Deep Learning-based 3D Hand Pose and Shape Estimation from a Single Depth Image: Methods, Datasets and Application

Thesis

approved by
the Department of Computer Science
at the Technische Universität Kaiserslautern
for the award of the Doctoral Degree
Doctor of Engineering (Dr.-Ing.)

to

Muhammad Jameel Nawaz Malik

Date of Defense: November 11, 2020

Dean: Prof. Dr. Jens Schmitt

Reviewer: Prof. Dr. Didier Stricker

Reviewer: Prof. Dr. Antonis Argyros

Reviewer: Prof. Dr. Karsten Berns

D 386

Abstract

3D hand pose and shape estimation from a single depth image is a challenging computer vision and graphics problem with many applications such as human computer interaction and animation of a personalized hand shape in augmented reality (AR). This problem is challenging due to several factors for instance high degrees of freedom, view-point variations and varying hand shapes. Hybrid approaches based on deep learning followed by model fitting preserve the structure of hand. However, a pre-calibrated hand model limits the generalization of these approaches. To address this limitation, we proposed a novel hybrid algorithm for simultaneous estimation of 3D hand pose and bone-lengths of a hand model which allows training on datasets that contain varying hand shapes. On the other hand, direct joint regression methods achieve high accuracy but they do not incorporate the structure of hand in the learning process. Therefore, we introduced a novel structure-aware algorithm which learns to estimate 3D hand pose jointly with new structural constraints. These constraints include fingers lengths, distances of joints along the kinematic chain and fingers inter-distances. Learning these constraints help to maintain a structural relation between the estimated joint keypoints. Previous methods addressed the problem of 3D hand pose estimation. We open a new research topic and proposed the first deep network which jointly estimates 3D hand shape and pose from a single depth image. Manually annotating real data for shape is laborious and sub-optimal. Hence, we created a million-scale synthetic dataset with accurate joint annotations and mesh files of depth maps. However, the performance of this deep network is restricted by limited representation capacity of the hand model. Therefore, we proposed a novel regression-based approach in which the 3D dense hand mesh is recovered from sparse 3D hand pose, and weak-supervision is provided by a depth image synthesizer. The above mentioned approaches regressed 3D hand meshes from 2D depth images via 2D convolutional neural networks, which leads to artefacts in the estimations due to perspective distortions in the images. To overcome this limitation, we proposed a novel voxel-based deep network with 3D convolutions trained in a weakly-supervised manner. Finally, an interesting application is presented which is in-air signature acquisition and verification based on deep hand pose estimation. Experiments showed that depth itself is an important feature, which is sufficient for verification.

Acknowledgement

I would like to thank all the committee members and especially my doctoral supervisor Prof. Dr. Didier Stricker for his continuous support and encouragement throughout my PhD time period.

I am highly grateful to Dr. Ahmed Elhayek, the former senior researcher at DFKI Kaiserslautern, for encouraging and guiding me throughout my work. His effective and continuous support helped me to better accomplish the tasks during my PhD. I am also thankful to my co-authors Fabrizio Nunnari, Soshi Shimada, Sk Aziz Ali and Vladislav Golyanik, for a fruitful collaborative work.

I would also like to thank the secretariat of the augmented vision (AV) department for helping me to smoothly carry out the necessary paper works.

Generally, I thank everybody at the AV department for chats, discussions, company, exciting conference visits, and for creating this great place to work.

Last and most importantly, I would like to thank my all family members, especially my mother Nighat Shaheen and my father Muhammad Muneer Malik, for understanding and supporting me, praying for me, and always motivating me to achieve my goals.

Contents

Chapter 1

Introduction

Contents

1.1. Problem Statement

Vision-based markerless 3D hand pose estimation is an essential requirement for many applications of Human-Computer Interaction (HCI), Virtual Reality (VR), and Augmented Reality (AR). Whereas, the dense 3D hand mesh is a richer representation of human hand which is more useful than the sparse 3D hand pose. This dense representation finds many applications in computer vision and graphics, such as immersive interaction in VR and AR, handling and manipulating with virtual objects, as well as modern gaming environments. Some interesting applications of hand pose and shape estimation are shown in Figure 1.1. These applications require the estimation methods to be robust to occlusions, hand shape and size, background clutter, and camera noise. Further, they require a high accuracy and high speed for interactive applications. Hand pose estimation has been widely studied by the Computer Vision (CV) community, especially, in the last decade [keskin2012hand; oikonomidis2011full; tang2013real; sridhar2013interactive; mueller2018ganerated; moon2017v2v]. Vision-based simultaneous 3D hand mesh and pose estimation is a newly emerging research problem which has recently attracted an attention of the CV community [ge20193d; Adnane20193d; mueller2019real].

Estimating 3D hand pose from a single RGB image is a very challenging and ill-posed problem mainly, due to the ambiguity of monocular 2D images.

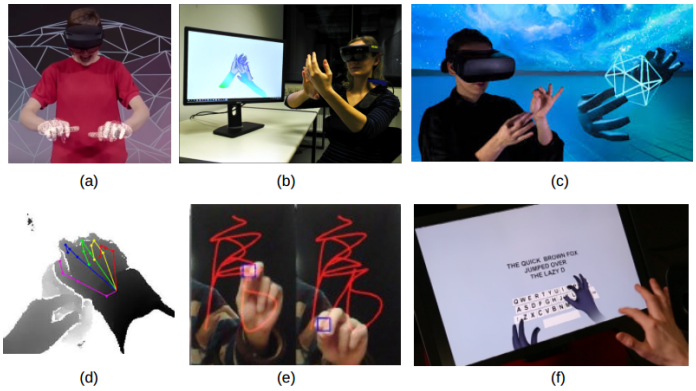


Figure 1.1.: *Some applications of hand shape and pose estimation. (a) Microsoft HoloLens 2 (b) Hands interaction [mueller2019real] (c) Leap Motion (d) Action recognition [garcia2018first] (e) In-Air Signature [fang2017novel] (f) Typing in VR [taylor2016efficient].*

With the arrival of low-cost and commercially available depth sensors (e.g. Microsoft Kinect and Intel RealSense cameras), inferring 3D joint locations became feasible. Industries such as Leap Motion and Oculus also developed VR headsets, however, their systems still lack in accuracy and robustness which is essential for many real-world applications. In the past few years, huge progress has been achieved in deep learning that demonstrates the power of learning knowledge from fully or partially annotated data. The success of depth cameras and deep learning has led to the rapid progress of 3D hand pose estimation in the recent years. On the other hand, progress in the essential task such as estimation of 3D hand mesh is hampered, as manual supervision for such a problem at large scale is extremely expensive. More specifically, annotating real images for full shape is highly time consuming, laborious and sub-optimal.

In this work, we aim to solve the unique challenges of hand shape and pose estimation, by establishing a deep learning based mapping from an input depth image to the 3D hand mesh vertices and 3D joint positions. For kinematic stability of the estimated pose and an improved generalized performance, we also estimate certain intermediate parametric representations of a hand model such as joint angles, bone vectors, bone-lengths and the structural constraints. Provided that there is no ground truth 3D hand mesh information available in public benchmarks, we create the first million-scale synthetic datasets (for both 1st person and 3rd person viewpoints) that contain depth images with accurate annotations of 3D hand joint positions, joint angles, 3D mesh vertices and hand parts segmentation. Our datasets open up new possibilities for advanced hand analysis.

The input to our hand pose and shape estimation algorithm is a single depth map acquired from a Time-of-Flight (ToF) camera. A depth map is essentially the projection of 3D points on the hand surface projected to a camera plane where each pixel value is the distance of the point to the camera plane.

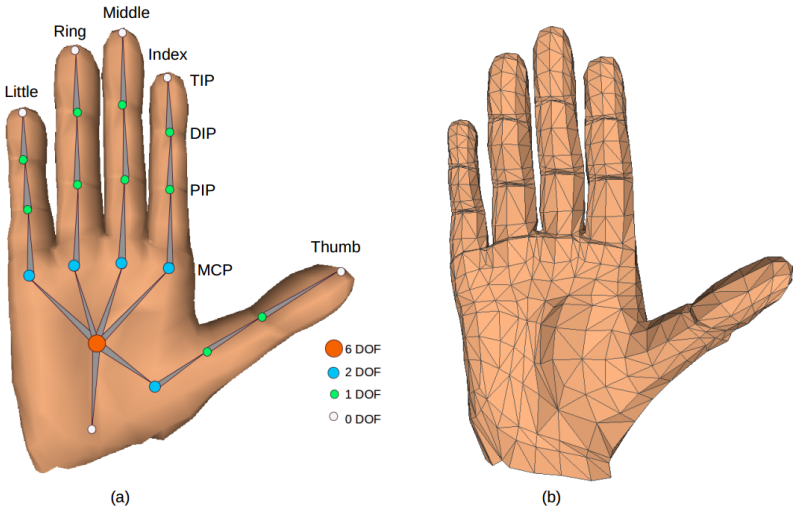


Figure 1.2.: *(a) Illustration of a hand skeleton which contains 22 3D joints and a total of 26 DOF. (b) shows a hand mesh which contains 1193 3D vertices.*

The quality of the depth map changes with the characteristics of the sensor noise (Section 1.2). The outputs of our algorithm are 3D joint locations of a hand skeleton (i.e. 3D Pose) and 3D vertex locations of a hand mesh (i.e. 3D Shape) as shown in Figure 1.2. In a typical human hand representation [lin2000modeling], each finger consists of three joints: the MetaCarpoPhalangeal (MCP), Proximal InterPhalangeal (PIP), and Distal InterPhalangeal (DIP) joint. A specific Degrees of Freedom (DOF) is assigned to each joint (Figure 1.2(a)). The root joint has 6 DOF for global rotation and global translation. The MCP joints have two DOF whereas, DIP and PIP joints have one DOF. By summation, this hand skeleton contains a total of 26 DOF.

1.2. Challenges

Markerless 3D hand pose and shape estimation is a difficult problem due to several challenging factors such as many degrees of freedom, severe occlusions, varying hand shapes and sizes and noisy data.

Many Degrees of Freedom: The hand is a complex and highly articulated object that exhibits large number of possible DOF. For global viewpoint and global translation, the hand root (palm center or wrist position) has 6 DOF (i.e. 3 DOF for global translation and the other 3 DOF for global orientation). For the articulation of fingers, several DOF for the joints are required to cover a huge range of complex hand poses. This large pose space coverage is especially hard for learning-based pose estimation methods, since they need a large amount of representative training data that ideally covers a full range of complex hand poses. The variations in viewpoints and articulations together make the hand pose estimation challenging and complex than that of rigid

objects.
Occlusions: Occlusion of hand parts while making variations in the viewpoint and articulations can pose significant problems in accurate hand pose and shape estimation. The pose of the occluded parts can not directly obtained using the other visible parts of hand because of many DOF and independent movements of the parts. Therefore, the joint locations of the occluded parts are normally inferred by exploiting the kinematic hierarchy or a statistical prior on the pose and shape space. Specially, egocentric viewpoints are highly challenging due to severe occlusion of fingers. These occlusions not only make the estimations hard, but also annotating real images for both pose and shape becomes extremely difficult.
Varying Hand Shapes and Sizes: Human hands differ in shapes and sizes with respect to age, gender, ethnicity, etc. According to the anthropometric surveys [gordon20142012; mclain2010use], hand lengths vary from 145mm to 215mm for women and 144mm to 231mm for men. Also, the shape of hand can also vary significantly, for example the length of middle finger can show variation in ranges 64mm to 77mm for women and 56mm to 68mm for men. Other factors which can influence the shape are hand mass, palm width and fingers inter-distances. This high variation requires the training and testing datasets to cover wide range of hand shapes and sizes for robustness and generalized performance. Moreover, there is a need to explicitly consider such differences in the learning-based estimation methods so that they are trained to be invariant to the different shapes.
Noisy Data: 3D hand pose and shape estimation methods need to be robust to noisy data. The images acquired from camera contain noise which is characterised according to the sensor type. The acquired depth images contain noisy edges, missing depth measurements and hanging pixels [tompson2014real; yuan2017bighand2]. In case of RGB images, shadows can appear and there can be artifacts due to different lighting conditions. Synthetic counterparts of real data show clear differences due to noise. This discrepancy between real and synthetic data causes difficulty in learning accurate pose and shape using from real images. Moreover, it is very hard to accurately annotate such a noisy real data. Thereby, this causes the annotations to be noisy or erroneous which can lead to convergence problems during training.

1.3. Thesis Outline

This thesis is structured into three parts.
Part I: CNN-Based Structured Hand Pose Estimation. In the first part we present CNN-based methods which explicitly incorporate the kinematic structure of hand skeleton to accurately predict 3D hand pose from a single depth image (Chapter 3).

In a hybrid approach where the forward kinematics layer is embedded inside a deep network, we show that simultaneous learning of pose and bone-lengths parameters is essential for generalized performance over varying hand shapes and sizes. Prediction of both the parameters allows to automatically adapt

the pose and shape of the hand skeleton thereby, leading to accurate 3D pose estimation.

Further, we propose a structure-aware CNN-based algorithm which learns to automatically segment the hand from a raw depth image and estimates 3D hand pose jointly with new structural constraints. The constraints include fingers lengths, distances of joints along the kinematic chain and fingers inter-distances. Learning these constraints help to maintain a structural relation between the estimated joint keypoints. Also, we convert sparse representation of hand skeleton to dense by performing n-points interpolation between the pairs of parent and child joints. The structural constraints and the interpolation can be easily used to improve the hand pose estimation accuracy of any CNN-based discriminative approach.

Part II: Deep Learning-based Hand Shape and Pose Estimation. In the second part we present 2D and 3D convolutions based approaches to accurately estimate 3D hand shape and pose from a single depth image.

In Chapter 4 we present the first real-time model-based method to accurately estimate the 3D hand shape and pose from a single depth image. We propose a new hand pose and shape layer (HPSL) which is embedded inside our deep network. This layer takes the estimated hand pose, bone scales and complex shape parameters as inputs, and outputs 3D joint positions and hand mesh. Moreover, we propose the first million-scale synthetic hand pose and shape dataset (i.e., SynHand5M) which provides accurate joint annotations, segmentation masks and mesh files of 5 million depth maps.

However, the performance of the model-based method could be restricted by the limited shape representation capacity of the statistical deformable hand model. Therefore, in Chapter 5, we introduce two different real-time structured approaches for accurate 3D hand shape and pose estimation by using 2D convolutional networks. The first is a weakly-supervised approach which consists of three novel components: (i)- Convolutional Neural Network (CNN) based deep network which produces 3D joints positions from learned 3D bone vectors using a new layer, (ii)- a novel shape decoder that recovers dense 3D hand mesh from sparse joints, (iii)- a novel depth synthesizer which reconstructs 2D depth image from 3D hand mesh. The second is a simple and effective direct regression approach to simultaneously estimate 3D hand shape, pose and structure constraints for both egocentric and 3rd person viewpoints. In this regard, we propose the first million-scale egocentric synthetic dataset, called SynHandEgo, which contains 1 million egocentric depth images with accurate shape and pose annotations, and color segmentation of hand parts.

The above mentioned 2D convolutions based approaches treat depth map as 2D data. Consequently, the deep network is likely to produce perspective distortions in the shape and pose estimations. To address this limitation, in Chapter 6, we propose the first 3D convolutions based architecture which establishes a one-to-one mapping between the voxelized depth map and the voxelized hand shape. This one-to-one mapping allows to more accurately reconstruct the hand shapes. For shape estimation, our architecture produces two different hand shape representations. The first is the 3D voxelized grid of

the shape which is accurate but does not preserve the mesh topology and the number of mesh vertices. The second representation is the 3D hand surface which is less accurate but does not suffer from the limitations of the first representation. To combine the advantages of these two representations, we register the hand surface to the voxelized hand shape. Moreover, our 3D data augmentation on voxelized depth maps allows to further improve the accuracy of 3D hand pose estimation.
Part III: In-Air Signature Verification. This part presents an interesting practical application of 3D hand pose estimation for In-Air signature verification (Chapter 7). We are the first to propose a 3D hand pose estimation based approach to accurately record signatures by making free hand movements in the air. Further, we explore the potential in the hidden depth feature of in-air signature trajectory and show that this feature itself is sufficient for in-air signature verification. In this regard, we propose a new dataset which consists of 600 signatures recorded from 15 different subjects. This dataset specifically includes explicit variations in the depth patterns of the collected signatures from the subjects (persons).

1.4. List of Publications

The thesis is supported by the following list of publications.
Journal Publications:

1. Jameel Malik, Ahmed Elhayek, Didier Stricker. WHSP-Net: A Weakly-Supervised Approach for 3D Hand Shape and Pose Recovery from a Single Depth Image. *In MDPI Sensors Journal, 2019.*

2. Jameel Malik, Ahmed Elhayek, Fabrizio Nunnari, Didier Stricker. Simple and Effective Deep Hand Shape and Pose Regression from a Single Depth Image. *In ELSEVIER Computers and Graphics Journal, 2019.*

3. Jameel Malik, Ahmed Elhayek, Sheraz Ahmed, Faisal Shafait, Muhammad Imran Malik, Didier Stricker. 3DAirSig: A Framework for Enabling In-Air Signatures Using a Multi-Modal Depth Sensor. *In MDPI Sensors Journal, 2018.*

Conference Publications:

1. Jameel Malik, Ibrahim Abdelaziz, Ahmed Elhayek, Soshi Shimada, Sk Aziz Ali, Vladislav Golyanik, Christian Theobalt, Didier Stricker. HandVoxNet: Deep Voxel-Based Network for 3D Hand Shape and Pose Estimation from a Single Depth Map. *Submitted to Computer Vision and Pattern Recognition (CVPR), 2020.*

2. Jameel Malik, Ahmed Elhayek, Didier Stricker. Structure-Aware 3D Hand Pose Regression from a Single Depth Image. *In Proceedings of the International Conference on Virtual Reality and Augmented Reality (EuroVR), 2018.*

3. Jameel Malik, Ahmed Elhayek, Fabrizio Nunnari, Kiran Varanasi, Kairash Tamaddon, Alex Heloir, Didier Stricker. DeepHPS: End-to-end Estimation of 3D Hand Pose and Shape by Learning from Synthetic Depth. *In IEEE International Conference on 3DVision 2018.*

4. Jameel Malik, Ahmed Elhayek, Didier Stricker. Simultaneous Hand Pose and Skeleton Bone-lengths Estimation from a Single Depth Image. *In IEEE International Conference on 3DVision 2017.*

Chapter 2

Related Work

Contents

Articulated hand pose estimation from a monocular RGB image or a single depth image has been actively studied in the past few years, especially due to the recent progress in deep learning. In this chapter, we discuss the state-of-the-art works related to hand pose estimation and simultaneous hand shape and pose estimation. Also, we discuss the state-of-the-art methods related to an interesting practical application of 3D hand pose estimation i.e., In-Air signature verification.

2.1. Hand Pose Estimation from Depth Images

In general, hand pose estimation methods can be divided into three categories, namely, generative methods, discriminative methods, and hybrid methods.

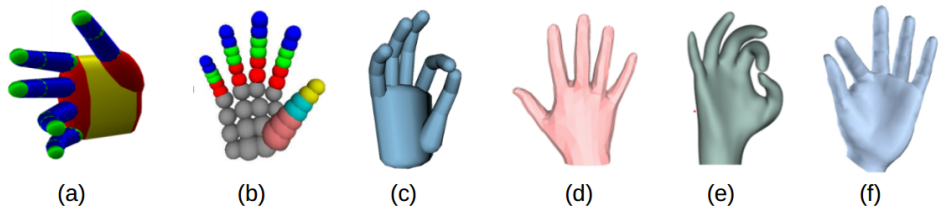


Figure 2.1.: *Hand Models. (a) Cylinder model [oikonomidis2011efficient] (b) Sphere model [qian2014realtime] (c) Cylinder model [tagliasacchi2015robust] (d) MANO model [romero2017embodied] (e) Smooth surface model [taylor2016efficient] (f) Mesh model [ge20193d].*

2.1.1. Generative Methods

Hand model-based approaches such as [oikonomidis2011efficient; oikonomidis2011full; oikonomidis2012tracking; stenger2006model; qian2014realtime; de2011model; makris2015hierarchical; tagliasacchi2015robust] belong to the class of generative methods. These methods tend to find the best hand model configuration that minimizes the discrepancy between the hand hypothesis and the actual observation. These methods have introduced several different hand models, kinematics constraints, cost functions and optimization techniques. Oikonomidis et al. [oikonomidis2011efficient] propose a polygonal hand mesh model. In their hand model, palm is modeled as a cylinder and two ellipsoids, while fingers are modeled using cones for bones and spheres for joints. The hand model proposed in [qian2014realtime] used 48 spheres. Gorce et al. [de2011model] build their mesh model using 1000 facets. [tagliasacchi2015robust] used a cylindrical hand model. Figure 2.1 shows several hand models that have been proposed in the literature. Various optimization techniques have been proposed to find the optimal hand configuration. Some of the methods introduced stochastic optimization algorithms such as Kalman filter [stenger2006model] and particle filter [maccormick2000partitioned]. Oikonomidis et al. [oikonomidis2011efficient] use Particle Swarm Optimization (PSO) algorithm [kennedy1995particle] to reach the best hand configuration. Andrea et al. [tagliasacchi2015robust] use Articulated-Iterated Closest Point (ICP) for the optimization. Generative methods achieve good performance at the cost of complex energy functions optimizations. While these methods can cover a wide range of hand poses, they do not explicitly optimize for the hand shape. Thus, for optimal performance, these methods require a carefully calibrated initial hand geometry. In addition, generative methods are suitable for continuous tracking through consecutive frames with predictable and small hand movements.

2.1.2. Discriminative Methods

Discriminative methods directly regress hand pose from an RGB, depth or RGB-D image. These methods can be further categorised into two classes, namely, random forest and deep learning based methods.

Random Decision Forest (RDF): 3D hand pose estimation methods such as [tang2014latent; tompson2014real; sun2015cascaded] trained random forests on image datasets and they have shown good performance. Keskin et al. [keskin2012hand] estimate hand pose using a multi-layered random decision forest and also, they perform the shape classification using depth images. Rogez et al. [rogez20143d] propose a tracking-by-detection framework which eliminates a need for manual initialization. They considered hand detection and pose estimation as classification problems, where the classifiers are used to classify K different discrete hand poses and the background. Tang et al. [tang2014latent] propose an Latent Regression Forest (LRF) approach for real-time 3D hand pose estimation from a single depth image. They recursively divide the input depth image into sub-regions, until each sub-region contains one joint. Tang el al. [tang2015opening] use separate random forests to predict partial hand poses in the kinematic hierarchy. [ji2018hierarchical] consider the hierarchical topology of the hand and use a separate random forest for each finger to predict 3D joint coordinates.

Deep Learning: With the recent advancement in deep learning, significant progress has been achieved in 3D hand pose estimation over the past few years. Tompson et al. [tompson2014real] are the first to use a CNN for hand pose estimation. They directly regress a separate 2D heatmap for each of the hand joint positions from the single CNN. They use these predicted heatmaps to estimate 3D hand pose using inverse kinematics. However, they estimate only the 2D joint positions, and the third coordinate is acquired from the depth map, which is problematic for occluded joints. Moreover, the performance of their method is restricted by the heatmap resolution. Ge et al. [ge2016robust] extend this work and employed multiple CNNs to estimate heatmaps using different projections of the depth image. However, their approach requires a multi-view fusion of the estimated heatmaps as a complex post-processing step. In a similar way, Wan et al. [wan2018dense] estimate 2D heatmaps but, in addition, they predict heatmaps which encode the proximity of the input 3D points and 3D vector offsets that point towards the 3D joint locations. Finally, to obtain the 3D joint positions, their method requires a complex post-processing step.

Deng et al. [deng2017hand3d] convert the input depth image into a 3D volumne and employ 3D CNN to estimate the 3D pose. Ge el al. [ge2017robust] generate the 3D volumetric representation of the hand pixels with projective D-TSDF from the 3D point cloud. They use a single 3D CNN which directly produces 3D hand positions from the 3D volumetric representations. Ge et al. [ge2018hand] propose Hand PointNet which directly processes the 3D point cloud of the depth map. Thereby, modeling the visible surface of the hand for pose regression. To further improve the accuracy of fingertips, they introduced a fingertip refinement network that directly takes the neighboring points of already estimated fingertip location as input to refine the fingertip location. Ge et al. [ge2018point] propose a point-to-point regression network which takes normalized 3D hand points as network input and outputs a set of heat-maps as well as unit vector fields on the input point

cloud. In the post processing step, they infer point-wise offsets to hand joints and estimate the 3D hand pose. Moon et al. [moon2017v2v] propose an effective and powerful voxel-to-voxel network which establishes a one-to-one relation between the depth map and 3D pose. They transformed the input depth image into a fixed resolution voxelized form and employ a 3D CNN to estimated 3D heatmaps of 3D joint positions. They demonstrate that the one-to-one mapping between the depth map and 3D pose is important for highly accurate 3D hand pose estimation. However, extending this work for shape estimation by directly regressing 3D heatmaps of mesh vertices is not feasible. Therefore, we will address this problem in this thesis and propose the first voxelized grid based approach to estimate hand shape and pose from a single depth image.

Wan et al. [wan2016hand] uses surface normals instead of depth map. However, surface normals are not directly accessible from current depth sensors and hence introduce an additional computational overhead. Neverova et al. [neverova2017hand] combine a CNN-based hand segmentation of the hand parts with regression of joint locations, but the segmentation is sensitive to the depth camera noise. Wan et al. [wan2017crossing] use Generative Adversarial Networks (GANs) and Variational Autoencoders (VAEs) to effectively learn a shared latent space for 3D hand pose estimation. However, the trainings of these several networks are performed in a complex procedure. Chen et al. [chen2018shpr] an effective end-to-end approach to estimate 3D hand pose directly from point sets. They propose a strategy that fuses information from semantic segmentation network and regression network to achieve more representative features for hand pose estimation. All these methods addressed only the problem of 3D hand pose estimation and also, they do not explicitly incorporate the kinematic constraints of the hand skeleton.

2.1.3. Hybrid Methods

These approaches tend to combine the advantages of discriminative and generative methods thereby, considering the kinematic constraints and making the hand tracking more robust. Sridhar et al. [sridhar2013interactive] propose a hybrid approach for hand tracking which combines a discriminative, part-based pose retrieval method with a generative pose estimation method. However, their method needs a complex multiview RGB camera system. Sharp et al. [sharp2015accurate] improve the robustness of the hand model fitting process with per-frame re-initializations, which are obtained by a discriminative predictor. Xu et al. [xu2017lie] exploit the kinematic hierarchy of the hand skeleton. They update and correct an initial pose along the kinematic chain using Lie-algebra. Sun et al. [sun2015cascaded] iteratively update an initial hand pose using six stages thereby, approaching the ground truth. The palm is updated in the first three stages with fingers fixed, and fingers are updated in the last three stages keeping the palm fixed. [sinha2016deephand] employ a separate regressor for each finger and estimated the spatial and temporal features which are combined in a nearest-neighbour formulation. Ye et al. [ye2016spatial] propose a spatial attention mechanism based approach which

specializes on each joint, and an additional optimization step to enforce the kinematic constraints. All these methods need multiple predictors for hand parts, and often require additional estimators for different iterations of their algorithms. Hence, a large number of regression models have to be trained and evaluated. This becomes infeasible with increasing number of joints to estimate. Particularly, Zhou et al. [zhou2016model] are the first to embed a hand model layer inside their deep network to enforce kinematics constraints on the estimated 3D pose. They learn hand joint angles from a single CNN which are further converted into 3D hand joint positions by the model layer. However, a big limitation of this work is an assumption of a fixed bone-lengths hand model geometry during the end-to-end training of their deep network. Clearly, this limitation restricts the generalization of this approach over different hand shapes and sizes. We will address this problem and come up with a new method that can be trained to estimate both the hand pose and bone-lengths of the hand skeleton.

2.2. Depth-Based Hand Pose Datasets

Here we briefly discuss several publicly available hand pose datasets which we use for training and evaluating our methods, and comparisons with the state-of-the-art works.

2.2.1. NYU Dataset

One of the most famous datasets for 3D hand pose estimation task is the NYU hand pose dataset [tompson2014real]. It contains $72,757$ training and 8252 test frames of multi-view RGB-D data. The images are captured from three different viewpoints and annotated with the 3D joint locations. The dataset was captured using the structured-light-based Primesense Carmine 1.09 sensor. Thus, the depth images show missing information as well as noisy edges, which makes the dataset very challenging. In our experiments, we use only the depth data from a single camera (i.e. frontal camera viewpoint). The dataset has accurate annotations and exhibits a high complexity of hand poses. One major shortcoming of this dataset is that the training set contains samples from a single user and the test set samples from two different users. Hence, this dataset does not contain variation in hand shapes and sizes. The ground truth contains 36 annotated joints however, the evaluation protocol for this dataset follows a subset of 14 joints for calculating the metrics.

2.2.2. ICVL Dataset

ICVL dataset [tang2014latent] contains 22K original depth frames including 10 subjects and two test sets with 800 frames each. However, by applying rotations, the total size of dataset exceeds 300K images along-with the ground truth. Intel creative gesture camera was used to acquire the depth images. The depth images have a high quality with hardly any missing depth values and sharp outlines with little noise. The dataset has good number of complex

hand poses but, not as complex as NYU dataset [barsoum2016articulated]. Ground truth is created using a search method, guided by a binary Latent Tree Model (LTM) [choi2011learning]. However, ground truth is not accurate and the variation in hand shapes and sizes is low. Moreover, their hand skeleton does not match the actual joint locations of a real hand skeleton [yuan2017bighand2].

2.2.3. MSRA-2015 Dataset

MSRA-2015 dataset [sun2015cascaded] contains 76,500 depth frames captured from Creative gesture camera. Images are captured from 9 different subjects, each performing 17 hand gestures. Ground truth is annotated using a semi-automatic and iterative process followed by manual corrections [qian2014realtime]. Training is usually performed on 8 subjects and the sequences from 1st subject are used for testing.

2.2.4. BigHand2.2M Dataset

The BigHand2.2M dataset [yuan2017bighand2] is the largest hand pose dataset so far. It contains over 580K training frames and over 295K test frames for the pose estimation task. It was captured using an Intel RealSense SR300, a structured-light sensor. The dataset was automatically annotated by using six 6D magnetic sensors and inverse kinematics that results in 21 3D joint locations. However, the labels are rather inaccurate due to the employed annotation process. The depth images have a good quality, and sharp outlines with little noise. The dataset is considered very challenging, as it has a large pose variability, and contains 10 different users, whereas five of them are only available for testing. The annotations of this dataset match the joint locations of a real hand skeleton.

There are some other existing real depth-based hand pose datasets from frontal camera view i.e. Dexter [sridhar2013interactive], SHREC-2017[1], MSRA-2014 [qian2014realtime], ASTAR [xu2016estimate]. However, these datasets either contain small number of original images, missing depth information, a few ground truth joint positions or many outliers in the annotations. Therefore, they are not considered in our works.

2.3. Hand Pose Estimation from RGB Images

3D Hand pose estimation from a single RGB image is particularly a difficult problem, specifically due to the inherent ambiguities of monocular color images. In the past few years, hand pose estimation from RGB images has been actively studied [simon2017hand; mueller2018ganerated; panteleris2017using]. However, due to the lack of RGB datasets, current state-of-the-art methods create synthetic datasets ([zimmermann2017learning]), or use GANs to

[1] http://www-rech.telecom-lille.fr/shrec2017-hand/

generate training data [shrivastava2017learning; mueller2018ganerated]. [simon2017hand] propose a multiview bootstrapping approach to estimate 2D hand pose by iteratively fine-tuning the estimation results. They propose an automatic method to generate an annotated RGB dataset using a panoptic studio camera setup. Mueller et al. [mueller2017ganerated] propose to use RGB-D input to estimate 3D hand pose from an egocentric viewpoint, where the hand is interacting with objects. Zimmermann et al. [zimmermann2017learning] estimate 3D hand pose from single RGB image in three steps: hand segmentation, 2D hand pose estimation and 3D hand pose estimation. They also created a perfectly annotated synthetic dataset however, it lacks realism. Paschalis et al. [panteleris2017using] estimate an absolute 3D hand pose using a two step pipeline. The first step is to estimate the 2D hand pose using a CNN. The second is to estimate 3D hand pose by optimizing the hand model in 3D space with inverse kinematics. Mueller et al. [mueller2017ganerated] perform 3D hand tracking from rgb images by combining a CNN with a kinematic 3D hand model. To train their model they use an image-to-image translation network to create large amount of semi-real (GANerated) images that have perfect annotation and also have the same statistical distributions as real images. However, their method requires a predefined hand model for each subject and the hand model is obtained by a per-user skeleton adaptation process.

2.4. Simultaneous Hand Shape and Pose Estimation

Deep learning based simultaneous estimation of 3D hand shape and pose is a novel and challenging problem which has many applications. We propose the first novel framework to estimate 3D shape and pose from a single depth image. Thereafter, a few more works from us, Ge et al. [ge20193d], Adane et al. [Adnane20193d] are proposed. Ge et al. [ge20193d] regress 3D hand mesh and pose using a weakly-supervised approach from a monocular RGB input. They estimate 2D heatmaps of hand joints as intermediate representations which are combined with RGB image features to estimate hand shape via a Graph CNN. Thereafter, they regress the 3D pose from the estimated hand shape. However, their approach greatly relies on the pseudo-ground truth of real data, which is obtained using a pre-trained model with labeled synthetic RGB dataset. Moreover, their 3D pose estimation accuracy directly depends on the quality of real hand shape estimation. We will address these problems in this thesis and propose several solutions for an accurate 3D hand shape and pose estimation from a single depth map.

2.5. Signature Verification

Signature-based verification methods can be classified into three categories, namely, offline, online and in-air signature verification. In offline methods, the signature is taken on paper and verification is performed using it's camera-captured or scanned image in 2D [robert2015offline; chandra2016offline; bhat-

tacharya2013offline]. For 2D signature verification, these methods have used support vector machine (SVM), pixel matching (PM) and artificial neural network (ANN). Online methods use a tablet or a pad to acquire a digital signature using e-pen or fingertip movement on a digital display screen. This way of capturing the signature allows to record dynamic features of the signature trajectory such as pen pressure and acceleration. Verification in online methods has been performed using hidden markov model (HMM) [van2007using] , SVM [gruber2010online] and dynamic time warping (DTW) [martens1996line; feng2003online]. Among these verification techniques, DTW has been the most effective and widely-used algorithm due to its ability to well align the temporal signals [bailador2011analysis]. In-air signatures have been acquired using the camera of a Google Glass (i.e., egocentric view) or by placing camera in front of the subject (i.e., 3rd person view), or using a movement sensor in a cell phone [fang2017novel; jeon2012system; sajid2015vsig; bailador2011analysis]. For in-air signature verification, [jeon2012system; sajid2015vsig; malik20183dairsig] employed DTW algorithm whereas, Fang et al. [fang2017novel] developed a fusion algorithm based on fast fourier transform (FFT) and DTW.

2.5.1. In-Air Signature Verification

Comprehensive reviews on off-line and on-line signature verification have been reported in [kumar2016survey; yadav2013survey; dalal2016performance]. Keeping in view the relevance with our work, here we discuss the published literature on in-air signature verification. Katagiri et al. [katagiri2002personal] propose the first free space personal authentication system. They adopt a high speed video camera to acquire in-air signature trajectory. For verification, they employ a commercial signature verification engine provided by CyberSIGN JAPAN Inc [2]. In [takeuchi2013multimodal], Takeuchi et al. combine hand shape features with RGB camera to capture handwriting motion in the air. Keeping in view the extended use of smartphones in various applications, Diep et al. [diep2015sigver3d] use motion sensor in a smartphone to record signature data. They use SVM for verification. Matsuo et al. [matsuo2007arm] introduce an adaptive template update method in order to improve long term stability in arm swing motion. Jeon et al. [jeon2012system] adopt a low cost depth camera to capture in-air signature trajectory. In order to record the signature trajectory, they introduced a heuristic approach to detect the palm center position. Bailador et al. [bailador2011analysis] investigate various pattern recognition techniques, i.e., HMM, Bayes classifier and DTW, for authentication. The best performance was shown by the DTW algorithm. In order to capture in-air signature trajectory, the authors used an embedded 3D accelerometer in a mobile phone. With the recent trend towards wearable technology, Sajid et al. [sajid2015vsig] propose a new in-air signature acquisition method using the Google Glass. They use a motion based video segmentation algorithm along-with a skin color based hand segmentation in order to acquire signature data. A video based in-air signature

[2] http://www.cybersign.com

verification system using a high speed RGB camera has been introduced by Fang et al. [fang2017novel]. They trace fingertip using an improved tracking learning detection (TLD) algorithm. For verification phase, the authors develop a fusion algorithm based on an improved DTW and the Fast Fourier Transform (FFT). Recently, Khoh et al. [khoh2018air] propose a predictive palm segmentation algorithm to create a motion history image (MHI) using a depth sensor. Afterwards, they produce a two-dimensional representation of hand gesture signature based on the MHI. All of the methods mentioned above, treat and process in-air signature trajectory in the conventional on-line form. However, we emphasize that in-air signature encloses a unique hidden depth feature which should not be ignored in the acquisition and the verification. In this thesis, we also investigate the potential in this important feature. On the other hand, the reported methods for fingertip tracking are based on heuristics which are not feasible for practical applications. Inspired by the recent progress in deep learning based hand pose estimation using a depth sensor [yuan2018depth], we propose a new real-time algorithm for in-air acquisition which regresses the 3D hand pose rather than detecting only fingertip or palm center. Therefore, the proposed method is not restricted to any specific hand pose and has the ability to perform well in cases of occlusion.

Chapter 3

CNN-Based Structured Hand Pose Estimation

Contents

3.1. Introduction

Current hybrid approaches based on deep learning followed by model fitting preserve the structure of hand during the training process. However, an individualized hand model hinders the generalization of these methods over varying hand shapes and sizes. Particularly, Zhou et al. [zhou2016model] propose an efficient model based deep learning approach as an alternative to generative post-processing step in hybrid methods. However, a big limitation of this work is an assumption of a fixed bone-lengths hand model geometry during end-to-end training. Our idea is to estimate not only the 3D hand pose but

also the bone-lengths of hand skeleton at the same time. Thus, we introduce a novel hybrid algorithm which simultaneously estimates the 3D hand pose and bone-lengths of hand skeleton (Section 3.2). To this end, hand scale parameters are learned to facilitate the end-to-end training process of model based deep learning approach thereby, leading to promising results for 3D hand pose estimation. In order to show the validity of our approach, a hand pose dataset with large variation in hand shapes and sizes is necessary. Several real hand pose datasets are publicly available, but individually, these datasets lack in varying hand shapes and sizes of subjects, number of original depth images and complexity of hand poses [barsoum2016articulated]. Therefore, we combine most commonly used real hand pose datasets and convert them into a single unified format, we call *HandSet*.

On the other hand, deep learning-based direct regression methods achieved highest accuracies on public benchmarks (see Chapter 2). Despite of the fact that these methods achieve higher accuracy, they do not well exploit the structural information of hands during the learning process [zhou2016model; yuan2018depth; malik2017simultaneous]. Specifically, independent learning of sparse joint positions with no consideration to joint connection structure and hand skeleton constraints leads to coarse predictions. This is the main reason these methods still generalize poorly on unseen hand shapes [yuan2018depth] and consequently, not directly usable in practical VR applications. Therefore, in Section 3.3, we introduce a novel structure-aware CNN-based discriminative approach which incorporates the structural constraints of hand skeleton and enhances the loss function for better learning of 3D hand pose. Our main idea is to jointly learn the 3D joint keypoints and the hand structure parameters. Thereby, facilitating the CNN to maintain a structural relation between the estimated joint keypoints. Our method is simple, efficient and effective. It optimizes a combined loss function of 3D joint positions and simple structural constraints of the hand skeleton. The constraints comprise of fingers lengths, fingers inter-distances and distances of joints in the kinematic chain of the hand skeleton (kinematic distances). These constraints are easy to learn and guide the optimization process to estimate more refined and accurate 3D hand pose. Another contribution which helps to improve the accuracy is to convert the sparse joints keypoints to dense representation. To this end, we perform n-points interpolation between the pairs of parent and child ground truth joint positions along the kinematic chain of hand skeleton. These simple strategies can be easily used to improve the accuracy of any CNN-based discriminative method without additional cost.

In addition, existing hand pose estimation methods assume already segmented hand region from a raw depth image as input to their algorithms. The hand segmentation approaches are mainly based on heuristics or ground truth annotation which make them difficult to use in practical applications. The problem of hand segmentation is not well addressed in the existing works. Hence, our second contribution is a new CNN-based hand segmentation method to extract the hand region from a raw depth frame. For training over images with varying backgrounds and camera noise, we combine several exist-

ing hand pose datasets including a new dataset which we capture to include more variation in hand shapes. The combined dataset will be public.

By performing exhaustive evaluation of our algorithm, we show the effectiveness of our hand segmentation algorithm, n-points interpolation strategy and learning the structural constraints jointly with the 3D hand pose. Experiments show that our method performs better than several state-of-the-art hand pose estimation on the NYU public benchmark.

3.2. Model-based Approach with Bonelengths Adaption

In this section, we explain our proposed model-based approach for simultaneous hand pose and skeleton bone-lengths estimation from a single depth image.

3.2.1. Combined Dataset and Pre-Processing

First step to merge different datasets is to select the number of common joint positions present in all datasets. ICVL dataset has least number of joints. We consider corresponding 16 joints in the NYU and MSRA-2015 datasets and remove additional joints for consistency. Since, each dataset uses different depth camera to acquire images, we need to pre-process the depth frames according to their respective camera intrinsics, frame resolutions and depth range. Inspired by the method in [zhou2016model], for depth invariance, the images are cropped around palm center in all three dimensions (u, v and depth) using a fixed size bounding box. Then, depth values are normalized to [−1, 1]. The 3D joint locations are also normalized in range [−1, 1] using the bounding box. The final pre-processed image is of 128 x 128 dimension and has 16 ground truth annotations which include 12 internal joints as shown in Figure 3.1 and four finger-tips. The *HandSet* contains $450K$ pre-processed training depth images, $18K$ test images and 20 different subjects.

