Imperial College London
Department of Electrial and Electronic Engineering

Pushing the Envelope for Estimating Poses and Actions via Full 3D Reconstruction

Seungryul Baek

Submitted in part fulfilment of the requirements for the degree of
Doctor of Philosophy

Declaration of Originality

This dissertation is submitted to the Department of Electrical and Electronic Engineering, Imperial College London, in fulfillment of the requirements for the degree of Doctor Philosophy. I, Seungryul Baek, hereby declare that this work is my own and includes nothing from the work of others except where specifically indicated in the text. The research presented in this dissertation resulted in the following listed papers with joint authorship as follows:

S. Baek, K. I. Kim, T.-K. Kim, "Pushing the Envelope for RGB-based Dense 3D Hand Pose Estimation via Neural Rendering", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2019,

S. Baek, K. I. Kim, T.-K. Kim, "Augmented skeleton space transfer for depth-based hand pose estimation", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018 **(Oral Presentation)**,

G. Garcia-Hernando, S. Yuan, **S. Baek**, T-K. 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 (CVPR) 2018,

S. Baek, Z. Shi, M. Kawade, T.-K. Kim, "Kinematic-Layout-aware Random Forests for Depth-based Action Recognition", in Proceedings of the British Machine Vision Conference (BMVC) 2017 **(Oral Presentation)**,

S. Baek, K. I. Kim, T.-K. Kim, "Real-time online action detection forests using spatio-temporal contexts", in Proceedings of the IEEE Winter Conference on Applications of Computer Vision (WACV) 2017,

Seungryul Baek, September 2019.

Abstract

Estimating poses and actions of human bodies and hands is an important task in the computer vision community due to its vast applications, including human computer interaction, virtual reality and augmented reality, medical image analysis.

Challenges: There are many in-the-wild challenges in this task (see chapter 1). Among them, in this thesis, we focused on two challenges which could be relieved by incorporating the 3D geometry: **(1) inherent 2D-to-3D ambiguity** driven by the non-linear 2D projection process when capturing 3D objects. **(2) lack of sufficient and quality annotated datasets** due to the high-dimensionality of subjects' attribute space and inherent difficulty in annotating 3D coordinate values.

Contributions: We first tried to jointly tackle the 2D-to-3D ambiguity and insufficient data issues by **(1) explicitly reconstructing 2.5D and 3D samples and use them as new training data to train a pose estimator**. Next, we tried to **(2) encode 3D geometry in the training process of the action recognizer** to reduce the 2D-to-3D ambiguity. In appendix, we proposed a **(3) new hand pose synthetic dataset** that can be used for more complete attribute changes and multi-modal experiments in the future.

Experiments: Throughout experiments, we found interesting facts: (1) 2.5D depth map reconstruction and data augmentation can improve the accuracy of the depth-based hand pose estimation algorithm, (2) 3D mesh reconstruction can be used to generate a new RGB data and it improves the accuracy of RGB-based dense hand pose estimation algorithm, (3) 3D geometry from 3D poses and scene layouts could be successfully utilized to reduce the 2D-to-3D ambiguity in the action recognition problem.

Keywords

pose estimation, action recognition, 3D reconstruction, 3D geometry, human body, human hand, convolutional neural network, deep learning, random forest, real dataset, synthetic dataset, RGB images, depth images, data augmentation, generative adversarial network, 3D mesh, mesh reconstruction from single images, MANO hand model, SMPL body model, discriminative model, generative model, privileged information, transfer learning, latent space.

CONTENTS

List of Figures

List of Tables

ACRONYMS

AR augmented reality. xviii, 1, 64

AUC area under the curve. xviii, 79

CAD60 cornell activity dataset 60. xviii

CNN convolutional neural network. xviii, 4, 5, 10, 16, 18, 19, 21–25, 27, 29–33, 41, 53, 54, 64, 65, 79, 83, 110

CRF conditional random field. xviii

DHPE dense hand pose estimation. xviii, 66, 68, 75, 78, 80, 83

DIP distal inter phalangeal. xviii, 2, 41, 44, 45

DMM depth motion maps. xviii, 29

DO dexter+object dataset. xviii

FPHA first-person hand action benchmark. xviii

FTP Fourier temporal pyramid. xviii, 30

G3D gesture-3D dataset. xviii

GAN generative adversarial network. xviii, 10, 24, 27, 38, 40, 46, 47, 49, 80, 106, 111

GANHAND GANerated hands dataset. xviii

GMM Gaussian mixture model. xviii, 20, 24

GPU graphic processing unit. xviii, 53, 54

HME hand mesh estimator. xviii, 64, 67, 68, 75, 78

HOG histogram of gradients. xviii, 28, 29, 34

HOPC histogram of oriented principal component. xviii, 30

HPD hand Pose discriminator. xviii, 38, 40, 46, 47, 49–51, 55

HPE hand pose estimator. xviii, 38–41, 46–51, 53–55, 57

HPG hand pose generator. xviii, 24, 38, 40, 41, 44, 46–51, 54–57

I3D two-stream inflated 3D convolutional neural network. xviii

ICP iterative closest point. xviii

ICVL Imperial Computer Vision and Learning Lab Hand Pose Dataset. xviii

IOU interaction over union. xviii

KCF kinematic consistency filter. xviii, 100

KLRF kinematic-layout-aware random forest. xviii, 88, 90, 92, 95, 98, 99, 101, 102, 104

KLT Kanade-Lucas-Tomasi. xviii, 29

LSTM long short-term memory network. xviii, 23, 27, 31–33

MANO hand model with articulated and non-rigid deformations. xviii, 2, 15, 19, 24, 67, 68, 70–72, 76, 78, 109, 111, 112, 114, 117, 121

MCP meta carpo phalangeal. xviii, 2, 41, 44, 45, 73

MP moving pose. xviii, 30

MSE mean squared error. xviii

MSRA microsoft research asia. xviii

NTU NTU action recognition database. xviii

NUCLA northwestern-UCLA multiview 3D human action dataset. xviii

NYU New York University Hand Pose Dataset. xviii

OAD online action detection. xviii

OOB out-of-bag. xviii, 96

PCA principal component analysis. xviii, 2, 18, 70, 71, 76, 116, 118

PCK percentage of correct keypoints. xviii, 79

PIP proximal inter phalangeal. xviii, 2, 41, 44, 45

PSO particle swarm optimization. xviii, 16, 17, 24, 25, 81

RF random forest. xviii, 19–21, 24, 32, 34, 90, 95, 97, 98, 101, 102, 104

RHD rendered hand dataset. xviii

RNN recurrent neural network. xviii, 31, 32

SH synthhands Dataset. xviii

SIFT scale-invariant feature transform. xviii, 28

SMPL a skinned multi-person linear model. xviii, 2, 15, 16, 18, 70

STB stereo hand pose tracking benchmark. xviii

STIP spatio-temporal interest points. xviii

SVM support vector machine. xviii, 29

TIP finger tip Joints. xviii, 44, 45

TSDF truncated signed distance function. xviii, 23

UWA3D university of western australia 3D multiview acitivity II dataset. xviii

VR virtual reality. xviii, 1, 64

GLOSSARY

A Appearance feature space extracted from the raw 2D image input.. xviii, 92, 94–96, 98, 100, 102, 104

D_P Hand pose database having images and their skeleton labels.. xviii, 40, 41

K Kinematic-Layout feature space that encodes the 3D geometry.. xviii, 92, 94–104

L 3D scene layout space.. xviii, 92

P 3D human body or hand skeleton space.. xviii, 92

V Depth sequence.. xviii, 92, 94–96, 99, 100

X_C 2D RGB image space.. xviii, 67–69, 73, 75

X_D 2.5D depth image space.. xviii, 40

Y_A Action label space.. xviii, 94

Y_P 3D pose label space.. xviii, 40, 67–69, 72, 73, 75

Ψ Split function of the trained random forests composed of threshold and a specific feature dimension.. xviii, 95–97, 100

$\mathcal{F}^+$ Proposed Kinematic-aware Random forests.. xviii, 95, 98, 99

$\mathcal{F}$ Random forests.. xviii, 95

$\mathcal{Q}$ Quality function of the random forests used to split each node.. xviii

f^A Action recognizer.. xviii, 94, 95

f^P Pose estimator.. xviii, 40, 67

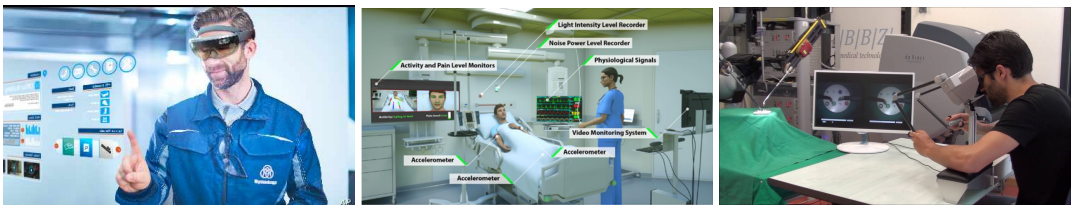
CHAPTER 1

INTRODUCTION

Contents

Understanding poses [18, 50, 70, 148, 242] and actions [5, 8, 49, 50] of human bodies and hands has been an important problem in the computer vision community, as those poses and actions (*e.g.* shaking hands, high fiving, sign languages) are essential to grasp complex daily interactions between persons or intended purposes of manipulating diverse objects. This technique has been crucial for practical products such as smart glasses in augmented reality (AR) and virtual reality (VR) scenario as well as more future-oriented techniques such as AI which assists doctors in monitoring patients, tele-operations and etc. (See Fig. 1.1). However, achieving a good understanding of this is not easy due to many in-the-wild challenges.

1.1 Problem definition

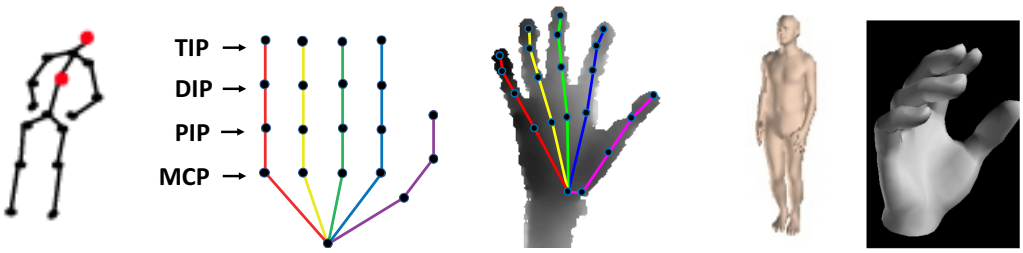


(a) Smart glass (MS Hololens) (b) Patient Monitoring (U. of Florida) (c) Tele-operation (bbzsrl.com)

Figure 1.1: *Diverse products exploiting pose estimation and action recognition of human bodies and hands.*

Pose estimation aims to recover the underlying pose of the subjects (i.e. human bodies and hands) from an unseen test image. Either a depth map or an RGB image can be used as the input, and the pose space can be defined in the form of skeletal keypoints or full 3D meshes of human bodies and hands depending on the applications: For the "hand" pose estimation, we used the 21 joint-based skeletal hand model proposed in [244] (Fig. 1.2b, 1.2c) and deformable 3D hand model, called hand model with articulated and non-rigid deformations (MANO) [161] (See Fig. 1.2e) as the target representation to recover. The hand skeletal model in Fig. 1.2b represents a human hand based on 25 joint angles and the lengths of 20 bones connecting joint pairs: each finger pose is represented as five angles (twist angle, flexion angle, abduction angle for the meta carpo phalangeal (MCP) joint and flexion angles for the distal inter phalangeal (DIP) and proximal inter phalangeal (PIP) joints and four bone lengths. The MANO hand model is composed of 10 and 45-dimenisional vectors as principal component analysis (PCA) coefficients for describing shape and articulation spaces of human hands. It is able to encode hand 3D meshes having 778 vertices and $1,538$ faces. Additional global rotation parameters and camera parameters are required to properly rotate, translate and scale the meshes. For "human body" pose estimation, we used the skeleton model of Fig. 1.2a having a number of 15 joints. We also used the deformable 3D human body model, namely a skinned multi-person linear model (SMPL) [102] (See Fig. 1.2d) having $6,980$ vertices, similarly to the MANO model in the hands domain.

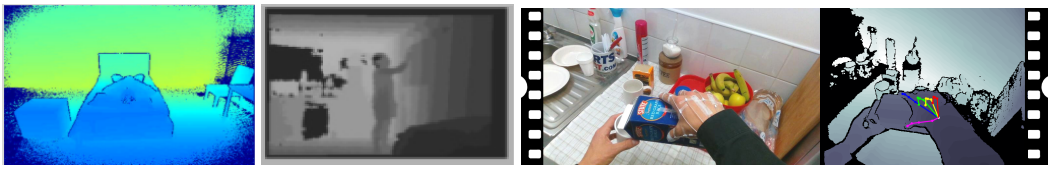
Action recognition aims to recover a proper action label to each video sequence. The actions



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

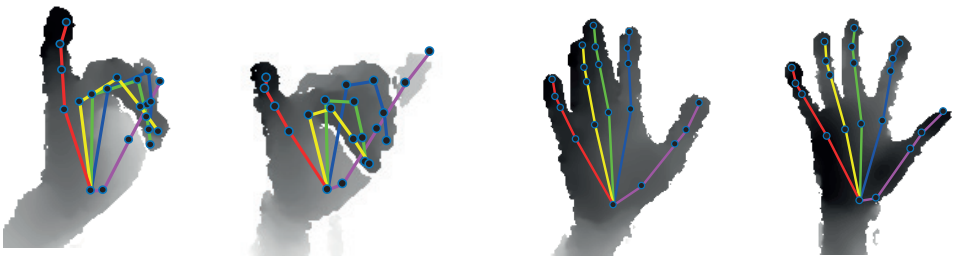
Figure 1.2: *Skeletal model of human and hand models (a) human model which consists of 23 joints [226], (b) hand model which consists of 21 joints [244]: one for wrist and four for each finger. Each finger has five degrees of freedom: flexion for DIP and PIP, flexion, abduction and twist for MCP. (c) a skeleton overlaid on the underlying depth map, (d) SMPL human body model having 6890 vertices, (e) MANO hand model having 778 vertices.*

are pre-defined before constructing the action recognizer. The action label space can be defined in multiple ways: 1) static actions or actions with subtle motions such as "sitting", "lying on the bed" and "drinking" can be the action labels, 2) dynamic actions such as "falling from the bed", "running" and "hopping" can also be action labels, 3) static and dynamic actions are mixed to define higher semantic action labels such as "cooking" and "playing tennis"; and 4) additional objects can be involved and some actions are defined as the combination of *nouns* and *verbs*. "Pouring milk bottles" and "opening milk bottles", for instance, can be action labels. The example plots for the action frames are visualized in the Fig. 1.3.



(a) (b) (c)

Figure 1.3: *The example frames of the actions defined. (a) Example action "lying on the bed" from our PATIENT dataset proposed in Sec. 3, (b) Example action "writing on the board"from CAD60 dataset [191], i.e. (c) Example action in the hand-object interaction scenario involving* Nouns *and* Verbs*, i.e. pouring milk from FPHA dataset [50].*



(a) (b)

Figure 1.4: *Similar hand skeletons overlaid with the corresponding depth maps in (a) camera perspective and (b) shape (subject) variations: Note that slight variations in hand skeletons are instantiated to significantly different depth maps.*

1.2 Challenges

The estimation of actions and poses of human bodies and hands comes with many critical in-the-wild challenges:

Inherent 2D-to-3D ambiguity. Either 2D RGB images or 2.5D depth maps are widely used as inputs to the pose estimation and action recognition systems. These inputs lose full 3D information when they are projected and captured in the form of 2D images. As exemplified in Fig. 1.4a, we visualized similar hand skeletons which were instantiated with significantly different depth maps, indicating the highly non-linear characteristics of the 3D-to-2D projection process. Furthermore, when (self)-occlusions are involved in the 2D images, it is hard to recover the accurate poses of occluded joints. There have been multiple-input convolutional neural network (CNN) structures to deal with the issue in the object domain [188]. However, this is possible only when the multi-viewed camera systems are available during the testing stage which is a different setting from ours (i.e. "estimating actions and poses from the single images"). In the single-image scenario, directly changing depth values can easily generate unrealistic hand shapes as data entries in depth maps are highly structured and correlated. Fig. 1.5 shows the example results with input depth maps rotated by 15, 30, 45 and 60 degrees. There appear many holes and the occluded thumb fingers visualized with the red circle in the rightmost figure could not be recovered in the later stage.

Figure 1.5: *Input depth image (first column) and depth images synthesized by rotating the input (second to last columns: 15, 30, 45, and 60 degrees of rotations, respectively). Depth values are noisy due to missing points in the z-axis. Red circle denotes the missing* Thumb *finger.*

High-dimensionality of the attribute space. Typical parameters for representing human bodies and hands are approximately 100-dimensional. This is already a high dimensional space which is non-trivial to be described by manual efforts. To capture more detailed shapes and textures of human bodies wearing clothes [2], the dimension needs to be much higher. In the hand domain, feature dimensions for shapes and textures can be lower than human bodies, as textures are mostly composed of skin colors and shape variations are not significant when compared to the human body (see Fig. 1.4b, shape variation is mostly parameterized by the bone lengths of the fingers [28, 39]). However, the camera perspective (i.e. viewpoint) space of the hand domain is more complex when compared to that of the human body where subjects are typically isolated, in the frontal viewpoints and upright positions. Hands exhibit frequent and severe self-occlusions; furthermore, defining *canonical views* is not straight-forward since hands are captured in a wide range of equally likely camera perspectives. This high-dimensionality of attribute space makes the pose estimation and subsequent action recognition tasks challenging.

Size variation. The size of the captured human bodies and hands are variable. This variation can make the state-of-the-art CNNs suffer from inferring correct labels, which does not have a principled way of dealing with the scale invariance yet. Naïvely applying the data augmentation methods by generating random re-scaling [126, 128] could be a remedy for this. However, the size space of human bodies (140cm to 200cm in height, 20cm to 100cm in width) and hands (173.5mm to 208mm in lengths, 72.1mm to 97.3mm in width) are inherently wide and could lead to differences in appearance space. This may cause failure in both pose estimation and subsequent action recognition.

Rapid motion. Natural movements of human bodies and hands are fast. Modern consumer depth sensors and RGB cameras only run at a modest frame-rate (less than 100 Hz). This inevitably causes motion blur within frames and reduces consistency between frames. Besides the frame-rate limitation of the sensors, the speed variations between frames make deciding proper temporal scales non-trivial and make the temporal modelling of the actions challenging [41].

Noisy data. Depth maps are often captured in a low resolution involving noises. Therefore, a large gap is observed between synthetic and realistic data [148, 174] as the synthetic data does not involve the natural sensor noise. This is known to affect the pose estimation tasks in several ways: 1) the model trained by pure synthetic data generated from the graphics tools is not easily generalized to the real-world data; this makes it hard to secure a large number of accurate annotations and quality image appearance pairs which are required for learning state-of-the-art CNNs; 2) within the real data, characteristics of the noise are random, and pose estimation often becomes unreliable [12, 132, 150, 241]. In turn, unreliable poses which are used as the intermediate representation could result in the inferior performance in the subsequent tasks such as action recognition.

Subject segmentation. Detecting tight bounding boxes or the exact pixel locations (i.e. segmentation masks) of the subjects is challenging. Several methods for detecting bounding boxes of spatial humans [153, 154, 226] and spatio-temporal human actions (i.e. action tubes [55]) have been proposed so far; still they are regarded as challenging tasks, as many failures are observed when subjects are within the cluttered backgrounds and involved with severe (self)-occlusions. Obtaining clean segmentation masks for human bodies [249] and hands [84] is also hard.

1.3 Our focus and solutions

In this thesis, we try to focus on the fundamental issue, namely the inherent 2D-to-3D ambiguity in the tasks. This challenge is fundamental as all 2D images we are seeing in our daily lives are captured through a non-linear perspective projection that loses the full 3D information.

To relieve the issue, we proposed several ways to incorporate the 3D geometry in action recognizer and pose estimator. While incorporating the 3D geometry in the problem, as a by-product, we could tackle the insufficient data and annotation issues of the pose estimation problem. We detailed targeted challenges and our solutions in the remainder of this section:

2D-to-3D ambiguity. One of the main challenges in 3D pose estimation and action recognition problems is the inherent ambiguity in 2D-to-3D mapping. Generally, either 2D RGB images [18, 70] or 2.5D depth maps [50, 242] are used as the input to the pose estimation and action recognition systems. These inputs are losing full 3D information when they are projected and captured in the form of 2D images, while the 3D information (i.e. occluded parts, appearance taken from multiple viewpoints or cross-view invariance) is often effective for reasoning about actions and poses.

Our solution: To relieve this challenge, we adopted two methods:

- Similar to what happens with the multi-task learning [107, 245] where the trained model is advantaged by the increased number of losses used, we were motivated to incorporate the additional contextual information encoding the 3D geometry to the action recognizer during its training process. This was intended to reduce the 2D-to-3D mapping ambiguities and at the same time to improve the overall action recognition performance.
- We additionally proposed 2.5D depth and 3D mesh reconstruction methods and data augmentation frameworks to generate samples with new viewpoints to explicitly supply our pose estimation model with multi-viewed training data.

Lack of data with quality annotations. Another important challenge is the lack of sufficient training data with quality annotations [6, 127, 244] to train the pose estimator, and the action recognizer. In particular, annotating 3D poses requires a huge amount of manual efforts as one has to figure out 60 to 80 coordinate values in a 3D space for subjects in each image. Even worse is that the depth information is not easy to be secured using the RGB cameras. Manual annotations using depth sensors [185] are proposed to tackle this; however, for RGB images or (self-) occluded regions in depth maps, the 3D coordinate values are not available and it prevents

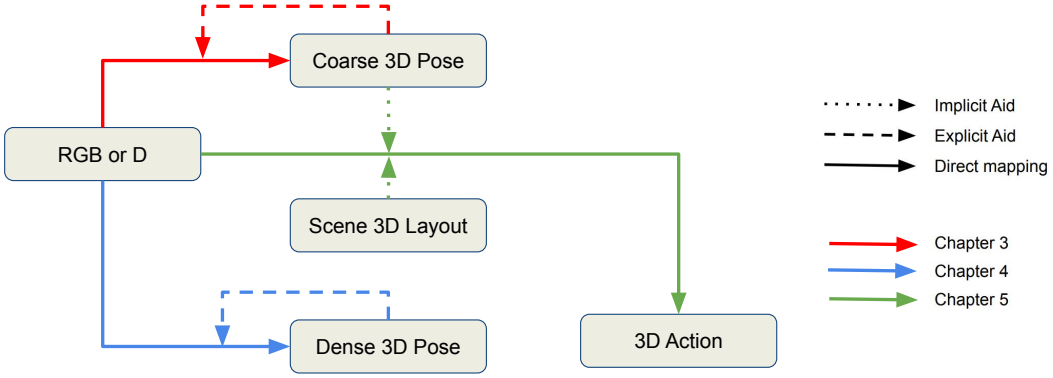


Figure 1.6: *Chapters through 3 to 5 are designed to reveal the relationships between 2D/3D cues available in estimating poses and actions of human bodies and hands.*

collecting high-quality annotations [18, 70, 244]. With the help of motion capture systems using magnetic sensors [242, 244], many datasets have been introduced in pose estimation and action recognition domains recently; however they do not suffice yet to cover all the variety of in-the-wild scenarios, especially when severe occlusions and background clutters [242] are introduced.

Our solution: In a similar spirit to the works in [148, 174, 176] that generate the synthetic databases to create a greater variety of input and annotation pairs, we tried to augment training data with new 2.5D or 3D data samples generated by manipulating their shape and viewpoint parameters.

1.4 Thesis outline and contributions

In chapter 2, we review the literature on pose estimation and action recognition for both hand and human body subjects. We categorize the literature according to their main challenges and corresponding solutions.

We propose methods concerning the two challenges defined in the previous subsection, and demonstrate their effectiveness throughout chapters 3 to 5. In Fig. 1.6, we visualize relationships among available inputs (i.e. 2.5D depth, 2D RGB), 3D geometry (i.e. 3D pose, scene 3D layout)

in action recognition/pose estimation tasks and their usage in each chapter.

The objective of this thesis is to fill the gap between the 2D input and targetted 3D outputs (i.e. 3D poses and actions) with the 3D geometric cues available.

In the pose estimation, the 2D-to-3D ambiguity and insufficient data issues are combined: subtle changes in the viewpoint and shape can result in significantly different 2D images; however obtaining 3D annotations for all the data takes huge efforts. Despite the difficulty, the pose estimation is important as it could affect the accuracy of subsequent tasks such as action recognition.

In the action recognition, 3D geometric cues such as 3D poses and 3D scene layouts are available, and they are useful to reveal some difficult actions (such as lying or cringing humans in a hospital); however there have been few works that can exploit such information. Developing a method to exploit such 3D geometric cues is a promising research direction.

The overall thesis is structured to have primarily two parts: pose estimation and action recognition. In both parts, **challenges are lie in the 2D-to-3D mapping ambiguity which makes the 3D pose estimation difficult**: 1) Hand pose estimation is challenging due to its significant appearance change. Hands exhibit severe viewpoint variations, differently to human bodies and faces which are mostly in the upright position and frontal viewpoint. Also, data collection is challenging due to many self-occlusions involved. 2) In the PATIENT action recognition problem, where humans are not in the upright position and frontal viewpoint, body pose estimation becomes non-trivial while estimating 3D poses and scene layouts is helpful for revealing their actions. We tackle the challenges by **exploiting the 3D geometry during the training stage of classifiers**.

1) First two chapters (i.e. chapter 3, 4) deal with the "pose estimation" of hands that exhibits severe viewpoint variations. In the hand domain, there is no complete database yet covering up the combination of viewpoint and other major variations (i.e. shape, pose). We try to reconstruct 2.5D depth maps (in chapter 3) or 3D meshes (in chapter 4) and use them as the additional training data while learning the mapping between 2D inputs and 3D poses.

2) The last chapter (i.e. chapter 5) deals with the "action recognition" of human bodies under the scenario of 24/7 monitoring patient's actions. Estimating accurate patient's body poses with the off-the-shelf pose estimator (e.g. Microsoft Kinect) is difficult, due to the unique and diverse postures of patients (e.g. lying, cringing). Also, 3D scene layouts such as bed and floor are useful to reveal the actions while estimating them at testing stage is non-trivial. We bypass the explicit estimation of body poses and scene layouts by developing the random forest (RF)-based framework that can exploit additional information only during the training stage (where ground-truths could be utilized). Using the framework, we incorporate the 3D geometry from both 3D scene layouts and body joints in our model.

In chapter 3, we tackle the problem of both 2D-to-3D ambiguities and insufficient 3D data issues for the 3D hand pose estimation problem (in the aspects of articulation, shape and viewpoints). We synthesize the new 2.5D samples using the generative adversarial network (GAN) framework with cyclic consistency losses to explicitly generate the multi-view information lost during the 2D projection process. At the same time, we synthesize the new samples in the shape space as well to enforce pose estimators see diverse samples in another attribute. Using this 2.5D reconstruction method, we can augment the training data and demonstrate that the coarse skeleton-based hand pose estimator achieved the state-of-the-art performance.

Related publication

S. Baek, K. I. Kim, T.-K. Kim, "Augmented Skeleton Space Transfer for Depth-based Hand Pose Estimation", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2018 **(Oral Presentation)**.

In chapter 4, we try to reconstruct the full 3D meshes of human hands. As there is no dataset in the hand domain which has RGB and corresponding full 3D mesh annotation pairs. To tackle this, we formulate our CNN-based framework to be supervised by the RGB image, 3D skeleton and 2D segmentation mask triplets using a differentiable renderer and a skeleton regressor, rather than using the explicit RGB and 3D mesh pairs. Furthermore, to tackle the insufficient 3D data and 2D-to-3D ambiguity, we proposed to generate new RGB samples from intermediate

3D meshes estimated, by manipulating their shape and viewpoint parameters. We demonstrated that the proposed method can obtain the state-of-the-art performance in RGB-based hand pose estimation benchmarks.

Related publication

S. Baek, K. I. Kim, T.-K. Kim, "Pushing the Envelope for RGB-based Dense 3D Hand Pose Estimation via Neural Rendering", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) 2019.

In chapter 5, we investigate the use of the 3D geometry information when classifying patient actions in a ward scenario. We utilize 3D geometry composed of 3D layouts and 3D human poses and propose the method to implicitly reduce the 2D-to-3D ambiguities during the training stage. As a result, overall action recognition accuracy has been improved accordingly. Considering that estimating the 3D geometry cue (*e.g.* distance between layout planes to the human body skeletons) itself is challenging during the testing time, we decided to use them only at the training stage, where ground-truths could be used, while only camera input was used during testing.

