the uncertainty-guided model adaptation (UMA) introduced by Cai et al. [55].*

We further evaluated the segmentation quality on captured real data. In contrast to before, the uncertainty-guided model adaption is based on the obtained video frames. Figure 5.7 visualizes input image crops and the respective predicted segmentation masks from each method. Contrasting with the color thresholding, all data driven methods correctly learned to differentiate between hand and arm. Furthermore, they are not impacted by similarly colored objects like the world map or floor. Our method stands out when it comes to segmenting fingers, as both the

CNN and UMA method fail to segment them in detail.

5.5. Fine grained optimization

Results from our iterative fine grained optimization are visualized in Figure 5.8. More specifically, it shows the target silhouette from the segmentation decoder as well as the rendered silhouette of the predicted mesh. Furthermore, to better understand the impact of the vertex offsets, we encoded them as color-coded heatmaps, ranging from blue (smallest offset) to red (largest offset).

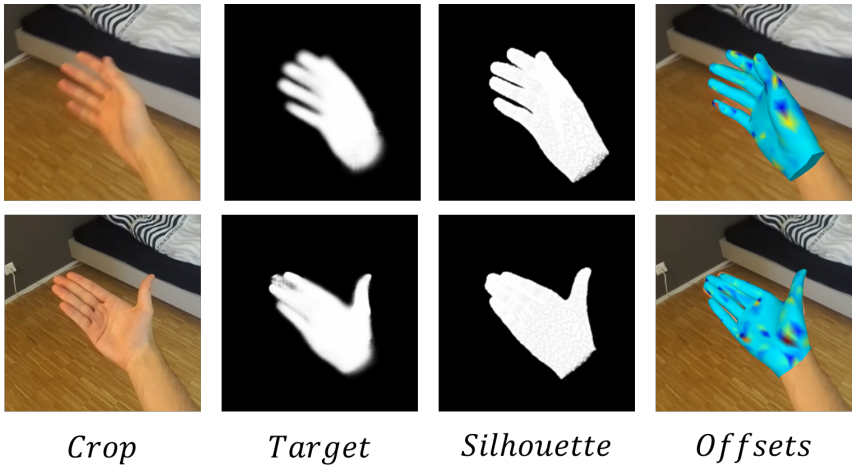


Figure 5.8.: Offsets obtained from fine grained optimization. *Visualization of obtained offsets by our iterative fine grained optimization based on differential rendering. Light blue corresponds to a very small vertex offset, while dark red indicates a larger one.*

5.6. Stereo extension

In this section, we report and discuss experiments related to the stereo extension of our proposed method. The training consists of 2 epochs of synthetic adaption and 8 epochs of learning the stereo weights, where we linearly decay the learning rate over the last 4 epochs as denoted in Table 5.6. The Adam optimizer [54] was used with an initial learning rate of 0.0002 and $\beta_1 = 0.9$ and $\beta_2 = 0.99$. For evaluation we use the original HandNet that was not adapted to the synthetic data.

Stage	number of epochs	epochs used for decay
Synthetic adaption	2	0
Keypoint	8	4

Table 5.6.: *Number of epochs for each stage of training the stereo extension.*

To better understand the stereo fusion, we provide qualitative results in Figure 5.9. Moreover, the fusion weights w are color-coded into a heatmap, where each joint corresponds to a row and small weights are represented by blue, while large values are indicated by red.