3.2.2. Hand Pose and Bone-Lengths Estimation

In this subsection, we explain our approach for simultaneous estimation of hand pose and bone-lengths of the hand skeleton using a hybrid forward kinematics layer and deep architectures.

Hybrid Forward Kinematics Layer: Figure 3.1 shows our hand skeleton. We assume a zero pose vector (i.e. pose with all parameters set to zero) as the reference hand pose. All other poses are defined relative to this reference pose. We initialize the hand skeleton by the averages of individual bone-lengths from ground truth annotations of each dataset. Given the hand pose and scale parameters, the hybrid forward kinematic layer (see Figure 3.3) implements a forward kinematic function F_k defined as:

$$F_k(\Theta, S) = J \tag{3.1}$$

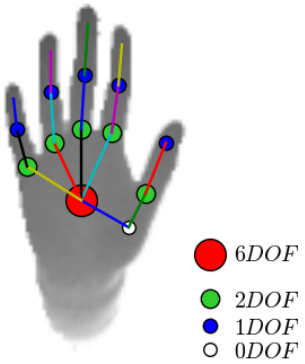


Figure 3.1.: *Illustration of our hand skeleton with 21DOF.*

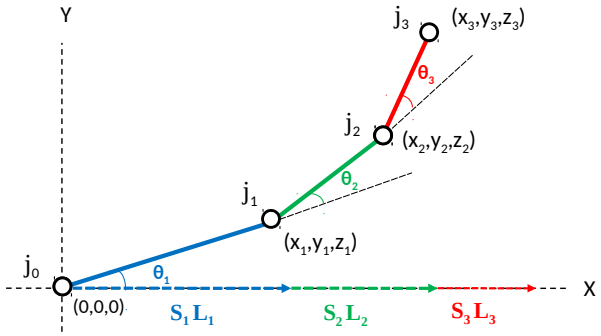


Figure 3.2.: *Illustration of 3D joint transformations of four adjacent joints of initial hand skeleton using forward kinematics process. Assuming 1DoF for each joint and considering three rotations at j_0, j_1, j_2 among z axis, the relative position of joint j_3 with respect to reference joint position j_0 can be calulated as $(x_3, y_3, z_3, 1)^T = [Trans_x(\boldsymbol{S}_1 L_1)] \times [Rot_z(\theta_1)] \times [Trans_x(\boldsymbol{S}_2 L_2)] \times [Rot_z(\theta_2)] \times [Trans_x(\boldsymbol{S}_3 L_3)] \times [Rot_z(\theta_3)] \times [0, 0, 0, 1]^T$.*

Where $\Theta = \{\theta_p\}$, $p = \{1, 2, \cdots, 21\}$ is a vector of pose parameters, $S = \{s_l\}$, $l = \{1, 2, \cdots, 15\}$ defines the hand scale factors associated with bone-lengths and $J = \{j_n\}$, $n = \{1, 2, \cdots, 16\}$ is a vector of the predicted joint positions.

The 3D transformation of each of the 16 joints in J is derived from its joint angles for rotation and scaled bone-lengths for translation. The global 3D position (x_n, y_n, z_n) of a joint is obtained by applying series of transformations (rotational and translational) along the path starting from hand root joint to this joint as shown in Figure 3.2.

Cost function is obtained by using Euclidean 3D joint location loss given as:

$$\frac{1}{2} \left\| F_k(\Theta, S) - J_{GT} \right\|^2 \tag{3.2}$$

Where J_{GT} is a vector of 3D ground truth joint positions.

Since, Equation 3.1 is differentiable with respect to both pose parameters Θ and hand scales S, hence, it can be used in deep network to compute gradients

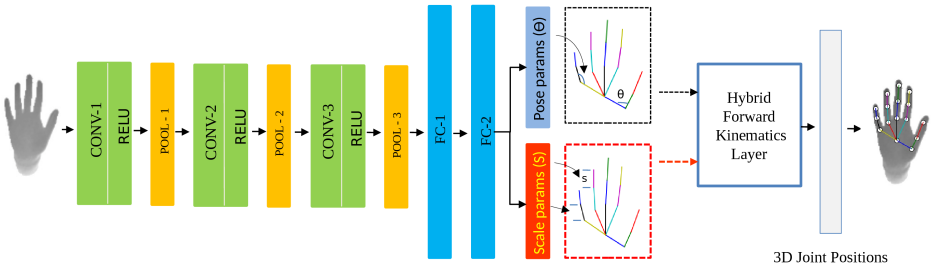


Figure 3.3.: *Illustration of our model for simultaneous hand pose and skeleton estimation. The algorithm starts from three convolutional layers and two fully connected layers. The last fully connected layer outputs hand pose parameters (Θ) and scale parameters (S) associated with the bone-lengths of the skeleton. In the end, a hybrid forward kinematic function is applied that outputs 3D joint positions using the hand scale and pose parameters.*

for back-propagation. The Jacobian of F_k with respect to Θ is defined as:

$$\frac{\partial F_k}{\partial \Theta} = \begin{bmatrix} \frac{\partial j_1}{\partial \theta_1} & \frac{\partial j_1}{\partial \theta_2} & \cdots & \frac{\partial j_1}{\partial \theta_{21}} \\ \frac{\partial j_2}{\partial \theta_1} & \frac{\partial j_2}{\partial \theta_2} & \cdots & \frac{\partial j_2}{\partial \theta_{21}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial j_{16}}{\partial \theta_1} & \frac{\partial j_{16}}{\partial \theta_2} & \cdots & \frac{\partial j_{16}}{\partial \theta_{21}} \end{bmatrix} \tag{3.3}$$

The Jacobian of F_k with respect to S can be defined in a similar way. Partial derivative of a joint j_n in J with respect to a pose parameter θ_p can be calculated as:

$$\frac{\partial j_n}{\partial \theta_p} = \Big(\prod_{c \in P_c} [\mathrm{Rot}_\phi(\theta)] \times [\mathrm{Trans}_{\phi_c}(S_c L_c)] \Big) [0,0,0,1]^T \tag{3.4}$$

where,

$$\mathrm{Rot}_\phi(\theta) = \begin{cases} \mathrm{Rot}_{\phi_c}(\theta_c) & \text{if } c \neq p \\ \mathrm{Rot}'_{\phi_c}(\theta_c) & \text{if } c = p \end{cases}$$

P_c is the set of joints along kinematic chain from j_n to the root joint and ϕ is the rotation axis.

Similarly, we compute partial derivative of a joint j_n in J with respect to a scale parameter s_l as:

$$\frac{\partial j_n}{\partial s_l} = \sum_{k \in P_k} \Big[\Big(\prod_{c \in P_c} [\mathrm{Rot}_{\phi_c}(\theta_c)] \times [\mathrm{Trans}_\phi(SL)] \Big) [0,0,0,1]^T \Big] \tag{3.5}$$

where,

$$\mathrm{Trans}_\phi(SL) = \begin{cases} \mathrm{Trans}_{\phi_c}(S_c L_c) & \text{if } c \neq k \\ \mathrm{Trans}'_{\phi_c}(S_c L_c) & \text{if } c = k \end{cases}$$

and, P_k is the set of parent joints of j_n that share the same scale parameter s_l.

Deep Architectures with Hand Scales: Human hands differ in individual fingers and palm sizes. There is a need to explicitly consider such differences during training. Therefore, we introduce various scales of hand as additional learning parameters to facilitate CNN training on *H*andSet as shown in Figure 3.3. These scales factors are learned by the CNN along-with the pose parameters.

We propose three implementations of our method explained in the following subsections and compare their performances in Section 3.2.4. We build our CNN architecture based on the baseline architecture proposed in [oberweger2015hands], mainly for the sake of fair comparison. The pipeline of our algorithm is shown in Figure 3.3. The architecture of CNN comprises of 3 convolutional layers using 5, 5, 3 kernel sizes respectively. Max pooling layers are then connected using strides 4,2,1 with zero padding. The feature maps from convolutional layers are of size 12 x 12 x 8. Two fully connected layers consist of 1024 neurons each. Dropout layers are added with dropout ratio of 0.3. All convolutional layers use ReLu as activation.

GlobalScale: In this architecture, we define a global scale for the hand skeleton such that it can symmetrically vary its size. In Figure 3.3, the last fully connected layer outputs pose parameters and additional global hand scale parameter s, shared by all 15 bones of the hand skeleton. Larger scale value results in bigger hand skeleton and vice versa. The hybrid forward kinematic layer takes this scale parameter as input along-with pose parameters and computes 3D joint positions according to Equation 3.1. The partial derivative of a joint with respect to the global scale parameter can be computed using Equation 3.5.

5Scales: This architecture associates five separate hand scale parameters from tips of the five fingers to the palm center (root joint). These parameters allow the individual fingers to vary their lengths according to their respective scale values, thereby adding a flexibility to both shape and size of the hand skeleton. These parameters are defined by S as:

$$S = \left\{ s_{f1}, s_{f2}, s_{f3}, s_{f4}, s_{f5} \right\} \tag{3.6}$$

Given the pose parameters Θ and S, forward kinematic function defined by Equation 3.1 is applied to estimate more accurate 3D joint locations. Using Equation 3.5, the partial derivative of a joint with respect to its associated finger scale parameter is calculated.

MultiScale: In this architecture, we assign a separate scale to each bone of our hand skeleton. Each bone-length can be estimated independently of other bones. Hence, this architecture provides the maximum flexibility to adapt shape and size of the hand skeleton.

3.2.3. Network Implementation Details

For end-to-end training of our model, we use Caffe open source framework for deep networks [jia2014caffe]. The network is trained until convergence with a fixed learning rate of 0.001 using 0.9 as SGD momentum. We perform data

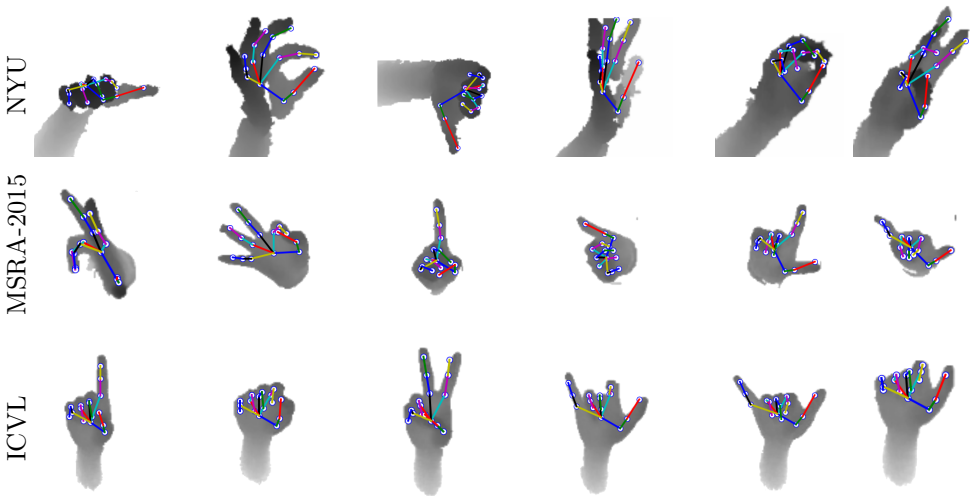


Figure 3.4.: *Sample results from our 5Scales architecture. The predicted 3D joint positions are displayed on the depth images. The rows show images from NYU, MSRA-2015 and ICVL datasets, respectively from top to bottom.*

augmentations i.e. rotations and scalings during training phase. The complete framework runs on a PC with Nvidia GeForce 1070 GPU. One forward pass takes $7ms$.

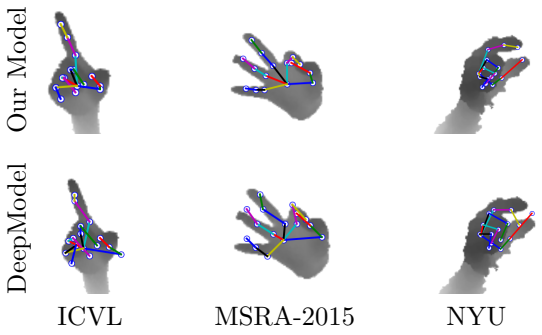


Figure 3.5.: *Sample images with overlaid predicted 3D joint positions from our model with hand scale parameters (top row) and Zhou et al. [zhou2016model] without hand scale parameters (bottom row), when trained on HandSet dataset.*

3.2.4. Experimental Evaluation

In this subsection, we illustrate the accuracy of our model through both qualitative and quantitative results and comparisons with the state-of-the-art hybrid methods. We do not claim to exceed the accuracy of recently published discriminative methods [ge2017robust; guo2017region] which neglect hand model geometry i.e. kinematics and physical constraints. Instead, we provide a performance comparison with the existing hybrid methods to validate our algorithm that fully exploits a flexible hand model geometry and

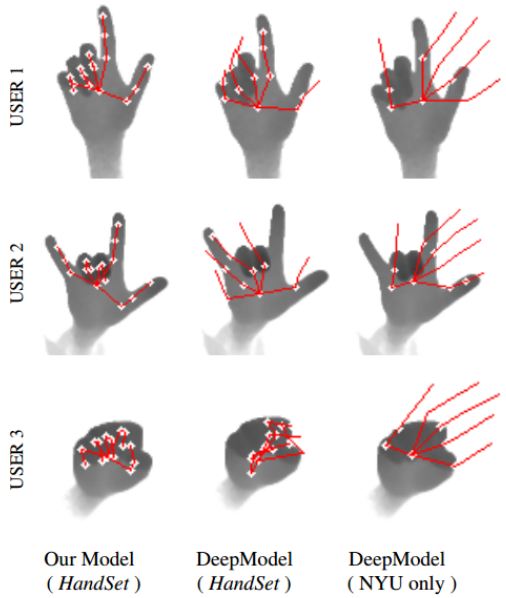


Figure 3.6.: *Hand pose inference results on unseen images from our model and Zhou et al. [zhou2016model]. Our model shows good results while the compared model fails to converge.*

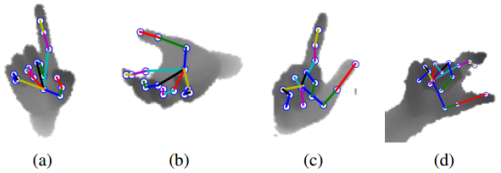


Figure 3.7.: *Some failure cases are shown from our GlobalScale architecture (a and b) and MultiScale architecture (c and d). In (a) and (b), smaller global scale leads to incorrect poses. In (c) and (d), we see inconsistency in bone-lengths due to independent scales estimation.*

estimates the 3D hand pose and bone-lengths of the hand skeleton simultaneously. Notably, famous public datasets such as NYU and ICVL contain low variation in hand shapes and sizes (see Section 2). However, we demonstrate our results on these datasets for completeness. We use two common evaluation metrics. First is the average 3D joint location error on test dataset. Second, fraction of test frames for which maximum predicted 3D joint error is below a certain threshold in millimeter.

Qualitative Evaluation: Some challenging hand pose images from three datasets along-with predicted joint positions from our model are shown in Figure 3.4. We show some sample images with overlaid hand skeleton from our 5Scales model and deep model [zhou2016model] in Figure 3.5. Our model shows very good results whereas, the compared model is unable to converge successfully leading to inaccurate 3D hand joint positions and bone-lengths. We tested the 5Scales model with Zhou et al. [zhou2016model] on unseen images acquired from three different users. Our model is able to infer hand pose quite accurately whereas, the other model fails to converge (see Figure

Methods	3D Joint Location Error
Zhou et al. [zhou2016model]	18.7mm
MultiScale [Ours]	15.1mm
GlobalScale [Ours]	15.3mm
5Scales [Ours]	**12.7mm**

Table 3.1.: *Quantitative comparison of our three architectures and Zhou et al. [zhou2016model] on HandSet test dataset.*

Methods	3D Joint Location Error
Oberweger et al. [oberweger2015training]	**16.0mm**
Zhou et al. [zhou2016model]	17.0mm
Ours	16.2mm

Table 3.2.: *Quantitative comparison on NYU test set.*

3.6). Some failure cases from our two other architectures (GlobalScale and MultiScale) are shown in Figure 3.7. Incorrect bone-lengths estimation from GlobalScale architecture can happen due to a single scale parameter associated with all bones of the hand skeleton. On the other hand, in MultiScale architecture, independent learning of each bone-length of the hand skeleton may result in incorrect bone-lengths estimation.

Quantitative Evaluation: We trained our three architectures (GlobalScale, MultiScale and 5Scales) as well as publicly available model based deep architecture [zhou2016model] on *H*andSet. Notably, [zhou2016model] fails when trained on *H*andSet. This is mainly due to the fact that they assume a fixed hand model geometry during end-to-end training. We summarize the comparison of accuracies in Figure 3.8 and Table 3.1. Our 5scales architecture shows the best accuracy and proves that our approach works well with large variation in hand shapes and sizes. On NYU dataset, our accuracy is comparable to Oberweger et al. [oberweger2015training] on common joints (see Table 3.2). On ICVL dataset, our method shows improved performance in comparison to other state-of-the-art hybrid methods (see Table 3.3). Since, NYU dataset has no variation (one subject) and ICVL has low variation in hand shapes and sizes, therefore one can see a clear advantage of our method on ICVL dataset while a comparable performance on NYU dataset. Figure 3.9 shows a more detailed comparison on individual joints in ICVL dataset.

Methods	3D Joint Location Error
LRF [tang2014latent]	12.6mm
Zhou et al. [zhou2016model]	11.5mm
Ours	**10.0mm**

Table 3.3.: *Quantitative comparison on ICVL test set.*

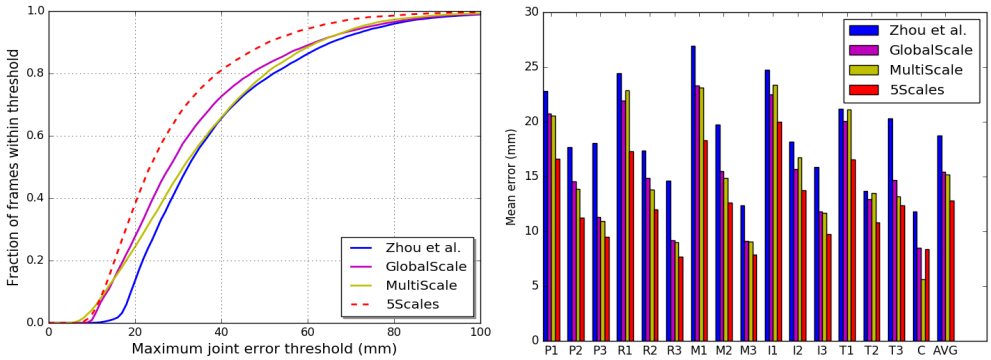


Figure 3.8.: *Qualitative comparison of our proposed architectures (GlobalScale, MultiScale and 5Scales) vs. Zhou et al. [zhou2016model] on HandSet test dataset. The upper shows the fraction of frames in error within thresholds and the lower shows the mean error on individual joints.*

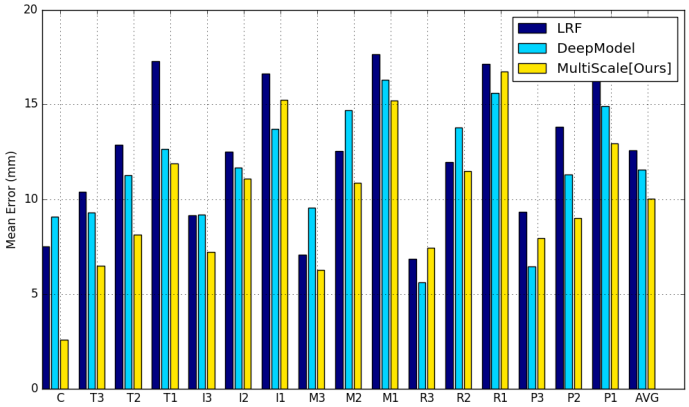


Figure 3.9.: *comparison with respect to mean error on individual joints with the state-of-the-art hybrid methods (LRF [tang2014latent], DeepModel [zhou2016model]) on ICVL dataset.*

3.3. Structure-Aware Direct Regression Approach

In this section, we explain our approach for direct hand pose regression which incorporates new hand structural constraints, and a CNN-based hand segmentation method from a raw depth frame.

3.3.1. Method Overview

The goal of our pipeline is to estimate more stable and accurate 3D joint positions J, given a raw depth input D_o. To this end, we simultaneously optimize for J, fingers lengths FL, fingers inter-distance FD and kinematics distances KD to facilitate the learning of 3D joint positions in a structured manner. Our pipeline is shown in Figure 5.2. D_o is resized and then colorized (using the JET colormap) by a function g. The output RGB image D_i is of size 227 x 227 x 3. D_i is passed as input to the PalmCNN to directly regress hand palm center (u,v) in image coordinates. Then, a cropping function f is

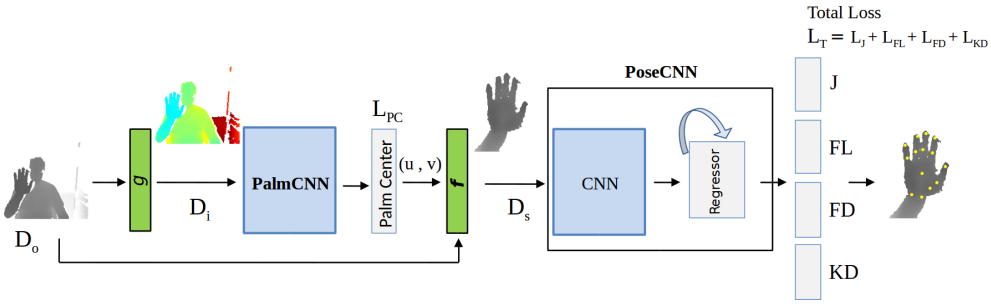


Figure 3.10.: *Our pipeline for hand segmentation and pose estimation. The raw depth frame D_o is given as input to a function g which resizes D_o to 227 x 227 x 3 dimension and colorizes it using the JET colormap. The output of g (D_i) is fed to the PalmCNN to regress 2D hand palm center (u,v). L_{PC} is the loss for the PalmCNN. The function f crops the hand region D_s given (u,v). D_s is fed to PoseCNN which outputs 3D joint positions J, fingers lengths FL, fingers inter-distances FD, and kinematic distances KD.*

applied to segment the 3D hand region D_s from the raw depth frame D_o. The colorization step is simple and helps to improve the accuracy; see Section 3.3.6. Finally, the PoseCNN takes D_s as input and estimates 3D joint positions J, fingers lengths FL, fingers inter-distance FD and kinematics distances KD. The PoseCNN comprises of a CNN and a regressor; see Section 3.3.3 for details. The PalmCNN and the PoseCNN are trained separately.

3.3.2. CNN-based Hand Segmentation

The function g simply resizes and colorizes D_o to be fed as input to the PalmCNN. The output D_i of g is an RGB image of size 227 x 227 x 3. The task of the PalmCNN is to estimate the pixel coordinates of the center of the hand region i.e. palm center (u,v). The CNN architecture of the PalmCNN is similar to the AlexNet [krizhevsky2012imagenet] except that the final fully connected layer regresses the palm center. The softmax loss layer is replaced by euclidean loss layer. The euclidean 2D palm center loss is given as:

$$L_{PC} = \frac{1}{2} \| PC - PC_{GT} \|^2 \tag{3.7}$$

Where L_{PC} is the palm center loss and PC_{GT} is the ground truth palm center. To train the PalmCNN, we combine four of the publicly available hand pose datasets (i.e NYU [tompson2014real], ICVL [tang2014latent], MSRA-2015 [sun2015cascaded] and Dexter-1 [sridhar2013interactive]) with a new dataset which we captured using creative senz3D camera [camera˙Senz3D]. This additional small scale dataset is captured because the public datasets lack in hand shape variation [malik2017simultaneous]. To obtain the ground truth palm center, we employ the generative method proposed by [tagliasacchi2015robust]. We captured depth images from five different subjects. Our dataset contains 8000 original depth images. Notably, the variation in hand position should cover the whole image space. Therefore, we create around 10

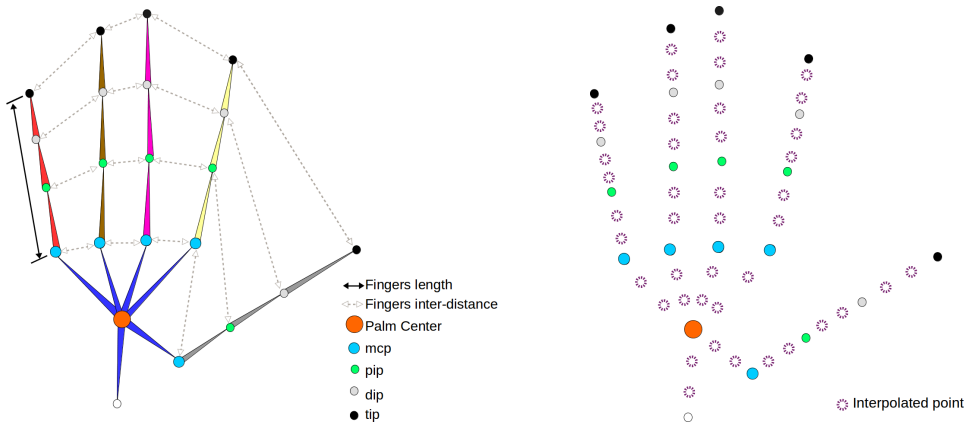


Figure 3.11.: *The left figure shows the graphical representation of two of the structural constraints i.e. Fingers lengths and Fingers inter-distances. The hand skeleton on the right shows the interpolated points ($n = 2$) between the sparse ground truth joint positions.*

augmented copies of every depth frame in the combined dataset by translating it around the whole image using the ground truth hand palm center position. The total number of training and testing frames are $4.55M$ and $200K$ respectively. We fine-tune the AlexNet (pre-trained on ImageNet dataset) with the combined dataset. The crop function f takes the estimated (u,v) and D_o as inputs and segments the 3D hand region; see Section 3.3.6 for details about f. The resultant image D_s is of size 224 x 224.

3.3.3. Hand Structural Constraints

In our pipeline, the PoseCNN aims to jointly estimate the hand joint keypoints J and additional constraints (i.e. fingers lengths FL, fingers inter-distance FD, kinematic distances FD). During training, these constraints help to maintain a structural relation between the joints positions. The ground truth for the constraints can easily be obtained from the ground truth joint positions. The euclidean 3D joint positions loss L_J is given as:

$$L_J = \frac{1}{2} \| J - J_{GT} \|^2 \tag{3.8}$$

Where $J_{GT} \in \mathbb{R}^{Px3}$ is a vector of 3D ground truth joint positions. P is the number of joint keypoints. The constraints are explained as follows:
Fingers lengths: We first calculate J-1 hand bone-lengths from the ground truth joint positions using the standard 3D euclidean distance formula. To obtain a finger's length fl, we add the bone-lengths from the base joint (mcp) to the finger-tip joint (tip) as shown in Figure 3.11. The equation for fl can be written as:

$$fl = bl_{mcp-pip} + bl_{pip-dip} + bl_{dip-tip} \tag{3.9}$$

Where bl_{x-y} is the bone-length from a parent joint x to a child joint y. Therefore, a set FL_{GT} is represented as:

$$FL_{GT} = \{fl_{pinky}, fl_{ring}, fl_{middle}, fl_{index}, fl_{thumb}\} \tag{3.10}$$

The euclidean fingers lengths loss L_{FL} is:

$$L_{FL} = \frac{1}{2} \| FL - FL_{GT} \|^2 \tag{3.11}$$

Where FL is the vector of estimated fingers lengths.
Fingers inter-distances: The distances between the *mcp* joints of consecutive fingers for a particular hand mostly remain fixed. However, the distances between *pip*, *dip* and *tip* joints between fingers can vary depending on the pose of the hand. The inter-distances between neighboring fingers can easily be obtained by calculating 3D euclidean distances between respective joints of the fingers; see Figure 3.11. For example, the inter-distances between *index* and *middle* fingers are evaluated as:

$$\begin{aligned} fd(index, middle) = \{ & d(mcp_{index}, mcp_{middle}), d(pip_{index}, pip_{middle}), \\ & d(dip_{index}, dip_{middle}), d(tip_{index}, tip_{middle}) \} \end{aligned} \tag{3.12}$$

Where $fd(.)$ is a set of inter-distances between the joints of two adjacent fingers and $d(.)$ represents 3D euclidean distance between two joints. Likewise, inter-distances for remaining finger pairs i.e. $(middle, ring)$, $(ring, pinky)$ and $(thumb, index)$ can be obtained using Equation 3.12. Hence, a set FD_{GT} can be expressed as:

$$\begin{aligned} FD_{GT} = \{ & fd(index, middle), fd(middle, ring), \\ & fd(ring, pinky), fd(thumb, index) \} \end{aligned} \tag{3.13}$$

The fingers inter-distances loss L_{FD} can be written as:

$$L_{FD} = \frac{1}{2} \| FD - FD_{GT} \|^2 \tag{3.14}$$

Where FD is the vector of estimated fingers inter-distances.
Kinematic distances: Hand skeleton bears an inherent kinematic structure which should not be ignored in the pose estimation task. In this work, we add a much needed loss function which incorporates kinematic distances of all the joints in the hand skeleton. Given the set of parents joints S_{p_j} of a joint p_j in J_{GT}, the kinematic distance kd_j from the root joint to p_j can be calculated as:

$$kd_j = \sum_{i=0}^{M-1} d(J_{GT_i}, J_{GT_{i+1}}) \tag{3.15}$$

Where $i \in S_{p_j}$ and M is the size of the set S_{p_j}. Using Equation 3.15, the kinematic distances of each joint in J_{GT} can be obtained. Hence, the loss L_{KD} can be written as:

$$L_{KD} = \frac{1}{2} \| KD - KD_{GT} \|^2 \tag{3.16}$$

Where KD and KD_{GT} are the vectors of estimated and ground truth kinematic distances.

Method Implementations	3D Joint Location Error J
PoseCNN(J)	15.2mm
PoseCNN($J \cup FL$)	14.7mm
PoseCNN($J \cup FD$)	13.6mm
PoseCNN($J \cup KD$)	13.9mm
PoseCNN($J \cup FL \cup FD \cup KD$)	**12.9mm**

Table 3.4.: *We evaluate five different implementations of our PoseCNN on the NYU hand pose dataset. The PoseCNN(J) is the baseline which is trained for estimating joint positions only. The PoseCNN($J \cup FL \cup FD \cup KD$) performs the best and shows an error improvement of 15.13% on the estimated J over the baseline.*

Total loss: Including the additional constraints (mentioned above) help to improve the accuracy of hand pose estimation task and maintain the structure of the hand skeleton; see Section 3.3.6. The final loss equation for the PoseCNN can be written as:

$$L_T = L_J + L_{FL} + L_{FD} + L_{KD}. \tag{3.17}$$

3.3.4. Hand Skeleton Interpolation

In order to get a dense representation of hand skeleton, we linearly interpolate n joints between each pair of parent and child joints in the kinematic hierarchy of the hand skeleton; see Figure 3.11. We try different number of interpolated points n and study their effects on the accuracy of the estimated pose; see Section 3.3.6. As an example, the formulas for interpolating two 3D points $P1$ and $P2$ between two 3D points P_a and P_b are:

$$P1 = 0.7 * P_a + 0.3 * P_b \quad , \quad P2 = 0.3 * P_a + 0.7 * P_b \tag{3.18}$$

3.3.5. CNN Architecture and Iterative Regression

The architecture of CNN in the PoseCNN is similar to ResNet-50 [he2016deep] except that final fully connected (FC) layer which outputs the features $\varphi \in \mathbb{R}^{1024}$. The features φ are concatenated with an initial estimate of $E = \{J, FD, FL \text{ and } KD\}$ i.e. $\phi = \{\varphi, E\}$. Initial estimate of E is obtained using the mean values of $\{J, FD, FL \text{ and } KD\}$ from the NYU ground truth annotations. This estimate is kept fixed during the training and the testing. ϕ is fed to a regressor which comprises of two FC layers with 1024 neurons each. Both the FC layers use dropout layers with ratio of 0.3. The last FC layer contains M neurons. Where $M = 2P(n+1) + 10n + 21$. The regressor aims to refine E in an iterative feedback manner i.e. $E_{t+1} = E_t + \delta E_t$. In our implementation, we use at least three iterations. Directly regressing E is challenging therefore, we observe that inclusion of the regressor is beneficial.

n-points Interpolation	3D Joint Location Error
	J
PoseCNN(1-point Interp.)	12.80mm
PoseCNN(2-point Interp.)	12.63mm
PoseCNN(3-point Interp.)	12.38mm
PoseCNN(4-point Interp.)	12.17mm
PoseCNN(5-point Interp.)	**11.9mm**

Table 3.5.: *We observe the effects of n-points interpolation between the pairs of parent and child joints in the kinematic hierarchy of the hand skeleton. The value of n varies from 1 to 5. 5-point interpolation shows 5.5% improvement in accuracy. For $n > 5$, we do not observe notable error improvement.*

3.3.6. Experimental Evaluation

In this subsection, we provide the implementation details, evaluation of our framework and comparison with the state-of-the-art hand pose estimation methods. The evaluation metrics are 3D joint location error and number of frames within certain thresholds. All the error metrics are reported in mm.
Implementation Details: We use Caffe [jia2014caffe], an open-source deep learning framework, to train the PalmCNN and the PoseCNN in our pipeline (see Figure 5.2). The networks run on a desktop using Nvidia Geforce GTX 1080 Ti GPU. The PalmCNN is trained on the combined dataset; see Section 3.3.2. The learning rate is set to 0.0001 with a batch size of 256 and 0.9 SGD momentum. One forward pass in the PalmCNN takes $4.5ms$. We train the PoseCNN on the NYU hand pose dataset [tompson2014real]. In order to segment the hand region from the raw depth input D_o, we use the estimated palm center from the PalmCNN. Given (u,v) and D_o, the hand region is cropped in 3D using a bounding box of size 300 and the camera focal length. The pre-processed image is of size 224 x 224 and the depth values are normalized to $[-1, 1]$. The 3D joints annotations J_{GT} in camera coordinates are also normalized to range $[-1, 1]$. We obtain FL_{GT}, FD_{GT} and KD_{GT} from the normalized J_{GT}. For training the PoseCNN, we use 0.001 learning rate with 0.9 SGD momentum and a batch size of 128. The forward pass for the PoseCNN takes $35ms$.
Method Evaluation: We comprehensively evaluate the PoseCNN and the PalmCNN. We first observe the effects of the proposed structural constraints on the accuracy of the estimated joint positions J. Second is to study the effects of interpolating n-points between the sparse joint positions.
Structural constraints: To this end, we train the following implementations of the PoseCNN on the NYU hand pose dataset which learns:

1. Joint positions J only.
2. Fingers lengths FL with J (i.e. $J \cup FL$).
3. Fingers inter-distances FD with J (i.e. $J \cup FD$).

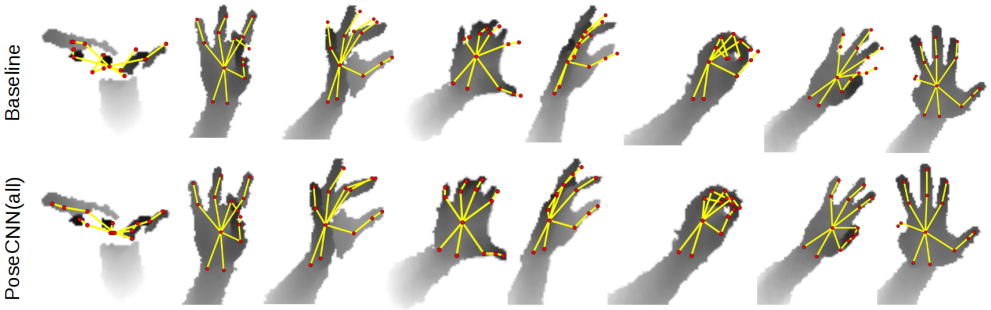


Figure 3.12.: *Qualitative evaluation of our PoseCNN. The top row shows the predicted hand joint positions overlaid on the preprocessed NYU depth images from our baseline implementation The bottom row shows the corresponding images with corrected joint positions from our PoseCNN(all) implementation.*

Methods	3D Joint Loc. Error	3D Palm Center Loc. Error
CoM	14.83mm	28.1mm
Ours (wo/colorization)	13.05mm	15.1mm
Ours (w/colorization)	**11.9mm**	**10.2mm**

Table 3.6.: *Influence of hand segmentation: Our hand segmentation method without colorization (wo/colorization) improves the joints prediction error by more than 1mm over center of hand mass (CoM) calculation method. Our method with colorization (w/colorization) further improves the accuracy by 19.75% over CoM.*

4. Kinematic distances KD with J (i.e. $J \cup KD$).

5. KD, FD and FL with J (i.e. $J \cup FL \cup FD \cup KD$).

Table 3.4 shows the quantitative results of the these implementations. In simplest form, the PoseCNN is trained to estimate 3D joint keypoints J only, we call this implementation as our *baseline* (PoseCNN(J)). On top of the *baseline*, we include the structural constraints one by one to observe the effects on the accuracy of estimated joints J. By including fingers lengths FL with J (i.e. PoseCNN($J \cup FL$)), we observe a small increase (3.28%) in accuracy of J. Inclusion of fingers inter-distances FD (PoseCNN($J \cup FD$)) and kinematic distances KD (PoseCNN($J \cup KD$)) improves the accuracy of the estimated J by 10.5% and 8.55% over the *baseline*, respectively. The best accuracy is achieved by the architecture which includes all the constraints (PoseCNN($J \cup FL \cup FD \cup KD$). It shows 15.13% improvement over the *baseline*.

Dense hand pose representation: We further experiment on the PoseCNN($J \cup FL \cup FD \cup KD$) by interpolating n-points between the pairs of parent and child joints in the kinematic hierarchy of the hand skeleton. Thereby, converting the sparse hand skeleton to dense representation. This leads to increase in number of joint positions depending on the value of n. Consequently, the size of the vectors FD and KD also increases. The quantitative results are summarized in Table 3.5. Our model (PoseCNN($J \cup FL$

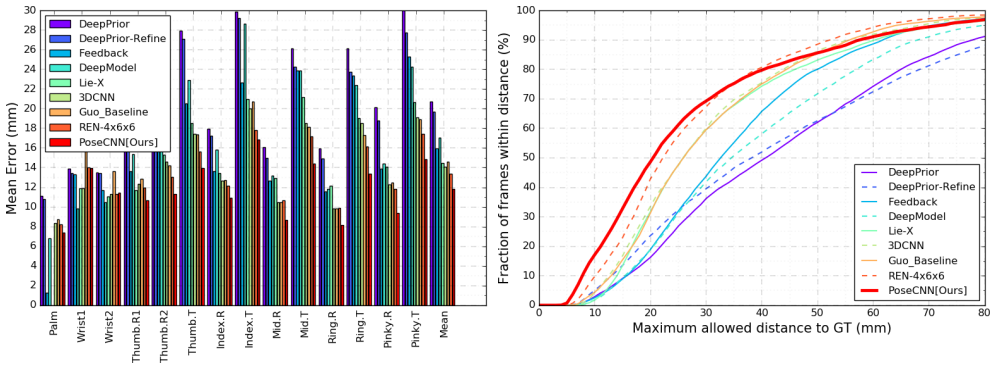


Figure 3.13.: *Quantitative comparison on the NYU test set [tompson2014real]. The right figure shows the fraction of frames within thresholds in mm. The left one shows the mean errors (mm) on individual joints of the NYU hand pose dataset. Our method PoseCNN(all) shows the average error of* 11.9*mm which is better than several state-of-the-art methods.*

$\cup\ FD \cup KD$)) with 5-points interpolation performs the best among the others. The results show improvement in accuracy of the estimated J using the interpolation strategy. Therefore, dense hand skeleton representation is useful for improved hand pose regression. For notational simplicity, we call this model as PoseCNN(*all*). This model improves the accuracy over the *baseline* by 21.71%.

The qualitative comparison of our *baseline* and PoseCNN(*all*) on the NYU dataset is shown in Figure 3.12. The estimated joint positions J are displayed on the sample preprocessed depth images. The predicted hand skeleton from our *baseline* architecture (PoseCNN(J)) can be of incorrect size (i.e. shorter or longer) due to independent learning of joint keypoints. Whereas, PoseCNN(*all*) which incorporates all the constraints along-with interpolated points produces more stable and reliable results. These results clearly show the effectiveness of our novel strategies, namely, structural constraints and the dense hand pose representation.

Hand segmentation: We evaluate our hand segmentation method (see Section 3.3.2) on the NYU dataset by studying the impact of colorization and comparing with the depth-thresholding followed by center of mass (CoM) computation method. The goal is to observe the effects of hand segmentation on the final 3D pose estimation accuracy. We train two different implementations of the PalmCNN. First, with colorized depth input (Ours(w/colorization)) and second, without colorization (Ours(wo/colorization)). Therefore, we get two different 3D palm centers for cropping the NYU depth images. Also, we obtain 3D palm centers from center of hand mass (CoM) calculation method. Using these three different palm centers, we obtain three distinct sets of pre-processed NYU training and testing frames. The PoseCNN(*all*) is trained for each of the three training sets. The effects on the accuracy of estimated J from the three PoseCNN(*all*) models are reported in Table 3.6. The best results are achieved by Ours(w/colorization) model. It shows an error improvement of 19.75% and

Methods	3D Joint Location Error
DeepPrior [oberweger2015hands]	20.75mm
DeepPrior-Refine [oberweger2015hands]	19.72mm
Crossing Nets [wan2017crossing]	15.5mm
Neverova et al. [neverova2017hand]	14.9mm
Feedback [oberweger2015training]	15.9mm
DeepModel [zhou2016model]	17.0mm
Lie-X [xu2017lie]	14.5mm
GuoBaseline [guo2017region]	14.6mm
3DCNN [ge2017robust]	14.11mm
REN [guo2017region]	13.3mm
DeepPrior++ [oberweger2017deepprior++]	12.3mm
PoseCNN(*all*) [**Ours**]	**11.9mm**

Table 3.7.: *Comparison with the state-of-the-art on the NYU test set [tompson2014real]: Our proposed model (PoseCNN(all)) exceeds in accuracy over the state-of-the-art hand pose estimation methods.*

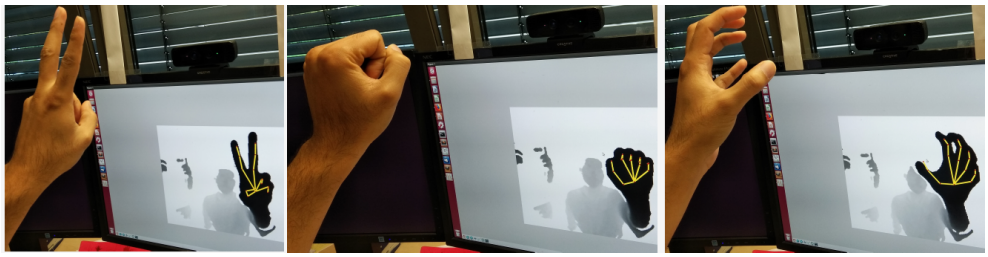


Figure 3.14.: *Real-time demonstration: We test our complete pipeline in real-time using the creative Senz3D depth camera. The camera is mounted on top of the display screen. The predicted hand skeleton (yellow) is overlaid on the depth image. Our system successfully tracks various challenging hand poses from frontal camera view.*

8.81% over the CoM and Ours(wo/colorization) methods; respectively.

