Markus Müller, BSc

Semi-Supervised Learning of Monocular 3D Hand Pose Estimation from Multi-View Images

MASTER'S THESIS

to achieve the university degree of
Diplom-Ingenieur

Master's degree programme
Software Engineering and Management

submitted to

Graz University of Technology

Supervisor

Prof. Dr. Horst Bischof
Institute of Computer Graphics and Vision

Graz, Austria, Oct. 2020

Abstract

Hand pose estimation enables a vast number of potential applications, for instance, Human-Computer Interaction (HCI), Virtual Reality (VR), sign language and hand gesture recognition. Full 3D hand pose estimation using only monocular color images is complicated, because of many ambiguities, heavy self-occlusion and strong articulation. Most hand pose estimation methods rely on Convolutional Neural Networks (CNNs) and have demonstrated good performance. However, in order to train those networks, a sufficiently large training dataset is required, which generally involves a large annotation effort. This might be feasible for a manageable amount of poses, but this method is impractical to cover a wide variety of different hand poses. In order to reduce the required amount of annotated data samples, semi- or unsupervised methods can be used to achieve a desired level of performance.

In this thesis, we combine ideas and concepts of multiple scientific works, that have mainly been used for human pose estimation and apply them on hand pose estimation. We propose a method, that learns a geometry-aware representation of the human hand from multi-view images without any 3D annotations. We use an encoder-decoder network that learns a geometry-aware latent representation to predict an image from one viewpoint given an image from another viewpoint. We utilize this latent representation to train a fully connected neural network on 3D hand pose estimation. Our results show, that using this latent representation is clearly superior to directly mapping an input image to the 3D joint locations of the hand, if the amount of 3D annotations is limited. Additionally, we introduce a 3D geometric constraint to regularize the 3D hand pose prediction.

Further, we present a fully semi-supervised approach, that learns to simultaneously predict novel views given an image from a different viewpoint and estimate the 3D pose of a hand. We compare the performance of these networks and show how they are affected by different parameter settings.

Kurzfassung

Die Anwendungsgebiete von Handposenschätzung sind zahlreich und umfassen beispielsweise Mensch-Computer Interaktionen, *VR*, Zeichensprache und Gestenerkennung. Das Schätzen von Handposen auf Grundlage einzelner Farbbilder gestaltet sich als sehr schwierig aufgrund von Mehrdeutigkeiten, starker Selbst-Abschattung und Gestikulation in den Bildern. Die meisten Methoden zum Schätzen von Handposen vertrauen daher auf künstliche neuronale Netzwerke und haben bereits vielfach ihre guten Leistungen demonstriert. Jedoch ist es nötig, genügend Trainingsdaten zur Verfügung zu stellen, um solche Netzwerke zu trainieren. Dies ist aber häufig mit einem großen Annotationsaufwand verbunden. Für eine kleinere Anzahl an Posen ist dieser Aufwand durchaus praktikabel, um aber ein breites Spektrum an verschiedensten Posen abzudecken, ist diese Methode nicht realisierbar. Um die Anzahl an benötigten annotierten Daten zu reduzieren, werden semi-überwachte maschinelle Lernmethoden angewandt, um auf ein gewünschtes Leistungsniveau zu kommen.

Um Handposen zu schätzen, kombinieren wir in dieser Masterarbeit mehrere Konzepte aus wissenschaftlichen Arbeiten, die hauptsächlich für die Schätzung von menschlichen Körperposen verwendet wurden. Wir präsentieren eine Methode, die in der Lage ist eine Repräsentation zu lernen, die bereits die Geometrie der Hand beinhaltet. Hierfür verwenden wir nur Bilder von menschlichen Händen aus unterschiedlichen Blickwinkeln, jedoch ohne jegliche 3D Annotationen. Wir nutzen zu diesem Zweck ein Encoder-Decoder Netzwerk, das diese latente Darstellung lernt, um ein Bild einer Hand von einem anderen Blickwinkel als im gegebenen Bild vorherzusagen. Diese latente Repräsentation verwenden wir in weiterer Folge, um ein neuronales Netzwerk für 3D Handposenschätzung zu trainieren. In dieser Arbeit zeigen wir, dass dieser Ansatz deutlich bessere Ergebnisse liefert, als die direkte Zuordnung von einem Eingabebild zur 3D Pose, sofern nur eine beschränkte Anzahl an 3D Annotationen vorhanden ist. Zusätzlich führen wir eine geometrische Beschränkung ein, um die 3D Handposenschätzung zu regularisieren.

Außerdem zeigen wir einen völlig semi-überwachten Ansatz, der das Vorhersagen von einem anderen Blickwinkel als im Eingabebild der Hand, sowie die Schätzung der 3D Handpose gleichzeitig lernt. Wir vergleichen die Leistung dieser Netzwerke und zeigen, wie sich unterschiedliche Parametereinstellung auswirken.

Acknowledgments

First and foremost I would like to thank my supervisor Prof. Bischof for giving me the opportunity to write this thesis. I am equally grateful for my advisors Georg Poier and Horst Possegger, who always guided me in the right direction and came up with new ideas whenever I was stuck on a problem.

Further, I want to thank my friends, especially Eric, who made my leisure time a pleasure by spending time with me doing sports and always offering a sympathetic ear.

Last and most importantly, I would like to thank my family, especially my parents Christa and Günther, and my sister Marina, for supporting my studies and decisions, and motivating me to pursue my goals.

Contents

List of Figures

List of Tables

1 Introduction

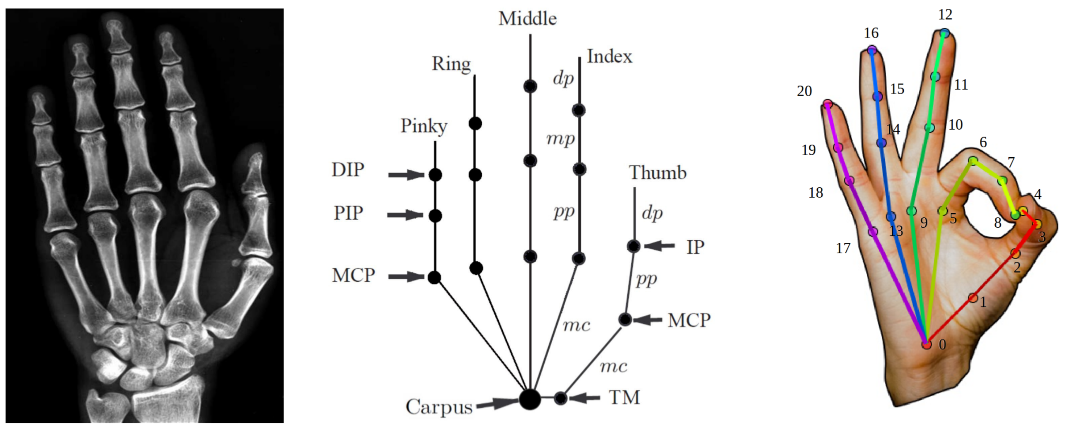
The hand is the tool of tools.

Aristotle

Contents

1.1 Problem Statement

Hand pose estimation is the problem of modeling the human hand as a set of specific parts and finding their positions in a hand image (2D estimation) or the simulation of hand part positions in a 3D space. The hand is almost always modeled as a number of joints and hand pose estimation mostly corresponds to estimating the position of these joints. The human hand has 27 bones, where 19 are located in the palm and fingers and the remaining 8 constitute the wrist. Hand models with 21 joints are most widely used in hand datasets (as shown in Fig. 1.1c), however, there are also datasets that use e.g., 14 [83] or 16 [79] joints [21, 26]. A kinematic model of a hand next to a x-ray image of a human hand and a hand model with 21 detected keypoints as we use in this thesis is shown in Figure 1.1.



(a) X-ray image [43]. **(b)** Kinematic model [67]. **(c)** 21 keypoints [76].

Figure 1.1: Human hand models with joints: **(a)** Anteroposterior x-ray image of a human hand from [43] **(b)** Kinematic hand model (illustration from [67]): dp - distal phalanx; mp - medial phalanx; pp - proximal phalanx; mc - metacarpal; DIP - distal interphalangeal; PIP - proximal interphalangeal; MCP - metacarpophalangeal (joining fingers to the palm); IP - interphalangeal (joining finger segments); TM - trapeziometacarpal; Carpus - carpometacarpal (connecting the metacarpal bones to the wrist) [26]. **(c)** Hand model with 21 keypoints determined by OpenPose [11, 76]. The dataset we use in this thesis has the same number of keypoints and the same structure.

1.2 Applications and Challenges

Hand pose estimation has many different fields of application such as sign language recognition, digital advertising, sterile computer use in operating theaters or home entertainment. Direct sensing in Human-Computer Interaction (HCI) allows humans to instantly communicate and manipulate machines. Huge investments of big technology companies like Google, Microsoft and Facebook on Augmented Reality (AR) and Virtual Reality (VR) have even enlarged the applications of hand pose estimation [21]. Besides the human hand, the motion of the head, eye gaze, face, arms or even the whole body can serve as input for these systems. The hand, however, is the most effective, general-purpose interaction tool, because of its dexterous functionality in communication and manipulation. In object manipulation interfaces, for example, the hand can be used for navigation, selection and manipulation in virtual environments. The human hand has many applications to serve as an efficient, high Degrees of Freedom (DOF) control device [26, 44].

It is very challenging to estimate the pose of a hand based on visual data. According to [26], major difficulties a hand pose estimation system encounters include the following:

1) High dimensionality: The human hand is an articulated object. Theoretically, it has more than 20 *DOF*, however, natural hand motion does not have that many, because of interdependencies between fingers and joints. There still remains a large

number of parameters to be estimated, including the location and orientation of the hand.

2) Self-occlusion: The projection of a hand comes in a large variety of shapes with many self-occlusions, which make it difficult to segment different parts and extract high level features.

3) Processing speed: Depending on the application, this can be a crucial point, since latency requirements in some applications are quite demanding in terms of computational power. A real-time computer vision system needs to process a huge amount of data, even for a single sequence of images. Moreover, some algorithms require expensive, dedicated hardware and possibly parallel processing capabilities to operate in real-time.

4) Uncontrolled environments: Locating a rigid object with an arbitrary background is already a challenging task. It is even more demanding to locate an articulated hand under these conditions. Many *HCI* systems are expected to operate under non-restricted backgrounds and different lighting conditions.

1.3 Goals and Outline

The goal of this thesis is to implement a neural network based on the idea of Rhodin *et al.* [64] and adapt and optimize it to estimate the 3D joint locations of a given RGB hand image. The intention is to find a solution that requires a minimum amount of annotated images for training and still achieves accurate results.

In Chapter 2, we give an overview of the different kinds of pose estimation in general and introduce various approaches and techniques for 2D/3D human pose estimation. Further, we explain hand pose estimation in more detail and introduce the method used by Rhodin *et al.* [64]. We describe our 3D hand pose estimation approach and point out the differences to the concept used by Rhodin *et al.* in Chapter 3. Additionally, we present the dataset we used for our method and explain how we prepared the images. Finally, we provide detailed evaluations of our experiments in Chapter 4 and draw conclusions in Chapter 5.

2 Background

Ideas come from everything.

Alfred Hitchcock

Contents

This thesis mainly focuses on 3D hand pose estimation. However, pose estimation comes in different varieties and has very distinctive meanings among different research fields. This chapter tries to give an overview of the different forms and how different approaches evolved over time.

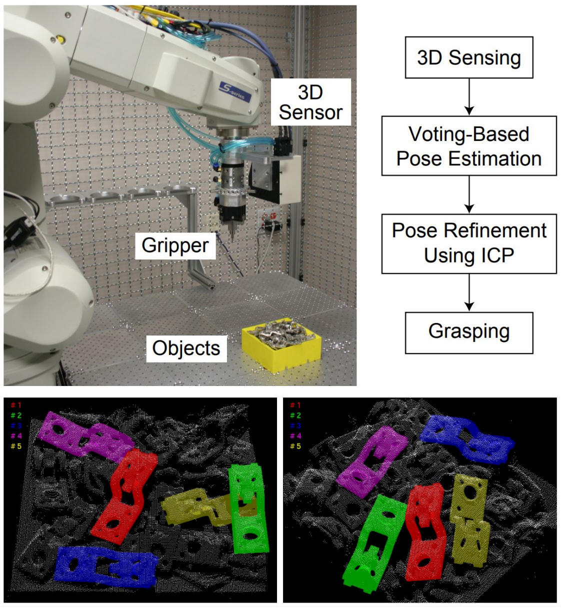
2.1 Pose Estimation

Pose estimation is the fundamental problem of deriving the pose of a person or object in an image or video. It can be seen as the recovery of the 3D geometric information from 2D images, as well as the problem of determining the position and orientation of a camera relative to a given person or object. Both are often done by identifying, locating, and tracking a number of keypoints on a given object or person. For objects, these keypoints could be corners or other significant features and for humans, these could represent principal joints like a shoulder, elbow or knee.

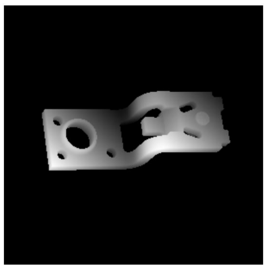
Pose estimation has many different applications including robotics [51], gaming [97], animation [63] along with augmented reality [49] and so on. There are also different kinds of pose estimation and some of them are described in the following sections.

2.1.1 Rigid Pose Estimation

Rigid pose estimation is mostly found in industrial robotic manufacturing and assembly tasks. There it is often necessary to place the parts to be manipulated in pre-defined positions using part feeders or fixtures, because robots are often lacking perception. Such constraints, however, lead to high production costs and low adaptability to new tasks. With the evolution of Industry 4.0, where a large variety of products is presumably fabricated in small production volumes, robots need to be equipped with flexible and adaptive skills, which means to work with arbitrary initial part conditions. One strategy for dealing with this kind of problems are model-based paradigms. This involves building 3D models of the objects used in specific tasks and then determining object poses by fitting their models to new images with the help of detected features [51, 52]. An example of a robotic bin-picking system using 3D models is shown in Figure 2.1.



(a) Overview of a robotic assembly system using a 3D scanner [16].



(b) *CAD* model [16].

Figure 2.1: **(a)** Shows a setup of a bin-picking system, which uses a 3D sensor attached on a robot arm to grasp randomly placed objects in a bin. On the right, the algorithm flowchart of the system is illustrated. The bottom images show the pose estimation results with the five best pose estimates superimposed on the scanned 3D point cloud. This scenario is challenging due to noise, missing data, clutter and occlusions. **(b)** Shows the corresponding 3D *CAD* model of a circuit breaker [16].

The system from Choi *et al.* [16] shown in Figure 2.1 represents a bin-picking system with a 3D sensor attached on a 6-axis industrial robot arm to estimate the poses of objects randomly placed in a bin. As shown in the flowchart, the setup first scans a bin of objects using the 3D sensor. Then a voting-based algorithm tries to detect and estimate the pose of the target object using the scanned 3D point cloud, given a 3D *CAD* model of the target. This results in multiple coarse pose hypotheses. A certain number of best pose hypotheses is then refined using an Iterative Closest Point (ICP) algorithm. During the refinement process, the *CAD* model is rendered using the current pose estimate and 3D points for the model are generated by sampling the surface of the rendered model. Afterwards, the refinement algorithm finds the nearest 3D point in the scanned point cloud for each 3D point in the model and updates the pose estimate employing the 3D point correspondences. After the refinement, the registration error is calculated by the average distance between the model points and the corresponding scene. The system grasps the object if the registration error is small and the estimated pose is safely reachable by the robot arm.

Vision-based object manipulation and grasping requires accurate estimation of the object's 6D pose (3D translation and 3D rotation) in an image. 6D object pose estimation is a very challenging computer vision task and is used in autonomous navigation and augmented reality as well, besides robotics. Such systems should be able to handle objects of varying texture and shape, different lighting conditions, sensor noise and be robust towards heavy occlusion. At the same time, the speed requirements of real-time tasks have to be met [36, 52, 87].

2.1.2 2D Human Pose Estimation

According to [74], human pose estimation is defined as the process of estimating the configuration of the body (pose) from a single, typically monocular, image.

Human pose estimation has been a challenge in computer vision for several years. The importance of it arises from its vast number of applications. In the context of *HCI* and activity recognition, for example, it enables higher level reasoning, and in the field of marker-less Motion Capture (MoCap) it constitutes one of the basic building blocks. *MoCap* technology applications range from character animation to clinical analysis of gait pathologies [74].

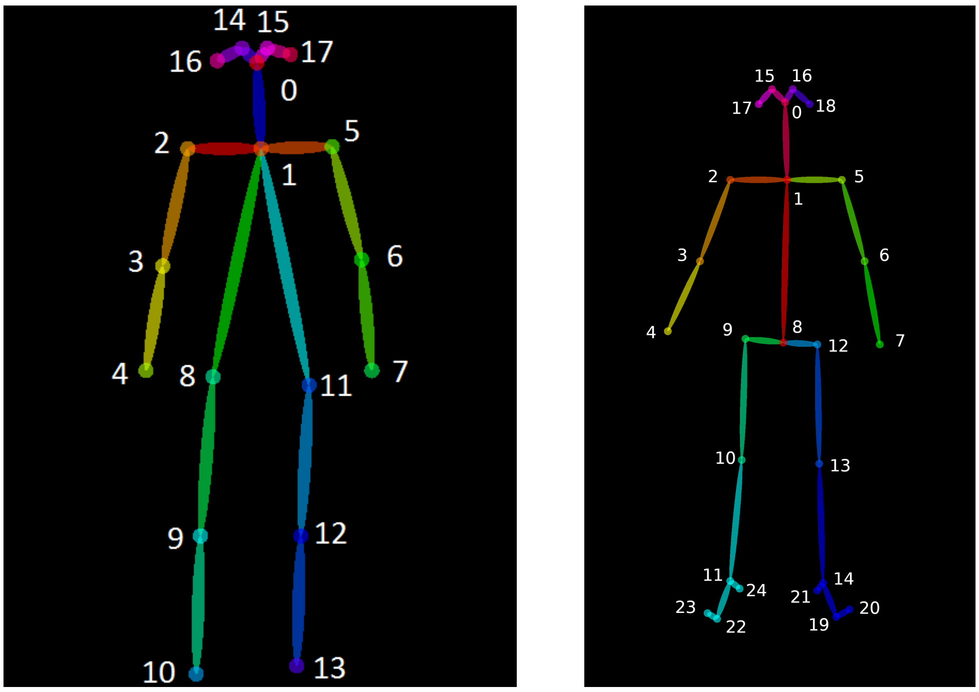
Pose Estimation has to deal with troublesome challenges. In [74] the following are stated:

1) Varying visual appearance of humans in images,
2) variability of lighting conditions,
3) differences in human physique,
4) partial occlusions as a result of self-articulation and layering of objects in the scene,
5) complexity of the human skeletal structure,
6) high dimensionality of the human pose, and

7) the loss of 3D information due to the 2D planar image projections.

Trying to produce satisfying results in general and unconstrained settings while tackling the previously mentioned challenges can be a very hard endeavor.

The output of a human pose detector can be represented in different ways. The most common human body models emerged from the Max-Planck Institute for Informatics (MPII) [2] and the Common Objects in Context (COCO) [50] datasets. The *MPII* dataset annotates ankles, knees, hips, shoulders, elbows, wrists, necks, torsos, and head tops, while *COCO* also includes some facial keypoints. For these datasets, however, foot annotations are limited to the ankle position only. Figure 2.2 shows a representation of the *COCO* and the BODY_25[1] keypoint model, which consists of the *COCO* model including feet [11]. The foot keypoint annotation consists of big toes, small toes and heels.



(a) *COCO* body model. **(b)** BODY_25 (*COCO*+Foot).

Figure 2.2: Keypoint annotation configuration for two example datasets [11].

Common techniques to deal with 2D pose estimation are *part-based*, *detection-based* and *regression-based* methods [54]. In part-based methods, an object is represented as a collection of parts in a deformable configuration using the Pictorial Structure (PS) framework, which is explained in Section 2.1.2. Detection-based methods deal with pose estimation as a heatmap prediction problem. Each pixel in a heatmap represents the detection score of a corresponding joint. An example for heatmap prediction is the Stacked

[1] `https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/output.md`

Hourglass Network which is explained in more detail in Section 2.1.2. Current State-of-the-Art (SotA) approaches are proposing sophisticated variations of this network architecture. These methods, however, do not provide joint coordinates directly. To find the pose in (x, y) coordinates, regression-based techniques are used. They use a non-linear function, that maps the input directly to the desired output, which can be 2D joint coordinates. An example for a such a method is described in Section 2.1.2, which employs a holistic solution using cascade regression for body part recognition.

Pictorial Structures

One of the first techniques to estimate the human pose from a single 2D image is the *PS* framework, introduced by Fischler and Elschlager [29] in 1973. As stated before, human pose estimation is the problem of localizing human joints. However, since the human body can strike many different poses, it is very difficult to find the location of body parts and joints from a single image. The authors of [29] used the concept of part-based modeling for facial structure estimation, utilizing pictorial structures. This is a general method, which consists of identifying and configuring parts to form a structure by discretizing the search space using dynamic programming. The representation of objects is done by a set of parts, that are organized in a deformable structure. Each part is identified separately and represented with spring-like connections between paired parts to obtain a deformable structure as shown in Figure 2.3 [41].

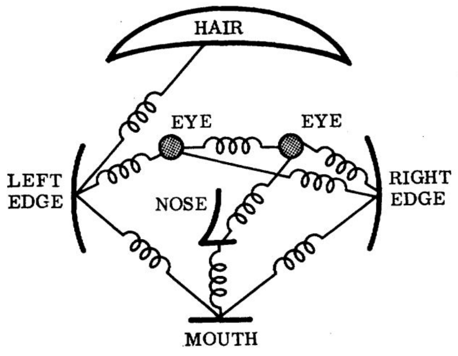


Figure 2.3: Schematic illustration of a face, indicating its components and their linkages [29].

The *PS* framework has been improved and made more tractable by using distance transforms for object recognition. Distance transform matching, also called Chamfer matching [7], is a technique to find the best fit of edge points from two different images by minimizing a generalized distance between them. The idea is to transform the edge points of one image by a set of parametric transformation equations, to match another geometrically related representation of the same geometric structure [32].

In this framework, a body is represented as an assembly of parts, which are connected by constraints imposed by the joints within the skeletal structure. This formulation reduces the inference complexity, since the search for probable body part locations can be done independently, only considering adjacent body parts that constrain them. This substantially prunes the total search space [74]. Each body element can be transformed to match its position in the given image. The various complex parameters can be understood as orientation, foreshortening and so on. Matching the pictorial structures to an image is determined by the minimization of an energy function [28, 41].

Figure 2.4 shows a limb-based human body model. The crosses indicate the joints between parts and each part corresponds to a limb. Pictorial structures work well when dealing with complicated backgrounds and for general object detection in 2D. Furthermore, they can be extended to 3D human body pose estimation [23].

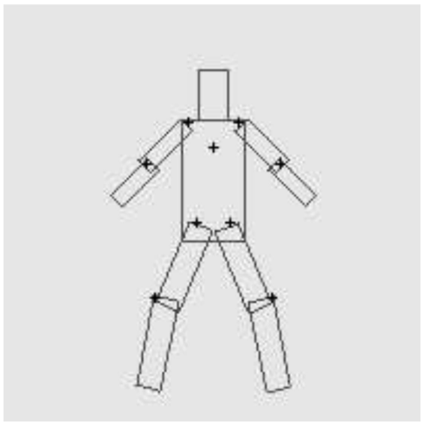


Figure 2.4: Limb-based human body model from *PSs* [28].

The *PS* framework has been a beneficial and important contribution to the field of human pose estimation. It enabled more flexible part configurations and multi-modal pairwise terms, color features [22] and mixtures of pictorial structures [92] for more efficient and accurate human pose estimation.

Another milestone was reached after the introduction of *DeepPose* [84]. This started shifting the research on human pose estimation from classical methods to Deep Neural Networks (DNNs).

Deep Neural Networks

The introduction of *DNNs* has improved the performance and accuracy of human pose estimation dramatically. Toshev and Szegedy [84] formulated pose estimation as a *DNN*-based regression problem to estimate the location of body joints. The introduction of cascading *DNN* regressors resulted in high precision pose estimation. This method allowed for reasoning about pose in a holistic fashion and had a simple but powerful formulation, which took advantage of previous advances in deep learning. For example, even if certain joints were occluded, they could be estimated if the pose was reasoned about holistically [41, 84].

Toshev and Szegedy [84] located each body joint by regression using the full image as an input. This formulation has the advantage of capturing the full context of each body joint, since the full image is used as an input signal for each regressor. Another benefit of this approach is, that it is much simpler to formulate than methods based on graphical models, because there is no need to specifically design feature representations and detectors for parts. Furthermore, no model topology or interactions between joints need to be designed. Instead, a generic convolutional *DNN* is able to learn these implicitly from data. *DeepPose* [84] was the first implementation of a *DNN* which used a holistic approach for human pose estimation.

Figure 2.5 shows a schematic view of the network layers with their corresponding dimensions. This generic Deep Convolutional Neural Network (DCNN) consists of seven layers, where each layer is a linear transformation followed by a non-linear one. The network takes a color image of predefined size as input and outputs the target values of the regression. In the case of *DeepPose* [84], the *DCNN* gives $2k$ joint coordinates as output, where k is the number of joints [41, 84]. The *DNN* architecture used in *DeepPose* has been based on the work of Krizhevsky *et al.* [45] for image classification, which is better known as *AlexNet*.

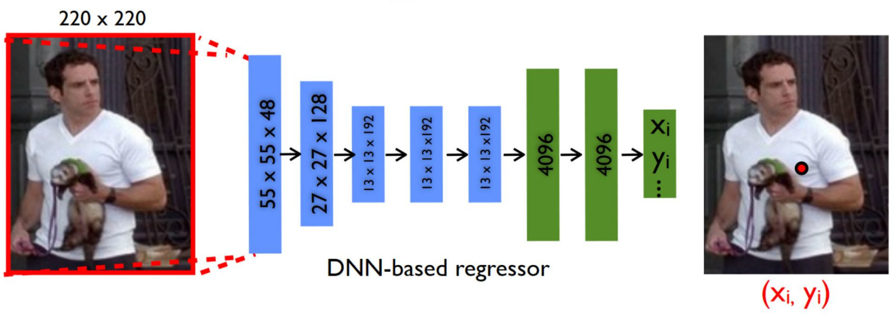


Figure 2.5: Schematic view of the *DNN*-based pose regression. Convolutional layers are blue and fully connected ones are green [84].

A different technique presented in [15] specifies a graphical model for the human pose which takes advantage of the fact that the local image measurements can not only be used for part (or joint) detection, but also to predict the spatial relationships between them (Image Dependent Pairwise Relations (IDPRs)). *DCNNs* are used to learn conditional probabilities for the presence of parts and their spatial relationship within image patches. This approach combines *DCNNs* with the representational flexibility of graphical models. This has led to better results compared to the previous methods [41, 84, 92].

Another improvement in 2D human pose estimation has been the combination of a local part-based approach and a holistic method presented in [27]. This technique is called Dual-Source Deep Convolutional Neural Network (DS-CNN) (illustrated in Fig. 2.6) and has achieved more efficient results than the previous methods. The *DS-CNN* takes a set of image patches as input and it uses the local part appearance, as well as the holistic view of each local part to estimate a human pose. This approach has achieved not only joint detection, which determines whether an image patch contains a body joint, but also joint localization, which finds the exact location of the joint in the image patch. The part patches that include local appearance were generated by region proposals with some restrictions. The regions should neither be too small, nor too large. If the size is too small, its appearance may not provide sufficient features. If the size is too large, it may cover multiple body parts and its appearance lacks sufficient resolution for joint detection and localization [27, 41].

The body patches on the other hand, have to cover the whole body or all joints. For the *DS-CNN* each training sample consists of a part patch, a body patch and a binary mask specifying the location of the part patch within the body patch. Based on the outputs of the network, the joint detection results from all sliding windows are combined to construct a heatmap that reflects the joint location likelihood at each pixel. In order to compute the final estimate for each joint location, a weighted average is computed over the regions of the heatmap [27, 41]. Figure 2.6 shows an illustration of the described *DS-CNN*.

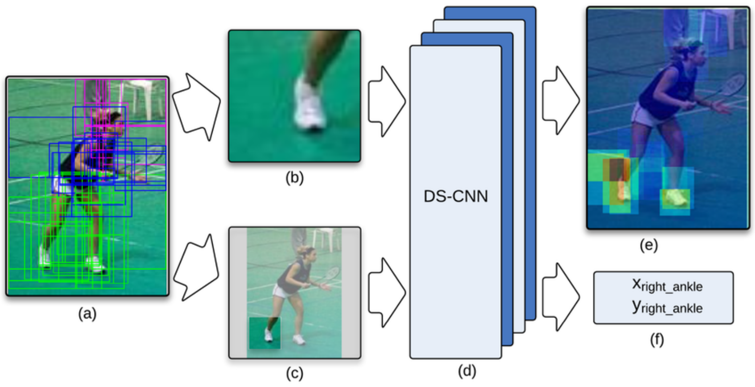


Figure 2.6: Schematic view of the *DS-CNN* method from [27]. (a) Input image including generated image patches. (b) Input of an image patch for the *DS-CNN*. (c) Input of a full body patch and holistic view of the local body part. (d) The *DS-CNN* itself. (e) *DS-CNN* output on joint detection. (f) *DS-CNN* output on joint localization.

Although the usage of convolutions in *DNNs* for human pose estimation caused a dramatic improvement, it does not capture information at every scale. One architecture that has tackled those challenges amongst others, has been the Stacked Hourglass Network, introduced by Newell *et al.* [59].

The Stacked Hourglass architecture allows for repeated bottom-up (from high resolutions to low resolutions), top-down (from low resolutions to high resolutions) inference across scales in conjunction with intermediate supervision. The hourglass design was incentivized by the need to capture information at every scale. While local evidence is crucial to spot features like hands and faces, a final pose estimation needs a coherent perception of the full body. Therefore, features in an image are merged at different scales to find cues like the orientation of the person, the arrangement of the limbs, the relationships of adjacent joints, and so on. The hourglass design is able to capture all these features and combine them to output pixel-wise predictions. In Figure 2.7 the stacked hourglass modules of the network are depicted [59].

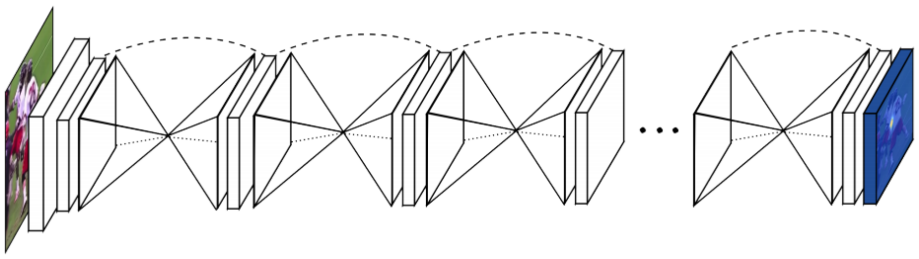


Figure 2.7: Illustration of the stacked hourglass network modules (image taken from [59]).

The design introduced by Newell *et al.* [59] consists of eight stacked hourglass modules. Figure 2.8 represents a single module. Weights are not shared across modules and a loss is applied to the predictions of all hourglasses utilizing the same ground truth. The network produces a set of heatmaps where each heatmap represents the likelihood of a specific joint at every pixel.

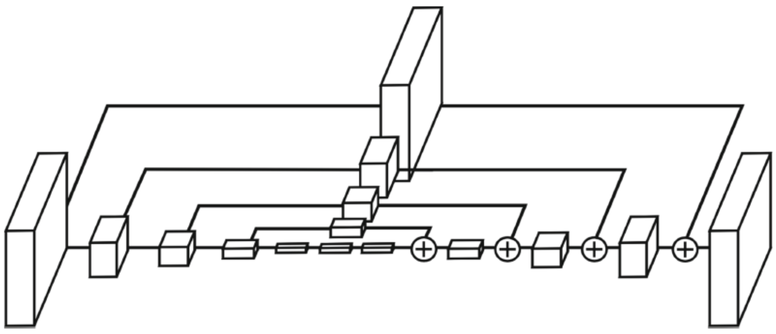


Figure 2.8: Schematic image of a single hourglass module. Each box resembles a residual module and the layers are identical across the whole stacked hourglass network [59].