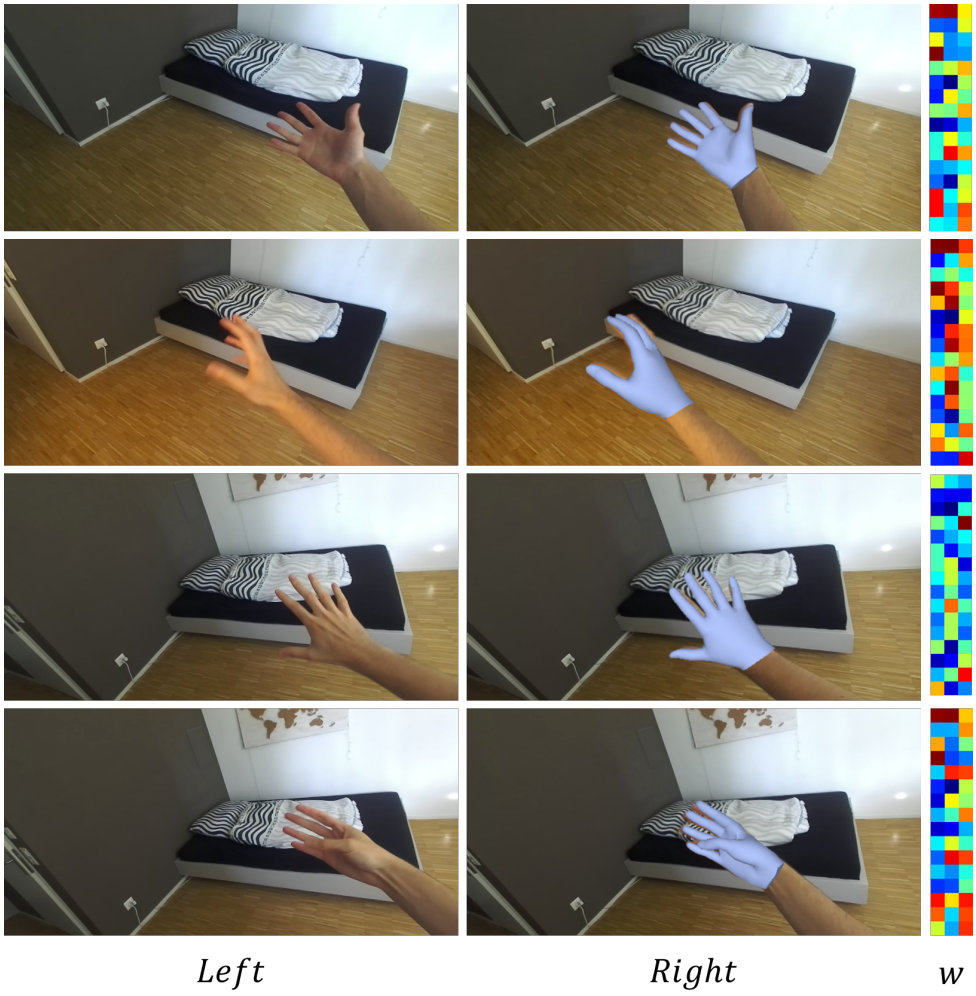


Figure 5.9.: Qualitative results of the stereo fusion. *Depicted are the left and right input images, as well as the fused prediction overlaid in the right view. Furthermore, the fusion weights are visualized as heatmaps, where each row corresponds to a joint. Light blue corresponds to small weights, whereas dark red represents large weights.*

5.7. Texture

To assess the performance of our proposed texture projection, we provide qualitative examples of the obtained texture maps as well as the rendered output hand mesh in Figure 5.10. There tends to be some noise around the borders, however, these parts are self occluded most of the time and therefore also not visible in the rendered output mesh.