Related publication

S. Baek, Z. Shi, M. Kawade, T.-K. Kim, "Kinematic-Layout-aware Random Forests for Depth-based Action Recognition", in Proceedings of the British Machine Vision Conference (BMVC) 2017 **(Oral Presentation)**.

Finally, chapter 6 concludes the thesis.

In appendix, we propose a new synthetic hand pose database by fitting the deformable 3D hand model to one of the biggest database in the hand domain. We expect it would be interesting to see the possibility of more diverse data augmentation possibility and multi-modal fusions using this database in the future.

CHAPTER 2

BACKGROUND

Contents

This chapter reviews previous literature on pose estimation and action recognition. Problems in understanding the 3D poses and actions of humans have been tackled by breaking them down into parts – human body, hand and face – and developing domain-specific approaches, separately. In this chapter, we present their challenges and methods focusing on human bodies and hands.

2.1 Pose estimation

Human bodies, hands and faces are major subjects in images and videos that we obtain in our daily lives. Understanding their poses is often critical for performing subsequent tasks

(e.g. action recognition, scene understanding or predicting future movement). There has been significant progress in estimating "2D poses" of human bodies, hands and faces [20]. The "2D poses" are defined based on the uv-coordinate space of the 2D images, disregarding the z-axis. However, our interactions with the world happen in the 3D space and more recent works have made progress towards estimating "3D poses" which are defined in the world coordinate (xyz) from a single RGB image or a depth map [14].

2.1.1 Human 3D model

3D models are useful tools for capturing 3D poses and shapes of articulated objects. The 3D model of the human is proposed by breaking it down into individual parts: mainly human body, hand and face.

Body. Three-dimensional body scanners enabled capturing detailed human body shapes. For example, the CAESAR dataset [156] has been collected for modelling human body shapes. Anguelov et al. [4] proposed the SCAPE model, which can deform according to articulated poses and shapes using data-driven approaches. Subsequent models followed the scheme, using either triangle deformations [62] or vertex-based displacements [102]. These 3D models [3, 4, 102], however, focus on capturing the overall shape and pose of the body, excluding hands and face (these methods assume that hands are clenched as fists and faces have a neutral expression). However, to properly understand diverse human behaviors, we must capture more than the body pose.

Hand. From the perspective of machine learning, both hand and body pose estimation could be formulated as regression problems and they share many similarities. Hand poses, however, exhibit domain-specific challenges:

Large variations in camera viewpoints. Hand pose estimation requires generally much more data than is required in the human body pose domain where most humans are upright position and seen from frontal viewpoints. Combined with the diverse viewpoints, hands exhibit many

more self-occlusions than human bodies, and this makes it hard to collect abundant databases with quality 3D annotations [244]. In chapter 3 and chapter 4, we will tackle this issue by synthesizing samples with new attributes (i.e. viewpoint, shape).

Delay in the development of a high-quality 3D hand model. Until recently, 3D hand models were typically artist designed [187] based on primitive shapes (i.e. cylinders, spheres, ellipsoids) [113] or simple shape spaces [201]. As hand poses exhibit many self-occlusions, in [9], a multi-view camera system was employed to improve the performnace of 3D hand capture; however these approaches were based on the fixed shape. Khamis et al. [83] proposed developing a 3D model captured from 50 peoples' depth maps, to deal with shape variations. Recently, following the formulation of the popular 3D body model, SMPL [102], Romero et al. [161] proposed the high-quality parametric 3D hand model (MANO) with shape and pose parameters. The model was developed by capturing high-quality 3D scans of 31 subjects in 51 different poses. In chapter 4, we exploited this MANO 3D hand model to reconstruct the dense representation of hands from single RGB images.

Face. Blanz and Vetter [11] pioneered developing 3D face model that could model shapes and the albedo of the human faces. Since that time, numerous methods have been proposed to capture face shapes and expressions from scanned data [17]. FLAME [95] tried to model the whole head by capturing rotations of the head and neck regions. Recently, statistical face models constructed from large-scale scans have been made publicly available [15, 29].

Unified models. There have also been recent efforts at combining 3D models for the human body, hand and face. Such combined pose estimation was first proposed in [76]. Joo et al. [76] offered a Frank model by combining three different available 3D models: SMPL [102] for the body model, an artist-created rig for the hand model and the FaceWarehouse [19] for the face model. They exploited the multi-view camera to reconstruct full 3D meshes of the human. Romero et al. [161] also combined body and hand models in a model called "SMPL+H". The model was created by stitching the SMPL body model with the proposed MANO [161] 3D hand model. Very recently, Xiang et al. [232] proposed extending the Frank model [76] for exploiting only single-view RGB cameras. Pavlakos et al. [138] proposed a similar framework

that also incorporated the additional face 3D model from "SMPL+H". Hasson et al. [63] proposed reconstructing hand and object 3D meshes together from hand-object interaction scenarios. Developing the unified models for human body, hand and face is a promising future direction, however, the completeness of current algorithms is yet inferior to the domain-specific approaches.

2.1.2 Dense pose estimation (3D model fitting methods)

The generative approaches, also known as "analysis by synthesis" are widely used in the literature to fit the 3D model to the inputs. The particle swarm optimization (PSO) is a popular algorithm for this purpose. It is able to fit 3D models to point clouds (depth maps) by minimizing the discrepancy. Gradient-based optimization methods are also used to solve it as the energy minimization problem [155, 224]. To fit to RGB images, recent methods try to exploit 2D estimations (e.g. 2D skeletons, segmentation masks) as the intermediate representation: Loper et al. [101] proposed a method to fit SCAPE [4] from motion captured skeletons. Using the SMPL body model with its accompanying 3D skeleton regressor (that geometrically maps 3D vertices to skeletons), Bogo et al. [14] proposed fitting SMPL model to the 2D skeletons estimated by [144]. The fitting is iteratively performed by minimizing the discrepancy between the skeleton obtained from SMPL model and that from [144]. Zhou et al. [251] and Hasler et al. [61] fit their 3D models to the segmentation masks obtained by the GrabCut. Panteleris et al. [133] take the similar approach in the hand pose estimation domain. They first estimate 2D skeletons from RGB images using [226] and fit their 3D hand model to estimated 2D skeletons. Joo et al. [76] also formulate energy terms considering the 2D skeleton consistency and kinematic priors, then minimize it to capture face, body and hands together using their Frank model. More recently, CNNs are used to infer 3D model parameters from single RGB images. CNNs are trained by full 3D supervision [57] or indirect weak supervision [77] using the differentiable renderer [79]. In the remainder of this subsection, we will give more detailed review for PSO-based and CNN-based works.

Particle swarm optimization. Particle swarm optimization (PSO) is a stochastic optimization algorithm introduced by Kennedy and Eberhart [81]. In PSO, multiple hypotheses are generated and randomly initialized; then the discrepancy between the manipulated 3D model hypotheses and input depth map is measured, and hypotheses communicate together to update for the global minimum. It is proven to achieve the global minimum. The PSO has been successfully applied to both human body [74] and hand pose estimation [130] works. They both used the primitive shape-based 3D models. Qian et al. [147] additionally combine the iterated closest point (ICP) method with PSO, to compensate for the heuristics of the PSO. PSO variants have also been proposed to improve the method's convergence properties and to tackle shape variations:

Non-convexity of objective function. One criticism of the generative approaches is that optimization is often based on non-convex objective functions and this results in the high time complexity and frequent local minimas. To tackle this, the concept of "partial PSO" or breaking the overall parameter space into parts has been proposed [136, 145, 168]. Sharifi et al. [168] adopted a 3D hand model having 45-dimensional parameters; but divided into 6-dimensional space. Park et al. [136] adopted the 26-dimensional 3D hand model and divided the space into 6-dimensional space. Poier et al. [145] proposed a two-staged PSO algorithm: first, the global location and viewpoint of hands are optimized by PSO, then each finger's pose is optimized by five instances of PSO.

Towards shape variations. Most generative methods do not consider shape variations. Therefore, hand shape was calibrated before running the algorithm. In the early hand pose estimation, the hand calibration was manually performed. Hu et al. [67] proposed using color LED markers to calibrate the hand model. Lian [99] proposed equipping markers on hands and developed the inverse kinematic solution for the hand shape calibration. Melax et al. [113] proposed a framework that requires the manual adjustment of hand shapes (e.g. finger length, palm width and so on). Tylor et al. [83] proposed reducing the number of parameters by transforming the parameter space of a 3D hand mesh model into a few bases. Qian et al. [147] proposed adopting an approximate ball model to improve the run-time speed as well as the fitting accuracy. Recently, Tkach et al. [155] tackle the inefficiency by adopting the efficient sphere-mesh model

and proposing the low dimensional calibration method. In the following work [203], Tkach et al. proposed the online method for hand personalization using the sphere-mesh model [202]. In chapter 3 and 4, we also involved the shape variations: In chapter 3, the hand shape variation is simplified to the bone length changes based on the research on the hand dimensions [28, 39] and in chapter 4, hand shapes are captured by the PCA for 3D mesh surfaces to encode more complicated shape variations (e.g. thin and fat).

CNNs and differentiable renderer. Recently, papers for reconstructing the 3D model of human bodies based on CNNs have been appeared: Güler et al. [57] proposed a way to reconstruct full 3D representations of human bodies using CNNs from single RGB images. This method, however, requires full 3D supervisions using many pairs of 2D images and 3D mesh annotations. End-to-end CNN methods for reconstructing the full 3D model (parameterized by SMPL) while not exploiting the full 3D annotation have been proposed for human bodies [77, 209]. Both works [77, 209] exhibit the similar method to learn the nonlinear mapping between 3D meshes and RGB inputs. In [77, 209], the SMPL layer is formulated to generate the 3D human body meshes from SMPL parameters in the differentiable manner. Furthermore, a skeleton regressor that geometrically maps vertices of SMPL to pre-defined body skeletons is provided by [102]. As both the SMPL layer and the skeleton regressor are differentiable and they could be embed in the CNNs. In overall, CNNs are formulated to first infer the SMPL parameters and generate 3D mesh vertices via the SMPL layer. Then, the intermediate 3D mesh vertices are used to generate skeletons via the skeleton regressor. At the training stage, the output 2D skeletons are compared to 2D skeleton ground-truths and loss is back-propagated through the network to update CNN parameters. At testing, the intermediate 3D mesh is regarded as the output. In [209], an additional testing-stage refinement within a CNN architecture is shown effective.

Differentiable renderer. With the aid of the general purpose differentiable renderers [79,100,103], it becomes easier to map image pixel values to the vertices of the 3D meshes, without needing the full 3D supervisions. These renderers have been proposed to generate 2D silhouettes, depth maps and even textured RGB maps without losing the differentiable characteristic. However, as the inverse rendering is not generally differentiable, approximation is done to keep the

backward pass differentiable [79, 103]. A probabilistic approach is explored in [100] to bypass the approximation. Prior to this, Mansinghka et al. [110] formulate the rendering process with the generative model and try to infer parameters of the scene model from the observations using Metropolis-Hastings sampler. Li et al. [96] also introduced a general-purpose differentiable ray tracer that uses a Dirac delta function-based novel edge sampling algorithm and an efficient importance sampling based on spatial hierarchies.

In chapter 4, we use the skeleton estimator that maps vertices of MANO 3D hand model to 21 hand skeletons and differentiable renderer of [79] to generate 3D skeletons and 2D segmentation masks from the 3D hand meshes. Then, the overall CNNs are supervised by RGB, 3D skeleton and 2D segmentation mask triplets which are available in conventional hand pose datasets.

2.1.3 Skeleton-based coarse pose estimation

Skeleton is defined as the vector concatenating xyz coordinate values of the key points that are pre-defined for the subject (e.g. hand, human body or face). While it is coarse and loses rich information about the subject; it is efficient and can describe key components of the subject. Even temporal actions of subjects can be effectively captured [236].

Discriminative approaches directly learn a posterior probability of mapping the visual features to the targetted poses. They have been successfully adopted in the skeleton estimation tasks. Algorithms require a labelled training database for learning such a non-linear mapping between visual features and poses. In earlier works, nearest neighbour classifiers were used to retrieve samples in the training data [160, 224]. Recently, random forest (RF)-based and CNN-based frameworks have been popularly used.

Random forest-based works. Shotton et al. [171] proposed a RF-based human body skeleton tacker operating in real-time. The approach treats 3D body skeleton estimation as a classification problem to use the RFs. The authors applied RFs for per-pixel classification of body parts and applied a mean-seeking algorithm afterwards, to output the final human body parts. Girshick et

al. [54] improved it by proposing a regression forest that is able to vote for occluded skeletons from each pixel. Several works in facial landmark estimation [31, 210] exploited similar voting-based approaches in their task. Then, Sun et al. [190] proposed a hierarchical approach by extending the work of Girshick et al. [54] to regress skeletal joints in a coarse-to-fine manner. The performance was acceptable for even practical products and this accelerated the use of human body poses as useful cues for subsequent tasks such as action recognition [45, 108, 235]. In the hand skeleton estimation, similar RF-based approaches have been applied, following the trends in human body skeleton estimation research [54, 171, 189]: a classification-based RF framework similar to [171] has been developed in [185, 196], followed by a regression-based RF framework similar to [54] in [215].

Encoding the kinematic prior in RFs. As discriminative approaches lack the ability to encode the kinematic information from physical objects, their results are sometimes physically implausible. To relieve this, several works have tried to incorporate such priors in their classifiers for encoding prior kinematic information. Ye et al. [237] proposed a method with kinematic chain structures to encode kinematic constrains into the classifier. Ding et al. [33] proposed template-fitting using the Gaussian mixture model (GMM)s. To encode the kinematic context information from hands, Zhu et al. [254] proposed to encode contextual information into the RFs.

Hierarchical approach in RFs. The structure of RFs are by nature hierarchically driven. Several works exploit this characteristic to perform pose estimation in the coarse-to-fine manner. Rogez et al. [157] proposed a unified framework for hand pose detection and classification. A more hierarchical method has been developed in [158] to perform the coarse-level part classification and the detailed pose-specific part classification. Tang et al. [194] presented the latent regression forest (LRF) for real-time 3D hand pose estimation. When growing trees, they exploted the information from kinematic graphs which are separately pre-learned from RFs and encode kinematic priors of hands. Keskin et al. [82] proposed dealing with different hand shapes involving multiple RFs. They adopted the RF-based framework of Shotton et al. [171]; however they cluster training data into K clusters depending on different hand shapes. A shape classification RF is trained using whole training data and for each cluster, part classification

forest is trained to predict hand joints. Li et al. [94] proposed segmentation index points (SIPs) on top of Tang et al.'s work [194] to deal with different topologies of hands. Sun et al. [190] proposed a hierarchical RF-based framework that first estimates coarse hand poses and then estimates detailed finger positions. Tang et al. [195] proposed involving multiple RFs to involve kinematic hierarchy in hand poses. They assign separate multiple RFs for each partial joints (from palm to finger tips) and RFs are trained to infer partial hand poses sequentially. Wan et al. [215] explored the hierarchical RF-based framework of [190]; however they explored a new feature constituted with "normal difference", rather than conventional "pixel difference".

Deep learning-based works. CNNs [64, 91, 178] are widely used in many computer vision problems, due to their state-of-the-art performance. In the pose estimation, pre-trained networks [64] are fine-tuned to pose estimation domains [16] or new network architectures suitable for 3D pose estimation (i.e. iterative inference for incorporating contextual information) have been proposed [226].

Depth-based 3D skeleton estimation. The problem is better set in the depth domain, compared to RGB counter-parts as the depth input inherently offers the partial 3D information [196]. Furthermore, deep learning methods have been successfully applied along with the million-scale hand pose dataset [244] proposed in this domain. In the depth domain, as the magnetic sensors are not visible, large-scale real data collection with quality 3D annotation is possible by equipping sensors on hands as in [244], while in the RGB domain, it is impossible as the 2D images would be spoiled by the sensors. Exploiting the large-scale public depth image datasets (e.g. ICVL [194], MSRA [190], NYU [205], BigHand2.2M [244], FPAH [50]), CNNs have shown good accuracy. Recent work [242] analyzed the state-of-the-art approaches and reported that mean 3D joint error for their results was around 10mm, which is quite small given the normal scale of human hands (300mm).

Due to the difference between depth maps and RGBs (e.g. number of channels and etc.), researchers have designed several new architectures for the depth domain: Oberweger et al. propose the DeepPrior arhictecture [128] and more deeper version of DeepPrior++ architecture [126] adopting the ResNet architecture [64]. Zhou et al. [252] proposed a network that

estimates joint angles of hands, which are fed into a forward kinematic layer to estimate the hand joints. Ye et al. [238] proposed the CNN architecture solving multiple scales and having spatial attention mechanism within a network. In chapter 3, we experimented with the CNN architecture proposed in Ye et al.'s work [238].

RGB-based 2D skeleton estimation. The process of recovering 2D skeletons from RGB images are well established as the regression problem. In the earlier work, Toshev et al. [207] proposed a CNN-based human body skeleton regressor, named "DeepPose" by directly regressing the coordinate values of each skeletal joint. After this, many CNN-based skeleton regression methods have been proposed: In [72], Jain et al. proposed a hierarchical CNN-based network that combines low-level and high-level spatial models. Tompson et al. [206] proposed a CNN-based network considering the relationship among body parts in a spirit similar to the Markov random fields. Pfister et al. [142] further developed heat map-based outputs and the effectiveness of the larger receptive fields. Wei et al. [226] then proposed a more robust CNN architecture that iteratively improves the heat-map prediction tasks using global contextual information. Due to vanishing gradients, they proposed to supervise all the heat-maps. Newell et al. [120] again verified the effects of intermediate supervisions in their architecture. 2D skeleton estimation is matured in the level of estimating multiple persons' poses for the human body pose: unstructured pairwise relationships between body parts of variable numbers of people have been captured using part affinity fields [20]. Pose residual networks [88] combining detection and keypoint estimation have been proposed and dense regression from keypoint candidates has been proposed in [124] to improve pose estimation tasks that use multiple persons.

Wei et al. [226]'s network architecture have been adopted to the hand skeleton estimation domain by the work of [256] and have established the baseline for 2D hand skeleton estimation. While in the hand pose estimation domain, hand appearances are varied by the diverse viewpoints it could have, real RGB data is scarce and automatic annotation of RGB images still remains challenging. Simon et al. [176] achieved promising results by mixing real data with a large amount of synthetic data. Further accuracy improvement has been obtained by the automatic annotation using label consistency in a multiple-camera studio [75]. However, 3D skeleton

estimation is more desirable.

RGB-based 3D skeleton estimation. Three-dimensional skeleton estimation from RGB images has inherent uncertainties, as the input RGB image does not provide any clues for depth estimation. In the early stages, Chen and Ramamnan [24] proposed estimating 3D human body skeletons from 2D skeleton estimation results using a nearest neighbor method designed to search for the closest 3D skeletons that match current 2D skeletons. Martinez et al. [111] proposed a simple and efficient CNN-based approach to lift 2D human body joints to the 3D space. Nie et al. [122] used the long short-term memory network (LSTM) network to predict the depth of 2D skeleton locations to estimate 3D coordinate values. Recent approaches have started to fuse 2D and 3D skeletal results iteratively [199]. In [204], 3D skeletal joints are refined by seeing 2D skeletal joints. Similarly, in the hand pose estimation domain, earlier work [256] attempted to learn direct mapping from RGB images to 3D skeletons. Recent methods [18, 70] have shown state-of-the-art accuracy by implicitly reconstructing depth images – that is 2.5D representations– and estimating the 3D skeleton based on them.

In chapter 4, we proposed a method for estimating full 3D meshes from single RGB images. From obtained 3D meshes, we can uniquely generate both 3D skeletons and 2D segmentation masks. Via this method, we compared the proposed algorithm with several 3D skeleton estimation algorithms and concluded that our method has obtained the state-of-the-art accuracy in the RGB-based 3D skeleton estimation task.

Deep learning architecture for 3D operation. As mentioned previously, hand poses exhibit a much wider scope of viewpoints, given an articulation and shape. It is hard to define a canonical viewpoint (e.g. a frontal view). To address the issue, intermediate 3D representation is used to properly represent 3D hand poses. The adopted intermediate 3D representation is for example, projected point clouds [52], D-truncated signed distance function (TSDF)s [53], voxels [116] or point sets [51]. Moon et al. [116] proposed a 3D CNN composed of encoder-decoder architecture to estimate per-voxel likelihoods for each hand joint. Ge et al. [52] projected the depth image onto three orthogonal planes (i.e. $x-y$, $y-z$ and $z-x$) and trained a 2D CNN for each projection, then fused the results. In [53], Ge et al. proposed a 3D CNN by replacing 2D

projections with a 3D volumetric representation (projective D-TSDF volumes). In [51], Ge et al. proposed using the PointNet [146] in the hand pose estimation domain, which is able to directly exploit the point clouds as the input. Though these approaches have shown promising results, they are basically limited in that they cannot re-generate lost 3D information of depth maps. For example, as in Fig. 1.5, thumb fingers are occluded in the depth map and they cannot be exploited in those approaches.

In chapter 3 and 4, we try to re-generate lost 3D information (self-occluded) employing the GAN framework and full 3D mesh (MANO), respectively.

2.1.4 Pose estimation with the refinement

As discriminative methods such as random forests (RFs), convolutional neural networks (CNNs) and generative methods such as particle swarm optimization (PSO) have their own pros. and cons., the hybrid approach that combines both are promising. The popular way of combining the two methods is performing the initialization with the discriminative method and refining it with the generative method. For example, Qian et al. [147] use fingertip detection results as initialization for PSO. Tompson et al. [205] apply a CNN to predict 14 joint positions as an initialization and apply PSO to refine them. Sridhar et al. [184] use RFs to classify pixels into parts and apply an optimization method for solving energies defined to consider the consistency. Sharp et al. [169] also take the similar strategy with PSO, however estimate joint angles rather than the joint locations. To do this, they first classify the input into one of 7 discretized poses (open, fist and so on) and then apply PSO. Sun et al. [190] propose to use a sequence of weak regressors that are trained to output residuals to be added to the current pose estimation result to make it close to their ground-truths. They iteratively refine the once estimated joints by first improving palm joints and the finger joints afterwards. In this work, palm joints are regarded as the global pose while finger joints are regarded and finer poses. Oberweger et al. [129] proposed guiding the iterative refinement by the hand pose generator (HPG) that generates depth images of hand poses given skeletons. Wu et al.'s algorithm constructs a skeletal GMM which acts as a

prior. The initial pose estimates are then refined by combining the prior with 2D re-projection and temporal consistency likelihood [228].

In both chapter 3 and chapter 4, we involved the testing refinement step for once estimated hand poses using the gradients for minimizing the losses at the testing step. In chapter 3, the loss is defined by the hand pose discriminator which is learned to capture the hand pose distribution and thus, hand poses are updated to be in the hand pose distribution. In chapter 4, the loss is defined by the discrepancy between the estimated final 3D meshes and estimated 2D evidences, so that 3D hand meshes are updated to match well to the 2D evidences. This is motivated by the fact that 2D evidences are better estimated than the 3D meshes as there is an inherent 2D-to-3D ambiguity.

2.1.5 RGB-D hand pose estimation datasets

The performance of discriminative approaches is highly correlated to the quality and size of the training data where it was trained on. A large-scale dataset with a good 3D annotation is desirable to learn a correct mapping between input and output 3D poses; however, collecting the large-scale real databases with quality 3D annotation is challenging and still an on-going research problem. However, to properly train CNNs, a large-scale database is required. Several large-scale databases have been proposed in both depth and RGB domains.

Depth dataset. In early works, the datasets (e.g. ICVL [194], MSRA [190], NYU [205]) were collected by the combination of model fitting methods such as PSO, use of tracking algorithm for temporal propagation and manual refinement. Therefore, the obtained 3D annotation is noisy and inaccurate. The automatic data collection pipeline has been developed afterwards. Yuan et al. [244] equipped magnetic sensors on hands and applied the inverse kinematics to reconstruct hand skeleton locations from obtained sensor locations. They used the reconstructed hand skeletons as the ground-truths. Furthermore, they scheduled to explore complete articulation transitions between 32 extreme hand poses (i.e. cases obtained by extremely folding or stretching each finger) with random viewpoint changes and involving 10 persons. This dataset is called as

the Big Hand 2.2M dataset [244] and is regarded as one of the biggest dataset in the aspect of articulation, viewpoint and shape variations in the depth-based hand pose estimation community. Yuan et al.'s automatic annotation was possible as the equipped magnetic sensors are not visible in the captured depth maps; however it is impossible to apply this to RGB domain as the equipped magnetic sensors could spoil the RGB appearance. Using the same skill, the FPAH [50] dataset that deals with the challenging scenario, the egocentric hand pose estimation, is proposed. The scenario is challenging as in the egocentric viewpoint, hand poses exhibit much more self-occlusions than in the third viewpoint. Also, in the scenario, the hand-object interaction was involved and hands are also occluded by the objects. However, even for hands exhibiting many (self-)occlusions, the automatic annotation works successfully.

Obtaining the *complete* realistic database in the combination of main variations of hand poses (i.e. viewpoints, poses and shapes) is still challenging due to the high dimensional space of the hand subject. In chapter 3, we tackle this challenge by synthesizing new 2.5D samples in the aspect of viewpoint and shape variations.

RGB dataset. The dataset issue is more imminent in the RGB domain as the automatic data collection pipeline such as Yuan et al.'s [244] cannot be applied, as the magnetic sensors involved could spoil the input appearance. However, several datasets have recently been proposed such as DO [186], STB [248], RHD [256], SH [118] and GANerated [117] datasets. Due to the difficulty in the annotation, most available datasets such as GANerated, RHD and SH are synthetic. They are rendered from the 3D model with simulated textures. DO [186] and STB [248] are realistic datasets; however the number of collected frames are small, as they were annotated based on the manual efforts.

The gap between real and synthetic data. Synthetic data are widely used as it is easy to obtain both appearance and quality annotations even in the 3D space. Thus, many state-of-the-art 3D pose estimation methods employ the 3D model for human bodies and hands to augment their training data space. Both direct data augmentation [117] and pre-training on synthetic data [176] were proposed. One critical issue with synthetic datasets is that there is a gap between real and synthetic data. It is well known that a model trained on pure synthetic data does not generalized

well to the real testing data [148, 174]. To reduce the gap, Shrivastava et al. [174] proposed the conditional GAN framework that inputs the pure synthetic depth maps and is trained to improve its realism using the adversarial loss. Mueller et al. [117] also proposed similar method for the RGB domain using the conditional GAN framework. While images obtained by the GAN framework is appealing; sometimes it is not optimal for the subsequent pose estimation task. To relieve this, in [148], Rad et al. proposed to close the distance between the real and synthetic feature spaces rather than the 2D image appearance.

2.2 Action recognition

Action recognition of human bodies and hand gestures has a long history. We will first review the available cues in the action recognition problem and review methods and database.

2.2.1 Various cues for action recognition

In action recognition, the spatial and temporal cues have been combined: Spatial cue captures the static appearance information of single frames while temporal cue conveys the movement of the observee or objects in the form of motion across frames. It includes low-level features that are computed over the spatio-temporal volume of the human action [34] or higher-level representations, such as body pose [217] or 3D scene layouts [32].

RGB cue. RGB images are the most popular inputs to the action recognition systems, as they can be trivially obtained from the usual video camera. Furthermore, many deep learning architectures, which was originally developed for the image classification such as AlexNet [91], VGG-16 [178] and ResNet [64], could be extended towards spatio-temporal feature extractors via pre-trained weights. One of the traditional methods of encoding spatio-temporal information using RGB streams is via applying long short term memory network (LSTM) [119] on these CNN features. Two stream networks [42, 177] also have been popularly used. Simonyan et

al. [177] combined two streamed features extracted from temporal (by optical flows) and spatial (by RGBs) inputs and obtained the promising results. Feichtenhofer et al. [42] further explored operations for combining the spatial and temporal streams and obtained the state-of-the-art performance using the convolutional operation.

Before deep learning. Prior to the deep learning, the hand-crafted features were extracted and encoded as the spatio-temporal feature vectors. For example, Bobick and Davis [13] constructed the action templates as the major representation that summarizes the history of motion from humans. Efros et al. [38] proposed a motion descriptor based on the optical flow that is computed over the human movements. Then, the nearest neighbor classifier is applied to classify the obtained representation according to action labels. Yilmaz and Shah [200] proposed to exploit the differential properties of spatio-temporal volumes for efficient encoding. Blank et al. [10] proposed to use the saliency to efficiently capture human's movement over time.