In the hourglass network, presented in [59], convolutional and max pooling layers are applied to process features down to a very low resolution. When the lowest resolution is reached, the network starts with the top-down sequence of upsampling and combining features across scales. The merging of information between two adjacent resolutions is done by nearest neighbor upsampling of the lower resolution followed by an element-wise addition of the two sets of features. Since the hourglass has a symmetric topology, every down-sampling operation has a corresponding up-sampling operation.

This recursive design of Convolutional Neural Networks (CNNs) has reached *SotA* results and became a guideline for researchers. The network handles challenging and diverse poses very well with a simple mechanism for reevaluation and assessment of initial predictions [41, 59]. However, this network still has some deficiencies e.g., relationships between keypoints are not considered, since each keypoint heatmap is estimated independently [39]. Furthermore, the encoder can be replaced with a pretrained Residual Neural Network (ResNet) to extract features more effectively [71].

Many of the previously mentioned approaches for 2D pose estimation have been extended and improved to handle the 3D pose estimation task.

2.1.3 3D Human Pose Estimation

3D human pose estimation has recently received significant attention from researchers, mainly due to an increasing range of applications in scientific and consumer domains, which are driven by current technological advances. The scopes of application range from gaming and sports performance analysis to *HCI* and Human-Robot Interaction (HRI), where computers or robots can be controlled by human gestures [69].

Recovering the 3D pose from 2D RGB images is generally more challenging than estimating the 2D pose. Some reasons are i) more ambiguities, ii) larger 3D pose space, iii) different 3D poses can be represented by similar image projections (ill-posed), iv) high non-linearity in human motions, v) variations in pose and appearance. The usage of depth maps has proven to be more effective for 3D pose estimation than 2D RGB images. However, depth cameras have a limited operating range, high power consumption and are often more expensive than ordinary RGB cameras. Moreover, the majority of media on the internet is still in a 2D RGB format [48, 69].

The task to estimate the position of human joints in 3D is not only challenging for computers, but for humans as well. Humans are, on average, not better at reproducing 3D poses under laboratory conditions, given visual stimuli than current computer vision algorithms [69]. For humans, it is significantly tougher and more time-consuming to reenact hard poses compared to easier ones. Consequently, hard poses lead to higher errors compared to easier ones. The categorization between hard and easy poses arose during the construction of the dataset on which humans were tested [56].

Similar body models, which find use in 2D pose estimation (as shown in Fig. 2.2) are as well used in 3D pose estimation. Usually constraints are enforced on body models to constrain the pose parameters. Such kinematic restrictions make sure that limb lengths, limb-length proportions, and joint angles follow certain rules [69].

Most of the human pose estimation algorithms can be broadly categorized as *generative* and *discriminative* methods [60, 85]. The major difference is, that generative methods start from a human body model initialized with a pose and project the pose to the image plane to verify image evidence, while discriminative methods start from the image data and usually learn a mechanism to model the relations between image projections and human poses based on training data [33, 69].

Discriminative Methods

Discriminative approaches do not assume a particular model, since they start from an RGB or depth image and estimate 3D poses by a mapping- or search-based algorithm. The model, which describes the relation between images and human poses can be obtained by learning a direct mapping from inputs x to class labels y, or by directly modeling the conditional distribution $p(y|x)$ as a parametric model [60, 69].

Discriminative methods can be further classified into *learning-based* and *example-based* methods. Learning-based approaches learn a mapping function from an image to the pose space from a training set. This mapping should generalize well for a new image from the testing set. Examples are [33, 69]

a) deep learning methods, which consist of multiple non-linear transformations. *DCNNs* are one of the most popular models for computer vision problems. Since they have a reduced number of parameters compared to fully connected networks, which makes training easier and reduces overfitting.

b) Space-learning based methods [25, 30], which use both topology space and subspace to learn a mapping, and

c) bag-of-words based methods [61]. The basic idea for this approach is to first extract the most representative features as a vocabulary and afterwards express each training sample based on image observations and the vocabulary in a statistical way.

Example-based methods estimate human poses based on a discrete set of specific poses with their corresponding pose descriptors. The final pose can then be estimated by interpolating the candidates obtained from a similarity search. Randomized trees [1] and random forests [12] are classification techniques that can handle this type of problem in a fast and robust manner [33, 69].

Discriminative methods have the advantages, that they are typically very fast at making predictions for new data points (at test time) and they usually have better predictive performance than generative methods [85].

Generative Methods

Generative approaches learn a model of the joint probability $p(x, y)$ from inputs x and class labels y. This is usually done by learning the conditional probability $p(x|y)$ and the prior probability $p(y)$ separately. The final posterior probabilities $p(y|x)$ are then calculated using Bayes' theorem [85].

One sub-category of generative methods are part-based approaches. In the part-based approach, the human skeleton is represented as a collection of body parts connected by constraints imposed by the joints within the skeletal structure. The human pose is estimated by learning body part appearance and position models. First, body part candidates are detected from image observations, then the detected body parts are assembled to fit images and a body structure. The *PS* model is a very good representation for part-based models. It has mainly been used for 2D human pose estimation (as described in Section 2.1.2) but it has been extended for 3D pose estimation [8]. The *PS* framework is very powerful and enables an efficient inference of respective body parts, because it represents the human body as a collection of parts arranged in a deformable configuration [33, 69].

Generative approaches generalize well and are typically better at handling missing or partially unlabelled data. Therefore, they can easily handle compositionality, e.g., faces with glasses and/or facial hair [69, 85].

Hybrid Approaches

Hybrid methods take advantage of both discriminative and generative approaches, and avoid their shortcomings to enable more precise pose predictions [33, 69]. The combination of these methods is usually implemented by initializing the pose obtained from the discriminative mapping functions and optimizing the human pose within a local area through generative methods [75]. One possible combination of model-free (discriminative) and model-based (generative) approaches is to introduce distance constraints in the discriminative methods and use generative methods to impose constraints between the output dimensions [68]. Another way to coalesce these methods is to use the observation likelihood from a generative model to verify the pose predictions obtained from a discriminative mapping function [65].

2.1.4 Hand Pose Estimation

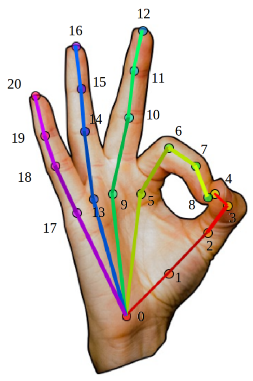
Most of the methods and techniques previously mentioned can be applied on hand pose estimation as well. The problem statement for hand pose estimation is of course different compared to the human pose estimation problem, however, the basic idea is the same.

Before deep learning solutions became popular, traditional machine learning and computer vision algorithms were used for hand pose estimation. Early approaches were mainly based on low level visual cues like silhouette [46, 72] or optical flow [53], and used generative models to resolve the depth ambiguity. Among many traditional approaches, random forest algorithms and its variations were the most popular ones [40, 47, 78, 80]. Random forests are a type of ensemble method and have been very successful as a general purpose classification and regression method. They make predictions by averaging over the predictions of several independent base models, where each model is trained in isolation from each other [19]. The Microsoft® Kinect™ system [73], for example, uses a random forest as a classifier for human body pose estimation. Essentially, the first step in their joint estimation pipeline is a prediction of body part labels of each pixel in the depth map using a random forest. The second step is to use the labelled pixels to predict joint locations [21, 86].

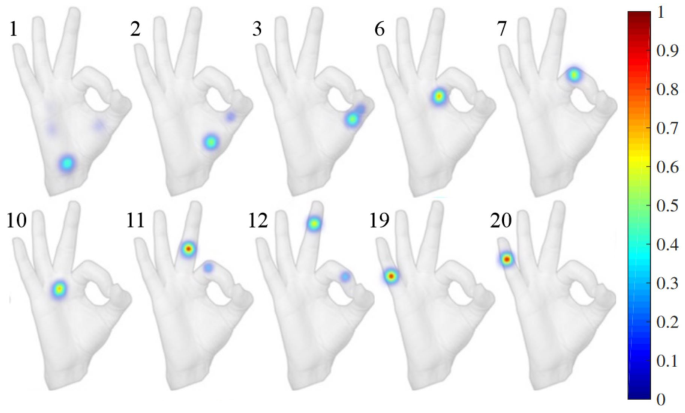
In what follows, we introduce some approaches, that use deep learning to solve the hand pose estimation problem. We categorize them as *detection-based* and *regression-based* as well as *depth-based* and *image-based* methods.

Detection-based Methods

Detection-based approaches produce a probability density map for each joint. For a hand model with 21 joints, a neural network would produce 21 different probability density maps as heatmaps for each input image. To finally find the exact location of each joint, an *argmax* function has to be applied on the corresponding heatmap [21, 94, 96]. Figure 2.9 shows a hand model with 21 keypoints next to an image of confidence maps. We use the same hand annotations from Figure 2.9a for our hand pose estimation approach as well.



(a) Detected keypoints.



(b) Confidence maps.

Figure 2.9: Output of a detection-based algorithm (images taken from [76]) **(a)** Annotated hand with 21 keypoints determined by OpenPose. **(b)** Selection of produced confidence maps.

The detection-based methods used in [57, 76] are described in detail below in *Depth-based Methods* and *Image-based Methods*, respectively.

Regression-based Methods

Regression-based approaches directly map an input image to the joint locations or the joint angles of a hand model. For a hand model with 21 joints, the final layer of a neural network must have 3×21 neurons to predict (x, y, z) coordinates of each joint. Regressing from highly disparate domains like pose and image is a very challenging task for a neural network. The advantage of these methods is their ability to capture global constraints and correlations among different joints, without having to solve for intermediate representations such as 2D coordinates [21, 86, 94].

Regression-based methods are used in [4, 31, 77] and are explained in more detail below. The approaches in [58, 100] use detection-based as well as regression based networks and are also explained in the following.

Depth-based Methods

Many contemporary works rely on depth images for hand and body pose estimation. A lot of the deep learning-based approaches for 3D hand pose estimation using single depth maps, directly regress the 3D coordinates of the keypoints from the 2D input depth map. The authors of *DeepHand* [77] implemented a regression-based method to find the 3D pose of a hand based on depth data from commercial 3D sensors. They hierarchically regressed the hand pose from global to local joint angle parameters. To estimate the pose of a hand, they separately trained a network for each finger and another network for the global hand orientation. The global pose orientation and the finger articulations were combined in the end to robustly estimate the hand pose. To extract the Region of Interest (ROI), they use a colored wristband as a simple indicator of the hand region, which helped with the removal of extraneous pixels like those below the wrist, which lead to better performance. Therefore, they used RGB images as input of their system as well [21, 77].

Other hand pose estimation techniques use Generative Adversarial Networks (GANs) [4, 13, 99]. A *GAN* [34] is a framework for estimating generative models through an adversarial process. Two models are trained simultaneously in this procedure: a generative model which captures the data distribution, and a discriminative model which estimates the probability whether a sample came from the training data rather than the generative model. The goal of training the generative model is to maximize the probability of the discriminative model making a mistake. In order to achieve good results, both models must be synchronized well during training which means that the generative model must not be trained too much without updating the discriminative network. *GANs* can improve semi-supervised learning techniques, where it can generate additional training data when only limited labeled data is available.

In [4] a *GAN* was used to estimate the 3D pose of a hand by making a one to one relation between depth disparity maps and 3D hand models. Since collecting a fully annotated and comprehensive dataset covering diverse camera perspectives, shapes and pose variations is laborious, the authors used a *GAN* to generate new data samples. Their idea was to synthesize data in the skeleton space, instead of doing so in the depth-map space. The generated hand skeletons were then used to train another generator, which synthesized the corresponding depth maps. The joint training of a hand pose generator and a hand pose estimator in a single unified framework made up an algorithm, which was robust to variations that go beyond the coverage of existing hand databases.

The authors of [31] followed a different approach and created a three dimensional *CNN* to capture the spatial structure from a single depth map as input and accurately regress full 3D hand pose in a single pass. In the first step, they segmented the human hand from the depth image and the 3D point cloud of the hand was encoded as 3D volumes storing the projective Directional Truncated Signed Distance Function (D-TSDF) values. Afterwards, these values were fed into a 3D *CNN*, which contains three 3D convolutional layers and three fully-connected layers. This network yielded a set of 3D hand joint locations in the 3D volume. The final joint locations were obtained by applying simple coordinate transformations. This method is robust to variations in hand sizes and global orientations, since they performed 3D data augmentation on the training set by directly applying 3D transformations on the 3D point clouds.

A Truncated Signed Distance Function (TSDF) is a volumetric representation of a scene for integrating depth images. It is well-suited for data-parallel algorithms, which are able to achieve real time processing at high frame rates depending on the hardware. In accurate *TSDFs*, each voxel stores the signed distance from the voxel center to the closest surface point. This distance is positive when the voxel is in front of the visible surface and negative when it is occluded by the the visible area. Projective *D-TSDFs* only find the closest point on the line of sight in the camera frame and the Euclidean distance is replaced with a 3D vector representing the three distances in each direction in the camera coordinate system [31, 89].

A similar voxel based approach was followed by the authors of [57], who came up with a voxel-to-voxel prediction network for pose estimation. A 2D depth image is converted into a 3D voxelized form, which serves as input for the network. From this voxelized grid, the network estimates the per-voxel likelihood for each keypoint. The position of the highest likelihood response for each keypoint is then warped to real world coordinates, which are the final results of the model. Since the input and output of the network are in 3D, it is based on the 3D *CNN* architecture that treats the Z-axis as an additional spatial axis. The network architecture they used is based on a slightly modified hourglass model, which was described in Section 2.1.2. Their network performed very good on almost all depth-based hand datasets compared to other algorithms and it could easily be applied on body pose estimation as well. The voxel-to-voxel network has overcome the problem of perspective distortion in the 2D depth map. Many prior methods have treated depth maps (which are intrinsically 3D data) as 2D images, which can distort the shape of the actual object as a consequence of projection from 3D to 2D space [21, 57].

Depth-based methods perform very well on the 3D hand pose estimation problem, however, depth cameras are not as commonly available as regular cameras and they only work reliably in indoor environments.

Image-based Methods

Neural networks that use simple RGB images as input for hand pose estimation can be used across a huge number of devices. However, reducing the input dimension to 2D makes the task a lot harder. Furthermore, training a network utilizing RGB images requires considerably more data compared to training a similar network using depth maps. Image-based methods need to isolate the hand beforehand with cropping and resizing operations, which are then passed to a pose estimation network. Using a segmentation network similar to *SegNet* [3] is one way to achieve this task. *SegNet* is a deep fully convolutional network for semantic pixel-wise segmentation, designed to segment general pictures like road or indoor scenes [3, 21].

The 3D hand pose estimation pipeline from [100] consists of three building blocks. In the first step, a segmentation network called *HandSegNet* localizes the hand within an input image. The mask provided by the segmentation was used to crop and normalize the inputs in size. The cropped images were then used to localize 2D keypoints as estimation of 2D score maps, where each map contains information about the likelihood that a certain keypoint is present at a spatial location. The 2D predictions were afterwards utilized as input for a third network, which derived the 3D hand pose from the 2D keypoints. This regression-based network predicts the most likely 3D configuration given the 2D evidence. The network does not predict the absolute 3D coordinates, but rather estimates the coordinates within a canonical frame and the transformation into the canonical frame. This explicitly enforces a representation that is invariant to the global orientation of the hand [21, 100].

The biggest issue for hand pose estimation from RGB images are occlusions, which happen when an object or the hand itself occludes parts of the hand. To overcome this problem, the authors of [76] introduced a multi-camera system to boost the performance of a hand keypoint detector. With the multi-view bootstrapping procedure, they were able to detect hand keypoints in realtime on RGB images and produce 3D markerless *MoCap* of hands by triangulation of multiple 2D detections. Since large datasets of annotated keypoints do not exist for hands, their approach was based on the assumption that even if a particular image of a hand has significant occlusions, there often exists an unoccluded view. At first, they trained a weak detector on a small annotated dataset to localize keypoints in *good* views and filter out incorrect detections, using robust 3D triangulation. Keypoints in images with severely occluded hands were identified by reprojecting the triangulated 3D hand joints. These newly generated annotations were then added to the training set and by iterative improvement of the keypoint detector, the detections got more and more accurate at each iteration. The multi-view constraints allow for geometrically consistent hand keypoint annotations, even on images that are difficult or impossible to annotate due to heavy occlusion. The multi-view bootstrapping technique can be used to generate annotations for any keypoint detector, that is prone to occlusions by improving the quality and quantity of annotations. The major bottleneck in many machine learning and computer vision problems is building an annotated dataset, that is large enough. Multi-view bootstrapping is one way to improve semi-supervised learning [76].

A different approach to overcome the problem with occlusions and especially the lack of enough annotated data has been proposed by [58]. They used synthetically generated data, that was created by an image-to-image translation network, producing geometrically consistent synthetic images. The main advantage of synthetic data is that the ground truth 3D joint positions are known. The drawback is, however, that they usually lack realism. A strong disadvantage of a network solely trained on synthetic images is that it has limited generalization to real images. To deal with this problem, they used an image-to-image translation network with the objective to translate synthetic to real images. They trained a *GAN* based on CycleGAN [99] that translates synthetic images to real images. A CycleGAN is able to capture special characteristics of one image collection and figure out how these characteristics can be translated into the other image collection, without any paired training examples. This problem is described as image-to-image translation, converting an image from one representation of a given scene to another one. The advantage of this network architecture is, that it does not require paired images, which means for hand data generation, that there does not need to exist a real counterpart for a given synthetic image.

The network, proposed by [58], learned mappings from synthetic to real images and from real to synthetic images. To both mappings, they incorporated an additional geometric consistency loss to ensure that both produce images, that maintain the hand pose during image translation. It is important to ensure that the ground truth joint locations of the synthetic images are also valid for the generated *real* images. These generated images were then used to train their *RegNet*, which is a *ResNet* derived from the *ResNet50* architecture [35] consisting of 10 residual blocks. The 2D joint predictions, represented as heatmaps and the 3D joint coordinates relative to the root joint were both obtained from the *ResNet*. Both were then used to fit a kinematic skeleton model. The hand pose was then derived from the adjusted kinematic model. Due to the enrichment of synthetic data, this method generalizes better than others since the training data better resembles the distribution of real hand images and it is more robust to occlusions [58, 99].

ResNets [35] make the training and optimization of *DNNs* a lot easier, since the layers are formulated as residual functions. Instead of hoping that each few stacked layers directly fit a desired underlying mapping, a deep residual learning framework explicitly allows these layers to learn a residual mapping. Formally, let the desired underlying mapping to be $\mathcal{H}(\mathbf{x})$ and let the stacked non-linear layers fit another mapping of $\mathcal{F}(\mathbf{x}) := \mathcal{H}(\mathbf{x}) - \mathbf{x}$. The original mapping is recast into $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ and it is easier to optimize the residual mapping than the original, unreferenced mapping. If an identity mapping were optimal, it would be easier to push the residual to zero, than to fit an identity mapping by a stack of non-linear layers. As shown in Figure 2.10, the formulation of $\mathcal{F}(\mathbf{x}) + \mathbf{x}$ can be realized by feedforward neural networks with *shortcut connections*. Those are skipping one or more layers to perform identity mapping and their outputs are added to the outputs of the stacked layer.

Besides depth-based and image-based methods, there are approaches which use RGBD images (combination of RGB and its corresponding depth image) during training as well. Such a technique was used in [20] with the goal of estimating the 3D pose of a hand from a single monocular RGB image. They trained a *CNN* on purely synthetic data

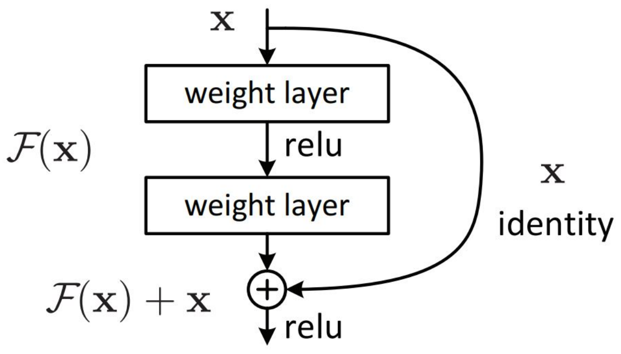


Figure 2.10: Residual building block, as illustrated in [35].

consisting of masked-out renderings of hands in different poses, shapes, illuminations and textures, including the corresponding 3D annotations. This network alone gave good initial predictions on various real data, however, there still existed a high discrepancy between the results on synthetic and real data. Therefore, the authors extended this network with an unsupervised refinement component, that minimized a depth loss, given a differential rendering of the initial pose estimate. The depth loss is minimized on unlabelled depth images, that have one-to-one correspondences to the real RGB input images. This component allowed to fine-tune the network on unseen unlabeled real RGB data, provided that an analogue unlabeled depth image was present at training time. The performance of the network has clearly improved after the refinement process [20].

2.2 Notation and Conventions

Before giving a closer review of the related literature, we introduce the mathematical notations which are used throughout the thesis. Scalar values are represented by italic fonts, e.g., x or c_i. Matrices and vectors are depicted in bold font, e.g., $\mathbf{M}$ or $\mathbf{v}$. Vector spaces are depicted in blackboard bold upper case letters, e.g., $\mathbb{R}^3$ or $\mathbb{Z}^3$. Functions, mapping between different spaces are given in upper case calligraphic letters, e.g., $\mathcal{P}$ or $\mathcal{H}$. An overview over the notation is given in Table 2.1.

Entity	**Notation**
Scalar	a, c_i
Vector in 2D	$\mathbf{v} = (x, y)^{\mathrm{T}}$
Vector in 3D	$\mathbf{X} = (x, y, z)^{\mathrm{T}}$
Matrix	$\mathbf{M} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$
Vector Space	$\mathbb{R}^3$
Mapping Function	$\mathcal{P} : \mathbb{R}^3 \rightarrow \mathbb{R}^2$

Table 2.1: List of notations used in this thesis.

2.3 Related Work

The previous sections gave an overview of currently existing approaches for 3D hand pose estimation from monocular images, which achieve *SotA* performance. In the following, we focus on the methods and techniques used for this thesis. The goal of this work is to utilize the approach of Rhodin *et al.* [64] and adapt it to work with hand data. They used an unsupervised geometry-aware representation for 3D human pose estimation using multi-view RGB images for training.

Rhodin *et al.* [64] implemented a method for 3D human pose estimation using multi-view images. Since human pose estimation, as well as hand pose estimation solutions mostly rely on *DNNs*, they require large amounts of labeled training data to work reliably. However, gathering a sufficient amount of labeled data for 3D pose estimation is laborious and costly. Rhodin *et al.* overcame this problem by learning a geometry-aware body representation from multi-view images without annotations. They utilized an encoder-decoder network to predict an image from one viewpoint given an image from another viewpoint. The idea of their approach was to learn a latent representation that captures the 3D geometry of the human body using images of the same person taken from multiple viewpoints. To learn this representation, they did not need any 2D or 3D pose annotations at all. Instead, they trained an encoder-decoder network to predict an image seen from one view from an image captured from a different view. Afterwards, they were able to learn to predict a 3D pose from the latent representation in a supervised way. The major advantage of this method is that the mapping to the 3D pose is much simpler because the latent representation already captures 3D geometry. Furthermore, this approach needs much fewer labeled data for training than existing methods that regress directly from an image to the 3D pose [64].

The latent representation represents a set of 3D points, which can be retrieved from a monocular view at test time. The latent 3D points can be used for Novel View Synthesis (NVS) as well. *NVS* [14] is the task of creating realistic images from previously unseen viewpoints [64]. Moreover, *NVS* is strongly tied to multi-view 3D reconstruction, since it requires transporting pixels across views through the geometry of scenes [93].

They designed the latent space in a way that it encodes 3D pose along with shape and appearance information and it can be learned without any 2D or 3D pose annotations. In order to do so, the authors used sequences of images acquired from multiple synchronized and calibrated cameras. The setup and acquisition process requires care, however, the amount of effort necessary is negligible compared to the effort needed to annotate tens of thousands of 2D or 3D poses. From these image sequences, they learned separate representations for the 3D pose, the appearance of the body, and of the background [64].

2.3.1 Autoencoders

Autoencoders have become standard tools to learn latent representations in unsupervised settings for individual images. An autoencoder is a particular neural network with the ability to encode an input into a compressed and meaningful representation, and then to decode it back, such that the reconstructed output is as similar as possible to the original input. The problem [5, 6] is to learn the functions $\mathcal{E} : \mathbb{R}^n \to \mathbb{R}^p$ (encoder) and $\mathcal{D} : \mathbb{R}^p \to \mathbb{R}^n$ (decoder) that satisfy

$$\min E(\mathcal{E},\mathcal{D}) = \min_{\mathcal{E},\mathcal{D}} \sum_{t=1}^{m} E(\mathbf{x}_t) = \min_{\mathcal{E},\mathcal{D}} \sum_{t=1}^{m} \Delta((\mathcal{D} \circ \mathcal{E})(\mathbf{x}_t), \mathbf{x}_t) \quad , \tag{2.1}$$

where:

1. n and p are positive integers (the case where $0 < p < n$ is primarily considered).
2. Δ is a dissimilarity or distortion function (e.g., Hamming distance, L_p-norm) defined over $\mathbb{R}^n$. The loss function measures the distance between the output of the decoder and the input.
3. $X = (\mathbf{x}_1, \ldots, \mathbf{x}_m)$ is a set of m (training) vectors in $\mathbb{R}^n$. When external targets are present, $Y = (\mathbf{y}_1, \ldots, \mathbf{y}_m)$ denotes the corresponding set of target vectors in $\mathbb{R}^n$.

Figure 2.11 shows an example of an autoencoder, where a handwritten digit is encoded to a latent representation and afterwards decoded.

In the case of a non auto-associative case, when external targets y_t are provided, the minimization problem becomes [5]:

$$\min E(\mathcal{E},\mathcal{D}) = \min_{\mathcal{E},\mathcal{D}} \sum_{t=1}^{m} E(\mathbf{x}_t, \mathbf{y}_t) = \min_{\mathcal{E},\mathcal{D}} \sum_{t=1}^{m} \Delta((\mathcal{D} \circ \mathcal{E})(\mathbf{x}_t), \mathbf{y}_t) \quad . \tag{2.2}$$

NVS methods, that rely on training encoder-decoder networks on multiple views of the same object are applied to leverage multi-view geometry. Images taken at the same time but from different viewpoints of an object, including their rotation matrix connecting the

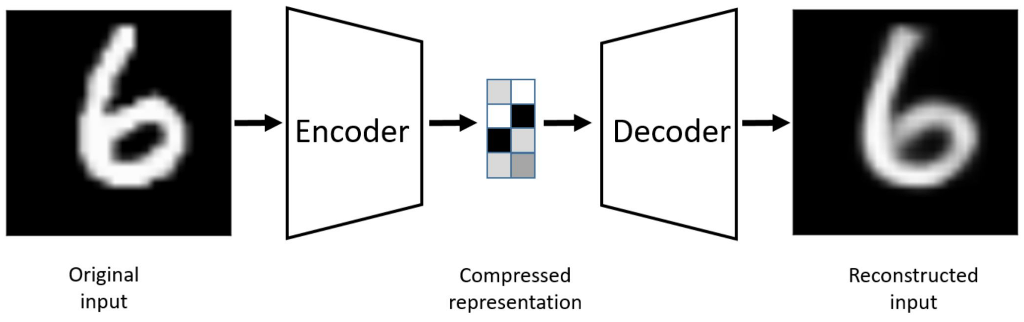


Figure 2.11: Example of an autoencoder. The input image is encoded to a compressed representation and then decoded. The decoded image should look as similar as possible to the input image [6].

two views can be fed to an encoder-decoder network and train it to encode the input image from the first view and decode it to depict the image from the second view. To enforce the latent representation to explicitly encode 3D information, Rhodin *et al.* [64] modeled the latent representation as a set of points in the 3D space. By doing so, they were able to model the view-change as a 3D rotation by matrix multiplication of the encoder output by the rotation matrix before using it as input to the decoder. This representation was effective as an intermediate for 3D pose estimation as well as for novel view synthesis.

3 Geometry-Aware 3D Hand Pose Estimation

Data is the new oil.

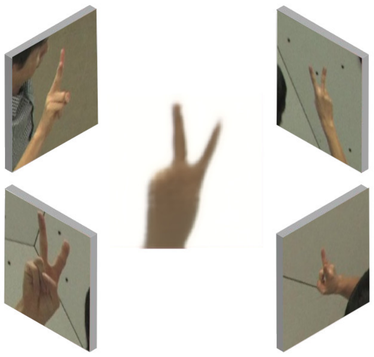
Clive Humby

Contents

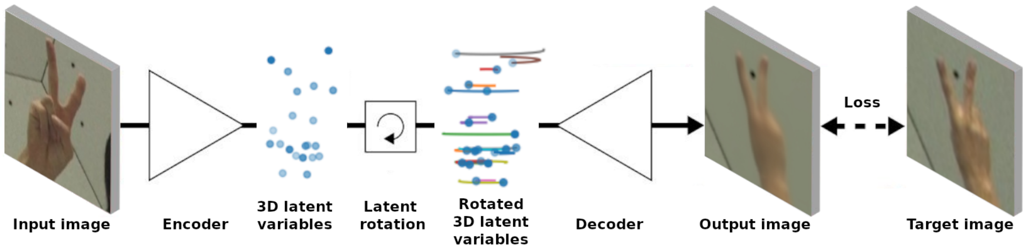
3.1 Our Approach

In this work, we use the idea of Rhodin *et al.* [64] and apply it on 3D hand pose estimation. Therefore, the following description is strongly based on the work of [64], however, adapted for hand pose estimation.

We use images of the same hand taken from multiple viewpoints to train an encoder-decoder network, which learns a latent representation, that captures the 3D geometry of the hand. We train the encoder-decoder to predict an image seen from one view from an image captured from a different one, without any 2D or 3D pose annotations, as shown in Fig. 3.1. Afterwards, we use the latent representation to learn a mapping to the 3D pose in a supervised manner. The crux of this method is, that the mapping to the 3D pose is much simpler, since the latent representation already captures 3D geometry. Furthermore, it requires considerably fewer examples for learning the mapping compared to many existing methods, that rely on multi-view supervision. Besides the approach of Rhodin *et al.*, we implement slightly different network configurations and compare their performance with one another. Figure 3.4 illustrates our realized network configurations next to an example of a common method, that attempts to regress directly from the image to the 3D pose (Fig. 3.4d).



(a) Unlabeled multi-view images.



(b) Unsupervised learning for our 3D hand pose estimation.

Figure 3.1: Unsupervised learning to obtain a pre-trained encoder for our 3D hand pose estimation pipeline (illustrations inspired by [64]). **(a)** We use unlabeled multi-view images to learn a geometry-aware representation. **(b)** Our encoder-decoder network is trained in an unsupervised manner. The colored lines in the rotated 3D latent variables indicate the trajectory of the 3D points. Due to the pre-training of the encoder, we only need a small amount of target supervision to learn a mapping from the latent representation to the actual 3D poses.

The latent representation $\mathbf{L}$ needs to encode the 3D pose along with shape and appearance, and can be learned without 2D or 3D pose annotations. To achieve this goal, we use sequences of images, acquired from multiple synchronized and calibrated cameras (details in Section 3.2). From these images, we learn separate representations of the hand's 3D pose and geometry, its appearance, and the background. Those will be referred to as $\mathbf{L}^{3\mathrm{D}}$, $\mathbf{L}^{\mathrm{app}}$, and $\mathbf{B}$, respectively.

Assuming we are given a set $\mathcal{U} = (\mathbf{I}_t^i, \mathbf{I}_t^j)_{t=1}^{N_u}$ of N_u image pairs without annotations, where i and j refer to the cameras used to capture the images, and t to the acquisition time. Let $\mathbf{R}^{i\to j}$ be the rotation matrix from the coordinate system of camera i to that of camera j. With this basis, we can turn to learning the individual components of $\mathbf{L}$.

To learn latent representations from individual images in unsupervised settings, we build upon autoencoders, which are a common choice for such tasks [62, 81, 82]. Let

$\mathcal{E}_{\theta_e}$ and $\mathcal{D}_{\theta_d}$ be the encoder and decoder respectively, where θ_e and θ_d are the weights controlling their behaviour. The encoder $\mathcal{E}_{\theta_e}$ is used to encode an image $\mathbf{I}$ into a latent representation $\mathbf{L} = \mathcal{E}_{\theta_e}(\mathbf{I})$, which is then decoded into the reconstructed image $\hat{\mathbf{I}} = \mathcal{D}_{\theta_d}(\mathbf{L})$. θ_e and θ_d are learned by minimizing the reconstruction error over the training set $\mathcal{U}$.