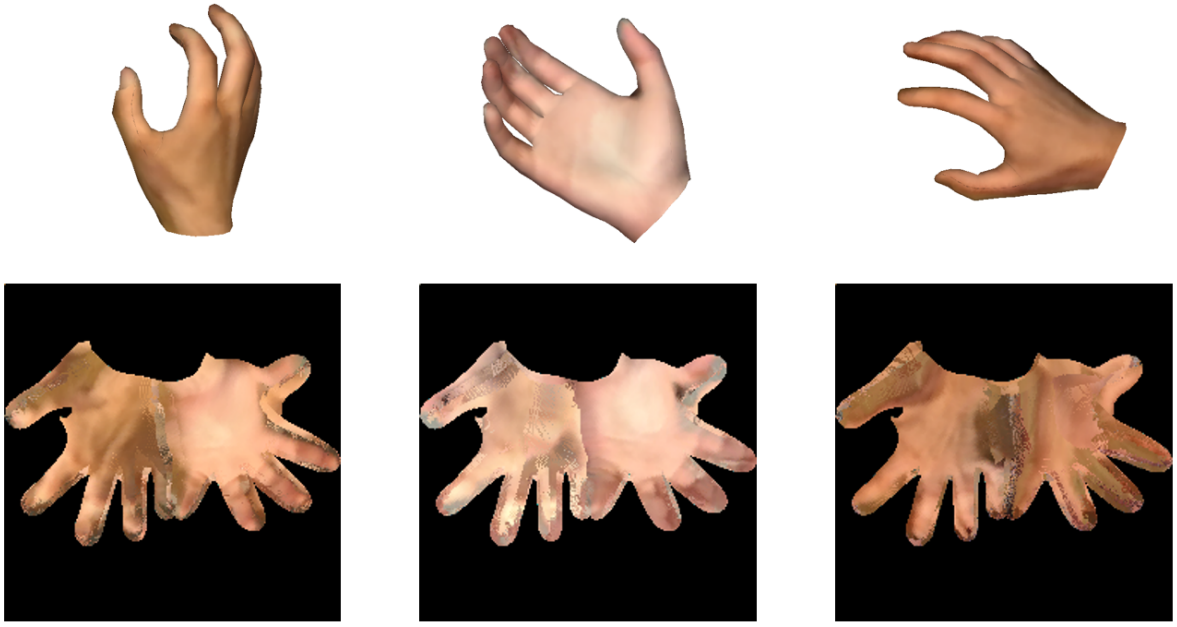


Figure 5.10.: Texture mapping results. *Visualization of obtained texture maps and the rendered output hand mesh.*

Additionally, we show that our projection based approach allows special skin characteristics such as tattoos to be transferred to the digital hand replication in Figure 5.11.

Figure 5.11.: Texture map with skin characteristic. *The captured hand has a smiley face drawn on the palm, which also gets replicated in the texture map and therefore transferred to the digital hand replication.*

5.8. Realistic Hands

To give a summarized overview of our method, we provide qualitative results of the important architectural components of our proposed method in Figure 5.12.

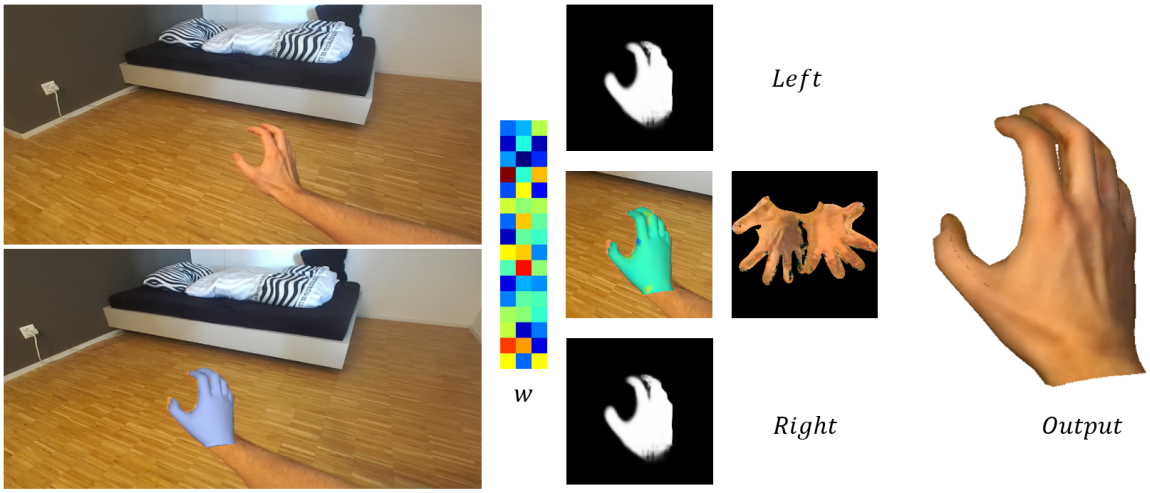


Figure 5.12.: Full result visualization of our method Realistic Hands. *The figure contains both input images, the stereo fusion weights, the predicted segmentation masks, the heatmap encoded offsets from the optimization as well as the texture map. Additionally, the predicted mesh is overlaid on the right input image and the obtained textured mesh is rendered as output.*

6

Conclusion and Future Work

6.1. Conclusion

In this thesis, we proposed and implemented a real-time capable method that is able to predict MANO parameters to obtain 3D hand reconstruction from monocular RGB images by fusing different input modalities. Moreover, we elaborated on how to extend the architecture towards a stereo setting, which is prevalent in VR headsets and helps to further improve the resilience and accuracy of our method. Unfortunately, such one-shot predictions of poses yield only mediocre mesh-image alignments when projected onto the original image. We proposed to mitigate this problem by introducing a test time optimization strategy based on differential rendering. The idea behind this more fine grained optimization step is to iteratively refine the vertex positions based on a jointly predicted segmentation mask, so that we obtain a truly personalized hand mesh that aligns accurately with the input image. This further enabled the generation of a matching realistic texture which comprises personal skin characteristics to yield a textured output mesh that can replace generic hand models of current VR systems to increase the user's immersion by providing a less artificial feel.