Real-time demonstration: We test our complete framework in real-time using a single creative Senz3D depth camera [camera˙Senz3D]. The camera is placed on top of the display screen. Our framework tracks the hand movements with challenging poses as shown in Figure 3.14. For better generalization, we train our PoseCNN(*all*) architecture on the *HandSet* dataset [malik2017simultaneous]. This dataset combines several public hand pose datasets (e.g. ICVL, NYU and MSRA-2015) in a single unified format. The PalmCNN successfully estimates the hand palm center. Thereafter, the PoseCNN reliably estimates the joint positions. The predicted hand skeleton is displayed on the input depth frame. The run-time of the pipeline is $42ms$.

Comparison with the State-of-the-art: The state-of-the-art methods use either the ground truth palm center or the CoM localization approach to segment the hand region from a raw depth image. However, these approaches are

not feasible for practical applications. In contrast, our CNN-based hand segmentation method automatically segments the hand region from a raw depth image and outperforms the commonly used CoM method (see Table 3.6). We compare our best performing model, PoseCNN(*all*), with the state-of-the-art hand pose estimation methods i.e. DeepModel [zhou2016model], DeepPriorRefine [oberweger2015hands], Crossing Nets[wan2017crossing], Feedback [oberweger2015training], LieX[xu2017lie], 3DCNN [ge2017robust] and REN [guo2017region]. The quantitative results are shown in Table 4.2 and Figure 3.13. Our algorithm exceeds in accuracy over these methods. The results clearly indicate the benefits of our hand segmentation approach, the interpolation strategy and simultaneous learning of the hand structural constraints with the joint positions.

3.4. Conclusion

We present a novel hybrid method that outputs 3D hand pose as well as bone-lengths of the hand skeleton. We demonstrate the effectiveness of our approach on depth images captured from unseen subjects. Our method uses one CNN and a hybrid forward kinematics layer to predict 3D joint positions of the hand from a single depth image. The CNN estimates hand scale parameters (associated to bones of the hand skeleton) and pose parameters. In the hybrid forward kinematics layer, the initial hand skeleton is reshaped according to estimated hand scale parameters and a differentiable forward kinematic function is applied. Three different implementations of our method are introduced that describe the hand scale parameters in distinct ways. The training process is simple and efficient and proposed algorithm is well suited for real-time applications. Qualitative and quantitative results verify that our method achieves improved performance over the state-of-the-art hybrid methods.

Provided with the fact that the current CNN-based regression approaches do not explicitly incorporate hand structure, we present a novel structure-aware 3D hand pose regression pipeline from a single raw depth image. We propose two strategies which can be easily used to improve the hand pose estimation accuracy of any CNN-based discriminative method. To this end, a novel CNN-based hand segmentation method regresses the hand palm center which is used to segment the hand region from a raw depth image. Thereafter, a new CNN-based regression network simultaneously estimates the 3D hand pose and its structural constraints. Thereby, enforcing the hand pose structure during the training process. The proposed constraints help to maintain a structural relation between the estimated joint positions. Moreover, we study the effects of n-points interpolation between the pairs of parent and child joints in the kinematic chain of the hand skeleton. By performing extensive evaluations, we show the effectiveness of our approach. Experiments demonstrate competitive performance to the state-of-the-art hand pose estimation methods.

Chapter 4

Model-Based Deep Hand Shape and Pose Estimation

Contents

4.1. Introduction

Large amounts of training data, enriched with all possible variations in each of the challenging aspects stated above, are a key requirement for deep learning based methods to generalize well and achieve significant gains in accuracy. The recent real dataset [yuan2017bighand2] gathers a sufficient number of annotated images. However, it is very limited in hand shape variation (i.e. only 10 subjects). Progress in essential tasks such as estimation of hand surface and hand-part segmentation is hampered, as manual supervision for such problems at large scale is extremely expensive. In this paper, we generate a synthetic dataset that addresses these problems. It not only allows us to create virtually infinite training data, with large variations in shapes

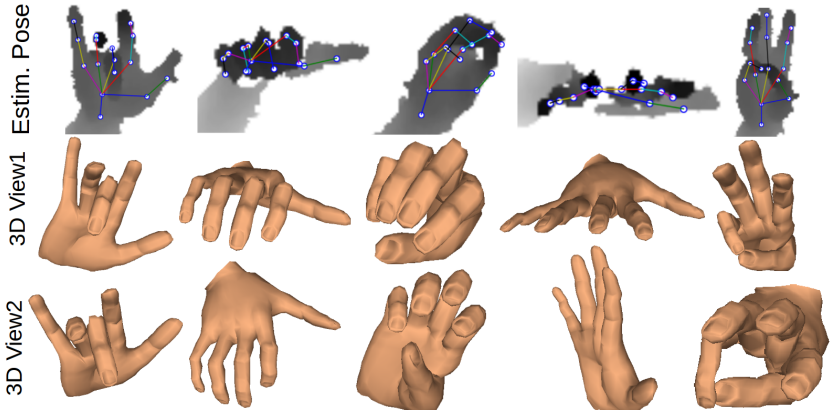


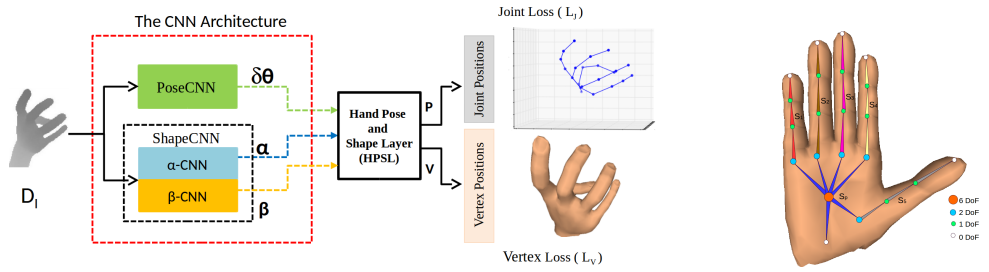
Figure 4.1.: *Real hand pose and shape recovery: We describe a deep network for recovering the 3D hand pose and shape of NYU depth images by learning from synthetic depth. Note that we infer 3D pose and shape even in cases of missing depth and occluded fingers.*

and view-points, but it also produces annotations that are highly accurate even in the case of occlusions. One weakness of synthetic datasets is their limited realism. A solution to this problem has been proposed by [shrivastava2017learning; mueller2017ganerated], where a generative adversarial training network is employed to improve the realism of synthetic images. However, producing realistic images is not the same problem as improving the recognition rates of a convolutional neural network (CNN) model. In this work, we address this latter problem, and specifically focus on a wide variation of hand shapes, including extreme shapes that are not very common (in contrast to [MANO:SIGGRAPHASIA:2017]). We present SynHand5M: a new million scale synthetic dataset with accurate ground truth joints positions, angles, mesh files, and segmentation masks of depth frames; see Figure 4.3. Our SynHand5M dataset opens up new possibilities for advanced hand analysis.

Previous deep learning-based methods addressed the problem of 3D hand pose estimation only. In this work, we proposed the first real-time deep network which jointly estimates a full hand mesh representation and 3D hand pose from a single depth image. We propose to embed a novel hand pose and shape layer (HPSL) inside the deep network to jointly optimize for 3D hand pose and shape surface. The proposed CNN architecture simultaneously estimates the hand pose parameters, bones scales and shape parameters. All these parameters are fed to the HPSL which implements not only a new forward kinematics function, but also the fitting of a morphable hand model and linear blend skinning to produce both 3D joint positions and 3D hand surface. The whole pipeline is trained in an end-to-end manner.

4.2. Method Overview

We aim to jointly estimate the locations of $J = 22$ 3D hand joints , and $\vartheta = 1193$ vertices of hand mesh from a single depth image D_I. Our hand skeleton in



(a) Algorithm pipeline (b) Our hand model

Figure 4.2.: *(a) An overview of our method for simultaneous 3D hand pose and surface estimation. A depth image D_I is passed through three CNNs to estimate pose parameters $\delta\theta$, bones scales α and shape parameters β. These parameters are sent to HPSL which generate the hand joints positions P and hand surface vertices V. (b) Our hand model with* 26 *DoFs overlaid with the neutral hand shape The bone colors illustrate* 6 *bone-length scales*

rest pose is shown in Figure 4.2(b). It has J hand joints defined on 26 DoFs. The hand root has 6 DoF; 3 for global orientation and 3 for global translation. All other DoFs are defined for joints articulations. The 26 dimensional pose vector is initialized for the rest pose, called θ_{init}. Any other pose Θ can be constructed by adding change $\delta\theta$ to the rest pose i.e. $\Theta = \theta_{init} + \delta\theta$. The bone-lengths B, are initialized by averaging over all bone-lengths of different hand shapes in our synthetic dataset. In order to add flexibility to the hand skeleton, 6 different hand bones scales, α, are associated to bone-lengths. Our hand mesh has ϑ vertices and 1184 faces. The neutral hand surface is shown in Figure 4.2(b). We use 7 hand shape parameters β which allow to formulate the surface geometry of a desired hand shape in reference pose; see Section 4.4.

Our pipeline is shown in Figure 4.2(a). Firstly, a new CNN architecture estimates $\delta\theta$, α and β given a depth input D_I. This architecture consists of PoseCNN which estimates $\delta\theta$ and ShapeCNN which estimates α and β. Thereafter, a new non-linear hand pose and shape layer (HPSL) performs forward kinematics, hand shape surface reconstruction and linear blend skinning. The outputs of the layer are 3D joint positions and hand surface vertices. These outputs are used to compute the standard euclidean loss for joint positions and vertices; see Equation A.9. The complete pipeline is trained end-to-end in a fully supervised manner.

4.3. Joint Hand Shape and Pose Estimation

In this section, we discuss the components of our pipeline which are shown in Figure 4.2(a). We explain the novel Hand Pose and Shape Layer (HPSL) in detail because it is the main component which allows to jointly estimate hand pose and shape surface.

4.3.1. The CNN Architecture

Our CNN architecture comprises of three parallel CNNs to learn $\delta\theta$, α and β, given D_I. The PoseCNN leverages one of the state-of-the-art CNN architecture [guo2017region] to estimate joint angles $\delta\theta$. However, the CNN was originally used to regress 3D hand joint positions. We refer the reader to [guo2017region] for network details of Region Ensemble (REN). In our implementation, the final regressor in REN outputs 26 dimensional $\delta\theta$. The ShapeCNN consists of two simpler CNNs similar to [oberweger2015hands]; called α-CNN and β-CNN. Each of them has 3 convolutional layers using kernels sizes 5,5,3 respectively. First two convolution layers are followed by max pool layers. The pooling layers use strides of 4 and 2. The convolutional layers generate 8 feature maps of size 12 x 12. Lastly, the two fully connected (FC) layers have 1024 neurons each with dropout ratio of 0.3. After the second FC layer, the final FC layers in α-CNN and β-CNN output 6 dimensional α and 7 dimensional β parameters respectively. All layers use the ReLu as activation function.

4.3.2. Hand Pose and Shape Layer (HPSL)

HPSL is a non-linear differentiable layer, embedded inside the deep network as shown in Figure 4.2(a). The task of the layer is to produce 3D joint positions $P \in \mathcal{R}^{3xJ}$ and vertices of hand mesh $V \in \mathcal{R}^{3x\vartheta}$ given the pose parameters Θ, hand bones scales α and shape parameters β. The layer function can be written as:

$$(P, V) = \mathbf{HPSL}(\Theta, \beta, \alpha) \tag{4.1}$$

We compute the respective gradients in the layer for back-propagation. The Euclidean 3D joint location and 3D vertex location losses are given as:

$$L_J = \frac{1}{2}\|P - P_{GT}\|^2 \quad , \quad L_V = \frac{1}{2}\|V - V_{GT}\|^2 \tag{4.2}$$

Where L_J and L_V are the 3D joint and vertex losses respectively. P_{GT} and V_{GT} are vectors of 3D ground truth joint positions and mesh vertices, respectively. Various functions inside the layer are detailed as follows:

Hand Skeleton Bone-lengths Adaptation: In order to adapt bone-lengths of hand skeleton during training over varying hand shapes in the dataset, [malik2017simultaneous] propose various bone-length scaling strategies. Following the similar approach, we assign a separate scale parameter for bone-lengths in palm s_p and 5 different scales for bones as shown in Figure 4.2(b). The HPSL acquires the scaling parameters $\alpha = [s_p, s_1, s_2, s_3, s_4, s_5]$ from the ShapeCNN during the training process.

Morphable Hand Model Formulation: Given the shape parameters β learned by our ShapeCNN, we reconstruct the hand shape surface by implementing a morphable hand model inside our HPSL. A morphable hand model $\Psi \in \mathcal{R}^{3x\vartheta}$ is a set of 3D vertices representing a particular hand shape. Any morphable hand model can be expressed as a linear combination of principle hand shape components, called morphable targets b_t [lewis2014practice]. Our principle hand shape components are defined for *Length*, *Mass*, *Size*, *Palm*

Length, *Fingers Inter-distance*, *Fingers Length* and *Fingers Tip-Size*. They represent offsets from a neutral hand shape b_0 similar to one shown in Figure 4.2(b). Each learned shape parameter β_t defines the amount of contribution of a principle shape components b_t towards formulation of final hand morphable model. Hence, a hand morphable model Ψ can be formulated using the following Equation:

$$\Psi(\beta) = b_0 + \sum_{t=1}^{7} \beta_t (b_t - b_0) \tag{4.3}$$

Forward Kinematics and Geometric Skinning: To estimate the 3D hand joints positions and surface vertices, we implement forward kinematics and geometric skinning functions inside our HPSL. As this layer is part of our deep network, it is essential to compute and back-propagate the gradients of these functions. The rest of this section addresses the definition of these functions and their gradients.

The deformation of the hand skeleton from the reference pose θ_{init} to the current pose Θ can be obtained by transforming each joint j_i along the kinematic chain by simple rigid transformations matrices. In our algorithm, these matrices are updated based on bones scales α and the changes in pose parameters $\delta\theta$ which are learned by our ShapeCNN and PoseCNN, respectively. The kinematics equation of joint j_i can be written as:

$$\begin{aligned} j_i &= \mathrm{F}_{j_i}(\Theta, \alpha) = \mathrm{M}_{j_i}[0,0,0,1]^T \\ &= \Big(\prod_{k \in S_{j_i}} [\mathrm{R}_{\phi_k}(\theta_k)] \times [\mathrm{T}_{\phi_k}(\alpha B)] \Big)[0,0,0,1]^T \end{aligned} \tag{4.4}$$

where M_{j_i} represents the transformation matrix from the zero pose (i.e. joint at position $[0,0,0,1]$) to the current pose. S_{j_i} is the set of joints along kinematic chain from j_i to the root joint and ϕ_k is one of the rotation axes of joint k.

For animating the 3D hand mesh, we use linear blend skinning [lewis2000pose] to deform the set of vertices ϑ according to underlying hand skeleton kinematic transformations. The skinning weights ω_i, define the skeleton-to-skin bindings. Their values represent the influence of joints on their associated vertices. Normally, the weights of each vertex are assumed to be convex (i.e. $\sum_{i=1}^{n} \omega_i = 1$) and $\omega_i > 0$. The transformation of a vertex $\mathrm{v}_x \in \Psi$ can be defined as:

$$\begin{aligned} \mathrm{v}_x &= \Upsilon_{\mathrm{v}_x}(\Theta, \beta, \alpha) = \sum_{i \in P_{\mathrm{v}(x)}} \omega_i C_{j_i} \mathrm{v}_x(\beta) \\ &= \sum_{i \in P_{\mathrm{v}_x}} \omega_i C_{j_i} (b_0^{\mathrm{v}_x} + \sum_{t=1}^{7} \beta_t (b_t^{\mathrm{v}_x} - b_0^{\mathrm{v}_x})) \end{aligned} \tag{4.5}$$

where P_{v_x} is the set of joints influencing the vertex v_x and C_{j_i} is the transformation matrix of each joint j_i from its reference pose θ_{init} to its actual position in the current animated posture. C_{j_i} can be represented as:

$$C_{j_i} = \mathrm{M}_{j_i} \mathrm{M}_{j_i}^{*-1} \tag{4.6}$$

where $\mathrm{M}_{j_i}^{*-1}$ defines the inverse of reference pose transformation matrix.

4.3.3. HPSL Gradients Computation

For backward-pass in the **HPSL**, we compute gradients of the following equation with respect to the layer inputs:

$$\mathbf{HPSL}(\Theta, \beta, \alpha) = (\ \boldsymbol{F}(\Theta, \alpha)\ , \Upsilon(\Theta, \beta, \alpha)\ ). \tag{4.7}$$

Each vertex $\mathrm{v}_x = \mathrm{HPSL}_{\mathrm{v}_x}(\Theta, \beta, \alpha)$ in the reconstructed hand morphable model Ψ is deformed using Equation 4.5. Hence, its gradients with respect to a shape parameter β_t can be computed as:

$$\frac{\partial(\mathrm{HPSL}_{\mathrm{v}_x})}{\partial \beta_t} = \sum_i \omega_i C_{j_i}(b_t^{v_x} - b_0^{v_x}) \qquad \text{for } t = 1, 2, \ldots, 7$$

According to Equation A.1, bones scales influence the joints positions and vertices positions. Hence, the resultant gradient with respect to a hand scale parameter α_s, can be calculated as:

$$\frac{\partial(\mathbf{HPSL})}{\partial \alpha_s} = \frac{\partial \mathbf{F}}{\partial \alpha_s} + \frac{\partial \Upsilon}{\partial \alpha_s} \qquad \text{for } s = 1, 2, \ldots, 6$$

To compute the partial derivative of $\mathbf{F}$ with respect to α_s, we need to derivate each joint with respect to its associated scale parameter. The gradient of a joint with respect to α_s, can be computed by replacing the scaled translational matrix containing α_s by its derivative and keep all other matrices same; see Appendix A.1. In a similar way, the gradient of a vertex v_x with respect to α_s can be computed by:

$$\begin{aligned}\frac{\partial \Upsilon_{\mathrm{v}_x}}{\partial \alpha_s} &= \sum_i \omega_i \frac{\partial C_{j_i}}{\partial \alpha_s} \mathrm{v}_x \\ &= \sum_i \omega_i [\mathrm{M}_{j_i} (\mathrm{M}_{j_i}^{*-1})' + (\mathrm{M}_{j_i})' \mathrm{M}_{j_i}^{*-1}] \mathrm{v}_x\end{aligned}$$

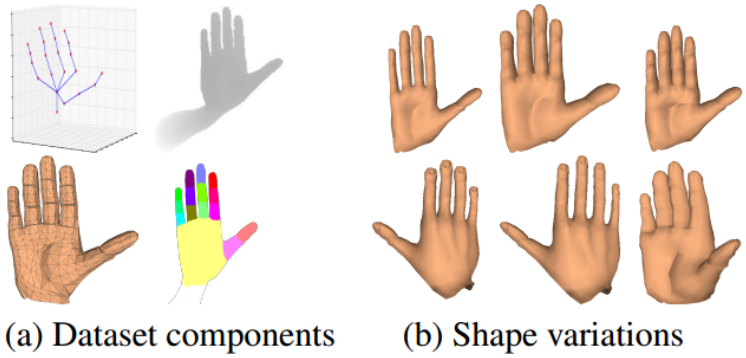
Likewise, for the pose parameters Θ, we compute the following equation:

$$\frac{\partial(\mathbf{HPSL})}{\partial \theta_p} = \frac{\partial \mathbf{F}}{\partial \theta_p} + \frac{\partial \Upsilon}{\partial \theta_p} \qquad \text{for } p = 1, 2, \ldots, 26$$

Accordingly, the derivative of a joint with respect to a pose parameter θ_p, is simply to replace the rotation matrix of θ_p by its derivation. And, the derivative of a vertex v_x with respect to θ_p is computed by:

$$\begin{aligned}\frac{\partial \Upsilon_{\mathrm{v}_x}}{\partial \theta_p} &= \sum_i \omega_i \frac{\partial C_{j_i}}{\partial \theta_p} \mathrm{v}_x \\ &= \sum_i \omega_i [(\mathrm{M}_{j_i})' \mathrm{M}_{j_i}^{*-1}] \mathrm{v}_x \qquad \text{for } p = 1, 2, \ldots, 26\end{aligned}$$

More details about the gradients computation can be found in the Appendix A.1.



(a) Dataset components (b) Shape variations

Figure 4.3.: *The SynHand5M dataset contains 5 million images. (a) The dataset ground truth components: hand poses (joints angles and 3D positions), depth maps, mesh files, and hand parts segmentation. (b) Samples illustrating the big variation in shape.*

4.4. Synthetic Dataset

There are two main objectives of creating our synthetic dataset. First is to jointly recover full hand shape surface and pose provided that there is no ground truth hand surface information available in public benchmarks; see Section 4.5.2. Second objective is to provide a training data with sufficient variation in hand shapes and poses such that a CNN model can be pre-trained to improve the recognition rates on real benchmarks; see Section 4.5.3. This problem is different from generating very realistic hand-shape, where a real-world statistical hand model [MANO:SIGGRAPHASIA:2017] can be applied. However, the variation in shape is more challenging for real-world databases e.g. BigHand2.2M [yuan2017bighand2] database was captured from only 10 users, and the MANO [MANO:SIGGRAPHASIA:2017] database was built from the contribution of 31 users. Instead, we generate a bigger hand shape variation which may not be present in a given cohort of human users.

Our SynHand5M dataset offers $4.5M$ train and $500K$ test images; see Figure 4.3(a) for SynHand5M components. SynHand5M uses the hand model generated by ManuelBastionLAB [ManuelBastionLAB] which is a procedural full-body generator distributed as add-on of the Blender [blender] 3D authoring software. Our virtual camera simulates a Creative Senz3D Interactive Gesture Camera [camera˙Senz3D]. It renders images of resolution 320x240 using diagonal field of view of 74 degrees. In the default position, the hand palm faces the camera orthogonally and the fingers point up. We procedurally modulate many parameters controlling the hand and generate images by rendering the view from the virtual camera. The parameters characterizing the hand model belong to three categories: hand shape, pose and view point.

Without constraints the hand generator can easily lead to impossible hand shapes. So, in order to define realistic range limits for modulating hand shapes, we relied on the DINED [molenbroek04dined] anthropometric database. It is a repository collecting the results of several anthropometric databases, including the CAESAR surface anthropometry survey [robinette˙caesar˙1999]. We

manually tuned the ranges of the 7 hand shape parameters (see Section 4.3.2) in order to cover 99% of the measured population in this dataset; see the Appendix A.2 for more details.

To modulate the hand pose, we manipulate the 26 DoFs of our hand model; see Figure 4.2(b). For each finger, rotations are applied to flexion of all phalanges plus the abduction of the proximal phalanx. Additionally, in order to increase the realism of the closed fist configuration, the roll of middle, ring, and pinky fingers is derived from the abduction angle of the same phalanx. The rotation limits are set to bring the hand from a closed fist to an over-extended aperture, respecting anatomical constraints and avoiding the fingers to enter the palm.

The hand can rotate about three DoFs to generate different view points: roll around its longitudinal axis (i.e. along the fingers), rotate around the palm orthogonal axis (i.e. rolling in front of the camera), and rotate around its transversal axis (i.e. flexion/extension of the wrist).

4.5. Experimental Evaluation

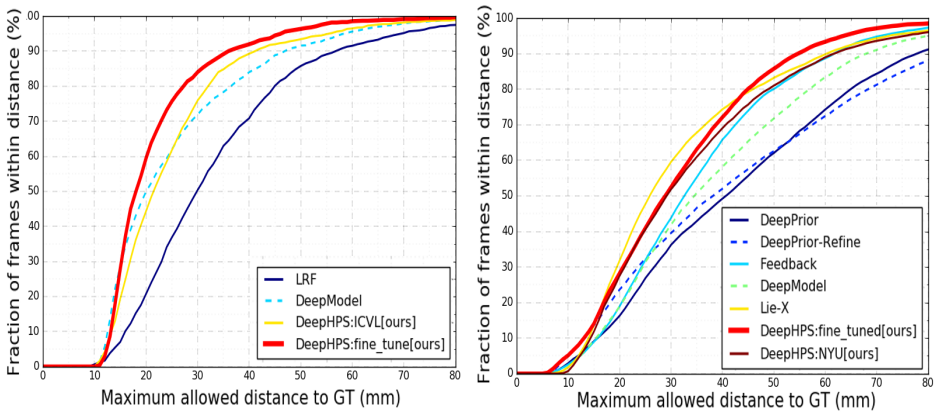
In this section, we provide the implementation details, quantitative and qualitative evaluations of the proposed algorithm and the proposed dataset. We use three evaluation metrics; mean 3D joint location error (JLE), 3D vertex location error (VLE) and percentage of images within certain thresholds in mm.

As mentioned earlier (Chapter 2), discriminative methods neither explicitly account for the hand shapes nor consider kinematics constraints. Moreover, in contrast to hybrid methods, discriminative methods generalize poorly to unseen hand shapes; see [yuan2018depth]. Our proposed hybrid method does not exceed in accuracy over recent discriminative works but, it does not suffer from such limitations. Therefore, it is not fair to compare with these methods. However, we compare with the state-of-the-art hybrid methods and show improved performance. Notably, we propose the first algorithm that jointly regresses hand pose, bone-lengths and shape surface in a single network.

4.5.1. Implementation Details

For training, we pre-process the raw depth data for standardization and depth invariance. We start by computing the centroid of the hand region in the depth image. The obtained 3D hand center location (i.e. palm center) is used to crop the depth frame. The camera intrinsics (i.e. focal length) and a bounding box of size 150, are used during the crop. The pre-processed depth image is of size 96×96 and in depth range of $[-1, 1]$. The annotations in camera coordinates are simply normalized by the bounding box size and clipped in range $[-1, 1]$.

We use Caffe [jia2014caffe] which is an open-source training framework for deep networks. The complete pipeline is trained end-to-end until convergence. The learning rate was set to 0.00001 with 0.9 SGD momentum. A batch size of 256 was used during the training. The framework is executed on a



(a) ICVL (b) NYU

Figure 4.4.: *Quantitative evaluation. (a) show the results of our algorithm (DeepHPS) on ICVL test set, when trained on ICVL and fine-tuned on ICVL. (b) is the same but with NYU. To fine-tune, we pretrain DeepHPS on our SynHand5M. Our results on ICVL and NYU show improved accuracy over the state-of-the-art hybrid methods. The curves show the number of frames in error within certain thresholds.*

desktop equipped with Nvidia Geforce GTX 1080 Ti GPU with 16GB RAM. One forward pass takes $3.7ms$ to generate 3D hand joint positions and shape surface. For simplicity, we name our method as DeepHPS.

4.5.2. Algorithm Evaluation

In this subsection, we evaluate our complete pipeline using the SynHand5M. Moreover, we devise a joint training strategy for both real and synthetic datasets to show qualitative hand surface reconstruction of real images.

Evaluation on the synthetic dataset: The complete pipeline is trained end-to-end using SynHand5M for pose and shape recovery. For fair comparison, we train the state-of-the-art model based learning methods [zhou2016model; malik2017simultaneous] on SynHand5M. [malik2017simultaneous] works for varying hand shapes in contrast to the closely related method [zhou2016model]. The quantitative results are shown in Table 4.1. Our method clearly exceeds in accuracy over the compared method and additionally reconstructs full hand surface. The qualitative results are shown in Figure 4.6. The estimated 22 joint positions are overlaid on the depth images while the reconstructed hand surface is shown using two different views named as ***3D View1*** and ***3D View2***. For better visualization, view2 is similar to ground truth view. The results demonstrate that our DeepHPS model infers correct hand shape surface even in cases of occlusion of fingers and large variation in view points.

Evaluation on the NYU real dataset: In order to jointly train our whole pipeline on both real and synthetic data, we found 16 closely matching common joint positions in SynHand5M and the NYU dataset. These common joints are different from the 14 joints used for the public comparisons [tompson2014real].

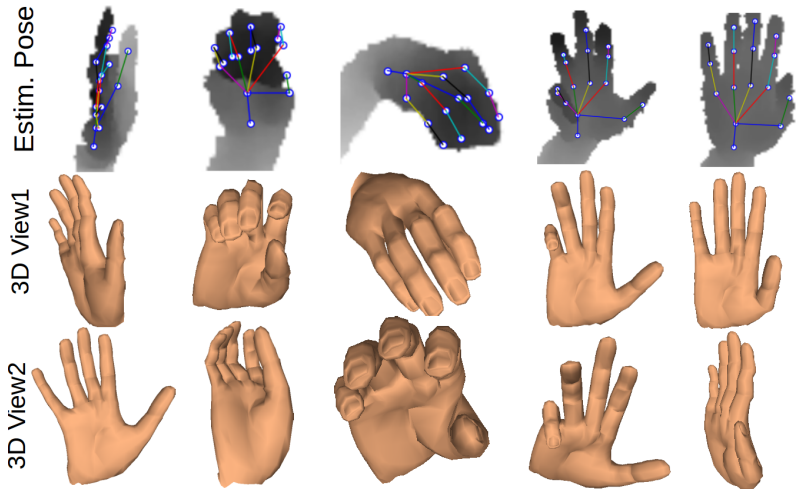


Figure 4.5.: *Real hand pose and shape recovery: More results on hand pose and surface reconstruction of NYU[tompson2014real] images. Despite of unavailability of ground truth hand mesh vertices, our algorithm produces plausible hand shape.*

The loss equation is;

$$L = L_J + \mathbb{1}L_V \tag{4.8}$$

where $\mathbb{1}$ is an indicator function which specifies whether the ground truth for mesh vertices is available or not. In our setup, it is 1 for synthetic images and 0 for real images. For real images, backpropagation from surface reconstruction part is disabled.

The qualitative pose and surface shape results on sample NYU real images are shown in Figure 4.1 and 4.5. Despite of the missing ground truth surface information and presence of high camera noise in NYU images, the resulting hand surface is plausible and the algorithm performs well in case of missing depth information and occluded hand parts.

Method \Error(mm)	3D Joint Loc.	3D Vertex Loc.
DeepModel [zhou2016model]	11.36	–
HandScales [malik2017simultaneous]	9.67	–
DeepHPS [**Ours**]	**6.3**	**11.8**

Table 4.1.: *Quantitative Evaluation on SynHand5M: We show the 3D joint and vertex locations errors(mm). Our method additionally outputs mesh vertices and outperforms model based learning methods [zhou2016model; malik2017simultaneous].*

4.5.3. Comparison on Public Benchmarks

The public benchmarks do not provide ground truth hand mesh files. Therefore, we provide quantitative results for pose inference on two of the real hand pose datasets (i.e. NYU and ICVL). For comparisons, NYU dataset use 14

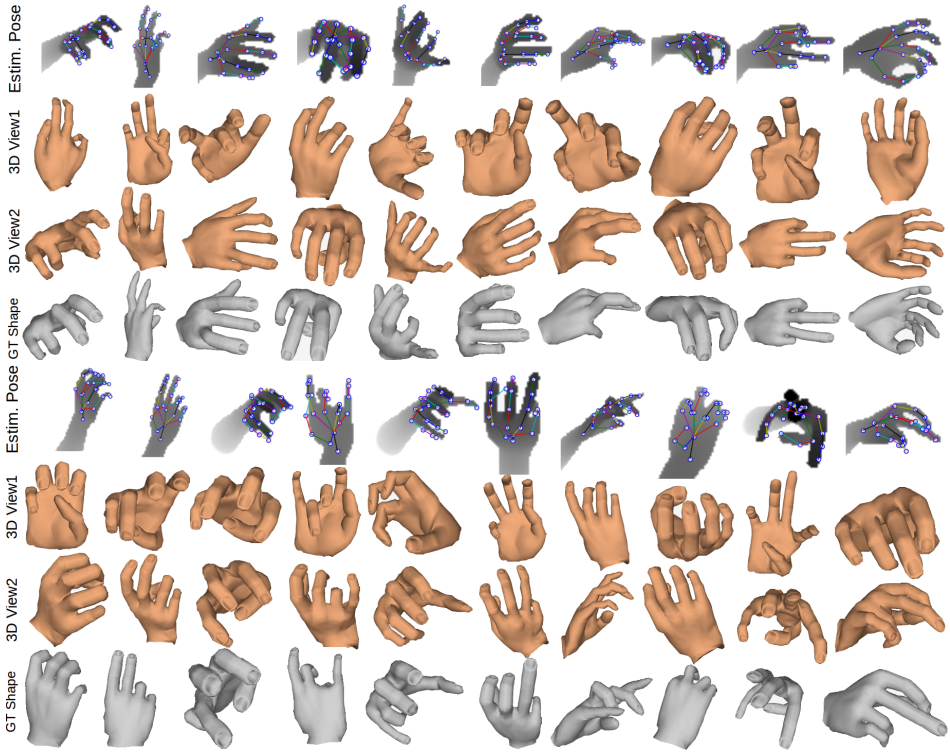


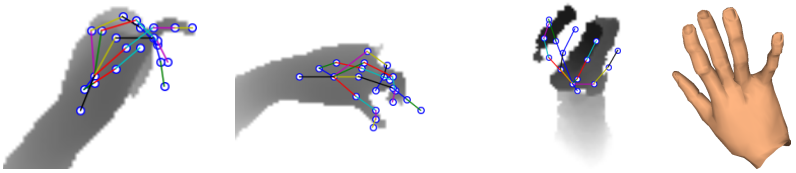
Figure 4.6.: *Synthetic hand pose and shape recovery: We show example estimated hand poses overlaid with the preprocessed depth images from our SynHand5M. We show the reconstructed surface from two different views (yellow) and the ground truth surface (gray). 3D View2 is similar to the ground truth view. Our algorithm infers correct 3D pose and shape even in very challenging condition, like occlusion of several fingers and large variation in view points.*

joint positions [tompson2014real] whereas ICVL dataset [tang2014latent] use 16 joint positions.

Our DeepHPS algorithm is trained on NYU and ICVL individually, called DeepHPS:NYU and DeepHPS:ICVL models. Then, we fine-tune the pre-trained DeepHPS (on SynHand5M dataset) with the NYU and ICVL, we call DeepHPS:fine-tuned models. The 3D joint location errors of the trained models are calculated on 8252 NYU and 1596 ICVL test images respectively. The quantitative results are shown in Figure 4.4 and Tables 4.2 and 4.3. DeepHPS:fine-tuned models achieve an error improvement of 13.3% and 10.12% over DeepHPS:ICVL and DeepHPS:NYU models respectively.

On the ICVL and NYU datasets, we achieve improvement in the joint location accuracy over the state-of-the-art hybrid methods.

Failure case: Our framework works well in case of missing depth information and occlusions. However, under severe occlusions and a lot of missing depth information, it may fail to detect the correct pose and shape; see Figure 4.7.



(a) (b)

Figure 4.7.: *Failure case:(a) incorrect pose due to highly occluded hand parts. (b) incorrect pose and shape due to significant missing depth information.*

Methods	3D Joint Location Error
DeepPrior [oberweger2015hands]	20.75mm
DeepPrior-Refine [oberweger2015hands]	19.72mm
Crossing Nets [wan2017crossing]	15.5mm
Feedback [oberweger2015training]	15.9mm
DeepModel [zhou2016model]	17.0mm
Lie-X [xu2017lie]	14.5mm
DeepHPS:NYU [Ours]	15.8mm
DeepHPS:fine-tuned [**Ours**]	**14.2mm**

Table 4.2.: *Quantitative comparison on NYU [tompson2014real]: Our fine-tuned DeepHPS model on the NYU dataset shows the state-of-the-art performance among hybrid methods.*

4.6. Conclusion

We demonstrate the simultaneous reconstruction of hand pose and shape surface from a single depth image. For training, we synthetically generate a large scale dataset with accurate joint positions, segmentation masks and hand meshes of depth images. Our dataset will be a valuable addition for training and testing CNN-based models for 3D hand pose and shape analysis. Furthermore, it improves the recognition rate of CNN models on hand pose datasets. In our algorithm, intermediate parametric representations are estimated from a CNN architecture. Then, a novel hand pose and shape layer is embedded inside the deep network to produce 3D hand joint positions and shape surface. Experiments show improved accuracy over the state-of-the-art hybrid methods. Furthermore, we demonstrate plausible results for the re-

Methods	3D Joint Location Error
LRF [tang2014latent]	12.57mm
DeepModel [zhou2016model]	11.56mm
Crossing Nets [wan2017crossing]	10.2mm
DeepHPS:ICVL [Ours]	10.5mm
DeepHPS:fine-tuned [**Ours**]	**9.1mm**

Table 4.3.: *Quantitative comparison on ICVL [tang2014latent]: The DeepHPS model fine-tuned on the ICVL dataset outperforms the state-of-the-art hybrid methods.*

covery of hand shape surface on real images. Improving the performance of CNN-based hybrid methods is a potential research direction. These methods bear a lot of potential due to their inherent stability and scalability. In future, we wish to extend our dataset with wider view points coverage, object interactions and RGB images. Another aspect for future work is predicting fine-scale 3D surface detail on the hand, where real-world statistical hand models [MANO:SIGGRAPHASIA:2017] possibly give better priors.

Chapter 5

Hand Shape and Pose Estimation using 2D Convolutional Networks

Contents

5.1. Introduction

Discriminative approaches do not exploit the hand structure well, which may result in poor estimation of 3D pose on unseen data [yuan2018depth]. On the other hand, structured hand pose estimation methods either implicitly incorporate hand structure [ge2018point; oberweger2017deepprior++]

or embed a kinematic hand model in a deep network [zhou2016model; malik2017simultaneous; dibra2017refine]. However, the kinematic model parameterization is highly nonlinear, which is difficult to optimize in deep networks [sun2017compositional]. In contrast, we propose a simple and effective structured 3D pose estimation approach that estimates 3D bone vectors using a CNN, which are converted to 3D hand joint positions by a bone-to-joint layer. The novel layer allows resolving the limitations of both discriminative and structured methods as it preserves the hand structure and produces more accurate 3D hand pose because learning bones representation is easier than learning angles of kinematic model [sun2017compositional].

On the other hand, deep learning-based simultaneous estimation of 3D hand shape and pose is a novel problem that has not been well investigated yet. This task is highly challenging given the fact that ground truth of real hand shapes is not available. Manual annotation of 3D hand shape is highly time consuming, laborious and sub-optimal. Malik et al. [malik2018deephps] employed the standard linear blend skinning (LBS) function using fixed set of synthetic blendshape targets for hand shape reconstruction which limits this approach to incorporate nonlinear and large variations in hand shapes. [Adnane20193d] proposed a structured hand shape and pose estimation method from monocular RGB input using the statistical MANO hand model [romero2017embodied]. However, this approach is also limited by a small training data and the LBS based on linear bases. Recently, Ge et al. [ge20193d] proposed a weakly-supervised regression based approach that highly depends on a pseudo ground truth of real hand shapes, which is obtained using a pre-trained model with labeled synthetic RGB dataset. Moreover, their 3D pose estimation accuracy directly depends on the quality of real hand shape estimation.

In this chapter, we propose two different hand shape and pose estimation approaches based on 2D convolutions. The first is a novel weakly-supervised algorithm that estimates both 3D hand mesh and pose from a single depth image by learning from unlabeled real data and labeled synthetic data. We argue that learning dense 3D hand mesh from sparse 3D hand joint positions along with a depth synthesizer as a source of weak-supervision is very effective and produces accurate and reasonable hand shapes. The second is simple and effective real-time approach for regressing the sparse hand joints and hand structure constraints, as well as a 3D hand mesh for both egocentric and third person viewpoints. The structure constraints (i.e., bone lengths, kinematic distances, and inter-finger distances [malik2018structure]) are simultaneously optimized to maintain the structural relationships between the estimated joints [sun2017compositional]. Also, none of the existing hand pose datasets provide egocentric ground truth shape information. Annotating real images with shape representations is highly challenging and the results can be sub-optimal. Therefore, we produced SynHandEgo as the first egocentric synthetic hand dataset containing accurate ground truth data for 3D meshes, 3D poses, and color segmentations of hand parts. This dataset can facilitate the reconstruction of egocentric hand shapes using learning-based algorithms given that the annotation of real depth images with accurate hand shape in-

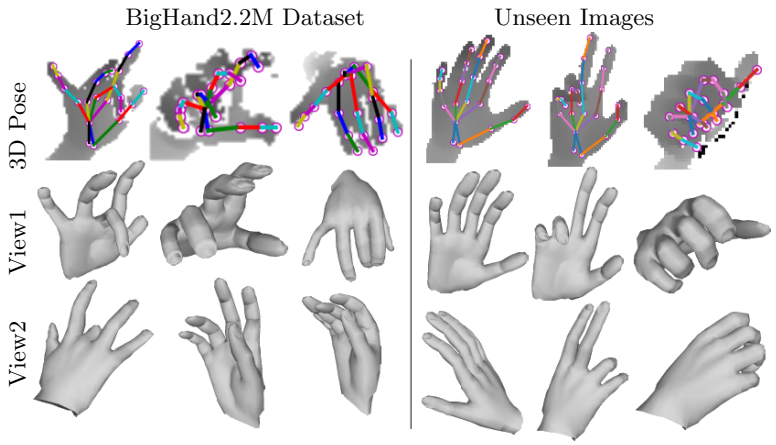


Figure 5.1.: *Our proposed approach accurately recovers full 3D hand mesh and 3D pose from a single depth image. We show our results on real dataset as well as on unseen images from real-time demo.*

formation is almost impossible due to severe occlusion issues.