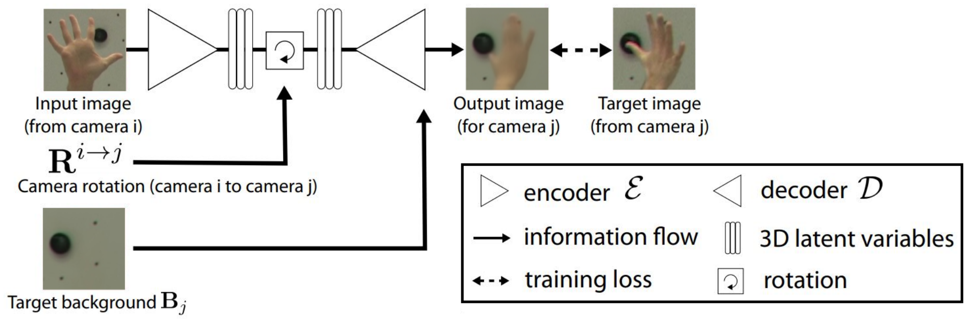


Figure 3.2: Geometry-aware 3D representation learning (illustration based on [64]). In our approach, we learn a latent representation, that encodes geometry and consequently 3D pose information in an unsupervised manner. This method extends a conventional autoencoder with a 3D latent space, rotation operation, and a background fusion module. We enforce explicit encoding of 3D information by rotating the 3D latent space. The background fusion enables applications to natural images.

Our method of using multi-view geometry is influenced by Novel View Synthesis (NVS) methods [17, 81, 82, 90], that rely on training encoder-decoder networks on multiple views of the same object. Similar to Rhodin *et al.* [64], we let $(\mathbf{I}_t^i, \mathbf{I}_t^j) \in \mathcal{U}$ be two images taken from different views but at the same time t. Since we know the rotation matrix $\mathbf{R}^{i\to j}$ connecting both viewpoints, we feed this information as an additional input to the encoder and decoder, and train them to encode $\mathbf{I}_t^i$ and resynthesize $\mathbf{I}_t^j$. By doing so, novel views of the corresponding object can be rendered by manipulating the rotation parameter $\mathbf{R}^{i\to j}$. In order to enforce an explicit encoding of 3D information within the latent variables, we model the latent representation $\mathbf{L}^{\text{3D}} \in \mathbb{R}^{3\times N}$ as a set of N points in 3D space. The encoder $\mathcal{E}_{\theta_e}$ and decoder $\mathcal{D}_{\theta_d}$ are designed in a way, that they have a three-channel output and input respectively. With this architecture, we can model the view change as a 3D rotation through matrix multiplication of the encoder output by the rotation matrix before it is used as input of the decoder. The resulting autoencoder $\mathcal{A}_{\theta_e,\theta_d}$ can be formally written as

$$\hat{\mathbf{I}}_t^j = \mathcal{A}_{\theta_e,\theta_d}(\mathbf{I}_t^i, \mathbf{R}^{i\to j}) = \mathcal{D}_{\theta_d}(\mathbf{R}^{i\to j}\mathbf{L}_{i,t}^{\text{3D}}) \ \text{, with} \quad \mathbf{L}_{i,t}^{\text{3D}} = \mathcal{E}_{\theta_e}(\mathbf{I}_t^i) \quad , \tag{3.1}$$

where $\hat{\mathbf{I}}_t^j$ is the reconstructed image from view j.

The weights θ_e and θ_d are optimized to minimize $\|\mathcal{A}_{\theta_e,\theta_d}(\mathbf{I}_t^i, \mathbf{R}^{i\to j}) - \mathbf{I}_t^j\|_1$ over the training set $\mathcal{U}$. The decoder $\mathcal{D}$ needs to learn how to decode the 3D latent vector $\mathbf{L}^{\text{3D}}$ without knowing how to rotate the input to a novel view. Therefore, the encoder $\mathcal{E}$ is forced to map to a proper 3D latent space, which still can be decoded by $\mathcal{D}$ after an arbitrary rotation. At this point, $\mathbf{L}^{\text{3D}}$ not only encodes the 3D geometry, but also the background and the appearance of the hand. We now need to separate this information from each other and create two new vectors $\mathbf{L}^{\text{app}}$ and $\mathbf{B}$ for appearance and background, so that $\mathbf{L}^{\text{3D}}$ only describe geometry and pose.

Now, we assume that we have the background $\mathbf{B}_j$ of all images from a given viewpoint j. We describe the details to construct the background for our dataset in Section 3.2. To factor out the background, we modify the decoder $\mathcal{D}$ by adding a direct connection to the target background $\mathbf{B}_j$, as shown in Fig. 3.2. We concatenate the background image with the output of the decoder and use an additional 1×1 convolutional layer to synthesize the decoded image. This extension prevents the rest of the network to learn about the background and it enforces that the $\mathbf{L}^{\text{3D}}$ vector does not contain information about it anymore.

In order to split the appearance from geometry in the latent representation, we split the output of the encoder $\mathcal{E}$ into two different vectors $\mathbf{L}^{\text{3D}}$ and $\mathbf{L}^{\text{app}}$, that describe pose and appearance respectively. We enforce this separation by simultaneously training the encoder-decoder network on two frames $\mathbf{I}_t$ and $\mathbf{I}_{t'}$ which depict the same subject but at different times t and t', as shown in Fig. 3.3. The decoder $\mathcal{D}$ uses $\mathbf{L}^{\text{3D}}_t$ and $\mathbf{L}^{\text{3D}}_{t'}$ as before, but it swaps $\mathbf{L}^{\text{app}}_t$ and $\mathbf{L}^{\text{app}}_{t'}$. Consequently, the decoder uses $\mathbf{L}^{\text{3D}}_t$ and $\mathbf{L}^{\text{app}}_{t'}$ to resynthesize the frame at time t and $\mathbf{L}^{\text{3D}}_{t'}$ and $\mathbf{L}^{\text{app}}_t$ for frame t'. We assume, that the appearance of the hands does not change drastically between frames t and t' and that the differences in the images are caused by 3D pose changes. This enforces $\mathbf{L}^{\text{3D}}$ encoding the pose and $\mathbf{L}^{\text{app}}_t$ encoding the appearance.

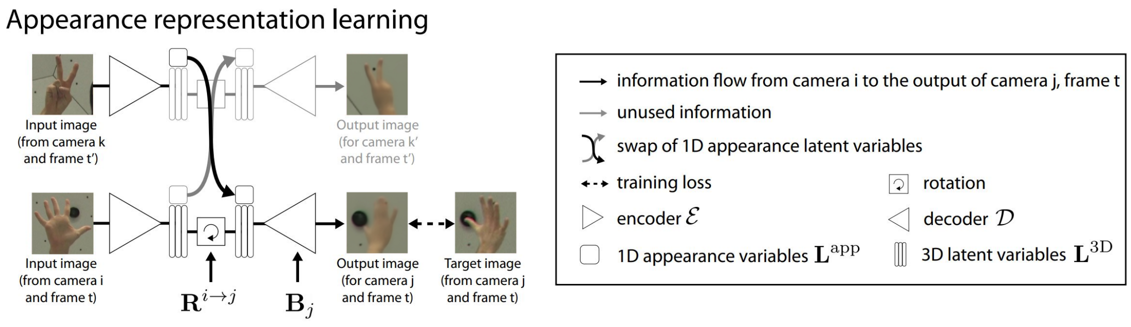

Appearance representation learning
Input image (from camera k and frame t')
Output image (for camera k' and frame t')
Input image (from camera i and frame t)
$\mathbf{R}^{i\to j}$
$\mathbf{B}_j$
Output image (for camera j and frame t)
Target image (from camera j and frame t)
information flow from camera i to the output of camera j, frame t
unused information
swap of 1D appearance latent variables
training loss
rotation
encoder $\mathcal{E}$
decoder $\mathcal{D}$
1D appearance variables $\mathbf{L}^{\text{app}}$
3D latent variables $\mathbf{L}^{\text{3D}}$

Figure 3.3: Appearance representation learning (illustration inspired from [64]). We split the latent space into a 3D geometry part and an appearance part, in order to encode the subject identity. The appearance part is not rotated, but swapped between two frames t and t' depicting the same subject to enforce it not to contain any geometric information.

Thus, the encoder $\mathcal{E}$ has two outputs, which is $\mathcal{E}_{\theta_e} : \mathbf{I}^i_t \to (\mathbf{L}^{\text{3D}}_{i,t}, \mathbf{L}^{\text{app}}_{i,t})$. The decoder $\mathcal{D}$ accepts those two and the background as input after rotating the geometric representation and swapping the appearance for two views i and j. The output of the encoder-decoder can be written as

$$\mathcal{A}_{\theta_e,\theta_d}(\mathbf{I}^i_t, \mathbf{R}^{i\to j}, \mathbf{L}^{\text{app}}_{k,t'}, \mathbf{B}_j) = \mathcal{D}_{\theta_d}(\mathbf{R}^{i\to j}\mathbf{L}^{\text{3D}}_{i,t}, \mathbf{L}^{\text{app}}_{k,t'}, \mathbf{B}_j) \quad . \tag{3.2}$$

Since we want to train $\mathcal{A}$ with multiple sequences that feature different people and backgrounds, we randomly select mini-batches of Z triplets $(\mathbf{I}_t^i, \mathbf{I}_t^j, \mathbf{I}_{t'}^k)$ in $\mathcal{U}$ with $t \neq t'$ from individual sequences. The first two views are taken at the same time but from different viewpoints and the third one is taken at a different time, but from an arbitrary viewpoint k. The loss function used to train the autoencoder parameters θ_e, θ_d is based on the reconstruction loss:

$$E_{unsuperv.} = E_{\theta_e,\theta_d} + \lambda_{feat} E_{feat} \quad , \tag{3.3}$$

with

$$E_{\theta_e,\theta_d} = \frac{1}{Z} \sum_{\substack{\mathbf{I}_t^i, \mathbf{I}_t^j, \mathbf{I}_{t'}^k \in \mathcal{U} \\ t \neq t'}} |\mathcal{A}_{\theta_e,\theta_d}(\mathbf{I}_t^i,\, \mathbf{R}^{i \to j},\, \mathbf{L}_{k,t'}^{\text{app}},\, \mathbf{B}_j) - \mathbf{I}_t^j| \quad , \tag{3.4}$$

and

$$E_{feat} = \frac{1}{Z} \sum_{\substack{\mathbf{I}_t^i, \mathbf{I}_t^j, \mathbf{I}_{t'}^k \in \mathcal{U} \\ t \neq t'}} |\mathcal{R}_{18}(\mathcal{A}_{\theta_e,\theta_d}(\mathbf{I}_t^i,\, \mathbf{R}^{i \to j},\, \mathbf{L}_{k,t'}^{\text{app}},\, \mathbf{B}_j)) - \mathcal{R}_{18}(\mathbf{I}_t^j)| \quad . \tag{3.5}$$

$\mathbf{L}_{k,t'} = (\mathbf{L}_{k,t'}^{\text{3D}}, \mathbf{L}_{k,t'}^{\text{app}})$ is the output of the encoder $\mathcal{E}_{\theta_e}$ applied to the image $\mathbf{I}_{t'}^k$. $\mathbf{B}_j$ is the background in view j and $\mathbf{R}^{i \to j}$ stands for the rotation matrix from view i to view j. The encoder $\mathcal{E}$ is applied twice, in order to get $\mathbf{L}_{i,t}^{\text{3D}}$ and $\mathbf{L}_{k,t'}^{\text{app}}$, while ignoring $\mathbf{L}_{i,t}^{\text{app}}$ and $\mathbf{L}_{k,t'}^{\text{3D}}$ with the swap, as explained before.

During training, we minimize a loss $E_{unsuperv.}$ that is the sum of the pixel-wise error E_{θ_e,θ_d} (Eq. (3.4)) and a second term E_{feat} (Eq. (3.5)), which is obtained by first applying a Residual Neural Network (ResNet) [35] $\mathcal{R}_{18}$ with 18 layers trained on ImageNet [66] on the output and target image, and then computing the absolute feature difference after the second block level. This additional term enhances the decodings and improves the pose reconstruction. Following [64], we average the individual pixel and feature differences, and balance their influence by weighting the feature loss by two ($\lambda_{feat} = 2.0$).

Up until now, we know how to learn an encoding for the multi-view geometry using an encoder-decoder network. Our goal, however, is to infer the 3D pose of a human hand from a monocular image. $\mathbf{L}^{\text{3D}}$ is a $3 \times N$ matrix and we already use it to generate novel views. $\mathbf{L}^{\text{3D}}$ can also be understood as a set of N 3D points, but they do not have any semantic meaning yet. In almost all practical applications, one wants to derive a pre-defined representation, like a skeleton with $K = 21$ major hand joints (as depicted in Fig. 2.9a), encoded as a vector $\mathbf{P} \in \mathbb{R}^{3K}$.

To accomplish such a representation, we want a mapping $\mathcal{F} : \mathbf{L}^{\text{3D}} \to \mathbb{R}^{3K}$. This mapping function can be interpreted as a different decoder that reconstructs 3D poses instead of images. Learning a mapping from $\mathbf{L}^{\text{3D}}$ to the 3D hand joints requires a much smaller amount of annotated data (as we will see in Chapter 4), than what would be needed for learning a mapping directly from the images, as in many other approaches to hand pose estimation.

Let $\mathcal{L} = \{(\mathbf{I}_t, \mathbf{P}_t)\}_{t=1}^{N_s}$ be a small set of N_s labeled examples consisting of image pairs and corresponding ground-truth 3D poses $\mathbf{P}$. $\mathcal{F}$ is modeled as a Deep Neural Network (DNN) with parameters θ_f, and it is trained by minimizing the sum of the objective function F_{θ_f} (Eq. (3.7)) and a 3D geometric induced loss function L_{geo} (Eq. (3.8)):

$$L_{3D} = \lambda_F F_{\theta_f} + \lambda_{geo} L_{geo} \quad , \tag{3.6}$$

where

$$F_{\theta_f} = \frac{1}{N_s} \sum_{t=1}^{N_s} \|\mathcal{F}_{\theta_f}(\mathbf{L}_t^{3D}) - \mathbf{P}_t\| \quad , \tag{3.7}$$

with $(\mathbf{L}_t^{3D}, \cdot) = \mathcal{E}_{\theta_e}(\mathbf{I}_t)$, since the encoder outputs both $\mathbf{L}^{3D}$ and $\mathbf{L}^{app}$.

Since the latent variable $\mathbf{L}^{3\mathrm{D}}$ already encodes the 3D hand pose and shape, $\mathcal{F}$ can be implemented as a simple fully-connected neural network. The encoder-decoder network (trained in an unsupervised manner) introduced before and the fully-connected network (trained in a supervised manner) combined form the semi-supervised setup, as illustrated in Fig. 3.4a. The unsupervised representation already does a lot of the hard work in the challenging task of lifting the image to a 3D representation, that simplifies the final mapping.

Unlike Rhodin *et al.* [64], we try to further improve the accuracy of the 3D pose reconstruction, by implementing a 3D geometric constraint induced loss L_{geo} and combine it with F_{θ_f} from Eq. (3.7). We calculate the geometric loss in a similar way as it is described by Zhou *et al.* [98]. They applied this loss on their 3D human pose estimation pipeline and it is based on the fact, that ratios between bone lengths remain relatively fixed in a human skeleton. We implement a similar function, since for the human hand, there also exists strong evidence of the existence of some constant ratio between finger segments [10].

Let $R_i = \{dp_i, mp_i, pp_i, mc_i\}$ be a set of bones within a skeleton group $i = \{\mathrm{Thumb}, \mathrm{Index}, \mathrm{Middle}, \mathrm{Ring}, \mathrm{Pinky}\}$ (see Fig. 1.1b for reference). Let l_e be the length of bone e obtained from the predicted keypoints $\mathcal{F}_{\theta_f}(\mathbf{L}_t^{3D})$, and $\bar{l}_e$ denotes the length of bone e in a canonical hand skeleton. We derive the canonical skeleton by calculating the average of each finger segment of the hands in the training set. The length of each bone is calculated using the Euclidean distance between the corresponding keypoints. The ratio $l_e/\bar{l}_e$ for each bone e in each group R_i should be the same and remain fixed for each individual subject. The geometric loss measures the sum of variance among $\{l_e/\bar{l}_e\}e \in R_i$ of each R_i:

$$L_{geo} = \sum_i \frac{1}{|R_i|} \sum_{e \in R_i} \left(\frac{l_e}{\bar{l}_e} - \bar{r}_i\right)^2 \quad , \tag{3.8}$$

where

$$\bar{r}_i = \frac{1}{|R_i|} \sum_{e \in R_i} \frac{l_e}{\bar{l}_e} \quad . \tag{3.9}$$

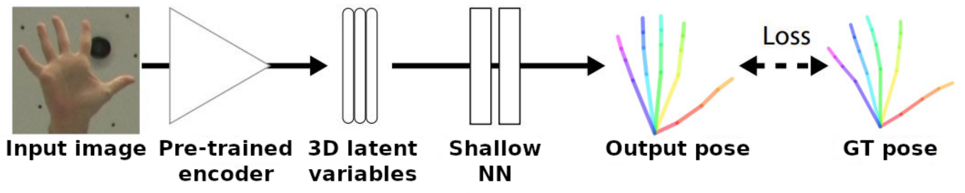
λ_F and λ_{geo} in the final loss function L_{3D} from Equation (3.6) are the corresponding weighting factors.

In contrast to Rhodin *et al.* [64], we additionally do a joint training of the encoder-decoder network $\mathcal{A}_{\theta_e,\theta_d}$ and the fully-connected neural network $\mathcal{F}_{\theta_f}$ by minimizing the joint loss

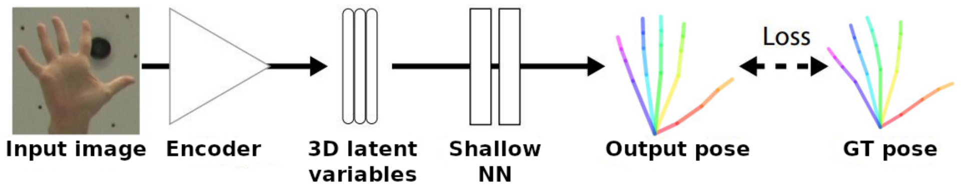
$$L_{semi-superv.} = E_{unsuperv.} + L_{3D} \quad . \tag{3.10}$$

All of our implemented hand pose estimation approaches are illustrated in Figure 3.4.

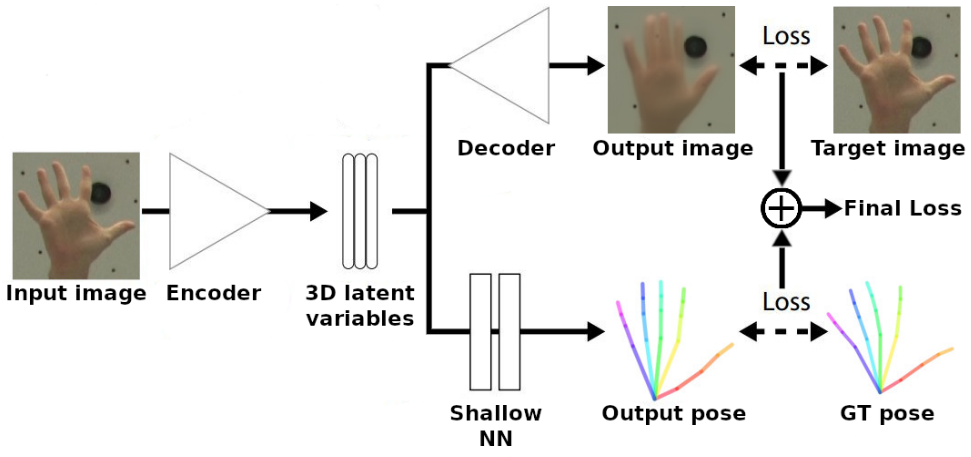
Hitherto, we explained our approach for 3D hand pose estimation, however, we still need to collect and prepare our data we use for training and testing our method. This will be explained in the following sections.



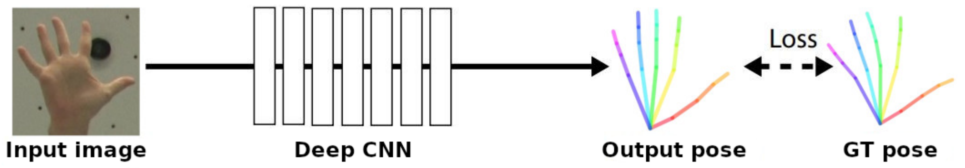
(a) Pipeline using a pre-trained encoder to recover 3D hand poses, as proposed by Rhodin *et al.* [64]



(b) Network, that directly maps an input image to the 3D pose, with an untrained encoder.



(c) Fully semi-supervised network, where the encoder-decoder network and the pose network are trained simultaneously.



(d) Illustration of common *SotA* pipelines for 3D hand pose estimation.

Figure 3.4: Comparison of our three implemented approaches **(a)**-**(c)** to a common pipeline, that *SotA* methods use for 3D hand pose estimation (illustrations inspired by Rhodin *et al.* [64]). **(a)** To recover the position of the 3D joints of a hand from a monocular image, we compute the latent representation of the input image and feed it to the shallow network to compute the pose. **(b)** This network maps an input image directly to the 3D pose, without the need of a pre-trained encoder. **(c)** The fully semi-supervised network trains the encoder-decoder network and the shallow pose network simultaneously. This network predicts both, the novel view and the 3D pose of the input image. **(d)** In contrast, most *SotA* methods which train a network to regress directly from the input image to the 3D pose require a much deeper network and therefore a lot more training data.

3.2 Dataset

The dataset we use in this thesis comes from the Carnegie Mellon University (CMU) Panoptic Dataset provided by Joo *et al.* [38]. Their goal was to capture the 3D motion of a group of people engaged in natural human interactions. They built a massive multi-view system with heterogeneous sensors including 480 Video Graphics Array (VGA) cameras, 31 High Definition (HD) cameras and 10 Kinects™ . The vast number of cameras placed at different viewpoints provide robustness against occlusion and the subjects are not restricted in terms of their viewing direction. The uniform arrangement of the cameras enabled them to observe scenes from all directions without restricting the motion of the subjects by a predefined dominant system direction. The *HD* views provide the details for the scene, and the Kinects™ produce initial point clouds to generate a dense trajectory stream.

They designed the physical frame of the Panoptic Studio in the way of a face-transitive solid, called a truncated pentagonal hexecontahedron. The transitivity of the faces allows for a modular architecture and ensures that the structure remains easy to upgrade and customize with different panels. The structure of the dome is shown in Figure 3.5. It has a diameter of 5.49 m and a total height of 4.15 m. The floor of the dome is 1.40 m below the center to increase access to the edges. The geodesic dome structure consists of 6 pentagonal panels, 40 hexagonal panels, and 10 trimmed base panels. 20 of the hexagonal panels house a set of 24 *VGA* cameras each. The 31 *HD* cameras are installed at the center of the corresponding hexagonal panels, 5 projectors are installed at the center of each pentagonal panel, and 10 Kinect™ sensors are placed to form two rings with 5 evenly spaced sensors. Each of the *VGA* cameras has a resolution of 640×480 and each *HD* camera has a resolution of 1920×1080. Both camera types capture at a frame rate of approximately 30 Frames per Second (FPS). In total, their system consists of 521 cameras, which are calibrated using Structure from Motion (SfM).

The team made all the data available on the *CMU* Panoptic Dataset website[1], including all synchronized camera feeds, calibration, 3D pose reconstruction results, and 3D trajectory streams. The reconstructed 3D hand poses have $K = 21$ keypoints, as illustrated in Fig. 2.9a. For our purpose, we use the 31 *HD* camera feeds (because of the higher resolution), calibration data, and the 3D hand pose reconstructions as ground truth. For training our network, we employ the sequences *171026_pose3* and *171204_pose2*, and for testing the sequence *171026_pose2*. In total, we have around 24 minutes of video material for training (48 683 frames per camera) and approximately 9 minutes for testing (16 366 frames per camera) from 31 different *HD* views. The video sequences feature 9 different subjects for training and 2 subjects for testing, who separately step into the dome and perform different body and hand movements.

[1]`http://domedb.perception.cs.cmu.edu/`

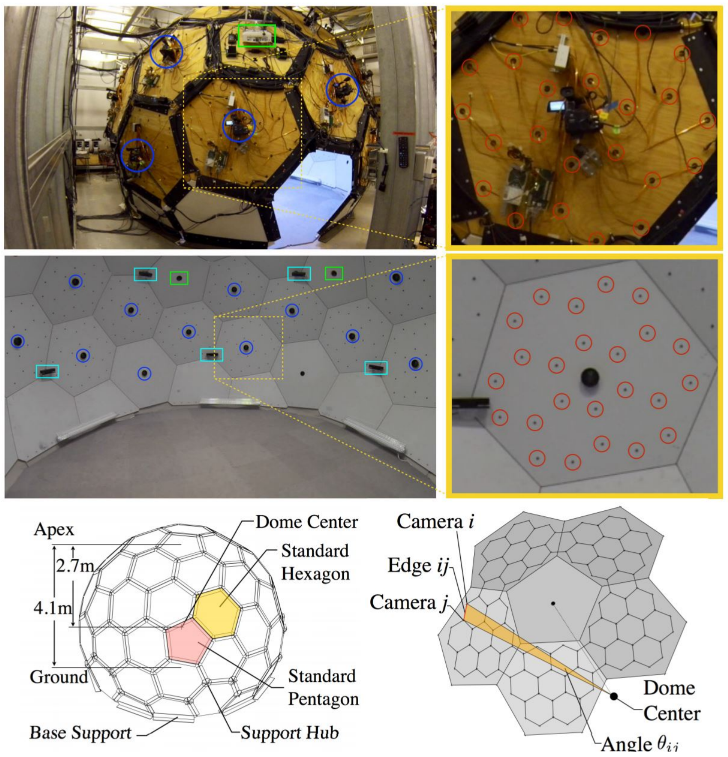


Figure 3.5: Structure of the Panoptic Studio (image taken from [38]). **(Top Row)** Shows the exterior of the geodesic dome with the equipment mounted on the surface. **(Middle Row)** The interior of the dome, where the red circles show the *VGA* cameras, the blue circles mark the *HD* cameras, the cyan rectangles indicate the Kinects™ and the green rectangles signify the projectors. **(Bottom Left)** The panels are designed to ensure interchangeability. **(Bottom Right)** The optimized camera positions ensure uniform angles with respect to the dome center between each camera and all its neighbors (e.g., Camera i is a neighbor of Camera j).

3.3 Cropping, Augmentation and Background Estimation

For human pose estimation, as well as for hand pose estimation algorithms, it is common practice to first crop the subject of interest to factor out scale and global position, which are inherently ambiguous for monocular reconstruction and Novel View Synthesis (NVS). To approximate hand detection, we derive the crop information by utilizing the predicted 3D joint locations from Joo *et al.* [38]. For each frame, we project the 3D keypoints into each 2D camera plane and calculate a bounding box around the 2D keypoints farthest away from each other and add a border with a width of 10 pixel to every edge, as illustrated in Fig. 3.6. After cropping, we resize all image patches to the same size of 128 × 128. Now we have to compute the rotation between two views with respect to the hand center instead of the image center and shear the cropped image so that it appears as if it was taken from a virtual camera pointing in the crop direction. To calculate the center of a hand, we construct a simple 3D bounding box around a hand pose $\mathbf{P}_t$ at time t and calculate the geometric center $\mathbf{C}_{\mathbf{P}_t}$ of this box:

$$\mathbf{C}_{\mathbf{P}_t}(\mathbf{P}_t) = \min(\mathbf{P}_t) + \frac{\max(\mathbf{P}_t) - \min(\mathbf{P}_t)}{2} \quad . \tag{3.11}$$

To calculate the new rotation matrix that points to $\mathbf{C}_{\mathbf{P}_t}$, we followed the idea of the *gluLookAt*[2] function from OpenGL® . Let $\mathbf{C}_{\mathbf{P}_t}$ be the target point, $\mathbf{U}$ be the up-vector of the camera and $\mathbf{C}_c = -\mathbf{R}_c^{\mathrm{T}}\mathbf{T}_c$ be the camera center, where $\mathbf{R}_c$ and $\mathbf{T}_c$ are the rotation matrix and translation vector from the given calibration data. The new rotation matrix $\mathbf{R}$ is given by

$$\mathbf{R} = \begin{bmatrix} \mathbf{S} \\ \mathbf{U}' \\ \mathbf{L} \end{bmatrix} \quad , \tag{3.12}$$

with

$$\mathbf{L} = \frac{\mathbf{C}_{\mathbf{P}_t} - \mathbf{C}_c}{\|\mathbf{C}_{\mathbf{P}_t} - \mathbf{C}_c\|} \quad , \tag{3.13}$$

$$\mathbf{S} = \frac{\mathbf{U} \times \mathbf{L}}{\|\mathbf{U} \times \mathbf{L}\|} \quad , \tag{3.14}$$

and

$$\mathbf{U}' = \frac{\mathbf{L} \times \mathbf{S}}{\|\mathbf{L} \times \mathbf{S}\|} \quad . \tag{3.15}$$

The new translation vector is calculated as $\mathbf{T} = -\mathbf{R}\mathbf{C}_c$.

Furthermore, for some experiments we apply random in-plane rotations to increase the diversity of the training set. Let α be a random angle by which the image should be rotated around the z-axis of a right-handed Cartesian coordinate system. The rotation matrix $\mathbf{R}_z$ is constructed as

$$\mathbf{R}_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad 1 \le \alpha < 360 \quad . \tag{3.16}$$

The new extrinsic rotation matrix $\mathbf{R}^{aug}$ and the new augmented hand keypoints $\mathbf{P}_t^{aug}$ can be simply calculated as

$$\mathbf{R}^{aug} = \mathbf{R}_z\mathbf{R} \tag{3.17}$$

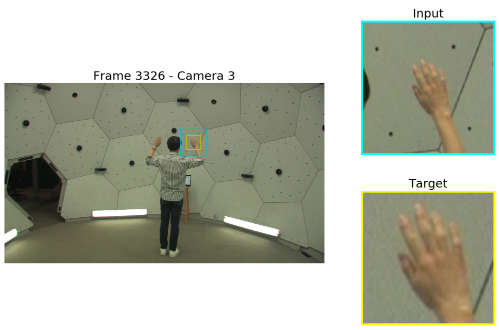
and

$$\mathbf{P}_t^{aug} = \mathbf{R}^{aug}\mathbf{R}^{\mathrm{T}}\mathbf{P}_t \quad . \tag{3.18}$$

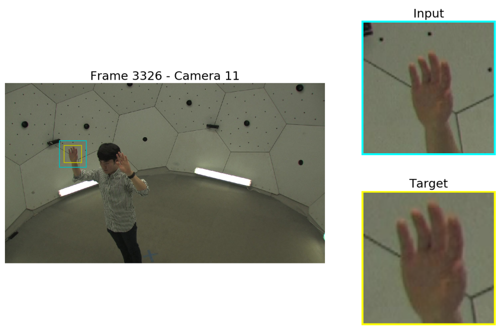
The rotation $\mathbf{R}^{i \to j}$ and background crop $\mathbf{B}_j$ depend on time t, however, we neglect this dependency in our notation for readability. We do the cropping and augmentation process for the left as well as for the right hands. An image from the same hand captured from 20 different camera viewpoints at the same time t can be seen in Figure 3.8.

[2] `https://www.khronos.org/registry/OpenGL-Refpages/gl2.1/xhtml/gluLookAt.xml`

For background estimation, we compute the median for each pixel over a specific set of frames. We select the frames in a way, that the full body of the actor, except the hands, is visible for more than half the time at the same spatial location. For hand pose estimation it is especially difficult, since e.g., we want the subject's upper body to be part of the background if the hand is located between the camera and the upper body. Therefore, we define different sets of frames from a sequence, where hand movement is present but only slight movement in the rest of the body. Then we calculate the pixel-wise median over each set individually. Figure 3.7 shows two background images derived from the same set of frames but from different views and the corresponding background patches to the hand crops from Figure 3.6. Because of occasional heavy body movement, we went through the training data by hand, in order to remove samples where the background patch does not fit the hand patch at all.

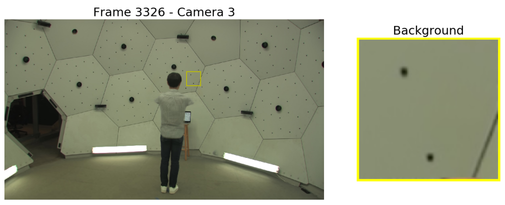


(a) Hand cropping from one of our training samples.