Captured videos and images often contain useless information which is un-related to the human actions (e.g. backgrounds). In the global representation, both useful and useless components are mixed. To filter out such useless components, local representations of actions based on the interest point detection have been proposed. Local representation become popular after the work of Laptev [92] where the Harris corner detector is extended to the temporal domain and applied to RGB videos to encode the space-time interest points. However, the detected interest points were rather sparse. Instead of using sparse interest points, Dollár et al. [34] proposed using dense interest points that are able to reflect a more complete information. A small spatio-temporal volume is proposed as the interest points, and appearance features such as gradients, optical flows are calculated to create a vocabulary via quantization methods (e.g. k-means clustering) similar to the bag-of-words model.

Some researchers proposed the feature representation by extending the 2D representation used in image classification tasks to the spatio-temporal domain. Scovanner et al. [163] extended the scale-invariant feature transform (SIFT) [125] to the spatio-temporal domain and named it as the "3D-SIFT". Klaser et al. [87] extended the histogram of gradients (HOG) descriptor [30] to propose the "HOG3D". As an alternative, Laptev et al. [93] proposed the histogram of the optical

flow (HOF) by extracting optical flow and calculating HOG features on it. Combination of this feature extensions and support vector machine (SVM) recorded the state-of-the-art performance in 2009 [221].

Fixed spatio-temporal location for extracting features is ineffective, as motion information is often lost due to sudden motion changes. To tackle the problem, many algorithms have been proposed using spatio-temporal tracking of points over time. The spatio-temporal trajectory information was proven effective for action recognition task [112, 114, 220]. Messing et al. [114] proposed to extract trajectories by using a point detector [92] and a Kanade-Lucas-Tomasi (KLT) feature tracker [104]. Matikainen et al. [112] also used a KLT feature tracker for tracking trajectories. Wang et al. [220] proposed using denser sampling for extracting spatio-temporal points. However dense sampling method improved the overall action recognition accuracy at the cost of the high time complexity. To reduce the time complexity, Oneata et al. [131] and Peng et al. [140] explored the use of Fisher vectors [141] for learning the compact representation. Wang and Schmid [217] also proposed similar Fisher vector encoding and have obtained the state-of-the-art accuracy.

Depth cue. The recent emergence of cost-effective and easy operation depth sensors [172] have opened the door to a new family of methods for action recognition from depth sequences. Compared to conventional color images, depth maps offer several advantages: 1) Depth maps encode rich 3D structural information, including informative shape, boundary, geometric cues of a human body and an entire scene. 2) Depth maps are insensitive to changes in lighting and illumination conditions that make it possible to monitor patient/animal 24/7. 3) It is invariant to texture and color variations, which eases the task of human detection and segmentation.

As the feature encoding method developed for the RGB domain is non-optimal for the depth domain, many works have been proposed for encoding the depth appearance. Yang et al. [233] proposed the depth motion maps (DMM) by stacking motion energy (i.e. substraction of two consecutive frames) of depth maps projected onto three orthogonal Cartesian planes. Then they extracted HOG descriptors from DMMs. Wang et al. [223] defined Hierarchical DMMs by using different offsets between frames and extracting CNN features from them. More recently,

Rahmani et al. proposed a view-invariant descriptor histogram of oriented principal component (HOPC) [151] to deal with the 3D action recognition from unknown and unseen views.

The deep learning networks are also non-optimal for the depth domain. Especially, Rahmani et al. pointed out the cross-view invariance issue and proposed the view-invariant representation [152] using CNNs. It has shown the state-of-the-art accuracy on both single-view and multi-viewed depth-based action recognition benchmarks. In chapter 5, we employed the Rahmani et al. [152] pre-trained network as our feature extractor.

Skeleton/pose cue. Pose estimation is shown beneficial for understanding human actions in [45, 108, 235], while action recognition can also facilitate 3D human pose estimation as shown in [241]. The joint modeling of action and pose has been studied on RGB data in [123]. Pose estimation is performed at testing stages, which either helps further action recognition [45, 235] or is helped by prior action recognition [241]. In either case, accurate pose estimation at testing is aimed at. A well trained skeleton tracker can provide a high-level cue for depth sequences. The use of skeleton joints has been suggested by [105, 213, 222] for alleviating ambiguities in action recognition. Wang et al. [222] represent the interaction between human body parts and environmental objects with an ensemble of human joint-based features. Vemulapalli et al. [213] have represented entire skeletons as a point in the Lie group and capture temporal dynamics by applying the Fourier temporal pyramid (FTP). Their feature descriptors are constituted with a moving pose (MP) descriptor proposed in [246] that encode poses as the atomic motions and used for mining useful key- frames adopting K-nearest neighbor approach. Skeleton joints have also been used to constrain the dictionary learning for feature representation [105]. There have been several works [48, 49, 247] that use skeleton/pose cues both at testing and training stages. In those works, estimated poses are relatively stable and provide good discrimination among actions, since most human poses are captured in the upright position and the camera is located in front of humans. However, human pose estimation is not always stable due to the noisy depth maps, self-occlusions by camera views and diverse human poses as demonstrated in [216]. To solve these issues, Wang et al. [216] considered the best-K joint configurations to reduce the

joint estimation errors. In our work, estimating human body joints is even more challenging, due to ambiguous and unique human poses (*e.g.* lying) in a hospital environment. To avoid the unreliable dependency, we used the ground-truth of human poses and their 3D relation to layouts to aid model decision at training while bypassing their explicit estimation at testing.

Recently, some works have utilized CNNs to automatically learn features directly from skeleton data [80] or from Lie group representations [69]. Researchers have also proposed deep sequential models that involve vanilla recurrent neural network (RNN)s [36] and with LSTMs [212] to model temporal dependencies. However, these models have exhibited inferior performance compared to recent models that explicitly exploit static information [218] or well-suited time-series mining called Gram Matrix [250]. In demonstrating the benefit of combining feature learning and sequential deep models, Du et al. [36] have first proposed a hierarchical, end-to-end architecture that uses a hierarchical bi-directional RNN. This approach contrasts with Veeriah et al. [212], who have directly fed hand-crafted features into a RNN with LSTM.

3D Scene layout cue. 3D Scene layouts provide the useful information for categorizing actions. Often, actions have high correlation with its surrounding 3D scene layouts. For example, in chapter 5, our targeted actions have the close relationship with the bed and floor layouts (e.g. "falling from the bed" or "lying on the bed"). There have been several works for inferring indoor scene layouts such as object, wall, floor, and ceiling from 2D inputs [65, 219]. Furthermore, many researchers have shown the correlation among pose estimation, action recognition and the 3D scene layouts: Fouhey et al. [44] and Delaitre et al. [32] show that by observing human behavior, a strong correlation can be found between human actions and properties of a scene and its objects. Similarly, Savva et al. [162] observe and track people as they interact with the environment using RGB-D sensors. These methods aim at improving the estimation of 3D scene geometry. Recently, Tulsiani et al. [208] proposed to predict object shapes and poses simultaneously with the indoor scene layout estimation and have obtained the improved accuracy. Gupta et al. [59] proposed using the environmental cues when understanding the human-object interactions. Yu et al. [239] proposed the video representation for understanding actions using the scene layouts as the feature descriptor. In chapter 5, we try to use these 3D scene layout

cues as well as 3D human body joints, to aid the 24/7 patient monitoring scenario. Due to the challenging and unseen viewpoints, the 3D geometric information is encoded only during the training stage where the ground-truths could be secured.

2.2.2 Random forest, convolutional neural networks

RF-based method. Standard RFs are popularly used for tackling the action recognition problem, as it shows good performance with low complexity. Fothergill et al. [43] proposed using RFs in the action recognition domain by fully exploiting multiple frames and classifying their pooled features according to the action labels. Zhu et al. [255] adopted the bag-of-words model in the context of the action recognition problem, thereby capturing poses and creating "bags-of-poses" that encode critical motions of human actions. Mikolajczyk and Uemura [115] proposed an unified framework for action recognition and localization. To encode the temporal cues during the training stage of the RFs, [73, 165, 240] exploited the Hough forest [46] framework to infer the temporal offsets to the center of a video. Chen et al. [25] proposed a temporal smoothing term in the RF training. Also, Charles et al. [22] proposed a temporal context term in by encoding the consistency of the confidence maps through temporal cues which are calculated from the optical flow on human body poses.

CNN-based methods. Convolutional neural networks (CNN) [91] have shown the state-of-the-art performance for representation learning on various computer vision tasks. The CNN has been applied in action recognition as well in recent decades. Simonyan and Zisserman [177] proposed learning two streamed CNN features by exploiting the appearance from RGB image inputs and the motions from optical flow inputs. The network training was transferred from the VGG network which was pre-trained on the image recognition tasks. Feichtenhofer et al. [42] explored diverse temporal fusion methods in the two streamed networks [177] and obtained significant improvements using convolutional fusion method. To more effectively capture long-range temporal dependencies, recurrent neural network (RNN) with long-short term memory (LSTM) [66] have been used. The feature-learning methods for color frames

within an LSTM architecture was introduced in [35]. Feature learning could be performed by constituting the motion stream either with the two-stream model [121] or by an attention mechanism [98]. Later, 3D CNN [211] is proposed to encode spatio-temporal information from RGBs. The temporal axis is regarded as the third axis and 3D convolution is applied in the spatio-temporal video inputs. Recently, Carreira and Zisserman [21] proposed a new CNN architecture that embeds 3D convolution operations. Besides, a method to lengthen the range of dependency in videos were investigated in [225].

2.2.3 RGB-D action recognition datasets

Most action recognition datasets that use the RGB-D sensors have human body pose annotations via Microsoft Kinect [172].

Single-viewed gaming or daily activity dataset. Since RGB-D sensors work only in the indoors, action classes are usually limited to daily actions, gaming or human-computer interaction scenario. The first proposed RGB-D benchmark was MSR-Action3D [97] that contains actions in a gaming scenario with a fixed camera perspective. Similar popular datasets containing the gaming scenario are MSRC12 [43] and UT-Kinect datasets [231]. Gaming actions are usually recorded from a single camera viewpoint with the humans in frontal and upright position. Popular datasets regarding daily actions are MSR-DailyActivity3D [222] and Florence-3D [164]. These are usually more challenging because they involve objects and users are not necessarily in the frontal viewpoint.

Multi-viewed dataset. One limitation of the aforementioned datasets is that they are recorded from a single and fixed camera perspective. UWA3D Multiview II [152] and NTU RGB+D [166] datasets were proposed to explore the use of multiple camera viewpoints. The NTU RGB+D [166] dataset has more than $56,000$ videos and 60 classes. This is one of the largest-scale real dataset in the daily action recognition with multiple viewpoints. It also provide IR images.

In chapter 5, We proposed dealing with 24/7 patient monitoring scenario, where the human body pose is difficulty to be secured by the Microsoft Kinect [171]. On the other hand, in the intended scenario, the 3D scene layouts (e.g. bed and floor), 3D body pose and their relationship are useful to reveal the targetted actions such as 'lying on the bed" or "falling from the bed". Also, to meet the real-time requirement, we employed an efficient RF-based framework.

2.2.4 Hand gesture recognition (egocentric action recognition)

Action recognition from an egocentric viewpoint has opened new challenges for the action recognition problem. The difference from the 3rd-person videos is that the egocentric viewpoint does not capture the actor's body. In this new viewpoint, hands and manipulated objects become the most prominent subjects. Additionally, the camera is moving, as it is mounted at a person's head position. Thus, there could be noises or abrupt motion changes in captured videos.

Temporal reasoning of hand movements representing semantic actions (i.e. hand gestures) is an important topic. The aim of the task is similar to that of recognizing human body actions. However due to the domain difference (mainly different viewpoints [106] – that is, an egocentric view and self-occlusions introduced by manipulated objects [157]), more specialized algorithms have been proposed. Fathi et al. [40] found that hand features encode rich information for the egocentric action recognition problem. Pirsiavash and Ramanan [143] proposed using HOG descriptors to model objects and introduced the importance of the incorporation of object cues in the problem. Most recent works [71, 106, 179] also proposed using features extracted from both hands and objects. Ishihara et al. [71] extract HOG descriptors and encode the temporal information using IDTs. Ma et al. [106] and Singh et al. [179] apply two-stream networks [42, 177] in an egocentric setting and have obtained good performance.

Skeleton/pose cue. In full-body human action recognition, it is known that using higher-level and viewpoint invariant features such as body pose can benefit action recognition [227, 235], although this has not yet been studied in detail for hands. Compared to full-body actions, hand actions present unique differences that make the use of pose as a cue not obvious: style and

speed variations across subjects are more pronounced due to a higher degree of mobility of fingers and the motion can be very subtle. A setback for using hand pose for action recognition is the absence of reliable pose estimators off-the-shelf in contrast to full body [171, 226], mainly due to the absence of hand pose annotations on real (*cf.* synthetic) data sequences, notably when objects are involved [159]. One of the seminal work is [50] that proposes a hand gesture benchmark that includes hand-object interacting gestures, hand pose annotations and 3D object annotations jointly. In this work, authors introduce a new dataset of first-person dynamic hand action sequences with more than 100,000 RGB-D frames annotated with 3D hand poses and 6D poses of manipulated 3D objects, using 6D magnetic sensors attached to the fingertips and inverse kinematics. Tekin et al. [198] proposed a framework that simultaneously captures actions, 6D object poses and hand poses using the database proposed in [50].

RGBD cue. Raw RGB frames or depth sequences are the primary inputs for the egocentric action recognition problem. As in the hand domain, differently to the human bodies, an off-the-shelf pose estimator is not available due to varied camera viewpoints, Thus, most of methods rely on the RGB or Depth frames. The early work in first-person action recognition [71] found that daily actions are well explained by looking at hands, a similar observation found in third-person view [234]. In these approaches, hand information is extracted from hand silhouettes [106, 180] or discrete grasp classification [71] using low-level image features.

2.3 Evaluation criteria

This section introduces the evaluation metrics for both action recognition and pose estimation tasks. Different criterion is used for each task.

2.3.1 Accuracy measurement on action recognition

Action recognition problem is the multi-class classification problem. We used several existing measures for evaluating our algorithms: ‘accuracy’, ‘precision’. The ‘precision’ is defined to

be the percentage measure of the number of correctly classified videos in relation to the total number of videos. One drawback of this measure is its sensitivity to the class imbalance of the specific databases, which denotes the problem where some classes are much more common than others. Thus, this measure could introduce a bias to selecting the model. To relieve the issue, 'average class accuracy' was proposed to compute the number of correctly classified videos over the total videos for a given class and considers the average as the final performance indicator.

2.3.2 Accuracy measurement on pose estimation

Mean testing error in *mm* unit is a widely used measurement for evaluating the pose estimation systems. It can help users to judge how well pose estimation algorithms perform on average. However, this measurement is not enough to evaluate the detailed behavior of the algorithms. For example, one cannot tell the proposed algorithm performs well on which kind of data solely using the "mean testing error" . To visualize overall distributions of hand pose samples, *proportion (in %) of the frames with all joints error*$<\epsilon$ (in Euclidean distance per joint) being smaller than a tolerance parameter ϵ [197] is proposed, which is defined as:

$$r_f = \frac{N_f}{N} \tag{2.1}$$

where N is total number of frame, $N_f = g(\epsilon)$ is the number of frames whose joints are all with in euclidean distance of ϵ to the ground truth. This metric shows overall distribution of the hand pose testing samples, and shows the percentage of good and bad samples in a glance.

3 CHAPTER

COARSE POSE ESTIMATION VIA EXPLICIT 2.5D DATA RECONSTRUCTION AND AUGMENTATION

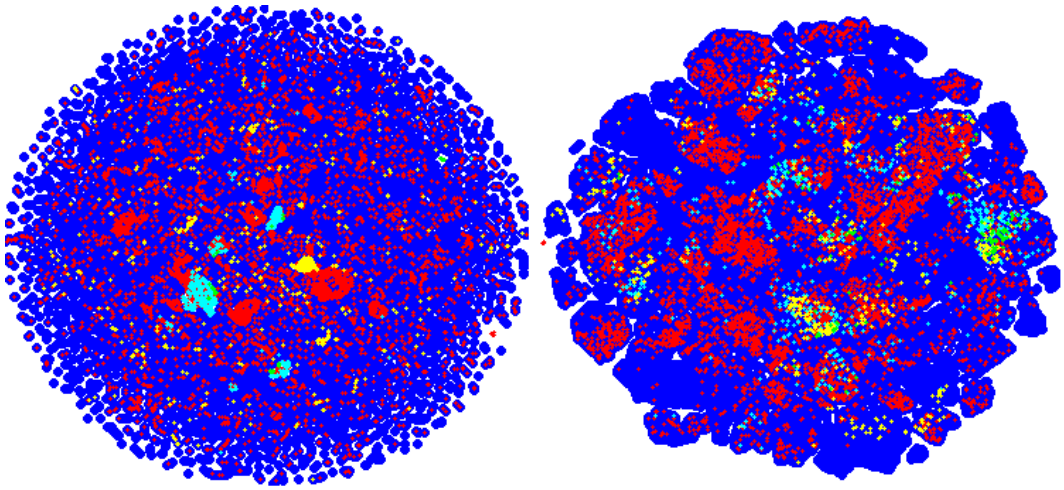
Contents

This chapter describes the method to jointly tackle the 'inherent 2D-to-3D ambiguity' and 'insufficient data and quality annotation' issues in the pose estimation problem, mentioned in chapter 1. We propose to reconstruct new 2.5D depth data having novel viewpoint

to shape parameters based on the conditional generative adversarial network (GAN) [137] framework. We use the bone length as the simplified shape parameter in this chapter, based on the research on hand dimensions [28, 39]. While a deterministic rendering using a 3D model is possible, synthetic depth maps exhibit an observable difference from real data [174]: It is known that the model trained by pure synthetic data does not generalize well to the real testing data. Thus, in [174], conditional GAN [137]-based approaches are proposed to fill the gaps. Similarly, in this chapter, we use the conditional GAN [137] framework for hand pose generator (HPG) and scheduled the attributes (i.e. viewpoint, shape) of synthesized data to be diversified. An alternative is training a hand pose estimator (HPE) with a *simple* depth map and skeleton pair manipulation, e.g. by in-plane rotations and translations. As shown in our experiments, while this way of augmenting data helps, the resulting database coverage, however, is limited.

Crucial to the success of training a depth-based 3D HPE is the availability of comprehensive datasets covering diverse camera perspectives, shapes, and pose variations. However, collecting such annotated datasets is challenging. We propose to *complete* existing databases by generating new database entries. The key idea is to synthesize data in the skeleton space (instead of doing so in the depth-map space) which enables an easy and intuitive way of manipulating data entries. Since the skeleton entries generated in this way do not have the corresponding depth map entries, we exploit them by training a separate hand pose generator (HPG) which synthesizes the depth map from the skeleton entries. By training the HPG and HPE in a single unified optimization framework enforcing that 1) the HPE agrees with the paired depth and skeleton entries; and 2) the HPG-HPE combination satisfies the cyclic consistency (both the input and the output of HPG-HPE are skeletons) observed via the newly generated unpaired skeletons, our algorithm constructs a HPE which is robust to variations that go beyond the coverage of the existing database.

Our training algorithm adopts the GAN training process. As a by-product, we obtain a hand Pose discriminator (HPD) that is capable of picking out realistic hand poses. Our algorithm exploits this capability to refine the initial skeleton estimates in testing, further improving the accuracy. We test our algorithm on four challenging benchmark datasets (ICVL [194], MSRA [190],



(a) Viewpoint and pose space (b) Viewpoint space

Figure 3.1: *t-SNE embeddings of skeletal poses of Big Hand 2.2M, ICVL, NYU, MSRA datasets, and our augmented skeletons. Each dataset covers up to a certain degree of shape and viewpoint variations but none of them is* comprehensive *as indicated by the presence of empty space between different clusters. Our data augmentation process fills in the space and provides a more comprehensive coverage of viewpoints and poses. To experience the full detail of this figure, readers are advised to view the electronic version.*

NYU [205], BigHand2.2M [244] datasets) and demonstrate that our approach outperforms or is on par with state-of-the-art methods quantitatively and qualitatively.

3.1 Motivation

A straightforward approach to construct a robust hand pose estimator (HPE) might be to train it on a large dataset that covers such variations. However, as far as we are aware, existing datasets are limited in the coverage of camera viewpoint, shape, and/or pose variations (see Sec. 3.2).

When the data space is visualized using the ground-truth annotations of hand poses, shapes, and camera perspectives in such databases, one can identify the *missing* regions in the space, e.g. camera perspectives that are not covered by the database (see Fig. 3.1).

We present an algorithm that mitigates these limitations by augmenting the camera views and shapes. In training, we synthesize unseen human hands in the *skeleton space* and *transfer* them

to synthetic depth maps: This helps to avoid the challenge in manipulating the depth maps and provide an easy and intuitive way to close the gaps in the data space by editing existing data points. To facilitate the transfer of generated skeletons to depth maps, we introduce two new data processing networks: Inspired from the recent success of 2D/3D image generation [135], we train the hand pose generator (HPG) that transfers input skeletons to corresponding depth maps. As in GAN [56, 149], we train the second, hand pose discriminator (HPD) that distinguishes real depth maps from these synthesized by the HPG. Combining and jointly training HPG, hand pose estimator (HPE), and HPD enable the automatic transfer of the augmented skeletons to depth maps by enforcing the consistency over existing paired skeleton and depth map database entries and the self-consistency over unpaired augmented skeletons. The HPD's ability (combined with HPG) to pick out realistic human hands can also be used in testing: During testing, we synthesize multiple hand pose hypotheses out of the initial HPE prediction, and generate the final refined prediction by combining them using the HPG and HPD as a prior.

To summarize, we contribute by 1) a new HPG and HPE combination that enables to exploit both existing paired skeletons and depth map entries and newly synthesized depth maps in a single unified framework; 2) a strategy that refines the HPE prediction during testing by generating multiple pose hypotheses and combining them using HPD and HPG as a prior. In the experiments with four challenging datasets, we demonstrate that our robust algorithm outperforms or is on par with state-of-the-art algorithms on each dataset.

3.2 Hand pose estimation by skeleton augmentation

Given a database of input depth maps and the corresponding ground-truth hand pose annotations $D_P = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{I} \subset X_D \times Y_P$, our goal is to construct a hand pose estimator (**HPE**) $f^P : X_D \rightarrow Y_P$ that recovers the underlying pose $\mathbf{y}'$ of an unseen test depth map $\mathbf{x}'$. When the *paired* dataset D_P is large enough to cover variations in poses, shapes, views, etc., a straightfor-

ward approach to train such a HPE is to minimize the mean squared loss over D_P:

$$\mathcal{L}_P(f^P) = \sum_{i=1}^{l} ||f^P(\mathbf{x}_i) - \mathbf{y}_i||_2^2. \tag{3.1}$$

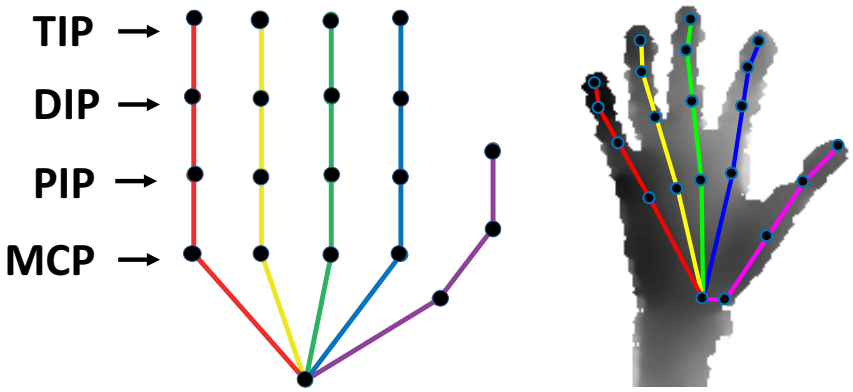
For this baseline, we use the CNN architecture in [244]: Each input depth map $\mathbf{x}$ is presented as a 96×96-dimensional array while for the output $\mathbf{y}$, we adopt the 63-dimensional skeletal pose vector representing the (x, y, z)-coordinate values of 21 hand joints (Fig. 3.2).

Unfortunately, existing datasets do not comprehensively cover the wide variety of hand shape and views. Therefore, we explicitly fill-in these missing regions by synthesizing data entries in the skeleton space Y. Once such *unpaired* skeletal poses $D_U = \{\mathbf{z}_i\}_{i=1}^{u}$ are synthesized, training the HPE is performed based on a combination of the standard estimation error $\mathcal{L}_P$ via D_P (Eq. 3.5) and the cyclic consistency requirements induced from D_U (see Fig. 3.5 and Eq. 3.8). To facilitate this process, we train a hand pose generator (HPG) $f^G : Y \rightarrow X$ that receives a skeleton (either $\mathbf{y}$ or $\mathbf{z}$) and synthesizes the corresponding depth map $\mathbf{x}$. Note our skeletal representation incorporates camera perspectives (i.e. rotational transformation of skeletons): 63-dimensional skeletal pose values are assigned based on the coordinate system defined by the viewpoints.

3.2.1 Skeleton augmentation

Skeletal hand shape model. We use the 21 joint-based skeletal hand shape model proposed in [244] (Fig. 3.2). This model represents a human hand based on 25 joint angles and the lengths of 20 bones connecting joint pairs: Each finger pose is represented as 5 angles (twist angle, flexion angle, abduction angle for the meta carpo phalangeal (MCP) joint and flexion angles for the distal inter phalangeal (DIP and proximal inter phalangeal (PIP) joints and 4 bone lengths.

Hand datasets: obtaining D_P. The performance of hand pose estimator (HPE) depends on its training datasets. The Big Hand 2.2M dataset collected by Yuan et al. [244] is the largest dataset including 2.2 million frames extracted from sequences of $\binom{32}{2} = 496$ transitions between $2^5 = 32$ extreme hand poses. While it provides a comprehensive hand pose coverage, *Big*



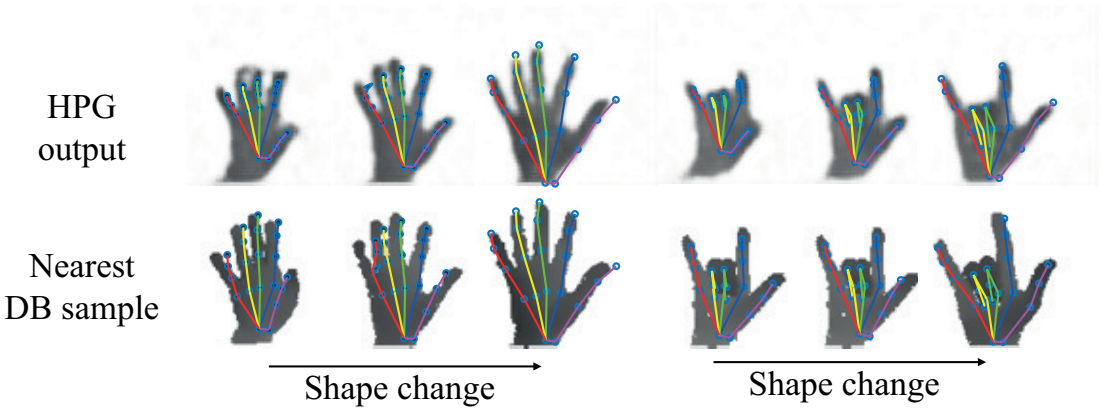
(a) (b)

Figure 3.2: *Our skeletal hand model (a) consists of 21 joints [244]: one for wrist and four for each finger. Each finger has five degrees of freedom: flexion for DIP and PIP, flexion, abduction and twist for MCP. (b) a skeleton overlaid on the underlying depth map.*

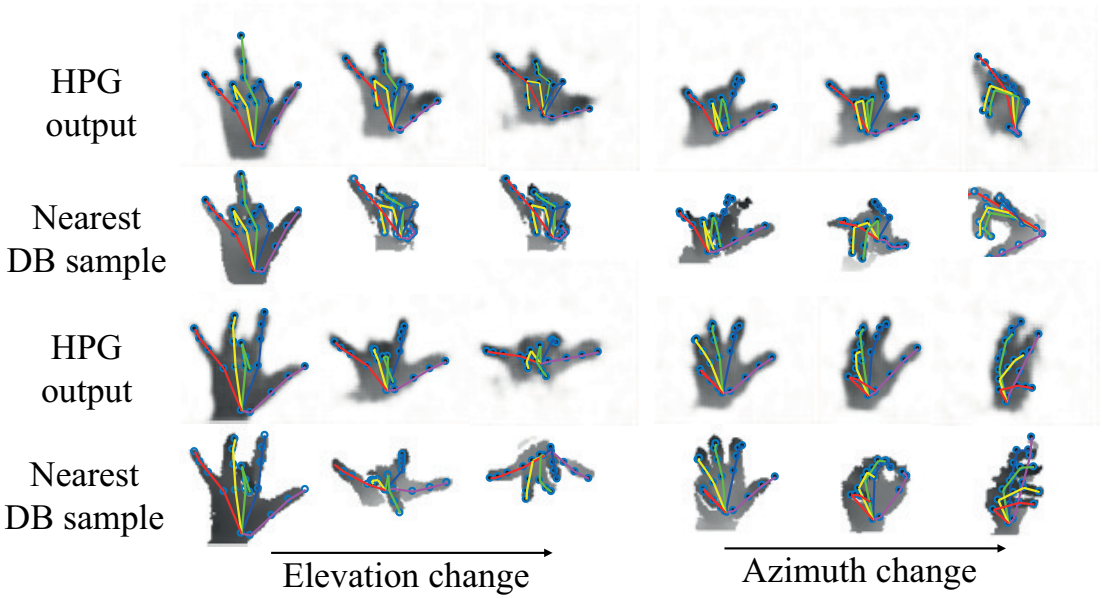
Hand 2.2M still lacks the variety in hand shapes (only 10 subjects) and in camera views (see Fig. 3.3). Other popular datasets include ICVL [194], MSRA [190] and NYU [205]. The ICVL benchmark [194] includes only 1 subject and provides a limited coverage of viewpoints and poses consisting of 17,604 frames. The NYU dataset provides a broader range of viewpoints (81,009 frames) but with limited shape variations (one subject for training and another subject for testing). The MSRA [190] benchmark is similar in scale to NYU (76,375 frames) but with a more comprehensive viewpoint coverage. However, its shape and pose variations are limited to 9 subjects and 17 discretized poses per subject, respectively.