Our joint training strategy with real and synthetic datasets allows shapes to be reconstructed from real depth images without requiring ground truth hand shape data. Our approaches successfully recovered plausible hand shapes from two real benchmarks, i.e., NYU and BigHand2.2M (see Figures 5.1 and 5.12). Experiments demonstrated that our method improved the hand pose estimation accuracy based on the NYU dataset compared with the existing methods that produce more than joint positions.

5.2. Weakly-Supervised Approach

In this section, we explain our first approach based on 2D convolutions for simultaneous hand shape and pose estimation trained in a weakly-supervised manner.

5.2.1. Method Overview

Figure 5.2 shows an overview of our method. Given a single gray scale cropped depth image $\mathcal{D}_\mathcal{I}$, the task is to estimate 3D hand joint positions $\mathcal{J} \in \mathcal{R}^{3\times \mathrm{P}}$ and 3D hand mesh vertices $\mathcal{V} \in \mathcal{R}^{3\times \mathrm{N}}$, where P represents the number of joints and N = 1193 is the number of mesh vertices. $\mathcal{D}_\mathcal{I}$ is passed to a CNN-based bones regressor, which directly regresses 3D bone vectors $\mathcal{B} \in \mathcal{R}^{3\times(\mathrm{P}-1)}$. A bone $b_n \in \mathcal{R}^3$ is the 3D offset of the nth joint (j_n) relative to its immediate parent joint $(j_{parent(n)})$, and can be calculated as:

$$b_n = j_n - j_{parent(n)} \tag{5.1}$$

The direction of bone vector is from parent joint to child joint in the kinematic chain of hand skeleton. as shown in Figure 5.3 (right). $\mathcal{B}$ is an intermediate parametric representation of joints that is fed to a parameter free

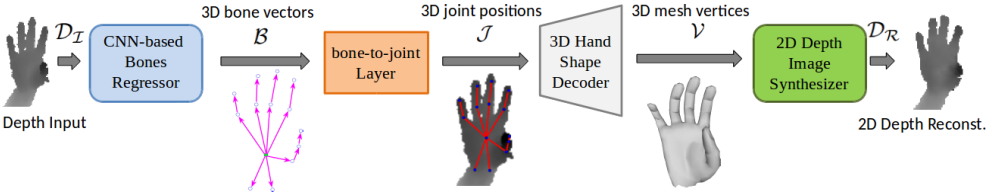


Figure 5.2.: *Illustration of our complete method. A hand center cropped depth image $\mathcal{D}_\mathcal{I}$ is provided to a CNN-based bones regressor, which estimates the 3D bone vectors $\mathcal{B}$ as an intermediate representation. $\mathcal{B}$ is passed to a non-parametric bone-to-joint layer, which converts the 3D bone vectors to 3D joint positions ($\mathcal{J}$). Then, a linear hand shape decoder converts the sparse hand joints positions to dense mesh vertices ($\mathcal{V}$). Finally, a 2D depth image synthesizer reconstructs depth image $\mathcal{D}_\mathcal{R}$ from reconstructed $\mathcal{V}$. The depth synthesizer acts as a weak-supervision in training and is excluded during testing.*

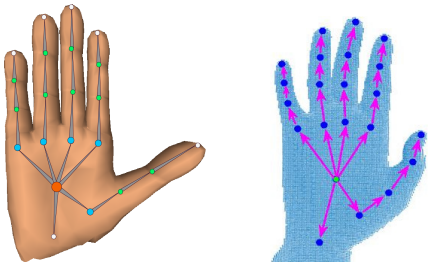


Figure 5.3.: *(left) SynHand5M dataset hand model; and (right) bone vectors and joints of BigHand2.2M dataset hand model.*

bone-to-joint layer. This layer allows preserving the structure of hand skeleton (see Section 5.2.2). For notation simplicity, CNN-based bones regressor and bone-to-joint layer are collectively named as **Module 1**. Thereafter, a linear 3D hand shape decoder (**Module 2**) decodes dense mesh $\mathcal{V}$ from sparse pose $\mathcal{J}$ (see Section 5.2.3). In the final stage, a 2D depth synthesizer (**Module 3**) produces a synthesized depth image $\mathcal{D}_\mathcal{R}$ from $\mathcal{V}$ (see Section 5.2.4), which acts as weak-supervision in training. All modules are individually trained and then collectively fine-tuned using mixed synthetic and real datasets (see Section 5.2.5). Module 3 is excluded in testing phase.

5.2.2. Structured Hand Pose Estimation

In this section, we discuss Module 1 of our pipeline. For better generalized performance, it is important to include hand structure while estimating 3D joint positions [yuan2018depth]. We respect this requirement by introducing a simple bone-to-joint layer, which is embedded inside deep learning. The CNN-based bones regressor estimates intermediate parametric representation $\mathcal{B}$. The CNN architecture is similar to that in [wang2018region], which was originally used for directly estimating $\mathcal{J}$. We select this architecture because of its scalability and its highly effective region ensemble (REN) strategy of boosting the accuracy of positions estimation (we refer the reader to [wang2018region]

for architecture details). Here, we use an ensemble of nine regions and modify the last fully connected (FC) layer to output $\mathcal{B}$. The learning of bones is fully supervised. The bones loss $\mathcal{L}_{\mathcal{B}}$ is given by the following equation:

$$\mathcal{L}_{\mathcal{B}} = \frac{1}{2} \left\| \mathcal{B} - \mathcal{B}_{GT} \right\|^2 \tag{5.2}$$

where $\mathcal{B}_{GT}$ is a vector of ground truth bones.

The proposed bone-to-joint layer is a differentiable and parameter free layer. The task of this layer is to produce $\mathcal{J}$ given the estimated $\mathcal{B}$ from CNN-based bones regressor. The transformation for one joint j_i can be represented as:

$$j_i = \Big(\prod_{k \in P_{j_i}} \mathrm{T}_{\phi_k}(\mathcal{B}_k) \Big) [0, 0, 0, 1]^{\mathrm{T}} \tag{5.3}$$

where P_{j_i} is the set of parent joints of j_i in the kinematic chain. T represents a 4x1 translation matrix. ϕ_k represents the translation along k-axis, $\mathcal{B}_k$ is translational value corresponding to ϕ_k and $[0, 0, 0, 1]$ is the root joint (i.e., palm center) position. Notably, there are no rotation matrices involved in the transformation since the articulations are represented only by 3D bone vectors. The Euclidean joint locations loss $\mathcal{L}_{\mathcal{J}}$ is given as:

$$\mathcal{L}_{\mathcal{J}} = \frac{1}{2} \left\| \mathcal{J} - \mathcal{J}_{GT} \right\|^2 \tag{5.4}$$

where $\mathcal{J}_{GT}$ is a vector of ground truth joint positions. The gradient computations for the bone-to-joint layer are provided in the Appendix A.3.

5.2.3. Hand Shape Decoding

As mentioned above, the major bottleneck in 3D hand shape recovery is the missing shape ground truth of real images because manual annotation of real images for shape is a highly time consuming and sub-optimal process. Hence, there is a need to effectively utilize sparse 3D joint annotations in the real datasets in order to learn a reasonable hand shape. In this respect, we propose a novel hand shape decoding method, which is inspired by unsupervised autoencoders [schmidhuber2015deep; baldi2012autoencoders]. The 3D hand pose can be considered as the sparse representation of dense hand mesh. We exploit this inherent relationship between pose and mesh and employ only the decoding part of a linear autoencoder, which maps pose to shape by learning from synthetic data. The architecture of our hand shape decoder is shown in Figure 5.4. Given the latent pose representation $\mathcal{J}$, the reconstructed mesh $\mathcal{V}$ can be represented as:

$$\mathcal{V} \sim \mathrm{Dec}(\mathcal{J}) = p(\mathcal{V}_{GT} | \mathcal{J}) \tag{5.5}$$

where $p(\mathcal{V}_{GT} | \mathcal{J})$ is the decoded distribution. The decoder tries to reconstruct $\mathcal{V}$ as close as possible to the ground truth $\mathcal{V}_{GT}$. Both $\mathcal{J}$ and $\mathcal{V}$ are in the range $[-1, 1]$, therefore *tanh* is used as an activation function after every FC layer. The reconstruction loss $\mathcal{L}_{\mathcal{R}}$ can be written as:

$$\mathcal{L}_{\mathcal{R}} = \frac{1}{2} \left\| \mathcal{V} - \mathcal{V}_{GT} \right\|^2 \tag{5.6}$$

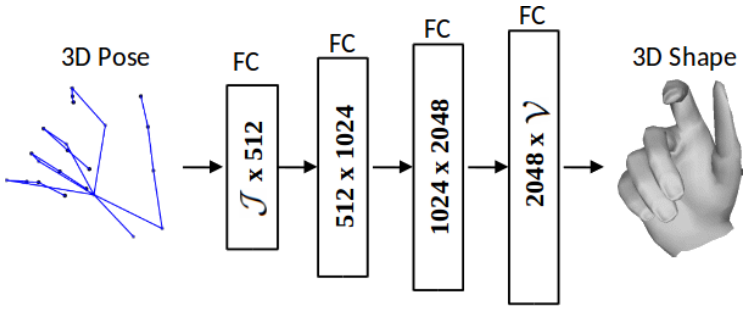


Figure 5.4.: *Architecture of the proposed linear 3D hand shape decoder (Module 2), which estimates the dense 3D hand mesh representation (shape) from the given sparse 3D joint positions (pose).*

The training details with mixed real and synthetic data are presented in Section 5.2.5.

5.2.4. Depth Image Synthesis

As discussed above, weak-supervision is an essential component of our pipeline due to the missing shape ground truth of real images. We provide a source of weak-supervision on shape learning by utilizing the input depth image $\mathcal{D}_\mathcal{I}$ and synthesize $\mathcal{D}_\mathcal{R}$ from the reconstructed $\mathcal{V}$, as shown in Figure 5.2. Inspired by the approaches proposed in [oberweger2015training; cai2018weakly], which synthesize a depth map from sparse joint positions, we build the architecture for our depth image synthesizer to generate depth image from richer dense mesh representation, as shown in Figure 5.5. It consists of six deconvolution layers, which use *ReLu* as activation functions, except the last layer that uses *tanh*. The sizes of the 2D feature maps increase gradually but decrease in number until $\mathcal{D}_\mathcal{R}$ of size 96×96 is finally synthesized. The kernel sizes for the deconvolution layers are 5×5, 6×6, 9×9, 12×12, 27×27 and 51×51, respectively. We use standard L2 norm to minimize the difference between the synthesized $\mathcal{D}_\mathcal{R}$ and ground truth $\mathcal{D}_\mathcal{I}$ as:

$$\mathcal{L}_\mathcal{D} = \frac{1}{2} \left\| \mathcal{D}_\mathcal{R} - \mathcal{D}_\mathcal{I} \right\|^2 \tag{5.7}$$

The samples of synthesized depth images of NYU [tompson2014real], BigHand2.2M [yuan2017bighand2] and SynHand5M [malik2018deephps] datasets are shown in Figure 4 of the Appendix A.5.

5.2.5. Network Training

This section gives details about the data preprocessing and training methodology of our complete pipeline. The raw depth images are first hand center cropped based on center of hand mass (CoM). Following [guo2017region], CoM is calculated by depth thresholding assuming that hand is the closest object to the camera. For normalization of depth images, the cropping is done along both spatial and depth dimensions using a bounding box of fixed size 150.

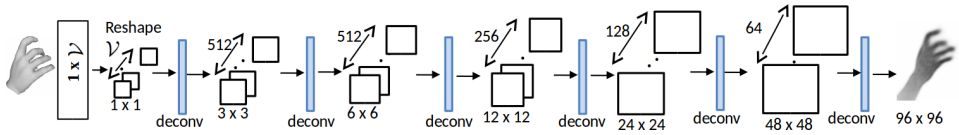


Figure 5.5.: *Architecture of the proposed 2D depth image synthesizer which is capable of reconstructing a 2D depth image from the given input 3D hand mesh representation by expanding the size of feature maps in both dimensions and finally producing a single gray scale depth frame. deconv stands for transposed convolutions.*

The final preprocessed image size is 96×96 and is normalized in range $[-1, 1]$. Both joint positions and mesh vertices are made relative to palm center (i.e., CoM) and divided by the bounding box size. After the normalization, all annotations lie in range $[-1, 1]$. For generalization of the network, we augment training data by applying rotation and scaling in ranges $[-45°, 45°]$ and $[0.8, 1.1]$, respectively.

After the preprocessing, we train each module of our network individually, and collectively fine-tune them in an end-to-end manner (see Figure 5.2). We use Caffe [jia2014caffe] for the network training. Module 1 (see Figure 2 in the Appendix A.4) is trained for jointly optimizing $\mathcal{B}$ and $\mathcal{J}$ in a fully-supervised manner, using a learning rate (LR) of 0.01 and a batch size of 128. Module 2 (see Figure 5.4) is jointly trained with real and synthetic datasets, using ground truth annotations pair ($\mathcal{J}$,$\mathcal{V}$) in a semi-supervised manner. Since $\mathcal{V}$ is not available for real datasets, we use a simple indicator function layer which implements the following equation:

$$\mathcal{L} = \mathbb{1}\mathcal{L}_{\mathcal{R}} \tag{5.8}$$

where $\mathbb{1}$ is an indicator function. This layer sends $\mathcal{V}$ to the loss layer only for synthetic images using a binary flag value, which is 1 for synthetic and 0 for real. The gradients flow in backward pass is disabled for real data. LR is set to 10^{-4} with a batch size of 128. Module 3 (see Figure 5.5) is individually trained to synthesize $\mathcal{D}_{\mathcal{R}}$ using only the synthetic dataset because of unavailability of $\mathcal{V}$ for real data. The training pair is ground truth ($\mathcal{V}$,$\mathcal{D}_{\mathcal{I}}$). LR of 10^{-5} is used with a batch size of 64. The models run on a desktop PC equipped with Nvidia GeForce GTX 1070 GPU. All networks are trained until convergence. Finally, all modules are put together in a complete pipeline (Figure 5.2) and fine-tuned on mixed real and synthetic datasets. The overall loss equation of the network can be written as:

$$\mathcal{L}_{\text{Full}} = \mathcal{L}_{\mathcal{B}} + \mathcal{L}_{\mathcal{J}} + \mathbb{1}\mathcal{L}_{\mathcal{R}} + \mathcal{L}_{\mathcal{D}} \tag{5.9}$$

A batch size of 128 is used with an LR of 10^{-7} and the full pipeline is trained in an end-to-end manner. Module 3 is excluded during the testing. One forward pass takes only **2.9** ***ms*** to produce both 3D hand mesh and pose.

5.2.6. Datasets, Baselines and Evaluation Metrics

None of the existing real hand pose datasets provide ground truth hand shape information. Therefore, we qualitatively evaluated the recovered 3D real hand mesh using two datasets: NYU [tompson2014real] and BigHand2.2M [yuan2017bighand2].

For notation simplicity, we denote the train set of NYU dataset as $\mathcal{T}_{\mathcal{N}}$. BigHand2.2M is the largest real dataset, which provides 956 K training depth frames captured from 10 different subjects. The test set for the pose estimation task contains 296 K images. However, the annotations for the test set are not available. Hence, for completeness, we first selected 90% of 956 K (i.e., 860 K) as train set ($\mathcal{T}_{\mathcal{B}}$) and the remaining frames (i.e., 96 K) as test set. Joint annotations of BigHand2.2M dataset are shown in Figure 5.3 (right). We manually calculated the hand palm center by taking the mean of the metacarpal joints and the wrist joint. On the other hand, SynHand5M [malik2018deephps] is the largest synthetic hand pose dataset, which contains 5 million depth images with 21 3D joints (see Figure 5.3, left) and 1193 3D hand mesh vertices as ground truth annotations. Its train set ($\mathcal{T}_{\mathcal{S}}$) and test set distributions are 4.5 M and 500 K, respectively.

To study the impacts of individual modules on the accuracy of 3D hand pose estimation task, we compared our **Full** model, which is the complete pipeline (see Figure 5.2), with three baselines. **Baseline 1** directly regresses $\mathcal{J}$ (using Module 1 without the bone-to-joint layer). **Baseline 2** is comprised of complete Module 1 while **Baseline 3** constitutes the first two modules of our pipeline (see Section 5.2.1). We used four error metrics [malik2018deephps] to evaluate the accuracy of the estimated pose and hand mesh: (i) **3D $\mathcal{J}$ Err.**, is the mean 3D joint position error over all test frames; (ii) **3D $\mathcal{B}$ Err.** is the average 3D bone location error; (iii) **3D $\mathcal{V}$ Err.** gives the mean 3D vertex location error; and (iv) the percentage of success frames within thresholds. All error metrics are reported in mm.

5.2.7. Evaluation of 3D Hand Shape Estimation

This subsection gives the experimental details on 3D hand mesh estimation task using SynHand5M [malik2018deephps], NYU [tompson2014real] and BigHand2.2M [yuan2017bighand2] datasets.

Synthetic hand mesh recovery: As SynHand5M [malik2018deephps] is fully-labeled for pose and shape, we trained Baseline 3 and our Full model in a fully-supervised manner using the training strategy explained in Section 5.2.5. Quantitative results are summarized in Table 5.1. Our Baseline 3 (without using 2D depth image synthesizer) outperforms the state-of-the-art DeepHPS method [malik2018deephps]. Our Full model further improves the accuracy of shape estimation over Baseline 3 by 19.6%. Figure 5.6 shows the qualitative results on some challenging hand poses of SynHand5M dataset.

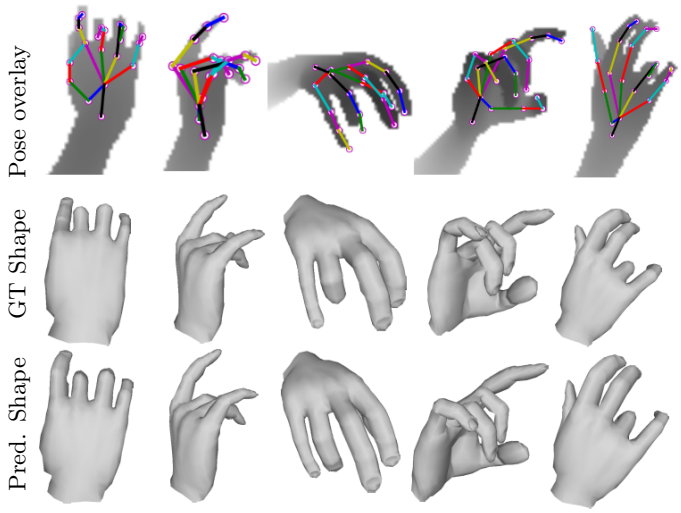


Figure 5.6.: *Synthetic hand pose and shape recovery: 3D shape and pose estimation results on SynHand5M [malik2018deephps] dataset.*

Table 5.1.: *Quantitative results on synthetic [malik2018deephps] dataset. We compared with the state-of-the-art approaches that produce more than joint positions. Notably, our approach outperforms the recent DeepHPS method, which produces 3D hand pose and 3D shape. All errors are reported in mm.*

Method	3D $\mathcal{B}$ Err.	3D $\mathcal{J}$ Err.	3D $\mathcal{V}$ Err.
DeepModel [zhou2016model]	–	11.36	–
HandScales [malik2017simultaneous]	6.5	9.67	–
DeepHPS [malik2018deephps]	5.2	6.3	11.8
Baseline 3 [ours]	4.37	5.24	6.37
Full [ours]	**3.71**	**4.32**	**5.12**

Real hand mesh Recovery: To effectively learn real hand shapes, Module 3 acts as an important source of weak-supervision in training. To recover the hand shapes of NYU dataset, we combined the train sets of SynHand5M and NYU datasets i.e., $\mathcal{T}_{\mathcal{SN}} = \mathcal{T}_{\mathcal{S}} + \mathcal{T}_{\mathcal{N}}$, in one unified format and shuffled them. NYU contains a larger set of joint annotations (i.e., 36 joints) than SynHand5M, therefore we selected 16 closely matching joints that are common to both datasets [malik2018deephps]. Our Full model was end-to-end trained on $\mathcal{T}_{\mathcal{SN}}$ with total loss of the network given by Equation (5.9). The mesh loss of Module 2 was computed by implementing the indicator function (Equation (5.8)). The qualitative results of hand pose and shape recovery on NYU test set are shown in Figure 5.7. Our algorithm successfully reconstructs reasonable hand shapes of complex poses. Clearly, the quality of shape reconstruction depends on the accuracy of the estimated 3D pose.

Examples of synthesized depth images from Module 3 are shown in the Appendix A.4. Similarly, we jointly trained real BigHand2.2M and synthetic

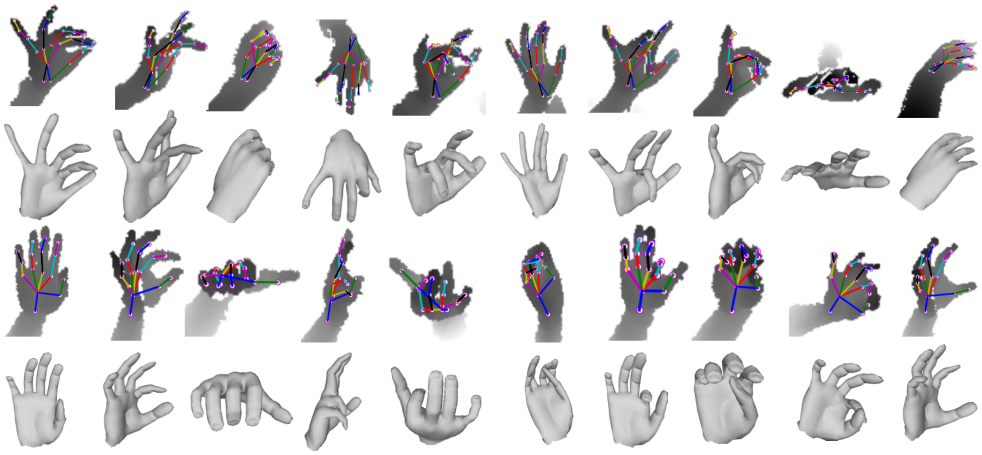


Figure 5.7.: *Real hand pose and shape recovery: Our weakly-supervised algorithm is capable of reconstructing accurate and reasonable hand shapes without using any ground truth of hand shapes of real images. We demonstrate the 3D shape and pose estimation results from our proposed method for two real datasets: BigHand2.2M (top) and NYU (bottom).*

SynHand5M datasets using a mixed train set, i.e., $\mathcal{T}_{\mathcal{BS}} = \mathcal{T}_{\mathcal{B}} + \mathcal{T}_{\mathcal{S}}$. Both datasets have same annotations, as shown in Figure 5.3. Qualitative results of BigHand2.2M shapes recovery are shown in Figure 5.7 and demonstrate successful hand shapes reconstruction even in cases of missing depth information and high occlusions, such as egocentric viewpoint images. More qualitative results from the live stream of depth camera are presented in the Appendix A.7.

For more rigorous evaluation of our approach for real hand shape recovery, we built a new model, which is inspired by the recent work of [ge20193d]. In this model, hand mesh is first estimated using the CNN of Module 1, which directly regresses mesh vertices $\mathcal{V}$ from input depth image $\mathcal{D}_{\mathcal{I}}$, and then a 3D hand pose regressor estimates 3D pose $\mathcal{J}$ from the reconstructed $\mathcal{V}$. Finally, the depth image synthesizer synthesizes the depth image $\mathcal{D}_{\mathcal{R}}$ from $\mathcal{J}$.

For notation simplicity, we call this model as **Model 1** and compared its performance with our Full model on NYU dataset (Table 5.2 shows the pipelines using the notations). Figure 5.8 shows the qualitative comparison on the sample test images of NYU. Hence, the direct hand shape regression using a single depth image is cumbersome, which may lead to highly inaccurate shape estimation. The pipeline of Model 1 is given in the Appendix A.6.

Comparison with the state-of-the-art: To qualitatively compare our recovered real hand shape with the state-of-the-art DeepHPS method [malik2018deephps], we implemented this method and trained it on $\mathcal{T}_{\mathcal{BS}}$. The results on the sample test images of BigHand2.2M dataset are shown in Figure 5.9. Artifacts are clearly visible using DeepHPS method due to fixed linear bases and difficulty in learning complex hand shape and scale parameters in the deep network. In our case, we learn shape from pose, which results

Table 5.2.: *We compared Model 1 with Full model on NYU dataset, which indicates that directly regressing hand mesh from a single depth image is cumbersome and leads to highly inaccurate pose estimation. Mean pose error is in mm.*

Method	Pipeline	3D $\mathcal{J}$ Err.
Full	$\mathcal{D}_{\mathcal{I}} \rightarrow \mathcal{J} \rightarrow \mathcal{V} \rightarrow \mathcal{D}_{\mathcal{R}}$	**10.39**
Model 1	$\mathcal{D}_{\mathcal{I}} \rightarrow \mathcal{V} \rightarrow \mathcal{J} \rightarrow \mathcal{D}_{\mathcal{R}}$	23.63

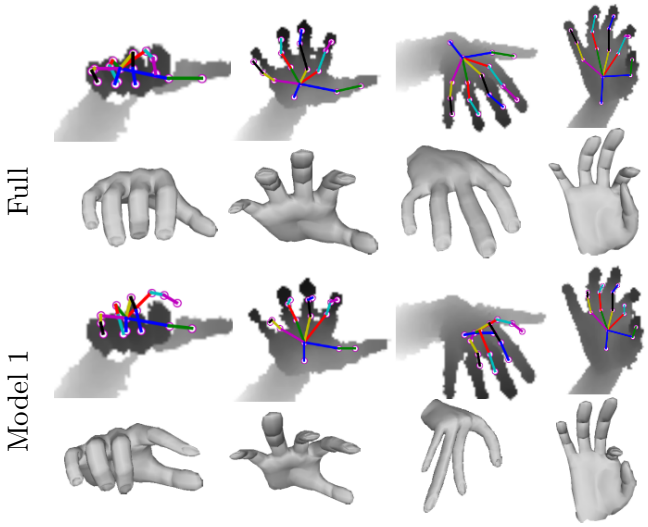


Figure 5.8.: *Qualitative comparisons on 3D shape and pose estimation from Full model and Model 1, which clearly show that regressing pose from estimated shape may result in highly inaccurate shape and consequently adverse pose estimation results.*

in plausible hand shape recovery. We also observed the effect of our Module 3 in training and compared the results of real shape recovery using our Baseline 3. The last column in Figure 5.9 shows the shape estimation results from Baseline 3, i.e., without using the depth synthesizer. The inaccurate mesh reconstruction with Baseline 3 proves that the addition of a weak-supervision from Module 3 is necessary to get reasonable real hand shape reconstruction.

Discussion: Notably, our algorithm learns to reconstruct hand shapes from real depth images by learning from synthetic depth. Therefore, the consistency in depth and joint annotations of real and synthetic images is important to recover the plausible real hand shape and pose. Thus, our approach is unlikely to produce correct and plausible hand shapes for older real hand pose datasets such ICVL [tang2014latent] and MSRA2015 [sun2015cascaded], which are not fully consistent in depth and joint annotations with synthetic SynHand5M [malik2018deephps] dataset.

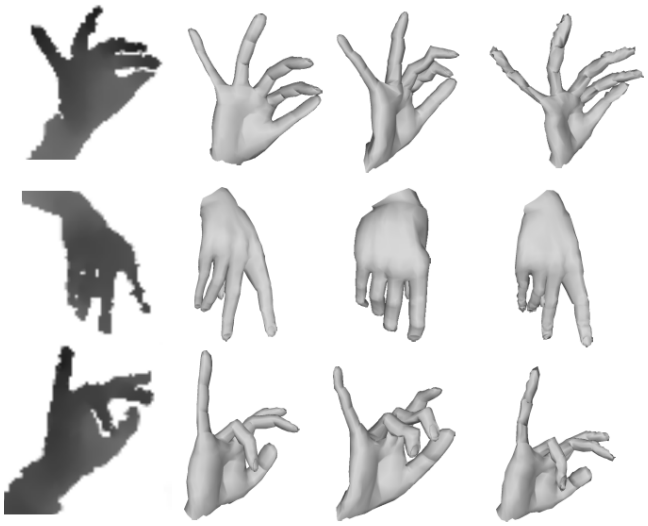


Figure 5.9.: *Real hand shape estimation from different methods on BigHand2.2M [yuan2017bighand2]. Our Full model outperforms LBS-based DeepHPS [malik2018deephps] and Baseline 3 (our method without depth synthesizer).*

Table 5.3.: *Self-comparisons on NYU [tompson2014real] dataset: The effectiveness of different modules of our pipeline. Our Full model shows the effectiveness of jointly fine-tuning the modules altogether. All errors are reported in mm.*

Method	3D $\mathcal{B}$ Err.	3D $\mathcal{J}$ Err.
Baseline 1	–	11.83
Baseline 2	8.40	10.70
Baseline 3	8.24	10.39
Full	**7.80**	**9.24**

5.2.8. Evaluation of 3D Hand Pose Estimation

This subsection provides quantitative and qualitative evaluations of our approach on the task of 3D hand pose estimation. We provide self-comparisons and comparisons to the state-of-the-art methods on NYU and SynHand5M datasets. For the sake of completion, we also provide 3D pose estimation results on BigHand2.2M dataset.

SynHand5M synthetic dataset: We trained our Baseline 3 and Full model on SynHand5M dataset. The quantitative results for joint positions and bone vectors estimations are provided in Table 5.1. Our algorithm outperforms the state-of-the-art methods, which shows the effectiveness of our weak-supervised algorithm and its superior performance compared to the state-of-the-art LBS method [malik2018deephps].

BigHand2.2M real dataset: We evaluated the accuracy of

Table 5.4.: *NYU [tompson2014real] dataset: * methods that produce more than 3D joints positions; + methods that do not respect hand structure and produce only 3D hand pose. WHSP-Net outperforms previous methods that output 3D hand shape and pose, and shows competitive performance to the 3D pose estimation approaches.*

Method	3D $\mathcal{J}$ Err. (mm)
Feedback [oberweger2015training]	15.9
HandPointNet [ge2018point]	10.54
DenseReg [wan2018dense]	10.214
SHPR-Net [chen2018shpr]	10.77
+MURAUER [poier2019murauer]	9.45
+V2V-PoseNet [moon2017v2v]	8.41
+FeatureMapping [rad2017feature]	**7.44**
*DeepModel [zhou2016model]	17.0
*HandScales [malik2017simultaneous]	16.0
*DeepHPS [malik2018deephps]	14.20
*WHSP-Net (Ours)	**9.24**

3D pose estimation on our created test set from BigHand2.2M dataset [yuan2017bighand2]. We trained our Full model on mixed train set $\mathcal{T}_{\mathcal{BS}}$. Qualitative results are shown in Figure 5.7, which demonstrate successful 3D pose recovery of complex hand poses even in cases of missing depth and large occlusions. Quantitatively, the 3D joint error on our created test set (see Section 5.2.6) comes out to be 11.84 mm.

Self-comparisons: To rigorously evaluate our algorithm, we performed self-comparisons of our baseline architectures and Full model on real NYU dataset. The networks were jointly trained with combined NYU, BigHand and synthetic SynHand5M datasets and optimized for the loss given by Equation (5.9). We used the hand model of [zhou2016model] for implementing the bone-to-joint layer. Baseline 1 is similar to the CNN architecture proposed in [wang2018region], which we use to directly regress $\mathcal{J}$. Table 5.3 shows the joints estimation accuracy of Baseline 1. Baseline 2, which incorporates hand skeleton structure (see Section 5.2.2), achieves a 9.6% increase in pose estimation accuracy. Since $\mathcal{L}_{\mathcal{B}}$ is included in Baseline 2, the 3D bone error is also reported in Table 5.3. Baseline 3 includes hand mesh learning, which marginally improves the pose estimation accuracy by 2.8% and bones estimation accuracy by 1.9% over Baseline 2. Our Full model shows the best accuracy on joint positions and bone vectors estimations by including Module 3 in training. Figure 5.10 (left and middle) illustrate quantitative results of the self-comparisons. The curves that cover the most area achieve the highest accuracy. Qualitative comparisons of Baseline 1, Baseline 2 and the Full model are shown in Figure 5.11. Furthermore, we quantitatively evaluated Model 1 (see Section 5.2.7), which shows lower accuracy of 3D pose estimation due to inaccurate hand mesh estimation. We compared its performance to our Full

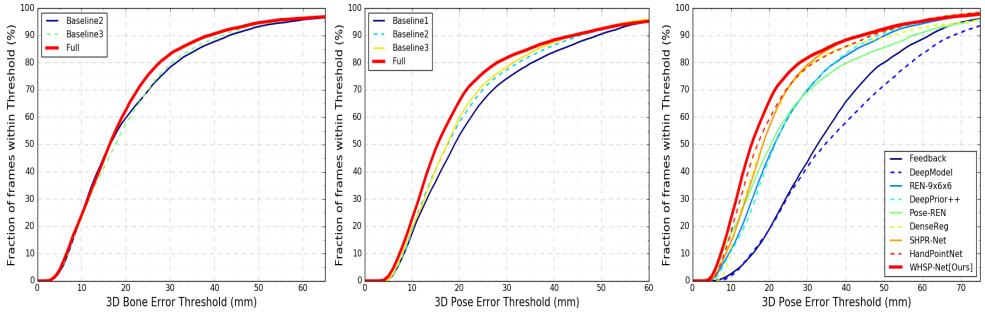


Figure 5.10.: *NYU [tompson2014real] dataset: Quantitative results on 3D pose and bone vectors estimation (mm). The curves show the percentage of success frames within certain threshold values: (left) comparison of the 3D bone vectors estimation accuracy of Full model with two Baselines; (middle) comparison of three Baselines with Full model on joint positions estimation; and (right) comparison of our Full model with the state-of-the-art hand pose estimation methods*

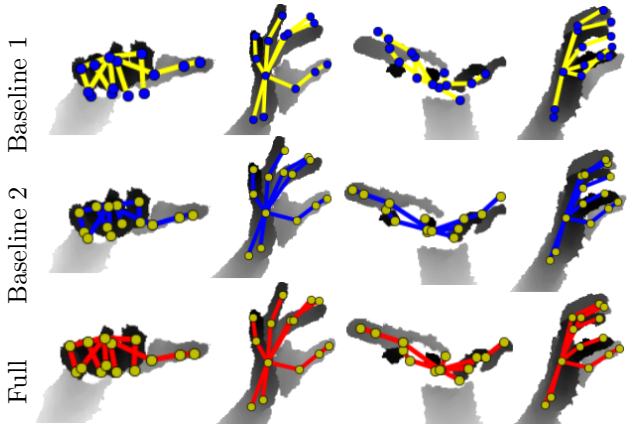
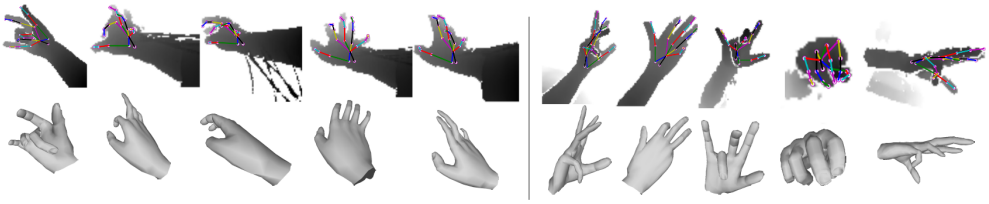


Figure 5.11.: *The 3D pose improvement achieved by our Full model compared with the two Baselines on NYU [tompson2014real].*

model (see Table 5.2).

Comparison with the state-of-the-arts: We compared the 3D hand pose estimation accuracy of our Full model (WHSP-Net) with state-of-the-art approaches. Figure 5.10 (right) and Table 5.4 show the quantitative comparisons. Notably, discriminative methods such as V2V-PoseNet [moon2017v2v] and FeatureMapping [rad2017feature] achieve better accuracy than our method, but they generalize poorly on unseen data [yuan2018depth]. Moreover, V2V-PoseNet is not real-time because of the time consuming gray scale depth input to voxel conversion and the complex 3D-CNN architecture. Furthermore, our method is not discriminative, rather it respects the structure of hand skeleton as well as additionally produces full 3D hand mesh. Therefore, our approach lies in the category of methods that output more than joints.

In addition to the 3D pose, DeepModel [zhou2016model] outputs joint angles; HandScales [malik2017simultaneous] produces joint angles and bone-



(a) **Egocentric** view (b) **3rd person** view

Figure 5.12.: *3D hand shape and pose recovery from a real depth image. We developed a simple and effective real-time convolutional neural network (CNN)-based direct regression approach for simultaneously recovering the 3D hand shape and pose from a single depth image. (a) and (b) show the recovery of the real hand pose and shape based on samples from the BigHand2.2M[yuan2017bighand2] dataset. Feasible real hand shapes can be recovered when no shape annotations are available in the real dataset.*

lengths; and DeepHPS [malik2018deephps] generates joint angles, bone-lengths, complex shape parameters and full 3D hand shape. Our method outperforms these methods, as shown in Table 5.4. Our method shows competitive performance to the state-of-the-art methods that do not explicitly consider the hand structure and produce only the 3D pose [poier2019murauer; moon2017v2v; rad2017feature]. Our algorithm is real-time, producing the 3D pose and shape in $2.9ms$ per frame.

5.3. Simple and Effective Direct Regression Approach

In this section, we explain our second approach based on 2D convolutions for simultaneous hand shape and pose estimation.

5.3.1. Proposed Approach

Figure 5.13 presents an overview of our approach. Given a single view and gray scale depth image $\mathrm{I_D}$, the task involves directly regressing the hand joints $\mathrm{J} \in \mathcal{R}^{3\mathrm{xP}}$, mesh vertices $\mathrm{V} \in \mathcal{R}^{3\mathrm{xN}}$, and structure constraints S, where P is the number of joints and N = 1193 is the number of vertices. As mentioned earlier, direct hand pose regression methods (e.g., [rad2017feature; guo2017region]) may lead to unstable pose estimation because they do not explicitly consider the hand structure in the learning process. In order to maintain the structural relationships between the estimated joints, we follow the method proposed by Malik et al.[malik2018structure] and simultaneously optimize $\mathrm{S} \in \mathcal{R}^{(3\mathrm{xP}-8)}$, including the bone lengths, kinematic distances, and inter-finger distances. The ground truth for S can easily be obtained from J (see [malik2018structure] for details). The loss equations are given by the Euclidean distances as:

$$\mathrm{L_J} = \frac{1}{2}\left\|\mathrm{J} - \mathrm{J}_{GT}\right\|^2 \quad , \quad \mathrm{L_V} = \frac{1}{2}\left\|\mathrm{V} - \mathrm{V}_{GT}\right\|^2 , \qquad \mathrm{L_S} = \frac{1}{2}\left\|\mathrm{S} - \mathrm{S}_{GT}\right\|^2 \tag{5.10}$$

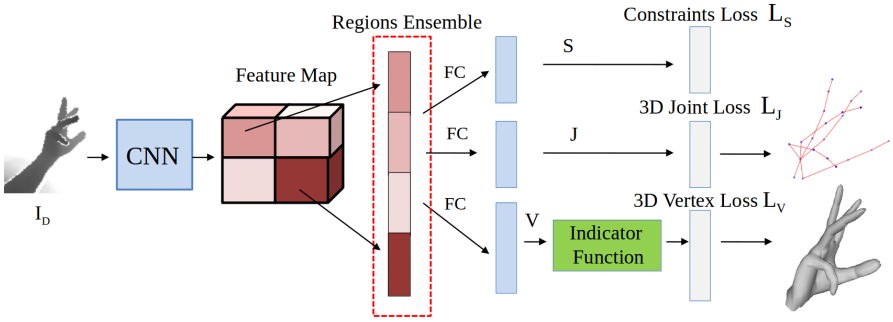


Figure 5.13.: *Overview of our approach for hand shape and pose regression. A depth input (I_D) is given to the convolutional neural network (CNN), which provides a feature map for four distinct regions of I_D. After applying the regions ensemble strategy [guo2017region], the structural constraints S, joint positions J, and mesh vertices V are directly regressed. The indicator function specifies whether the ground truth is available for the vertices or not, which allows the network to be trained using a combination of real and synthetic data. FC: fully connected layer.*

where L_J, L_V, and L_S represent the joint, vertex, and constraint losses, respectively, and J_{GT}, V_{GT}, and S_{GT} are the ground truths for the pose, shape, and constraints. The combined loss equation can be written as:

$$L = L_J + L_S + \mathbb{1}L_V, \tag{5.11}$$

where $\mathbb{1}$ is an indicator function. During the forward pass, $\mathbb{1}$ selects V only for synthetic images using a binary flag value. This value is 1 for synthetic images and 0 for real images. Similarly, back-propagation for V is disabled for real images.

Network Architecture We employ a state-of-the-art CNN [guo2017region] used only for pose regression and modify it to simultaneously regress S, J, and V (see Figure 5.13). A depth input I_D with a size of 96x96 is passed through a shared CNN to produce the feature map measuring 12x12x64. The shared CNN comprises six convolutional layers using a filter size of 3x3. Three max pooling layers with a stride of 2 are used after each pair of convolutional layers. Two residual connections are made between the last two pairs of convolutional layers (for more details of the CNN architecture, see [guo2017region]). The feature map is divided into four regions where each measures 6x6x64. These regions are flattened to produce fully connected (FC) layers where each has a size of 2048. The FC layers are then ensembled using feature concatenation to create a high-dimensional feature vector with a size of 8192. Finally, three lower dimensional regression FC layers are connected to produce S, J, and V separately (as shown in Figure 5.13). V is semi-supervised (see Equation 5.11) so we introduce a layer that implements an indicator function $\mathbb{1}$. This layer forwards only the valid V to the vertex loss layer. Back-propagation is enabled only for synthetic images.