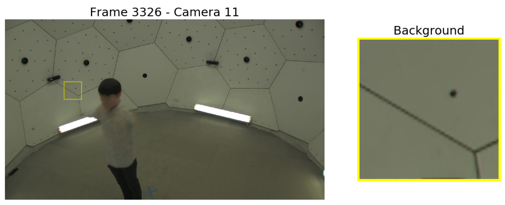


(b) Hand cropping from one of our training samples from a different camera.

Figure 3.6: Hand cropping from one of our training samples from different camera views but at the same time t. The crop with the cyan border shows the input patch for our network. Our input images are cropped with a fixed pixel size, independent of the distance from the hand to the camera. Therefore, the size of the hand varies with its distance to the camera. The crop with the yellow border shows the target patch of our network. These crops are tighter around the hand, so that the network can generate the predicted hands with roughly the same size, assuming that all hands have the same size. Note that during training, the target patch is from a different camera view than the input patch, e.g., the target patch for the input in **(a)** would be the target image from **(b)**.



(a) Background image from one of our training samples.



(b) Background image from one of our training samples but from a different viewpoint.

Figure 3.7: Background from one of our training samples from different camera viewpoints but at the same time t. The yellow borders indicates the background crops for the corresponding target hand crops in Fig. 3.6.

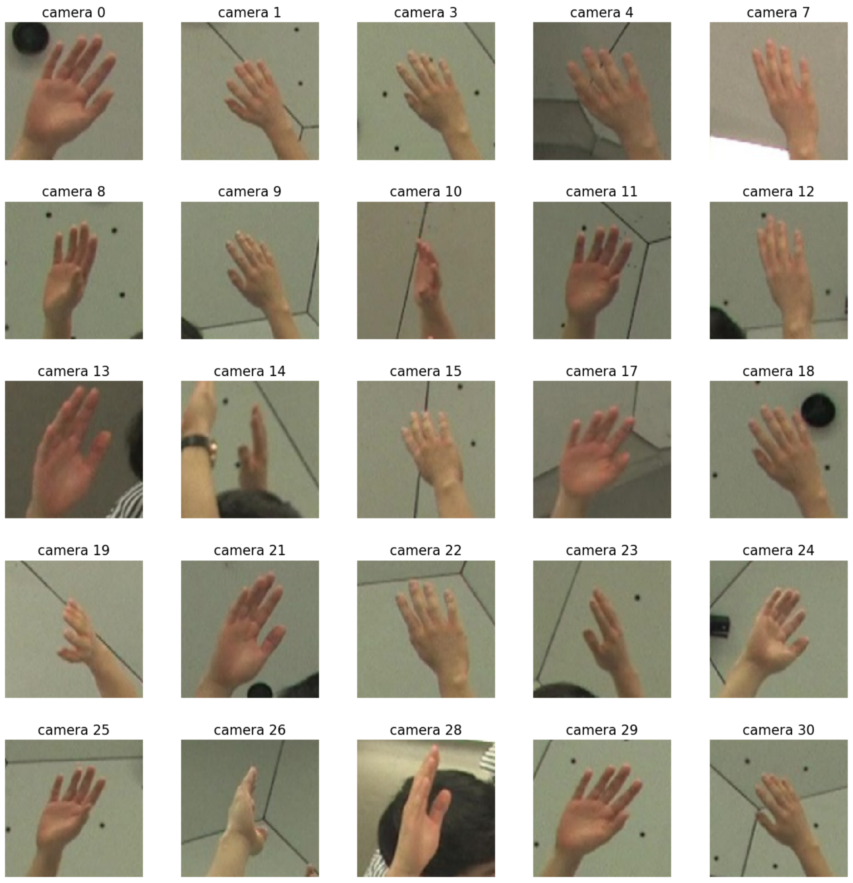


Figure 3.8: Set of hand patches from 20 different camera views captured at the same time t.

3.4 Clustering

Upon visual inspection, we find out that the majority of images in our training data depict hands hanging downwards. To prevent a bias of the network towards those poses, we introduce a clustering method to provide a more balanced training dataset, as well as to have a balanced amount of left and right hands. We use Hierarchical Agglomerative Clustering (HAC) [95] which is a bottom-up algorithm, that treats each 3D hand pose as a singleton cluster at the out-set and successively merges (or *agglomerates*) pairs of clusters until all clusters have been merged into a single cluster that contains all hand poses. A dendrogram is usually used to visualize a *HAC* result, as shown in Figure 3.9. Each merge of clusters is represented by a horizontal line and the y-coordinate of this line is the similarity of the two clusters that were merged. A dendrogram allows us to reconstruct the history of merges, that resulted in the depicted clustering by moving up from the bottom layer to the top node [55].

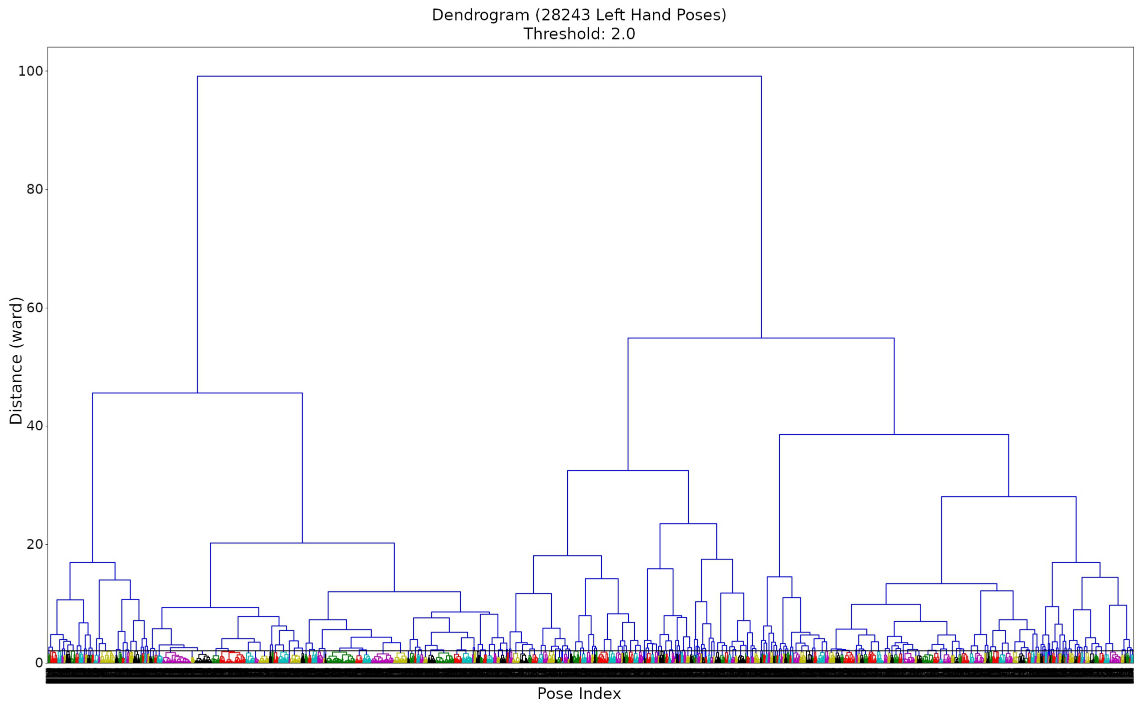


Figure 3.9: *HAC* of 28 243 3D (left) hand poses visualized as dendrogram. The threshold is set to 2.0 which results in 260 different clusters.

Hierarchical clustering does not require a predefined number of clusters. In our case, we cut the dendrogram at a threshold of 2.0, since we want a minimum combination similarity of 2.0. We tried different numbers for the threshold, but setting it to 2.0 gave us the best clustering of hand poses. As shown in Figure 3.9, cutting the diagram at $y = 2.0$ yields to 260 clusters for the 3D hand poses of left hands. For the right hands, we also cut the dendrogram at $y = 2.0$ and get 255 clusters, which results in a total of 515 different clusters.

We calculate the initial distances between 3D hand poses $\mathbf{P}$ by the mean Euclidean distance between the corresponding 3D joints. All poses $\mathbf{P}$ are centered by their hand center $\mathbf{C}_{\mathbf{P}_t}$ from Equation (3.11) and normalized by their mean and standard deviation. The distance calculation can be formulated as

$$d_{3D}(\mathbf{P}_i, \mathbf{P}_j) = \frac{1}{K} \sum_{k=0}^{K} \|\mathbf{C}_k^i - \mathbf{C}_k^j\| \quad , \tag{3.19}$$

where $\mathbf{C}_k^i$ is the 3D coordinate of the k^{th} joint in pose $\mathbf{P}_i$. The initial distances are stored in a condensed 1D matrix, which is used to form clusters.

We perform Ward's [24, 88] linkage method on the condensed distance matrix. The method is used to measure the distance between the clusters in order to create the tree structure. The Ward linkage method, also called *minimum variance clustering*, defines the distance between two clusters u and v by the increase of the sum of squares if they are merged. The sum of squares will initially be zero with hierarchical clustering algorithms and grows as clusters are merged. Ward's method tries to keep this growth as small as possible [55, 91].

The distance $d(u, v)$ between two clusters u and v is calculated as [70]

$$d(u, v) = \sqrt{\frac{|v| + |s|}{T} d(v, s)^2 + \frac{|v| + |t|}{T} d(v, t)^2 - \frac{|v|}{T} d(s, t)^2} \quad , \tag{3.20}$$

where u is the newly joined cluster consisting of clusters s and t, v is an unused cluster in the forest, $T = |v| + |s| + |t|$, and $| * |$ is the cardinality of its argument. This is also known as the incremental algorithm.

4 Experiments

If you don't find the time, if you don't do the work, you don't get the results.

Arnold Schwarzenegger

Contents

4.1 Overview

In this chapter, we evaluate our approach on 3D hand pose estimation and show, that our approach allows us to use far less annotated training data than other State-of-the-Art (SotA) methods to achieve better accuracy. Since we have ground-truth 3D data for all of our hand patches, we can easily compare different levels of supervision, unsupervised, semi-supervised, or fully supervised. Furthermore, we qualitatively evaluate the latent space itself to investigate how well it encodes geometry and appearance of the hand.

4.2 Dataset Splits and Metrics

To learn our latent representation, we take the unlabeled set $\mathcal{U}$ from the Panoptic dataset. Within the complete dataset, we distinguish between the different actors in the video

Scenario	**Number of Annotations**
Fully supervised training with the 3D annotations of all nine training subjects	28 592
S0 + S3 + S4 + S5 + S6	18 624
S0 + S3 + S4	13 136
S0 + S3	9 392
S0	6 512
50% of S0	3 248
10% of S0	640
5% of S0	320
1% of S0	64

Table 4.1: List of scenarios, that provide different levels of supervision evaluated in this thesis. The numbers of annotation are all multiples of 16, since we use mini-batches of size 16 to train our pose network $\mathcal{F}$.

streams and refer to them as S0, S3, S4, S5, S6, S7, S8, S9, S10, where SN specifies all sequences of the N^{th} subject, however, without the available 3D labels. To provide the required supervision to train the shallow network $\mathcal{F}$ depicted in Figure 3.4a, we define the scenarios listed in Table 4.1.

Furthermore, we compare the following network configurations:

1) Network $\mathcal{M}$ is the proposed hand pose estimation network from Figure 3.4a using a pre-trained encoder.

2) Network $\mathcal{N}$, as illustrated in Figure 3.4b, directly maps an input image to the 3D pose, without pre-training the encoder with unlabeled multi-view images. The parameters in the encoder and the pose network are optimized simultaneously.

3) Network $\mathcal{O}$ is a semi-supervised approach, where the encoder-decoder network and the pose network are trained simultaneously, as depicted in Figure 3.4c.

In all cases we use S1 and S2 for testing.

Since we have 515 clusters, with some clusters having more than 500 poses, we randomly pick at most 260 hand poses per cluster for the unsupervised multi-view training of the encoder-decoder. A threshold of 260 poses per cluster seemed to be very plausible to create a balanced dataset, without having poses being over- or underrepresented.

We train our pose network $\mathcal{F}$ with different λ-settings in the loss function (recall Eq. (3.6)). We always set $\lambda_F = 1.0$ and λ_{geo} is set to 10.0, 1.0, 0.1, and 0.01.

We evaluate pose predictions in terms of the Mean Per Joint Position Error (MPJPE) [37], and its normalized variant Normalized Mean Per Joint Position Error (N-MPJPE). The *MPJPE* in Equation (4.1) is calculated as the average differences in the $K = 21$ 3D joint positions $\hat{\mathbf{P}}$, that were reconstructed and the ground truth joint positions $\mathbf{P}^{gt}$. Before the calculation, $\hat{\mathbf{P}}$ is centered to the point $\mathbf{C}_{\mathbf{P}^{gt}}$ derived from Equation (3.11). For the *N-MPJPE*, the scale of the poses is normalized to the ground truth before computing the *MPJPE*.

$$\text{MPJPE} = \frac{1}{K}\sum_{k=1}^{K} \|\hat{\mathbf{P}}_k - \mathbf{P}_k^{gt}\| \tag{4.1}$$

The *N-MPJPE* is computed as

$$\text{N-MPJPE} = \frac{1}{K}\sum_{k=1}^{K} \|s \cdot \hat{\mathbf{P}}_k - \mathbf{P}_k^{gt}\| \quad , \tag{4.2}$$

with the scaling factor

$$s = \frac{\langle \hat{\mathbf{P}}, \mathbf{P}^{gt} \rangle}{\langle \hat{\mathbf{P}}, \hat{\mathbf{P}} \rangle} \quad . \tag{4.3}$$

Furthermore, we use the Percentage of Correct Keypoints (PCK) score to evaluate our hand pose estimation accuracy. Before we calculate the *PCK*, the scale of the poses is normalized. The *PCK* defines a certain keypoint to be correct, if it falls within a sphere of a given radius around the ground truth. The smaller the radius, the stricter the criterion and the more accurate the estimated hand joints are considered correct. Additionally, we report the Area Under the Curve (AUC) of the corresponding *PCK* in a 0 mm to 100 mm range. To calculate the *AUC*, we use the trapezoidal rule [18] to approximate the integral.

4.3 Implementation

We implemented our approach in PyTorch and used a Nvidia® GeForce® RTX 2070 graphics card and an Intel® Core™i7 -8700K CPU to train and test our networks. Our encoder $\mathcal{E}$ is an off-the shelf Residual Neural Network (ResNet) with 50 layers [35]. An input image with the resolution of 128×128 gets mapped to $\mathbf{L}^{app} \in \mathbb{R}^{128}$ and $\mathbf{L}^{3D} \in \mathbb{R}^{\ell \times 3}$, where we evaluate different settings of ℓ.

The decoder $\mathcal{D}$ maps $\mathbf{L}^{3D}$ to a feature map of dimension $128 \times 16 \times 16$ with a fully connected layer followed by a dropout with a probability of 0.3 and a Rectified Linear Unit (ReLU) layer. $\mathbf{L}^{app}$ is duplicated to form a spatially uniform map of size $128 \times 16 \times 16$.

These two maps are concatenated and used to reconstruct the input by applying four blocks of two convolutions, where the first convolution is preceded by a bilinear interpolation and all other pairs by transposed convolutions. Every convolution is followed by a batch-normalization and *ReLU* activation functions. We use mini-batches of size 32 for the encoder-decoder network.

The pose decoder $\mathcal{F}$ is a fully connected network with two hidden layers of dimension 2048. Poses and images are normalized by their mean and standard deviation on the training set. We use mini-batches of size 64 and the Adam [42] optimizer with a learning rate of 10^{-3} for 40 epochs to optimize the parameters θ_e, θ_d and θ_f. Moreover, we use a learning rate scheduler, which reduces the learning rate by a factor of 0.1 if the training loss does not decrease for more than two epochs.

For our network configurations $\mathcal{N}$ and $\mathcal{O}$, we do not make use of a pre-trained encoder. We use mini-batches of size 64 and the other settings are as described above. For $\mathcal{N}$, we optimize the parameters θ_e and θ_f simultaneously and the mini-batches only contain annotated hand images.

Network $\mathcal{O}$, on the other hand, optimizes θ_d as well, along with θ_e and θ_f. For the majority of our tested scenarios, we have a huge amount of images without 3D annotations and only a small amount of annotated ones. To compensate for the imbalance in our two classes, we implemented a random minority oversampling method [9]. We simply replicate selected samples from our class of annotated images, to achieve a consistent distribution of classes in our batches during training.

4.4 Evaluation

In Figure 4.1, we show the performance differences in terms of the *N-MPJPE* for our previously defined network configurations $\mathcal{M}$, $\mathcal{N}$ and $\mathcal{O}$ using various parameter settings. We see in Figure 4.1, that at a certain number of annotated training samples, the accuracy does not substantially increase anymore and that the geometric loss λ_{geo} seems to have an insignificant impact (also shown in Table 4.2), differently from what we expected.

The graphs for network $\mathcal{M}$ are very flat across the whole range of tested numbers of 3D annotations and the *N-MPJPE* is much lower for smaller numbers of annotated training samples compared to networks $\mathcal{N}$ and $\mathcal{O}$. As we anticipated, if we pre-train the encoder $\mathcal{E}$ with augmented unlabeled data, we achieve an even better accuracy if the number of annotated 3D data is low. This performance gain, however, diminishes the larger the amount of annotated data gets. If the number of annotations is high, all variations of $\mathcal{M}$ achieve similar results independent of their parameter settings. Our tested λ_{geo} values and setting ℓ to 21 and 50 do not seem to make much of a difference for network $\mathcal{M}$ in terms of accuracy.

The graphs for the networks $\mathcal{N}$ and $\mathcal{O}$ are steeper than those for network $\mathcal{M}$, which results in a higher *N-MPJPE* for a small number of annotations, whereas $\mathcal{N}$ has a much higher error than $\mathcal{O}$. If the amount of 3D annotations becomes large enough, the accuracy of $\mathcal{N}$ and $\mathcal{O}$ is better than of the variations of $\mathcal{M}$.

In Figure 4.2, we compare our networks $\mathcal{M}$, $\mathcal{N}$ and $\mathcal{O}$ in terms of their *PCK* and the corresponding *AUC*. For network $\mathcal{M}$, we only compare results for the settings $\ell = 50$ with $\lambda_{geo} = 10$, $\ell = 21$ with $\lambda_{geo} = 0.1$, and $\ell = 21$ with $\lambda_{geo} = 0.1$ using augmented unlabeled images, since the tested configurations have an almost identical performance, and to increase readability.

We see, that for a low number of annotated images, network $\mathcal{M}$ with the pre-trained encoder estimates hand poses most accurately. We achieve the best results using the

λ_{geo}	N-MPJPE [mm]		MPJPE [mm]	
	$\ell = 21$	$\ell = 50$	$\ell = 21$	$\ell = 50$
10.0	21.26	**21.58**	22.23	**22.56**
1.0	21.22	21.65	22.18	22.61
0.1	**21.17**	21.88	**22.13**	22.85
0.01	21.33	21.65	22.27	22.62

Table 4.2: Performance as function of the number of different λgeo-settings for network $\mathcal{M}$. The table compares different settings for the λ_{geo} parameters of Eq. (3.6) in terms of the *N-MPJPE* and *MPJPE*.

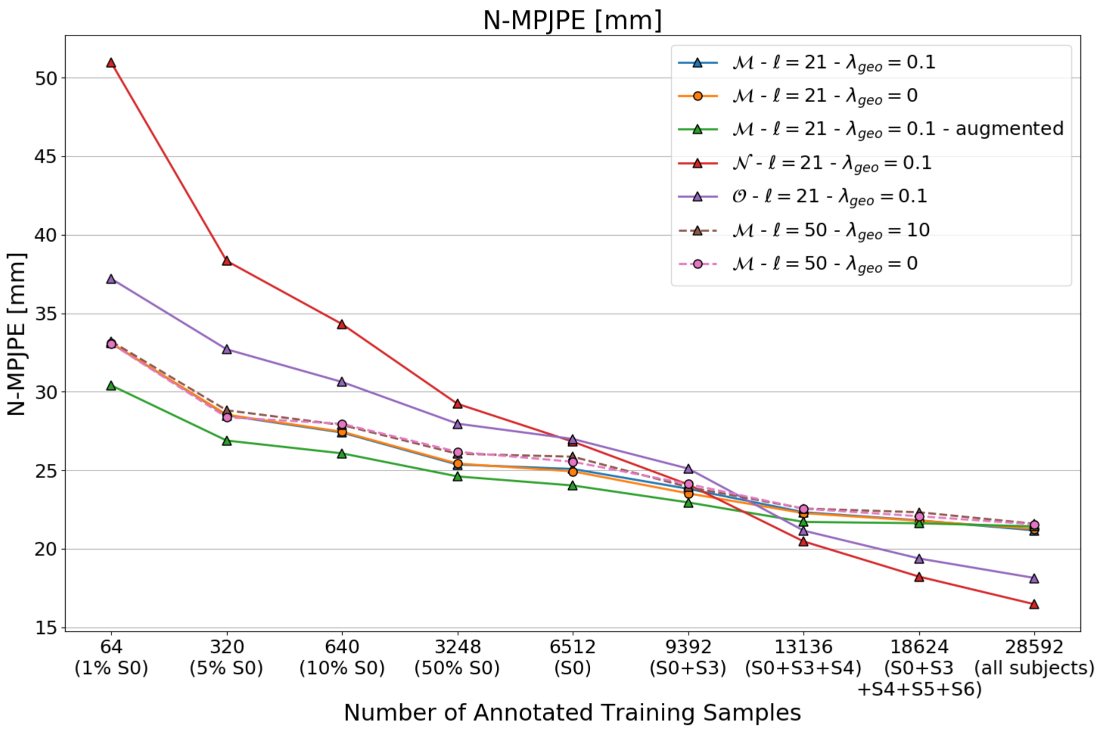


Figure 4.1: Performance as function of the number of training samples. The graph compares different network configurations by their *N-MPJPE* at different numbers of annotated training data.

encoder $\mathcal{E}$, that was trained with augmented unlabeled data. The curves for network $\mathcal{M}$ do not change as much for the scenarios we tested, compared to networks $\mathcal{N}$ and $\mathcal{O}$. Especially $\mathcal{N}$ shows a bad accuracy, if the number of 3D annotations used for training is low. However, it becomes more accurate faster, the more 3D annotations are available, compared to the other networks. Network $\mathcal{O}$ outperforms $\mathcal{N}$, if very little annotated data is available, but $\mathcal{N}$ slightly surpasses $\mathcal{O}$ as the amount of 3D annotations grows.

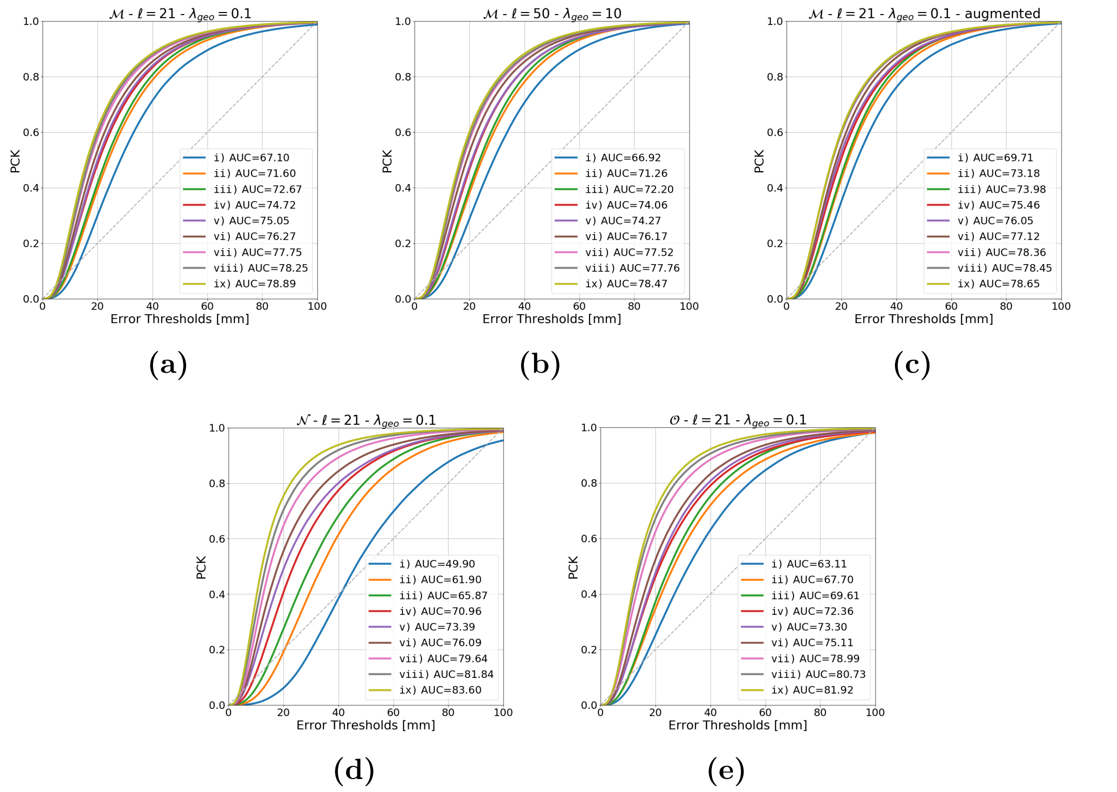


Figure 4.2: Comparison of our network structures by their *PCK*. The plots show the *PCK* over the respective thresholds within 0 mm - 100 mm. The correlations of each graph to our specified scenarios in Table 4.1 are as follows: `i)` 1% of S0, `ii)` 5% of S0, `iii)` 10% of S0, `iv)` 50% of S0, `v)` S0, `vi)` S0 + S3, `vii)` S0 + S3 + S4, `viii)` S0 + S3 + S4 + S5 + S6 and `ix)` all subjects. The graphs correspond to **(a)** $\mathcal{M}$ - $\ell = 21$ - $\lambda_{geo} = 0.1$ **(b)** $\mathcal{M}$ - $\ell = 50$ - $\lambda_{geo} = 10$ **(c)** $\mathcal{M}$ - $\ell = 21$ - $\lambda_{geo} = 0.1$ - augmented **(d)** $\mathcal{N}$ - $\ell = 21$ - $\lambda_{geo} = 0.1$ **(e)** $\mathcal{O}$ - $\ell = 21$ - $\lambda_{geo} = 0.1$.

Table 4.3 compares the results of the *N-MPJPE* (Fig. 4.1) and *AUC* (Fig. 4.2). This table shows, that as soon as we use S0, S3 and S4 (13 136 annotated images) for training, $\mathcal{N}$ and $\mathcal{O}$ outperform network $\mathcal{M}$.

To avoid confusion, Table 4.3 only displays the results of network $\mathcal{M}$ with the settings $\ell = 21$ with $\lambda_{geo} = 0.1$, and $\ell = 21$ with $\lambda_{geo} = 0.1$ using augmented unlabeled images, since the versions with different λ_{geo} settings and $\ell = 50$ only exhibit minor differences in performance.

In the following subsections, we evaluate the quality of our latent representation and we show examples, that illustrate the quality of the synthesized images, as well as the 3D hand pose estimations for different amounts of 3D annotations.

Scenario	**N-MPJPE [mm] ↓**				**AUC ↑**			
	$\mathcal{M}$	$\mathcal{M}_{\textbf{aug}}$	$\mathcal{N}$	$\mathcal{O}$	$\mathcal{M}$	$\mathcal{M}_{\textbf{aug}}$	$\mathcal{N}$	$\mathcal{O}$
All nine subjects	21.17	21.42	**16.46**	18.13	78.89	78.65	**83.60**	81.92
S0+S3+S4+S5+S6	21.80	21.62	**18.22**	19.37	78.25	78.45	**81.84**	80.73
S0+S3+S4	22.32	21.70	**20.46**	21.15	77.75	78.36	**79.64**	78.99
S0+S3	23.81	**22.95**	24.09	25.10	76.27	**77.12**	76.09	75.11
S0	25.08	**24.02**	26.82	26.99	75.05	**76.05**	73.39	73.30
50% of S0	25.36	**24.60**	29.22	27.96	74.72	**75.46**	70.96	72.36
10% of S0	27.40	**26.07**	34.30	30.62	72.67	**73.98**	65.87	69.61
5% of S0	28.49	**26.89**	38.33	32.70	71.60	**73.18**	61.90	67.70
1% of S0	33.08	**30.40**	50.97	37.19	67.10	**69.71**	49.90	63.11

Table 4.3: Comparison of selected network designs in terms of their *N-MPJPE* and *AUC* over our defined scenarios. To increase the readability, we only list the results of network $\mathcal{M}$ with the settings $\ell = 21$ with $\lambda_{geo} = 0.1$ ($\mathcal{M}$), and $\ell = 21$ with $\lambda_{geo} = 0.1$ using augmented unlabeled images ($\mathcal{M}_{aug}$). For *N-MPJPE* lower scores are better (denoted by ↓), whereas for *AUC* higher scores are better (denoted by ↑).

4.4.1 Semi-Supervised 3D Hand Pose Estimation

Below, we evaluate the quality of our latent representation and we show examples, that illustrate the quality of the synthesized images, as well as the 3D hand pose estimates for different amounts of available 3D annotations. We show the results for our encoder-decoder network and $\mathcal{M}$ using the following parameter settings:

1) $\ell = 21$ - $\lambda_{geo} = 0.1$ (Fig. 4.3 and Fig. 4.4),

2) $\ell = 50$ - $\lambda_{geo} = 10.0$ (Fig. 4.5 and Fig. 4.6) and

3) $\ell = 21$ - $\lambda_{geo} = 0.1$ where the encoder was pre-trained with augmented unlabeled multi-view images (Fig. 4.7, and Fig. 4.8).

Figure 4.3, Figure 4.5 and Figure 4.7 show Novel View Synthesis (NVS) predictions for five input images from the test set, that show different hand poses. As described in Chapter 3, the encoder $\mathcal{E}$ encodes an input image into variables $\mathbf{L}^{3D}$ and $\mathbf{L}^{app}$, which are meant to represent geometry and appearance, respectively. In order to validate this behaviour, we multiply $\mathbf{L}^{3D}$ by different rotation matrices $\mathbf{R}$ and feed the result along with the original $\mathbf{L}^{app}$ into the decoder $\mathcal{D}$, which produces novel views. Since the network does not learn the background, we can, e.g., set the background to white, as shown in aforesaid figures. We see in the novel views of Figure 4.3, that in some cases especially the fingers are better reconstructed than in Figure 4.5, however, the encoder-decoder network sometimes fails to reconstruct them at all with $\ell = 21$. If we use augmented multi-view images to train our encoder-decoder network, it eagerly tries to separate individual fingers using darker lines, especially visible in the first two inputs of Figure 4.7.

In Figure 4.4, Figure 4.6 and Figure 4.8, we see novel views and pose reconstructions at different levels of supervision, for an input image from the test set. In all three figures, we see that using only 64 3D hand pose annotations to train $\mathcal{M}$, the result looks very bad (last row in each figure). However, as the number of available annotations for training increases, the quality of the pose reconstructions becomes much better, as the presented results in Table 4.3 already suggested.

We further tried to back-project the latent representation $\mathbf{L}^{3D}$ into the input image to find clues, if the 3D points correspond to any visual features of the input. However, the latent variables do not seem to correlate with any visual characteristics e.g., individual fingers, joint positions or hand orientation. Figure 4.9 illustrates novel views and the corresponding visualization of $\mathbf{L}^{3D}$ from the same input image for different parameter settings. However, the 3D latent variables $\mathbf{L}^{3D}$ do not seem to have any visible correlations with the 21 keypoints of the hand in the input image. $\mathbf{L}^{3D}$ probably still encodes parts of the background and appearance, because of impurities in the training set and too different illuminations of the hands from different viewpoints.