Skeleton augmentation: constructing D_U. For each of the four datasets aforementioned, we enlarge its skeleton space coverage by adding variations in viewpoints and shapes. We do not consider pose (i.e. articulation) augmentation as we observed in preliminary experiments that synthesizing realistic hand poses without having access to statistical or physical hand models is challenging.

New camera perspectives (viewpoints) of an existing skeleton entry are synthesized by applying (3D) rotations along $y-z$ and $x-z$ panes, prescribed by the corresponding rotation degrees θ_1 and θ_2. In-plane rotations (along $x-y$ pane) can also be considered but we exclude them in the skeleton augmentation process as the corresponding paired data is straightforwardly constructed



(a) Shape variation



(b) Viewpoint variation

Figure 3.3: *Synthesized skeletons* $\mathbf{y}'$ *overlaid on the depth maps* $\mathbf{z}'$ *transferred by HPG* ($\mathbf{x}' = f^G(\mathbf{y}')$). *These new data entries* augment *the coverage of the database: The nearest skeletons (and the paired depth maps) in the database deviate significantly from the query synthesized skeletons.*

by rotating the skeleton and depth map pairs. In the experiments (Table 3.1(b)) we demonstrate that both simple in-plane rotations and our skeletal augmentation help improve the performance and furthermore, they are complementary: The combination of the two data augmentation modes is better than either taken alone.

Adopting existing models from human hand shape analysis [28,39], we characterize hand shapes based on the width to length ratio of each finger. Accordingly, new skeletons are generated by

Algorithm 1: Skeleton augmentation process

Input: a skeleton sample y_i, τ, θ_1, θ_2, and the number of augmentation M
Output: the augmented skeleton sample y_i^*
for *m=1 to* M **do**
  Sample τ, θ_1, θ_2 from $\mathcal{N}(1, 0.5^2)$, $\mathcal{N}(0, \frac{\pi}{4}^2)$, $\mathcal{N}(0, \frac{\pi}{4}^2)$, respectively.
  for *h=1 to* 5 **do**
    Calculate the bone lengths of the h-th finger: measure the Euclidean distances between MCP and PIP, PIP and DIP, DIP and TIP joints.
    Calculate the joint angles by Algorithm 2.
    Manipulate the bone lengths by multiplying τ.
    Reconstruct the h-th finger by Algorithm 4.
  end
  Manipulate the rotations by multiplying the rotation matrices θ_1, θ_2.
end

varying the finger lengths of existing data entries as measured in Euclidean distances between finger tip Joints (TIP) to DIP, DIP to PIP and PIP to MCP while fixing the palm (see Fig. 3.2). While fixing 6 palm positions, we first identify 5 angles (i.e. flexion angles for TIP, DIP and PIP and twist angle, flexion angle abduction angle for the MCP) and 3 bone lengths (distances from MCP to PIP, from PIP to DIP and from DIP to TIP) for each finger and given them, reconstruct each finger by rotating/translating their end points to attach them to the palm. We assume that the ratios of finger lengths are fixed and thus only the bone length of each finger is manipulated by multiplying a global constant τ in the above reconstruction process.

The variation parameters θ_1, θ_2, and τ are sampled from Gaussian distributions: $\mathcal{N}(1.0, 0.5^2)$ for τ and $\mathcal{N}(0, \frac{\pi}{4}^2)$ for θ_1 and θ_2. While in general, more sophisticated data manipulation strategies can be adopted, our preliminary visual evaluation on a small sample revealed that skeletons generated in this way look realistic. Figure 3.3 shows example skeletons (overlaid on the corresponding depth maps synthesized by HPG; see Sec. 3.2.2) generated from this process. Note that the alternative way of directly manipulating depth maps could be much more challenging as the variables are highly structured and correlated, and therefore naïvely manipulating depth pixels would lead to unrealistic hand shapes. We apply the manipulation process M times for each database entry constructing an unpaired database D_U ($|D_U| = M|D_P|$). Table 3.1(b) shows the effect of varying M on the final pose estimation performance.

Algorithm 2: Calculate the joint angles of a finger

Input: 4 skeleton joints locations (MCP, PIP, DIP, and TIP; see Fig. 3.2) of a finger;
Output: 5 joint angles of the input finger;
Compute the two flexion angles for DIP and PIP using two joint sets: (PIP, DIP, TIP) and (MCP, PIP, DIP) (by Eq. 3.2);
Reconstruct the base finger (by Algorithm 3);
Align the base finger to the observed skeleton counter part by calculating a rigid transform and obtain 3 joint angles (twist, flexion and abduction angles) for the MCP.

Algorithm 3: Reconstruct the base finger model

Input: 3 bone lengths (*i.e.* distances between MCP and PIP, PIP and DIP and DIP and TIP), 2 joint angles (*i.e.* flexion angles for DIP and PIP)
Output: Reconstructed 4 finger joints (*i.e.* MCP, PIP, DIP and TIP locations)
Assign MCP's 3D location as $(0, 0, 0)$.
Assign PIP's 3D location as $(0, 0, 0)$ and shift it in the y-axis by the bone length between PIP and MCP.
Assign DIP's 3D location as $(0, 0, 0)$ and shift it in the y-axis by the the distance between DIP and PIP, then rotate it along the x-axis by flexion angle for the PIP. Add PIP location as the starting point.
Assign TIP's 3D location as $(0, 0, 0)$ and shift it in the y-axis by the distance between TIP and DIP, then rotate it along the x-axis by the sum of flexion angles for the PIP and DIP. Add DIP as the starting point.

Algorithm 4: Reconstruct the hand model

Input: 5 skeleton joint angles, 3 bone lengths of each finger
Output: 4 skeleton joints (*i.e.* MCP, PIP, DIP and TIP) of the finger
Reconstruct the base finger (by Algorithm 3).
Rotate the base finger by the twist, abduction and abduction angles of the MCP, obtained in Algorithm 2.
Attach the finger to the palm.

Figure 3.1 visualizes the results of skeleton augmentation: We observe that even the biggest *Big Hand 2.2M* dataset is far from being fully covering the wide variations in shapes and camera viewpoints as evidenced by almost 10-times larger area coverage accomplished by our augmented dataset.

Algorithm 1 combined with Algorithm 2,3,4 contain more details about the skeleton augmentation process. The joint angle θ between two 3D joint vectors $\mathbf{a}$ and $\mathbf{b}$ is calculated as

follows:

$$\theta = \text{atan}(||\mathbf{a} \times \mathbf{b}||, \mathbf{a} \cdot \mathbf{b}). \tag{3.2}$$

3.2.2 Transferring skeletons to depth maps

Hand pose generator (HPG) . Our HPG f^G synthesizes a depth map $\mathbf{x}$ given the input skeleton parameters $\mathbf{y}$. We adopt Pathak et al.'s conditional GAN architecture [137] that combines both L_2-loss and adversarial loss (Eq. 3.4): The L_2 loss (defined via P) measures the deviation of the synthesized depth maps from the ground-truths while the adversarial loss generates data distribution and helps the generator to synthesize more plausible data samples.

Hand pose discriminator (HPD). We construct auxiliary models that provide feedback on the quality of synthesized data. To leverage the cyclic nature of HPE and HPG combinations (i.e. $f^E(f^G)$ and $f^G(f^E)$ map Y to itself and X to itself, respectively; see the next paragraph), we train two such discriminators: The depth hand pose discriminator (HPD_X) f^{D_X} is the same as the standard GAN discriminator; It outputs 1 for real data entries and 0 for the synthesized entries. The role of skeleton hand pose discriminator (HPD_Y) f^{D_Y} is to decide whether the estimated finger joints conform the human skeleton model (Fig. 3.2). It aims to accept original skeletal entries $\mathbf{y}$ in P as well as the augmented entries $\mathbf{z}$ in U, while rejecting outputs from the HPE. Therefore, incorporating f^{D_Y} into the joint GAN training enables us to steer the training of the generator f^G towards the skeletal poses that were not covered in the original dataset P.

Network architecture for HPE, HPG and HPD. Figure 3.4 shows the architectures of our HPE, HPG, HPD_X and HPD_Y. The HPE architecture follows that of [244]: the input depth map is a 96×96-dimensional array and the output is a 63-dimensional skeletal pose vector representing the (x, y, z)-coordinates of 21 hand joints (Fig. 3.2). We adopt two stages of max-pooling and ReLU activation.

The architecture of our HPG is inspired by Radford *et al.*'s image generation network [149]. It has the same architecture as in [149] except for the sizes of the output (96×96 matching the

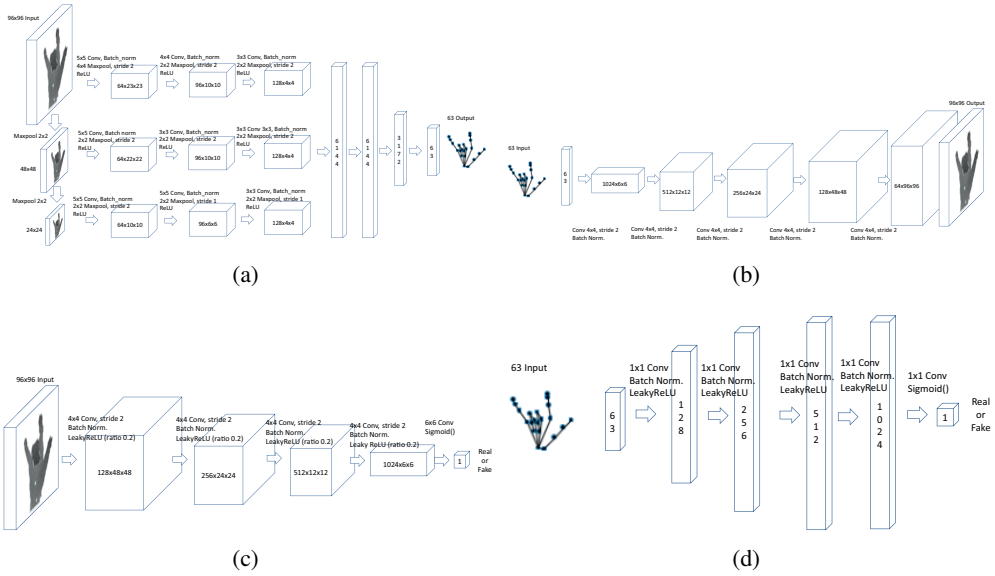


Figure 3.4: *Individual network architecture for (a) HPE, (b) HPG, (c) HPD_X, and (d) HPD_Y. The HPE architecture is inspired from [244]. HPG and HPD_X are inspired from the GAN algorithm [149] and HPD_Y has a similar architecture to HPD_X but is designed to have* 63*-dimensional vector as an input.*

size of the HPE input) and input (63 skeleton dimensions) layers, and the corresponding first convolutional kernel size (6×6). It adopts fully convolutional units and batch normalization in each layer, followed by the *Tanh* unit in the last layer

The architectures of HPD_X is the same as Radford et al.'s GAN discriminator [149]. It has 4 convolutional layers with the kernel size 4 and stride 2, which turns a 96×96 sized depth input to a 6×6 sized response map. The last convolution layer with the kernel size 6×6 then produces a scalar value. The *Sigmoid* unit is applied to get the output value in $[0, 1]$, as the probability for the binary classes

HPD_Y has a similar architecture to HPD_X but its input is a 63-dimensional vector. Thus, we apply fully connected layers to obtain the final binary labels (rather than the convolutional units). The number of response maps remains the same as that of HPD_X. The *Sigmoid* unit is applied in the last layer. For both HPD_X and HPD_Y, we apply batch normalization and LeakyReLU in all convolutional layers.

Training HPG and HPE. Our goal is to jointly train hand pose generator (HPG) and hand

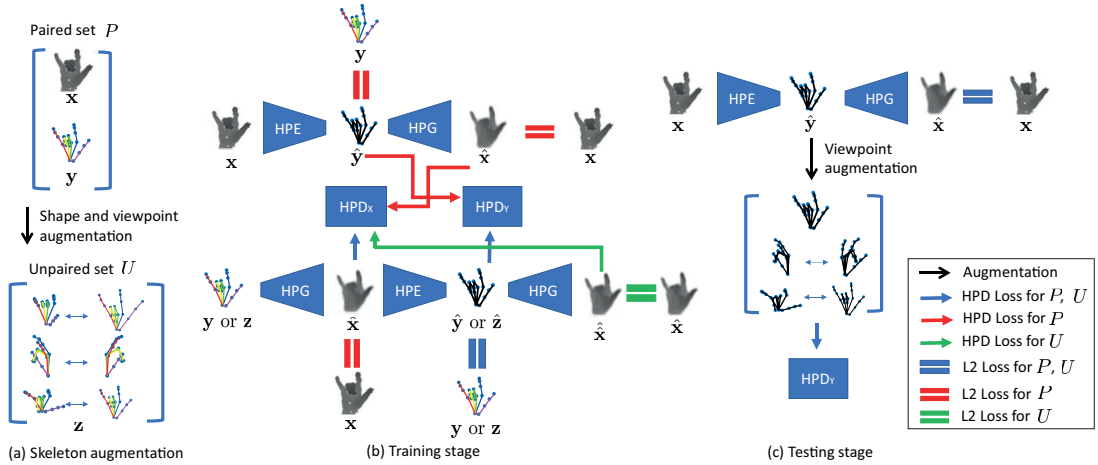


Figure 3.5: *Schematic diagrams of our algorithm. (a) Manipulating skeletons is easier than manipulating depth maps; (b) During training, HPE, HPG, HPD$_X$, and HPD$_Y$ are optimized by 1) reducing the classical training error of HPE induced via P; 2) enforcing the cyclic consistency of HPE-HPG combination $f^E(f^G) : Y \to Y$, HPG-HPE combination $f^E(f^G) : X \to X$ on P as well as the HPG-HPE-HPG consistency on unpaired data U; (c) In testing, our algorithm refines the initial hand pose prediction as guided by HPG and HPD$_Y$ as a prior. In the diagram, Red and Green lines represent interactions with the paired set P and unpaired set U, respectively. The Blue lines represent interactions with both U and P.*

pose estimator (HPE) by fully exploiting the paired data P as well as the augmented unpaired data U. Since skeletons $\mathbf{z}$ in unpaired data U have no corresponding depth maps $\mathbf{x}$ for explicit supervision, our training algorithm adopts cyclic consistency: When the depth map $f^G(\mathbf{y}) \in X$ generated from a hypothesized input pose $\mathbf{z}$ is fed to the HPE, the resultant $f^E(f^G(\mathbf{z})) \in Y$ should be *similar* to $\mathbf{z}$. Similarly, the HPE result $f^E(\mathbf{x}) \in Y$ for an input depth map $\mathbf{x}$ can be fed to HPG and the resulting simulated depth map $f^G(f^E(\mathbf{x})) \in X$ should be similar to the original input $\mathbf{x}$.

Accounting for these requirements, our energy $\mathcal{L}$ consists of four components: The individual training losses for HPE and HPG, respectively, and the consistency losses for $f^E(f^G)$ and $f^G(f^E)$ measured based on the paired (P) and unpaired (U) data:

$$\mathcal{L}(f^G, f^E, f^{D_X}, f^{D_Y}) = \mathcal{L}_G(f^G, f^{D_X}) + \mathcal{L}_E(f^E, f^{D_Y}) + \lambda(\mathcal{L}_P(f^E, f^G) + \mathcal{L}_U(f^E, f^G)) \tag{3.3}$$

where the individual training losses $\mathcal{L}_G$ and $\mathcal{L}_E$ are defined as:

$$\mathcal{L}_G(f^G, f^{D_X}) = \mathbb{E}_{\mathbf{x}}[\log f^{D_X}(\mathbf{x})] + \mathbb{E}_{\mathbf{y}}\big[\log(1 - f^{D_X}(f^G(\mathbf{y}))\big] + ||f^G(\mathbf{y}) - \mathbf{x}||_2^2 \tag{3.4}$$

$$\mathcal{L}_E(f^E, f^{D_Y}) = \mathbb{E}_{\mathbf{y}}[\log f^{D_Y}(\mathbf{y})] + \mathbb{E}_{\mathbf{x}}\big[\log(1 - f^{D_Y}(f^E(\mathbf{x}))\big] + ||f^E(\mathbf{x}) - \mathbf{y}||_2^2 \tag{3.5}$$

with the expectations $\mathbb{E}_{\mathbf{x}}$ and $\mathbb{E}_{\mathbf{y}}$ taken respectively under the empirical marginal distributions $p_X(\mathbf{x})$ and $p_Y(\mathbf{y})$ (defined by P). The first two terms denote the adversarial loss while the last term denotes the Euclidean loss. In Fig. 3.5, Euclidean loss is denoted as the symbol"=" while the adversarial loss is denoted as the arrows "⇜" to the HPDs. The combination of Euclidean loss and adversarial loss is proposed in [137] as the way to implement the conditional GAN framework.

The consistency losses for the paired ($\mathcal{L}_P$) and unpaired ($\mathcal{L}_U$) datasets are respectively given as:

$$\begin{aligned}\mathcal{L}_P(f^E, f^G) = {} & \mathbb{E}_{\mathbf{y}}\big[\log f^{D_X}(\mathbf{x}) + ||f^E(f^G(\mathbf{y})) - \mathbf{y}||_2^2\big] + \mathbb{E}_{\mathbf{x}}\big[||f^G(f^E(\mathbf{x})) - \mathbf{x}||_2^2 \\ & + \log(1 - f^{D_X}(f^G(f^E(\mathbf{x}))))\big] + \mathbb{E}_{\mathbf{y}}\big[\log f^{D_Y}(\mathbf{y}) + \log(1 - f^{D_Y}(f^E(f^G(\mathbf{y}))))\big],\end{aligned} \tag{3.6}$$

$$\begin{aligned}\mathcal{L}_U(f^E, f^G) = {} & \mathbb{E}_{\mathbf{z}}\big[\log f^{D_Y}(\mathbf{z}) + \log(1 - f^{D_Y}(f^E(f^G(\mathbf{z}))) + ||f^E(f^G(\mathbf{z})) - \mathbf{z}||_2^2 \\ & + ||f^G(f^E(f^G(\mathbf{z}))) - f^G(\mathbf{z})||_2^2 + \log(1 - f^{D_X}(f^G(\mathbf{z}))) + \log(1 - f^{D_X}(f^G(f^E(f^G(\mathbf{z})))))\big],\end{aligned} \tag{3.7}$$

where the expectation $\mathbb{E}_{\mathbf{z}}$ is taken over the empirical distribution $P(\mathbf{z})$ of the augmented skeletons $\mathbf{z} \in Y$ (defined by U). The rationale behind these formulation is that if estimated $\hat{\mathbf{x}} = f^E(\mathbf{y})$ or $\hat{\mathbf{y}} = f^G(\mathbf{x})$ is inputted again to HPG or HPE, it should be same as its original input: $\mathbf{x} = f^E(f^G(\mathbf{x}))$ or $\mathbf{y} = f^G(f^E(\mathbf{y}))$. We explicitly enforce such supervision using the cyclic consistency with $\mathcal{L}_P$ and $\mathcal{L}_U$. In Fig. 3.5b, the red, green and blue arrows denote paths used by paired data P, unpaired data U, and both P and U, respectively.

Discussions. The cyclic consistency of $f^E(f^G)$ and $f^G(f^E)$ combinations on unpaired data U are inspired by the consistency loss of cyclic GAN [253] that exploits the mutual consistency of two transfer functions $f^E : X \to Y$ and $f^G : Y \to X$ (adapted to our problem setting). Indeed, the last terms in $\mathcal{L}_P$ and $\mathcal{L}_U$ are smooth versions of the consistency loss in [253]. However, the main goal of cyclic GAN training is to automatically infer the correspondences between

Algorithm 5: Training process for HPG and HPE

Input: Depth map and skeleton pairs $P = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^{l}$ and unpaired skeletons $U = \{\mathbf{z}_i\}_{i=1}^{u}$; Hyper-parameters: the number T of epochs and the size N' of mini-batch;
Output: HPE f^E, HPG f^G, HPD$_X$ f^{D_X}, and HPD$_Y$ f^{D_Y}.
Initialization: randomly allocate parameters of f^E, f^G, f^{D_X}, and f^{D_Y};
for *t=1 to* T **do**
  for *n=1 to* N' **do**
    Evaluate (feed-forward) f^G and f^{D_X} and their respective gradients ∇f^G and ∇f^{D_X} on P (Eq. 3.4);
    Evaluate f^E, f^{D_Y}, ∇f^E, and ∇f^{D_Y} (Eq. 3.5);
    Evaluate $f^E(f^G)$, $f^G(f^E)$, f^{D_Y}, and their gradients on U and P (Eqs. 3.6 and 3.7);
    Update f^G, f^{D_X}, f^{D_Y} combining the calculated gradients.
    Evaluate $f^E(f^G)$, $f^G(f^E)$, and f^{D_Y}, and their gradients on U and P (Eqs. 3.6 and 3.7) and update f^E accordingly.
  end
end

two unpaired sets U_X and U_Y (adapted to our problem setting) without having to use explicitly paired data. Therefore, in [253], the consistency is enforced on two sets of unpaired data U_X and U_Y. Our algorithm aims to achieve a similar goal but it is provided with a small paired dataset P plus a large unpaired dataset $U := U_X$ only in the skeleton space (as augmenting data in Y is challenging). Therefore, our algorithm indirectly induces the consistency of $f^E(f^G)$ and $f^G(f^E)$ by putting a consistency loss over a complete circle (the third and last terms in the $\mathcal{L}_U$ expectation: Eq. 3.7):

$$\mathbf{z} \xrightarrow{f^G} \hat{\mathbf{x}} \xrightarrow{f^E} \hat{\mathbf{z}} \xrightarrow{f^G} \hat{\hat{\mathbf{x}}} \approx \hat{\mathbf{x}}. \tag{3.8}$$

Furthermore, our consistency losses incorporate the contributions from the discriminators f^{D_X} and f^{D_Y}. This helps in decoupling the updates of f^E and f^G (within each mini-batch) : We empirically observed that simultaneously updating f^G and f^E by combining all gradients is prone to overfitting, i.e. the resulting HPE-HPG combination memorizes the depth map entries in U but it cannot faithfully re-generate over unseen depth maps. This can be attributed to the significantly larger dimensionalities of the depth map space X (e.g., 96^2) than the parameterized skeleton space Y (63): By simultaneously updating f^G and f^E, the algorithm tends to emphasis

the losses observed at X. These different scaling behaviors of X- and Y-losses can be addressed by explicitly scaling them, but it requires tuning a separate scaling parameter. Instead, our algorithm avoids such degeneracy by simply decoupling the updates of f^E and f^G. Algorithm 5 and Fig. 3.5(b) summarizes the training process.

Our model is trained using the Adam optimizer [86] with its default parameters: $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$. The learning rate and the regularization parameter λ (Eq. 3.3) are fixed at 10^{-4} and 10^{-4}, respectively based on cross-validation on *Big Hand 2.2M* dataset, which are fixed throughout the entire experiments (over other datasets).

Refining predictions at testing. Once the hand pose estimator f^E is trained, it can be directly applied to an unseen input depth map $\mathbf{x}'$ to generate the output pose estimate $\mathbf{y}'$. However, during the training of HPE-HPG combination, we constructed an auxiliary hand HPD f^{D_Y} that (combined with the HPG) can identify realistic skeleton configurations. Therefore, we refine the initial estimate $\mathbf{y}'$ as guided by HPG and HPD: Our initial result $\mathbf{y}'$ is updated using the gradient back-propagated from the HPG and HPD$_Y$ (see Fig 3.5(c)):

$$\mathbf{y}^* = \mathbf{y}' - \gamma \nabla \left(-f^{D_Y}(\mathbf{y}') + \lambda_{ref} ||f^G(\mathbf{y}') - \mathbf{x}'||_2^2 \right). \tag{3.9}$$

where $\gamma = 10^{-5}$ and $\lambda_{ref} = 0.01$ are fixed for all datasets by cross-validation on *Big Hand 2.2M*. This corresponds to a (computationally cheap) single step of energy minimization. In this way, the refined skeleton joints move towards matching the distribution of plausible skeleton joints.

Multi-view gradient ensemble. Throughout the training process, f^{D_X} and f^G have access to the augmented skeletons U and the corresponding transferred depth maps $f^G|_U$, covering a variety of viewpoints. We generalize our refinement strategy to a multi-view scenario by exploiting this accumulated *multi-view knowledge*: First, our refinement step generates R-different views of the initial estimate $\mathbf{y}'$ by rotating it R-times, similarly to the skeleton augmentation process in training. These multiple view hypotheses are then fed to f^{D_Y} and F^G to generate the respective gradient updates (Eq. 3.9). The final prediction is then obtained by rotating back the updated results $\{\mathbf{y}^*\}$ to the original views and taking the average. For the rotated skeletons, λ_{ref} is set

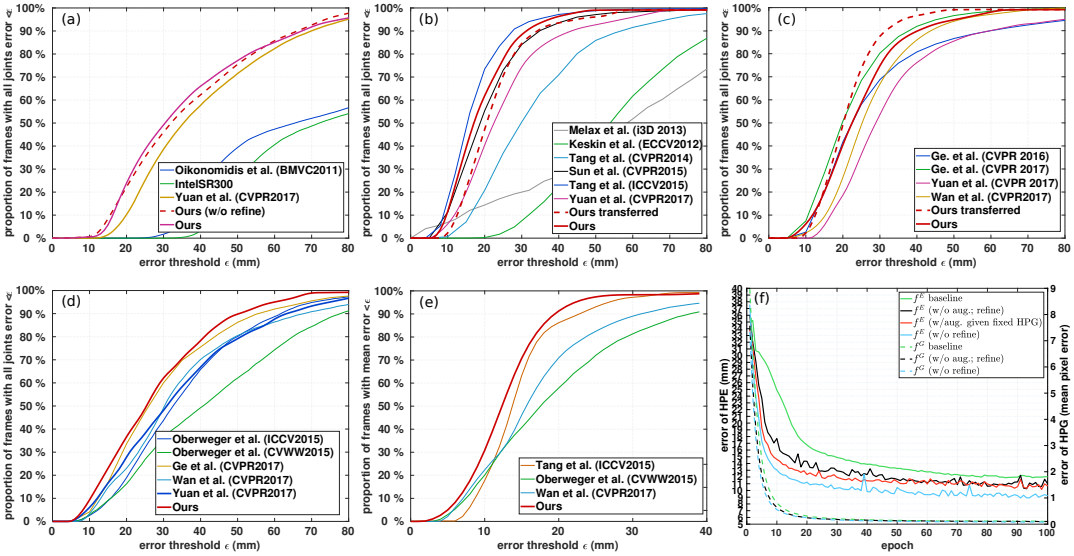


Figure 3.6: *Accuracies of different hand pose estimators for four datasets measured in* proportion of frames with worst error<ϵ *criteria (a)-(d); (e) Accuracies for NYU, measured in* proportion of frames with mean error<ϵ *criteria (for a fair comparison with Tang et al. (ICCV2015)); (a-e: the larger the area under each curve is better); Ours (shape; w/o refine): our method trained with* $\mathcal{P}$*, and* $\mathcal{U}$ *augmented with only shape variations; Ours (rotation; w/o refine): our method trained with* $\mathcal{P}$*, and* $\mathcal{U}$ *augmented with only viewpoint variations; Ours (w/o refine): our method trained with* $\mathcal{P}$ *and* $\mathcal{U}$ *fully augmented, without refinement at testing; Ours: our final algorithm including data augmentation and the refinement step; Ours transferred: our method trained on Big Hand 2.2M dataset and tested on the respective dataset (see the cross-dataset experiments paragraph); (f) The 2D plot for test errors of HPE and HPG in the same epoch (trained on ICVL). We note strong correlations in the HPE and HPG errors.*

to 0 in Eq. 3.9 as they do not have the corresponding rotated depth maps. We fix R at 50 trading off the run-time complexity with the accuracy.