The experiments showed that our approach outperforms other state of the art methods in monocular RGB hand pose and shape estimation. Moreover, the integrated hand segmentation network exhibits state of the art performance, which enabled qualitative improvements through the proposed fine grained test time optimization. Furthermore, the accurate mesh-image alignments allowed for texturing, which yields visually pleasing personalized hand meshes as a result. Additionally, we demonstrated the usefulness of the stereo extension for increased robustness. Due to low computational requirements, our method allows for real-time deployment.

To summarize, our hybrid approach is more expressive and personalized than work relying on the MANO hand model and also more efficient and robust than methods that directly try to infer a mesh. Additionally, we are one of the first to explore texturing based on camera input, yielding a truly unique and personalized hand mesh for use in VR.

6.2. Future Work

There are several directions of research that could be explored further. On the one hand, the monocular and stereo hand mesh estimation could be improved. This includes the creation of new and better real datasets, as they appear to be a limiting factor especially for the challenging egocentric perspective. Also, methodological improvements on the deep neural network architecture could lead to better results. On the other hand, the realistic appearance component could be studied further and in more detail. Due to the fine grained optimization, we were able to project the texture map, but generative approaches for the texture map creation could also be studied and lead to fruitful results. Further, the decomposition of texture maps in intrinsic components to allow for scene adaptive relighting could be explored.

A

Appendix

A.1. Architecture

Layer		Output Size	Input Channels	Output Channels
ImgHead	Down1	[112,112]	3	64
	Down2	[56,56]	64	256
	Down3	[28,28]	256	512
KpHead	Down1	[112,112]	3	64
	Down2	[56,56]	64	256
	Down3	[28,28]	256	512
Fusion	Conv2d	[28,28]	1024	128
	Conv2d	[28,28]	128	1024
	Conv2d	[28,28]	1024	512
Tail	Down4	[14,14]	512	1024
	Down5	[7,7]	1024	2048
Segment	Up1	[14,14]	2048	1024
	Up2	[28,28]	1024	512
	Up3	[56,56]	512	256
	Up4	[112,112]	256	128
	Up5	[224,224]	128	64
	Conv2d	[224,224]	64	1