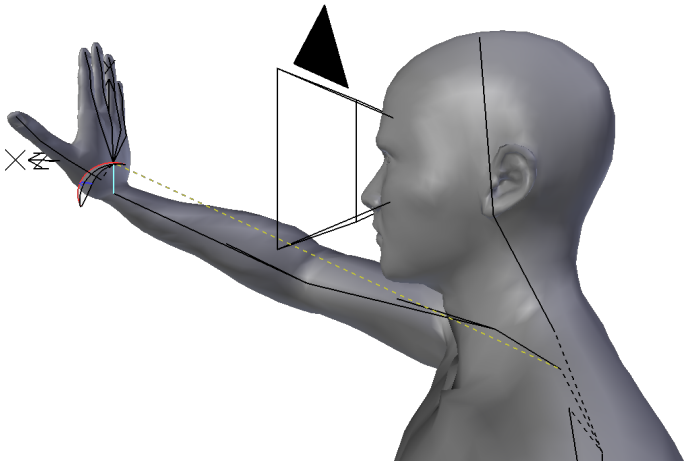


Figure 5.14.: *Setup of the Blender scene used to create the SynHandEgo dataset. The virtual camera was placed between the eyes of the character.*

5.3.2. Egocentric Synthetic Dataset

The SynHandEgo dataset was generated with Blender [blender] using a humanoid created with the MB-Lab add-on [mblab]. A virtual character was created and provided with an inverse kinematics controller for moving and rotating its right hand, wrist, elbow, shoulder, and clavicle. Rotation limits were set for the whole arm, including the fingers, according to realistic ergonomic ranges. A virtual camera that simulated a Senz3D depth sensor [camera˙Senz3D] was mounted between the eyes of the character. Mueller et al. [Mueller2017RealTimeHT] mounted the camera on a shoulder whereas our camera position is optimal for VR/AR applications. The hand was set at the initial position in front of the character (as shown in Figure 5.14). Our custom code routine generated 1 million hand configurations by uniformly sampling random values within the allowed ranges for three categories of degrees of freedom. The first category was hand rotation where the hand was rotated on three axes in ranges that respected ergonomically realistic positions. The second category was finger rotation where all of the fingers were simultaneously rotated within their rotation limits, which allowed the inclusion of uncomfortable or even unrealistic poses in order to provide samples from border-line conditions in the explored space. A collision detection routine discarded the poses where the fingers penetrate each other. Figure 5.16 shows samples of the depth images with overlaid ground truth 3D poses and the respective 3D hand shapes. The third category was hand shapes where the size and proportions of the hand were modulated in the following seven dimensions: *length, mass, size, palm length, inter-finger distance, finger length, and fingertip size.* The realism of the resulting hand proportions was ensured by measuring the hand sizes within the ranges provided in the DINED anthropometry dataset [molenbroek04dined] (see Figure 5.15(b)). Moreover, we provide accurate color segmentations of the hand parts, as shown in Figure 5.15(a), which may be useful for hand part segmentation-based methods such as that proposed by Neverova

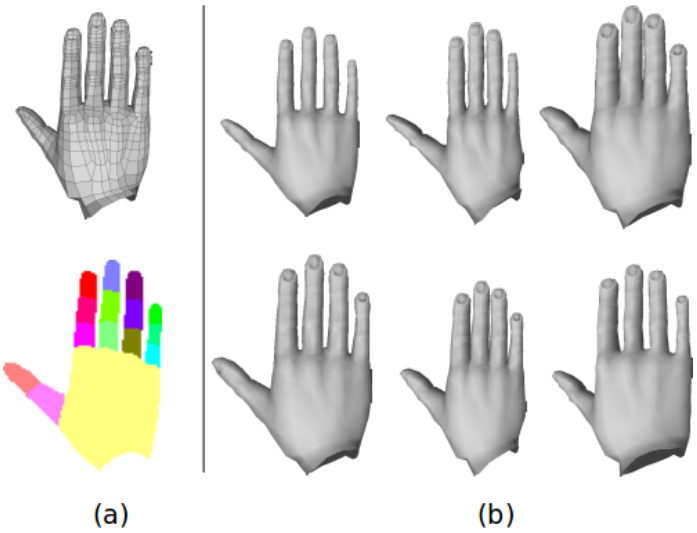


Figure 5.15.: *Proposed egocentric hands dataset. (a) We provide accurate hand segmentation and full 3D hand mesh representations in addition to the 3D pose and depth image. (b) Large hand shape variation. We used realistic hand size measurements provided in the DINED anthropometric database [molenbroek04dined].*

et al. [neverova2017hand]. Segmentation was conducted at the polygon level by manual assignment of the polygon's color to either a phalanx, the palm, or forearm. The colors were generated by sequentially assigning the value of each RGB component to 0.0, 0.5, or 1.0. These colors might seem similar to the human eye but their values are very different in the RGB color space. We divided the dataset into a training set T_E containing 900K images and a test set of 100K frames. As proposed by Malik et al. [malik2018deephps], P and N are the same in SynHand5M.

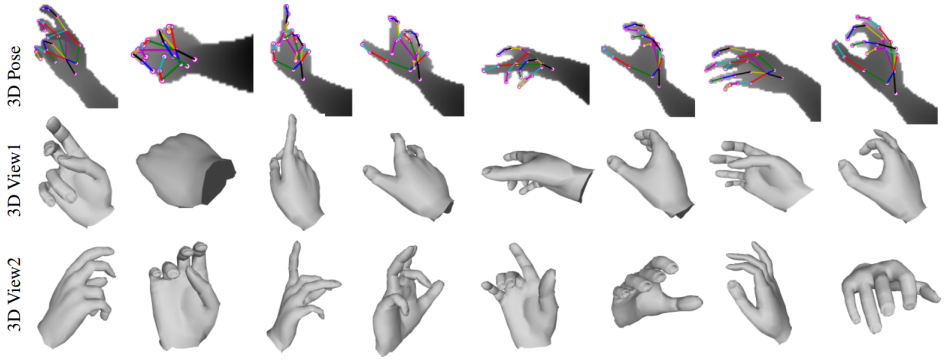


Figure 5.16.: *Sample images from the SynHandEgo egocentric dataset. Preprocessed depth images with overlaid ground truth 3D hand poses and the respective ground truth 3D hand meshes from two different viewpoints. Our dataset includes a wide range of hand poses and shapes.*

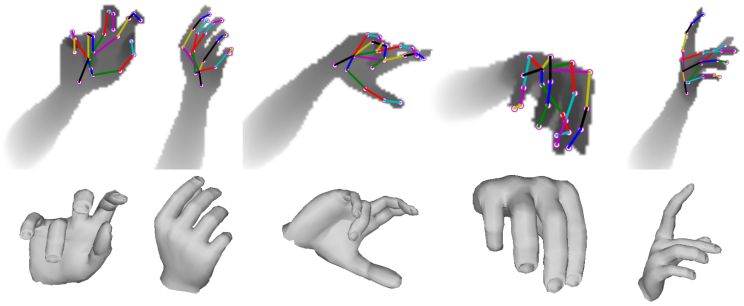


Figure 5.17.: *Qualitative 3D hand pose and shape inference results based on the SynHand5M [malik2018deephps] synthetic dataset.*

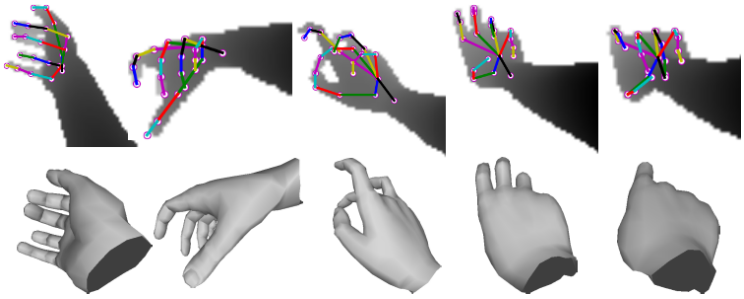


Figure 5.18.: *Qualitative results on SynHandEgo dataset: 3D hand shape and pose recovery from sample egocentric images.*

5.3.3. Implementation Details

We used the method described by Guo et al. [guo2017region] for standard preprocessing of the depth frames and the annotations. All of the images were normalized using the hand mass centers and a bounding box with a fixed size of 150. The final values for the preprocessed depth images and annotations were in the range of $[-1, 1]$. To augment the data, we randomly scaled and rotated the training data in the ranges of $[0.8, 1.1]$ and $[-45°, 45°]$, respectively. We trained our network using the Caffe framework [jia2014caffe]. In order to conduct training based on T_S, the learning rate (LR) was set to 0.0005, SGD momentum to 0.9, and the batch size to 512. For combined real and synthetic data training, LR was set to 0.00005. Training was performed on a desktop PC with an Nvidia Geforce GTX 1070 GPU. A single forward pass required only **2.2ms** to generate both the 3D pose and shape. The networks were trained until they reached convergence.

5.3.4. Synthetic Hand Shape and Pose Recovery

SynHand5M dataset: We trained three different implementations of our network to determine the effectiveness of simultaneously learning the hand shape, pose, and structure constraints. In the first implementation, which is similar to that given by Gue et al. [guo2017region], 22 3D joint key-points from the SynHand5M dataset (i.e., $J \in \mathcal{R}^{66}$) were directly regressed. The network converged after 1000K iterations using LR = 0.05. The quantitative

Method \Error(mm)	3D Joint Loc.	3D Vertex Loc.
DeepModel [zhou2016model]	11.36	–
HandScales [malik2017simultaneous]	9.67	–
DeepHPS [malik2018deephps]	6.3	11.8
J [guo2017region]	5.83	–
J ∪ V [Ours]	5.14	7.12
J ∪ V ∪ S [Ours]	**4.62**	**6.61**

Table 5.5.: *Quantitative results obtained using the SynHand5M test set. The results show that simultaneously learning the pose, shape, and structural constraints improved the accuracy of 3D hand pose estimation by 20.7% compared with the baseline architecture (J) [guo2017region]. All of the errors are reported in millimeters (mm).*

results are shown in Table 5.5. In the second implementation, the shape was optimized together with the pose (J ∪ V) by adding two additional layers to the first implementation: an FC layer with a size of 3579 for regressing the mesh vertices and a non-parametric indicator function layer (as shown in Figure 5.13). This network implementation required 2000K iterations to converge with LR = 0.0005. The estimated pose improved by 11.8% compared with the first implementation. In addition, the shape estimation accuracy improved by 39.6% compared with DeepHPS [malik2018deephps]. In the third implementation, the structural constraints were learned simultaneously with the pose and shape (J∪V∪S), where convergence occurred in 2500K iterations with LR = 0.0005 and the performance was better than the other approaches (Table 5.5). This network implementation contained an additional FC layer with a size of 58 to regress S (see Section 5.3.1). Figure 5.17 shows the qualitative 3D pose and shape estimation results for some challenging hand poses.

SynHandEgo dataset: We trained our network (J ∪ V ∪ S) on T_E with full supervision based on the joint positions, mesh vertices, and structural constraints. The network required 500K iterations to converge with LR = 0.0005. Figure 5.18 shows the qualitative 3D pose and shape recovery results. Quantitatively, the joint and vertex location errors with the test set were 5.5mm and 7mm, respectively.

5.3.5. Real Hand Shape and Pose Recovery

The synthetic data provided weak supervision of the mesh vertices for real hand shape recovery. However, training using both the synthetic and real data allowed our network to learn the shapes and poses of real hands despite the lack of ground truth shape information for real images. We aimed to simultaneously recover both the real hand shape and the pose. However, for the sake of completeness, we conducted comparisons with the state-of-the-art hand pose estimation methods using the NYU dataset.

NYU dataset: We trained our network (J ∪ V ∪ S) based on four datasets, which were combined to form a single training set: T_{NBSE} = T_N ∪ T_B ∪

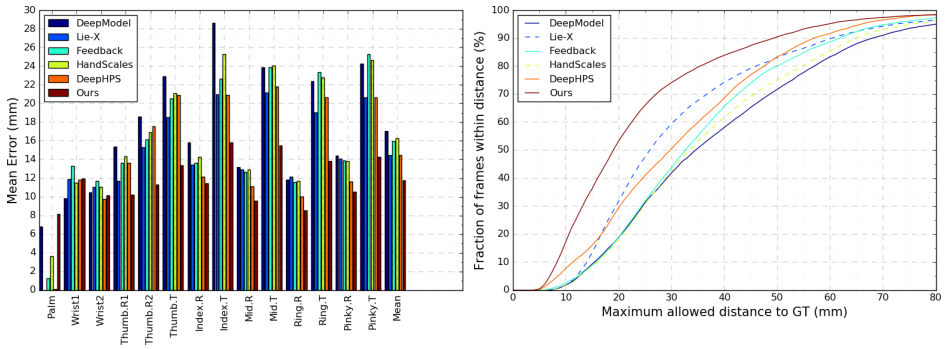


Figure 5.19.: *NYU dataset: mean error (left) and fraction of successful frames (right). Our method achieves improved accuracy compared to the state-of-the-arts that output more than the joint positions.*

Methods	3D Joint Location Error
Crossing Nets [wan2017crossing]	15.5
Feedback [oberweger2015training]	15.9
DeepHPS [malik2018deephps]	14.2
REN-4x6x6 [guo2017region]	13.2
Ours	**11.8**

Table 5.6.: *Quantitative comparison based on the NYU [tompson2014real] test set using several state-of-the-art methods. Our method improved the accuracy compared with the methods that output more than the joint positions. In addition, our method improved the accuracy by 10.6% compared with the baseline method [guo2017region]. All of the errors are reported in mm.*

$T_S \cup T_E$. Not all of the joints in the NYU dataset were consistent with the other datasets, so we followed the method proposed by Malik et al. [malik2018deephps] for selecting the 16 closely matching joints present in all of the datasets. After training based on T_{NBSE} with full supervision for J and S, and semi-supervision for V (see Equation 5.11), we recovered the plausible 3D hand shapes from the NYU dataset. Figure 5.20 shows reconstructions of the 3D shapes obtained from the sample test depth images in the NYU dataset. The network required 5000K iterations to reach convergence using LR = 0.00005. We qualitatively compared our reconstructed 3D hand shapes with those obtained using the state-of-the-art DeepHPS method [malik2018deephps]. DeepHPS is hindered by the generation of artifacts during shape reconstruction because of the limited representational capacity of the hand model as well as difficulties optimizing complex hand shapes, bone scales, and joint angle parameters inside the deep network. Moreover, all of these parameters were implicitly learned. To compare the performance of our method in the 3D pose estimation task, we trained our network using a subset of 14 joints from the NYU dataset and the corresponding closely matching joints in other datasets. Figure 5.19 shows quantitative comparisons with several state-of-the-art methods. Figure 5.19 (left) shows the errors based on individual joints from the NYU dataset and the mean error. Figure 5.19 (right)

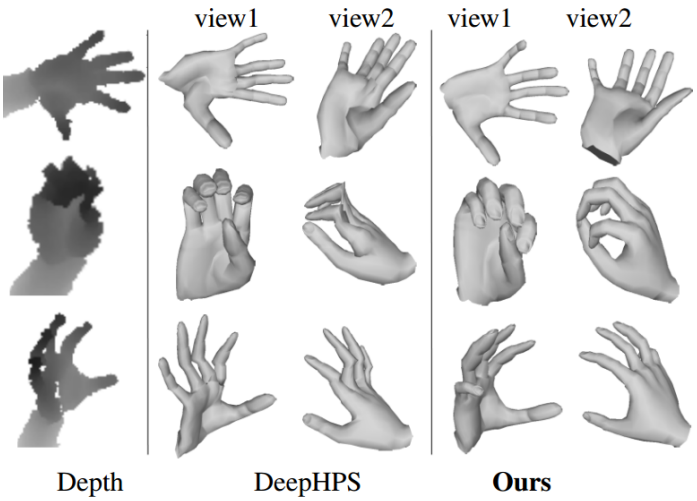


Figure 5.20.: *Qualitative comparison of the 3D hand shapes recovered with the state-of-the-art DeepHPS method [malik2018deephps] using the NYU [tompson2014real] dataset. The DeepHPS method produced artifacts due to the limited representational capacity of their hand model as well as difficulties optimizing complex hand shapes, bone scales, and joint angle parameters. By contrast, our algorithm recovered more accurate real hand shapes.*

shows the fraction of successful frames within various thresholds (in *mm*). The joint location errors over all of the test frames are presented in Table 5.6. As mentioned earlier, DeepHPS [malik2018deephps] is the only existing method that estimates both 3D the hand shape and pose with the NYU dataset. Our approach performed significantly better than the DeepHPS method in the pose estimation task, thereby demonstrating the benefit of directly regressing the dense mesh together with the sparse joints. In addition, our approach improved the accuracy by 10.6% compared with the baseline REN architecture [guo2017region], which only regresses the joint positions. Thus, our method significantly improved the accuracy of hand pose estimation compared with the state-of-the-art methods that produce more than joint positions, i.e., DeepModel [zhou2016model], HandScales [malik2017simultaneous], Feedback [oberweger2015training], and DeepHPS [malik2018deephps]. In addition to 3D pose estimation, DeepModel [zhou2016model] estimates the joint angle parameters, HandScales [malik2017simultaneous] predicts the joint angles and bone lengths in the hand skeleton, Feedback [oberweger2015training] synthesizes 2D depth images, and the DeepHPS [malik2018deephps] method estimates the joint angles, bone lengths, complex hand shape parameters, and 3D hand mesh vertices. We did not conduct comparisons with direct regression methods [moon2017v2v; rad2017feature] that do not incorporate hand structure in their pipelines and that only produce 3D hand poses.

BigHand2.2M dataset: We combined the BigHand2.2M, SynHand5M, and SynHandEgo datasets into one training set: $T_{BSE} = T_B \cup T_S \cup T_E$. The 21 joints in the BigHand2.2M dataset were consistent with the joints in both of

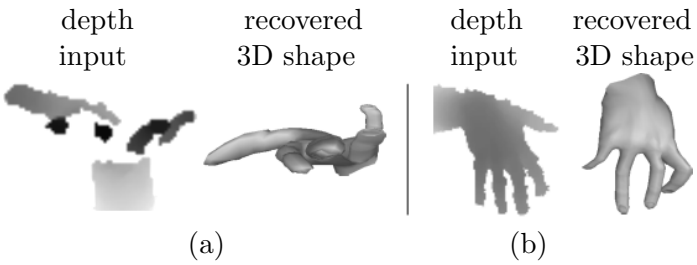


Figure 5.21.: *Failure case: Our method failed to recover correct hand shapes from real depth images when large amounts of depth information were missing (a), or when the hand pose in the real depth image was not covered by the pose space in the synthetic dataset, which was a rare occurrence (b).*

the synthetic datasets. However, the palm center positions were missing from the BigHand5M dataset. Thus, we calculated the palm centers by taking the mean of the centers of the metacarpal joints and the wrist joint. Hence, all 22 joints were used in the combined training process. After simultaneously training with the real and synthetic data, we recovered 3D hand shapes for the challenging poses in BigHand2.2M from both egocentric and third person viewpoints (as shown in Figure 5.12). The network converged within 5000K iterations using LR = 0.00005. Quantitatively, the 3D joint location error with the test set was 13.5mm.

Failure Cases: When significant depth information was missing from the real images, our algorithm failed to recover the plausible hand shapes (as shown in Figure 5.21(a)). In addition, if the hand pose in the real depth image differed significantly from the pose space covered by the synthetic dataset, which occurred rarely, our network could not recover the correct hand shape (see Figure 5.21(b)).

5.4. Conclusion

In this chapter, we present two different approaches based on 2D convolutions for simultaneous 3D hands shape and pose estimation from a single depth image. The first is a novel weakly-supervised method which consists of three novel components: (i) Structured 3D hand pose estimator; (ii) 3D hand shape decoder; and (iii) 2D depth image synthesizer. The hand shape decoder learns to recover 3D hand mesh representation from the estimated 3D pose. To provide a much needed weak-supervision on shape estimation, we propose a new depth image synthesizer which reconstructs 2D depth image from learned 3D hand mesh. Our method is jointly fine-tuned in an end-to-end manner by using unlabeled real data and labeled synthetic data. Extensive evaluations show plausible and reasonable hand shapes reconstruction in real-time despite an unavailability of ground truth for real hand shapes. The proposed approach outperforms state-of-the-art methods that produce more than joint positions and shows competitive results compared to 3D pose estimation methods.

The second is a simple and effective real-time CNN-based approach for directly regressing the 3D hand shape and pose for both egocentric and third person viewpoints by learning from synthetic depth. We also create the first egocentric synthetic hand pose dataset, which provides accurate annotations for 3D hand shapes and poses. In addition, we provide color segmentation of the hand parts. This dataset will facilitate future research into full hand shape and pose estimation from egocentric viewpoints, provided that obtaining the real hand shape ground truth is a hard and sub-optimal problem. Our network is trained simultaneously using real and synthetic data, which allows the successful recovery of plausible real hand shapes. Learning the pose and the structural constraints is fully supervised, whereas the shape learning is semi-supervised. Experiments show that our approach performs better than the state-of-the-art methods with the synthetic SynHand5M dataset in terms of both hand shape and pose estimation tasks, and it also improves the pose estimation accuracy based on the real NYU dataset compared with the existing methods that output more than the joint positions.

Chapter 6

Hand Shape and Pose Estimation using 3D Convolutional Networks

Contents

6.1. Introduction

Current approaches treat and process depth maps with 2D CNNs, even though depth maps are intrinsically a 3D data. Training a 2D CNN to estimate 3D hand pose or shape given 2D representation of a depth map is highly non-linear and results in perspective distortions in the estimated outputs [moon2017v2v]. V2V-PoseNet [moon2017v2v] is the first work that uses 3D voxelized grid of depth map to estimate 3D joints heatmaps and, thus, avoids perspective distortions. However, extending this work for shape estimation by directly regressing 3D heatmaps of mesh vertices is not feasible in practice.

In this work, we propose the first 3D convolutions based architecture which simultaneously estimates 3D shape and 3D pose given a voxelized depth map. To this end, we introduce novel architectures based on 3D convolutions which

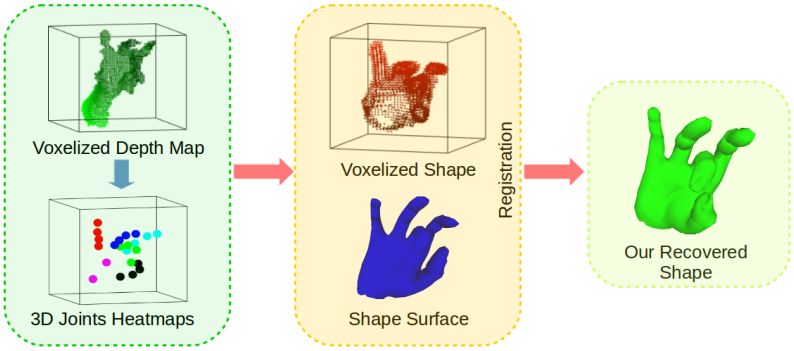


Figure 6.1.: *3D voxelized depth map and accurately estimated 3D joints heatmaps (Left block) are used to estimate two hand shape representations (Middle block). To combine the advantages of these representations, we accurately register the shape surface to the voxelized shape (Right block). Our architecture with 3D convolutions establishes a one-to-one mapping between voxelized depth map, voxelized hand shape and 3D joints heatmaps.*

estimate two different representations of hand shape (Secs. 6.2–6.4). The first representation is the hand shape on a voxelized grid. This representation is estimated from a new *voxel-to-voxel* network which establishes a one-to-one mapping between the voxelized depth map and the voxelized shape. However, the estimated voxelized shape does not preserve the hand mesh topology and the number of vertices. For this reason, we also estimate hand surface (the second representation) with our *voxel-to-surface* network. Since this network does not establish the one-to-one mapping, the accuracy of the estimated hand surface is low. However, the hand topology is preserved. In order to combine the advantages of both representations, we propose registration methods to fit the hand surface to the voxelized hand shape. Since real hand shape annotations are not available, we employ two 3D CNN-based synthesizers which act as sources of weak supervision by generating voxelized depth maps from our shape representations (see Figure 6.2). To increase the robustness and accuracy of the hand pose estimation, we perform 3D data augmentation on the voxelized depth maps (Sec. 6.3.2).

We conduct ablation studies and perform extensive evaluations of our proposed method on real and synthetic datasets. Our approach improves the accuracy of hand shape estimation on SynHand5M dataset by 47.8% and outperforms the state-of-the-art. Our method produces visually more reasonable and plausible hand shapes of NYU and BigHand2.2M datasets compared to the state-of-the-art approaches (Sec. 6.5).

6.2. Method Overview

Given a single input depth image, our goal is to estimate N 3D hand joint locations $\mathcal{J} \in \mathcal{R}^{3\times N}$ (i.e. 3D pose) and K = 1193 3D vertex locations $\mathcal{V} \in \mathcal{R}^{3\times K}$ (i.e. 3D shape). Figure 6.2 shows an overview of the proposed approach. The input depth image is converted into a voxelized grid (i.e. V_D) of size $88 \times 88 \times 88$, by

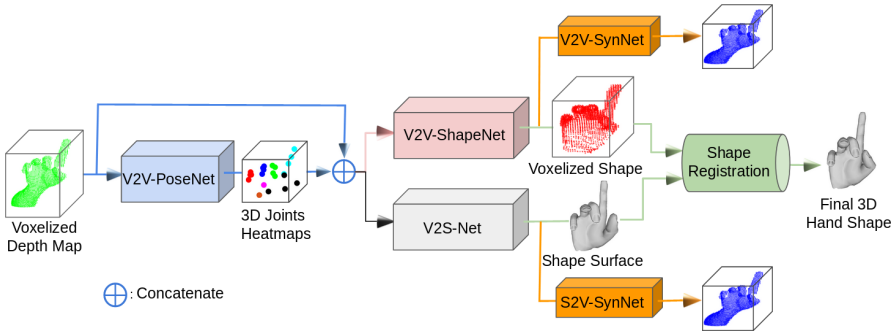


Figure 6.2.: *Overview of our approach for 3D hand shape and pose recovery given a voxelized depth map. V2V-PoseNet estimates 3D joints heatmaps (i.e. pose). Hand shape is obtained in two phases. First, V2V-ShapeNet and V2S-Net estimate the voxelized shape and shape surface, respectively. V2V-SynNet and S2V-SynNet synthesize the voxelized depth thereby, act as sources of weak-supervision. They are excluded during testing. In the second phase, shape registration accurately fits the shape surface to the voxelized shape.*

using intrinsic camera parameters and a fixed cube size. For hand pose estimation, $\mathrm{V_D}$ is provided as an input to the *voxel-to-voxel* pose regression network (i.e. V2V-PoseNet) that directly estimates 3D joint heatmaps $\{\mathcal{H}_j\}_{j=1}^{\mathrm{N}}$. Each 3D joint heatmap is represented as $44 \times 44 \times 44$ voxelized grid. We resize $\mathrm{V_D}$ to $44 \times 44 \times 44$ voxel grid size (i.e. $\mathrm{V'_D}$) and concatenate it with the estimated $\mathcal{H}_j$, to provide as an input to our shape estimation network. We call this concatenated input as $\mathcal{I}_S$. The voxelized hand shape (i.e. $64 \times 64 \times 64$ grid size) is directly regressed via 3D CNN-based *voxel-to-voxel* shape regression network (i.e. V2V-ShapeNet), by using $\mathcal{I}_S$ as an input. Notably, V2V-ShapeNet establishes a one-to-one mapping between the voxelized depth map and the voxelized shape. Therefore, it produces accurate voxelized shape representation but does not preserve the topology of hand mesh and the number of mesh vertices. To regress hand surface, $\mathcal{I}_S$ is fed to the 3D CNN-based *voxel-to-surface* regression network (i.e. V2S-Net). Since the mapping between $\mathcal{I}_S$ and hand surface is not one-to-one, it is therefore less accurate. *Voxel-to-voxel* and *surface-to-voxel* synthesizers (i.e. V2V-SynNet and S2V-SynNet) are connected after V2V-ShapeNet and V2S-Net, respectively. These synthesizers reconstruct $\mathrm{V'_D}$ and act as sources of weak supervision during training. They are excluded during testing. To combine the advantages of the two shape representations, we register the estimated hand surface to the estimated voxelized hand shape. We employ 3D CNN-based DispVoxNet network [ShimadaDispVoxNets2019] for synthetic data, and non-rigid gravitational approach (NRGA) [Ali˙NRGA˙2018] for real data.

6.3. The Proposed HandVoxNet Approach

In this section, we explain our proposed HandVoxNet approach by highlighting the function and effectiveness of each its component. We develop an effective solution that produces reasonable hand shapes via 3D CNN-based deep net-

works. To this end, our approach fully exploits accurately estimated 3D joints heatmaps as a strong pose prior, as well as voxelized depth maps. Given that collecting accurate real hand shape ground truth is hard and laborious, we develop a weakly-supervised network for real hand shape estimation by learning from accurately labeled synthetic data. Moreover, our 3D data augmentation on voxelized depth maps allows to further improve the accuracy and robustness of 3D hand pose estimation.

6.3.1. 3D Hand Shape Estimation

As aforementioned, estimating 3D hand shape from a 2D depth map by using 2D CNN is a highly non-linear mapping. It compels the network to perform perspective distortion-invariant estimation which causes difficulty in learning the shapes. To address this limitation, we develop a full voxel-based deep network that effectively utilizes the estimated 3D pose and voxelized depth map to produce reasonable 3D hand shapes. Our proposed approach for 3D shape estimation comprises of two main phases. In the first phase, we estimate the shape surface and the voxelized hand shape. In the second phase, we register the estimated shape surface to the estimated voxelized hand shape by employing a 3D CNN-based registration for synthetic data and NRGA-based fitting process for real data.

Voxelized Shape Estimation. Our idea is to estimate 3D hand shape in the voxelized form via 3D CNN-based network. It allows the network to estimate the shape in such a way that minimizes the chances for perspective distortion. Inspired by the approach proposed in the recent work [malik2019whsp], we consider sparse 3D pose as the latent representation of dense 3D shape. However, in this work, we combine 3D pose with the depth map which is important to better represent the shape of hand. Furthermore, here we use more accurate and useful representations of 3D pose and 2D depth image which are 3D joints heatmaps and voxelized depth map, respectively. V2V-ShapeNet module is shown in Figure 6.2. It can be considered as the 3D shape decoder:

$$\hat{\mathcal{V}}_S \sim \mathrm{Dec}(\mathcal{H}_j \oplus \mathrm{V}'_{\mathrm{D}}) = p(\mathcal{V}_S|\mathcal{I}_S) \tag{6.1}$$

where $p(\mathcal{V}_S|\mathcal{I}_S)$ is the decoded distribution. The decoder learns to reconstruct the voxelized hand shape $\hat{\mathcal{V}}_S$ as close as possible to the ground truth voxelized hand shape $\mathcal{V}_S$. The V2V-ShapeNet is a 3D CNN-based architecture that directly estimates the probability of each voxel in the voxelized shape indicating whether it is the background (i.e. 0) or the shape voxel (i.e. 1). The per-voxel binary cross entropy loss $\mathcal{L}_{\mathcal{V}_S}$ for voxelized shape reconstruction is given as:

$$\mathcal{L}_{\mathcal{V}_S} = -(\mathcal{V}_S \, log(\hat{\mathcal{V}}_S) + (1 - \mathcal{V}_S) \, log(1 - \hat{\mathcal{V}}_S)) \tag{6.2}$$

where $\mathcal{V}_S$ and $\hat{\mathcal{V}}_S$ are the ground truth and the estimated voxelized hand shapes, respectively. The architecture of V2V-ShapeNet is provided in the the Appendix A.8.1.

Since the annotations for real hand shapes are not available, weak-supervision is therefore essential in order to effectively learn real hand shapes.

For this reason, we propose a 3D CNN-based V2V-SynNet (see Figure 6.2) which acts as a source of weak-supervision during training. This module is removed during testing. V2V-SynNet synthesizes the voxelized depth map from the estimated voxelized shape representation. The per-voxel binary cross entropy loss $\mathcal{L}_{\mathrm{V_D}}$ for voxelized depth map reconstruction is given as:

$$\mathcal{L}_{\mathrm{V_D}} = -(\mathrm{V_D}\ log(\hat{\mathrm{V}}_\mathrm{D}) + (1 - \mathrm{V_D})\ log(1 - log(\hat{\mathrm{V}}_\mathrm{D})) \tag{6.3}$$

where $\mathrm{V_D}$ and $\hat{\mathrm{V}}_\mathrm{D}$ are the ground truth and the reconstructed voxelized depth maps, respectively. The architecture of V2V-SynNet is provided in the Appendix A.8.3.

Shape Surface Estimation. The hand pose of both the shape surface and voxelized shape representations needs to be similar for an improved shape registration. In order to facilitate the registration, we employ V2S-Net deep network which directly regresses $\mathcal{V}$. Based on the similar concept of hand shape decoding (as mentioned before), $\mathcal{I}_S$ is provided as an input to this network while the decoded output is the reconstructed hand mesh (see Figure 6.2). The hand shape surface reconstruction loss $\mathcal{L}_{\mathcal{V}_T}$ is given by the standard Euclidean loss as:

$$\mathcal{L}_{\mathcal{V}_T} = \frac{1}{2} \left\| \hat{\mathcal{V}}_T - \mathcal{V}_T \right\|^2 \tag{6.4}$$

where $\mathcal{V}_T$ and $\hat{\mathcal{V}}_T$ are the respective ground truth and reconstructed hand shape surfaces. As explained before, in case of missing real hand shape ground truth, the weak-supervision on mesh vertices is provided by S2V-SynNet. In this case, the input to the S2V-SynNet is $\hat{\mathcal{V}}_T$ which is in 3D coordinates form. The loss function for the S2V-SynNet is similar to Eq. (6.3). Further details of S2V-SynNet and V2S-Net can be found in the Appendix A.8.2 and A.8.3.

CNN-based Shape Registration. Thanks to fully connected (FC) layers, V2S-Net is able to estimate hand shapes while preserving the order and number of points of the shapes. Losing local spacial information is also known as a drawback of FC layers. In contrast to FC layers, a lot of works show fully convolutional networks (FCN) perform well in geometry regression tasks [golyanik2018hdm; moon2017v2v; shimada2019ismo; wu2016learning]. However, estimating the voxelized hand shape by 3D convolutional layer results in an inconsistent number of points and loses a property of point order.

Hence, the ideal architecture is a network which estimates the hand shape without losing local spacial information while preserving the topology of the hand shape. To achieve this, we register the estimated shape by V2S-Net to the probabilistic shape representation estimated by FCN (V2V-ShapeNet) using DispVoxNets pipeline [ShimadaDispVoxNets2019].

Original DispVoxNets pipeline is comprised of two stages, i.e. global displacement estimation and refinement stage. The refinement stage is used to remove roughness on the point set surface. In contrast to the original approach, we replace the refinement stage with Laplacian Smoothing [vollmer1999improved]. This is possible because we assume the mesh topology is already known, and it is preserved by our pipeline.

In the DispVoxNet pipeline, the hand surface shape $\hat{\mathcal{V}}_T$ is first converted into a voxelized grid $\hat{\mathcal{V}}'_T$ (i.e. $64\times64\times64$ voxelized grid size). DispVoxNet estimates per-voxel displacements of the dimension $64^3\times3$ between the reference $\hat{\mathcal{V}}_S$ and voxelized hand surface $\hat{\mathcal{V}'}_T$. The displacement loss $\mathcal{L}_{\text{Disp}}$ is given as:

$$\mathcal{L}_{Disp.} = \frac{1}{Q^3}\left\|\mathbf{d} - D_{vn}(\hat{\mathcal{V}}_S, \hat{\mathcal{V}}'_T)\right\|^2, \tag{6.5}$$

where Q and $\mathbf{d}$ are the voxelized grid size and the ground truth displacement, respectively. Since it is difficult to obtain $\mathbf{d}$ between voxelized shape $\hat{\mathcal{V}}_S$ and hand surface $\hat{\mathcal{V}}_T$, the displacements are first computed between $\mathcal{V}_T$ and $\hat{\mathcal{V}}_T$, and are discretised to obtain $\mathbf{d}$. For more details of ground truth voxelized grid computation, please refer to [ShimadaDispVoxNets2019]. The architectural details of DispVoxNet are provided in the Appendix A.8.4.

NRGA-Based Shape Registration. In our voxel-based 3D hand shape and pose estimation pipeline (Fig. 6.2), DispVoxNet [ShimadaDispVoxNets2019] component requires shape annotations in its source-to-target displacement field learning phase. These annotations are available only for synthetic data set which leaves a domain gap on the performance of DispVoxNet when tested on real data set. To bridge this gap, we apply NRGA [Ali˙NRGA˙2018] to improve $\hat{\mathcal{V}}_T$ by registering it with $\hat{\mathcal{V}}_S$. NRGA is selected for this deformable alignment task over other methods [amberg2007optimal; papazov2011deformable], as it supports local-topology preservation of input hand surface and shown to be more robust at noise handling. Although NRGA is a point cloud alignment method, it provides an option to relax deformation magnitude in the neighbouring regions of the hand mesh vertices. To justify our *surface-to-voxel* ($\hat{\mathcal{V}}_T \longrightarrow \hat{\mathcal{V}}_S$) alignment method, NRGA is applied with a modification (**M1**) on its nearest neighbour scheme. The method estimates a rigid transformation for every vertex $v \in \hat{\mathcal{V}}_T$ and diffuses the transformation in a subspace formed by a set of neighbourhood vertices of v. NRGA builds a k-d tree on template, which is $\hat{\mathcal{V}}_T$ in our case, and neighbourhood vertices are selected as the k-nearest neighbours (typically $0.1\% - 0.2\%$ of the total points in template).
M1: *To apply NRGA in our pipeline, we use 4-ring connected vertices instead of k-nearest neighbours* (more details are in the Appendix A.9).

6.3.2. Data Augmentation in 3D

Our method for hand shape estimation relies on the accuracy of the estimated 3D pose. Therefore, the hand pose estimation method has to be accurate and robust. Training data augmentation helps to improve the performance of a deep network [oberweger2017deepprior++]. Existing methods of hand pose estimation have used data augmentation in 2D. This is mainly because these methods have treated the depth map as 2D data. The representation of depth map in voxelized form makes it convenient to perform data augmentation in all three dimensions. In this paper, we propose a new 3D data augmentation policy which improves the accuracy and robustness of hand pose estimation (see Section 6.5.3).

During V2V-PoseNet training, we apply simultaneous rotations in all three axes (x,y,z) to each 3D coordinate (i, j, k) of $\mathrm{V_D}$ and $\mathcal{H}_j$, by using the following Euler transformations:

$$[\hat{i}, \hat{j}, \hat{k}]^{\mathrm{T}} = [\mathrm{Rot_x}(\theta_{\mathrm{x}})] \times [\mathrm{Rot_y}(\theta_{\mathrm{y}})] \times [\mathrm{Rot_z}(\theta_{\mathrm{z}})][i, j, k]^{\mathrm{T}} \tag{6.6}$$

where $(\hat{i}, \hat{j}, \hat{k})$ is the transformed voxel coordinate. $\mathrm{Rot_x}(\theta_{\mathrm{x}})$, $\mathrm{Rot_y}(\theta_{\mathrm{y}})$ and $\mathrm{Rot_z}(\theta_{\mathrm{z}})$ are 3×3 rotation matrices around x, y and z axes. The values for θ_{x}, θ_{y} and θ_{z} are selected randomly in the ranges $[-40°, +40°]$, $[-40°, +40°]$ and $[-120°, +120°]$, respectively. In addition to rotations in 3D, following [moon2017v2v], we perform scaling and translation in the respective ranges $[+0.8, +1.2]$ and $[-8, +8]$.

6.4. The Network Training

$\mathrm{V_D}$ is generated by projecting the raw depth image pixels into 3D space. Hand region points are then extracted by using a cube of size 300 that is centered on hand palm center position. 3D point coordinates of hand region are discretized in range [1, 88]. Finally, to obtain $\mathrm{V_D}$, the voxel value is set to 1 for the 3D point coordinate of hand region and 0 otherwise. Following [moon2017v2v], $\mathcal{H}_j$ are generated as 3D Gaussians. Similar to the process of generating $\mathrm{V_D}$, $\mathcal{V}_S$ is obtained by voxelizing the hand mesh. $\mathcal{V}_T$ is created by normalizing the mesh vertices in range [−1, +1]. We perform this normalization by subtracting the vertices from the palm center and then, dividing them by half of the cube size.

We train V2V-PoseNet [moon2017v2v] on NYU, BigHand2.2M and SynHand5M datasets separately with the 3D data augmentation technique mentioned in Section 6.3.2. For SynHand5M dataset, we train V2S-Net and V2V-ShapeNet (including the synthesizers S2V-SynNet and V2V-SynNet) separately using RMSProp as an optimization method with a batch size of 8 and a learning rate $\mathrm{LR} = 2.5 \times 10^{-4}$. After training the pose and shape networks, we put these networks together in the pipeline (see Figure 6.2) and fine-tune them in an end-to-end manner with synthetic, and combined real and synthetic data. The total loss $\mathcal{L}_{\mathrm{T}}$ is as follows:

$$\mathcal{L}_{\mathrm{T}} = \mathcal{L}_{\mathcal{H}} + \mathbb{1}\mathcal{L}_{\mathcal{V}_S} + \mathbb{1}\mathcal{L}_{\mathcal{V}_T} + \mathcal{L}_{\mathrm{V_D}} \tag{6.7}$$

where $\mathcal{L}_{\mathcal{H}}$ is heatmaps loss [moon2017v2v] and $\mathbb{1}$ represents an indicator function layer. This layer forwards the estimations to the loss layer only for synthetic data using a flag value, which is 1 for synthetic and 0 for real data. It disables the gradients flow during the backward pass in case of real data. For fine-tunings, we use RMSProp optimization method with a batch size of 6 and a learning rate 2.5×10^{-5}. DispVoxNet is trained only on SynHand5M dataset due to its availability of the ground truth geometry. During the training, Adam optimizer with a learning rate of 3.0×10^{-4} was employed. The training continues until the convergence of $\mathcal{L}_{\mathrm{Disp}}$ with batch size 12. All models are trained till convergence on a desktop personal computer (PC) equipped with Nvidia Titan X GPU.