3.3 Experiments

We evaluate our algorithm on four depth-based hand pose estimation datasets: Big Hand 2.2M, MSRA, ICVL, and NYU. Each dataset differs in their intended use cases and properties, and therefore, we adopt different experimental settings per dataset.

Setup. For Big Hand 2.2M, we use the experimental settings proposed by the authors of the dataset, Yuan et al. [244] (see Sec. 3.2.1 for the discussion on this dataset): We use 90% and 10% of database entries for training and validation, respectively. For testing, 37,000 frames

captured from a subject not contained in the training/validation sets are used. For the MSRA dataset containing 9 subjects, we follow the setting of [52, 53, 190] where the depth map and skeleton pairs of 8 subjects are used for training and the remaining data pairs (of one subject) are used for testing. We repeated the experiments for each subject left out and the observed accuracies are averaged. For both Big Hand 2.2M and MSRA, the output space represents 21 skeleton joints. For the ICVL dataset, we use the ground-truth annotations (21 skeleton joints) provided by Tang et al. [195, 244]. This model differs from the 16-joint model provided by the authors of the original data [194]. While in principle, our model can be applied to any output configurations, we adopt this 21-joint model for simplicity of evaluation and to facilitate the cross-dataset transfer experiments (to be discussed shortly). The combination of training and testing sets also follows from [195, 244]: 16,008 frames are used for training while the remaining 1,596 frames are used in testing. For the NYU dataset, we adopt the experimental settings of [129, 174, 214]: 72,757 frames and 8,252 frames are used for training and testing, respectively. Following their experimental settings, we estimate the 14 target skeleton joints (out of 36 joints in the original datasets). For comparison, we adopt several state-of-the-art methods that share the same evaluation protocol: For Big Hand 2.2M, we compare with CNN estimator employed by Yuan et al. [244] which constitutes our baseline HPE. We also evaluate two existing generative hand model-based approaches: FORTH [130] and Intel RealSense SR300 camera [1]. On ICVL, we compare with Sun et al.'s cascaded refinement algorithm (denoted as Sun et al.) [190] and Tang et al.'s hierarchical decision forests-based algorithm (Tang et al.) [195] which constitutes the state-of-the-art on this benchmark. For MSRA, we compare with two state-of-the-art methods, Ge et al. [52] and Ge et al. [53]). Both algorithms [52, 53] adopt the multi-view approach and therefore, they are especially effective for MSRA that covers diverse view points. For the NYU dataset, in addition to Sun et al.'s cascade algorithm [129] and Ge et al.'s multi-view approach [214], we compare with two generative model-based algorithms (Wan et al. [214]) constituting the-state-of-the-art on this dataset. All components of our networks were implemented with the Torch library and they are trained and evaluated on an Intel 3.40 GHz i7 machine with two NVIDIA GTX 1070 graphic processing unit (GPU)s. Training our network on 10 times augmented Big Hand 2.2M dataset takes 3-4 days (100 epochs). At

testing, our algorithm processes 300 frames per second using the GPU.

System evaluation. We use a commonly used criteria for hand pose estimates [197]: the *proportion (in %) of the frames with all joints error*$<\epsilon$ (in Euclidean distance per joint) being smaller than a tolerance parameter ϵ. Figure 3.6 shows the results: For all benchmarks, our algorithm ('Ours' in Fig. 3.6 and Table 3.1(a)) constantly improved upon the baseline HPE (Yuan et al.'s CNN estimator [244]) by a significant margin. In comparison to Tang et al.'s algorithm (Tang et al. (ICCV 2015)) [195], our algorithm shows higher and lower accuracies on NYU (Fig. 3.6(e)) and ICVL (Fig. 3.6(b)), respectively confirming the complementary nature of the two approaches. On MSRA containing a wide range of camera views but limited shapes and poses (see Sec. 3.2), Ge et al.'s multi-view-based approach (Ge et al. CVPR 2017) [53] achieved the best results, followed by our algorithm. Overall, our algorithm outperforms or is on par with state-of-the-art methods.

Cross-dataset experiments. In principle, the space of (augmented) skeletons is independent of specific datasets and representations and therefore, it can be shared across multiple benchmark datasets. We tested this possibility by applying our model trained on Big Hand 2.2M to ICVL and MSRA datasets: For MSRA, our model trained only on Big Hand 2.2M ('Ours transferred' in Fig. 3.6(c)) achieved the best results outperforming Ge et al.'s state-of-the-art model [53]: MSRA is limited in the range of poses (only 17 gestures), which can be compensated by transferring skeletons augmented from the much larger Big Hand 2.2M dataset. For ICVL, the transferred version is slightly worse than the original but still outperforms several existing algorithms.

Evaluation of design choices. Our approach enables 1) to easily augment skeleton datasets and 2) to transfer them consistently to depth maps. This approach was facilitated by training the HPE and HPG in a single unified criteria guided by the paired (P) and unpaired (U) data. To gain an insight into the contribution of each algorithm component, we evaluated the corresponding variations of our final algorithm: Table 3.1(a) shows that each component of our algorithm indeed makes a significant contribution to building a system as a whole: Skeletal data augmentation helps improve the performances of both HPG and HPE. Even without data augmentation, jointly

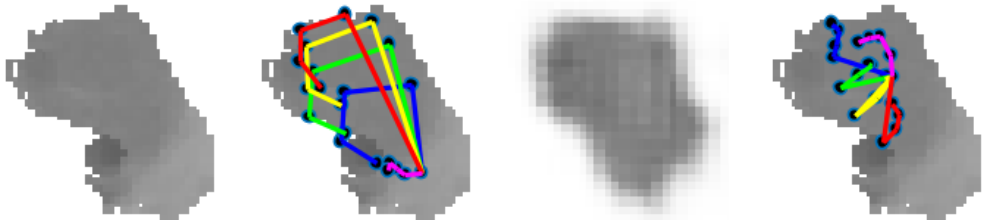
Table 3.1: *Evaluation of design choices: (a) Test errors of our HPG (unitless) and HPE (in mm) under varying design conditions: f^G (baseline) and f^E (baseline): HPG and HPE trained independently on the paired dataset P, respectively; f^E (w/o aug.; refine): HPE trained only on P pairs (Algorithm 5); f^E and f^E (w/o refine): HPEs trained (with skeleton augmentation) with and without the refinement step at testing, respectively; f^G (w/o aug.; refine) and f^G (w/o refine): HPGs trained jointly with f^E (w/o aug.; refine) and f^E, respectively. (b) Test error of HPE on* Big Hand 2.2M *with varying numbers and types of skeleton augmentation.*

(a)

Configuration	Big Hand 2.2M	ICVL	MSRA	NYU
f^G (baseline)	0.151	0.588	0.482	0.451
f^G (w/o aug.; refine)	0.124	0.516	0.470	0.415
f^G (w/o refine)	0.102	0.486	0.438	0.396
f^E (baseline)	17.1	12.1	16.3	17.3
f^E (w/o aug.; refine)	15.7	10.4	14.4	16.4
f^E (w/o refine)	14.1	9.1	13.1	14.9
f^E	13.7	8.5	12.5	14.1

(b)

Configuration	Error (mm)
HPE baseline	17.1
Ours (w/o aug.; refine)	15.7
Ours (w/ in-plane-rot 10x.; w/o aug.; refine)	14.9
Ours (5× aug.; w/o refine)	15.1
Ours (10× aug.; w/o refine)	14.1
Ours (20× aug.; w/o refine)	14.0
Ours (w/ in-plane-rot; 10x aug.; w/o refine)	12.5



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

Figure 3.7: *A failure example (MSRA dataset): (a) input depth map, (b) ground-truth skeleton overlaid on the input, (c) new depth map synthesized by our generator based on the ground-truth skeleton, and (d) skeleton estimated by our hand pose estimator overlaid on the depth map. When the input represents a significantly different skeletal pose from the database, the corresponding synthesized depth map (c) is blurry even when based on the ground truth, leading to a large pose estimation error (d).*

training HPE and HPG within our framework, already improves the performance. Refining the prediction during testing as guided by HPG and HPD_Y plus multi-view synthesis, further significantly improves the pose estimation accuracy. Figure 3.6(f) confirms the importance of joint HPE/HPG training: A simpler data augmentation alternative to our joint training approach is to hold the HPG f^G trained on P and fixed, then individually train HPE on the resulting augmented skeletons U and transferred depth maps $f^G(U)$ (HPE w/aug. given fixed HPG), while this approach improves upon the HPE trained on P (HPE baseline), our final algorithm shows much more significant improvements.

Influence of the size and type of skeleton augmentation. Table 3.1(b) shows that the HPE test error constantly decreases as the augmented dataset grows, confirming the importance of dataset augmentation. The accuracy gain saturates when the augmented set is around 10

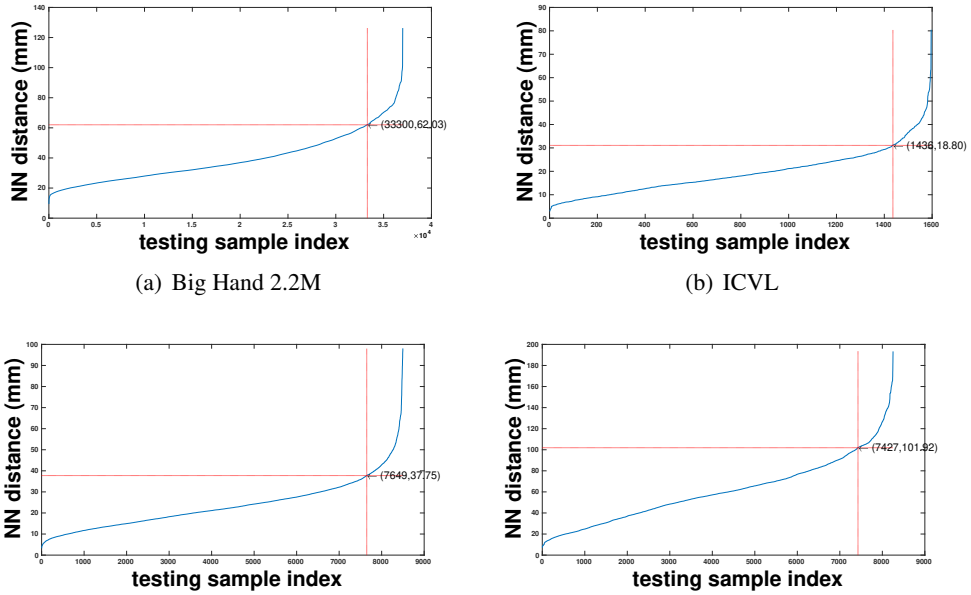
times larger than the original suggesting $M = 10$ as a good trade-off between the (training) computational efficiency and accuracy. Finally, we observe that our approach of skeleton-based data augmentation and transfer is complementary to traditional view-dependent data augmentation approaches: The straightforward in-plane rotation approach applied to the skeleton and depth map pairs also significantly improve the performance of hand pose estimation, and combining the two approaches further boosts the accuracy.

3.4 Qualitative evaluation

The quantitative analysis in previous section has demonstrated that benefiting from the new skeletal augmentation strategy, our algorithm significantly improves upon the state-of-the-art hand pose estimation algorithms. In this section, we further demonstrate with examples that such accuracy gain is more pronounced when the test data entries are *distinct* from the original training data, confirming the effectiveness of skeletal augmentation.

To facilitate the analysis, we categorize the test entries based on their *projection distances* on the (un-augmented) training set, *i.e.* distances to the closest training data points (in the skeleton space):[1] Each test entry is classified as *hard* if its projection distance belongs to the larger 10% of projection distances of all test data entries. All other test patterns are considered *easy*. Figure 3.8 displays the distribution of calculated projection distances and the corresponding categories and Figs. 3.10 and 3.11 show hand pose estimation, and auxiliary depth-map reconstruction results on *hard* and *easy* examples, respectively. For patterns in the *easy* category, both existing algorithms and the proposed algorithm generate good skeleton estimates (Fig. 3.11). However, hard examples deviate significantly from the training set and therefore, they pose great challenges for existing algorithms as in general, the learned estimators generalize to the extent covered by training data. For instance, when applied to *hard* patterns, the baseline HPG (HPG; Fig. 3.10(c)) trained without our data augmentation process generated overall blurry depth maps, smearing

[1] We use the ground-truth skeletons to ease the visualization of the results. Our algorithm does not require ground-truth annotations.



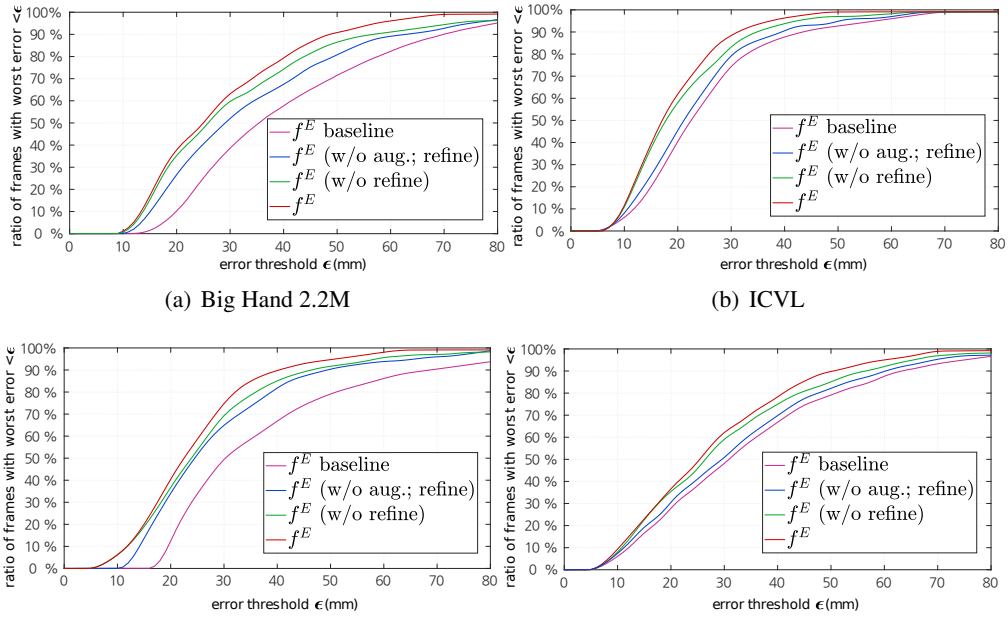
(a) Big Hand 2.2M (b) ICVL

(c) MSRA (d) NYU

Figure 3.8: *Distances of test data points to the closest un-augmented training entries (measured in skeletal Euclidean distance). The test points are sorted in the ascending distance order. Thresholding data points based on their respective distances categorizes them to* easy *and* hard *classes (highlighted as the crossing points of the horizontal and vertical red lines).*

details in the finger boundaries (1st, 3rd, and 6th examples from top to bottom). Naïvely training a HPE on this dataset resulted in spurious skeletal pose estimates (Fig. 3.10(e)). In contrast, our HPG results (both with and without refinement in testing) faithfully reconstructed the depth maps which are much sharper and crisper in locating finger boundaries. This is in accordance with the corresponding improved skeletal pose estimation results (Fig. 3.10(f-e)). We observe a similar tendency in the results of other state-of-the-art hand pose estimation algorithms (Fig. 3.10(bottom)). To summarize, by explicitly widening the training set coverage of skeletal shape and view, our algorithm achieved better generalization performance than existing algorithms in both skeletal pose estimation and depth map synthesis tasks.

Figure 3.9 shows the performance of our HPE with different design configurations. Our skeleton augmentation and testing refinement strategies both contribute significantly to higher accuracy over the baseline.



(a) Big Hand 2.2M (b) ICVL

(c) MSRA (d) NYU

Figure 3.9: *Performance of varying HPE design configurations as* ratio of frames with worst error<ϵ *(the proportion of frames whose worst joint error is less than ϵ). These results correspond to Table 1 of the main paper where accuracy is measured in Euclidean distance (to the ground-truth).*

3.5 Conclusion

Existing depth-based hand (pose estimation) datasets are limited in their extent in shapes, poses, and/or camera viewpoints. Traditional data augmentation approaches directly manipulate the depth map and skeleton pairs and therefore, their augmentation capabilities are limited to simple 2D view-dependent manipulations. We introduced a framework that extends this domain to a variety of hand shapes and poses. Our algorithm enables to augment data only in the skeleton space where data manipulation is intuitively controlled and greatly simplified and thereafter, automatically transfers them to realistic depth maps. This was made possible by jointly training the hand pose estimator and hand pose generator in a single unified framework. The resulting algorithm significantly outperforms or is on par with state-of-the-art hand pose estimation algorithms.

Our skeleton augmentation process enables the generator (and the corresponding pose estimator)

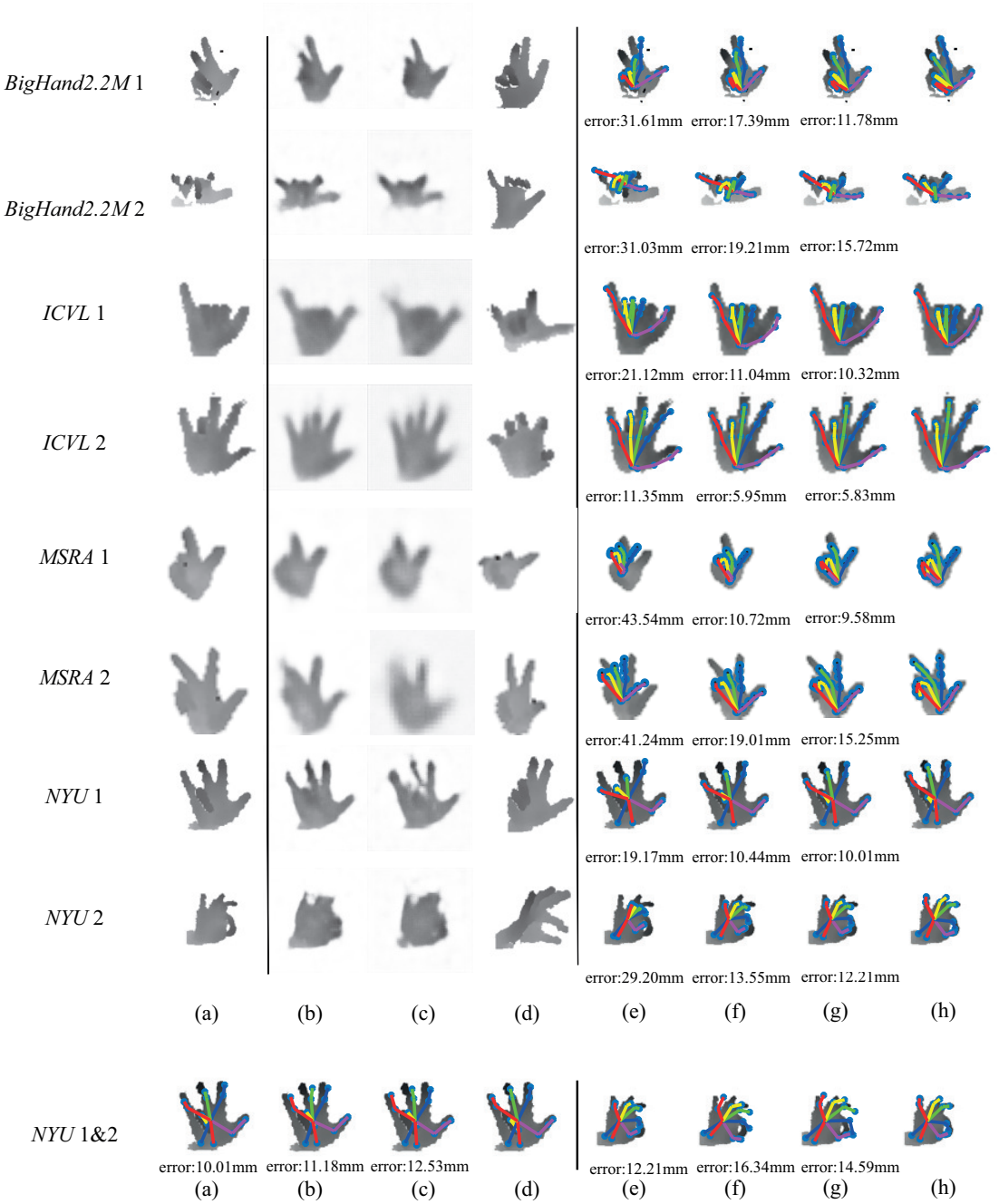


Figure 3.10: *Hand pose generation and estimation results for* hard *examples. (top) eight test example depth maps (two from each dataset): (a) input depth map, (b) reconstruction by our HPG, (c) reconstruction by the HPG baseline, (d) nearest (in skeletons) depth map in the training set, (e) Yuan* et al.*'s algorithm [244], (f) our HPE (w/o refine.), (g) our HPE (w refine.), (h) ground-truth; (bottom) hand pose estimation results for* NYU1 *(left) and* NYU2 *(right) from (a) and (e) our HPE (w refine), (b) and (f) Oberweger* et al.*'s algorithm [129], (c) and (g) Wan* et al.*'s algorithm [214], and (d) and (h) ground-truth.*

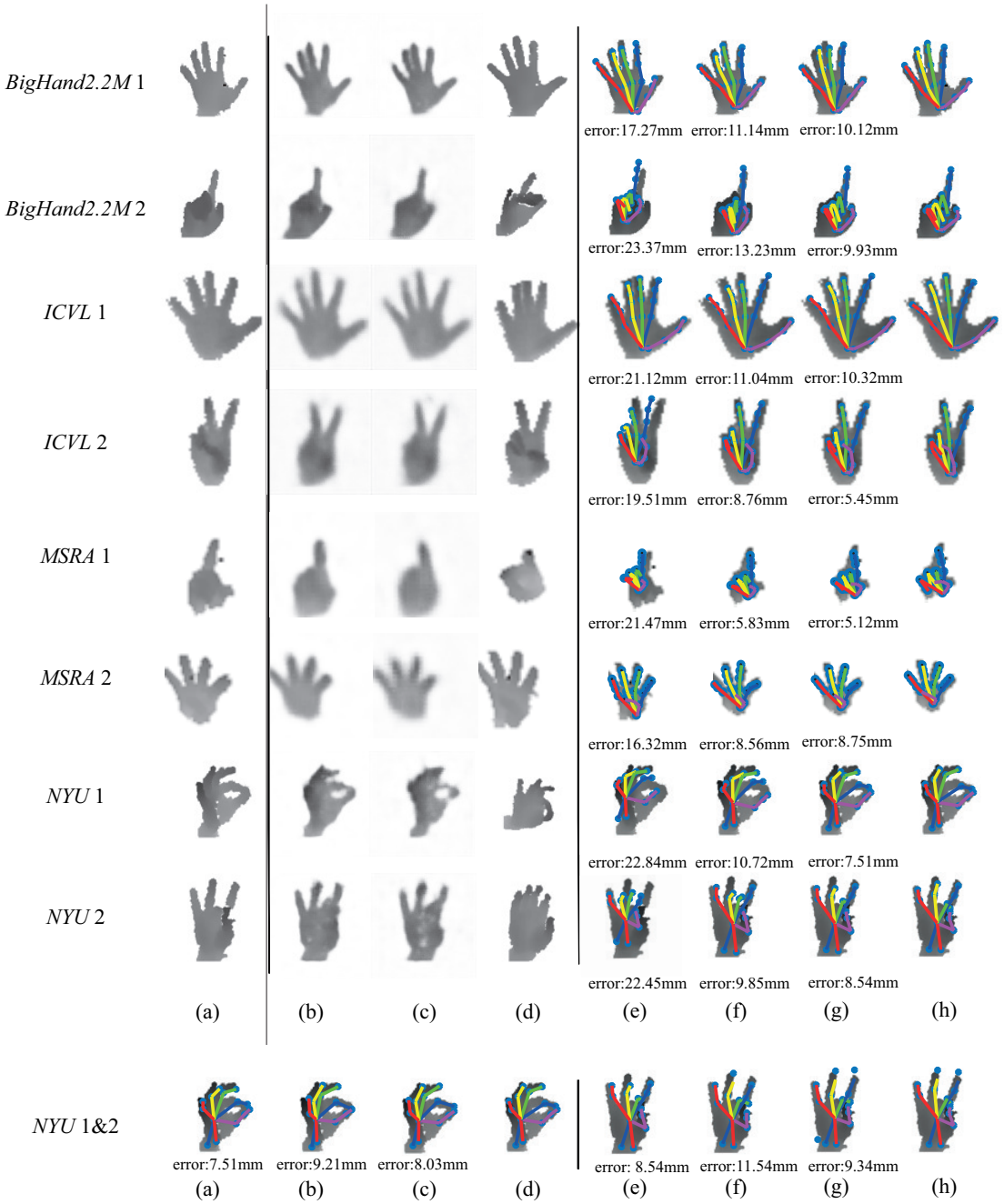


Figure 3.11: *Hand pose generation and estimation results for* easy *examples. (top) eight test example depth maps (two from each dataset): (a) input depth map, (b) reconstruction by our HPG, (c) reconstruction by the HPG baseline, (d) nearest (in skeletons) depth map in the training set, (e) Yuan* et al.*'s algorithm [244], (f) our HPE (w/o refine.), (g) our HPE (w refine.), (h) ground-truth; (bottom) hand pose estimation results for* NYU1 *(left) and* NYU2 *(right) from (a) and (e) our HPE (w refine), (b) and (f) Oberweger* et al.*'s algorithm [129], (c) and (g) Wan* et al.*'s algorithm [214], and (d) and (h) ground-truth.*

to absorb a wide range of skeleton variations. However, when the input test entries exhibit significantly-different skeletal poses from any of the (original+augmented) training database entries, the corresponding synthesized depth maps tend to be blurry, indicating an ambiguity. This can eventually lead to pose estimation errors (Fig. 3.7). Future work should address this, e.g., by *actively* sampling such difficult poses in the augmented skeleton space during training.

CHAPTER 4

DENSE POSE ESTIMATION VIA EXPLICIT FULL 3D DATA RECONSTRUCTION AND AUGMENTATION

Contents

In this chapter, we try to reconstruct the full 3D meshes of hands from single 2D RGB images. The main contribution of this chapter is related to relieving the "insufficient data issue" mentioned in chapter 1. In the hand pose estimation domain, as far as we are aware, there is no dataset having 2D RGB and 3D mesh pairs. Thus, training a network with such full

supervision is non-trivial. To tackle the challenge, we adopt a compact parametric 3D hand model that represents deformable and articulated hand meshes. To obtain the 3D mesh model fitted to RGB images, we investigate and contribute in three ways: 1) Neural rendering: inspired by recent work on human body pose estimation works, our hand mesh estimator (HME) is implemented by a neural network and a differentiable renderer, supervised by 2D segmentation masks and 3D skeletons. HME demonstrates good performance for estimating diverse hand shapes and improves pose estimation accuracies. 2) Iterative testing refinement: Our fitting function is differentiable. We iteratively refine the initial estimate using the gradients, in the spirit of iterative model fitting methods like ICP. The idea is supported by the latest research on human body pose estimation. 3) Self-data augmentation: collecting sized RGB-mesh (or segmentation mask)-skeleton triplets for training is a big hurdle. Once the model is successfully fitted to input RGB images, its meshes i.e. shapes and articulations, are realistic, and we augment view-points on top of estimated dense hand poses. Experiments using three RGB-based benchmarks show that our framework offers beyond state-of-the-art accuracy in 3D pose estimation, as well as recovers dense 3D hand shapes. Each technical component above meaningfully improves the accuracy in the ablation study.