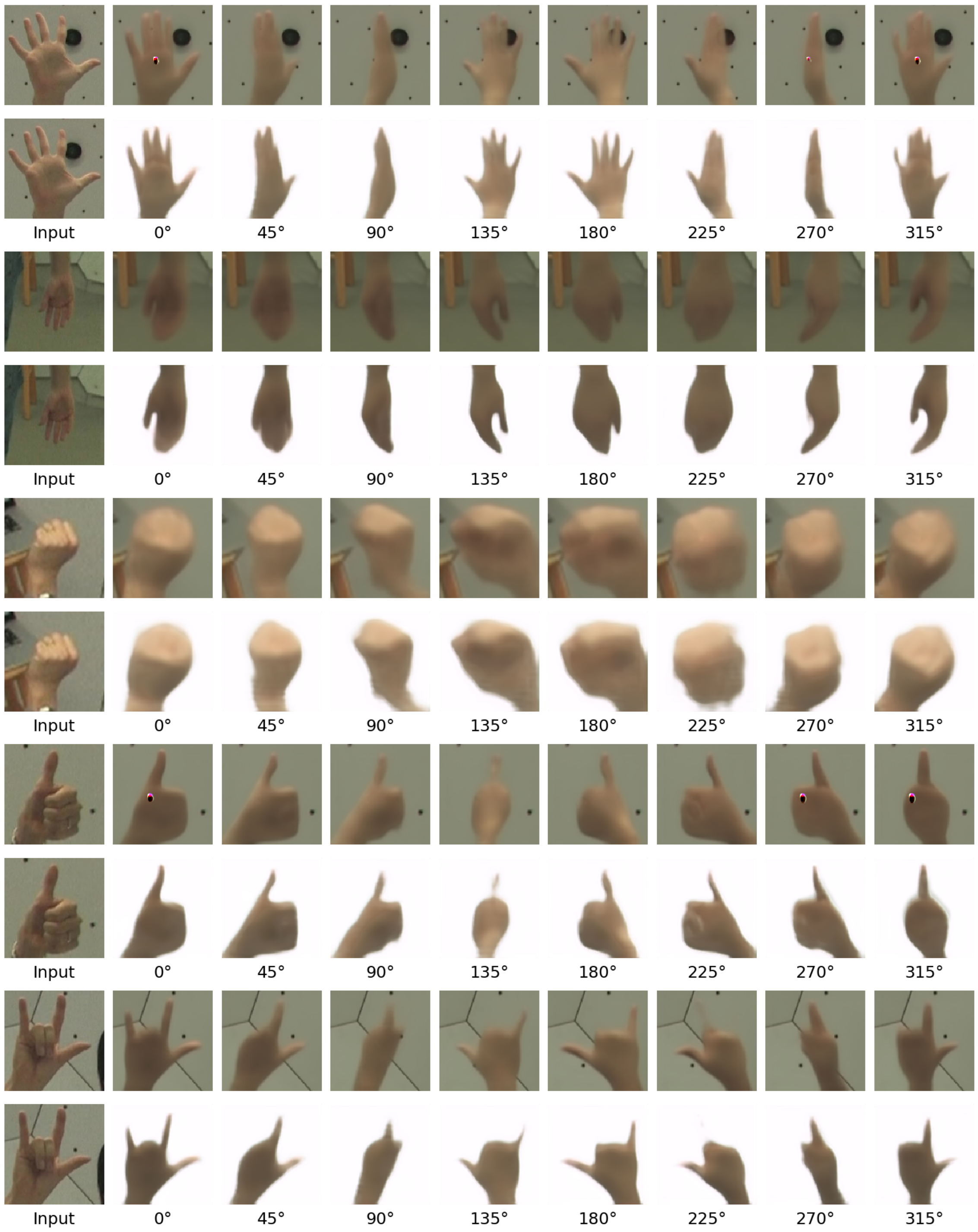


Figure 4.3: *NVS* predictions of the encoder-decoder with $\ell = 21$. The leftmost image of each row shows an input image from the test set. We show the synthesized images for previously unseen viewpoints with and without the background. For more complex poses (e.g., last two samples), the encoder-decoder network has problems to properly synthesize individual fingers at novel views.

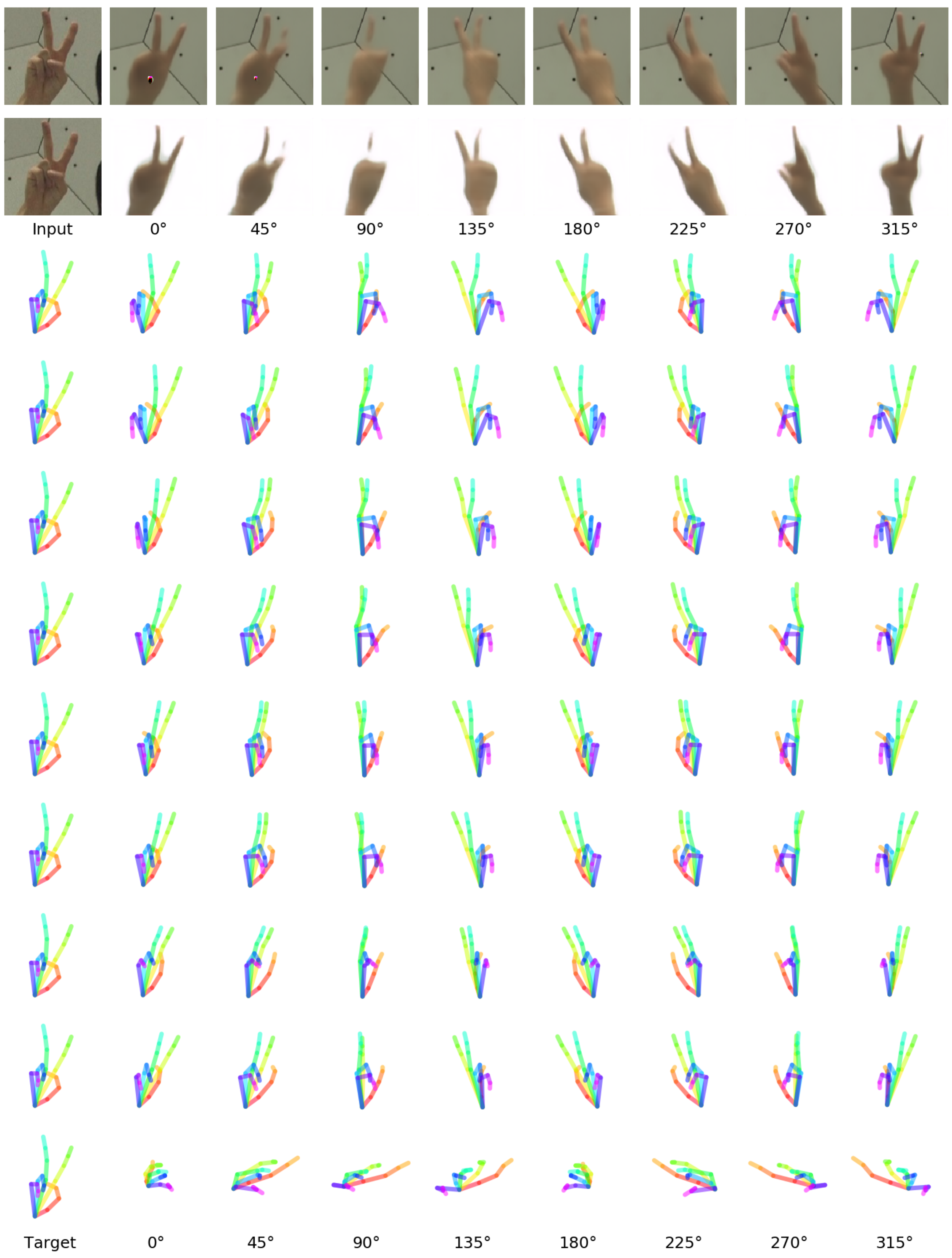


Figure 4.4: *NVS* and pose predictions of $\mathcal{M}$ with $\ell = 21$ and $\lambda_{geo} = 0.1$. The first two rows show a sample input image from the test set on the left and the corresponding synthesized images for previously unseen viewpoints with and without the background. The pose predictions were made with networks trained with different levels of supervision, corresponding to the rows of Table 4.1 with the number of annotated 3D data decreasing from top to bottom. The leftmost poses represent the target 3D hand pose of the input image. As the number of available annotations for training increases (bottom to top), the quality of the pose reconstructions becomes much better.

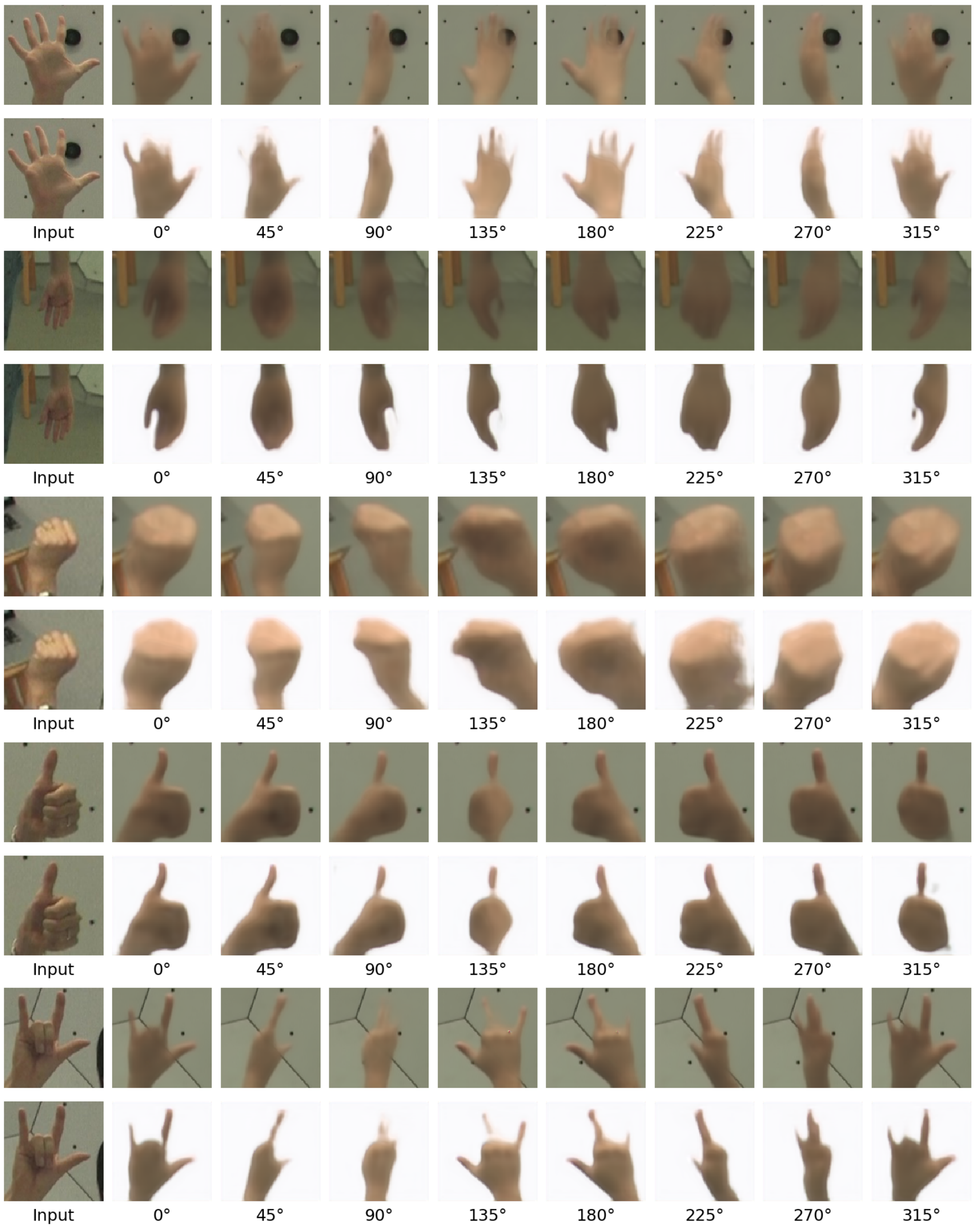


Figure 4.5: *NVS* predictions of the encoder-decoder with $\ell = 50$. The leftmost image of each row shows an input image from the test set. We show the synthesized images for previously unseen viewpoints with and without the background. For more complex poses (e.g., first and last input sample), the encoder-decoder network has problems to properly synthesize individual fingers at novel views.

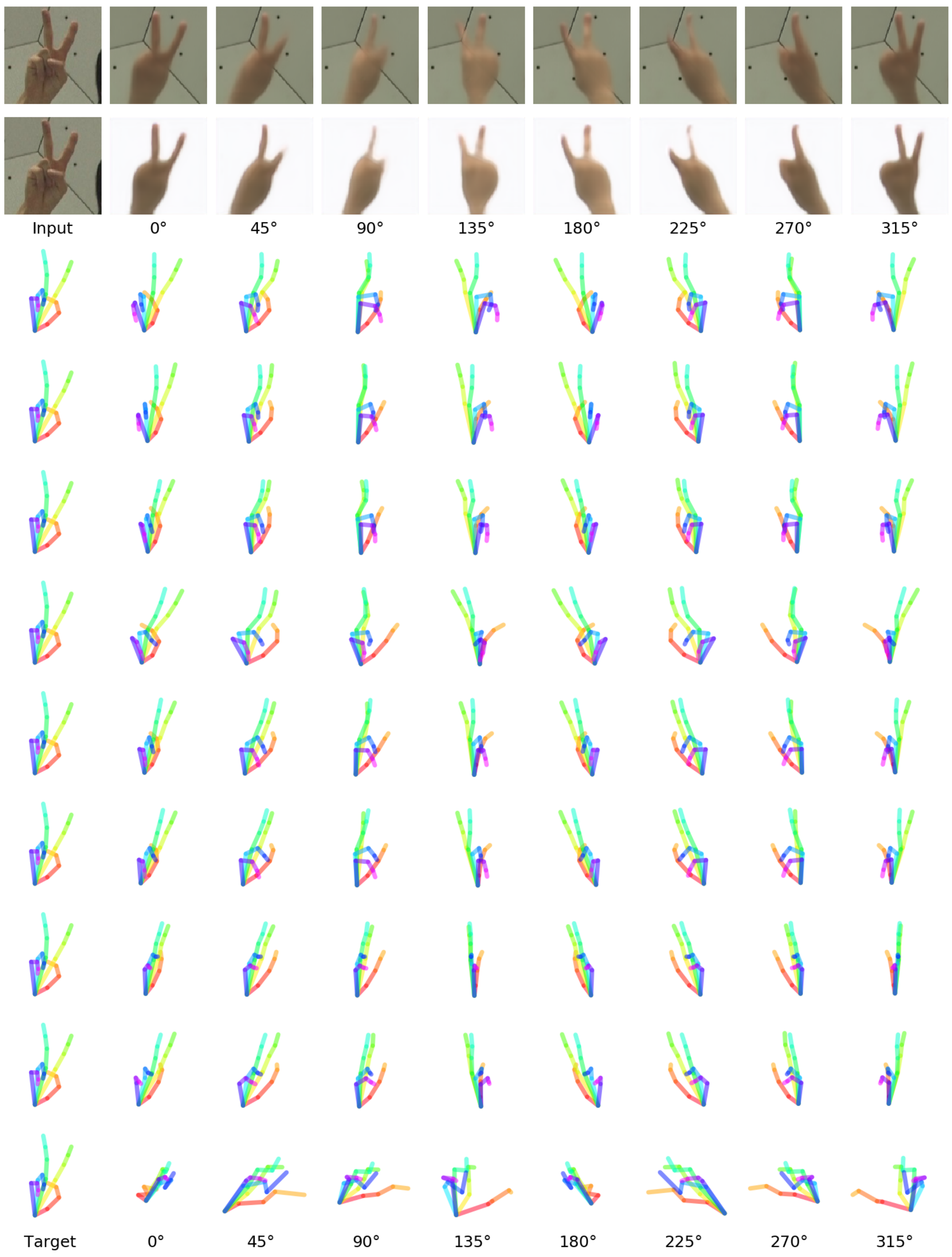


Figure 4.6: *NVS* and pose predictions of $\mathcal{M}$ with $\ell = 50$ and $\lambda_{geo} = 10.0$. The first two rows show a sample input image from the test set on the left and the corresponding synthesized images for previously unseen viewpoints with and without the background. The pose predictions were made with networks trained with different levels of supervision, corresponding to the rows of Table 4.1 with the number of annotated 3D data decreasing from top to bottom. The leftmost poses represent the target 3D hand pose of the input image. As the number of available annotations for training increases (bottom to top), the quality of the pose reconstructions becomes much better.

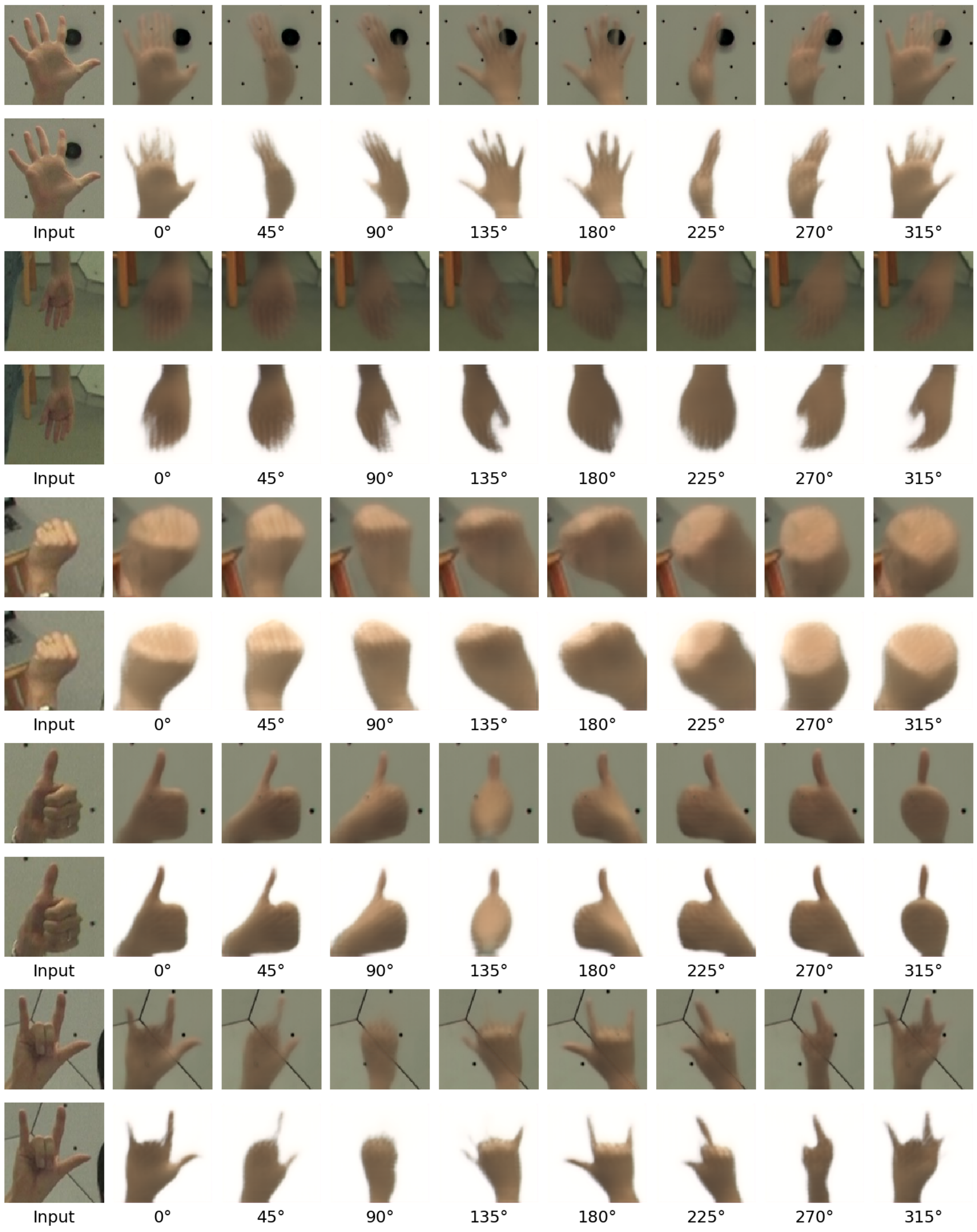


Figure 4.7: *NVS* predictions of the encoder-decoder with $\ell = 21$, using augmented unlabeled multi-view images. The leftmost image of each row shows an input image from the test set. We show the synthesized images for previously unseen viewpoints with and without the background. The encoder-decoder network eagerly tries to separate individual fingers using darker lines, especially visible in the first two inputs and for more complex poses (e.g., last input sample), it has problems to properly synthesize individual fingers at novel views.

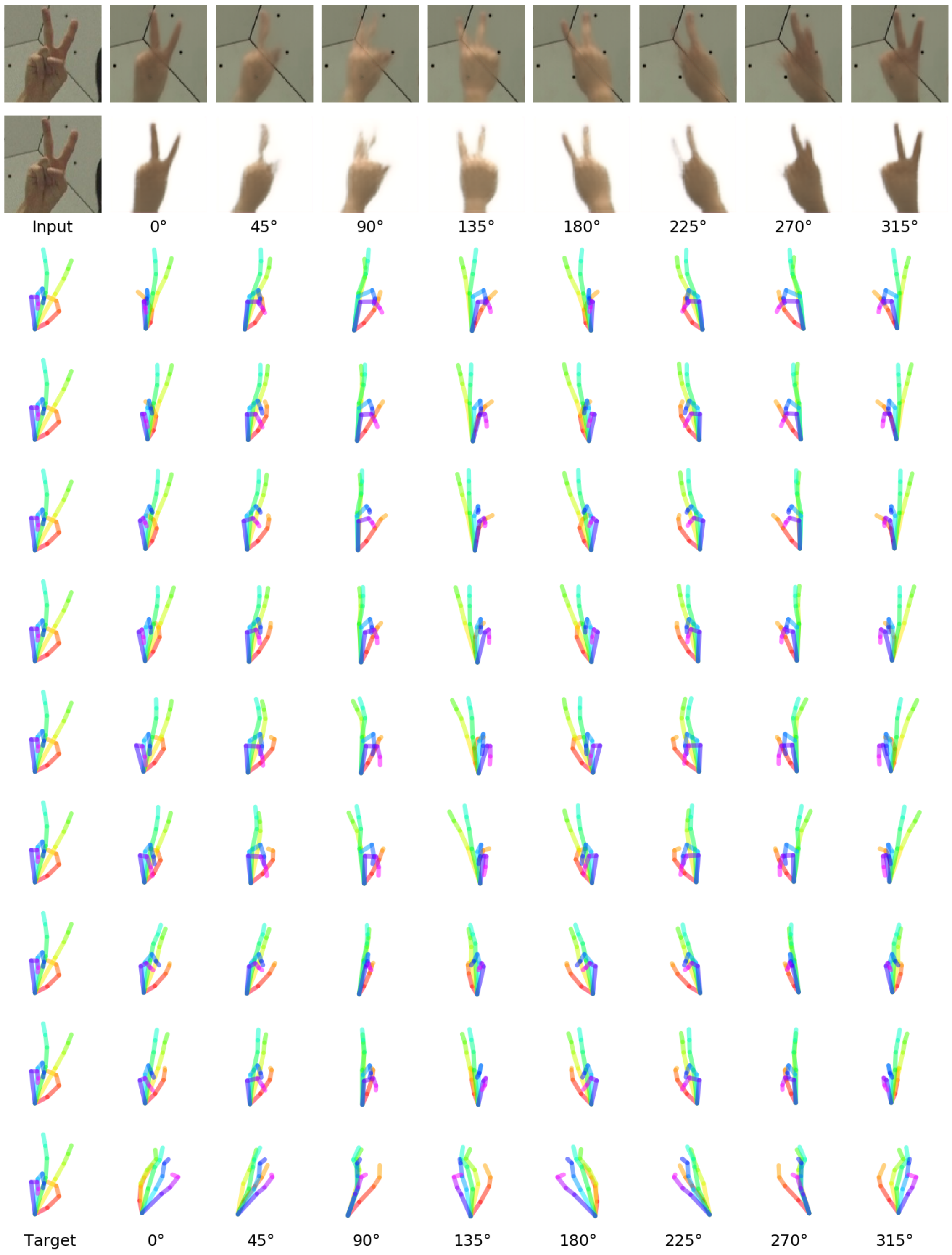
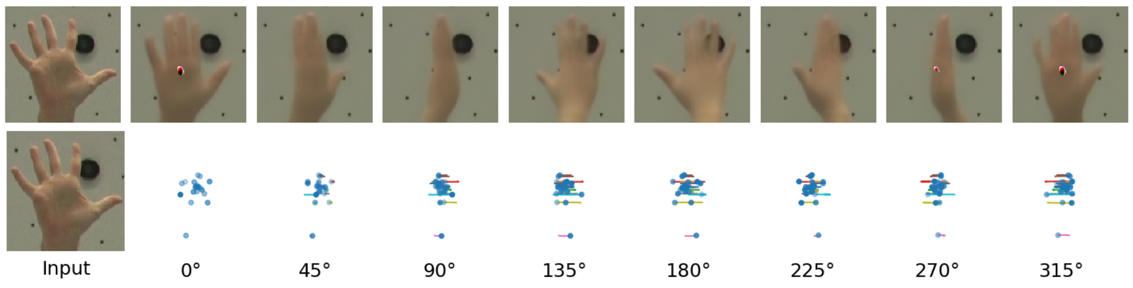
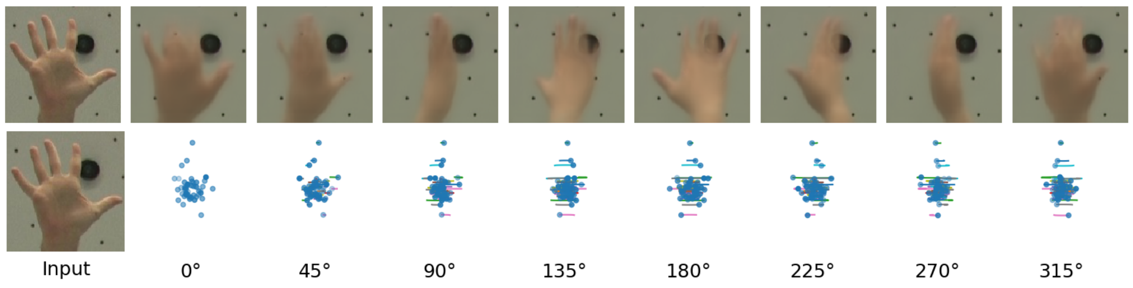


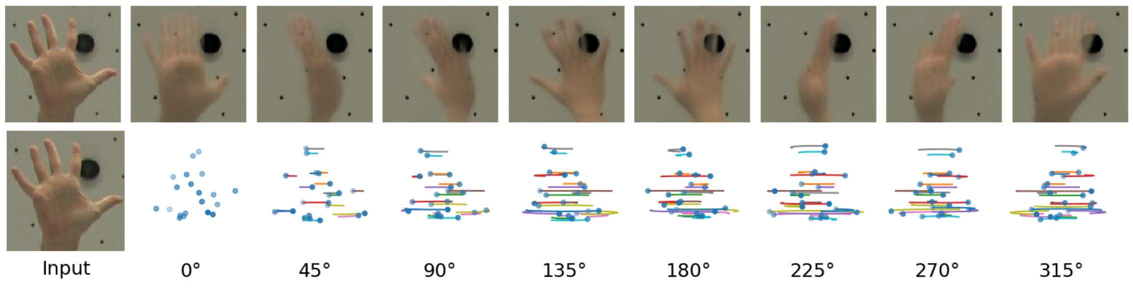
Figure 4.8: *NVS* and pose predictions of $\mathcal{M}$ with $\ell = 21$ and $\lambda_{geo} = 0.1$, using an encoder that was pre-trained with augmented unlabeled multi-view images. The first two rows show a sample input image from the test set on the left and the corresponding synthesized images for previously unseen viewpoints with and without the background. The pose predictions were made with networks trained with different levels of supervision, corresponding to the rows of Table 4.1 with the number of annotated 3D data decreasing from top to bottom. The leftmost poses represent the target 3D hand pose of the input image. As the number of available annotations for training increases (bottom to top), the quality of the pose reconstructions becomes much better.



(a) $\ell = 21$



(b) $\ell = 50$

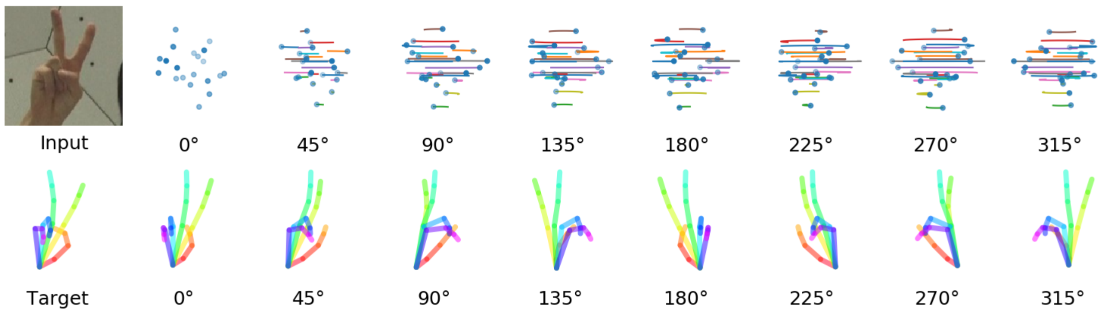


(c) $\ell = 21$, using augmented unlabeled multi-view images.

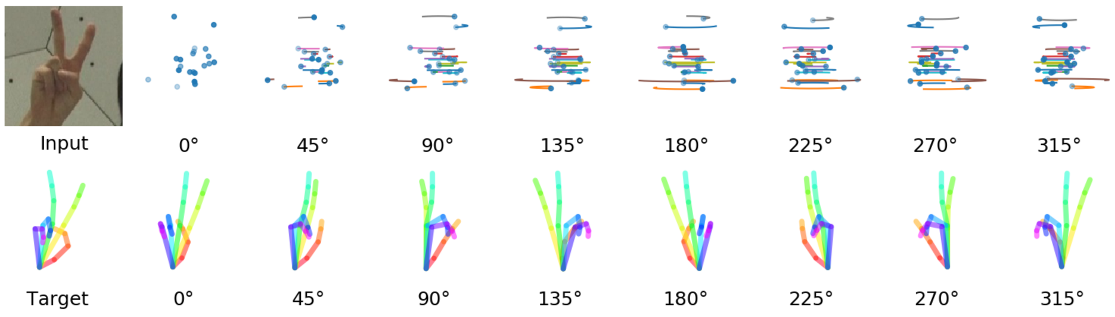
Figure 4.9: *NVS* predictions of the encoder-decoder network and the corresponding 3D latent variables. The leftmost image of each row shows an input image from the test set. The 3D latent variables $\mathbf{L}^{3D}$ do not seem to have any visible correlations with the 21 keypoints of the hand in the input image. $\mathbf{L}^{3D}$ probably still encodes parts of the background and appearance, because of impurities in the training set and too different illuminations of the hands from different viewpoints.

4.4.2 Supervised 3D Hand Pose Estimation

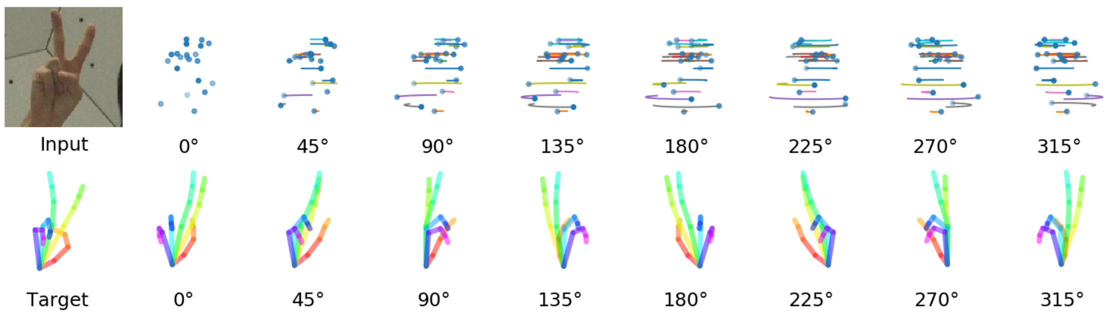
In Figure 4.10, Figure 4.11 and Figure 4.12, we show results for network $\mathcal{N}$. This network directly maps an input image to the 3D pose, without pre-training the encoder with unlabeled multi-view images. In these figures, we see sample input images from the test set, and the corresponding 3D latent variables $\mathbf{L}^{3D}$ and predicted 3D hand poses at different viewpoints. The colored lines, that appear in the rotated latent representations, illustrate the trajectory of the latent 3D points. The figures clearly show, how the 3D hand pose predictions evolve as more and more annotated images were used for training. The quality of the predicted poses strongly depends on the complexity of the pose. For more simple poses (Figure 4.11) only around 3 248 annotations and for more complex poses (Figure 4.10 and Figure 4.12) more than 13 136 annotations during training are needed for decent 3D hand pose predictions. As discussed before, the 3D latent variables $\mathbf{L}^{3D} \in \mathbb{R}^{21\times3}$ do not seem to have a visible correlation with the 21 keypoints of the hand in the input image.