Methods	3D $\mathcal{V}$ Err. (mm)
V2S-Net (w/o $\mathcal{H}_j$)	8.78
V2S-Net (w/o V'_{D})	3.54
V2S-Net (w/ $\mathcal{H}_j \oplus \mathrm{V}'_{\mathrm{D}}$)	**3.36**
Methods	3D $\mathcal{S}$ Err.
V2V-ShapeNet (w/o $\mathcal{H}_j$)	0.007
V2V-ShapeNet (w/o V'_{D})	0.016
V2V-ShapeNet (w/ $\mathcal{H}_j \oplus \mathrm{V}'_{\mathrm{D}}$)	**0.005**

Table 6.1.: *Ablation study on inputs (i.e. $\mathcal{H}_j$ and V'_{D}) to V2S-Net and V2V-ShapeNet. We observe that combining both inputs is useful for these two networks. w/ and w/o refer to with and without.*

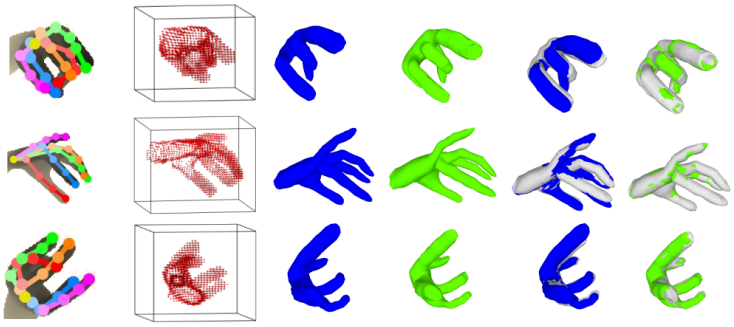


Figure 6.3.: *Qualitative results on SynHand5M [malik2018deephps] dataset. Estimated hand pose overlay (1^{st} col), voxelized shape (2^{nd} col), hand surface (3^{rd} col), final shape (4^{th} col), and the overlays of hand surface and final shapes with ground truth (gray color) are illustrated.*

6.5. Experiments

We perform qualitative and quantitative evaluations of our complete pipeline on the fully labeled SynHand5M [malik2018deephps] dataset. Ablation studies are performed on the same dataset. We qualitatively evaluated real hand shape recovery for NYU [tompson2014real] and BigHand2.2M [yuan2017bighand2] datasets. Furthermore, we study the impact of our 3D data augmentation on powerful V2V-PoseNet [moon2017v2v].

6.5.1. Datasets and Evaluation Metrics

Although there are many depth-based hand pose datasets [yuan2017bighand2], only a few of them (i.e. BigHand2.2M [yuan2017bighand2], NYU [tompson2014real], SynHand5M [malik2018deephps]) provide adequate training data and annotation which resemble the joint locations of a real hand. NYU real benchmark offers joint annotations for 72757 and 8252 RGBD images of the training ($\mathcal{T}_{\mathrm{N}}$) and test sets, respectively. Their hand model contains 36 joint locations which make it possible to combine this dataset with the recent

Methods	3D $\mathcal{V}$ Err. (mm)
DeepHPS [malik2018deephps]	11.8
WHSP-Net [malik2019whsp]	5.12
Ours (w/o Synthesizers)	2.92
Ours (w/ Synthesizers)	**2.67**

Table 6.2.: *Comparison with the state-of-the-arts on SynHand5M. Our full method, with V2V-SynNet and S2V-SynNet synthesizers, outperforms the state-of-the-art method [malik2019whsp] by 47.85%.*

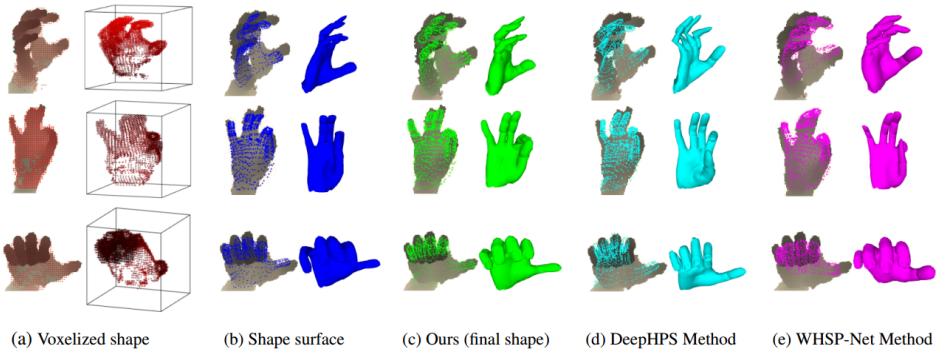
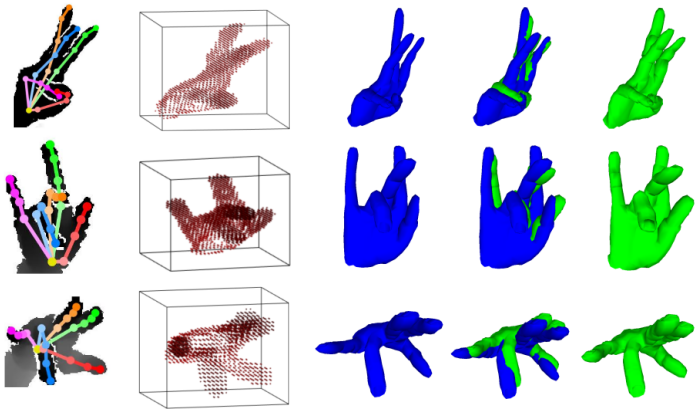


Figure 6.4.: *Shape reconstruction of NYU [tompson2014real] dataset: (a), (b) and (c) show the 2D overlays and 3D visualizations of estimated voxelized hand shape, shape surface, and the final shape after registration, respectively. (d) and (e) show the corresponding results of hand shapes from DeepHPS [malik2018deephps] and WHSP-Net [malik2019whsp] methods. Our approach produces visually more accurate hand shapes than the existing approaches.*

benchmarks (*e.g.,* BigHand2.2M). BigHand2.2M is a million-scale real benchmark. For pose estimation, it provides accurate joint annotations for 956K training ($\mathcal{T}_{\text{B}}$) depth images acquired from 10 subjects. Their hand model contains 21 joint locations which resemble real hand skeleton. The size of the BigHand2.2M's test set is 296K. The annotation of hand palm center is not given in the BigHand2.2M dataset. Hence, we obtain the hand palm center position by taking the average of the metacarpal joints and the wrist joint positions. SynHand5M dataset contains fully annotated 5 million depth images for both the 3D hand pose and shape. The sizes of it's train ($\mathcal{T}_{\text{S}}$) and test sets are 4.5M and 500K, respectively. The joint annotations of BigHand2.2M are fully compatible with SynHand5M.

We use three evaluation metrics: (i) the average 3D joint location error over all test frames (3D $\mathcal{J}$ Err.); (ii) mean vertex location error over all test frames (3D $\mathcal{V}$ Err.); and (iii) mean voxelized shape error (i.e. per-voxel binary cross entropy) over all test data (3D $\mathcal{S}$ Err.).



(a) (b) (c) (d) (e)

Figure 6.5.: *Shape reconstruction of BigHand2.2M dataset: (a) shows the 2D pose overlay. (b) and (c) show recovered voxelized shape and shape surface, respectively. (d) shows the overlays of shape surface and registered shape. (e) shows the final hand shape.*

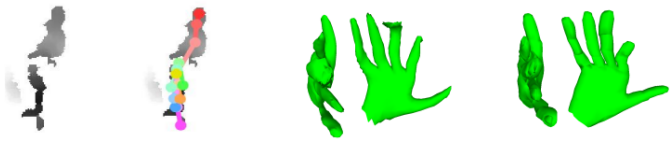
Components	Runtime (*sec*)
V2V-PoseNet	0.011
V2V-ShapeNet	0.0015
V2S-Net	0.0038
DispVoxNet	0.162
*NRGA	59 - 70

Table 6.3.: *Runtime: (first four rows) forward-pass of deep networks on GPU. * shows that the NRGA runs on CPU.*

6.5.2. Evaluation of Hand Shape Estimation

In this subsection, we evaluate our method on SynHand5M, NYU and BigHand2.2M benchmarks.

Synthetic Hand Shape Reconstruction. We train our complete pipeline on the fully labeled SynHand5M dataset by following the training methodology explained in Section 6.4. We conduct two ablation studies to show the effectiveness of our design choice. First is the regression of $\mathcal{V}_T$ and $\mathcal{V}_S$ by using input V'_D (i.e. without (w/o) $\mathcal{H}_j$) and the synthesizers. Similar experiments are repeated by using $\mathcal{H}_j$ (i.e. w/o V'_D) and $\mathcal{I}_S$ (i.e. with (w/) $\mathcal{I}_S \oplus \mathrm{V}'_\mathrm{D}$) as separate inputs to V2V-ShapeNet and V2S-Net. The results are summarized in Table 6.1, which clearly show the benefit of concatenating voxelized depth map with 3D heatmaps. The second ablation study is to observe the impacts of V2V-SynNet and S2V-SynNet, given $\mathcal{I}_S$ as an input to the complete shape estimation network. We trained V2S-Net and V2V-ShapeNet with and without using their respective synthesizers (see Figure 6.2). The quantitative results and comparisons with the state-of-the-art methods on SynHand5M test



(a) Depth (b) 3D Pose (c) DispVoxNet (d) NRGA

Figure 6.6.: *Failure case: Our method is unable to produce plausible shapes in cases of severe occlusion and missing depth information.*

set are summarized in Table 6.2. Our method with synthesizers improves on ours without synthesizers, and achieves 47.8% improvement in accuracy compared to a very recent approach WHSP-Net [malik2019whsp]. The qualitative results of synthesized samples of voxelized depth maps are shown in the Appendix A.8.3. The qualitative results of shape representations and pose are shown in Figure 6.3. DispVoxNet accurately fits the estimated hand surface to the estimated voxelized hand shape thereby, improving the hand surface reconstruction accuracy by 20.5% (i.e. $3.36mm \rightarrow 2.67mm$). Notably, in Table 6.1, the accuracy of our hand surface estimation is better than WHSP-Net method (see Table 6.2), which clearly shows the effectiveness of employing 3D CNN based network for mesh vertices regression.

Real Hand Shape Reconstruction. To estimate plausible real hand shape representations, the synthesizers are essential (see Figure 6.2). For NYU hand surface and voxelized shape recovery, we combine the training sets of NYU and SynHand5M (i.e. $\mathbf{T}_{\mathrm{NS}} = \mathcal{T}_{\mathrm{N}} + \mathcal{T}_{\mathrm{S}}$) by selecting closely matching 22 common joint positions in both the datasets. However, note that the common joint positions are still not exactly similar in both the datasets. V2S-Net and V2V-ShapeNet recover plausible hand shape representations while NRGA-based method performs a successful registration (as shown in Figure 6.4(a),(b) and (c)). It is observed that the voxelized shape is more accurately estimated than the hand surface. Thereby, the registration process is able to further refine the hand surface. Using the similar combined training strategy, we combine BigHand2.2M and SynHand5M datasets and shuffle them (i.e. $\mathbf{T}_{\mathrm{BS}} = \mathcal{T}_{\mathrm{B}} + \mathcal{T}_{\mathrm{S}}$). Samples of estimated hand shape representations for BigHand2.2M dataset are shown in Figure 6.5.

We qualitatively compare our reconstructed hand shapes of NYU dataset with state-of-the-art methods. For better illustration of shape reconstruction accuracy, we show the 2D overlay of hand mesh onto the corresponding depth image (as shown in Figure 6.4(d) and (e)). Model-based DeepHPS method [malik2018deephps] suffers from artifacts while regression-based WHSP-Net approach [malik2019whsp] produces perspective distortions and incorrect sizes of shapes. Whereas our method (Figure 6.4(c)) recovers visually more accurate and reasonable hand shapes. Table 6.3 shows the runtimes of different components of our pipeline.

Failure Cases. Our approach fails to estimate plausible hand shapes in cases of severe occlusion of hand parts and missing information in the depth map (see Figure 6.6).

Methods	3D $\mathcal{J}$ Err. (mm)
DeepHPS [malik2018deephps]	6.30
WHSP-Net [malik2019whsp]	4.32
V2V-PoseNet	3.81
Ours (Full method)	**3.75**

Table 6.4.: *3D hand pose estimation results on SynHand5M dataset. We compare the accuracy of our full method (i.e. HandVoxNet) with state-of-the-art methods.*

Dataset	Method	3D $\mathcal{J}$ Err. (mm)
NYU	V2V-PoseNet [moon2017v2v]	9.22
	V2V-PoseNet (w/ our 3D aug.)	**8.72**
BigHand2.2M	V2V-PoseNet [moon2017v2v]	9.95
	V2V-PoseNet (w/ our 3D aug.)	**9.27**

Table 6.5.: *3D hand pose estimation results on NYU [tompson2014real] and BigHand2.2M [yuan2017bighand2] datasets using our 3D data augmentation.*

6.5.3. Evaluation of Hand Pose Estimation

In our approach, the accuracy of the estimated hand shape is dependent on the accuracy of estimated 3D pose (see Section 6.3). Therefore, the hand pose estimation method needs to be robust and accurate. In this work, we perform a new 3D data augmentation on voxelized depth maps which further improves the accuracy of 3D hand pose estimation on real datasets. Notably, our focus is to develop an effective approach for simultaneous hand pose and shape estimation. However, for completion, we show our results and comparisons on hand pose estimation with SynHand5M, NYU and BigHand2.2M datasets.

SynHand5M Dataset: We do not perform training data augmentation on SynHand5M because this dataset originally contains large viewpoint variations [malik2018deephps]. We train our full method and V2V-PoseNet [moon2017v2v] on SynHand5M dataset. The quantitative results on the test set are presented in Table 6.4. We observed that the backpropagation from the shape regression pipeline is effective thereby, improves the accuracy of the estimated 3D pose. We achieve 13.19% improvement in the accuracy compared to WHSP-Net approach [malik2019whsp].

NYU and BigHand2.2M Datasets: V2V-PoseNet [moon2017v2v] is a powerful pose estimation method that exploits the 3D data representations of hand pose and depth map. By employing the 3D data augmentation strategy (see Section 6.3.2), we achieve 5.42% and 6.83% improvements in the accuracy compared to the original V2V-PoseNet models on NYU and BigHand2.2M datasets, respectively (see Table 6.5). Figure 6.8 shows the average errors on individual hand joint positions. We observe a noticeable improvement in the accuracy of the finger-tips. The qualitative results and comparisons with the state-of-the-art methods for hand pose estimation are shown in Figure 6.7.

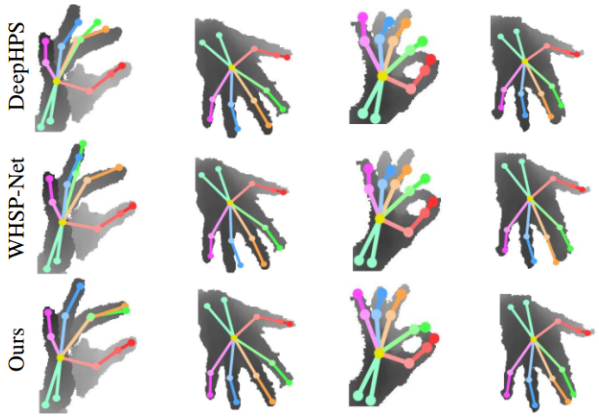


Figure 6.7.: *Samples of NYU [tompson2014real] depth images with 2D overlay of the estimated 3D hand pose. Our method produces more accurate results compared to WHSP-Net [malik2019whsp] and DeepHPS [malik2018deephps] methods.*

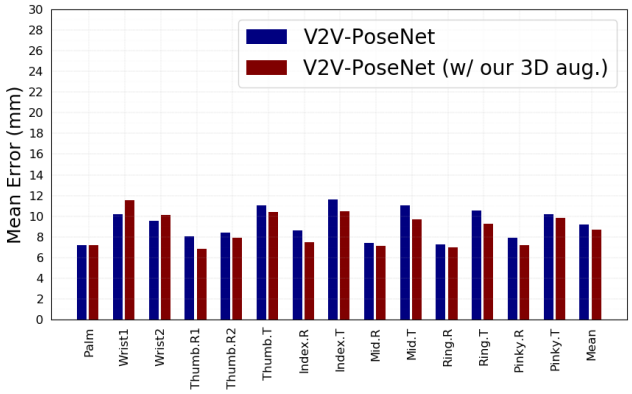


Figure 6.8.: *We study the impact of our 3D data augmentation on the pose estimation accuracy of V2V-PoseNet [moon2017v2v] on NYU [tompson2014real] dataset. The graph shows mean errors on individual hand joints.*

6.6. Conclusion

We propose a novel voxel-based approach for 3D hand shape and pose estimation from a single depth map. 3D voxelized representation of depth map and accurately estimated hand joints 3D heatmaps are used to reconstruct two different hand shape representations (i.e. voxelized shape and shape surface). To combine the pros of the two shape representations, we propose two different hand shape registration methods for synthetic and real data which accurately fit the shape surface to the voxelized shape. Furthermore, our 3D data augmentation policy on voxelized grids enhances the accuracy of 3D hand pose estimation on real datasets. Our method outperforms the state-of-the-art for hand shape recovery on synthetic dataset, and produces visually more accurate and reasonable hand shapes of real images compared to the previous

methods. These results indicate that the one-to-one mapping between voxelized depth map, voxelized shape and 3D joints heatmaps is essential for an accurate hand shape and pose recovery.

Chapter 7

In-Air Signature Verification using Depth Camera

Contents

7.1. Introduction

In-air signature is a new modality which allows a user to sign in the air by making free hand movements thereby, eliminating the need for a writing surface. Notably, this modality inherently contains important information in the third dimension (i.e., depth), in addition to the 2D spatial pattern. Existing methods for in-air signature verification use either an RGB or depth camera, a wearable camera (e.g., Google Glass) or a movement sensor in a cell

phone [fang2017novel; jeon2012system; sajid2015vsig; bailador2011analysis]. However, these methods address the problem of in-air signature acquisition and verification in the conventional way. More precisely, the focus of these approaches has been inclined towards the utilization of the 2D spatial and temporal features. Lack of consideration towards the hidden depth information has restricted the exploration of the full potential in the 3D signature trajectory. In this work, we investigate the potential of the unique depth pattern. We show that the depth itself is a strong feature, which is sufficient for in-air signature verification. On the other hand, fingertip tracking is a challenging problem, especially due to the occlusions of fingers and viewpoint changes during signing freely in the air. The acquisition of a correct in-air signature trajectory is crucial to verification. This problem has not been well-addressed because the existing approaches try to locate only the fingertip using heuristics. Some of the approaches rely on palm center point tracking [jeon2012system; khoh2018air] which does not accurately mimic the pointing finger movement while signing in the air. Furthermore, due to their complex in-air signature acquisition systems, they are not suitable for real-time applications. In principle, the skeleton of a human hand is a kinematic structure where each child joint is connected to its parent joints [zhou2016model; malik2017simultaneous]. Therefore, for a stable and reliable tracking of the position of a fingertip, the complete 3D pose of a hand should be estimated. In contrast to existing fingertip-tracking approaches, we exploit the huge progress of the convolutional-neural-network (CNN) based hand pose estimation using a low cost multimodal depth sensor [yuan2018depth] and trained a CNN to estimate the hand joints' keypoints in 3D; see Section 7.3.3. Estimating a full hand pose is more stable, especially in the case of occluded fingertips, as it learns to estimate all features of the hand. We create our own database of in-air signatures for analysis and verification. We perform a detailed ablation study, which especially reveals the significance of the hidden depth feature in verification. We propose an improved spatial-features-based verification strategy which incorporates the depth information; see Section 7.5.1. We employed the most common and effective multidimensional dynamic time warping (MD-DTW) algorithm for matching, since our focus is to investigate and highlight the potential in individual features of the in-air signature using the best practice for verification.

7.2. Framework Overview

The block diagram of our proposed 3D in-air signature acquisition and verification framework is shown in Figure 7.1. For the signature acquisition, we propose a CNN-based hand pose estimation method to predict the 3D hand joint positions from a single depth image. The input depth frame D_i is captured using Intel's creative senz3D depth camera [camera˙Senz3D]; see Section 7.3.1 for details of our acquisition setup. The hand region is segmented from D_i using center of hand mass (CoM) followed by a crop function; see Section 7.3.2. The output D_s is fed to the PoseCNN, which predicts the 3D hand

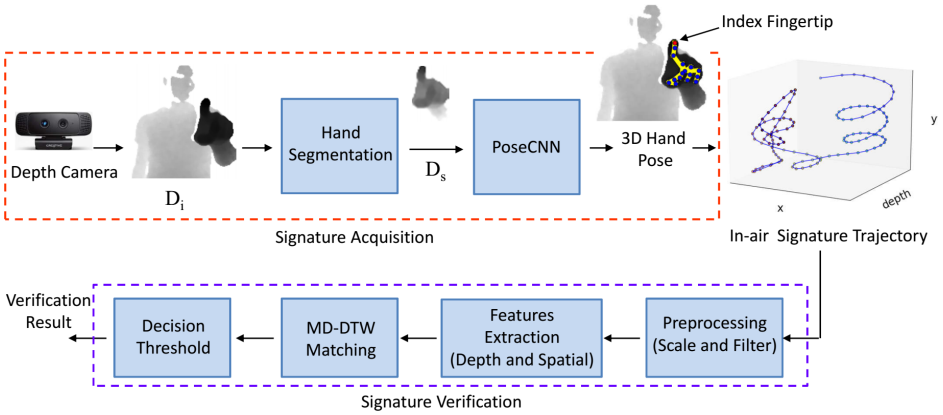


Figure 7.1.: *An overview of our method for in-air signature acquisition and verification. In the acquisition phase, the hand region is first segmented from a raw depth frame. Then, the estimated 3D position of the index fingertip is recorded for every frame using a CNN-based hand pose estimation method. For verification, the test signature is scaled and filtered. Thereafter, the spatial and depth features are extracted for matching using the MD-DTW algorithm. Finally, the test signature is verified by the decision threshold.*

pose; see Section 7.3.3. The estimated joint position of the index fingertip in each depth frame is used to record the 3D signature trajectory. The recorded in-air signature trajectory is preprocessed for normalization and smoothing; see Section 7.4.1. Thereafter, spatial and depth features are extracted from the 3D signature. For matching, MD-DTW is used to obtain a similarity measure between the selected feature of the preprocessed test signature and the corresponding precomputed feature template. In the final step, the test signature is verified by the decision threshold; see Sections 7.4.3 and 7.4.4.

7.3. In-Air Signature Acquisition

In this section, we explain our 3D in-air signature acquisition setup, fingertip-tracking approach, and the dataset creation.

7.3.1. Data Acquisition Setup

Figure 7.2 shows our in-air signature acquisition setup. A user is allowed to sign freely in the air within the field of view (FoV) of Intel's creative senz3D depth camera mounted on top of the screen. The FoV of the camera is 74° diagonal. Two position markers are placed on either side of the depth camera to provide an approximate start and end position for recording the signature. Our acquisition system allows to easily select between left or right hand before signing. During the signature acquisition, the user's hand should be the closest object to the camera. Notably, our method is not restricted to a specific hand pose for signing in the air. However, most of the users participating in our database creation used a natural pointing index finger pose (as shown

Figure 7.2.: *Our setup for in-air signature acquisition. The depth camera is mounted on top of the screen. The position markers on both sides of the depth camera allow capturing of in-air signature within the field of view (FoV) of the camera. Three GoPro cameras are placed around a user to record the hand motion in 3D space from different view points. Camera 3 specifically records the depth variation.*

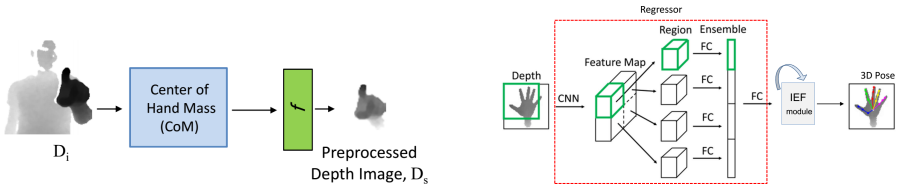
in Figure 7.1). Our system allows a user to see a 2D projection of the 3D signature trajectory in real-time on a signature pad, which is displayed on a monitor screen. Our acquisition system is robust to variations in ambient light intensity in indoor environments.

7.3.2. Hand Segmentation

An accurate segmentation of the hand region from a raw depth frame is important for learning-based hand pose estimation approaches. We used a hand segmentation method similar to that described in Reference [guo2017region] (Figure 7.3a). The segmentation process has two steps. The first step is to find an accurate 3D location of the hand palm center. As mentioned earlier, the hand is assumed to be the closest object to the camera; therefore, a simple depth value-based thresholding can be used to separate the human body from the hand. We used a depth threshold of 600 mm. Then, the 3D location of the palm center is calculated by averaging all the pixels which belong to the hand region (i.e., pixel values less than 600 mm). The second step is to preprocess or crop the hand region in 3D using the obtained palm center. In Figure 7.3a, the function f crops the hand region around the calculated palm center using a bounding box. The size of the bounding box is 150 mm. Then, depth values are normalized to $[-1, 1]$. The resultant image is of a size of 96×96. The runtime of our hand segmentation method is 0.47 ms.

7.3.3. Fingertip Tracking

Stable and reliable fingertip tracking is essential for the correct recording of a 3D in-air signature. For this purpose, we exploited the huge progress of CNN-based hand pose estimation methods. One of the major advantages associated with these methods is that they estimate the complete hand pose rather than detecting only the fingertip or palm center. This is particularly important in cases of severe occlusions of fingers during signing in the air. An overview of



(**a**) Hand Segmentation (**b**) The PoseCNN

Figure 7.3.: *(**a**) shows our approach for hand segmentation from a raw depth frame. First, the center of hand mass (CoM) is calculated, provided that the hand is the closest object to the depth camera. Then, the function f crops the hand region in 3D. (**b**) The PoseCNN takes the cropped hand image as input and regresses 3D sparse joints keypoints.*

our method is shown in Figure 7.3b. The PoseCNN is used to estimate the 16 3D joint positions of the hand skeleton from a single depth image. The first part of the PoseCNN (i.e., Regressor) is adopted from [guo2017region], which originally regressed 3D hand poses using a single shared CNN for feature extraction and a powerful yet simple region ensemble (REN) strategy. In our implementation, the final fully connected (FC) layer of the regressor outputs features $\varphi \in \mathbb{R}^{512}$ instead of joint positions.

Architecture of the Regressor: The architecture of the shared CNN for feature extraction comprises six convolution layers using 3×3 kernel sizes. A rectified Linear Unit (ReLu) is connected with each of the convolution layers as an activation function. A max pooling layer with a stride of 2 is connected after every consecutive pair of convolution layers. Two residual connections are incorporated between the pooling layers. The output features are of size $12 \times 12 \times 64$. Then, two FC layers of dimension 2048 are connected with a dropout ratio of 0.5. As shown in Figure 7.3b, the feature maps from different regions of the input depth image are divided into a 2×2 grid. Thereafter, the features from the FC layers of the grid regions are simply concatenated. The final FC layer after the concatenation produces $\varphi \in \mathbb{R}^{512}$. We refer the reader to Reference [guo2017region] for further details of the shared CNN architecture and the REN strategy.

IEF module: We integrate an iterative error feedback (IEF) module to the end of the regressor for refinement of the estimated hand pose. The output of the regressor φ is concatenated with an initial estimate of hand pose H_p i.e., $\phi = \{\varphi, H_p\}$. H_p is obtained by averaging all the joint positions from the ground truth annotations of the datasets. ϕ is fed to the IEF module, which comprises two FC layers with 512 neurons each. Both the FC layers use dropout layers with a ratio of 0.3. The last FC layer contains 48 neurons, corresponding to the 16 3D joint positions. The IEF module basically refines H_p in an iterative feedback manner such that $H_p(t+1) = H_p(t) + \delta H_p(t)$. We use three iterations.

Training of the PoseCNN: In order to improve the generic performance of the PoseCNN, especially for varying hand shapes, we trained on a combined dataset (i.e., *HandSet*) proposed in Reference [malik2017simultaneous]. Our

network runs on a desktop using Nvidia's Geforce GTX 1080 Ti GPU. We used a learning rate (LR) of 0.001 with a 0.9 stochastic gradient descent (SGD) momentum and a batch size of 256. One forward pass through the PoseCNN takes 3.2 ms.

Accuracy of predicted fingertips positions: We quantitatively evaluated the accuracy of estimated fingertips positions on the NYU test dataset. The 3D joint location error on fingertips comes out to be 13.2 mm, which is better than the lowest reported error (15.6 mm) in Reference [wang2018region]; see Table 7.1.

Table 7.1.: *The table shows the mean 3D joint location error (mm) for fingertips of various methods on the NYU [tompson2014real] hand pose test dataset.*

Method	3D Joint Loc. Err.
DeepModel [zhou2016model]	24.4 mm
Oberweger et al. [oberweger2015training]	23.2 mm
REN [wang2018region]	15.6 mm
Ours	13.2 mm

7.3.4. The Dataset Creation

There are two main motivations for creating our dataset for in-air signature verification. The first is to study the potential of the hidden depth feature. The second is to exploit the great progress in CNN-based hand pose estimation for stable and reliable fingertip tracking. For video recordings of genuine signatures which are shown to impostors, we used three GoPro cameras in our capture setup; see Figure 7.2. Two of the cameras (Cameras 1 and 2) were placed behind and right-front of the subject to record the spatial pattern of the signature. The third camera (Camera 3) recorded from the side view to visualize the depth variation in the signature. The users were asked to practice multiple times before the actual recordings as signing in the air is generally not a well-familiar modality. We emphasized on making explicit variations in depth during signing, which allows to fully exploit the hidden depth feature in the in-air signature trajectory. Our database includes 600 signatures from 15 users. We recorded 15 genuine signatures from each of the users and obtained 25 forgeries for every original writer from 5 impostors. Ten out of 15 genuine signatures were used for the testing phase and the remaining were used for the training phase; see Section 7.4. Samples of genuine preprocessed signatures with the corresponding 2D spatial views and unique depth patterns are shown in Figure 7.4. The color variations in the 3D view of a signature show variation in the depth pattern; see Figure 7.4a. Notably, each signature has a unique depth pattern (Figure 7.4c) which is challenging to forge jointly with the spatial pattern; see Section 7.5.

7.4. In-air Signature Verification

In this section, we explain the preprocessing, extracted features, training, and testing phases. We adopted a commonly used MD-DTW algorithm for matching, mainly because it can align temporal signals well even though they are not consistent in time.

7.4.1. Preprocessing

The recorded in-air signature is preprocessed for normalization and smoothing. An appropriate preprocessing of a signature can affect the results of signature verification [lee1996reliable; jeon2012system]. First, we removed a few redundant 3D points from the start and end of a signature trajectory whose displacement was less than 3 pixels. The removed points corresponded to a small wait time before starting the actual hand motion and a time to close the recording after the end of the signature. In order to remove discontinuities due to fast hand movements, we applied a moving average filter with a window size of 5, which resulted in a smoother signature trajectory. Thereafter, we normalized the signatures to compensate for variations in *position* and *scale*. For normalization, the transformation from absolute to relative values in 3D can be obtained using the following formulas:

$$X_j^* = (X_j - X_{min})/(X_{max} - X_{min}) \tag{7.1}$$

$$Y_j^* = (Y_j - Y_{min})/(Y_{max} - Y_{min}) \tag{7.2}$$

$$Z_j^* = (Z_j - Z_{min})/(Z_{max} - Z_{min}), \tag{7.3}$$

where X_j,Y_j, and Z_j are the original or absolute values of a signature. X_j^*, Y_j^*, and Z_j^* are the transformed values. X_{min},X_{max},Y_{min},Y_{max},Z_{min}, and Z_{max} are the minimum and maximum values of X_j,Y_j, and Z_j. A test signature before and after the preprocessing step is shown in Figure 7.5.

7.4.2. Feature Extraction

Figure 7.6 shows all the feature combinations we used in our verification process. We studied the impact of the hidden depth feature in different ways. The spatial (X,Y) is a commonly used 2D representation of in-air signatures; see Figure 7.6b. However, we argue that only the spatial (X,Y) is not a complete representative of an in-air signature trajectory. Therefore, we extracted two new types of spatial features, i.e., spatial (X,Z) and spatial (Y,Z) which implicitly incorporate the depth feature. We also studied the impact of these two features when combined with the spatial (X,Y); see Section 7.5. Nevertheless, the most interesting feature is the hidden depth pattern (Figure 7.6e) which has not been fully explored in the previous works.

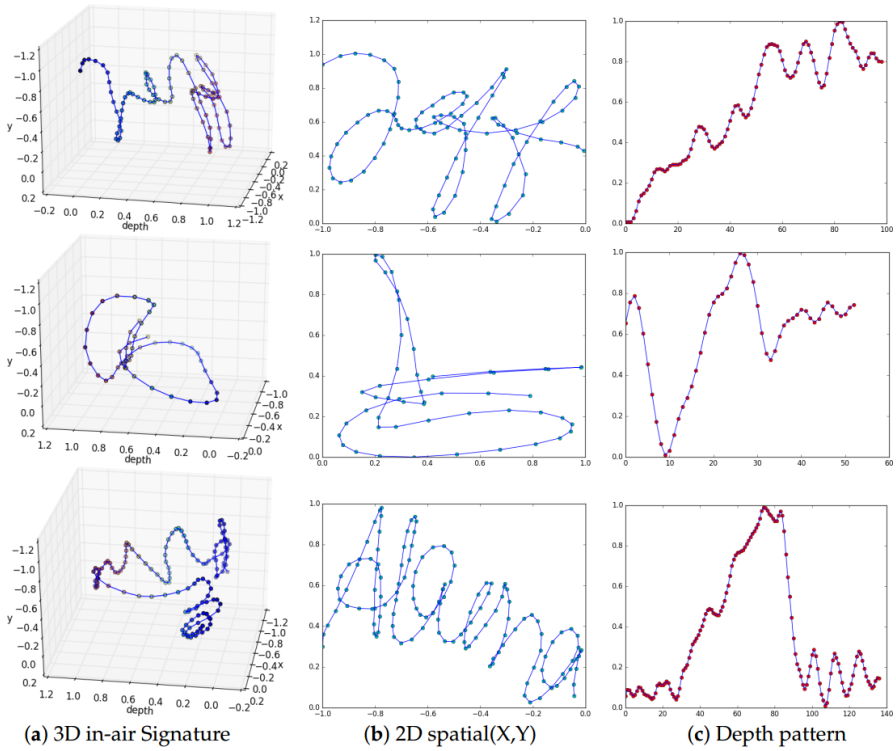


Figure 7.4.: *Samples of genuine in-air signatures from our dataset. Each one of the rows shows (a) the 3D in-air signature trajectory, (b) the 2D spatial view, and (c) depth pattern. The depth pattern of each signature is particularly unique and, therefore, it is an important hidden feature.*

7.4.3. Training Phase

In this phase, we computed the feature templates and the respective feature thresholds using 75 genuine training samples. We used neither forgeries nor original signatures from the test set. It is worth noting that many pattern recognition researchers use models, e.g., NN, SVM, while training them on the positive (genuine) and negative (forgery) samples at the same time [malik2012terminology; nguyen2007off]. According to forensic handwriting examiners [malik2013signature], this is unrealistic as, in the real world, one can never limit the forgery set and every signature, other than the concerned genuine signatures, can be considered a forgery. Furthermore, in real forensic cases, a verification system can only have genuine specimen samples and one or more questioned signatures. Henceforth, the best approach while using such models is to train them only on genuine specimen signatures. This can be done using specialized one class classifiers, like SVM/NN, for one class classification [guerbai2012one; bergamini2009combining; amer2013enhancing; manevitz2007one]. As explained earlier, we used five features; see Figure 7.6. Hence, a total of five feature templates and five respective feature thresholds for each of the 15 users are computed. A feature template is generated by

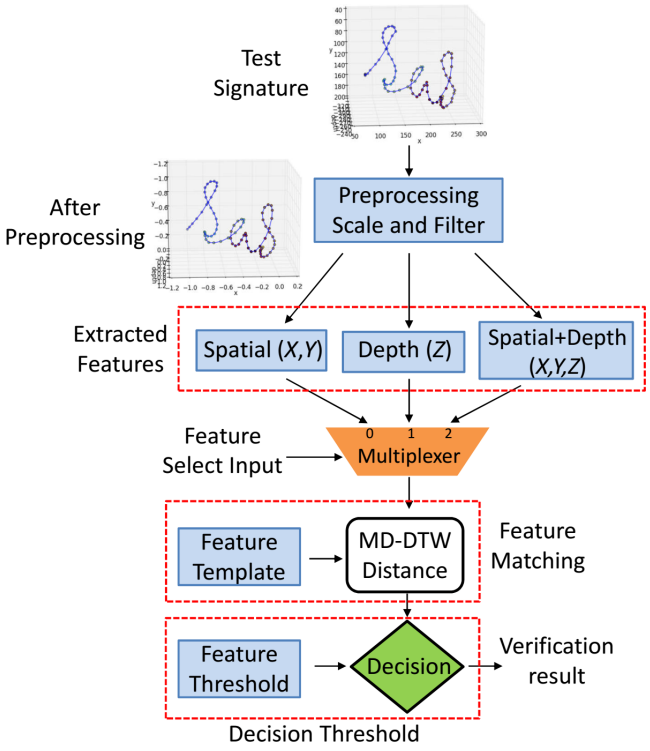
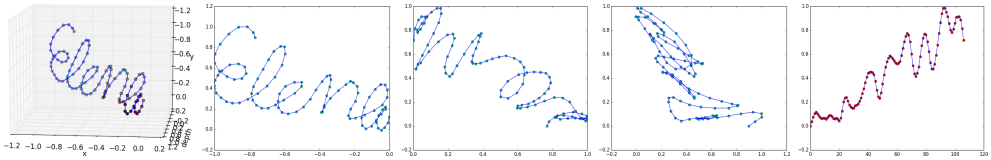


Figure 7.5.: *The flow diagram of the testing phase of our in-air signature verification system. The test signature is preprocessed for normalization and smoothing. The extracted features include spatial, depth, and spatial plus depth. Then, a multiplexer with a control input is used to select one of the extracted features. The selected feature is matched with the corresponding feature template using the MD-DTW algorithm. Finally, the verification result is produced by the decision threshold.*

averaging the features of the five training samples. We calculated a feature threshold value from five training samples of a signee, which are reserved for the training phase using the 4-fold cross validation strategy (i.e., using limited signatures for estimating how the system will perform when used to make predictions on data not used during training.

4-fold cross validation strategy: In this methodology, we randomly shuffled five genuine training signature samples and divided them into two groups. The first group contained four training samples, which were taken as the training set. The second group contained only one training sample, which was considered the dummy test set. More specifically, let $\mathbf{S} = \{S_{t_1}, S_{t_2}, S_{t_3}, S_{t_4}, S_{t_5}\}$ be the five training samples of a signature, where $S_x \in R^{dxL_x}$. L_x is the length of the signal S_x and d is the number of dimensions of one point in the signal. In the first round, we split $\mathbf{S}$ into two subsets, $\mathbf{S}_a = \{S_{t_2}, S_{t_3}, S_{t_4}, S_{t_5}\}$ and $\mathbf{S}_b = \{S_{t_1}\}$. This is simply taking the first sample S_{t_1} out of comparison in this round. For $\mathbf{S}_a$, we make a 4 x 4 confusion matrix $\mathbf{C}_1$ using Equations (A.11) and (A.12). From $\mathbf{C}_1$, we manually select a threshold value th_1 such that any compared threshold value greater than th_1 will declare the signature as forged.



(**a**) Signature (**b**) Spatial(X,Y) (**c**) Spatial(X,Z) (**d**) Spatial(Y,Z) (**e**) Depth

Figure 7.6.: *Illustration of different features which are used for in-air signature verification. We fully exploited different combinations of the features inherently present in the in-air signature trajectory to improve the performance of the verification system. The unique depth feature of a user especially plays a vital role in verification phase.*

In the second round, we eliminate S_{t_2} and calculate another 4 x 4 matrix $\mathbf{C}_2$ and find th_2 . In a similar way, we calculate $\mathbf{C}_3$, $\mathbf{C}_4$, and $\mathbf{C}_5$ and select the respective thresholds th_3, th_4, and th_5. Finally, we simply take the mean th_m of these five threshold values. The th_m is used in the final decision threshold process.

7.4.4. Testing Phase

Figure 7.5 shows the flow chart of the testing phase. After the preprocessing step and the feature extraction, a feature select input of a 3 x 1 multiplexer allows to select one of the features, i.e., spatial, depth, or spatial plus depth. After the selection of a desired feature, a similarity measure is found with the corresponding feature template using the MD-DTW algorithm [sanguansat2012multiple] as follows:

MD-DTW Matching: Let $s_1 \in \mathrm{R}^{d\mathrm{x}L_{s_1}}$ and $s_2 \in \mathrm{R}^{d\mathrm{x}L_{s_2}}$ be the two time series signals, where L_{s_1} and L_{s_2} are the lengths of s_1 and s_2, respectively, and d is the dimension of a single point in the signal. The distance matrix M(i,j) can be computed using the L2-norm without square root operation as:

$$\mathrm{M(i,j)} = \sum_{k=1}^{d} (s_1(k,\mathrm{i}) - s_2(k,\mathrm{j}))^2. \tag{7.4}$$

After obtaining the matrix M(i,j), the distance or similarity score between the elements of s_1 and s_2 on the DTW path can be found using the following equation:

$$\mathrm{D(i,j)} = \mathrm{M(i,j)} + \min \begin{cases} \mathrm{D(i\text{-}1,j)} \\ \mathrm{D(i\text{-}1,j\text{-}1)} \\ \mathrm{D(i,j\text{-}1)} \end{cases} \tag{7.5}$$

Decision Threshold: In the final step, as shown in Figure 7.5, the obtained similarity score is simply compared with the corresponding feature threshold th_m; see Section 7.4.3. The test signature is verified if the DTW distance is less than the feature threshold.