4.1 Motivation

Recovering hand poses and shapes from images enables many real-world applications, e.g. hand gesture as a primary interface for AR/VR. Most existing methods have focused on recovering sparse hand poses i.e. skeletal articulations from either a depth or RGB image. However, estimating dense hand poses including 3D *shapes* (Fig. 4.1) is important as it helps understand e.g. human-object interactions [5,8,26,50] and perform robotic grasping, where surface contacts are essential. Discriminative methods based on CNN have shown very promising performance in estimating 3D hand poses either from RGB images or depth maps (More detailed analysis are given in Sec. 2.1.3). However, the predictions are based on coarse skeletal representations, and no explicit kinematics and geometric mesh constraints are often considered. On the other

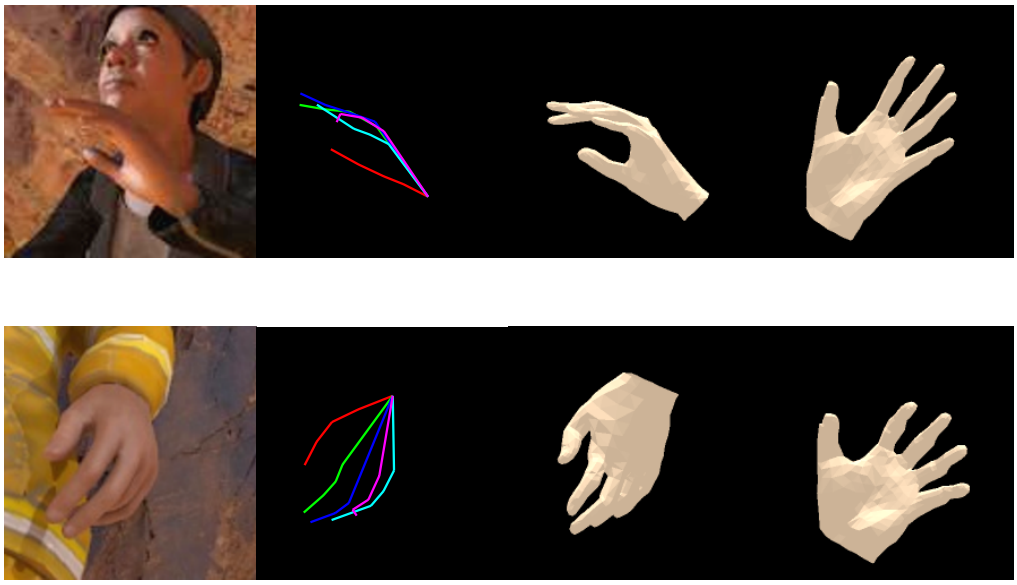


Figure 4.1: *Dense hand pose estimation examples. Our system estimates 3D shapes, as well as articulations and viewpoints. Left to right: input images,* coarse *skeletal representations,* dense *hand pose representations, and recovered hand shapes in a canonical articulation/viewpoint. Dense pose estimation provides a richer description of hands and improves the pose estimation accuracy.*

hand, establishing a personalized hand model requires a generative approach that optimizes the hand model to fit to 2D images. Optimization-based methods, besides their complexity, are susceptible to local minima and the personalized hand model calibration contradicts the generalization ability for hand shape variations. Sinha et al. [181] learn to generate 3D hand meshes from depth images. A direct mapping between a depth map and its 3D surface is learned; however, the accuracy shown is limited. Malik et al. [109] incorporate a 3D hand mesh model to CNN and learn a mapping between depth maps and mesh model parameters. They further raycast 3D meshes with various viewpoint, shape and articulation parameters to generate a million-scale dataset for training CNN. Our work tackles the similar but in RGB domain and offers an iterative mesh fitting using a differentiable renderer [79]. This improves once estimated mesh parameters using the gradient information at testing. Our framework also allows indirect 2D supervisions (2D segmentation masks, keypoints) instead of using 3D mesh data.

3D keypoints involve the full xyz world coordinate values in contrast to 2D keypoints whose coordinate values are defined using the uv coordinates without a depth value. While 2D keypoint

detection is well established in the RGB domain [176], estimating 3D keypoints from RGB images is less trivial. Recently, several methods were developed [18, 70, 133, 256]. While directly lifting 2D estimations to 3D space was attempted in [256], 2.5D depth maps are first estimated as clues for 3D lifting in state-of-the-art techniques [18, 70]. In this chapter, we exploit a deformable 3D hand mesh model, which inherently offers a full description of both hand shapes and articulations, 3D priors for recovering depths, and self-data augmentation. Different from purely generative optimization methods e.g. [133], we propose a method based on a neural renderer and CNNs.

Our contribution is in implementing a full 3D dense hand pose estimation (DHPE) using single RGB images via a neural renderer. Our technical contributions are largely threefold:

1) DHPE is composed of convolutional layers that first estimate 2D evidences (RGB features, 2D keypoints) from an RGB image and then estimate the 3D mesh model parameters. Since RGB/mesh pairs are not sufficient, the network is learned to fit the 3D model using 2D segmentation masks and skeletons as supervision, similar to recent work on human body pose estimation [77, 139, 209], via neural renderer. The dense shape estimation helps improve the pose estimation.

2) At the testing time, we iteratively refine the initial 3D mesh estimation using the gradients. The gradients are computed over self-supervisions by comparing estimated 3D meshes to predicted 2D evidences, as ground-truth labels are not available at testing. While previous work [209] fixed 2D skeletons/segmentation masks during the refinement step, we recursively improve 2D skeletons and exploit the improved skeletons and feature similarities for 2D masks.

3) To further deal with limited annotated training data, especially for diverse shapes and view-points, we supply fitted meshes and their 2D projected RGB maps by varying shapes and view-points, to fine-tuning the network. Purely synthetic data imposes a synthetic-real domain gap, and annotating 3D meshes or 2D segmentation masks of real images is difficult. Once the model is successfully fitted to input RGB images, its meshes, i.e. shapes and poses are close-to-real.

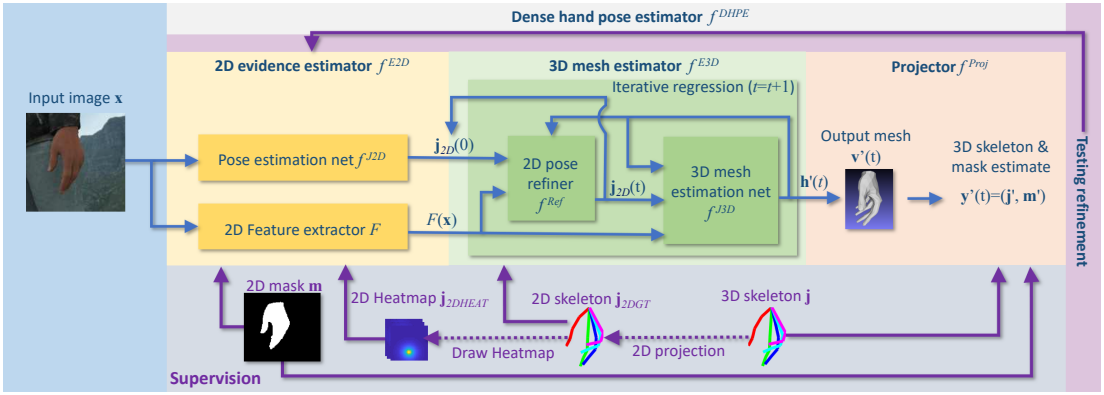


Figure 4.2: *Schematic diagram of the proposed DHPE framework. Our DHPE receives an input RGB image* $\mathbf{x}$ *and estimates the corresponding hand shape and pose as parameters* $\mathbf{h}$ *of the MANO [161] hand model. Training DHPE is guided via 1) an additional projector* f^{Proj} *that enables us to provide supervision via 3D skeletons* $\mathbf{j}$ *and foreground segmentation masks* $\mathbf{m}$*; 2) decomposing the DHPE into the 2D evidence estimator* f^{E2D} *and 3D mesh estimator* f^{E3D} *which stratifies the training via the intermediate 2D feature estimation step. At testing, once the output mesh parameter* $\mathbf{h}'$ *is estimated, it is iteratively refined via enforcing its consistency over intermediate 2D evidences* $F(\mathbf{x})$ *and* $\mathbf{j}_{2D}$*.*

The proposed method can also be seen as a hybrid method [90, 205, 238] combining merits of discriminative and generative approaches.

4.2 Proposed dense hand pose estimator

Our goal is to construct a pose estimator $f^P: X_C \rightarrow Y_P$ that maps an input RGB image $\mathbf{x} \in X_C$ to the corresponding estimated 3D hand mesh $\mathbf{v} \in Y_P$. Each input is given as an RGB pixel array of size 224×224 while an output mesh corresponds to 3D positions of 778 vertices encoding 1,538 triangular faces. As the resultant output is the mesh, we call the pose estimator as the HME $f^P := f^{HME}$. Instead of directly generating a mesh $\mathbf{v}$ as a 778×3-dimensional raw vector, our HME estimates a 63-dimensional parameter vector $\mathbf{h}$ that represents $\mathbf{v}$ by adopting the MANO 3D hand mesh model [161] (Sec. 4.2.2). Once a hand mesh $\mathbf{v}$ (or equivalently, its parameter $\mathbf{h}$) is estimated, the corresponding skeletal pose $\mathbf{j}$ consisting of 21 3D joint positions can be recovered via a 3D skeleton regressor f^{Reg} discussed shortly. This 21 3D joints are in the same topology with the skeleton model we used in the chapter 3.

Learning the HME can be cast into a standard multivariate regression problem if a training

Table 4.1: *Notational summary*

$\mathbf{y}$	$\mathbf{y} = (\mathbf{j}, \mathbf{m}) \in \mathcal{Y} = (\mathcal{J}, \mathcal{M}) \subset \mathbb{R}^{(21\times 3)\times(224\times 224)}$
f^{DHPE}	dense hand pose estimator ($f^{DHPE} = f^{Proj} \circ f^{HME}$)
f^{Proj}	projection operator ($f^{Proj} = [f^{Reg}, f^{Ren}]$)
f^{Reg}	3D skeleton regressor
f^{Ren}	renderer

database of pairs of input images and the corresponding ground-truth meshes are available. However, we are not aware of any existing database that provides such pairs. Inspired by the success of existing human body reconstruction work [77, 139, 209], we take an indirect approach by learning a DHPE which combines the HME and a new *projection operator*. The projection operator consists of a 3D *skeleton regressor* and a *renderer* which respectively recover a 3D skeletal pose ($\mathbf{j} \in \mathcal{J}$) and a 2D foreground hand segmentation mask ($\mathbf{m} \in \mathcal{M}$) from a hand mesh $\mathbf{v}$. Table 4.1 shows our notation, and Eq. 4.1 and Fig. 4.2 summarize the decomposition of DHPE:

$$X_C \overbrace{\overbrace{\xrightarrow{f^{E2D}} \mathcal{Z} \xrightarrow{f^{E3D}}}^{f^{HME}=f^{E3D}\circ f^{E2D}} Y_P \xrightarrow{f^{Proj}=(f^{Reg},f^{Ren})}}^{f^{DHPE}=f^{Proj}\circ f^{HME}} \mathcal{Y} = (\mathcal{J}, \mathcal{M}). \tag{4.1}$$

Extending the skeleton regressor provided by the MANO model [161], our skeleton regressor f^{Reg} maps a hand mesh $\mathbf{v}$ to its skeleton $\mathbf{j}$ consisting of 21 3D joint positions. It is a linear regressor implemented as three matrices of size 778×21 (see Sec. 2.1 of the supplemental for details).

Our renderer f^{Ren} generates the foreground hand mask $\mathbf{m}$ by simulating the camera view of $\mathbf{x}$. We adopt the differentiable neural renderer proposed by Kato et al. [79].

By construction, the projection operator respects the underlying camera (via f^{Ren}) and hand shape (via f^{Reg}) geometry and it is held fixed throughout the entire training process. This facilitates the training of f^{HME} indirectly via training f^{DHPE}. However, even under this setting, the problem still remains challenging as estimating a 3D mesh given an RGB image is a seriously ill-posed problem. Adopting recent human body pose estimation approaches [139, 209], we

further stratify learning of $f^{DHPE}: X_C \rightarrow Y$ by decomposing $f^{HME}: X_C \rightarrow Y_P$ into a 2D evidence estimator $f^{E2D}: X_C \rightarrow \mathcal{Z}$ and a 3D mesh estimator $f^{E3D}: \mathcal{Z} \rightarrow Y_P$.

Our 2D evidence $\mathbf{z} \in Z$ consists of a 42-dimensional 2D skeletal joint position vector $\mathbf{j}_{2D}$ (21 positions $\times$ 2; as in [18, 70]) and a 2,048-dimensional 2D feature vector $F(\mathbf{x})$ (Eq. 4.2). The remainder of this section provides details of these two estimators.

4.2.1 2D evidence estimator $f^{E2D} = (F, f^{J2D})$

Silhouettes (or foreground masks) and 2D skeletons have been widely used as the mid-level cues for estimating 3D body meshes [78, 139, 209]. However, for hands, accurately estimating foreground masks from RGB images is challenging due to cluttered backgrounds [84, 256]. We observed that naïvely applying the state-of-the-art foreground segmentation algorithms (e.g. [256]) often misses fine details, especially along the narrow finger regions (see Fig. 4.5 for examples) and this can significantly degrade the performance of the subsequent mesh estimation step. We bypass this challenge by learning instead, a 2D foreground feature extractor F: $F(\mathbf{x})$ encapsulates the textural and shape information of the foreground regions in $\mathbf{x}$.

Our 2D feature extractor F is trained to focus on the foreground regions by minimizing the deviation between the features extracted from the entire image $\mathbf{x}$ and the foreground region $\mathbf{x} \odot \mathbf{m}$ extracted via the ground-truth mask $\mathbf{m}$ (see Fig. 4.3d): The training loss (per data point) for F is given as

$$L_{Feat}(F) = \|F(\mathbf{x}) - F(\mathbf{x} \odot \mathbf{m})\|_2^2, \tag{4.2}$$

where $\odot$ denotes element-wise multiplication. F employs the ResNet-50 [64] architecture whose output is a 2,048-dimensional vector.

Estimating 2D skeletal joints from an RGB image has been well studied. Similarly to [18, 70], we embed the state-of-the-art 2D pose estimation network f^{J2D} [226, 256] into our framework. The initial weights provided by the authors of [256] are refined based on the 2D joint estimation loss:

(a) *(b)* *(c)* *(d)*

Figure 4.3: *A 2D evidence estimation example. (a) input image* $\mathbf{x}$*, (b) ground-truth 2D segmentation mask* $\mathbf{m}$ *of* $\mathbf{x}$*, (c) 2D skeletal position heat map of the finger tip of middle finger overlaid on* $\mathbf{x}$*, and (d) masked image* $\mathbf{x} \odot \mathbf{m}$*.*

$$L_{J2D}(f^{J2D}) = \|f^{J2D}(\mathbf{x}) - \mathbf{j}_{2DHeat}\|_2^2. \tag{4.3}$$

The output of f^{J2D} is a matrix of size $21 \times 32 \times 32$ encoding 21 heat maps (per joint) of size 32×32. $\mathbf{j}_{2DHeat}$ is the ground-truth heat map. Given the estimated heat maps, 2D skeleton joints $\mathbf{j}_{2D}$ are extracted by finding the maximum for each joint. Figure 4.3(b) shows an example of 2D evidence estimation.

4.2.2 3D mesh estimator $f^{E3D} = (f^{J3D}, f^{Ref})$

Taking the 2D evidence $\mathbf{z}$ ($2,048$-dimensional feature vector $F(\mathbf{x})$ and 21×3-dimensional 2D skeleton $\mathbf{j}_{2D}$) as input, the 3D mesh estimation network f^{J3D} constructs parameters of a deformable hand mesh model and a camera model. We use the MANO model representing a hand mesh based on 45-dimensional pose parameters $\mathbf{p}$ and 10-dimensional shape parameters $\mathbf{s}$ [161]. The original MANO framework uses only 6-dimensional PCA subspace of $\mathbf{p}$ for computational efficiency. However, we empirically observed that to cover a variety of hand poses, all 45-dimensional features are required: We use the linear blend skinning formulation as in the SMPL model [102].

Given the MANO parameters $\mathbf{p}$ and $\mathbf{s}$, a mesh $\mathbf{v}$ is synthesized by conditioning them on the hypothesized camera $\mathbf{c}$, which comprises of the 3D rotation (in quaternion space) $\mathbf{c}_q \in \mathbb{R}^4$, scale

$\mathbf{c}_s \in \mathbb{R}$, and translation $\mathbf{c}_t \in \mathbb{R}^3$: An initial mesh is constructed by combining MANO's PCA basis vectors using the shape parameter $\mathbf{s}$, rotating the bones according to the pose parameter $\mathbf{p}$, and deforming the resulting surface via linear blend skinning. The final mesh $\mathbf{v}$ is then obtained by globally rotating, scaling, and translating the initial mesh according to $\mathbf{c}_q$, $\mathbf{c}_s$, and $\mathbf{c}_t$, respectively. The entire mesh generation process is differentiable. Under this model, our 3D mesh estimation network f^{J3D} is implemented as a single fully-connected layer of 2153×63 weights and it estimates a 63-dimensional mesh parameter:

$$\mathbf{h} = [\mathbf{p}, \mathbf{s}, \mathbf{c}_q, \mathbf{c}_s, \mathbf{c}_t]^\top. \tag{4.4}$$

Iterative mesh refinement via back-projection. Adopting Kanazawa et al.'s approach [77], instead of estimating $\mathbf{h}$ directly, we iteratively refine the initial mesh estimate $\mathbf{h}(0)$ by recursively performing regression on the parameter offset $\Delta\mathbf{h}$: At iteration t, f^{J3D} takes $\mathbf{z}$ and the current mesh estimate $\mathbf{h}(t)$, and generates a new offset $\Delta\mathbf{h}(t)$:

$$\mathbf{h}(t+1) = \mathbf{h}(t) + \Delta\mathbf{h}(t). \tag{4.5}$$

At $t = 0$, $\mathbf{h}(0)$ as an input to f^{J3D}, is constructed as a vector of zeros except for the entries corresponding to the pose parameter $\mathbf{p}$ which is set as the mean pose of the MANO model. We fix the number of iterations at 3 as more iterations did not show any noticeable improvements in preliminary experiments.

In [77], each offset prediction $\Delta\mathbf{h}(t)$ is built based only on 2D features $F(\mathbf{x})$. Inspired by the success of [18, 70], we additionally use 2D skeletal joint $\mathbf{j}'_{2D}$ as input. However, as an estimate, $\mathbf{j}'_{2D}(0)$ (obtained at iteration 0) is inaccurate, causing errors in the resulting $\Delta\mathbf{h}(t)$ estimate. Then, these errors accumulate over iterations (Eq. 4.5) lessening the benefit of the entire iterative estimation process. We address this by an additional 2D pose refiner f^{Ref} which iteratively refines the 2D joint estimation $\mathbf{j}'_{2D}(t)$ via *back-projecting* the estimated mesh $\mathbf{h}'(t)$: At t, f^{Ref} receives the estimated mesh parameter $\mathbf{h}'(t)$, 2D feature $F(\mathbf{x})$, 3D skeleton $f^{Reg}(\mathbf{v}')$, and $\mathbf{j}'_{2D}(t)$, and generates a refined estimate $\mathbf{j}'_{2D}(t+1)$. The 2D pose refiner f^{Ref} is implemented as a single fully connected layer of size 2216×42 and it is trained using the loss L_{Ref} by closing its

prediction close to the ground-truth 2D skeletons ($\mathbf{j}_{2DGT}$):

$$L_{Ref} = \left\| f^{Ref}\left([\mathbf{j}'_{2D}(t), F(\mathbf{x}), \mathbf{h}'(t), f^{Reg}(\mathbf{v}')]\right) - \mathbf{j}_{2DGT} \right\|_2^2 . \tag{4.6}$$

In the experiments, we demonstrate that 1) using both 2D features and 2D skeletons (Fig. 4.4(e): 'Ours (w/o Test. ref. and Refiner f^{Ref})') improves performance over Kanazawa et al.'s 2D feature-based framework [77] (Fig. 4.4(e): 'Ours (w/o Test. ref., f^{Ref} and 2D losses (L_{Feat}, L_{J2D})'); 2) explicitly building the refiner f^{Ref} under the auxiliary 2D joint supervision (Fig. 4.4(e): 'Ours (w/o Test. ref.)') further significantly improves performance. More importantly, the refiner takes a crucial role in our testing refinement step (Sec. 4.2.5) which improves a once predicted 3D mesh $\mathbf{v}'$ by enforcing its consistency with the corresponding 2D joint evidence $\mathbf{j}'_{2D}(t)$ (as well as other intermediate results) throughout the iteration (Eq. 4.12).

4.2.3 Skeleton regressor f^{Reg} and neural renderer f^{Ren}

Skeleton regressor f^{Reg} receives a (predicted) mesh consisting of 778 vertices $\mathbf{v} \in Y_P \subset \mathbb{R}^{778\times 3}$ and generates 21 skeletal joint positions $\mathbf{j} \in \mathcal{J} \subset \mathbb{R}^{21\times 3}$. Our regressor builds upon the original MANO regressor which is implemented as three multi-dimensional linear regressors, each aligned with a coordinate axis [161]: The $x-$axis regressor receives the $x-$coordinate values of $\mathbf{v}$ and synthesizes the $x-$axis coordinate values of 16 skeletal joints. The $y-$axis and $z-$axis regressors are constructed similarly. In this way, the original MANO regressor estimates only 16 joint positions. The remaining five, finger tip positions are estimated by simply *selecting* a point of $\mathbf{v}$ that corresponds to a finger tip, per axis.[1] These additional regressors are implemented for each finger tip and for each axis, as a 778-dimensional vector where all elements are zero except for the entry corresponding to the vertex location of the corresponding finger tip, where value 1 is assigned. As a whole, our regressor f^{Reg} is represented as three matrices of size (778×21): As a linear regressor, f^{Reg} is differentiable with respect to its input and output arguments.

[1] Unlike other skeletal joint locations which lie inside the mesh $\mathbf{v}$, finger tips lie on (the surface) $\mathbf{v}$. Therefore, selecting a point on $\mathbf{v}$ can give a good joint location estimate.

Neural renderer f^{Ren} is implemented by adopting the work of [79]. We only used their binary segmentation mask renderer as our f^{Ren}, to differentiably obtain the binary segmentation masks from inferred 3D meshes.

4.2.4 Joint training

Training the 2D evidence estimator $f^{E2D} : X_C \rightarrow \mathcal{Z}$ and 3D mesh estimator $f^{E3D} : \mathcal{Z} \rightarrow Y_P$ given fixed projection operator $f^{Proj} : Y_P \rightarrow \mathcal{Y}$ is performed based on a training set consisting of input images, and the corresponding 3D skeleton joints and 2D segmentation masks $D = \{(\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^{l} \subset X_C \times \mathcal{Y}, \mathbf{y}_i = (\mathbf{j}_i, \mathbf{m}_i)$. Since all component functions of f^{E2D}, f^{E3D}, and f^{Proj} are differentiable with respect to the weights of F and f^{E3D}, they can be optimized based on on standard gradient descent-type algorithm: Our overall loss (per training instance $\mathbf{x}_i, \mathbf{y}_i$) is given as (see Eqs. 4.2 and 4.6):

$$\begin{aligned} L(f^{E3D}, F) = L_{Art}(f^{E3D}, F) + L_{Lap}(f^{E3D}, F) + L_{Feat}(F) \\ + \lambda L_{Sh}(f^{J3D}, F) + L_{Ref}(f^{J3D}, F). \end{aligned} \tag{4.7}$$

The articulation loss L_{Art} measures the deviation between the skeleton estimated from $\mathbf{x}_i$ and its ground-truth $\mathbf{j}_i$:

$$L_{Art} = \| [f^{DHPE}(\mathbf{x}_i)]_{\mathcal{J}} - \widehat{\mathbf{j}}_i \|_2^2, \tag{4.8}$$

where $[\mathbf{y}]_{\mathcal{J}}$ extracts the $\mathbf{j}$-component of $\mathbf{y} = (\mathbf{j}, \mathbf{m})$ and $\widehat{\mathbf{j}}$ spatially normalizes $\mathbf{j}$ similarly to [70, 256]: First, the center of each skeleton is moved to the corresponding middle finger's MCP position. Then each axis is normalized to a unit interval $[0, 1]$: The $x, y-$coordinate values are divided by g (=1.5 times the maximum of height and width of the tight 2D hand bounding box). The $z-$axis value is divided by $(z_{Root} \times g)/\mathbf{c}_f$ where z_{Root} is the depth value of the middle finger's MCP joint and $\mathbf{c}_f$ is the focal length of the camera. At testing, once normalized skeletons are estimated, they are inversely normalized to the original scale.

To facilitate the training of the skeleton regressor f^{Reg}, similarly to [70, 256], we explicitly normalize its output space $\mathcal{J}$: Our articulation loss L_{Art} measures the deviation between the

skeleton estimated from the training input $\mathbf{x}_i$ and the corresponding growth-truth $\mathbf{j}_i$:

$$L_{Art} = \|[f^{DHPE}(\mathbf{x}_i)]_{\mathcal{J}} - \widehat{\mathbf{j}}_i\|_2^2, \tag{4.9}$$

where $[\mathbf{y}]_{\mathcal{J}}$ extracts the $\mathbf{j}$-component of $\mathbf{y} = (\mathbf{j}, \mathbf{m})$.

Here, $\widehat{\mathbf{j}}$ spatially normalizes $\mathbf{j}$: First, a tight 2D hand bounding box is extracted from the corresponding ground-truth 2D skeleton of $\mathbf{x}_i$, and the center of the skeleton is moved to its middle finger's MCP position. Then, each axis is normalized to a unit interval $[0, 1]$: The $x, y-$coordinate values are divided by 1.5 times the maximum g of height and width of the bounding box. The $z-$axis value is divided by $z_{Root}/(\mathbf{c}_f \times g)$ where z_{Root} is the depth value of the middle finger's MCP joint and $\mathbf{c}_f$ is the focal length of the camera.

At testing, once normalized skeletons are generated, they are inversely normalized to the original scale based on the parameters g and z_{Root}.

Estimation of the bounding box size g. First, we use Zimmermann and Brox's hand detector [256] to infer bounding boxes in each video frame. The corner coordinates of the detected boxes are then temporally smoothed by taking an average over the past five frames.

For RHD, following Cai et al.'s experimental settings [18], the bounding boxes are extracted from the ground-truth 2D skeletons provided in the dataset, to facilitate a fair comparison with their algorithm.

Estimation of the hand depth z_{Root}. We use the 3D root depth estimation algorithm proposed by Iqbal et al. [70]. For RHD, for pair comparison with [18], the ground-truth depth values accompanying this dataset are used.

Our shape loss L_{Sh} facilitates the recovery of hand shapes as observed indirectly via projected 2D segmentation masks:

$$L_{Sh} = \left\|[f^{DHPE}(\mathbf{x}_i)]_{\mathcal{M}} - \mathbf{m}_i\right\|_2^2, \tag{4.10}$$

where $[\mathbf{y}]_{\mathcal{M}}$ extracts the $\mathbf{m}$-component of $\mathbf{y} = (\mathbf{j}, \mathbf{m})$.

The Laplacian regularizer L_{Lap} enforces spatial smoothness in the mesh $\mathbf{v}$. This helps avoid generating implausible hand meshes as suggested by Kanazawa et al. [78].

Hierarchical recovery of articulation and shapes. We observed that naïvely minimizing the overall loss L with a constant shape loss weight λ (Eq. 4.7) tends to impede convergence during training (Fig. 4.4(f)): Our algorithm simultaneously optimizes the mesh ($\mathbf{v}$) and camera ($\mathbf{c} = \{\mathbf{c}_q, \mathbf{c}_s, \mathbf{c}_t\}$) parameters which over-parameterize the rendered 2D view, e.g. the effect of scaling $\mathbf{v}$ itself can be offset by inversely scaling $\mathbf{c}_s$. This often hinders the alignment of $\mathbf{v}$ with the ground-truth 2D mask and thereby rendering the network inappropriately update the mesh parameters. Therefore, we let the shape loss L_{Sh} take effect only when the articulation loss L_{Art} becomes sufficiently small: λ is set per data instance based on the L_{Art}-value:

$$\lambda = \begin{cases} 1 & \text{if } \left\| \left[[f^{DHPE}(\mathbf{x}_i)]_{\mathbf{j}} \right]_{2D} - [\mathbf{j}_i]_{2D} \right\|_2^2 < \tau \\ 0 & \text{otherwise,} \end{cases} \tag{4.11}$$

where $[\mathbf{j}]_{2D}$ projects 3D joint coordinates $\mathbf{j}$ to 2D view based on $\mathbf{c}$ (Eq. 4.4). The threshold τ is empirically set to 15 pixels. Initially, with zero λ, L_{Art} dominates in L, which helps globally align the estimated meshes with 2D ground-truth evidence. As the training progresses, the role of L_{Sh} becomes more important ($\lambda = 1$) contributing to recovering the detailed shapes.