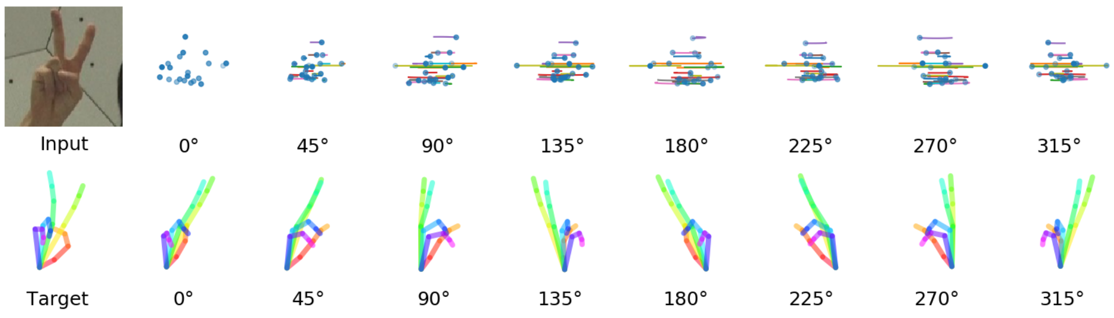
(a) All subjects (28 592 annotations)



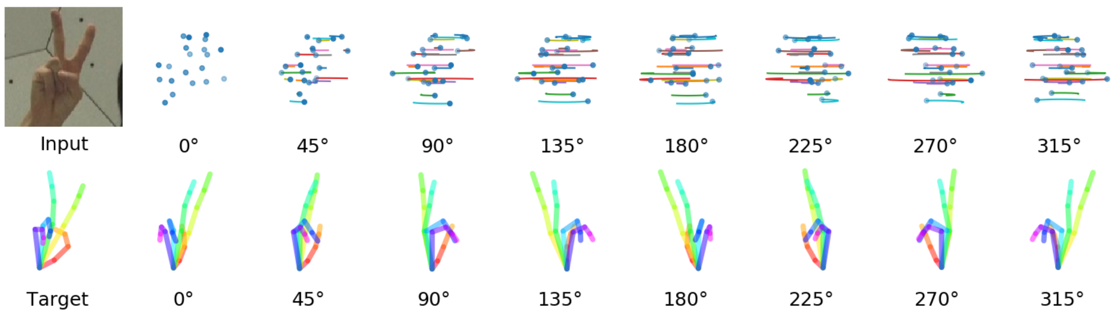
(b) S0, S3, S4, S5 and S6 (18 624 annotations)



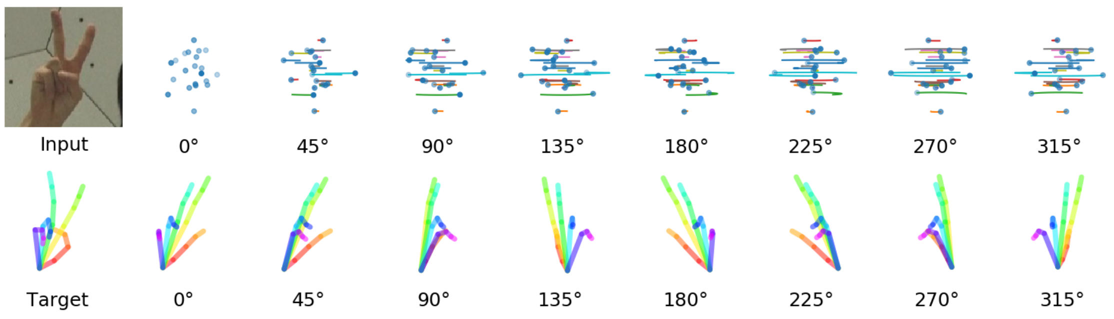
(c) S0, S3 and S4 (13 136 annotations)



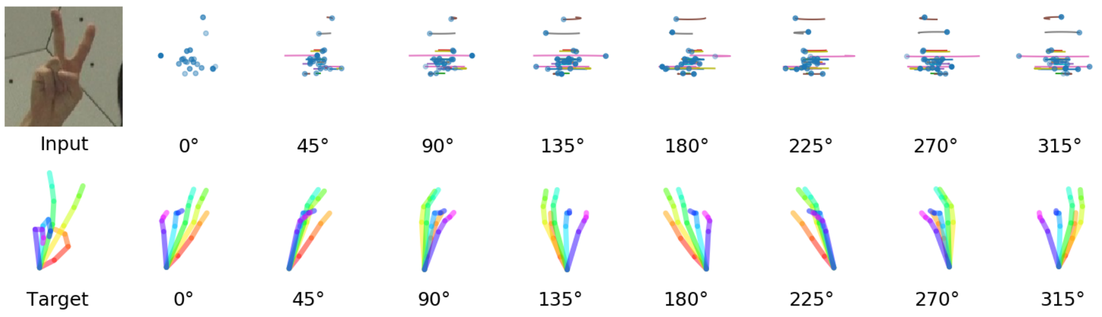
(d) S0 and S3 (9 392 annotations)



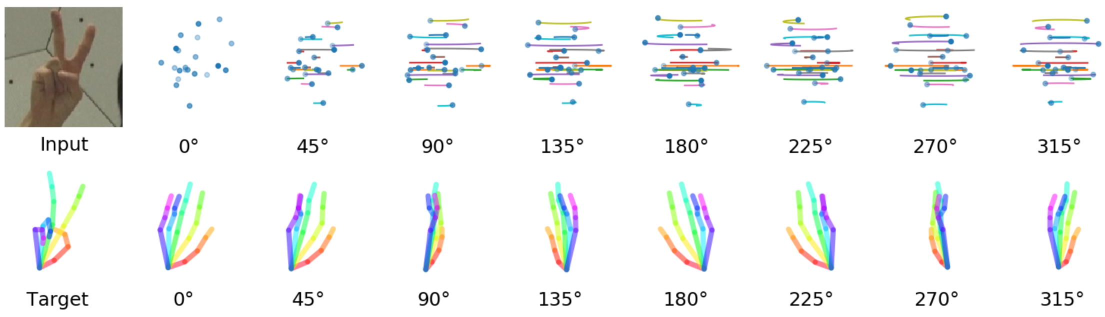
(e) 100% of S0 (6 512 annotations)



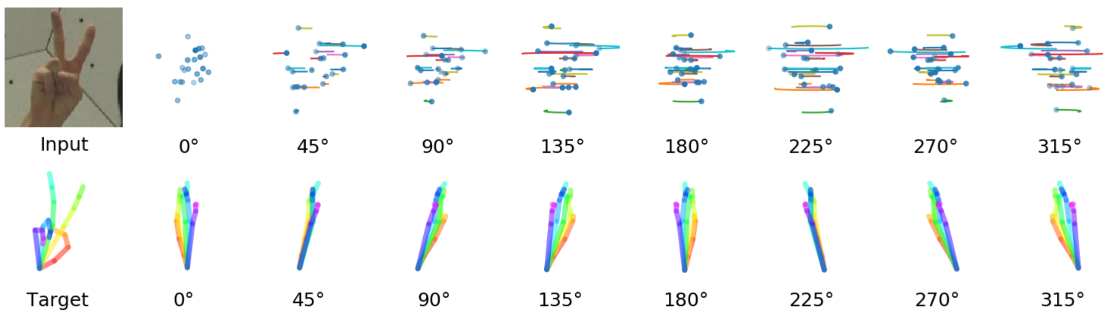
(f) 50% of S0 (3 248 annotations)



(g) 10% of S0 (640 annotations)

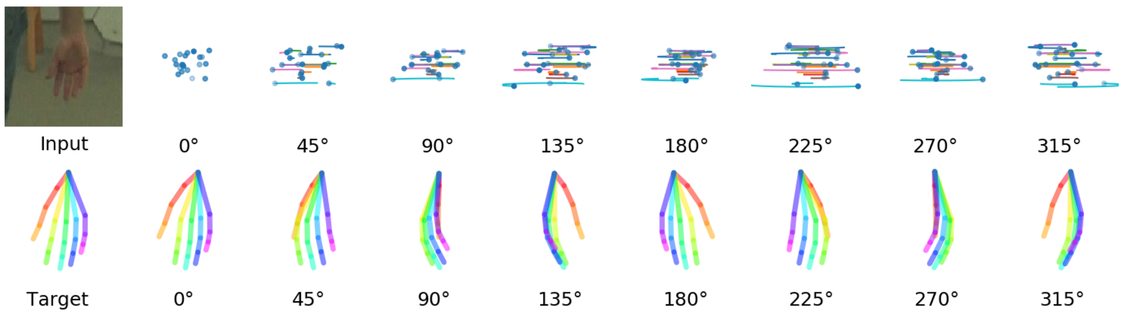


(h) 5% of S0 (320 annotations)

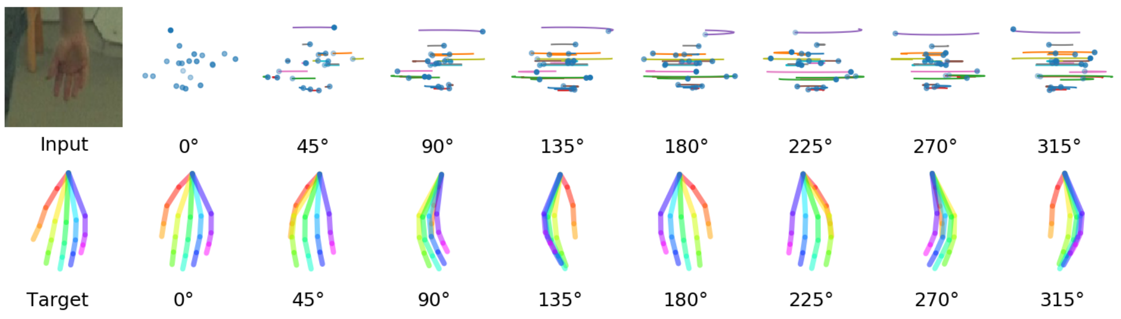


(i) 1% of S0 (64 annotations)

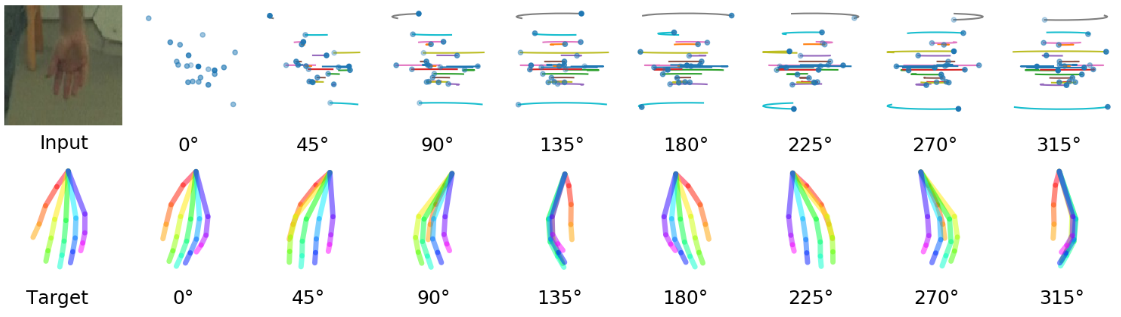
Figure 4.10: Network $\mathcal{N}$ experiments - two fingers up.



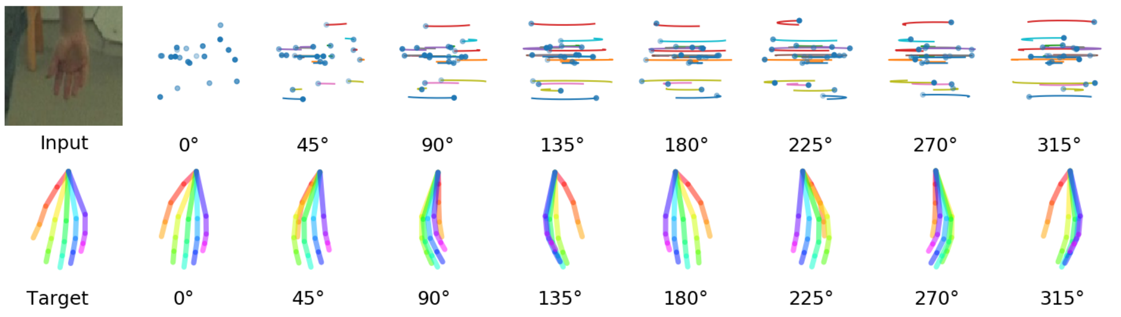
(a) All subjects (28 592 annotations)



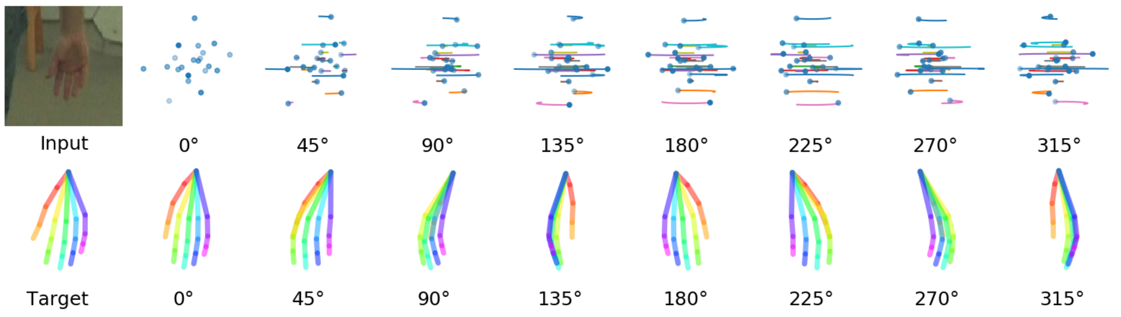
(b) S0, S3, S4, S5 and S6 (18 624 annotations)



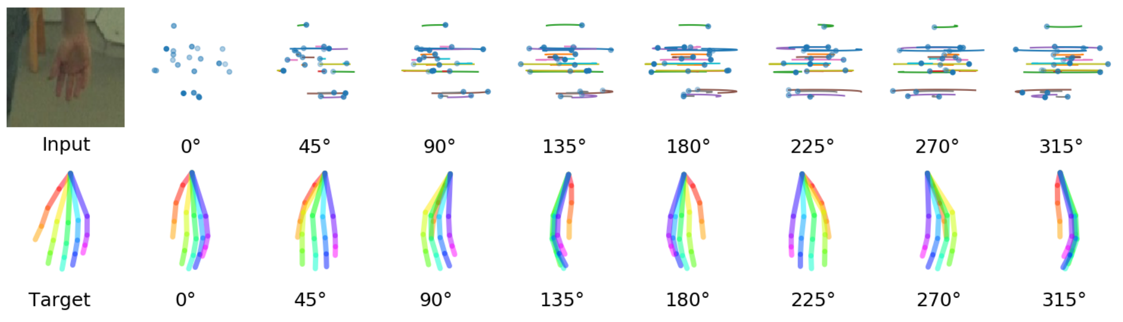
(c) S0, S3 and S4 (13 136 annotations)



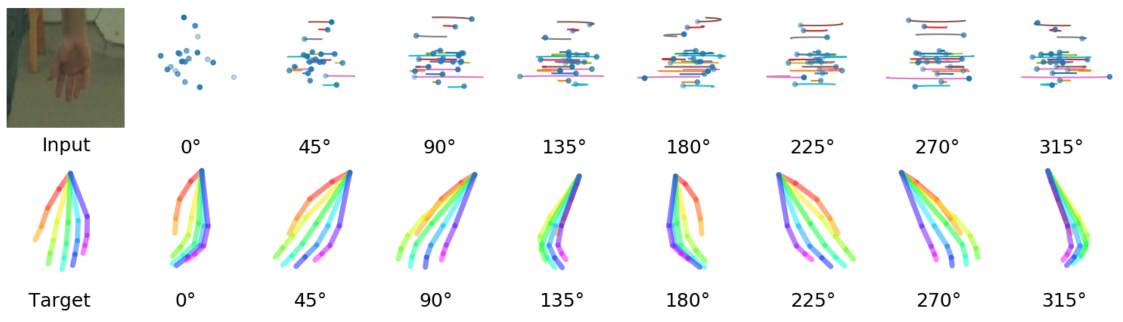
(d) S0 and S3 (9 392 annotations)



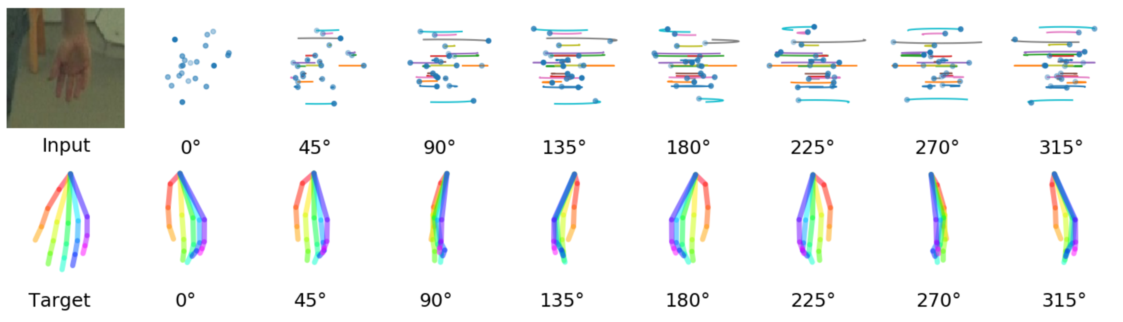
(e) 100% of S0 (6 512 annotations)



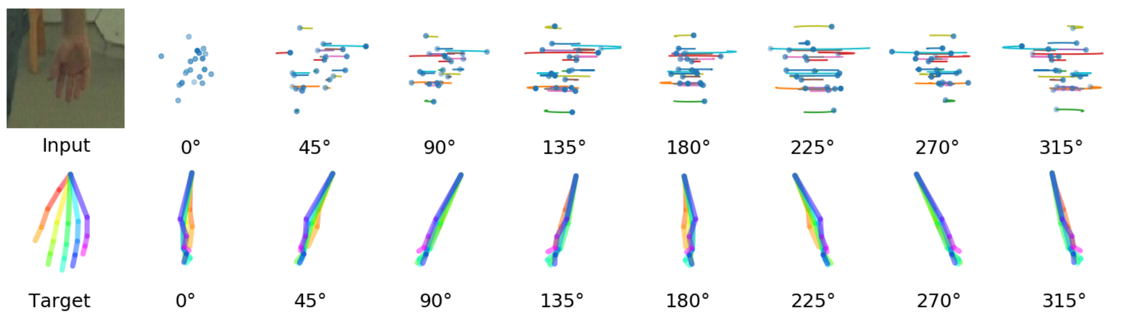
(f) 50% of S0 (3 248 annotations)



(g) 10% of S0 (640 annotations)

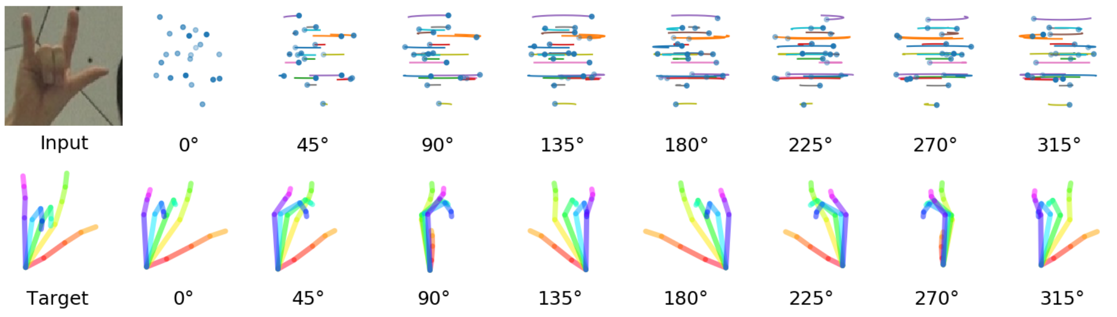


(h) 5% of S0 (320 annotations)

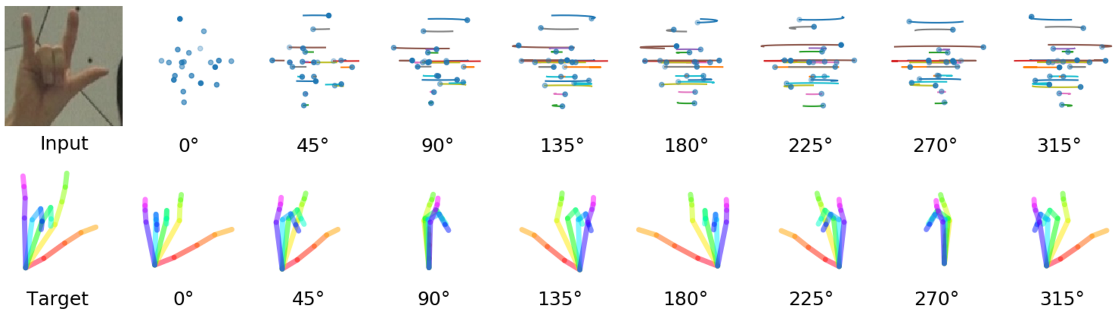


(i) 1% of S0 (64 annotations)

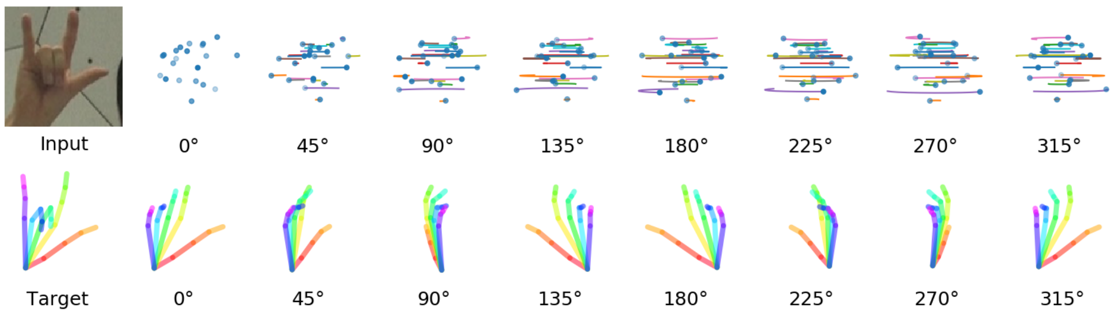
Figure 4.11: Network $\mathcal{N}$ experiments - hand hanging downward.



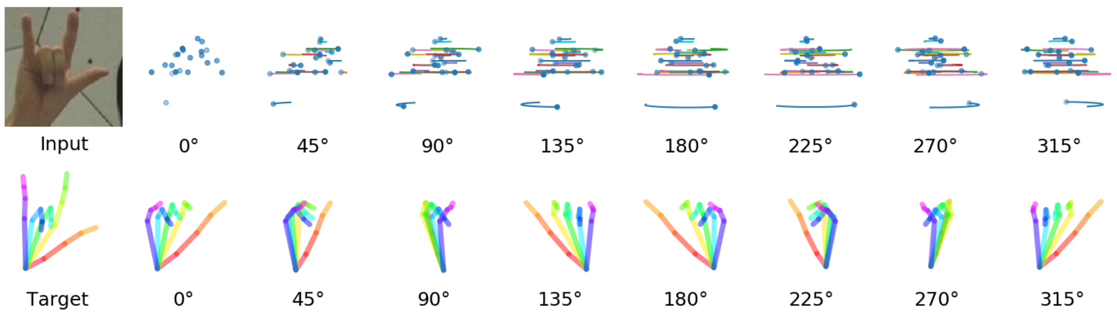
(a) All subjects (28 592 annotations)



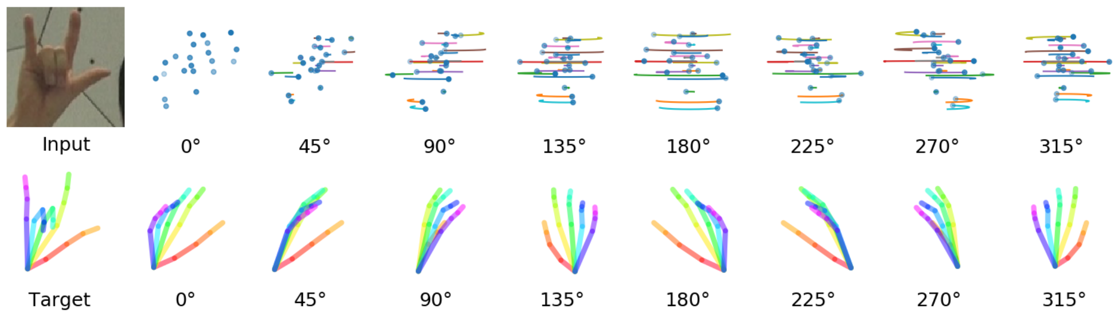
(b) S0, S3, S4, S5 and S6 (18 624 annotations)



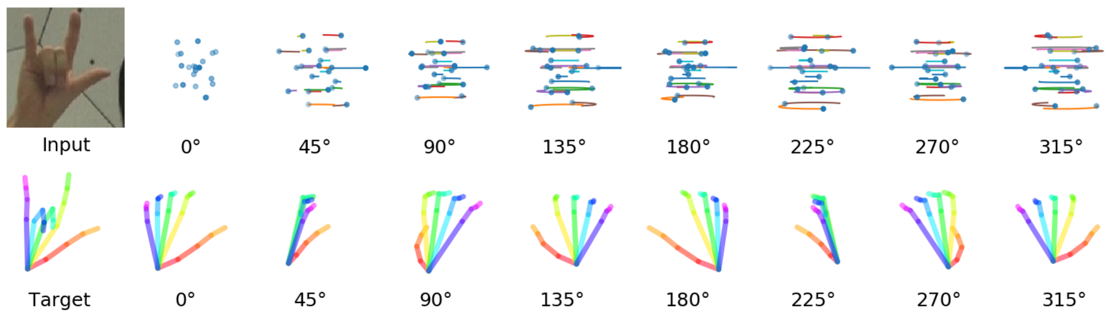
(c) S0, S3 and S4 (13 136 annotations)



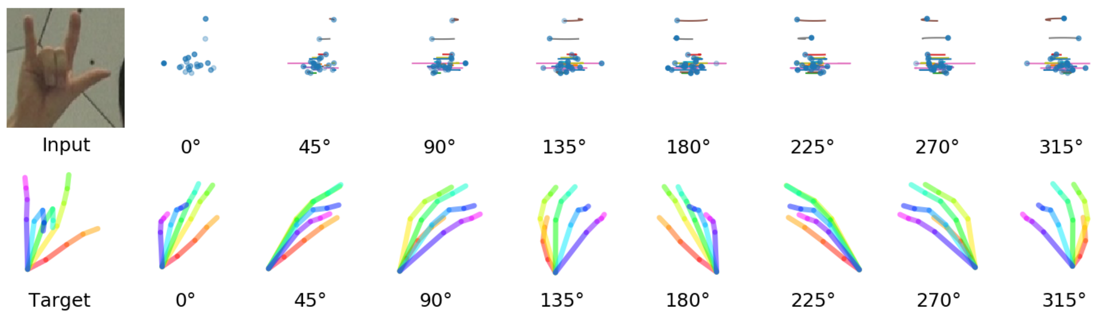
(d) S0 and S3 (9 392 annotations)



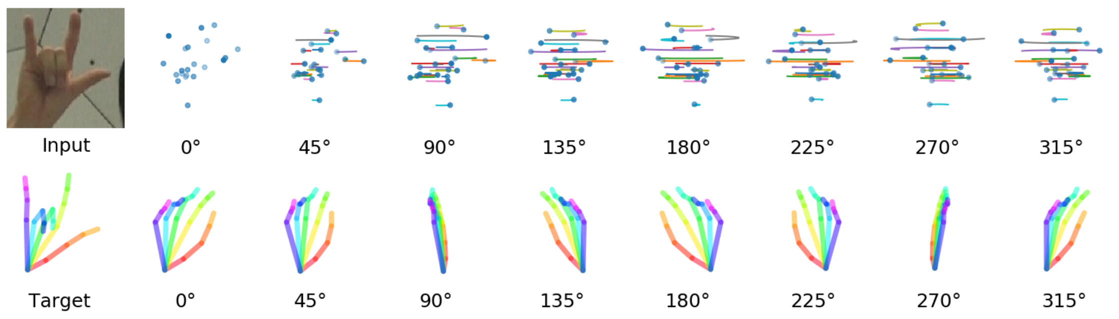
(e) 100% of S0 (6 512 annotations)



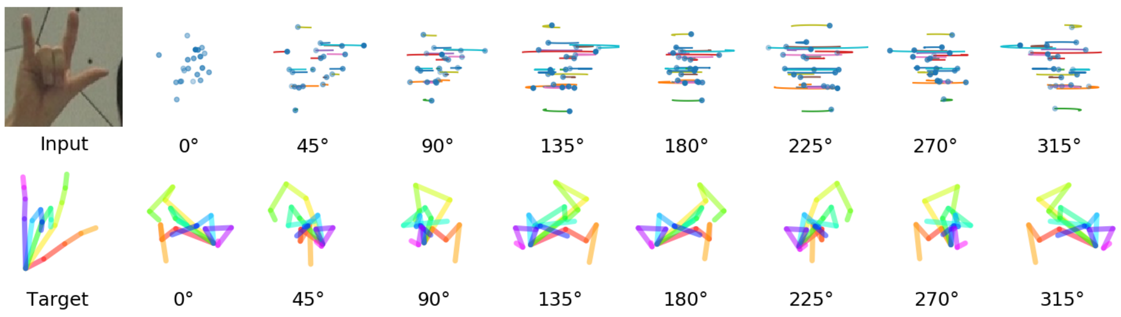
(f) 50% of S0 (3 248 annotations)



(g) 10% of S0 (640 annotations)



(h) 5% of S0 (320 annotations)

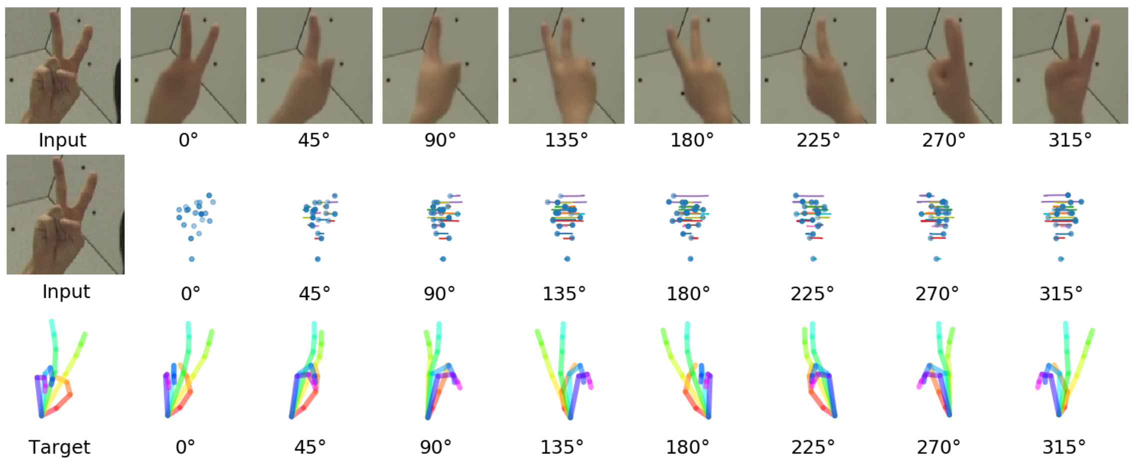


(i) 1% of S0 (64 annotations)

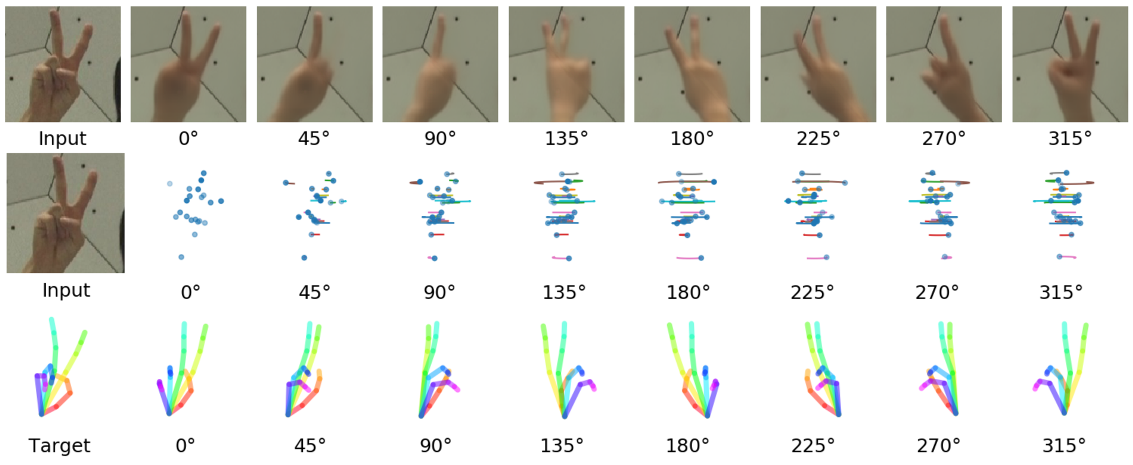
Figure 4.12: Network $\mathcal{N}$ experiments - three fingers up.