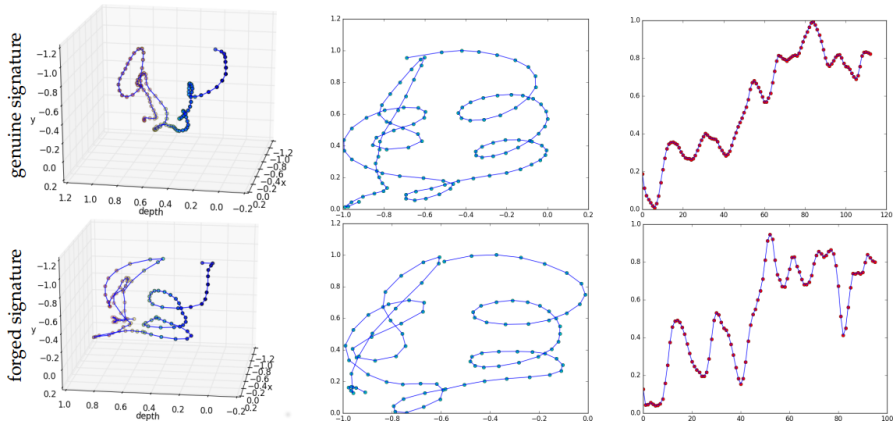


Figure 7.7.: *Comparison of spatial and depth patterns of the genuine and the corresponding forged signature. The top row shows a sample of a genuine signature and its corresponding spatial and depth patterns and the bottom one shows the respective forged signature. The color change shows the variation in depth pattern (3D view in the first column). Clearly, the depth pattern of the forged signature is different than the original one, although spatially they seem to be close.*

7.5. Experiments and Results

In this section, we detail the experiments performed on our dataset. The performances are reported using the false rejection rate (FRR), false acceptance rate (FAR), and equal error rate (EER) as evaluation metrics.

7.5.1. Ablation Study

In this subsection, we detail the ablation study, which was performed on the extracted features (Figure 7.6). The impact of every feature on the performance of verification was investigated and the results are reported on our captured dataset. We propose four different implementations of a verification module based on the extracted features from the in-air signature trajectory. *Depth-based signature verification (DSV) module*: To study the effectiveness of the hidden depth feature in verification, we implemented the verification module based on only the 1D depth Z of the signature trajectory. In Figure 7.5, the feature select input of the multiplexer is set to 1 in order to select the extracted depth feature from the test signature. The distance measure between the depth feature of the test signature and the precomputed depth feature template was calculated using Equations (A.11) and (A.12). The obtained similarity score was compared with the precomputed depth feature threshold to verify the test signature. Quantitative results on individual users are shown in Table 7.2. In Table 7.3, the *DSV* module shows FAR, FRR, and EER of 1.33%, 2.00%, and 0.51%, respectively. Qualitatively, the depth patterns of the genuine and forged signatures are shown in Figure 7.7. Despite the fact that the spatial patterns of the forgeries are closer to the genuine signatures, the

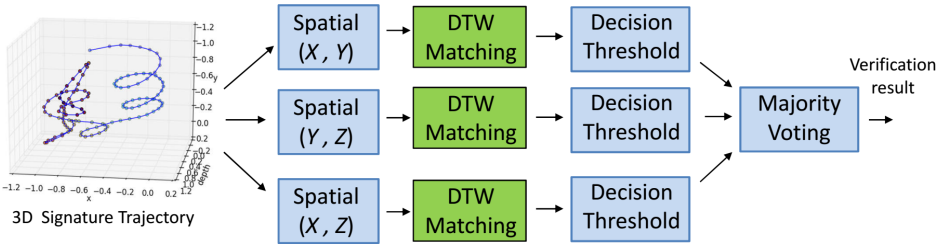


Figure 7.8.: *Our framework for an improved 2D spatial-based signature verification (ISSV) module. The spatial features (i.e., (X,Y), (Y,Z) and (X,Z)) are separately matched with the respective precomputed feature templates using a 2D-DTW algorithm. Thereafter, binary decisions are made by the decision thresholds. Lastly, the test signature is verified using a simple majority voting scheme.*

depth patterns are distinct. As mentioned ealier, the impostors were shown the video recordings of the signatures from different camera views. However, they were either unable to notice exact variations in depth or it was difficult to forge the depth pattern. These results show the importance of the depth feature, which alone can provide a reliable verification. We also observed that it is more challenging for the impostor to forge the depth pattern simultaneously with the spatial pattern.

2D spatial-based signature verification (SSV) module: We implemented this verification module using only the 2D spatial (X,Y) feature; see Figure 7.6b. The feature select input of the multiplexer was set to 0; see Figure 7.5. The similarity score between the extracted spatial feature of the test signature and the spatial (X,Y) feature template was obtained using Equations (A.11) and (A.12). Then, the DTW distance was compared to the spatial feature threshold for the verification. Quantitative results are shown in Tables 7.2 and 7.3. The performance of this verification module shows that considering only the spatial feature (X,Y) of the in-air signature trajectory results in a larger number of false acceptances and false rejections, thereby producing higher error rates.

Improved 2D spatial-based signature verification (ISSV) module: We attempted to improve the performance of the *SSV* module by incorporating additional spatial feature combinations (i.e., Spatial (X,Z) and Spatial (Y,Z)). The block diagram of the *ISSV* module is shown in Figure 7.8. The DTW matching is performed on these additional features in parallel to the traditional spatial (X,Y) using precomputed respective feature templates. Thereafter, binary decisions were obtained for each individual feature using the corresponding feature thresholds. Lastly, the final verification result was produced by a simple majority voting scheme, which declared the test signature as verified if no less than 2 features passed the corresponding decision thresholds. The verification results are reported in Tables 7.2 and 7.3 that clearly show an improved performance compared to the *SSV* module. There is a notable reduction in the number of false acceptances and false rejections. The EER is reduced by 15.9% compared to the *SSV* module. However, the performance

Table 7.2.: *The table shows the results of the four verification modules on our dataset. The number of false rejections (FR), false acceptances (FA), and total errors are provided for each of the 15 users. The 3D-SV module shows the least number of FA, while its number of FR is equivalent to the DSV module.*

Subj.	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**Err.**
DSV																
FR	0	0	0	0	0	0	0	1	0	0	0	0	1	0	1	3
FA	0	1	0	0	0	0	0	0	0	0	1	0	1	2	0	5
SSV																
FR	0	1	0	0	0	0	0	1	0	0	0	1	2	0	3	8
FA	0	2	0	0	0	1	0	2	0	0	2	0	1	3	0	11
ISSV																
FR	0	1	0	0	0	0	0	0	0	0	0	1	1	0	2	5
FA	0	1	0	0	0	0	0	1	0	0	1	0	1	2	0	6
3D-SV																
FR	0	1	0	0	0	0	0	0	0	0	0	1	0	0	1	3
FA	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	3

is still lagging behind the *DSV* module.

3D signature verification (3D-SV) module: In this verification module, we exploited the full 3D information (i.e., X, Y, Z) altogether. In Figure 7.5, the feature select input of the multiplexer was set to 2. The spatial plus depth feature (See Figure 7.6a) of the test signature was matched with the feature template and verified using the decision threshold. Quantitatively, Tables 7.2 and 7.3 show that number of false rejections and FRR of this verification module are the same as those for the *DSV* module, whereas the number of false acceptances, FAR, and EER are reduced. In summary, Our *3D-SV* module shows the best performance, since it includes complete 3D information altogether, which is inherently present in the in-air signature trajectory.

7.5.2. Comparison with Other Verification Methods

Since there are no publicly available datasets and codes available for in-air signatures, Table 7.4 lists the performances of other methods evaluated on their self-built datasets. Alongside, we show the performance of our two best implementations on our self-built dataset. Our *DSV* module shows the competitive performance, whereas the *3D-SV* module shows the best results. It shows that the hidden depth feature in the in-air signature is important for improved performance.

7.6. Conclusion

We present a real-time automatic in-air signature acquisition and verification framework using a low cost multi-modal depth camera. This work addresses

Table 7.3.: *The table shows the person independent FAR, FRR, and EER for each of the four verification modules. There are a total of 150 genuine test and 375 forged signatures. The 3D-SV module shows the best results, while the DSV module demonstrates competitive performance.*

Verification Module	FAR(%)	FRR(%)	EER(%)
DSV	1.33	2.00	0.51
SSV	2.93	5.33	0.69
ISSV	1.60	3.34	0.58
3D-SV	0.80	2.00	0.46

Table 7.4.: *The table shows the performances of the existing in-air signature methods and our method. Due to unavailability of a public dataset for in-air signatures, we report results on our dataset. While our 3D-SV module shows the best results, our DSV module, which is based on only depth analysis, shows the competitive performance.*

Method	Acq. Method	EER(%)
Nguyen et al. [diep2015sigver3d]	Accelerometer	EER: 0.8%
Hasan et al. [sajid2015vsig]	Google glass	Accuracy = 97.5%
Nidal et al. [kamel2008glove]	data glove	EER: 2.37%
Jeon et al. [jeon2012system]	depth camera	EER: 0.68%
Moon et al. [moon2017air]	Wifi signal	EER: 4.31%
DSV [**Ours**]	depth camera	EER: 0.51%
3D-SV [**Ours**]	depth camera	EER: 0.46%

two major limitations in the existing methods for in-air signature verification. First, given the fact that the existing approaches use heuristic methods for fingertip tracking, which are unstable and impractical, we propose a new CNN-based hand pose estimation method, which reliably tracks fingertips in real-time. The signature trajectory is recorded using an estimated 3D position of the index fingertip in each depth frame. Second, to explore the potential of the hidden depth feature in the in-air signature trajectory, we create our own dataset, which consists of 600 signatures recorded from 15 different subjects. We investigate the performance of the verification module by performing an ablation study on the spatial and depth features and performed extensive evaluations on our database. Experiments show that the depth feature itself is sufficient for in-air signature verification.

Chapter 8

Conclusion and Future Work

In this thesis we present solutions to important issues in CNN-based discriminative and hybrid approaches for 3D hand pose estimation. We introduce several approaches for simultaneous 3D hand shape and pose estimation which is new and challenging computer vision problem. Also, given the fact that annotating real images for shape is extremely hard and laborious, we propose million-scale synthetic datasets which provide accurate 3D pose and shape annotations for both egocentric and frontal camera viewpoints. Further, we present a practical application of 3D hand pose estimation i.e., In-Air signature verification, and create a new dataset which contains explicit variations in the depth patterns of the subjects.

More specifically, in Part I (Chapter 3), we introduce novel deep learning based structured hand pose estimation methods which explicitly consider the kinematic structure of hand skeleton during learning. In a model-based deep 3D hand pose estimation, we propose to simultaneously predict hand joint angles and scales to bone-lengths of a hand skeleton as intermediate representations. These representations are further passed to a new hybrid forward kinematics hand model layer which is embedded inside the deep network. We show that the simultaneous estimation of bone-lengths and pose parameters is essential for a generalized performance. Thereby, this allow to effectively train a hybrid approach on a dataset that contains varying hand shapes and sizes. Most CNN-based discriminative methods independently learn the sparse 3D hand joint positions, which may lead to kinematically unstable pose estimation. To address this issue, we propose a novel structure-aware CNN-based algorithm to jointly estimate 3D hand pose with new structural constraints. The constraints include fingers lengths, distances of joints along the kinematic chain and fingers inter-distances. Learning these constraints help to maintain a structural relation between the estimated joint keypoints.

In Part II, we present several novel methods for simultaneous 3D hand shape and pose estimation based on 2D and 3D convolutional networks. In Chapter 4, we introduce a novel Hand Pose and Shape Estimation (HPSE) layer which is embedded and optimised inside the deep network to generate both 3D hand pose and full hand mesh from learned hand pose, bones scales and shape

parameters. By employing a joint training strategy with real and synthetic data, we show plausible hand shape recovery from real depth images. However, the accuracy of this method is relatively low mainly because the optimization of complex hand pose and shape parameters inside a deep network is difficult and the representation capacity of synthetic hand model is limited. To address this issue, in Chapter 5, we propose structured regression-based approaches that does not suffer from the limitations of the model-based deep approach. In a weakly supervised approach, we show that reasonable hand shape can be learned and estimated by providing weak-supervisions on shape using input depth image and sparse 3D hand pose. Also, we present a simple and effective direct regression-based approach which learns to jointly estimate the 3D pose, the 3D shape and the structural constraints.

In Chapter 6 we present a novel 3D convolutions based architecture which establishes a one-to-one mapping between voxelized depth map and voxelized shape and 3D joint heatmaps. For shape estimation, our architecture produces two different hand shape representations. The first is the 3D voxelized grid of the shape which is accurate but does not preserve the mesh topology and the number of mesh vertices. The second representation is the 3D hand surface which is less accurate but does not suffer from the limitations of the first representation. To combine the advantages of these two representations, we register the hand surface to the voxelized hand shape. This method produces visually more accurate and reasonable hand shapes of real images compared to the previous methods. Furthermore, our 3D data augmentation policy on voxelized grids enhances the accuracy of 3D hand pose estimation on real datasets.

Finally, in Part III (Chapter 7), we present a real-time automatic in-air signature acquisition and verification framework using a depth camera. We show that in order to reliably record an In-Air signature trajectory, full 3D hand pose estimation is important rather than independently predicting only one fingertip position. Also, we demonstrate that depth is an important feature of In-Air signature which alone can produce highly reliable signature verification results.

There are some interesting future research directions which arise from the work presented in this thesis. One of the major challenges in simultaneous hand shape and pose estimation is the creation of an accurate real hand shape ground truth. In this respect, we have proposed synthetic datasets in herein however, they lack realism. Generating realistic synthetic dataset [mueller2018ganerated; shrivastava2017learning] is a potential research direction which can improve the recognition rates of a CNN-based network.

For an accurate 3D hand shape estimation, an efficient registration approach is important to fit a statistical hand model to an estimated voxelized grid of hand shape. Developing a new and effective hand shape registration approach can be a possible research direction. Whereas, we also plan to improve the runtimes of our registration methods that are proposed for deep voxel-based approach by realizing parallel computations with GPUs.

Keeping in view the significant progress in deep learning, In-Air signature

verification can be improved by developing accurate and robust deep learning based verification frameworks.

List of Figures

List of Tables

Bibliography

[mblab] *MB-Lab v1.5.0.* https://github.com/animate1978/MB-Lab. Feb. 2019.

[Adnane20193d] Philip H.S. Torr Adnane Boukhayma Rodrigo de Bem. "3D Hand Shape and Pose from Images in the Wild". In: *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[amer2013enhancing] Mennatallah Amer, Markus Goldstein, and Slim Abdennadher. "Enhancing one-class support vector machines for unsupervised anomaly detection". In: *Proceedings of the ACM SIGKDD Workshop on Outlier Detection and Description*. ACM. 2013, pp. 8–15.

[Ali˙NRGA˙2018] Sk Aziz Ali, Vladislav Golyanik, and Didier Stricker. "NRGA: Gravitational Approach for Non-Rigid Point Set Registration". In: *International Conference on 3DVision (3DV)*. 2018.

[amberg2007optimal] Brian Amberg, Sami Romdhani, and Thomas Vetter. "Optimal step nonrigid ICP algorithms for surface registration". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2007, pp. 1–8.

[bailador2011analysis] Gonzalo Bailador, Carmen Sanchez-Avila, Javier Guerra-Casanova, and Alberto de Santos Sierra. "Analysis of pattern recognition techniques for in-air signature biometrics". In: *Pattern Recognition* 44.10 (2011), pp. 2468–2478.

[baldi2012autoencoders] Pierre Baldi. "Autoencoders, unsupervised learning, and deep architectures". In: *Proceedings of ICML workshop on unsupervised and transfer learning*. 2012, pp. 37–49.

[barsoum2016articulated] Emad Barsoum. "Articulated hand pose estimation review". In: *arXiv preprint arXiv:1604.06195* (2016).

[bergamini2009combining] Cheila Bergamini, Luiz S Oliveira, Alessandro L Koerich, and Robert Sabourin. "Combining different biometric traits with one-class classification". In: *Signal Processing* 89.11 (2009), pp. 2117–2127.

[bhattacharya2013offline] Indrajit Bhattacharya, Prabir Ghosh, and Swarup Biswas. "Offline signature verification using pixel matching technique". In: *Procedia Technology* 10 (2013), pp. 970–977.

[blender] Blender. https://www.blender.org. Mar. 2018.

[cai2018weakly] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. "Weakly-supervised 3d hand pose estimation from monocular rgb images". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 666–682.

[chen2018shpr] Xinghao Chen, Guijin Wang, Cairong Zhang, Tae-Kyun Kim, and Xiangyang Ji. "SHPR-Net: Deep Semantic Hand Pose Regression From Point Clouds". In: *IEEE Access* 6 (2018), pp. 43425–43439.

[choi2011learning] Myung Jin Choi, Vincent YF Tan, Animashree Anandkumar, and Alan S Willsky. "Learning latent tree graphical models". In: *Journal of Machine Learning Research* 12.May (2011), pp. 1771–1812.

[chandra2016offline] Subhash Chandra and Sushila Maheskar. "Offline signature verification based on geometric feature extraction using artificial neural network". In: *Recent Advances in Information Technology (RAIT), 2016 3rd International Conference on*. IEEE. 2016, pp. 410–414.

[camera˙Senz3D] Creative. *Senz3D Interactive Gesture Camera*. https://us.creative.com/p/web-cameras/creative-senz3d. Mar. 2018.

[deng2017hand3d] Xiaoming Deng, Shuo Yang, Yinda Zhang, Ping Tan, Liang Chang, and Hongan Wang. "Hand3D: Hand Pose Estimation using 3D Neural Network". In: *arXiv preprint arXiv:1704.02224* (2017).

[dibra2017refine] Endri Dibra, Thomas Wolf, Cengiz Oztireli, and Markus Gross. "How to Refine 3D Hand Pose Estimation from Unlabelled Depth Data?" In: *In 3DV* (2017).

[dalal2016performance] Surjeet Dalal and Upasna Jindal. "Performance of integrated signature verification approach". In: *Computing for Sustainable Global Development (INDIACom), 2016 3rd International Conference on*. IEEE. 2016, pp. 3369–3373.

[diep2015sigver3d] Nguyen Ngoc Diep, Cuong Pham, and Tu Minh Phuong. "SigVer3D: Accelerometer Based Verification of 3-D Signatures on Mobile Devices". In: *Knowledge and Systems Engineering*. Springer, 2015, pp. 353–365.

[fang2017novel] Yuxun Fang, Wenxiong Kang, Qiuxia Wu, and Lei Tang. "A novel video-based system for in-air signature verification". In: *Computers & Electrical Engineering* 57 (2017), pp. 1–14.

[feng2003online] Hao Feng and Chan Choong Wah. "Online signature verification using a new extreme points warping technique". In: *Pattern Recognition Letters* 24.16 (2003), pp. 2943–2951.

[garcia2018first] Guillermo Garcia-Hernando, Shanxin Yuan, Seungryul Baek, and Tae-Kyun Kim. "First-person hand action benchmark with RGB-D videos and 3D hand pose annotations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 409–419.

[guerbai2012one] Yasmine Guerbai, Youcef Chibani, and Nassim Abbas. "One-class versus bi-class SVM classifier for off-line signature verification". In: *Multimedia Computing and Systems (ICMCS), 2012 International Conference on*. IEEE. 2012, pp. 206–210.

[ge2016robust] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel Thalmann. "Robust 3D hand pose estimation in single depth images: from single-view CNN to multi-view CNNs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3593–3601.

[ge2017robust] Liuhao Ge, Hui Liang, Junsong Yuan, and Daniel 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*. 2017.

[ge2018hand] Liuhao Ge, Yujun Cai, Junwu Weng, and Junsong Yuan. "Hand PointNet: 3d hand pose estimation using point sets". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8417–8426.

[ge20193d] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. "3D Hand Shape and Pose Estimation from a Single RGB Image". In: *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

[golyanik2018hdm] Vladislav Golyanik, Soshi Shimada, Kiran Varanasi, and Didier Stricker. "HDM-Net: Monocular Non-rigid 3D Reconstruction with Learned Deformation Model". In: *International Conference on Virtual Reality and Augmented Reality*. 2018, pp. 51–72.

[gordon20142012] Claire C Gordon, Cynthia L Blackwell, Bruce Bradtmiller, Joseph L Parham, Patricia Barrientos, Stephen P Paquette, Brian D Corner, Jeremy M Carson, Joseph C Venezia, Belva M Rockwell, et al. *2012 Anthropometric survey of US Army personnel: Methods and summary statistics*. Tech.

rep. Army Natick Soldier Research Development and Engineering Center MA, 2014.

[gruber2010online] Christian Gruber, Thiemo Gruber, Sebastian Krinninger, and Bernhard Sick. "Online signature verification with support vector machines based on LCSS kernel functions". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 40.4 (2010), pp. 1088–1100.

[ge2018point] Liuhao Ge, Zhou Ren, and Junsong Yuan. "Point-to-point regression pointnet for 3d hand pose estimation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 475–491.

[guo2017region] Hengkai Guo, Guijin Wang, Xinghao Chen, Cairong Zhang, Fei Qiao, and Huazhong Yang. "Region Ensemble Network: Improving Convolutional Network for Hand Pose Estimation". In: *In ICIP* (2017).

[he2016deep] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[ji2018hierarchical] Yanli Ji, Haoxin Li, Yang Yang, and Shuying Li. "Hierarchical topology based hand pose estimation from a single depth image". In: *Multimedia Tools and Applications* 77.9 (2018), pp. 10553–10568.

[jia2014caffe] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. "Caffe: Convolutional architecture for fast feature embedding". In: *Proceedings of the 22nd ACM international conference on Multimedia*. ACM. 2014, pp. 675–678.

[jeon2012system] Je-Hyoung Jeon, Beom-Seok Oh, and Kar-Ann Toh. "A system for hand gesture based signature recognition". In: *Control Automation Robotics & Vision (ICARCV), 2012 12th International Conference on*. IEEE. 2012, pp. 171–175.

[kumar2016survey] Ashok Kumar and Karamjit Bhatia. "A survey on offline handwritten signature verification system using writer dependent and independent approaches". In: *Advances in Computing, Communication, & Automation (ICACCA)(Fall), International Conference on*. IEEE. 2016, pp. 1–6.

[kennedy1995particle] James Kennedy and Russell Eberhart. "Particle swarm optimization (PSO)". In: *Proc. IEEE International Conference on Neural Networks, Perth, Australia*. 1995, pp. 1942–1948.

[keskin2012hand] Cem Keskin, Furkan Kıraç, Yunus Kara, and Lale Akarun. "Hand pose estimation and hand shape classification using multi-layered randomized decision forests". In: *Computer Vision–ECCV 2012* (2012), pp. 852–863.

[khoh2018air] Wee How Khoh, Ying Han Pang, and Andrew Beng Jin Teoh. "In-air hand gesture signature recognition system based on 3-dimensional imagery". In: *Multimedia Tools and Applications* (2018), pp. 1–25.

[katagiri2002personal] Masaji Katagiri and Toshiaki Sugimura. "Personal authentication by free space signing with video capture". In: *The 5th Asian Conference on Computer Vision*. Vol. 6. 2002.

[kamel2008glove] Nidal S Kamel, Shohel Sayeed, and Grant A Ellis. "Glove-based approach to online signature verification". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 30.6 (2008), pp. 1109–1113.

[krizhevsky2012imagenet] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[lee1996reliable] Luan L Lee, Toby Berger, and Erez Aviczer. "Reliable on-line human signature verification systems". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 6 (1996), pp. 643–647.

[lewis2000pose] John P Lewis, Matt Cordner, and Nickson Fong. "Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation". In: *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co. 2000, pp. 165–172.

[lewis2014practice] John P Lewis, Ken Anjyo, Taehyun Rhee, Mengjie Zhang, Frederic H Pighin, and Zhigang Deng. "Practice and Theory of Blendshape Facial Models." In: ().

[de2011model] Martin de La Gorce, David J Fleet, and Nikos Paragios. "Model-based 3d hand pose estimation from monocular video". In: *IEEE transactions on pattern analysis and machine intelligence* 33.9 (2011), pp. 1793–1805.

[lin2000modeling] John Lin, Ying Wu, and Thomas S Huang. "Modeling the constraints of human hand motion". In: *Proceedings workshop on human motion*. IEEE. 2000, pp. 121–126.

[malik2013signature] MI Malik, M Liwicki, L Alewijnse, W Ohyama, M Blumenstein, and B Found. "Signature verification and writer identification competitions for on-and offline skilled forgeries (sigwicomp2013)". In: *12th Int. Conf. on Document Analysis and Recognition, Washigton, DC, USA*. 2013.

[malik20183dairsig] Jameel Malik, Ahmed Elhayek, Sheraz Ahmed, Faisal Shafait, Muhammad Malik, and Didier Stricker. "3DAirSig: A Framework for Enabling In-Air Signatures Using a Multi-Modal Depth Sensor". In: *Sensors* 18.11 (2018), p. 3872.

[malik2018deephps] Jameel Malik, Ahmed Elhayek, Fabrizio Nunnari, Kiran Varanasi, Kiarash Tamaddon, Alexis Heloir, and Didier Stricker. "DeepHPS: End-to-end Estimation of 3D Hand Pose and Shape by Learning from Synthetic Depth". In: *International Conference on 3D Vision (3DV)*. 2018, pp. 110–119.

[ManuelBastionLAB] ManuelBastioni. *v1.5.0*. http://www.manuelbastioni.com. Mar. 2018.

[matsuo2007arm] Kenji Matsuo, Fuminori Okumura, Masayuki Hashimoto, Shigeyuki Sakazawa, and Yoshinori Hatori. "Arm swing identification method with template update for long term stability". In: *Advances in Biometrics* (2007), pp. 211–221.

[martens1996line] Ronny Martens and Luc Claesen. "On-line signature verification by dynamic time-warping". In: *Pattern Recognition, 1996., Proceedings of the 13th International Conference on*. Vol. 3. IEEE. 1996, pp. 38–42.

[mclain2010use] Trevor M McLain. "The use of factor analysis in the development of hand sizes for glove design". In: (2010).

[moon2017v2v] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. "V2V-PoseNet: Voxel-to-Voxel Prediction Network for Accurate 3D Hand and Human Pose Estimation from a Single Depth Map". In: *arXiv preprint arXiv:1711.07399* (2017).

[malik2017simultaneous] Jameel Malik, Ahmed Elhayek, and Didier Stricker. "Simultaneous Hand Pose and Skeleton Bone-Lengths Estimation from a Single Depth Image". In: *In International Conference on 3D Vision (3DV)* (2017).

[malik2018structure] Jameel Malik, Ahmed Elhayek, and Didier Stricker. "Structure-Aware 3D Hand Pose Regression from a Single Depth Image". In: *International Conference on Virtual Reality and Augmented Reality*. Springer. 2018, pp. 3–17.

[malik2019whsp] Jameel Malik, Ahmed Elhayek, and Didier Stricker. "WHSP-Net: A Weakly-Supervised Approach for 3D Hand Shape and Pose Recovery from a Single Depth Image". In: *Sensors* 19.17 (2019), p. 3784.

[maccormick2000partitioned] John MacCormick and Michael Isard. "Partitioned sampling, articulated objects, and interface-quality hand tracking". In: *European Conference on Computer Vision*. Springer. 2000, pp. 3–19.

[makris2015hierarchical] Alexandros Makris, Nikolaos Kyriazis, and Antonis A Argyros. "Hierarchical particle filtering for 3d hand tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2015, pp. 8–17.

[malik2012terminology] Muhammad Imran Malik and Marcus Liwicki. "From terminology to evaluation: Performance assessment of automatic signature verification systems". In: *Frontiers in handwriting recognition (ICFHR), 2012 international conference on*. IEEE. 2012, pp. 613–618.

[molenbroek04dined] JF Molenbroek. *DINED, Anthropometric Database*. `https://dined.io.tudelft.nl/`. 2004.

[moon2017air] Han-Cheol Moon, Se-In Jang, Kangrok Oh, and Kar-Ann Toh. "An In-Air Signature Verification System Using Wi-Fi Signals". In: *Proceedings of the 2017 4th International Conference on Biomedical and Bioinformatics Engineering*. ACM. 2017, pp. 133–138.

[mueller2017ganerated] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. "GANerated Hands for Real-time 3D Hand Tracking from Monocular RGB". In: *arXiv preprint arXiv:1712.01057* (2017).

[Mueller2017RealTimeHT] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. "Real-Time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017), pp. 1163–1172.

[mueller2018ganerated] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. "GANerated hands for real-time 3D hand tracking from monocular RGB". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 49–59.

[mueller2019real] Franziska Mueller, Micah Davis, Florian Bernard, Oleksandr Sotnychenko, Mickeal Verschoor, Miguel A Otaduy, Dan Casas, and Christian Theobalt. "Real-time pose and shape reconstruction of two interacting hands with a single depth camera". In: *ACM Transactions on Graphics (TOG)* 38.4 (2019), pp. 1–13.

[manevitz2007one] Larry Manevitz and Malik Yousef. "One-class document classification via neural networks". In: *Neurocomputing* 70.7-9 (2007), pp. 1466–1481.

[neverova2017hand] Natalia Neverova, Christian Wolf, Florian Nebout, and Graham W Taylor. "Hand pose estimation through semi-supervised and weakly-supervised learning". In: *Computer Vision and Image Understanding* 164 (2017), pp. 56–67.

[nguyen2007off] Vu Nguyen, Michael Blumenstein, Vallipuram Muthukkumarasamy, and Graham Leedham. "Off-line signature verification using enhanced modified direction features in conjunction with neural classifiers and support vector machines". In: *Document Analysis and Recognition, 2007. ICDAR 2007. Ninth International Conference on.* Vol. 2. IEEE. 2007, pp. 734–738.

[oikonomidis2011efficient] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. "Efficient model-based 3D tracking of hand articulations using Kinect." In: *BmVC.* Vol. 1. 2. 2011, p. 3.

[oikonomidis2011full] Iason Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. "Full dof tracking of a hand interacting with an object by modeling occlusions and physical constraints". In: *2011 International Conference on Computer Vision.* IEEE. 2011, pp. 2088–2095.

[oikonomidis2012tracking] Iasonas Oikonomidis, Nikolaos Kyriazis, and Antonis A Argyros. "Tracking the articulated motion of two strongly interacting hands". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition.* IEEE. 2012, pp. 1862–1869.

[oberweger2017deepprior++] Markus Oberweger and Vincent Lepetit. "Deepprior++: Improving fast and accurate 3d hand pose estimation". In: *International Conference on Computer Vision (ICCV) workshop*. Vol. 840. 2017, p. 2.

[oberweger2015hands] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. "Hands deep in deep learning for hand pose estimation". In: *In CVWW* (2015).

[oberweger2015training] Markus Oberweger, Paul Wohlhart, and Vincent Lepetit. "Training a feedback loop for hand pose estimation". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3316–3324.

[papazov2011deformable] Chavdar Papazov and Darius Burschka. "Deformable 3D shape registration based on local similarity transforms". In: *Computer Graphics Forum*. Vol. 30. 5. Wiley Online Library. 2011, pp. 1493–1502.

[panteleris2017using] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. "Using a single RGB frame for real time 3D hand pose estimation in the wild". In: *arXiv preprint arXiv:1712.03866* (2017).

[poier2019murauer] Georg Poier, Michael Opitz, David Schinagl, and Horst Bischof. "MURAUER: Mapping Unlabeled Real Data for Label AUstERity". In: *IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2019, pp. 1393–1402.

[qian2014realtime] Chen Qian, Xiao Sun, Yichen Wei, Xiaoou Tang, and Jian Sun. "Realtime and robust hand tracking from depth". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 1106–1113.

[robinette˙caesar˙1999] K.M. Robinette, H. Daanen, and E. Paquet. "The CAESAR project: a 3-D surface anthropometry survey". In: IEEE Comput. Soc, 1999, pp. 380–386. ISBN: 978-0-7695-0062-1. DOI: 10.1109/IM.1999.805368. URL: http://ieeexplore.ieee.org/document/805368/ (visited on 05/29/2018).

[rogez20143d] Grégory Rogez, Maryam Khademi, JS Supančič III, Jose Maria Martinez Montiel, and Deva Ramanan. "3d hand pose detection in egocentric rgb-d images". In: *European Conference on Computer Vision*. Springer. 2014, pp. 356–371.

[rad2017feature] Mahdi Rad, Markus Oberweger, and Vincent Lepetit. "Feature Mapping for Learning Fast and Accurate 3D Pose Inference from Synthetic Images". In: *arXiv preprint arXiv:1712.03904* (2017).

[robert2015offline] S Nithya Robert and B Thilagavathi. "Offline signature verification using support vectore machine". In: *Innovations in Information, Embedded and Communication Systems (ICIIECS), 2015 International Conference on*. IEEE. 2015, pp. 1–6.

[romero2017embodied] Javier Romero, Dimitrios Tzionas, and Michael J Black. "Embodied hands: modeling and capturing hands and bodies together". In: *ACM Transactions on Graphics (TOG)* 36.6 (2017), p. 245.

[MANO:SIGGRAPHASIA:2017] Javier Romero, Dimitrios Tzionas, and Michael J. Black. "Embodied Hands: Modeling and Capturing Hands and Bodies Together". In: *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*. 245:1–245:17 36.6 (Nov. 2017), 245:1–245:17.

[sanguansat2012multiple] Parinya Sanguansat. "Multiple multidimensional sequence alignment using generalized dynamic time warping". In: *WSEAS Transactions on Mathematics* 11.8 (2012), pp. 668–678.

[schmidhuber2015deep] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[sinha2016deephand] Ayan Sinha, Chiho Choi, and Karthik 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*. 2016, pp. 4150–4158.

[sharp2015accurate] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. "Accurate, robust, and flexible real-time hand tracking". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. ACM. 2015, pp. 3633–3642.

[shimada2019ismo] Soshi Shimada, Vladislav Golyanik, Christian Theobalt, and Didier Stricker. "IsMo-GAN: Adversarial Learning for Monocular Non-Rigid 3D Reconstruction". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2019.

[ShimadaDispVoxNets2019] Soshi Shimada, Vladislav Golyanik, Edgar Tretschk, Didier Stricker, and Christian Theobalt. "DispVoxNets: Non-Rigid Point Set Alignment with Supervised Learning Proxies". In: *International Conference on 3D Vision (3DV)*. 2019.

[shrivastava2017learning] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Josh Susskind, Wenda Wang, and Russ Webb. "Learning from simulated and unsupervised images through adversarial training". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 3. 4. 2017, p. 6.

[simon2017hand] Tomas Simon, Hanbyul Joo, Iain Matthews, and Yaser Sheikh. "Hand keypoint detection in single images using multiview bootstrapping". In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2. 2017.

[sridhar2013interactive] Srinath Sridhar, Antti Oulasvirta, and Christian Theobalt. "Interactive markerless articulated hand motion tracking using RGB and depth data". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2456–2463.

[sajid2015vsig] Hasan Sajid and S Cheung Sen-ching. "VSig: Hand-gestured signature recognition and authentication with wearable cam-

era". In: *Information Forensics and Security (WIFS), 2015 IEEE International Workshop on*. IEEE. 2015, pp. 1–6.

[stenger2006model] Björn Stenger, Arasanathan Thayananthan, Philip HS Torr, and Roberto Cipolla. "Model-based hand tracking using a hierarchical bayesian filter". In: *IEEE Transactions on Pattern Analysis & Machine Intelligence* 9 (2006), pp. 1372–1384.

[sun2015cascaded] Xiao Sun, Yichen Wei, Shuang Liang, Xiaoou Tang, and Jian Sun. "Cascaded hand pose regression". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 824–832.

[sun2017compositional] Xiao Sun, Jiaxiang Shang, Shuang Liang, and Yichen Wei. "Compositional human pose regression". In: *The IEEE International Conference on Computer Vision (ICCV)*. Vol. 2. 3. 2017, p. 7.

[tagliasacchi2015robust] Andrea Tagliasacchi, Matthias Schröder, Anastasia Tkach, Sofien Bouaziz, Mario Botsch, and Mark Pauly. "Robust Articulated-ICP for Real-Time Hand Tracking". In: *Computer Graphics Forum*. Vol. 34. 5. Wiley Online Library. 2015, pp. 101–114.

[tang2014latent] Danhang Tang, Hyung Jin Chang, Alykhan Tejani, and Tae-Kyun Kim. "Latent regression forest: Structured estimation of 3d articulated hand posture". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 3786–3793.

[tang2015opening] Danhang Tang, Jonathan Taylor, Pushmeet Kohli, Cem Keskin, Tae-Kyun Kim, and Jamie Shotton. "Opening the black box: Hierarchical sampling optimization for estimating human hand pose". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3325–3333.

[taylor2016efficient] Jonathan Taylor, Lucas Bordeaux, Thomas Cashman, Bob Corish, Cem Keskin, Toby Sharp, Eduardo Soto, David Sweeney, Julien Valentin, Benjamin Luff, et al. "Ef-

ficient and precise interactive hand tracking through joint, continuous optimization of pose and correspondences". In: *ACM Transactions on Graphics (TOG)* 35.4 (2016), p. 143.

[takeuchi2013multimodal] Akiji Takeuchi, Yusuke Manabe, and Kenji Sugawara. "Multimodal soft biometrie verification by hand shape and handwriting motion in the air". In: *Awareness Science and Technology and Ubi-Media Computing (iCAST-UMEDIA), 2013 International Joint Conference on*. IEEE. 2013, pp. 103–109.

[tompson2014real] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. "Real-time continuous pose recovery of human hands using convolutional networks". In: *ACM Transactions on Graphics (ToG)* 33.5 (2014), p. 169.

[tang2013real] Danhang Tang, Tsz-Ho Yu, and Tae-Kyun Kim. "Real-time articulated hand pose estimation using semi-supervised transductive regression forests". In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 3224–3231.

[van2007using] Bao Ly Van, Sonia Garcia-Salicetti, and Bernadette Dorizzi. "On using the Viterbi path along with HMM likelihood information for online signature verification". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 37.5 (2007), pp. 1237–1247.

[vollmer1999improved] Jörg Vollmer, Robert Mencl, and Heinrich Mueller. "Improved laplacian smoothing of noisy surface meshes". In: *Computer graphics forum*. Vol. 18. 3. Wiley Online Library. 1999, pp. 131–138.

[wan2017crossing] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. "Crossing nets: Combining gans and vaes with a shared latent space for hand pose estimation". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017.

[wan2018dense] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. "Dense 3d regression for hand pose estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 5147–5156.

[wang2018region] Guijin Wang, Xinghao Chen, Hengkai Guo, and Cairong Zhang. "Region Ensemble Network: Towards Good Practices for Deep 3D Hand Pose Estimation". In: *Journal of Visual Communication and Image Representation* (2018).

[wu2016learning] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. "Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling". In: *Advances in neural information processing systems*. 2016, pp. 82–90.

[wan2016hand] Chengde Wan, Angela Yao, and Luc Van Gool. "Hand Pose Estimation from Local Surface Normals". In: *European Conference on Computer Vision*. Springer. 2016, pp. 554–569.

[xu2016estimate] Chi Xu, Ashwin Nanjappa, Xiaowei Zhang, and Li Cheng. "Estimate hand poses efficiently from single depth images". In: *International Journal of Computer Vision* 116.1 (2016), pp. 21–45.

[xu2017lie] Chi Xu, Lakshmi Narasimhan Govindarajan, Yu Zhang, and Li Cheng. "Lie-X: Depth image based articulated object pose estimation, tracking, and action recognition on lie groups". In: *International Journal of Computer Vision* 123.3 (2017), pp. 454–478.

[yadav2013survey] Madhuri Yadav, Alok Kumar, Tushar Patnaik, and Bhupendra Kumar. "A survey on offline signature verification". In: *International Journal of Engineering and Innovative Technology (IJEIT) Volume* 2 (2013), pp. 337–340.

[yuan2017bighand2] Shanxin Yuan, Qi Ye, Björn Stenger, Siddhant Jain, and Tae-Kyun Kim. "Bighand2. 2m benchmark: Hand pose dataset

and state of the art analysis". In: *Computer Vision and Pattern Recognition (CVPR), 2017*. IEEE. 2017, pp. 2605–2613.

[yuan2018depth] Shanxin Yuan, Guillermo Garcia-Hernando, Björn Stenger, Gyeongsik Moon, Ju Yong Chang, Kyoung Mu Lee, Pavlo Molchanov, Jan Kautz, Sina Honari, Liuhao Ge, et al. "Depth-Based 3D Hand Pose Estimation: From Current Achievements to Future Goals". In: *IEEE CVPR*. 2018.

[ye2016spatial] Qi Ye, Shanxin Yuan, and Tae-Kyun Kim. "Spatial Attention Deep Net with Partial PSO for Hierarchical Hybrid Hand Pose Estimation". In: *European Conference on Computer Vision*. Springer. 2016, pp. 346–361.

[zimmermann2017learning] Christian Zimmermann and Thomas Brox. "Learning to estimate 3d hand pose from single rgb images". In: *International Conference on Computer Vision*. 2017.

[zhou2016model] Xingyi Zhou, Qingfu Wan, Wei Zhang, Xiangyang Xue, and Yichen Wei. "Model-based deep hand pose estimation". In: *In IJCAI* (2016).