As the generation of the 2D skeleton $\mathbf{j}_{2D}$ from an estimated heat-map $f^{J2D}(\mathbf{x})$ is not differentiable (see Eq. 4.3), our 2D pose estimation network f^{J2D} cannot be trained based on L. Thus we train it in parallel using L_{J2D} (Eq. 4.3). For both cases, we use the standard Adam optimizer with the learning rate γ set at 10^{-3}.

4.2.5 Testing refinement

To facilitate the training of the HME, we constructed an auxiliary DHPE that decomposes into three component functions: f^{Proj}, f^{E2D}, and f^{E3D} (see Eq. 4.1). An important benefit of this step-wise estimation approach is that it enables us to check and improve once predicted output mesh by comparing it with the intermediate results: For testing, the underlying mesh $\mathbf{v}$ of a given test image $\mathbf{x}$ can be first estimated by applying $f^{HME} = f^{E3D} \circ f^{E2D} : X_C \rightarrow Y_P$.

Algorithm 6: Training process

Input:
–Training data $D = \{(\mathbf{x}_i, (\mathbf{j}_i, \mathbf{m}_i))\}_{i=1}^{l}$:
$\mathbf{x}$: RGB image;
$(\mathbf{j}, \mathbf{m})$: ground-truth
3D skeleton and 2D segmentation mask;
–Projection operator $f^{Proj} = (f^{Reg}, f^{Ren})$;
–MANO model: PCA shape basis;
mean pose vector;
–Hyper-parameters: number T of epochs
size N' of mini-batch;

Output: (Weights of)
–3D mesh estimator f^{E3D};
–2D evidence estimator $f^{E2D} = (F, f^{J2D})$;

Initialization:
–Randomize (parameters) of f^{E3D};
–Pre-train F based on [64];
–Pre-train f^{J2D} based on [256];

for $t = 1, \ldots, T$ **do**
  for $n = 1, \ldots, N/N'$ **do**
    For each data point $\mathbf{x}$ in the mini-batch D_n,
    evaluate (feed-forward) f^{DHPE} on $\mathbf{x}$:
    Generate mesh parameter $\mathbf{h}'$, 3D skeleton $\mathbf{j}'$ and 2D segmentation mask $\mathbf{m}'$;
    Generate 2D evidences $(F(\mathbf{x}), \mathbf{j}'_{2D}(t))$, mesh parameter $\mathbf{h}'$, 3D skeleton $\mathbf{j}'$ and 2D segmentation mask $\mathbf{m}'$;
    if $t > 20$ **then**
      Augment D with new synthetic data instances generated from $\mathbf{h}'$ (Eq. 4.4), by changing its shape $\mathbf{s}'$ and viewpoint $\mathbf{q}'$;
    end
    Calculate gradient ∇L with respect to (the weights of) f^{E3D} (Eq. 4.7) on D_n, and update f^{E3D};
    Calculate gradients ∇L_{Feat} (Eq. 4.2) and ∇L with respect to F on D_n, and update F;
    Calculate gradient ∇L_{J2D} (Eq. 4.3) with respect to ∇f^{J2D} on D_n, and update f^{J2D};
  end
end

If the resulting prediction $\mathbf{v}'$ (equivalently, $\mathbf{h}'$) is accurate, it must be in accordance with the intermediate results $F(\mathbf{x})$ and $\mathbf{j}'_{J2D}$ generated from $\mathbf{x}$. Checking and further enforcing this consistency can be facilitated by noting that our loss function L and its components are

Algorithm 7: Testing process

Input: Test image $\mathbf{x}$;
Output:
–3D mesh $\mathbf{v}'$;
–2D segmentation mask $\mathbf{m}'$;
–3D skeleton $\mathbf{j}'$;
Feed-forward f^{DHPE} on $\mathbf{x}$: Generate 2D evidence $\mathbf{j}'_{2D}(t)$, and 3D mesh $\mathbf{v}'$ and it's parameter $\mathbf{h}'$;
for $t = 1, \ldots, 50$ **do**
  Update $\mathbf{h}'(t)$ using Eq. 4.12.
end
Generate $\mathbf{v}'$ from $\mathbf{h}'(t)$ and $\mathbf{m}'$;
Generate $\mathbf{j}'$ from $\mathbf{v}'$;

differentiable with respect to the mesh parameter $\mathbf{h}$. By reinterpreting L as a smooth function of $\mathbf{h}$ given fixed f^{Proj}, f^{E2D}, and f^{E3D}, we can refine the initial prediction $\mathbf{h}'(0)$ by enforcing such consistency:

$$\mathbf{h}'(t+1) = \mathbf{h}'(t) - \gamma \cdot \nabla_{\mathbf{h}} \Big(\left\| \left[[f^{DHPE}(\mathbf{x})]_{\mathcal{J}} \right]_{XY} - \mathbf{j}'_{J2D} \right\|_2^2 + \lambda \left\| F(\mathbf{x}) - F(f^{Ren}(\mathbf{v}') \odot \mathbf{x}) \right\|_2^2 + L_{Lap} \Big), \tag{4.12}$$

where $[\mathbf{j}]_{XY}$ extracts the $x, y-$coordinate values of skeleton joints from $\mathbf{j}$. Note that in the first gradient term, we use the 2D skeleton $\mathbf{j'}_{2D}$ since the ground-truth 3D skeleton $\widehat{\mathbf{j}}$ is not available at testing. This step benefits from explicitly building the 2D pose refiner f^{Ref} that improves the once estimated 2D joint $\mathbf{j'}_{2D}$ during iterative mesh estimation process (Eq. 4.6). Also, for the second, shape loss term of the gradient, since the ground-truth segmentation mask $\mathbf{m}$ is not available, we use the segmentation mask rendered via f^{Ren} based on the estimated mesh $\mathbf{v}'$ enforcing self-consistency. The number of iterations in Eq. 4.12 is fixed at 50. Our testing refinement step takes 250ms (5ms per iteration $\times$ 50 iterations) in addition to the initial regression step which takes 100ms. Figure 4.4(e) shows the performance variation with varying number of iterations.

4.2.6 Self-supervised data augmentation

An important advantage of incorporating a generative mesh model (i.e. MANO) into the training process of the HME (via $\mathbf{h}$; see Eq. 4.4) is that it can synthesize new data as guided by (and subsequently, guiding) HME. The MANO model provides explicit control over the shape of synthesized 3D hand mesh.[1] Combining this with a camera model, we can generate pairs of 3D meshes, and the corresponding rendered 2D masks and RGB images.

To render RGBs, we adopt the neural texture renderer f^{TRen} proposed by Kato et al. [79]: During training, once a seed mesh $\mathbf{v}^S$ is predicted, the corresponding camera and shape parameters $\mathbf{h}^S$ are changed to generate new meshes $\{\mathbf{v}_j^N\}$: The shape parameter $\mathbf{s}$ is sampled uniformly from the interval covering three times the standard deviation per dimension. For camera perspectives, the rotation matrix along each of $x, y, z-$ axes are sampled uniformly on $[0, 2\pi]$. Once the foreground hand region is rendered via f^{TRen}, it is placed on random backgrounds obtained from the NYU depth database [175].

For each new mesh $\mathbf{v}^N$, we generate a triplet $\left(f^{TRen}(\mathbf{v}^N), f^{Ren}(\mathbf{v}^N), f^{Reg}(\mathbf{v}^N)\right)$ constituting a new training instance for the DHPE. We empirically observed that when the training of the DHPE reaches 20 epochs, it tends to generate seed meshes $\{\mathbf{v}_i^S\}$ which faithfully represent realistic hand shapes (even though they might not accurately match the corresponding input images $\{\mathbf{x}_i\}$). Therefore, we initiate the augmentation process after the first 20 training epochs. For each mini-batch, three new data instances are generated per seed prediction $\mathbf{v}^S$ gradually enlarging the entire training set (see supplemental for examples). A similar self-supervised data augmentation approach has been adopted for facial shape estimation [85].

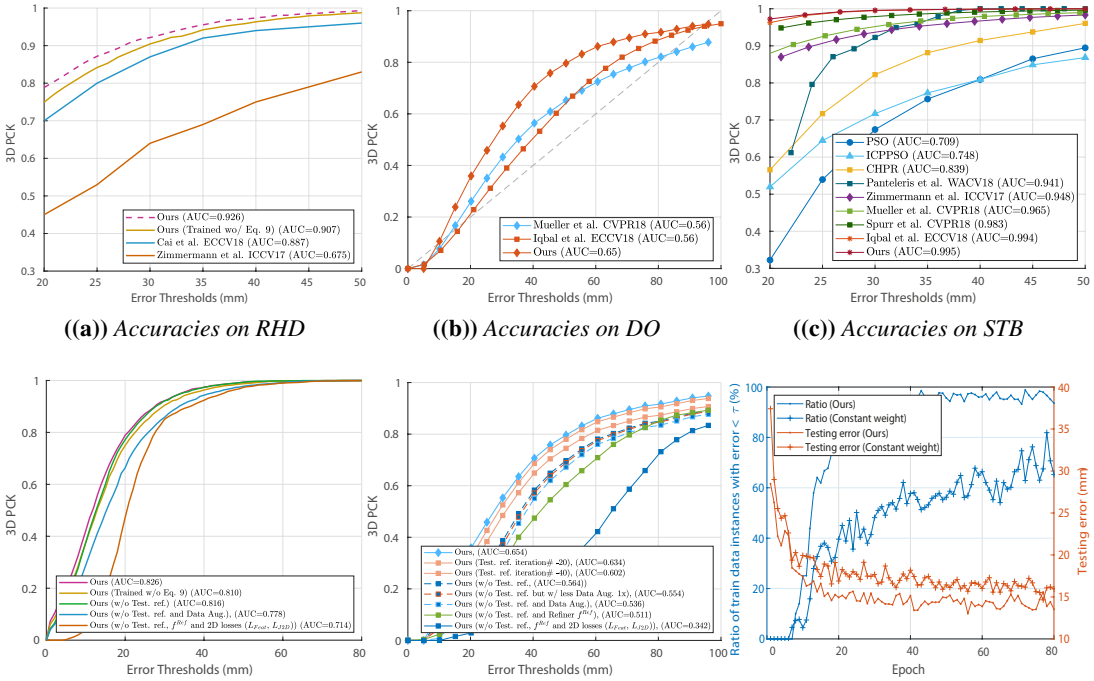
[1] While it is also possible to generate new poses, we do not explore this possibility since we observed that it often leads to implausible hand poses.

4.3 Experiments

Experimental settings. We evaluate the performance of our algorithm on 3 hand pose estimation datasets. Stereo hand pose dataset (STB) provides frames of 12 stereo video sequences each recording a single person performing various gestures [248]. It contains total 36,000 frames. Among them, 30,000 frames sampled from 10 videos constitute a training set while the remaining 6,000 frames (from 2 videos) are used for testing. The rendered hand pose dataset (RHD) contains 43,986 synthetically generated images showing 20 different characters performing 39 actions where 41,258 images are provided for training while the remaining 2,728 frames are reserved for testing [256]. Both datasets are recorded under varying backgrounds and lighting conditions and they are provided with the ground-truth 2D and 3D skeleton positions of 21 keypoints (1 for palm and 4 for each finger), on which the accuracy is measured. The Dexter+Object dataset (DO) contains 3,145 video frames sampled from 6 video sequences recording a single person interacting with an object (see Fig. 4.6(DO)) [185]. This dataset provides ground-truth 3D skeleton positions for the 5 finger-tips of the left hand: The overall accuracy is measured in these finger-tip locations. Following the experimental settings in [18, 70, 256] we train our system on 71,258 frames combining the original training sets of STB and RHD. For testing, the remaining frames in STB and RHD, respectively and the entire DO is used. The overall hand pose estimation accuracy is measured in the area under the curve (AUC) and the percentage of correct keypoints (PCK) with varying thresholds for each [18, 70, 256].

For comparison, we adopt seven hand pose estimation algorithms including five CNN-based algorithms ([18, 256] for RHD, [70, 117] for DO, and [117, 183, 256] for STB) and two 3D model fitting-based algorithms [81, 133].

Many existing CNN-based algorithms guide the learning process via building intermediate 2D evidence: Zimmermann and Brox's algorithm [256] first estimates 2D skeletons from the input RGB images and thereafter, maps estimated 2D skeletons to 3D skeletons. Spurr et al.'s algorithm [183] builds a latent space shared by RGB images and 2D/3D skeletons. Cai et al.'s algorithm [18] trains an RGB-to-depth synthesizer that generates intermediate depth-map



((a)) *Accuracies on RHD* **((b))** *Accuracies on DO* **((c))** *Accuracies on STB*

((d)) *Ablation study results RHD* **((e))** *Ablation study results DO* **((f))** *Progression of the DHPE accuracies*

Figure 4.4: *Performances of different algorithms on three benchmark datasets: (a-c) accuracies on RHD, DO, and STB, respectively; (d-e) evaluation of our algorithm design choices on RHD and DO, respectively; (f) progressions of the testing errors (orange curves) and the ratio of training data instances with small joint estimation errors ($< \tau$ in Eq. 4.11; blue curves) with λ fixed at* 0.01 *(curves with dot markers) and with λ scheduled based on Eq. 4.11 (curves with cross markers).*

estimates to guide skeleton estimation. Similarly, Iqbal et al.'s algorithm [70] reconstructs depth maps from input RGB images. Unlike these algorithms, our algorithm builds full *3D meshes* to guide the skeleton estimation process.

Mueller et al. [117] approached the challenge of building realistic training data pairs of 3D skeletons and RGB images by first generating synthetic RGB images from skeletons and then transferring these images into realistic ones via a GAN transfer network. On the other hand, our algorithm focuses on dense hand pose estimation and thus it generates pairs of RGB images and the corresponding 3D meshes.

Results. Figure 4.4 summarizes the results. Our algorithm improved Zimmermann and Brox's algorithm [256] with a large margin (Fig. 4.4(a)).

Table 4.2: *Performances of different hand segmentation algorithms on RHD (higher is better).*

Method	IOU score	Precision	Recall	F1-score
Ours	65.13%	82.82%	75.31%	78.88
[256]	35.40%	36.52%	92.06%	52.29
[84]	52.68%	71.65%	66.55%	69.00

Also, a significant accuracy gain (≈4mm on average) from Cai et al.'s approach [18] was obtained, demonstrating the effectiveness of our 3D mesh-guided supervision in comparison to 2.5D depth map-guided supervision. Figure 4.4(a) also demonstrates that jointly estimating the shape and pose (using Eqs. 4.9 and 4.10) leads to higher pose estimation accuracy than estimating only the pose. Figure 4.4(b) (DO) shows that our algorithm clearly outperforms [70, 117]: It should be noted that the comparison with [117] is not fair since their networks were trained on a database tailored for object interaction scenarios as in DO. Figure 4.4(c) (STB) further demonstrates that in comparison to the state-of-the-art CNN-based algorithms [117, 183, 256] as well as 3D model fitting approaches (PSO and [133]), our algorithm achieves significant performance improvements. The superior performance of our algorithm over [117] shows the effectiveness of our data generation approach. The performance of our algorithm is on par with [70] on this dataset but ours outperforms [70] on DO.

Ablation study. Figure 4.4(d-e) shows the result of varying design choices in our algorithm on DO and RHD: The testing refinement step (Sec. 4.2.5) had a significant impact on the final performance: The results obtained without this step ('w/o Test. ref.') is considerably worse on DO; On RHD, results of 'w/o Test. ref.' are similar. Our data augmentation strategy (Sec. 4.2.6) further significantly improved the accuracy over 'w/o Data Aug.' cases. Figure 4.4(d-e) further show that explicitly providing supervision to our 2D evidence estimator via 2D losses (L_{J2D} and L_{Feat}; Eqs. 4.2-4.3) constantly improve the overall accuracy over 'w/o 2D loss' cases which correspond to Kanazawa et al.'s iterative estimation framework [77]. Figure 4.4(f) visualizes the effectiveness of our hierarchical shape and articulation optimization strategy (Eq. 4.11): In comparison to the case of constant λ value ('Constant weight'), automatically scheduling λ based on Eq. 4.11 led to larger fraction of training instances that have small articulation errors ($< \tau$; Eq. 4.11) which then led to faster decrease of test error.

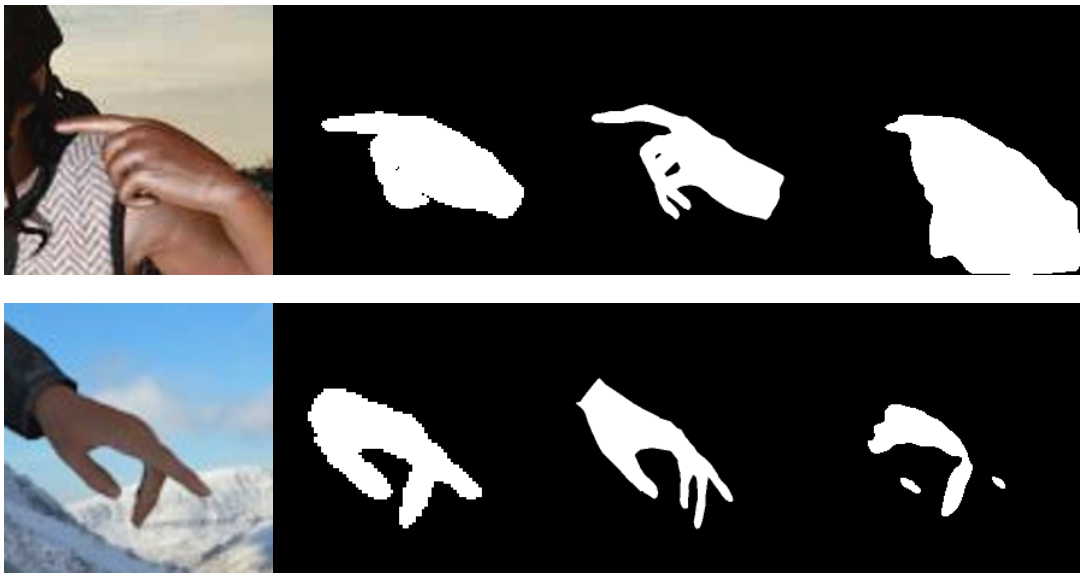


Figure 4.5: *Hand segmentation examples. Left to right: input images, ground-truth masks, our results, the results of the state-of-the-art hand segmentation algorithm [84].*

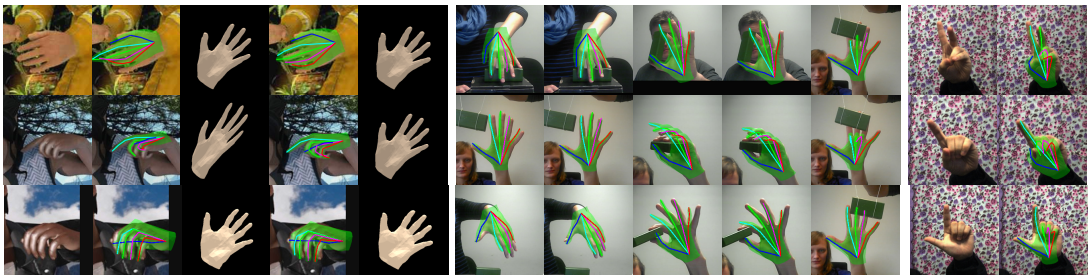


Figure 4.6: *Example dense hand pose estimation results. (RHD): (a) input images; (b-c) and (d-e) our results obtained without and with the shape loss* L_{Sh} *(Eq. 4.10), respectively; (b,d) dense hand pose estimation results, and (c,e) estimated shapes in canonical hand pose. (DO): (a,c) and (b,d) our results obtained without testing refinement and after applying* 20 *iterations of testing refinement, respectively; (e) failure and success cases under occlusion. (STB): (a-b) input images and our results.*

Qualitative evaluation. For qualitative evaluation, we show example dense hand pose estimation results in Fig. 4.6. Figure 4.6(RHD) demonstrates the importance of the shape loss L_{Sh} (Eq. 4.10) conforming the quantitative results of Fig. 4.4(a,d).

Figure 4.7 shows additional dense hand pose estimation results extending Fig. 7 of the main paper. When compared with the results obtained by a variation of our algorithm that does not

use the shape loss (b–d: L_{Sh}; Eq. 4.10), our final algorithm (e–g) achieved much higher shape estimation accuracy (c and f, especially in 1st, 2nd, 5th, and 7th examples), which led to better alignment of hand contours (c and f) and eventually to significantly lower pose estimation error (b and e). These examples confirm the quantitative results shown in Fig. 4.8 and demonstrate the benefits of shape estimation even when the final goal is to estimate skeletal poses.

Hand segmentation performance. Systematically evaluating the performance of dense hand pose estimation (including shape) is challenging due to the lack of ground-truth labels. Therefore, we assess the performance indirectly based on the 2D hand segmentation accuracy. Table 4.2 shows the results measured in 4 different object segmentation performance criteria. In comparison to the state-of-the-art hand segmentation networks [84, 256] (note [84] is fine-tuned for RHD), our algorithm led to significantly higher accuracy. Also, Fig. 4.5 shows that the state-of-the-art segmentation net [84] is distracted by other skin colored objects than hands (e.g. arms) while our algorithm, by explicitly estimating 3D meshes, can successfully disregarded these distracting backgrounds.

4.4 Conclusion

We have presented DHPE network, a CNN-based framework that reconstructs 3D hand shapes and poses from single RGB images. DHPE decomposes into the 2D evidence estimator, 3D mesh estimator and projector. The projector, via the neural renderer, replaces insufficient full 3D supervision with indirect supervision by 2D segmentation masks/3D joints, and enables generating new data. In the experiments, we have demonstrated that stratifying 2D/3D estimators improves accuracy, updating 2D skeletons helps self-supervision in the iterative testing refinement and improves 3D skeleton estimation, and jointly estimating 3D hand shapes and poses offers the state-of-the-art accuracy in both 3D hand skeleton estimation and 2D hand segmentation tasks.

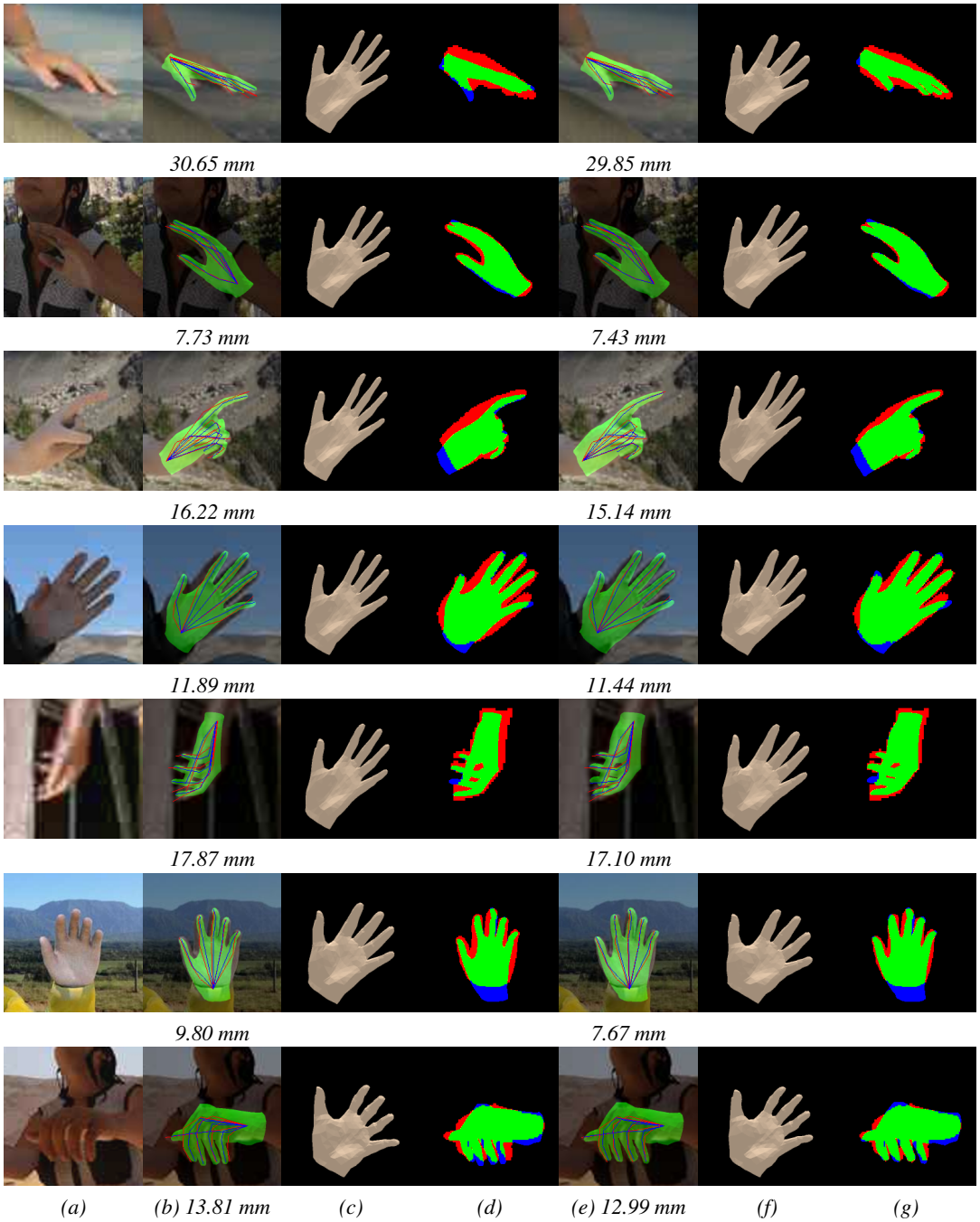


Figure 4.7: *DHPE examples. (a) input images, (b-d) and (e-g) results obtained without and with shape loss, respectively. (b,e) estimated hand meshes overlaid on the input image and the corresponding estimated skeletons (Blue) overlaid with their ground-truths (Red), (c,f) estimated shapes rendered in canonical articulation and viewpoints, and (d,g) Color-coded 2D segmentation masks: (Green and Blue: estimated masks; Green and Red: ground-truth masks; Red and Blue highlight errors). Our visualization method in (d) and (g) is inspired by [193].*

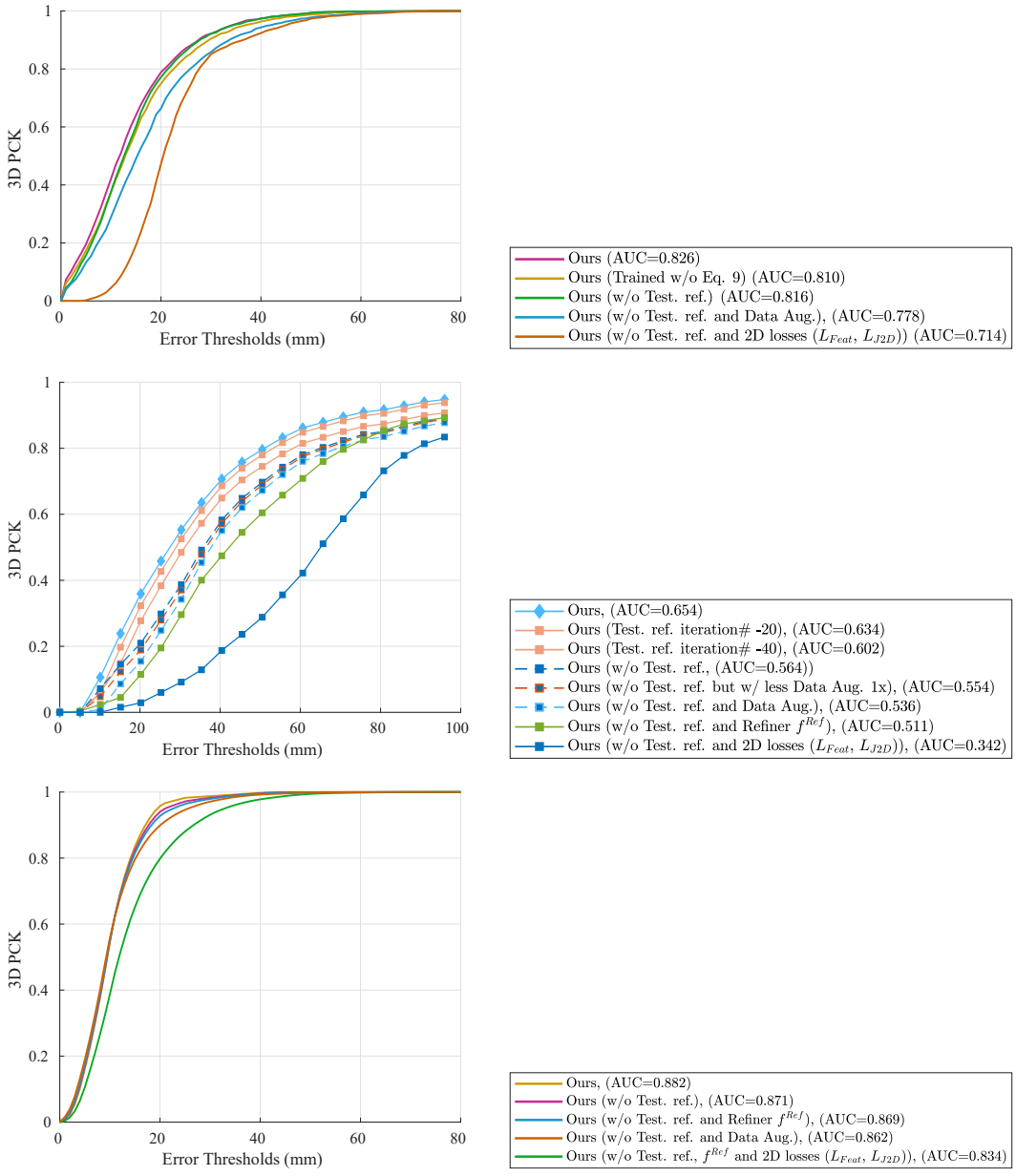


Figure 4.8: *Performance of our algorithm with different design choices. Top to bottom: results on RHD, DO, and STB, respectively.*