Table A.1.: *Output size and input/output channels of our HandNet architecture.*

A.2. Stereo results

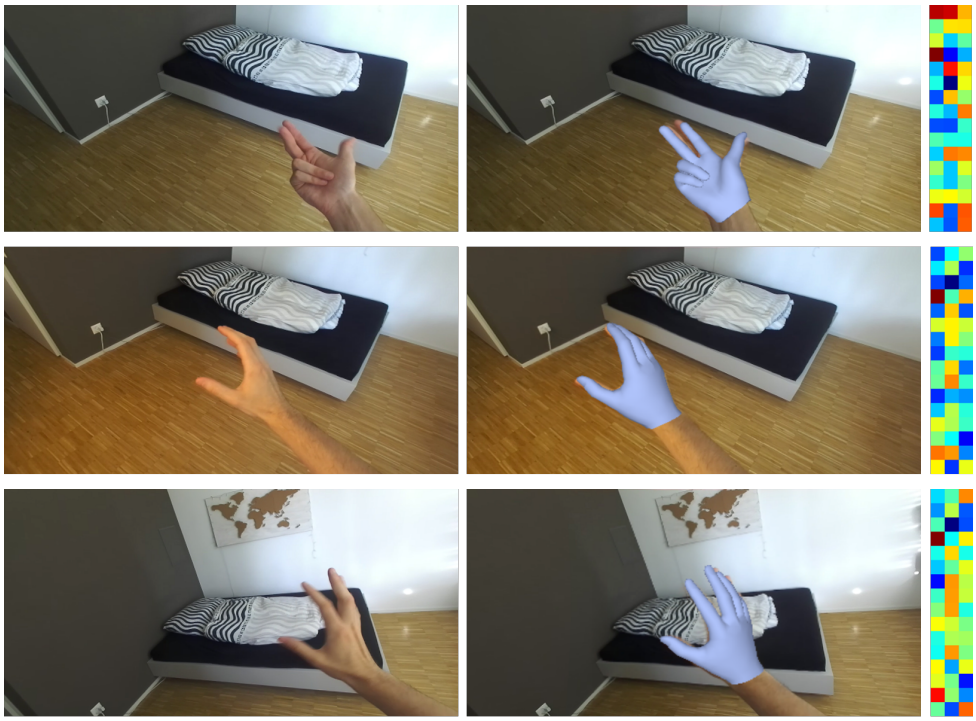


Figure A.1.: More qualitative results of the stereo fusion. *Depicted are the left and right input images, as well as the fused prediction overlaid in the right view. Furthermore, the fusion weights are visualized as heatmaps, where each row corresponds to a joint. Light blue corresponds to small weights, whereas dark red represents large weights.*

A.3. OpenCV camera to PyTorch3D

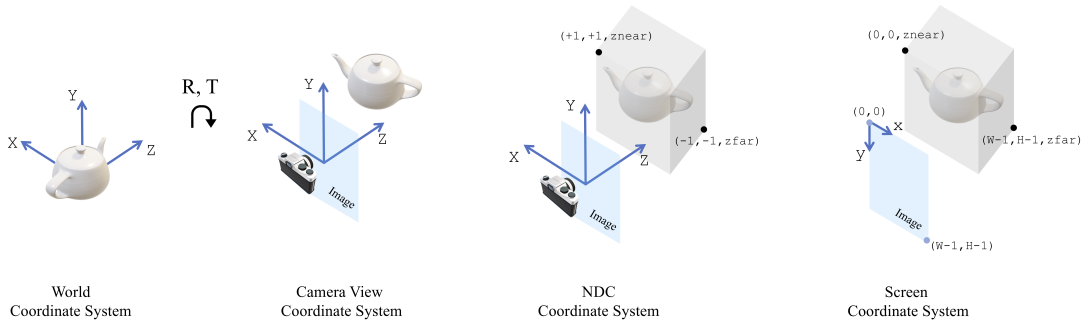


Figure A.2.: PyTorch3D camera system.

OpenCV coordinate system is X-right, Y-down, Z-out, while PyTorch3D uses X-left, Y-up, Z-out. Instead of flipping X and Y axes we input negative focal length to the PyTorch3D camera which is equal:

$$
\begin{aligned}
x_{ndc} &= (fx * 2/W) * X/Z - (px - W/2) * 2/W \\
y_{ndc} &= (fy * 2/H) * Y/Z - (py - H/2) * 2/H \\
x_{screen} &= (W - 1)/2 * (1 - x_ndc) \\
y_{screen} &= (H - 1)/2 * (1 - y_ndc)
\end{aligned}
$$

Substituting x_{ndc} and y_{ndc}:

$$
\begin{aligned}
x_{screen} &= (-fx * (W - 1)/W) * X/Z + (W - 1)/W * px \\
y_{screen} &= (-fy * (H - 1)/H) * Y/Z + (H - 1)/H * py
\end{aligned}
$$

Bibliography

[1] N. Qian, J. Wang, F. Mueller, F. Bernard, V. Golyanik, and C. Theobalt, "Html: A parametric hand texture model for 3d hand reconstruction and personalization," in *European Conference on Computer Vision*, pp. 54–71, Springer, 2020.

[2] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, A. Kipman, *et al.*, "Efficient human pose estimation from single depth images," *IEEE transactions on pattern analysis and machine intelligence*, vol. 35, no. 12, pp. 2821–2840, 2012.

[3] Y. Wu and Q. Ji, "Facial landmark detection: A literature survey," *International Journal of Computer Vision*, vol. 127, no. 2, pp. 115–142, 2019.

[4] C. Tomasi, S. Petrov, and A. Sastry, "3d tracking= classification+ interpolation.," in *ICCV*, vol. 3, p. 1441, 2003.