4.4.3 Fully Semi-Supervised 3D Hand Pose Estimation

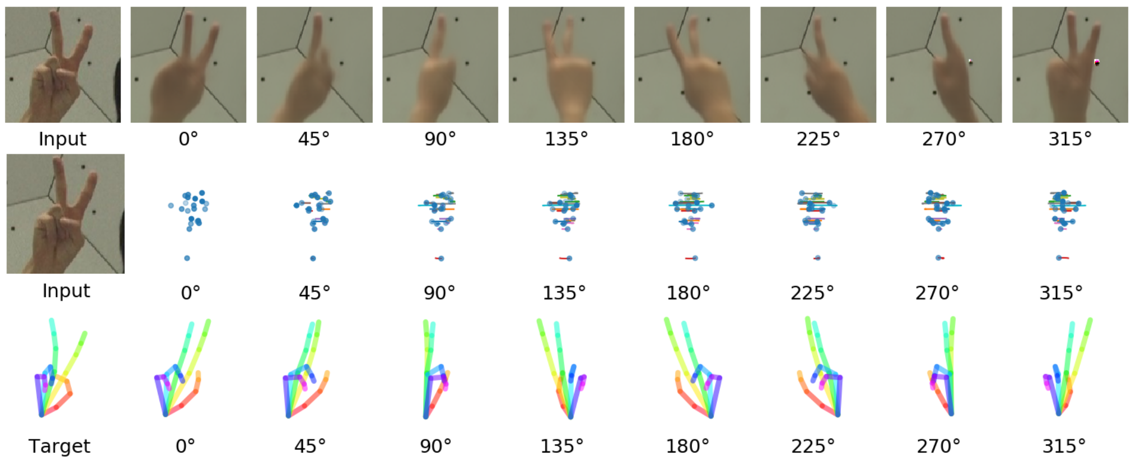
Figure 4.13, Figure 4.14 and Figure 4.15 show results for network $\mathcal{O}$. This network decodes latent variables into an output image and the 3D hand pose simultaneously. The figures illustrate the results for sample input images from the test set from different viewpoints. The first row of each sub-figure shows the *NVS* predictions from different views for the given input image. The second row depicts the corresponding latent space, including their trajectories, as the colored lines indicate. The third row illustrates the target pose (leftmost) and the predicted 3D hand pose from different viewpoints. The figures clearly show, how the *NVS* and 3D hand pose predictions evolve as more and more annotated images were used for training. The pose predictions for the input images in Figure 4.13 and Figure 4.14 where the pose network was trained with only 320 annotations are already very close to the target pose. For more complex poses (like the sample from Figure 4.14) almost all available annotations are needed for an acceptable pose prediction. As we already showed before, the 3D latent variables $\mathbf{L}^{3D} \in \mathbb{R}^{21\times3}$ do not seem to have a visible correlation with the 21 keypoints of the hand in the input image.



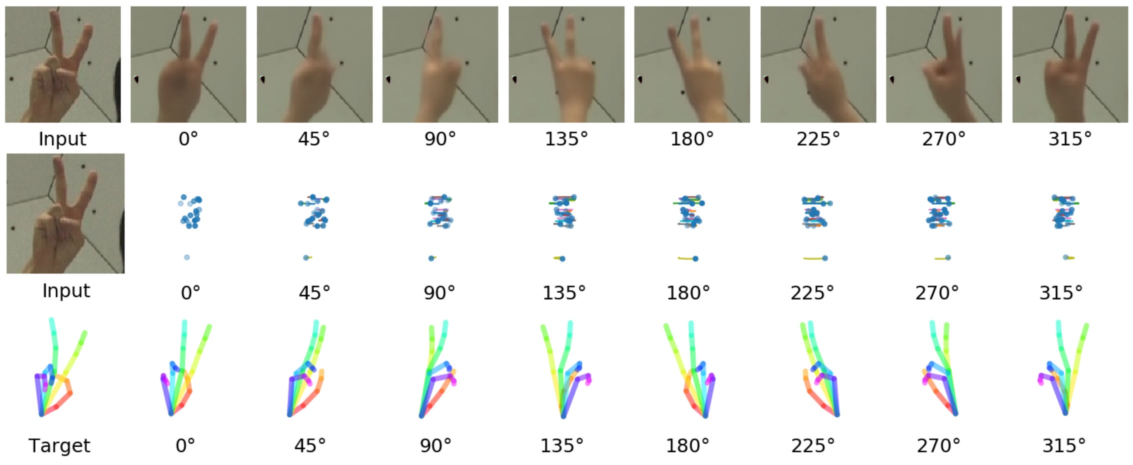
(a) All subjects (28 592 annotations)



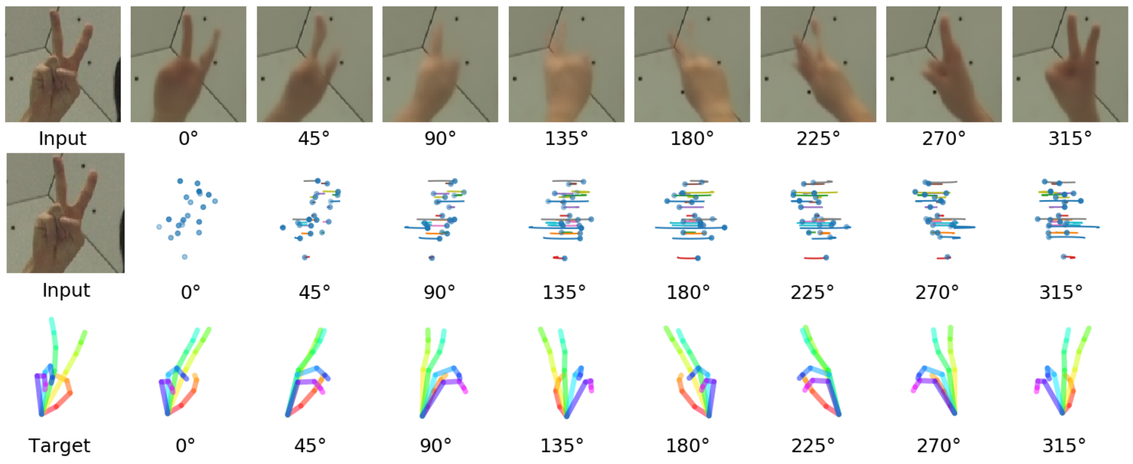
(b) S0, S3, S4, S5 and S6 (18 624 annotations)



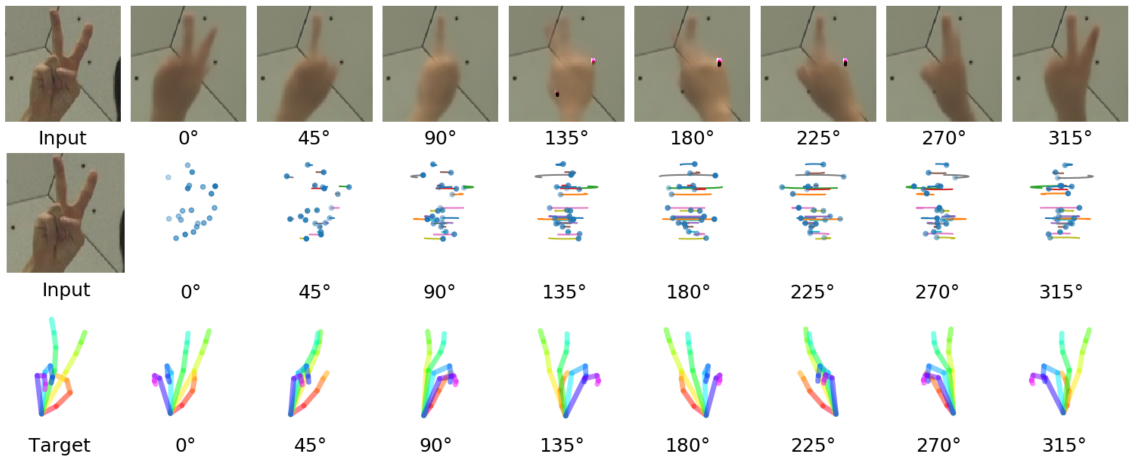
(c) S0, S3 and S4 (13 136 annotations)



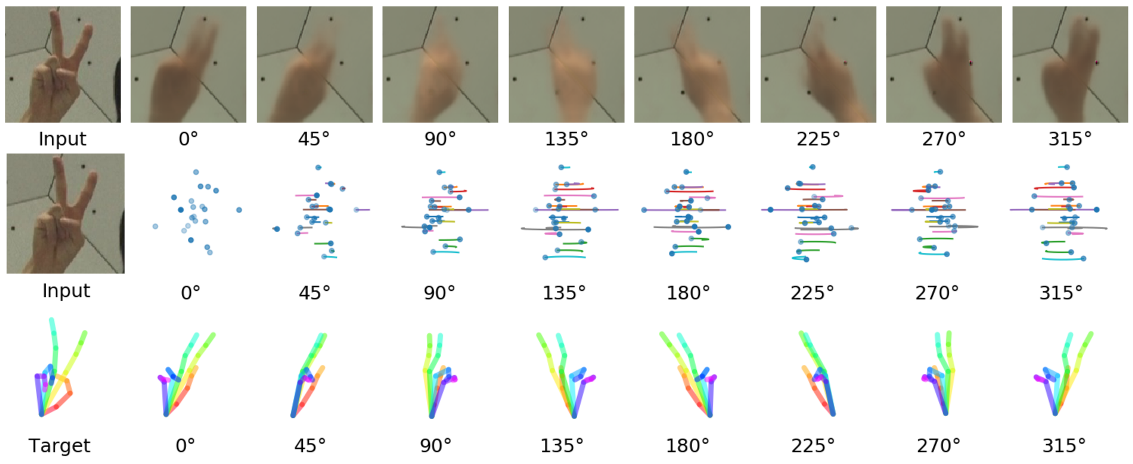
(d) S0 and S3 (9 392 annotations)



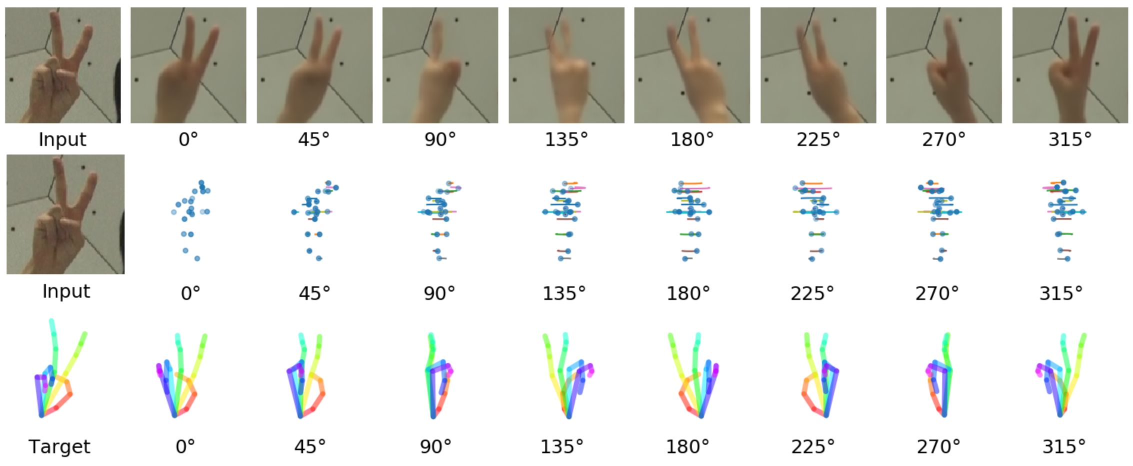
(e) 100% of S0 (6 512 annotations)



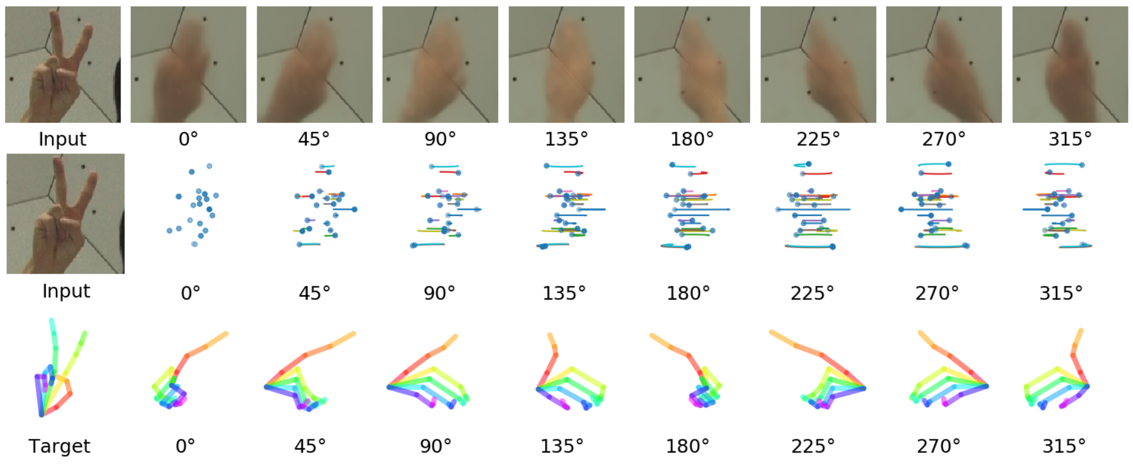
(f) 50% of S0 (3 248 annotations)



(g) 10% of S0 (640 annotations)

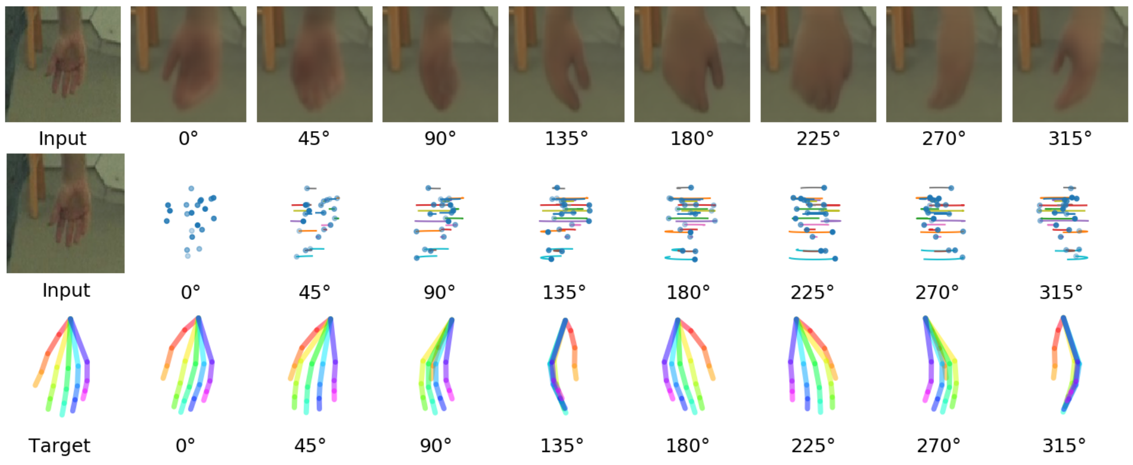


(h) 5% of S0 (320 annotations)

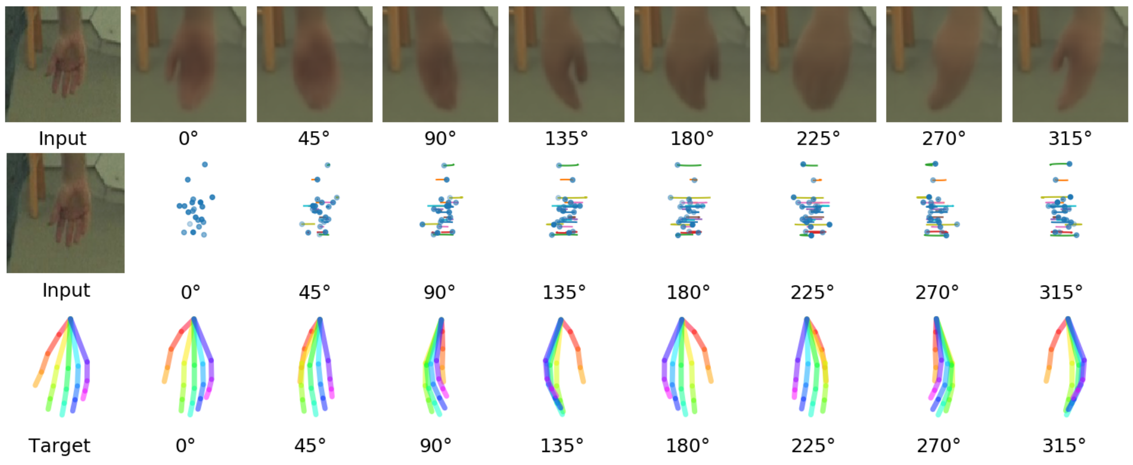


(i) 1% of S0 (64 annotations)

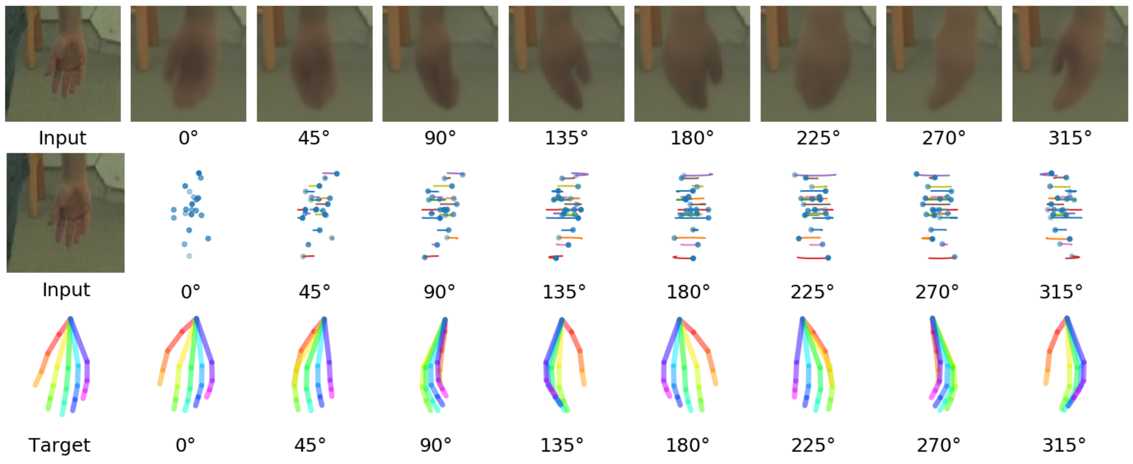
Figure 4.13: Network $\mathcal{O}$ experiments - two fingers up.



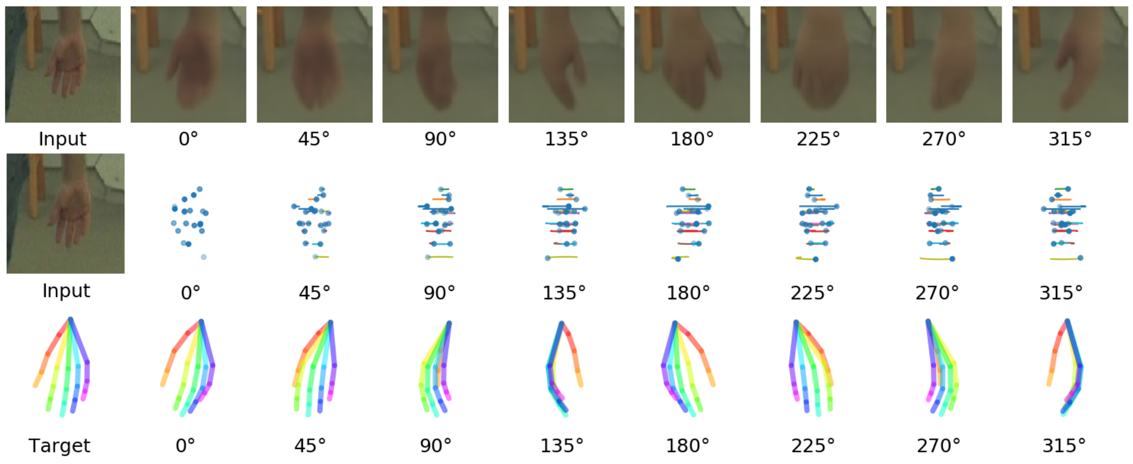
(a) All subjects (28 592 annotations)



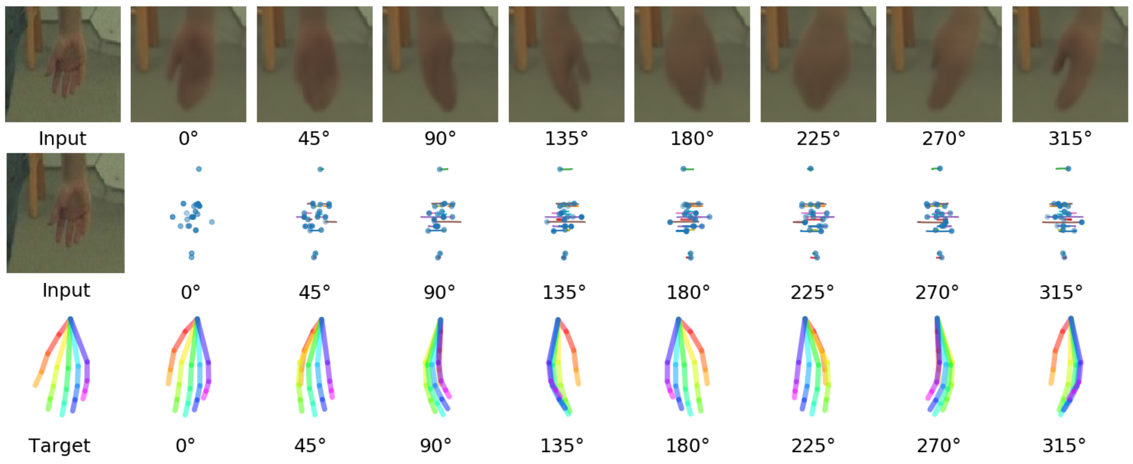
(b) S0, S3, S4, S5 and S6 (18 624 annotations)



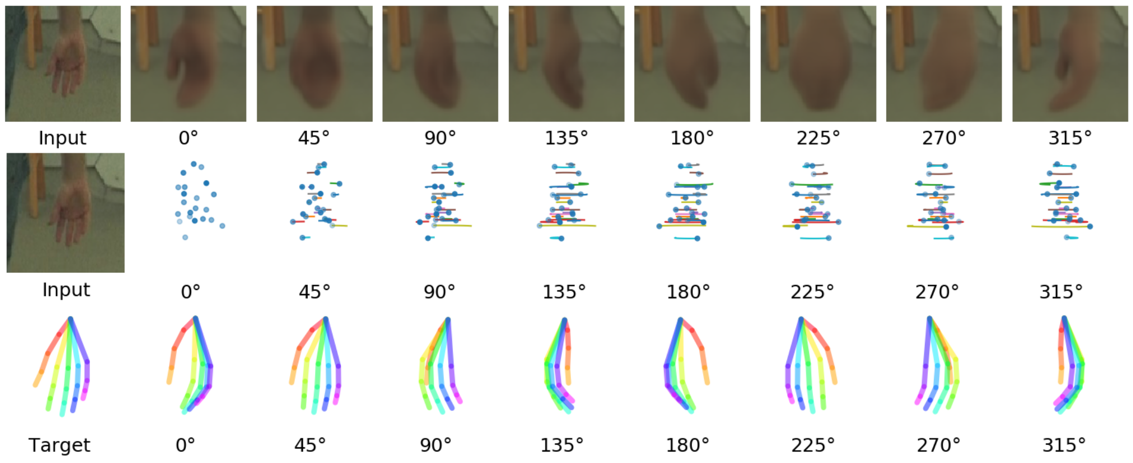
(c) S0, S3 and S4 (13 136 annotations)



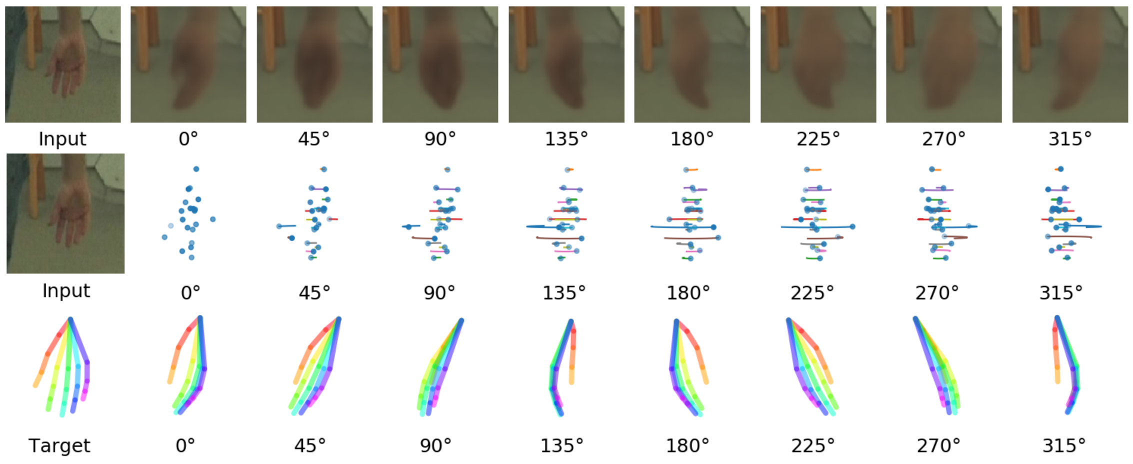
(d) S0 and S3 (9 392 annotations)



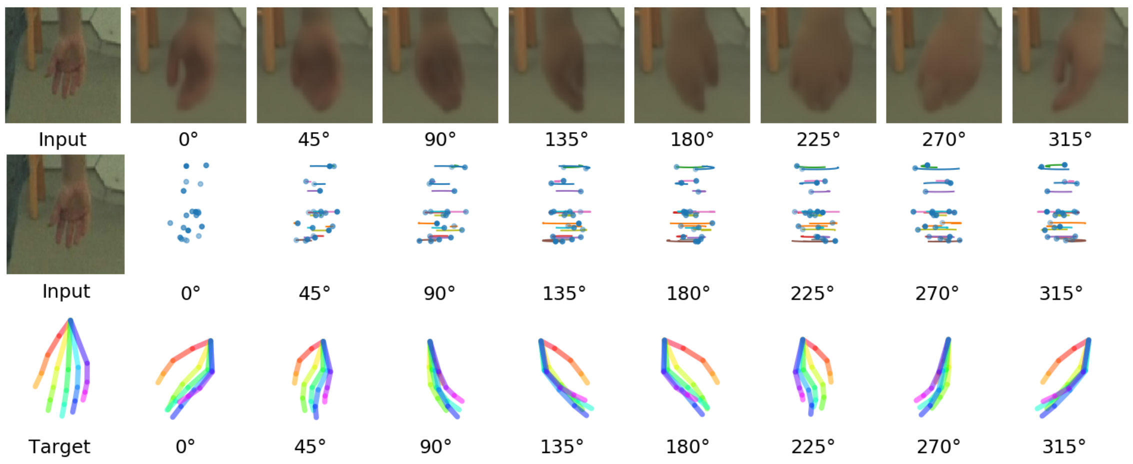
(e) 100% of S0 (6 512 annotations)



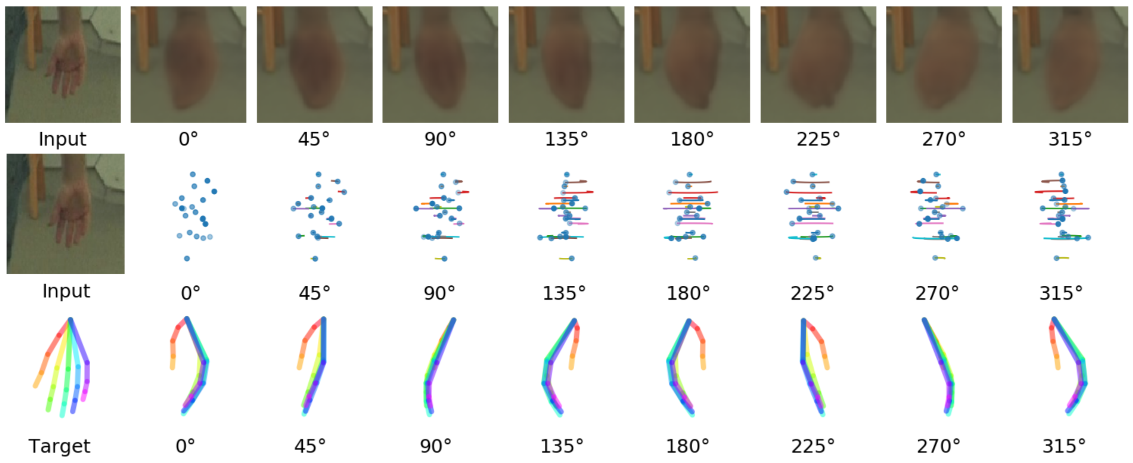
(f) 50% of S0 (3 248 annotations)



(g) 10% of S0 (640 annotations)

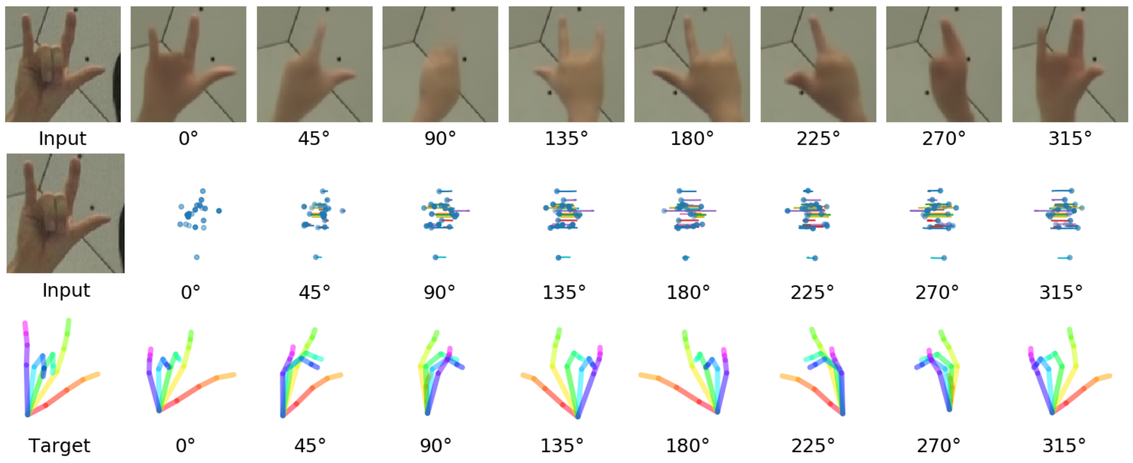


(h) 5% of S0 (320 annotations)

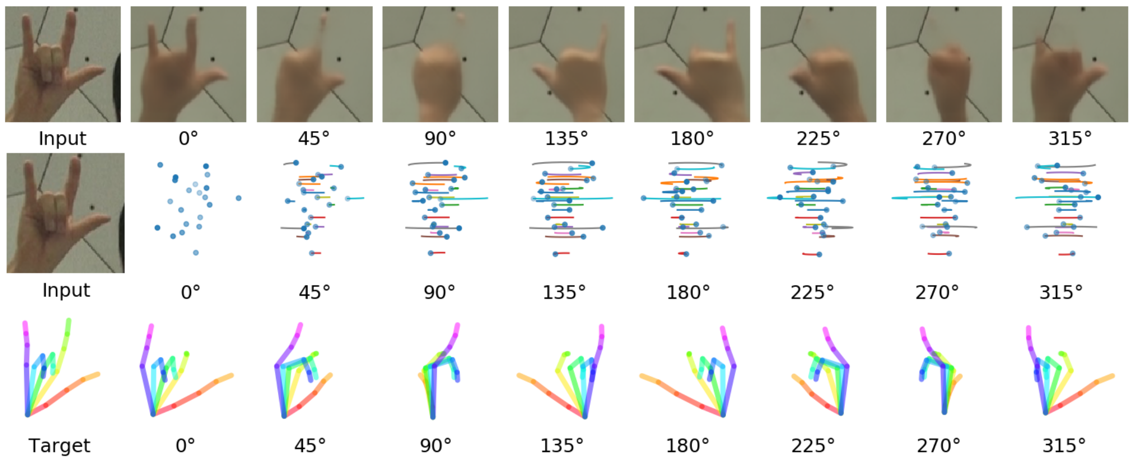


(i) 1% of S0 (64 annotations)

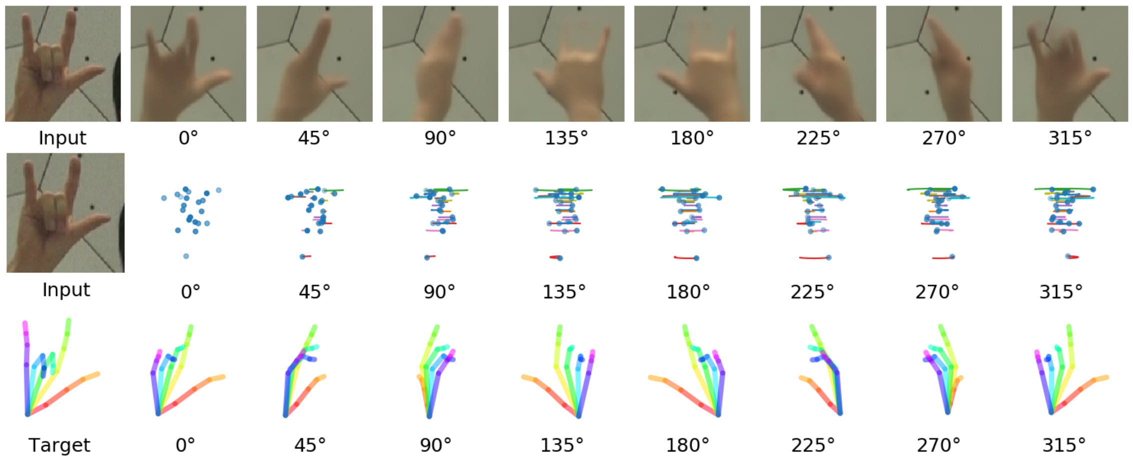
Figure 4.14: Network $\mathcal{O}$ experiments - hand hanging downward.