Appendix A

Appendix

A.1. The HPSL Gradient Derivation

Here, we provide the detailed mathematical derivation of the **HPSL** functions. For backward-pass in the **HPSL**, we compute gradients of the following equation with respect to the layer inputs:

$$\mathbf{HPSL}(\Theta, \beta, \alpha) = \big(\boldsymbol{F}(\Theta, \alpha) \ , \Upsilon(\Theta, \beta, \alpha) \ \big). \tag{A.1}$$

Each vertex v_x in the reconstructed hand morphable model Ψ is deformed using linear blend skinning [lewis2000pose]. Hence, for every vertex, the gradient of Equation A.1 with respect to a shape parameter β_t can be computed as:

$$\frac{\partial(\mathrm{HPSL}_{\mathrm{v}_x})}{\partial \beta_t} = \sum_i \omega_i C_{j_i} (b_t^{\mathrm{v}_x} - b_0^{\mathrm{v}_x}) \qquad \text{for } t = 1, 2, \ldots, 7$$

where, $\mathrm{HPSL}_{\mathrm{v}_x} = \Upsilon_{\mathrm{v}_x}(\Theta, \beta, \alpha)$. According to Equation A.1, bones scales α influence the joints positions and vertices positions. Hence, the resultant gradient with respect to a hand scale α_s, can be calculated as:

$$\frac{\partial(\mathbf{HPSL})}{\partial \alpha_s} = \frac{\partial \mathbf{F}}{\partial \alpha_s} + \frac{\partial \Upsilon}{\partial \alpha_s} \qquad \text{for } s = 1, 2, \ldots, 6$$

To compute the partial derivative of $\mathbf{F}$ with respect to α_s, we need to derivate $\mathrm{F}_{j_i}(\Theta, \alpha)$ with respect to its associated scale parameter α_s. Hence,

$$\frac{\partial F_{ji}}{\partial \alpha_s} = \sum_{k \in S_k} \Big[\Big(\prod_{n \in S_{j_i}} [\mathrm{Rot}_{\phi_n}(\theta_n)] \times [\mathrm{Trans}(\alpha B)] \Big) [0, 0, 0, 1]^T \Big] \tag{A.2}$$

where,

$$\mathrm{Trans}(\alpha B) = \begin{cases} \mathrm{Trans}(\alpha_n B_n) & \text{if } n \neq k \\ \\ \mathrm{Trans}(\alpha_n B_n)' & \text{if } n = k \end{cases}$$

and, S_k is the set of parent joints of j_i that share the same scale parameter α_s. S_{j_i} is the set of joints along kinematic chain from j_i to the root joint and

ϕ is the rotation axis. In a similar way, the gradient of Υ_{v_x} with respect to α_s can be computed as:

$$\begin{aligned}\frac{\partial \Upsilon_{\mathrm{v}_x}}{\partial \alpha_s} &= \sum_i \omega_i \frac{\partial C_{j_i}}{\partial \alpha_s} \mathrm{v}_x \\ &= \sum_i \omega_i [\mathrm{M}_{j_i} (\mathrm{M}_{j_i}^{*-1})' + (\mathrm{M}_{j_i})' \mathrm{M}_{j_i}^{*-1}] \mathrm{v}_x\end{aligned}$$

The derivative of $\mathrm{M}_{j_i}(\Theta, \alpha)$ can be calculated by the following equation:

$$\mathrm{M}_{j_i}' = \sum_{k \in S_k} \Big[\Big(\prod_{n \in S_{j_i}} [\mathrm{Rot}_{\phi_n}(\theta_n)] \times [\mathrm{Trans}(\alpha B)] \Big) \Big] \tag{A.3}$$

where,

$$\mathrm{Trans}(\alpha B) = \begin{cases} \mathrm{Trans}(\alpha_n B_n) & \text{if } n \neq k \\ \mathrm{Trans}(\alpha_n B_n)' & \text{if } n = k \end{cases}$$

In order to calculate the derivative of $\mathrm{M}_{j_i}^{*-1}(\alpha)$, we can perform the following computation:

$$(\mathrm{M}_{j_i}^{*-1})' = -\mathrm{M}_{j_i}^{*-1} \mathrm{M}_{j_i}^{*\prime} \mathrm{M}_{j_i}^{*-1} \tag{A.4}$$

Likewise, for the pose parameters Θ, we compute the following equation:

$$\frac{\partial(\mathbf{HPSL})}{\partial \theta_p} = \frac{\partial \mathbf{F}}{\partial \theta_p} + \frac{\partial \Upsilon}{\partial \theta_p} \qquad \text{for } p = 1, 2, \ldots, 26$$

Accordingly, the derivative of $\mathrm{F}_{j_i}(\Theta, \alpha)$ with respect to a pose parameter θ_p, is simply to replace the rotation matrix of θ_p by its derivation as given by the following equation:

$$\frac{\partial F_{j_i}}{\partial \theta_p} = \Big(\prod_{n \in S_{j_i}} [\mathrm{Rot}_{\phi}(\theta)] \times [\mathrm{Trans}(\alpha_n B_n)] \Big) [0, 0, 0, 1]^T \tag{A.5}$$

where,

$$\mathrm{Rot}_{\phi}(\theta) = \begin{cases} \mathrm{Rot}_{\phi_n}(\theta_n) & \text{if } n \neq p \\ \mathrm{Rot}_{\phi_n}(\theta_n)' & \text{if } n = p \end{cases}$$

The derivative of Υ_{v_x} with respect to θ_p can be computed as:

$$\begin{aligned}\frac{\partial \Upsilon_{\mathrm{v}_x}}{\partial \theta_p} &= \sum_i \omega_i \frac{\partial C_{j_i}}{\partial \theta_p} \mathrm{v}_x \\ &= \sum_i \omega_i [(\mathrm{M}_{j_i})' \mathrm{M}_{j_i}^{*-1}] \mathrm{v}_x \qquad \text{for } p = 1, 2, \ldots, 26\end{aligned}$$

M_{j_i}' can be calculated by the following equation as:

$$\mathrm{M}_{j_i}' = \Big(\prod_{k \in S_{j_i}} [\mathrm{Rot}_{\phi}(\theta)] \times [\mathrm{Trans}(\alpha_k B_k)] \Big) \tag{A.6}$$

where,

$$\mathrm{Rot}_{\phi}(\theta) = \begin{cases} \mathrm{Rot}_{\phi_k}(\theta_k) & \text{if } k \neq p \\ \mathrm{Rot}_{\phi_k}(\theta_k)' & \text{if } k = p \end{cases}$$

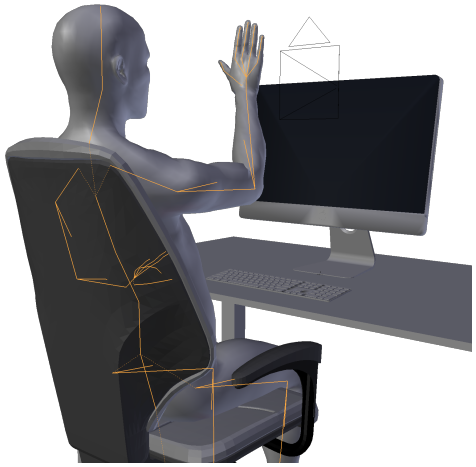


Figure A.1.: *The virtual desktop setup used to generate the images.*

A.2. SynHand5M Dataset

Here, we briefly discuss about the SynHand5M dataset generation. We have set up a realistic desktop environment (Figure A.1) by sitting the full character on an ergonomic chair, in front of an office desk on which lies a 27" monitor. As in most all-in-one PC configurations, the camera is embedded at the top-center of the monitor frame. In the default position, the hand palm faces the camera orthogonally and the fingers point up. The hand is positioned at 45cm from the camera. The center of the palm is aligned with the center of the camera view. Enforcing an ergonomically plausible posture for the whole character's body facilitates the coherent positioning of the elbow, thus leading to realistic wrist bending and forearm orientation. Our virtual camera simulates a Creative Senz3D Interactive Gesture Camera [camera˙Senz3D]. It renders images of resolution 320x240 using diagonal field of view of 74 degrees. One of the main objectives of our synthetic dataset is to provide training data with a wide range of variation, both for poses for hand shapes, such that a neural network model can be trained to accurately estimate average cases as well as extreme ones. In general, achieving a good range of variation for the hand poses is not a challenge; for a synthetic database, the image generator must randomize the fingers rotation of a min/max range of rotation, while for a real-world database subjects must over-articulate their motion in front of a camera. However, the variation in shape is more challenging for real-world databases, where a significant variability may not be present in a given cohort of human users. For example, the BigHand2.2M [yuan2017bighand2] database was captured from 10 users, and the MANO [MANO:SIGGRAPHASIA:2017] database was built from the contribution of 31 users. For this reason, for the generation SynHand5M hand shapes we rely of an artistic-driven body (and hand) generator. SynHand5M uses the hand model generated by ManuelBastionLAB [ManuelBastionLAB], which is a procedural full-body generator distributed as add-on of the Blender [blender] 3D authoring software. Without

constraints the hand generator can easily lead to impossible hand shapes, so we tune the ranges of the generation parameters using real-world statistical data from the DINED [molenbroek04dined] anthropometric database. DINED is a repository collecting the results of several anthropometric databases, including the CAESAR surface anthropometry survey [robinette˙caesar˙1999]. In the hand shape generator, the shape of the hand can be modulated on 7 hand shape parameters, namely: *Length*, *Mass*, *Size*, *Palm Length*, *Fingers Inter-distance*, *Fingers Length* and *Fingers Tip-Size*. In order to define realistic range limits, we extracted the average and standard deviation (sd) of the size of the hand of caucasian males from two DINED indices: (43) Hand Length, mean=183mm, sd=14; and (44) Palm Width, mean=83mm, sd=8. Then, we manually tuned the ranges of the *Hand-Size* parameters in order to cover the measured means ± 3sd (99% of the population). We first manually determined the min/max values of *Size*, *Mass*, and *InterDist* to match Palm Width (44) ± 3sd. Then, we determined the min/max values for *Length*, *PalmLength*, and *FingersLength* in order to match Hand Length (43) ± 3sd. Finally, since precise statistical data are unavailable for the *FingersTipSize*, we subjectively limited its range to $[0.2, 0.8]$ in order to avoid too unrealistic aspects. Since the 7 hand generation parameters accumulate in offsetting the same mesh vertices, in spite of the given constraints some parameters combination can lead to hand shapes beyond any statistical limits. However, in this context, the relative drop in realism is still acceptable because extreme hand shapes (although they will never match a real-world input) help the randomization to produce as much variability as possible and cover border-line cases.

The SynHand5M is generated by randomly sampling from the parameters, which are divided in three categories: hand pose, shape, and view point, and rendering the view from a virtual camera. To modulate the hand pose, we manipulate the 26 DoFs of our hand model; see Figure 3(b) in the main paper. For each finger, rotations are applied to flexion of all phalanges plus the abduction of the proximal phalanx. Additionally, in order to increase the realism of the closed fist configuration, the roll of middle, ring, and pinky fingers is derived from the abduction angle of the same phalanx. The rotation limits of the fingers have been set to bring the hand from a closed fist to an over-extended aperture, respecting anatomical constraints and avoiding the fingers to enter the palm. However, for some combinations, the rotation angles lead to inter-penetration between the thumb and one of the other fingers. Inter-penetrating configurations are automatically discarded from the dataset if a collision of the fingers' geometry occurs. In the default position, the hand palm faces the camera orthogonally and the fingers point up. The hand can rotate about three DoFs: roll around its longitudinal axis (i.e. along the forearm), rotate around the palm orthogonal axis (i.e. rolling in front of the camera), and rotate around its transversal axis (i.e. flexion/extension of the wrist). The SynHand5M database is divided in chunks of 100,000 images. Each chunk is stored in a different directory. Each chunk comes with a dataframe in CSV (Comma-separated values) text format. Each line of the dataframe

corresponds to a hand *configuration* (i.e., modulation of finger rotation, hand size, and hand rotation)

The dataframe columns report:

- The hand root x, y, z position in space.
- The hand root rotation in space, in both Euler-angles and quaternion format.
- The hand shape parameters.
- For each hand bone: the rotation relative to the parent bone (Euler angles), the location in space of the two bone extremities.

All of the above information are sufficient to re-generate the dataset images at different resolution. An additional information file contains details about the camera transformation and projection matrices, allowing for a conversion between 3D and pixel spaces.

For each configuration, the dataset contains three additional files:

- The depth image is saved as 16-bit gray-scale PNG file, where the gray value (in $[0, 65535]$) is the distance in mm from the camera sensor.
- The vertex coordinates are saved as x, y, z float triplets in a binary file.
- The colored segmentation of the hand is produced by performing an RGB rendering of the hand on which a manually painted texture is applied. The texture distinguishes between palm and phalanges. Segmentation images are saved as 24-bit true color PNG images.

A.3. Gradients of the bone-to-joint Layer

Here, we provide the gradient computation details of the bone-to-joint layer which is embedded in the structured 3D hand pose estimator (i.e., Module 1). For backward-pass in the layer, we compute gradient of the following function with respect to the layer inputs $\mathcal{B}$:

$$j_i = \Big(\prod_{k \in P_{j_i}} \mathrm{T}_{\phi_k}(\mathcal{B}_k) \Big) [0, 0, 0, 1]^{\mathrm{T}} \tag{A.7}$$

For each joint j_i, the gradient of Equation A.7 with respect to a translational value $\mathcal{B}_k$ can be computed as:

$$\frac{\partial j_i}{\partial \mathcal{B}_k} = \Big(\prod_{m \in P_{j_i}} \Gamma_{\phi_m}(\mathcal{B}_m) \Big) [0, 0, 0, 1]^{\mathrm{T}} \tag{A.8}$$

where,

$$\Gamma_{\phi_m}(\mathcal{B}_m) = \begin{cases} \mathrm{T}_{\phi_m}(\mathcal{B}_m) & \text{if } m \neq k \\ \mathrm{T}'_{\phi_m}(\mathcal{B}_m) & \text{if } m = k \end{cases}$$

and, $\mathrm{T}'_{\phi_m}(\mathcal{B}_m)$ is the derivation of translation matrix with respect to $\mathcal{B}_m$.

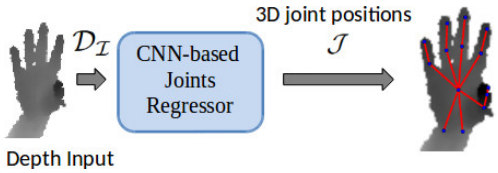


Figure A.2.: *Block diagram of Baseline 1 network. The preprocessed depth image input $\mathcal{D}_\mathcal{I}$ is passed to a CNN-based joints regressor which directly estimates the 3D joints coordinates $\mathcal{J}$. The CNN architecture is similar to [wang2018region].*

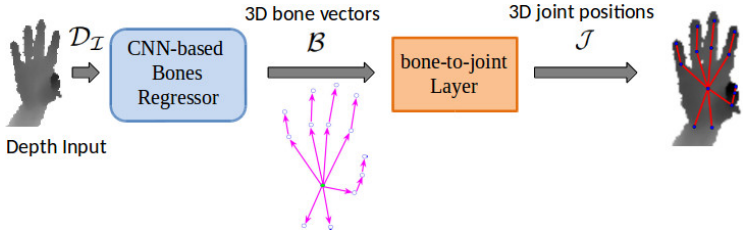


Figure A.3.: *Block diagram of Baseline 2 deep network. A CNN-based bones regressor estimates 3D bone vectors $\mathcal{B}$ from $\mathcal{D}_\mathcal{I}$ which are sent to a parameter free bone-to-joint layer that produces 3D joint positions $\mathcal{J}$. Learning 3D bone vectors helps to preserve the structure of the hand skeleton in training. The pipeline is trained in an end-to-end manner.*

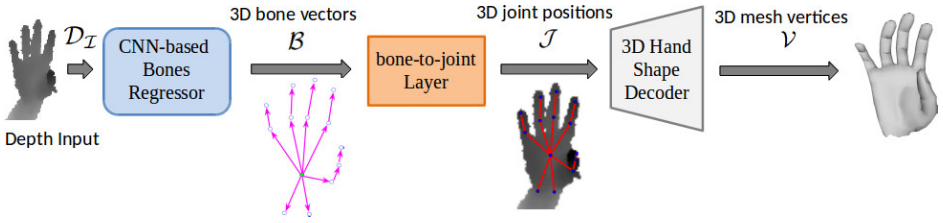


Figure A.4.: *Overview of Baseline 3 deep network for estimating both 3D hand pose and 3D hand shape. The joint positions $\mathcal{J}$ are estimated from a structured hand pose estimator which consists of the CNN-based bones regressor and the bone-to-joint layer. 3D hand shape decoder linearly decodes hand mesh vertices from joint positions. The complete pipeline is trained in an end-to-end manner.*

A.4. Network Details of the Baselines

We provide the network details of our **Baseline 1**, **Baseline 2** and **Baseline 3** (described in Section 6.1 in the paper). Figure A.2 shows the block diagram of Baseline 1. We leverage the CNN architecture proposed in [wang2018region] to directly regress 3D joint positions $\mathcal{J}$. We select this particular architecture because of its scalability and its effective region ensemble strategy which boosts the accuracy of estimated joint positions. The loss equation for Baseline 1 is given as:

$$\mathcal{L}_{\text{Baseline 1}} = \mathcal{L}_{\mathcal{J}} \tag{A.9}$$

In Baseline 2, a novel bone-to-joint layer is added to the network of Baseline 1. This addition allows to respect the structure of the hand; see Figure A.3. The main advantage of Baseline 3 over other model-based hand pose estimation methods is that the bone vectors are easier to learn than joint angles of kinematic hand model [sun2017compositional]. Therefore, we estimate 3D bone vectors as intermediate representation which are learned by the CNN-based bones regressor. These bone vectors are converted to joint coordinates using Equation A.7 by the bone-to-joint layer. The total loss of Baseline 2 is given as:

$$\mathcal{L}_{\text{Baseline 2}} = \mathcal{L}_{\mathcal{B}} + \mathcal{L}_{\mathcal{J}} \tag{A.10}$$

Where $\mathcal{L}_{\mathcal{B}}$ and $\mathcal{L}_{\mathcal{J}}$ are the loss terms for bones vectors and joints positions, respectively. Figure A.4 illustrates the pipeline of Baseline3 which estimates both 3D pose and 3D hand shape from a single depth image. The ground truth of hand shapes for real images is not available primarily because annotating such images for shape is time consuming and sub-optimal. Thus, the network of Baseline 3 is trained with mixed real and synthetic data. The total loss can be represented as:

$$\mathcal{L}_{\text{Baseline 3}} = \mathcal{L}_{\mathcal{B}} + \mathcal{L}_{\mathcal{J}} + \mathbb{1}\mathcal{L}_{\mathcal{R}} \tag{A.11}$$

where $\mathcal{L}_{\mathcal{R}}$ is the reconstruction loss for 3D hand mesh vertices and $\mathbb{1}$ is an indicator function. The implementation of $\mathbb{1}$ was achieved by adding a another new layer to our network. This layer forwards mesh vertices $\mathcal{V}$ to the euclidean loss layer only for synthetic images using a binary flag value which is 1 for synthetic and 0 for real. The gradients flow in backward pass is disabled for real data. However, mixed learning with labeled synthetic and unlabeled real data is not an optimal solution for learning accurate shapes. Therefore in our Full model, we provide a much needed weak-supervision using a 2D depth image synthesizer which learns to reconstructs depth images from estimated shapes.

A.5. Depth Image Synthesizer Evaluation

We provide the qualitative results of the proposed 2D depth image synthesizer. We train our Full model on combined NYU [tompson2014real] and SynHand5M [malik2018deephps] datasets by employing the network training

Figure A.5.: *Samples of synthesized 2D depth images of NYU [tompson2014real](Left), BigHand2.2M [yuan2017bighand2](Middle) and SynHand5M [malik2018deephps](Right) datasets. Top: Ground truth depth frames. Bottom: Synthesized depth images from the learned 3D hand shape. The proposed 2D depth image synthesizer reconstructs reasonable depth images and acts as an important source of weak supervision in training.*

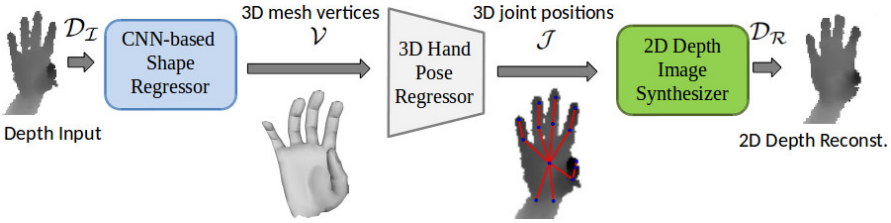


Figure A.6.: *Block diagram of Model 1 network. The preprocessed depth image $\mathcal{D}_\mathcal{I}$ is passed to the CNN-based hand shape regressor which directly regresses hand mesh vertices $\mathcal{V}$. Then, the 3D hand pose regressor estimates the joints coordinates from the reconstructed $\mathcal{V}$. Finally, the depth image $\mathcal{D}_\mathcal{R}$ is synthesized from the 3D pose $\mathcal{J}$ by the 2D depth image synthesizer.*

strategy (please refer to Section 5 in the paper). The qualitative results of samples of reconstructed NYU depth images are shown in Figure A.5(Left). We also train our Full model on combined BigHand2.2M [yuan2017bighand2] and SynHand5M datasets. The sample reconstructed images from BigHand2.2M dataset are shown in Figure A.5(Middle). The qualitative results of synthesized depth images of SynHand5M dataset are shown in Figure A.5(Right). Our depth synthesizer is able to reconstruct reasonable depth images from estimated hand mesh vertices during end-to-end training with mixed real and synthetic datasets. As the resulting depth images are just an estimate of the input images, they are not useful. However, accurately reconstructing them is an indication that the image synthesizer is doing its job correctly.

A.6. Network Architecture of Model 1

In order to show the effectiveness of our approach, we build a new pipeline (i.e., **Model 1**) which is inspired by the recent work [ge20193d]. The block

Figure A.7.: *3D hand mesh and 3D pose recovery from a single depth image. We show our estimation results on unseen images from real-time demo using Creative Senz3D camera.*

diagram of Model 1 is shown in Figure A.6. 3D hand mesh vertices $\mathcal{V}$ are directly regressed from a single depth image $\mathcal{D}_\mathcal{I}$ using a CNN-based shape regressor. The architecture of the CNN is similar to [wang2018region] which uses an ensemble of 9 different regions. However, we modify the last fully connected regression layer to output dense mesh $\mathcal{V}$ instead of joint positions. Then, 3D joint positions $\mathcal{J}$ are regressed from the estimated $\mathcal{V}$ using a linear 3D hand pose regressor. Finally, a 2D depth image synthesizer reconstructs 2D depth image from the estimated $\mathcal{J}$. The architecture of the synthesizer is similar to the one shown in Figure 5 of the paper except the first layer takes $\mathcal{J}$ instead of $\mathcal{V}$. The total loss equation of Model 1 is given as:

$$\mathcal{L}_{\text{Model 1}} = \mathbb{1}\mathcal{L}_\mathcal{R} + \mathcal{L}_\mathcal{J} + \mathcal{L}_\mathcal{D} \tag{A.12}$$

Where $\mathcal{L}_\mathcal{D}$ is the reconstruction loss for 2D depth image synthesizer. The qualitative and quantitative comparisons of Model 1 with our Full model are shown in Figure 8 and Table 2 in the paper. The results clearly show that directly regressing real hand mesh vertices from a single depth image is hard and it can lead to inaccurate shape estimation which consequently affects the accuracy of the estimated 3D pose. This fact is explicitly reported in [ge20193d] where they have used a pseudo ground truth of hand shapes of real images during fine-tuning with real data, in order to recover reasonable real hand shape using monocular RGB image. Whereas, our approach (Full model) is capable of recovering accurate and reasonable hand shapes without using any pseudo ground truth of real hand shapes.

A.7. Shape and Pose Estimation Results from Live Demo

We train our Full model on mixed real BigHand2.2M [yuan2017bighand2] and synthetic SynHand5M [malik2018deephps] datasets using the network training strategy explained in Section 5 of the paper. The network is trained till convergence. The system runs in real-time as it completes the forward pass in **2.9ms**. From live stream of Creative Senz3D depth camera [camera˙Senz3D], we preprocess a raw depth frame using a depth threshold value of 500mm and center of hand mass (CoM). Figure A.7 shows the 2D overlay of reconstructed pose and mesh from unseen depth images, and two different viewpoints of the reconstructed shape in 3D. These results show that our approach is capable of successfully recovering reasonable hand shapes and poses in real-time from unseen real world images. This illustrates that the proposed system is perfect for virtual reality application as it allows the user to see the animation of his hand in the virtual environment.

A.8. HandVoxNet Architecture

Here, we provide details about the network architectures of the V2V-ShapeNet, V2S-Net, V2V-SynNet, S2V-SynNet, and DispVoxNet.

A.8.1. Network Design of V2V-ShapeNet

V2V-ShapeNet regresses $\hat{\mathcal{V}}_S$ which is $64 \times 64 \times 64$ voxelized representation of hand shape, from input $\mathcal{I}_S$ (i.e. $(\mathrm{N}+1)$ $44 \times 44 \times 44$ voxelized grids). Since V2V-ShapeNet learns to estimate a dense 3D hand shape representation from sparse 3D hand joints and depth map, therefore it can be considered as a decoder which tries to reconstruct voxelized hand shape as close as possible to ground truth $\mathcal{V}_S$. V2V-ShapeNet establishes a one-to-one mapping between the voxelized hand shape, voxelized depth map, and 3D joints heatmaps. Table A.1 shows the architectural details of the 3D convolutions based V2V-ShapeNet. For weak-supervision, V2V-SynNet reconstructs the voxelized depth map from the estimated voxelized hand shape representation. The samples of reconstructed voxelized depth maps of NYU [tompson2014real] and BigHand2.2M [yuan2017bighand2] real benchmarks are shown in Figure A.8(a).

A.8.2. Network Design of V2S-Net

V2S-Net regresses K number of 3D hand mesh vertices $\hat{\mathcal{V}}_T$ from the input $\mathcal{I}_S$. The architecture of V2S-Net also consists of 3D convolutions except the last two layers that are fully connected (FC) layers. Table A.2 shows the architectural details of V2S-Net. Since V2S-Net regresses the shape in coordinates form, therefore it does not establish a one-to-one mapping between

ID	Layer	Output Sz	Kernel Sz	Stride/Padding	+
1	Input	(N+1) 44x44x44	-	-/-	-
2	3D Conv, BN, ReLU	(22) 44x44x44	7x7x7	1/3	-
3	3D Conv, BN, ReLU	(24) 38x38x38	7x7x7	1/0	-
4	3D Conv, BN, ReLU	(26) 32x32x32	7x7x7	1/0	-
5	3D Conv, BN, ReLU	(26) 32x32x32	3x3x3	1/1	-
6	3D Conv, BN	(26) 32x32x32	3x3x3	1/1	4
7	ReLU	(26) 32x32x32	-	-/-	-
8	3D DeConv, BN, ReLU	(8) 64x64x64	2x2x2	2/0	-
9	3D Conv, BN, ReLU	(8) 64x64x64	3x3x3	1/1	-
10	3D Conv, BN	(8) 64x64x64	3x3x3	1/1	8
11	ReLU	(8) 64x64x64	-	-/-	-
12	3D Conv, Sigmoid	(1) 64x64x64	1x1x1	1/0	-

Table A.1.: *V2V-ShapeNet architecture details: Output Sz consists of the number of channels and their spatial size. In + column, the output of layer ID is added with the current layer's output in a voxel-wise manner.*

the voxelized depth map and 3D joint heatmaps. For weak-supervision, S2V-SynNet reconstructs voxelized depth maps from the estimated hand mesh representation. The samples of reconstructed voxelized depth maps are shown in Figure A.8(b).

ID	Layer	Output Sz	Kernel Sz	Stride/Padding	+
1	Input	(N+1) 44x44x44	-	-/-	-
2	3D Conv, BN, ReLU	(22) 44x44x44	7x7x7	1/3	-
3	3D MaxPooling	(22) 22x22x22	2x2x2	2/0	-
4	3D Conv, BN, ReLU	(22) 22x22x22	3x3x3	1/1	-
5	3D Conv, BN	(22) 22x22x22	3x3x3	1/1	3
6	ReLU	(22) 22x22x22	-	-/-	-
7	3D Conv, BN	(16) 22x22x22	1x1x1	1/0	-
8	3D Conv, BN, ReLU	(16) 22x22x22	3x3x3	1/1	-
9	3D Conv, BN	(16) 22x22x22	3x3x3	1/1	7
10	ReLU	(16) 22x22x22	-	-/-	-
11	3D Conv, BN, ReLU	(8) 22x22x22	1x1x1	1/0	-
12	3D MaxPooling	(8) 11x11x11	2x2x2	2/0	-
13	3D Conv, BN, ReLU	(1) 11x11x11	1x1x1	1/0	-
14	Flatten	11*11*11	-	-/-	-
15	FC, ReLU	400	-	-/-	-
16	FC	K*3	-	-/-	-

Table A.2.: *V2S-Net architecture details.*

A.8.3. Network Designs of V2V-SynNet and S2V-SynNet

V2V-SynNet and S2V-SynNet act as sources of weak-supervisions during the training phase. They are not included in the testing phase. These synthesizers reconstruct voxelized depth maps $\hat{V}_D$ from the hand shape representations.

The details of the architectures of V2V-SynNet and S2V-SynNet are provided by Tables A.3 and A.4.

ID	Layer	Output Sz	Kernel Sz	Stride/Padding	+
1	Input	(1) 64x64x64	-	-/-	-
2	3D Conv, BN, ReLU	(8) 64x64x64	7x7x7	1/3	-
3	3D MaxPooling	(8) 32x32x32	2x2x2	2/0	-
4	3D Conv, BN	(16) 32x32x32	1x1x1	1/0	-
5	3D Conv, BN, ReLU	(16) 32x32x32	3x3x3	1/1	-
6	3D Conv, BN	(16) 32x32x32	3x3x3	1/1	4
7	ReLU	(16) 32x32x32	-	-/-	-
8	3D DeConv, BN, ReLU	(12) 38x38x38	7x7x7	1/0	-
9	3D Conv, BN, ReLU	(12) 38x38x38	3x3x3	1/1	-
10	3D Conv, BN	(12) 38x38x38	3x3x3	1/1	8
11	ReLU	(12) 38x38x38	-	-/-	-
12	3D DeConv, BN, ReLU	(8) 44x44x44	7x7x7	1/0	-
13	3D Conv, BN, ReLU	(8) 44x44x44	3x3x3	1/1	-
14	3D Conv, BN	(8) 44x44x44	3x3x3	1/1	12
15	ReLU	(8) 44x44x44	-	-/-	-
16	3D Conv, BN, ReLU	(8) 44x44x44	1x1x1	1/0	-
17	3D Conv, BN, ReLU	(8) 44x44x44	1x1x1	1/0	-
18	3D Conv, Sigmoid	(1) 44x44x44	1x1x1	1/0	-

Table A.3.: *V2V-SynNet architecture details.*

ID	Layer	Output Sz	Kernel Sz	Stride/Padding	+
1	Input	K*3	-	-/-	-
2	FC, ReLU	400	-	-/-	-
3	Reshape	(400) 1x1x1	-	-/-	-
4	3D DeConv, BN, ReLU	(128) 3x3x3	3x3x3	1/0	-
5	3D DeConv, BN, ReLU	(64) 6x6x6	3x3x3	2/1	-
6	3D DeConv, BN, ReLU	(32) 11x11x11	6x6x6	1/0	-
7	3D DeConv, BN, ReLU	(16) 22x22x22	3x3x3	2/1	-
8	3D Conv, BN, ReLU	(8) 44x44x44	1x1x1	1/0	-
9	3D Conv, BN, ReLU	(8) 44x44x44	1x1x1	1/0	-
10	3D Conv, Sigmoid	(1) 44x44x44	1x1x1	1/0	-

Table A.4.: *S2V-SynNet architecture details.*

A.8.4. Network Design of DispVoxNet

In contrast to the original DispVoxNets [ShimadaDispVoxNets2019] composed of the displacement estimation and refinement stages, we replace the refinement stage with Laplacian smoothing [vollmer1999improved]. By this, we simplify the pipeline and avoid the training of another instance of DispVoxNet in the refinement stage. We follow the original network architecture in [ShimadaDispVoxNets2019] (see Table A.5). DispVoxNet accepts two voxel grids

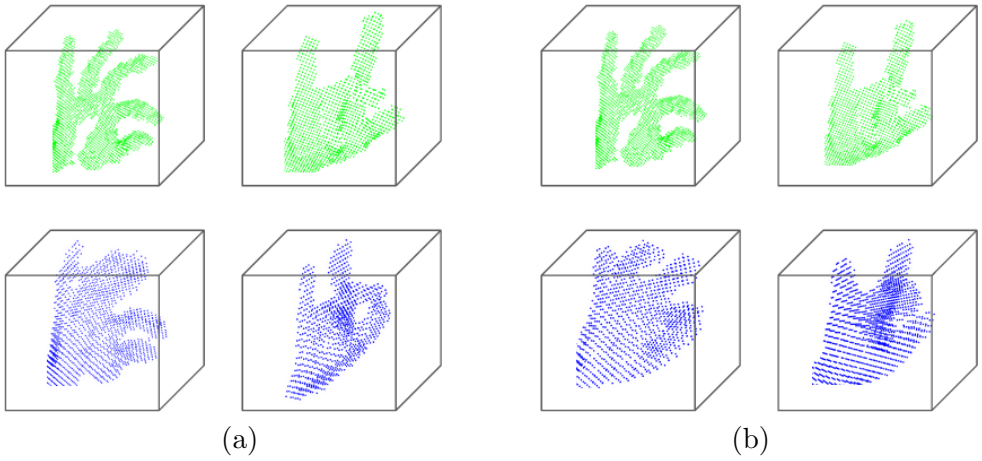


Figure A.8.: *Samples of synthesized voxelized depth maps of NYU [tompson2014real] and BigHand2.2M [yuan2017bighand2] datasets from V2V-SynNet (a) and S2V-SynNet (b). The first and second rows show the ground truth and the reconstructions, respectively.*

($\hat{\mathcal{V}}_S$ and $\hat{\mathcal{V}}'_T$)), and then returns the voxel displacements to register one shape to another. After applying the estimated displacements on the surface shape, we also apply Laplacian smoothing on it to reduce the roughness on the shape, and obtain the final hand shape. We train the DispVoxNet to register the surface shape to the voxelized shape in a supervised manner. However, since the corresponding information is unknown, collecting ground truth displacements between the voxel and surface shapes is difficult. To circumvent this problem, we obtain the displacements between the shape surface and the corresponding ground truth shape in SynHand5M dataset. This is possible because the shape surface generated by V2S-Net preserves the topology and number of mesh vertices during the training.

A.9. NRGA-Based Registration Details

We provide more details related to the modification **M1**. To highlight the role of applying **M1**, we summarize the shape deformation and the optimization scheme of NRGA.

Optimization Method. Given the estimated hand shape surface $\hat{\mathcal{V}}_T$ and the voxelized shape $\hat{\mathcal{V}}_S$, NRGA defines total Gravitational Potential Energy (GPE) of the system as:

$$\mathbf{E}(\mathbf{R},\mathbf{t}) = -\sum_{k=1}^{K}\sum_{j\epsilon\Gamma_k^{\hat{\mathcal{V}}_S}} \frac{\omega_k}{\left(\|\mathbf{R}_k\hat{\mathcal{V}}_T^k + \mathbf{t}_k - \hat{\mathcal{V}}_S^j\| + \epsilon\right)}, \tag{A.13}$$

which is the weighted (ω_k) sum of the inverse of the euclidean distances ($\|.\|$ denotes ℓ_2 norm) between the mesh vertices $\hat{\mathcal{V}}_T = [\hat{\mathcal{V}}_T^1, \hat{\mathcal{V}}_T^2, \cdots, \hat{\mathcal{V}}_T^K]$ and their neighbouring lattice vertices from the voxelized hand $\hat{\mathcal{V}}_S = [\hat{\mathcal{V}}_S^1, \hat{\mathcal{V}}_S^2, \cdots, \hat{\mathcal{V}}_S^M]$.

ID	Layer	Output Sz	Kernel Sz	Stride/Padding	⊕
1	Input	(2) 64x64x64	-	-/-	-
2	3D Conv	(8) 64x64x64	7x7x7	1/3	-
3	LeakyReLU	(8) 64x64x64	-	-/-	-
4	3D MaxPooling	(8) 32x32x32	2x2x2	2/0	-
5	3D Conv	(16) 32x32x32	5x5x5	1/2	-
6	LeakyReLU	(16) 32x32x32	-	-	-
7	3D MaxPooling	(16) 16x16x16	2x2x2	2/0	-
8	3D Conv	(32) 16x16x16	3x3x3	1/1	-
9	LeakyReLU	(32) 16x16x16	-	-/-	-
10	3D MaxPooling	(32) 8x8x8	2x2x2	2/0	-
11	3D Conv	(64) 8x8x8	3x3x3	1/1	-
12	LeakyReLU	(64) 8x8x8	-	-/-	-
13	3D Deconv	(64) 16x16x16	2x2x2	2/0	10
14	3D Deconv	(64) 16x16x16	3x3x3	1/1	-
15	LeakyReLU	(64) 16x16x16	-	-/-	-
16	3D Deconv	(32) 32x32x32	2x2x2	2/0	7
17	3D Deconv	(32)32x32x32	5x5x5	1/2	-
18	LeakyReLU	(32) 32x32x32	-	-/-	-
19	3D Deconv	(16) 64x64x64	2x2x2	2/0	4
20	3D Deconv	(16) 64x64x64	7x7x7	1/3	-
21	LeakyReLU	(16) 64x64x64	-	-/-	-
22	3D Deconv	(3) 64x64x64	3x3x3	1/1	-

Table A.5.: *DispVoxNet architecture details. ⊕ column contains the layer IDs whose outputs are concatenated and used as an input to the current layer. The negative slope for LeakyReLU is set to* 0.01.

From the definition of GPE, the weight ω_k stands as the product of the *gravitational constant* G and the masses of interacting vertex pair $(\hat{\mathcal{V}}_T^k, \hat{\mathcal{V}}_S^j)$. The number of vertices in the template hand shape is fixed to $K = 1193$. On the other hand, only the lattice points with output probabilities ≥ 0.8 are selected to represent $\hat{\mathcal{V}}_S$. This results in varying number of total vertices M in $\hat{\mathcal{V}}_S$ for different input samples. The above energy in eq. (A.13) is minimized to estimate the optimum transformation parameters, K rotations $\mathbf{R} = [\mathbf{R}_1, \cdots, \mathbf{R}_K]$ and translations $\mathbf{t} = [\mathbf{t}_1, \cdots, \mathbf{t}_K]$ for the hand shape vertices. By applying the rigid transformations, $\hat{\mathcal{V}}_T$ is deformed to match with the underlying shape of $\hat{\mathcal{V}}_S$. The GPE residuals in Eq. (A.13) are expressed with NRGA-alike notations. The method requires k-d trees to be built independently on source $\hat{\mathcal{V}}_T$ and target $\hat{\mathcal{V}}_S$ which help to obtain the nearest neighbours of every source vertex $\hat{\mathcal{V}}_T^K$. The number of nearest neighbours are fetched from $\hat{\mathcal{V}}_T^k$ and $\hat{\mathcal{V}}_S$ as a proportion ρ (typically $0.02 - 0.1\%$) of total points. These neighbours form local regions $\Gamma_k^{\hat{\mathcal{V}}_T}$, $\Gamma_k^{\hat{\mathcal{V}}_S}$ respectively. The vertices under the region $\Gamma_k^{\hat{\mathcal{V}}_S}$ appear in the form of multiple lattice corners, whereas vertices under $\Gamma_k^{\hat{\mathcal{V}}_T}$ are ***not*** selected as percentage of nearest neighbours, rather as a set of distinct vertices appearing inside 4-path distance from $\hat{\mathcal{V}}_T^k$ as shown in Figure A.9.

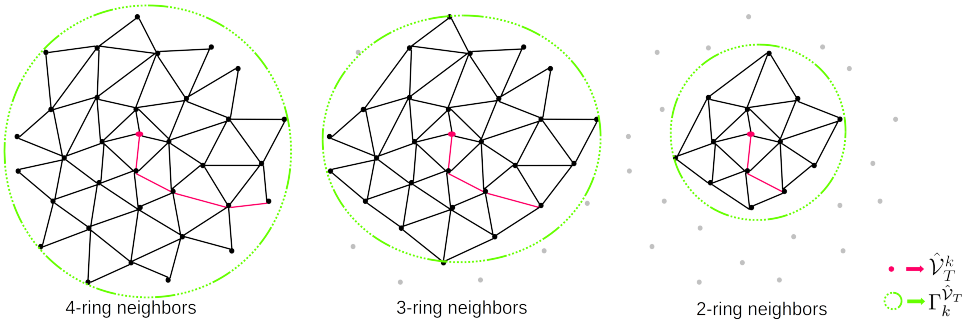


Figure A.9.: *A vertex $\hat{\mathcal{V}}_T^k$ (in red) selects either 4-ring, 3-ring or 2-ring neighbourhood vertices to define a local subspace of our deformable template $\hat{\mathcal{V}}_T$. The vertices (black color) enclosed inside the region $\Gamma_k^{\hat{\mathcal{V}}_T}$ are reachable from $\hat{\mathcal{V}}_T^k$ with the shortest-path-length $\leq n$ for $n-$ ring neighbours.*

Optimization Parameters. We set the parameters of NRGA algorithm used in our HandVoxNet as - **(i)** gravitational constant $G = 0.667$, **(ii)** masses $m(\hat{\mathcal{V}}_T) = 1.0$, $m(\hat{\mathcal{V}}_S) = 1.0$ of all the point vertices in $\hat{\mathcal{V}}_T$, $\hat{\mathcal{V}}_S$, **(iii)** gravitational force softening value $\epsilon = 0.2$, **(iv)** energy dissipation rate $\eta = 0.2$, **(v)** neighbourhood proportion $\rho = 0.02$, **(vi)** time integration step $\Delta t = 0.006$.

Appendix B

Curriculum Vitae

Muhammad Jameel Nawaz Malik

Email: mmalik@rhrk.uni-kl.de

Education

2009–2011 — Master of Science in Electrical Engineering
National University of Sciences and Technology (NUST), Pakistan

2005–2009 — Bachelor of Science in Electrical Engineering
University of Engineering and Technology (UET), Taxila, Pakistan

Professional Experience

12.2016–today — DFKI, Kaiserslautern, Germany
Researcher in the Augmented Vision Department

09.2011–11.2016 — NUST, Pakistan
Lecturer in the Electrical Engineering Department