[5] A. Mittal, A. Zisserman, and P. H. Torr, "Hand detection using multiple proposals.," in *Bmvc*, vol. 2, p. 5, 2011.

[6] S. Bambach, S. Lee, D. J. Crandall, and C. Yu, "Lending a hand: Detecting hands and recognizing activities in complex egocentric interactions," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1949–1957, 2015.

[7] D. Shan, J. Geng, M. Shu, and D. F. Fouhey, "Understanding human hands in contact at internet scale," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9869–9878, 2020.

[8] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[9] V. Athitsos and S. Sclaroff, "Estimating 3d hand pose from a cluttered image," in *2003*

IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings., vol. 2, pp. II–432, IEEE, 2003.

[10] G. Borgefors, "Hierarchical chamfer matching: A parametric edge matching algorithm," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 10, no. 6, pp. 849–865, 1988.

[11] I. Oikonomidis, N. Kyriazis, and A. A. Argyros, "Efficient model-based 3d tracking of hand articulations using kinect.," in *BmVC*, vol. 1, p. 3, 2011.

[12] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, *et al.*, "Accurate, robust, and flexible real-time hand tracking," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 3633–3642, 2015.

[13] A. Tagliasacchi, M. Schröder, A. Tkach, S. Bouaziz, M. Botsch, and M. Pauly, "Robust articulated-icp for real-time hand tracking," in *Computer Graphics Forum*, vol. 34, pp. 101–114, Wiley Online Library, 2015.

[14] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang, "3d hand pose tracking and estimation using stereo matching," *arXiv preprint arXiv:1610.07214*, 2016.

[15] P. Panteleris and A. Argyros, "Back to rgb: 3d tracking of hands and hand-object interactions based on short-baseline stereo," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 575–584, 2017.

[16] M. Oberweger, P. Wohlhart, and V. Lepetit, "Training a feedback loop for hand pose estimation," in *Proceedings of the IEEE international conference on computer vision*, pp. 3316–3324, 2015.

[17] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "Robust 3d hand pose estimation in single depth images: from single-view cnn to multi-view cnns," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3593–3601, 2016.

[18] L. Ge, H. Liang, J. Yuan, and D. Thalmann, "3d convolutional neural networks for efficient and robust hand pose estimation from single depth images," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1991–2000, 2017.

[19] G. Moon, J. Y. Chang, and K. M. Lee, "V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map," in *Proceedings of the IEEE conference on computer vision and pattern Recognition*, pp. 5079–5088, 2018.

[20] L. Ge, Y. Cai, J. Weng, and J. Yuan, "Hand pointnet: 3d hand pose estimation using point sets," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8417–8426, 2018.

[21] C. Zimmermann and T. Brox, "Learning to estimate 3d hand pose from single rgb images," in *Proceedings of the IEEE international conference on computer vision*, pp. 4903–4911, 2017.

[22] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt, "Ganerated hands for real-time 3d hand tracking from monocular rgb," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 49–59, 2018.

[23] L. Yang and A. Yao, "Disentangling latent hands for image synthesis and pose estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9877–9886, 2019.

[24] Y. Cai, L. Ge, J. Cai, and J. Yuan, "Weakly-supervised 3d hand pose estimation from monocular rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 666–682, 2018.

[25] R. A. Güler, N. Neverova, and I. Kokkinos, "Densepose: Dense human pose estimation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7297–7306, 2018.

[26] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, "Smpl: A skinned multi-person linear model," *ACM transactions on graphics (TOG)*, vol. 34, no. 6, pp. 1–16, 2015.

[27] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik, "End-to-end recovery of human shape and pose," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 7122–7131, 2018.

[28] N. Kolotouros, G. Pavlakos, M. J. Black, and K. Daniilidis, "Learning to reconstruct 3d human pose and shape via model-fitting in the loop," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2252–2261, 2019.

[29] J. Romero, D. Tzionas, and M. J. Black, "Embodied hands: Modeling and capturing hands and bodies together," *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, vol. 36, Nov. 2017.

[30] A. Boukhayma, R. d. Bem, and P. H. Torr, "3d hand shape and pose from images in the wild," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10843–10852, 2019.