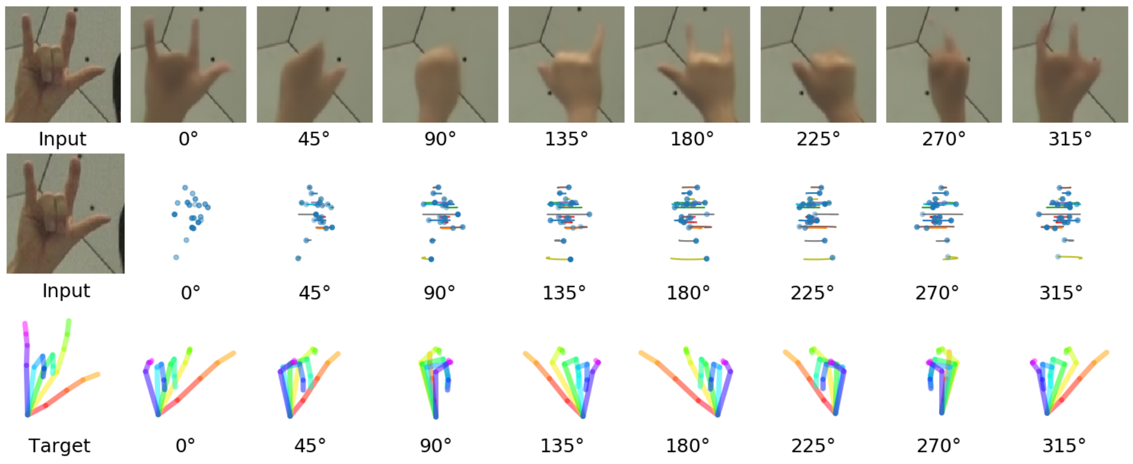
(a) All subjects (28 592 annotations)



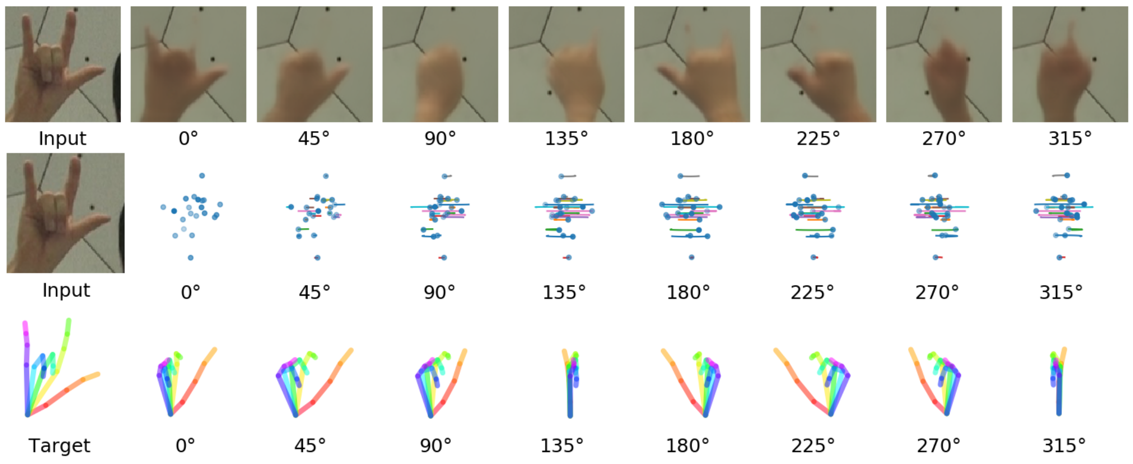
(b) S0, S3, S4, S5 and S6 (18 624 annotations)



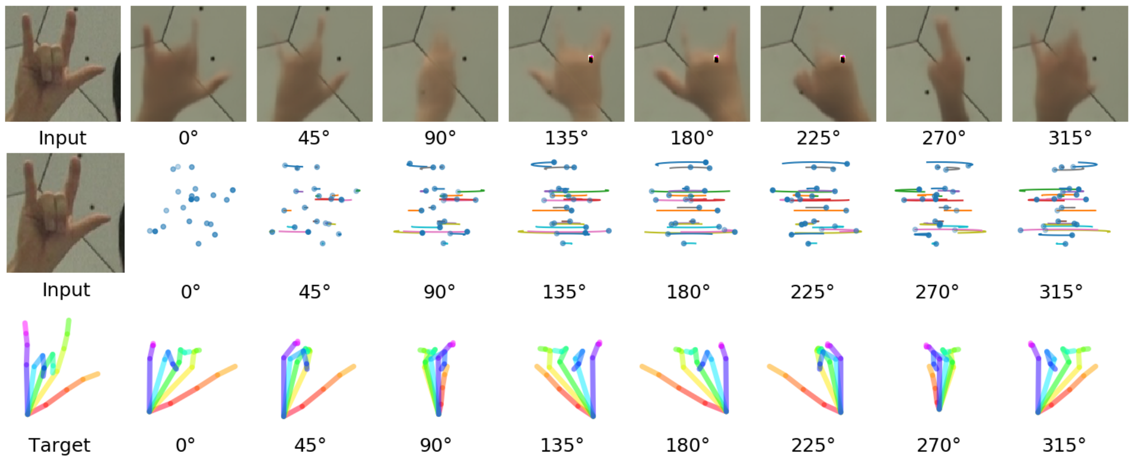
(c) S0, S3 and S4 (13 136 annotations)



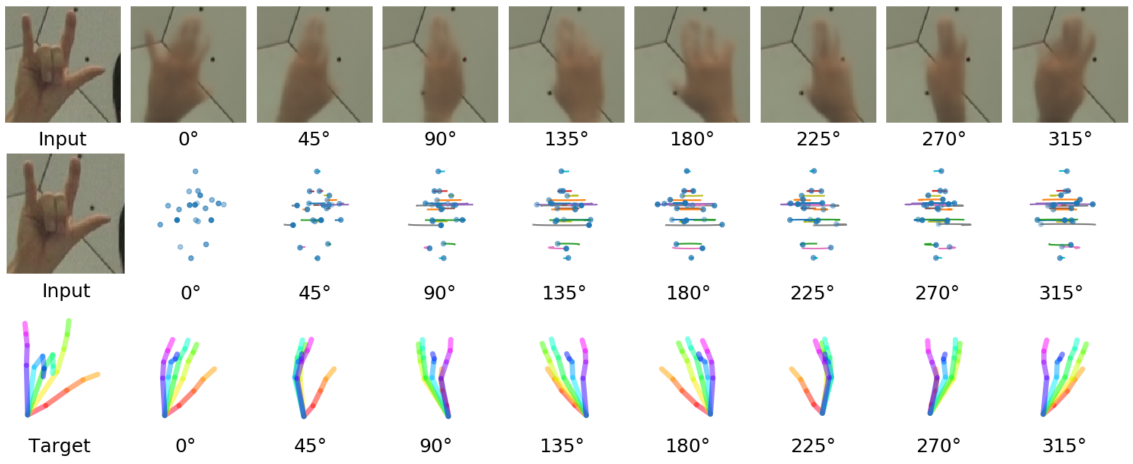
(d) S0 and S3 (9 392 annotations)



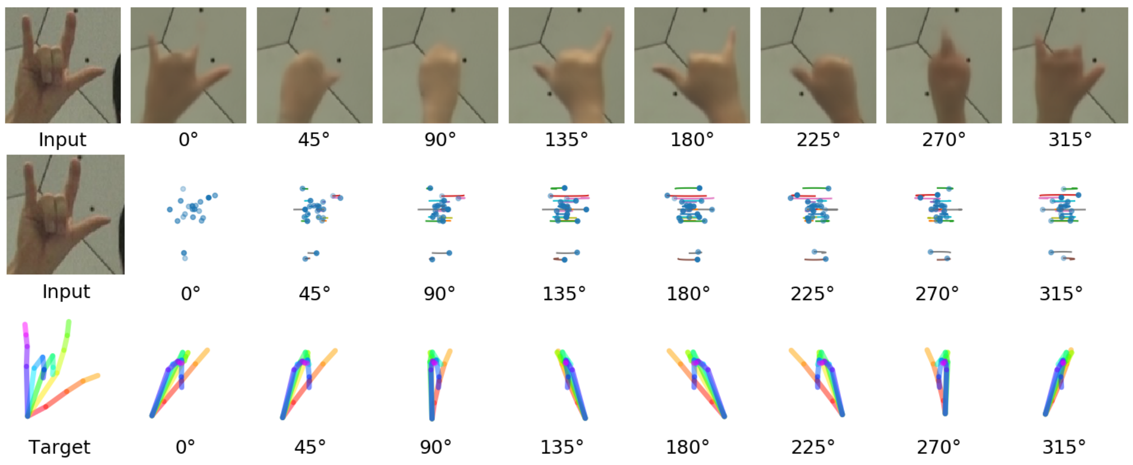
(e) 100% of S0 (6 512 annotations)



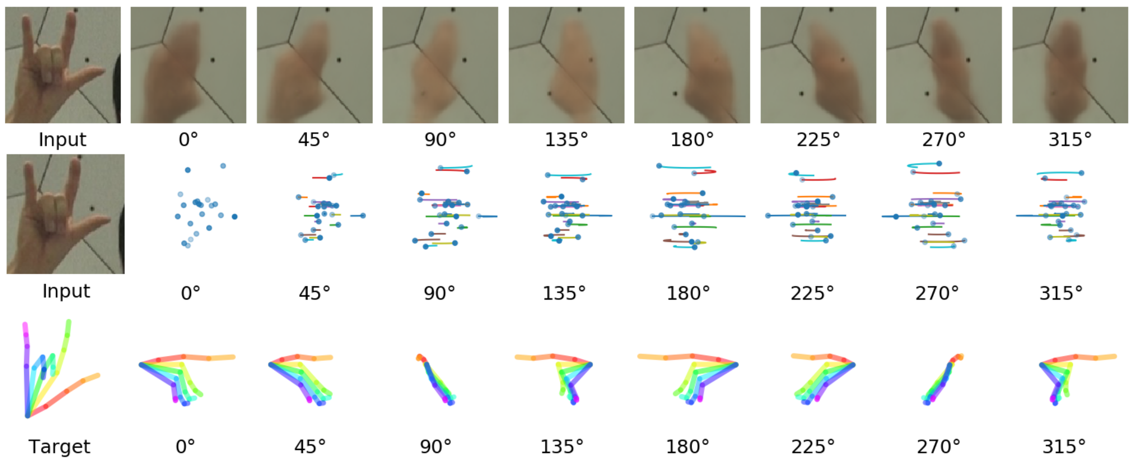
(f) 50% of S0 (3 248 annotations)



(g) 10% of S0 (640 annotations)



(h) 5% of S0 (320 annotations)



(i) 1% of S0 (64 annotations)

Figure 4.15: Network $\mathcal{O}$ experiments - three fingers up.

5 Conclusion

Why not stay until the end?

Ainz Ooal Gown (Overlord I)

Contents

5.1 Summary

In this thesis, we presented an approach for 3D hand pose estimation, that was previously used for 3D human pose estimation. We adapted and optimized this idea, to accomplish the task of 3D hand pose estimation, as described in Chapter 3. More specifically, we trained an encoder-decoder network with only unlabeled multi-view images in an unsupervised manner to learn a geometry-aware latent representation. This geometry-aware hand representation is effective as an intermediate representation for Novel View Synthesis (NVS) and for 3D hand pose estimation. We used this powerful latent representation to learn a mapping to the 3D pose in a supervised manner. The mapping to the 3D pose was much simpler, since the latent representation already captured 3D geometry. Furthermore, it required considerably fewer examples for learning the mapping compared to many existing methods, that rely on multi-view supervision. We call this pose network $\mathcal{M}$.

In Chapter 4, we demonstrated the quality of the geometry-aware hand representation in the provided *NVS* predictions. We further compared the results for our pose networks $\mathcal{M}$, $\mathcal{N}$ and $\mathcal{O}$. Network $\mathcal{N}$ directly maps an input image to the 3D pose, without pre-training the encoder with unlabeled multi-view images. The parameters in the encoder and the pose network were optimized simultaneously. Network $\mathcal{O}$ is a semi-supervised approach, where the encoder-decoder network and the pose network were trained simultaneously.

We showed with network $\mathcal{M}$, that using a geometry-aware representation in a semi-supervised approach for 3D hand pose estimation performs much better than methods, that do not use this intermediate step, when only little annotated data is available. The performance even increased, as we used augmented multi-view images to train the encoder-decoder network. Consequently, we can to a certain extend say, that the more unlabeled multi-view images with a large variety of poses is available to learn the geometry-aware representation, the better the 3D hand pose predictions will be, if the number of annotations is limited.

Our experiments further showed, that when the amount of annotated hand images is small, the accuracy for network $\mathcal{O}$ is worse than for network $\mathcal{M}$ and we achieved the worst performance, when we tried to map an input image directly to the 3D pose as in network $\mathcal{N}$. Surprisingly, however, when we had more than 13 000 annotations available during training, networks $\mathcal{N}$ and $\mathcal{O}$ outperformed the tested variations of network $\mathcal{M}$.

In conclusion, we can say that learning a geometry-aware hand representation solely from unlabeled multi-view images is very effective as an intermediate representation for a 3D hand pose estimation network, when only a limited amount of annotated hand images is available. An approach, that learns to directly a map an input image to the 3D hand pose will be recommended, if the amount of annotated images is huge.

5.2 Future Work

There are several directions for future work that naturally arise from the work presented in this thesis. One improvement to the geometry-aware latent representation would be to acquire a much larger amount of unlabeled multi-view images. This would also increase the diversity of hand poses. Our 3D hand pose estimation results already improved, when we trained the encoder-decoder network with augmented unlabeled multi-view images.

A separate hand segmentation network for the background estimation would definitely improve the *NVS* and probably the 3D pose estimation network as well. We used a pixel-wise median filter, as described in Chapter 3, and some artifacts in a few background estimations were still left. These impurities prevented our encoder-decoder network to properly learn a geometry-aware hand representation.

A further improvement could be to increase the number of annotations using synthetic data and the 3D hand pose estimation pipeline could be extended to estimate the poses of multiple hands in a scene.

A

List of Acronyms

AR	Augmented Reality
AUC	Area Under the Curve
CAD	Computer-Aided Design
CMU	Carnegie Mellon University
CNN	Convolutional Neural Network
COCO	Common Objects in Context
D-TSDF	Directional Truncated Signed Distance Function
DCNN	Deep Convolutional Neural Network
DNN	Deep Neural Network
DOF	Degrees of Freedom
DS-CNN	Dual-Source Deep Convolutional Neural Network
FPS	Frames per Second
GAN	Generative Adversarial Network
HAC	Hierarchical Agglomerative Clustering
HCI	Human-Computer Interaction
HD	High Definition
HRI	Human-Robot Interaction
ICP	Iterative Closest Point
IDPR	Image Dependent Pairwise Relation
MoCap	Motion Capture
MPII	Max-Planck Institute for Informatics
MPJPE	Mean Per Joint Position Error
N-MPJPE	Normalized Mean Per Joint Position Error
NVS	Novel View Synthesis
PCK	Percentage of Correct Keypoints
PS	Pictorial Structure
ReLU	Rectified Linear Unit

ResNet	Residual Neural Network
ROI	Region of Interest
SfM	Structure from Motion
SotA	State-of-the-Art
TSDF	Truncated Signed Distance Function
VGA	Video Graphics Array
VR	Virtual Reality

Bibliography

[1] Y. Amit and D. Geman. Shape quantization and recognition with randomized trees. *Neural Computation*, 9(7):1545–1588, 1997. (page 15)

[2] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 8)

[3] V. Badrinarayanan, A. Kendall, and R. Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017. (page 20)

[4] S. Baek, K. I. Kim, and T. Kim. Augmented skeleton space transfer for depth-based hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 18)

[5] P. Baldi. Autoencoders, unsupervised learning, and deep architectures. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2012. (page 24)

[6] D. Bank, N. Koenigstein, and R. Giryes. Autoencoders. *Computing Research Repository (CoRR)*, abs/2003.05991, 2020, `http://arxiv.org/abs/2003.05991`. (page 24, 25)

[7] H. G. Barrow, J. M. Tenenbaum, R. C. Bolles, and H. C. Wolf. Parametric correspondence and chamfer matching: Two new techniques for image matching. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 1977. (page 9)

[8] V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, and S. Ilic. 3d pictorial structures for multiple human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 16)

[9] M. Buda, A. Maki, and M. A. Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018. (page 46)

[10] A. Buryanov and V. Kotiuk. Proportions of hand segments. *International Journal of Morphology*, 28:755 – 758, 2010. (page 32)

[11] 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 (TPAMI)*, 2019. (page 2, 8)

[12] J. Y. Chang and S. W. Nam. Fast random-forest-based human pose estimation using a multi-scale and cascade approach. *ETRI Journal*, 35(6):949–959, 2013. (page 15)

[13] L. Chen, S. Lin, Y. Xie, Y. Lin, W. Fan, and X. Xie. DGGAN: depth-image guided generative adversarial networks for disentangling RGB and depth images in 3d hand pose estimation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2020. (page 18)

[14] S. E. Chen and L. Williams. View interpolation for image synthesis. In *Proceedings of the Conference on Computer Graphics and Interactive Techniques (SIGGRAPH)*, 1993. (page 23)

[15] X. Chen and A. L. Yuille. Articulated pose estimation by a graphical model with image dependent pairwise relations. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2014. (page 11)

[16] C. Choi, Y. Taguchi, O. Tuzel, M. Liu, and S. Ramalingam. Voting-based pose estimation for robotic assembly using a 3d sensor. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2012. (page 6)

[17] T. S. Cohen and M. Welling. Transformation properties of learned visual representations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. (page 29)

[18] D. Cruz-Uribe and C. Neugebauer. Sharp error bounds for the trapezoidal rule and simpson's rule. *Journal of Inequalities in Pure & Applied Mathematics (JIPAM)*, 3, 2002. (page 45)

[19] M. Denil, D. Matheson, and N. de Freitas. Narrowing the gap: Random forests in theory and in practice. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2014. (page 16)

[20] E. Dibra, S. Melchior, A. Balkis, T. Wolf, C. Öztireli, and M. H. Gross. Monocular RGB hand pose inference from unsupervised refinable nets. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 21, 22)

[21] B. Doosti. Hand pose estimation: A survey. *Computing Research Repository (CoRR)*, abs/1903.01013, 2019, `http://arxiv.org/abs/1903.01013`. (page 1, 2, 16, 17, 18, 19, 20)

[22] M. Eichner and V. Ferrari. Better appearance models for pictorial structures. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2009. (page 10)

[23] Z. Ekhtiyari and H. A. Noughabi. Using the pictorial structures in 3d human body pose estimation. *IOSR Journal of Computer Engineering*, 18:88–94, 2016. (page 10)

[24] A. El-Hamdouchi and P. Willett. Hierarchic document clustering using ward's method. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1986. (page 41)

[25] A. M. Elgammal and C. Lee. Inferring 3d body pose from silhouettes using activity manifold learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2004. (page 15)

[26] A. Erol, G. Bebis, M. Nicolescu, R. D. Boyle, and X. Twombly. Vision-based hand pose estimation: A review. *Computer Vision and Image Understanding (CVIU)*, 108(1-2):52–73, 2007. (page 1, 2)

[27] X. Fan, K. Zheng, Y. Lin, and S. Wang. Combining local appearance and holistic view: Dual-source deep neural networks for human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. (page 11, 12)

[28] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial structures for object recognition. *International Journal of Computer Vision (IJCV)*, 61(1):55–79, 2005. (page 10)

[29] M. A. Fischler and R. A. Elschlager. The representation and matching of pictorial structures. *IEEE Transactions on Computers*, 22(1):67–92, 1973. (page 9)

[30] O. Freifeld and M. J. Black. Lie bodies: A manifold representation of 3d human shape. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. (page 15)

[31] 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 (CVPR)*, 2017. (page 18, 19)

[32] A. Ghafoor, R. N. Iqbal, and S. Khan. Robust image matching algorithm. In *Proceedings of the IEEE EURASIP Conference focused on Video/Image Processing and Multimedia Communications (EC-VIP-MC)*, 2003. (page 9)

[33] W. Gong, X. Zhang, J. Gonzàlez, A. Sobral, T. Bouwmans, C. Tu, and E. Zahzah. Human pose estimation from monocular images: A comprehensive survey. *Sensors*, 16(12):1966, 2016. (page 14, 15, 16)

[34] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. C. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2014. (page 18)

[35] 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 (CVPR)*, 2016. (page 21, 22, 31, 45)

[36] A. Hietanen, J. Latokartano, A. Foi, R. Pieters, V. Kyrki, M. Lanz, and J. Kämäräinen. Benchmarking 6d object pose estimation for robotics. *Computing Research Repository (CoRR)*, abs/1906.02783, 2019. (page 7)

[37] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 36(7):1325–1339, 2014. (page 44)

[38] H. Joo, T. Simon, X. Li, H. Liu, L. Tan, L. Gui, S. Banerjee, T. Godisart, B. C. Nabbe, I. A. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: A massively multiview system for social interaction capture. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 41(1):190–204, 2019. (page 35, 36)

[39] L. Ke, M. Chang, H. Qi, and S. Lyu. Multi-scale structure-aware network for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 14)

[40] C. Keskin, F. Kiraç, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2012. (page 16)

[41] N. U. Khan and W. Wan. A review of human pose estimation from single image. In *Proceedings of the International Conference on Audio, Language and Image Processing (ICALIP)*, 2018. (page 9, 10, 11, 12, 13)

[42] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015. (page 45)

[43] O. Kose, F. Guler, A. Turan, K. Canbora, and S. Akalin. Prevalence and distribution of sesamoid bones of the hand: A radiographic study in turkish subjects. *International Journal of Morphology*, 30:1094 – 1099, 2012. (page 2)

[44] P. Krejov, A. Gilbert, and R. Bowden. Combining discriminative and model based approaches for hand pose estimation. In *Proceedings of the IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 2015. (page 2)

[45] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2012. (page 11)

[46] J. J. Kuch and T. S. Huang. Vision based hand modeling and tracking for virtual teleconferencing and telecollaboration. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 1995. (page 16)

[47] P. Li, H. Ling, X. Li, and C. Liao. 3d hand pose estimation using randomized decision forest with segmentation index points. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015. (page 16)

[48] S. Li and A. B. Chan. 3d human pose estimation from monocular images with deep convolutional neural network. In *Proceedings of the Asian Conference on Computer Vision (ACCV)*, 2014. (page 14)

[49] H. Lin and T. Chen. Augmented reality with human body interaction based on monocular 3d pose estimation. In *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems (ACIVS)*, 2010. (page 5)

[50] T. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014. (page 8)

[51] Y. Litvak, A. Biess, and A. Bar-Hillel. Learning pose estimation for high-precision robotic assembly using simulated depth images. In *Proceedings of the International Conference on Robotics and Automation (ICRA)*, 2019. (page 5, 6)

[52] M. Lourakis and X. Zabulis. Model-based pose estimation for rigid objects. In *Proceedings of the International Conference on Computer Vision Systems (ICVS)*, 2013. (page 6, 7)

[53] S. Lu, D. N. Metaxas, D. Samaras, and J. Oliensis. Using multiple cues for hand tracking and model refinement. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2003. (page 16)

[54] D. C. Luvizon, D. Picard, and H. Tabia. 2d/3d pose estimation and action recognition using multitask deep learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 8)

[55] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval.* Cambridge University Press, 1st edition, 2008. (page 40, 41)

[56] E. Marinoiu, D. Papava, and C. Sminchisescu. Pictorial human spaces: How well do humans perceive a 3d articulated pose? In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. (page 14)

[57] 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 (CVPR)*, 2018. (page 17, 19)

[58] 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 (CVPR)*, 2018. (page 18, 20, 21)

[59] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. (page 12, 13)

[60] A. Y. Ng and M. I. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2001. (page 14, 15)

[61] H. Ning, W. Xu, Y. Gong, and T. S. Huang. Discriminative learning of visual words for 3d human pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2008. (page 15)

[62] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3d view synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 28)

[63] A. Qammaz and A. A. Argyros. MocapNET: Ensemble of SNN encoders for 3d human pose estimation in RGB images. In *Proceedings of the British Machine Vision Conference (BMVC)*, 2019. (page 5)

[64] H. Rhodin, M. Salzmann, and P. Fua. Unsupervised geometry-aware representation learning for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018. (page 3, 23, 24, 25, 27, 28, 29, 30, 31, 32, 33, 34)

[65] R. Rosales and S. Sclaroff. Combining generative and discriminative models in a framework for articulated pose estimation. *International Journal of Computer Vision (IJCV)*, 67(3):251–276, 2006. (page 16)

[66] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. ImageNet large scale visual recognition challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. (page 31)

[67] G. Salvietti, G. Gioioso, M. Malvezzi, D. Prattichizzo, A. Serio, E. Farnioli, M. Gabiccini, A. Bicchi, I. Sarakoglou, N. G. Tsagarakis, and D. G. Caldwell. HANDS.DVI: A device-independent programming and control framework for robotic hands. In *Gearing Up and Accelerating Cross-fertilization between Academic and Industrial Robotics Research in Europe: - Technology Transfer Experiments from the ECHORD Project*, 2014. (page 2)

[68] M. Salzmann and R. Urtasun. Combining discriminative and generative methods for 3d deformable surface and articulated pose reconstruction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2010. (page 16)

[69] N. Sarafianos, B. Boteanu, B. Ionescu, and I. A. Kakadiaris. 3D human pose estimation: A review of the literature and analysis of covariates. *Computer Vision and Image Understanding (CVIU)*, 152:1–20, 2016. (page 14, 15, 16)

[70] SciPy 1.0 Contributors, P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, and P. van Mulbregt. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17(3):261–272, 2020. (page 41)

[71] H. Shi and Z. Wang. Improved stacked hourglass network with offset learning for robust facial landmark detection. In *Proceedings of the IEEE International Conference on Information Science and Technology (ICIST)*, 2019. (page 14)

[72] N. Shimada, Y. Shirai, Y. Kuno, and J. Miura. Hand gesture estimation and model refinement using monocular camera - ambiguity limitation by inequality constraints. In *Proceedings of the IEEE International Conference and Workshops on Automatic Face and Gesture Recognition (FG)*, 1998. (page 16)

[73] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011. (page 16)

[74] L. Sigal. Human pose estimation. In *Computer Vision: A Reference Guide*, pages 362–370. Springer, 2014. (page 7, 10)

[75] L. Sigal, A. O. Balan, and M. J. Black. Combined discriminative and generative articulated pose and non-rigid shape estimation. In *Proceedings of the Conference on Neural Information Processing Systems (NIPS)*, 2007. (page 16)

[76] T. Simon, H. Joo, I. A. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. (page 2, 17, 20)

[77] A. Sinha, C. Choi, and K. Ramani. DeepHand: Robust hand pose estimation by completing a matrix imputed with deep features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. (page 18)

[78] D. Tang, H. J. Chang, A. Tejani, and T. Kim. Latent regression forest: Structured estimation of 3d articulated hand posture. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 16)

[79] D. Tang, H. J. Chang, A. Tejani, and T. Kim. Latent regression forest: Structured estimation of 3d hand poses. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(7):1374–1387, 2017. (page 1)

[80] D. Tang, T. Yu, and T. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2013. (page 16)

[81] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Single-view to multi-view: Reconstructing unseen views with a convolutional network. *Computing Research Repository (CoRR)*, abs/1511.06702, 2015. (page 28, 29)

[82] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Multi-view 3d models from single images with a convolutional network. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2016. (page 28, 29)

[83] J. Tompson, M. Stein, Y. LeCun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (TOG)*, 33(5):169:1–169:10, 2014. (page 1)

[84] A. Toshev and C. Szegedy. DeepPose: Human pose estimation via deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014. (page 10, 11)

[85] I. Ulusoy and C. M. Bishop. Generative versus discriminative methods for object recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005. (page 14, 15, 16)

[86] C. Wan, T. Probst, L. V. Gool, and A. Yao. Dense 3d regression for hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 16, 17)

[87] C. Wang, D. Xu, Y. Zhu, R. M. Martin, C. Lu, L. Fei-Fei, and S. Savarese. DenseFusion: 6d object pose estimation by iterative dense fusion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. (page 7)

[88] J. H. Ward. Hierarchical grouping to optimize an objective function. *Journal of the American Statistical Association*, 58(301):236–244, 1963. (page 41)

[89] D. Werner, A. Al-Hamadi, and P. Werner. Truncated signed distance function: Experiments on voxel size. In *Proceedings of the International Conference Image Analysis and Recognition (ICIAR)*, 2014. (page 19)

[90] D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Interpretable transformations with encoder-decoder networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 29)

[91] G. Xu, Y. Zong, and Z. Yang. *Applied data mining*. CRC Press, 1st edition, 2013. (page 41)

[92] Y. Yang and D. Ramanan. Articulated human detection with flexible mixtures of parts. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 35(12):2878–2890, 2013. (page 10, 11)

[93] J. S. Yoon, K. Kim, O. Gallo, H. S. Park, and J. Kautz. Novel view synthesis of dynamic scenes with globally coherent depths from a monocular camera. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. (page 24)

[94] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Y. Chang, K. M. Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, J. Yuan, X. Chen, G. Wang, F. Yang, K. Akiyama, Y. Wu, Q. Wan, M. Madadi, S. Escalera, S. Li, D. Lee, I. Oikonomidis, A. A. Argyros, and T. Kim. Depth-based 3d hand pose estimation: From current achievements to future goals. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. (page 17)

[95] M. L. Zepeda-Mendoza and O. Resendis-Antonio. Hierarchical agglomerative clustering. In *Encyclopedia of Systems Biology*, pages 886–887. Springer, 2013. (page 40)

[96] T. Zhang, H. Lin, Z. Ju, and C. Yang. Hand gesture recognition in complex background based on convolutional pose machine and fuzzy gaussian mixture models. *International Journal of Fuzzy Systems (IJFS)*, 22(4):1330–1341, 2020. (page 17)

[97] Z. Zhang. Microsoft kinect sensor and its effect. *IEEE MultiMedia*, 19(2):4–10, 2012. (page 5)

[98] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei. Towards 3d human pose estimation in the wild: A weakly-supervised approach. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 32)

[99] J. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. (page 18, 21)

[100] 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 (ICCV)*, 2017. (page 18, 20)