[31] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh, "Openpose: Realtime multi-person 2d pose estimation using part affinity fields," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2019.

[32] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, "Hand keypoint detection in single images using multiview bootstrapping," in *CVPR*, 2017.

[33] X. Zhang, Q. Li, H. Mo, W. Zhang, and W. Zheng, "End-to-end hand mesh recovery from a monocular rgb image," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2354–2364, 2019.

[34] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid, "Learning joint reconstruction of hands and manipulated objects," in *CVPR*, 2019.

[35] L. Ge, Z. Ren, Y. Li, Z. Xue, Y. Wang, J. Cai, and J. Yuan, "3d hand shape and pose estimation from a single rgb image," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10833–10842, 2019.

[36] G. Moon and K. M. Lee, "I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image," *arXiv preprint arXiv:2008.03713*, 2020.

[37] B. Smith, C. Wu, H. Wen, P. Peluse, Y. Sheikh, J. K. Hodgins, and T. Shiratori, "Constraining dense hand surface tracking with elasticity," *ACM Transactions on Graphics (TOG)*, vol. 39, no. 6, pp. 1–14, 2020.

[38] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 ed., 2004.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

[40] A. Valada, R. Mohan, and W. Burgard, "Self-supervised model adaptation for multimodal semantic segmentation," *International Journal of Computer Vision*, pp. 1–47, 2019.

[41] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[42] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3431–3440, 2015.

[43] S. Liu, T. Li, W. Chen, and H. Li, "Soft rasterizer: A differentiable renderer for image-based 3d reasoning," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7708–7717, 2019.

[44] N. Ravi, J. Reizenstein, D. Novotny, T. Gordon, W.-Y. Lo, J. Johnson, and G. Gkioxari, "Accelerating 3d deep learning with pytorch3d," *arXiv:2007.08501*, 2020.

[45] C. Zimmermann, D. Ceylan, J. Yang, B. Russell, M. Argus, and T. Brox, "Freihand: A dataset for markerless capture of hand pose and shape from single rgb images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 813–822, 2019.

[46] Y.-H. Tsai, X. Shen, Z. Lin, K. Sunkavalli, X. Lu, and M.-H. Yang, "Deep image harmonization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3789–3797, 2017.

[47] R. Zhang, J.-Y. Zhu, P. Isola, X. Geng, A. S. Lin, T. Yu, and A. A. Efros, "Real-time user-guided image colorization with learned deep priors," *arXiv preprint arXiv:1705.02999*, 2017.

[48] "Carnegie mellon university - cmu graphics lab - motion capture library." `http://mocap.cs.cmu.edu/`.

[49] K. M. Robinette, S. Blackwell, H. Daanen, M. Boehmer, and S. Fleming, "Civilian american and european surface anthropometry resource (caesar), final report. volume 1. summary," tech. rep., Sytronics Inc Dayton Oh, 2002.

[50] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, I. Laptev, and C. Schmid, "Learning from synthetic humans," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 109–117, 2017.

[51] B. O. Community, *Blender - a 3D modelling and rendering package*. Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018.

[52] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao, "Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop," *arXiv preprint arXiv:1506.03365*, 2015.

[53] A. Knapitsch, J. Park, Q.-Y. Zhou, and V. Koltun, "Tanks and temples: Benchmarking large-scale scene reconstruction," *ACM Transactions on Graphics (ToG)*, vol. 36, no. 4, pp. 1–13, 2017.

[54] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[55] M. Cai, F. Lu, and Y. Sato, "Generalizing hand segmentation in egocentric videos with uncertainty-guided model adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14392–14401, 2020.

[56] S. Kolkur, D. Kalbande, P. Shimpi, C. Bapat, and J. Jatakia, "Human skin detection using rgb, hsv and ycbcr color models," *arXiv preprint arXiv:1708.02694*, 2017.

[57] G. Lin, A. Milan, C. Shen, and I. Reid, "Refinenet: Multi-path refinement networks for high-resolution semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1925–1934, 2017.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Realistic Hands: Multi-Modal 3D Hand Reconstruction using Deep Neural Networks and Differential Rendering

Authored by (in block letters):

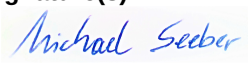
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**
Seeber	Michael

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**
24.05.2021, Zurich	

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.