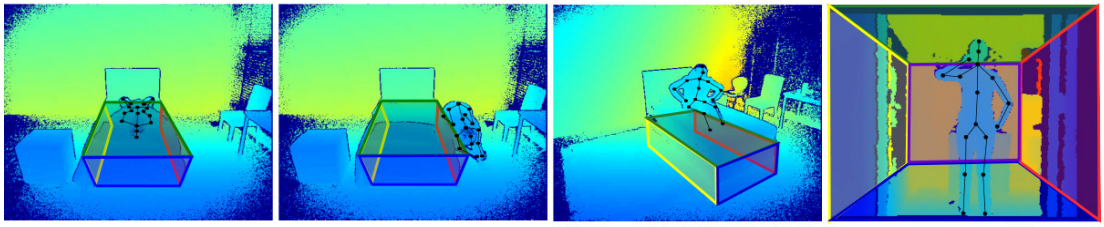
CHAPTER 5

ACTION RECOGNITION USING THE 3D GEOMETRY (BODY SKELETON, SCENE LAYOUTS)

Contents

This chapter deals with the action recognition problem. In the action recognition, multiple cues (e.g. RGB, depth, 3D skeleton and scene layout) are available as discussed in chapter 2. In the conventional action recognition scenario that deals with the daily or gaming actions, humans are mostly in the upright position and frontally looking at the camera viewpoint. In such a scenario, human body keypoints (i.e. skeletons) are well recognized by the commercial

human body skeleton estimator [172] and thus, subsequent action recognition can exploit them. However, the body pose estimator does not work well if the frame contains unique human postures and viewpoints. In this case, the subsequent action recognition also suffers. In this chapter, we aim to deal with such a case where the human body pose estimation is challenging: 24/7 monitoring patient's actions in the hospital scenario. In the problem of "24 hours monitoring patient actions" in a hospital scenario, we tackle actions such as "lying on the bed", "stretching an arm out of the bed" and "falling out of the bed". Due to the privacy issue and the 24/7 monitoring requirements, we are constrained to use the depth sequences without color information; however recognizing actions solely by depth information is challenging. Especially in the concerned scenario, 3D geometric information, *i.e.* relations between scene layouts and body kinematics is important to reveal the actions. We try to additionally incorporate such 3D geometric information in our action recognizer. However as mentioned, due to unique and diverse human postures in our data, securing the 3D geometric information at testing itself is a challenging problem. To address the problem, we propose kinematic-layout-aware random forest (KLRF) considering the geometry between scene layouts and skeletons (*i.e. kinematic-layout*), secured in the offline manner, in the training of forests to maximize the discriminant power of depth appearance. The proposed method uses the additional information only at training to obtain the robust classification model, while at testing the additional cue is not used. We integrate the kinematic-layout in the split criteria of random forests to guide the learning process by 1) measuring the usefulness of kinematic-layout information and switching the use of kinematic-layout, and 2) implicitly closing the gap between two distributions obtained by the kinematic-layout and the appearance, if the kinematic-layout appears useful. Experimental evaluations on our new dataset (PATIENT) demonstrate that our method outperforms various state-of-the-arts for this problem. We have also demonstrated accuracy improvements by applying our method to conventional single-view and cross-view action recognition datasets (*e.g.* CAD60, UWA3D).



(a) "lying on the bed" (b) "falling out of the bed" (c) a different viewpoint (d) CAD dataset

Figure 5.1: *Depth maps visualized with kinematic-layout. Note that kinematic-layout has a potential to improve the ambiguity of depth appearance. (a)-(c) are depth maps from PATIENT dataset while (d) is the depth map from CAD60 dataset.*

5.1 Motivation

The recent emergence of cost-effective and easy-operation depth sensors have opened the door to a new family of methods [68, 89] for action recognition from depth sequences. Compared to conventional color images, depth maps offer several advantages: 1) Depth maps encode rich 3D structural information, including informative shape, boundary, geometric cues of a human body and an entire scene. 2) Depth maps are insensitive to changes in lighting and illumination conditions that make it possible to monitor patient/animal 24/7. 3) It is invariant to texture and color variations, which benefits various recognition tasks.

These advantages have promoted the fast pace development of depth-based techniques for action recognition. A number of spatio-temporal representations [151, 152, 223] have been proposed to well represent the depth appearance, which is different from color maps. Recent approaches resorted to selecting the informative points around skeleton joints and modelling their temporal dynamics [37, 213, 222, 250], when human skeleton can be estimated from depth sequences. However, it is important to note that human pose estimation is known to be not always reliable and can fail when the human is not in an upright and frontal view position (e.g. lying) [151] or observed from unseen camera viewpoints [60]. Our scenario lies in these cases as in Fig 5.1 (a)-(c) and 5.2. To utilize the information which is not reliably obtainable at testing, we seek to formulate human poses and their 3D relations to layouts only during training by using their offline-secured ground-truths. Our aim is therefore to learn more robust classification models with more information at training and to obtain improved testing accuracy without explicit use

of them at testing.

In order to investigate these issues, in this chaper we make following contributions:

New action recognition dataset (PATIENT) has been collected containing patient behaviors (15 actions) in a ward by a depth camera. Actions in our dataset have close ties with scene layouts (*e.g.* *bed*, *floor*) and human body joints as in Fig. 5.1, 5.2; thus, utilizing kinematic-layout (*i.e.* 3D geometric relations between layouts and human body joints) is important to discriminate targeted actions. On the contrary, due to unique viewpoints and human poses in our dataset, skeleton information cannot be reliably tracked [60, 151] in a real-time manner, using a conventional depth sensor (*e.g.* kinect).

Kinematic-layout-aware random forests (KLRF) is introduced to improve the discriminant power of depth appearance by encoding the kinematic-layout. Considering that obtaining kinematic-layouts at testing itself is a challenging problem, we formulate KLRFs to use their offline-secured ground-truth implicitly at training and do not use them at testing (see Fig. 5.4 (a), (b)). Similarly to random forests (RFs), KLRFs learn split parameters by maximizing split objectives. However, in KLRFs, while split parameters are conditioned only on the appearance feature, split objectives are defined based on both appearance feature and kinematic-layout information. This is different from the case of RFs whose split parameters and split objectives are both conditioned only on the appearance feature. As a result, split parameters of KLRFs are determined satisfying more strict split objectives composed of multimodal information, compared to those of RFs. Furthermore, we constitute the KLRFs encode the kinematic-layout adaptively: first cluster data samples into two groups where a group whose kinematic-layout is useful and a group whose kinematic-layout is less useful, then adaptively use it depending on the usefulness.

Both cross and single-view settings are tested on our own scenario (*i.e.* PATIENT dataset) and conventional (*i.e.* CAD60, UWA3D) action recognition. Cross-view experiment is demonstrated to show the generalization ability of our method.

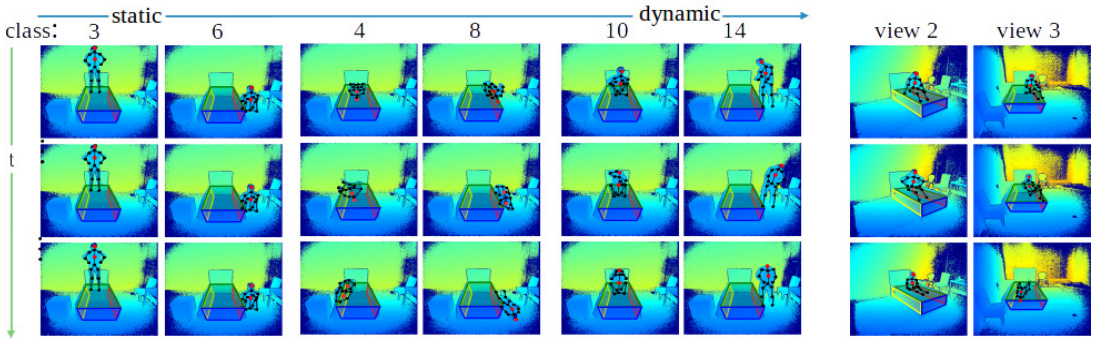


Figure 5.2: *Examples of our PATIENT dataset. Our dataset contains both static (left side) and dynamic actions (right side). Action labels are given in Sec. 5.2. Examples for different views are also shown in last two columns.*

5.2 PATIENT dataset

We collect our own dataset (*i.e.* PATIENT) in a hospital scenario which contains 15 actions, performed by 10 subjects in 3 different viewpoints having close ties with *bed* and *floor* layouts. The dataset contains both static and dynamic actions and all 15 actions are: (1) lying, (2) sitting and (3) standing on the bed; (4-5) stretching body parts out of the bed when the patient is lying and sitting; (6-7) sitting and standing on the floor; (8) falling out of the bed; (9-15) suffering status of actions (1-8) except (3). In Table 5.1, we compare the PATIENT dataset with recently proposed action recognition datasets.

In most action DBs in Table 5.1, human joints are well captured by kinect sensors at testing, since humans are in upright positions (*e.g.* standing, sitting) and the camera is located in front of humans. In our scenario, humans' depth appearance is ambiguous due to their unique poses (*e.g.* lying, sitting back) and camera views (*i.e.* not human's frontal). Thus, capturing human joints is not also easy [60, 151]. Another characteristic of our dataset is that actions that we aim to recognize are closely related to 3D geometric relations between layouts (*i.e. bed*, *floor*) and human joints. Thus, we provide manually labeled human body joints and layout planes (*i.e. bed*, *floor*) as the ground-truths to help reveal the actions. 3D scene layouts are fixed throughout the videos, thus, first frame's scene layouts are used for all frames. Three different 3D points are manually annotated for each layout to uniquely decide the plane. Also, human body skeletons

Dataset	Geometric info.	Samples	Classes	Subjects	Views	Human poses
CAD60 [192]	3D joints	60	12	4	1	Frontal/Upright
3D Action Pairs [132]	3D joints	360	12	10	1	Frontal/Upright
UTD-MHAD [23]	3D joints	861	27	8	1	Frontal/Upright
UWA3D [151]	3D joints	1075	30	10	5	Frontal/Upright
NTU [166]	3D joints	56880	60	40	80	Frontal/Upright
Ours	3D joints+**Layout**	450	15	10	3	**Various**

Table 5.1: *Dataset comparison to recent benchmarks.*

are labeled by manual efforts: reading depth values and doing approximation for occluded joints with neighboring pixels. The initial annotation is done for every 10 frames and temporally interpolated. Then, manual inspection was performed. We also generate 5 layout planes (*i.e.* *floor*, *left wall*, *mid wall*, *right wall*, *ceiling*) for conventional (CAD60, UWA3D) datasets, as in Fig. 5.1 (d). Fig. 5.2 shows example frames of our dataset spanning static and dynamic actions.

5.3 Kinematic-Layout-aware Random Forests

In this section, we first introduce our appearance A and kinematic-layout K (Sec. 5.3.1) and then present how our approach exploits both information at training (Sec. 5.3.2). Testing stage of KLRFs are explained in Sec. 5.3.3.

5.3.1 Appearance and kinematic-layout information

We construct the appearance A using the depth sequence V and the kinematic-layout K using layouts L and skeleton joints P for V, respectively. 1) We first extract depth cue $\mathbf{C}_t^D$, layout cue $\mathbf{C}_t^L$ and skeleton cue $\mathbf{C}_t^J$ for each frame t. 2) Then, we generate the spatio-temporal representation, A and K for a depth sequence V, by applying the Fourier transform on per-frame cues as in [152, 222]. The per-frame cues are defined as follows:

Depth cue $\mathbf{C}_t^D$: For each frame t, we extract the $4,096$ dimensional feature $\mathbf{C}_t^D$ from the $fc7$ layer of the CNN architecture proposed in [152]. This architecture is pre-trained on synthetic

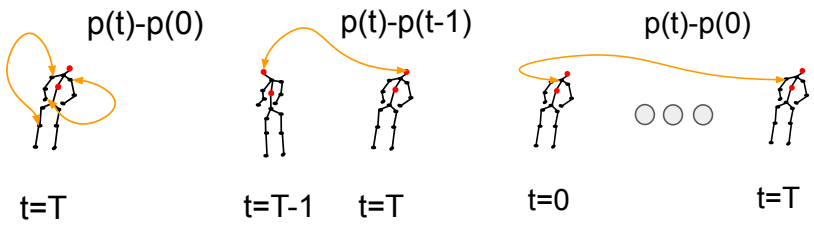


Figure 5.3: *Visualization of the skeleton cue* $\mathbf{C}_t^J$ *at* $t = T$*: (a) Skeleton pairwise distance vector* $\mathbf{d}_t^P$*; (b) Skeleton motion vector* $\mathbf{d}_t^M$*; (c) Skeleton offset vector* $\mathbf{d}_t^O$*. Orange arrows denote example paired joints used for calculating distances.*

multi-view depth maps and shown to produce the state-of-the-art accuracy on both single and multi-viewed 3D action recognition benchmarks [152].

Skeleton cue $\mathbf{C}_t^J$: Skeleton cue $\mathbf{C}_t^J$ is encoded similar to [213, 247, 255] that proposed a method to calculate feature vector from skeletal coordinate values. For example, they encode *per-frame pose* by calculating the pairwise distance between all skeleton pairs, the *motions* by calculating the same skeleton joint's displacement between two consecutive frames and *offsets* by calculating the offset to the initial skeleton locations. We constitute the skeleton cue $\mathbf{C}_t^J$ by concatenating three types (see Figure 5.3 visualizes the three types of skeleton cues) of cues as $\mathbf{C}_t^J = [\mathbf{d}_t^P; \mathbf{d}_t^M; \mathbf{d}_t^O]$ where $\mathbf{d}_t^P$, $\mathbf{d}_t^M$, $\mathbf{d}_t^O$ are defined as follows:

(1) Skeleton *Pairwise* distance vector, $\mathbf{d}_t^P = [\mathbf{p}_1(t) - \mathbf{p}_2(t), ..., \mathbf{p}_p(t) - \mathbf{p}_q(t), ..., \mathbf{p}_{P-1}(t) - \mathbf{p}_P(t)]$ is defined for $\forall p, \forall q, p \neq q \in [1, P]$ to encode current frame's human poses.

(2) Skeleton *Motion* vector, $\mathbf{d}_t^M = [\mathbf{p}_1(t) - \mathbf{p}_1(t-1), ..., \mathbf{p}_p(t) - \mathbf{p}_p(t-1), ..., \mathbf{p}_P(t) - \mathbf{p}_P(t-1)]$ is defined for $\forall p \in [1, P]$ to encode its temporal motion information.

(3) Skeleton *Offset* vector, $\mathbf{d}_t^O = [\mathbf{p}_1(t) - \mathbf{p}_1(1), ..., \mathbf{p}_p(t) - \mathbf{p}_p(1), ..., \mathbf{p}_P(t) - \mathbf{p}_P(1)]$ is defined for $\forall p \in [1, P]$ to encode human offset information to their initial values *i.e.* $t = 1$. Skeleton cue can consider the spatial location of human body parts.

Layout cue $\mathbf{C}_t^L$: For each frame t, we propose to extract $\mathbf{C}_t^L$ by 3D displacements between

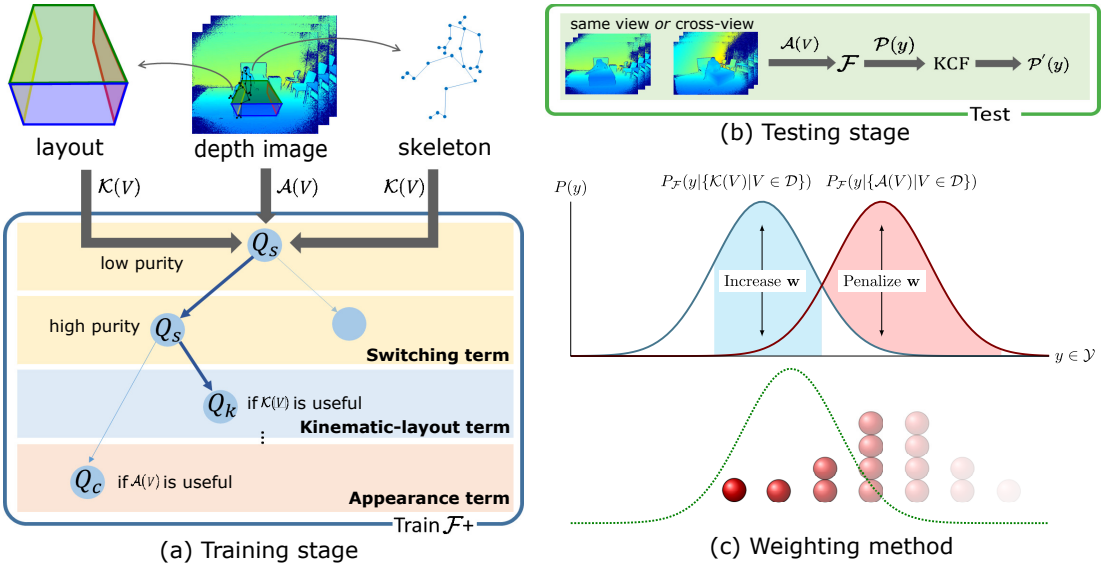


Figure 5.4: *Flowchart of our method. (a) Training stage of KLRFs, (b) Testing stage of KLRFs, (c) Weighting method to reduce the gap between $P_{\mathcal{F}}(y|\{A(V)|V \in D\})$ and $P_{\mathcal{F}}(y|\{K(V)|V \in D\})$. Red balls denote samples constituting the appearance-based distribution $P_{\mathcal{F}}(y|\{A(V)|V \in D\})$ with their weights in fade-out. Green line denotes the gap-reduced class distribution.*

layout planes $\mathbb{L} = \{\mathbf{L}_1, ...\mathbf{L}_l, ..., \mathbf{L}_L\}$ and skeleton joints $\mathbb{P}(t) = \{\mathbf{p}_1(t), ...\mathbf{p}_p(t), ..., \mathbf{p}_P(t)\}$ as:

$$\mathbf{C}_t^L = [\mathbf{d}_{t11}; ...; \mathbf{d}_{t1L}; \mathbf{d}_{t21}; ...; \mathbf{d}_{t2L}; ...; \mathbf{d}_{tP1}; ...; \mathbf{d}_{tPL}] \tag{5.1}$$

where $\mathbf{d}_{tpl} = \mathbf{p}_p(t) - \tilde{\mathbf{p}}_{\mathbf{L}_l}$, $\mathbf{p}_p(t)$ is a 3-dimensional vector whose entry corresponds to its x, y and depth value and $\tilde{\mathbf{p}}_{\mathbf{L}_l}$ is a projection of $\mathbf{p}_p(t)$ to the plane $\mathbf{L}_l$, respectively.

This layout cue provides information on how humans interact with their environments. There exists a strong physical and functional coupling between human actions/poses and the 3D geometry of a scene [32, 44, 58]. We try to consider such physical constraints to support actions such as "sitting" and "lying" by layout planes.

5.3.2 Learning KLRFs

In the action recognition problem, as mentioned in chapter. 1, we construct the action recognizer $f^A : V \to Y_A$ that maps depth video space V to the action label space Y_A (we will call this as Y for simplicity). We first extract appearance A and kinematic-layouts K from each frame

of the video $v \in V$ and maps the appearance A to the action label space Y using the random forests (RFs), $f^A := \mathcal{F}$. RFs $\mathcal{F}$ are constructed as the baseline which learn a mapping from the appearance A to the label set Y:

$$\mathcal{F} : A \mapsto Y. \tag{5.2}$$

To improve the action recognition accuracy using the 3D geometry information, i.e. kinematic-layouts K, we propose KLRFs $\mathcal{F}^+$ to optimize the mapping in Eq. 5.2 with the help of kinematic-layouts K at training. Considering the fact that there are actions which are well recognized by the appearance A rather than the kinematic-layouts K, we also propose to encode kinematic-layouts K when it appears to be useful, as follows:

$$\mathcal{F}^+ : \begin{cases} A \overset{K}{\mapsto} Y, & \text{, if } K \text{ is useful} \\ A \mapsto Y & \text{, otherwise} \end{cases} \tag{5.3}$$

Standard RF training. RFs are ensembles of binary trees, containing two types of nodes: *split* and *leaf*. At training, trees are grown by deciding the split function $\Psi(A(\cdot)^\gamma, \tau)$ recursively from the root node, where $A(\cdot)^\gamma$ denotes the γ-th value in the appearance feature and τ is a threshold. At each *split* node, arrived samples $V \in D$ are divided into two subsets D_l and D_r ($D_l \cap D_r = \emptyset$) by a set of split function candidates $\{\Psi^c\}$ that is generated randomly. Samples whose $A(V)^\gamma$ are less than τ go to the left child node (D_l) while others go to the right child node (D_r). Among candidates, the one that maximizes the split objective $\mathcal{Q}$ is selected as a split function Ψ^*:

$$\Psi^* = \arg \max_{\Psi \in \{\Psi^c\}} \mathcal{Q}(\Psi). \tag{5.4}$$

Trees are grown while sample number is above the minimum threshold (*i.e.* 5 in our experiments) or information gain is positive, where the information gain is defined as $\mathcal{Q}(\Psi^*) - \mathcal{Q}(\Psi^0)$ and Ψ^0 is the reference split that have all samples in D_l and no samples in D_r. The terminating node becomes a *leaf* node and saves class distribution of arrived samples to use it at testing. Note that $\mathcal{Q}$ in Eq. 5.4 can depend on both appearance A and kinematic-layout K since it is used at offline training, while $\Psi(A(\cdot)^\gamma, \tau)$ depends only on A to prevent the dependency of K at testing.

Pre-trained forests $\mathcal{F}_K$, $\mathcal{F}_A$: Before training each tree, we pre-train two forests $\mathcal{F}_K$ and $\mathcal{F}_A$ using out-of-bag (OOB) samples[1] and their kinematic-layout and appearance, respectively. Forests are pre-trained to obtain two class distributions for a sample V (*i.e.* $P(y|A(V)) = \mathcal{F}_A(V)$, $P(y|K(V)) = \mathcal{F}_K(V)$) and two class distributions for each node (*i.e.* $P_{\mathcal{F}}(y|\{K(V)|V \in D\}) = \frac{1}{|D|}\sum_{V\in D}\mathcal{F}_K(V)$, $P_{\mathcal{F}}(y|\{A(V)|V \in D\}) = \frac{1}{|D|}\sum_{V\in D}\mathcal{F}_A(V)$) at each tree training. Pre-trained forests are used whenever either Q_s or Q_k is selected for each node split.

Switching term Q_s: This term measures the usefulness of kinematic-layout K and selects Ψ that clusters samples into two groups: a group whose K is useful and another group whose K is less useful. The underline rationale of this term is our observation that kinematic-layout K does not always help improve the classification accuracy. For some samples, appearance A is enough or better than kinematic-layout K (see Fig. 5.7 (a), (b) and (d)). We define the score $U(V) \in [-1,1]$ to measure the usefulness of kinematic-layout for sample V:

$$U(V) = \mathcal{F}_{K,y^*}(V) - \mathcal{F}_{A,y^*}(V) \tag{5.5}$$

where y^*, $\mathcal{F}_{K,y^*}(V)$ and $\mathcal{F}_{A,y^*}(V)$ are the the ground-truth class label, y^*-th dimensional value of $\mathcal{F}_K(V)$ and $\mathcal{F}_A(V)$, respectively. The positive $U(V)$ implies that kinematic-layout K is empirically more useful than the appearance A, while negative $U(V)$ means the opposite. The $Q_s = \left[1 + \sum_{m\in\{l,r\}} \frac{|D_m|}{|D|}\text{var}\left(\{U(V)|V \in D_m\}\right)\right]^{-1}$ prefers Ψ that clusters samples into left or right child nodes by the value of $U(V)$, where var$(\cdot)$ is the variance operator.

Appearance term Q_c: This term is same as Shannon entropy measure employed in standard classification RFs [173]. It measures the *uncertainty* of class distributions in D_l and D_r based on the appearance A. It prefers Ψ that makes the class posterior distribution, empirically the class histograms, in D_l and D_r are dominated by a certain class.

Kinematic-layout term Q_k: To prevent the explicit usage of kinematic-layout K at testing, this term implicitly exploits the K at training, by minimizing the gap between two class distributions:

[1] Samples, not used in current tree training for **b**ootstrap **agg**regat**ing**.

$P_{\mathcal{F}}(y|\{K(V)|V \in D\})$, $P_{\mathcal{F}}(y|\{A(V)|V \in D\})$ at each node training. The gap is minimized by controlling each sample's weight and Q_k is defined based on the weighted distribution as in Eq. 5.7. The weight $\mathbf{w}^* = [w_1, ..., w_{|D|}]^\top \in \mathbb{R}^{|D|\times 1}$ is optimized by:

$$\mathbf{w}^* = \min_{\mathbf{w}} ||\mathbf{A} \cdot \mathbf{w} - \mathbf{b}||_2^2 \text{ s.t. } \forall w_i \geq 0, \tag{5.6}$$

where w_i denotes each sample's weight, the i-th column of $\mathbf{A} \in \mathbb{R}^{|Y|\times|D|}$, $A_i \in \mathbb{R}^{|Y|\times 1}$ corresponds to each sample's $P(y|A(V))$ and $\mathbf{b} \in \mathbb{R}^{|Y|\times 1}$ corresponds to $P(y|\{K(V)|V \in D\})$. The Eq. 5.6 can be optimized by the least-square solver with non-negativity constraints (*e.g.* `lsqnonneg` function in MATLAB). Meanwhile, as in Fig. 5.4 (c), a sample V', whose $P(y = l_1|A(V')) = \mathcal{F}_{A,l_1}(V)$ is high, is emphasized if $P(y = l_1|\{K(V)|V \in D\}) > P(y = l_1|\{A(V)|V \in D\})$ while surpressed, otherwise (for $1 \leq l_1 \leq |Y|$). Samples with high discrepancy can be benefitted by kinematic-layouts K; thus they are emphasized and carefully considered for deciding Ψ while others are surpressed regarded as a noise. The Q_k is defined as the Shannon entropy measure on the weighted class histograms $n_{\mathbf{w}}(y, D_m)$ as follows:

$$Q_k \quad = \sum_{m\in\{l,r\}} \sum_y n_{\mathbf{w}}(y, D_m) \log \frac{n_{\mathbf{w}}(y,D_m)}{\sum_{i=1}^{|D|} w_i} \tag{5.7}$$

where $n_{\mathbf{w}}(y, D) = \sum_{V\in D} w_i \cdot \mathbb{I}(y = y^*)$ and $\mathbb{I}(\cdot)$ is an impulse function.

Discussion on the alternative kinematic-layout term. The original classification objective Q_c based on the entropy measure cannot incorporate additional variables. The kinematic-layout term Q_k is proposed to incorporate the additional kinematic-layout information K in the split objective of the RFs and it reflects K when deciding the split parameter Ψ using the Eq. 5.4. However, there could be alternative choices for Q_k: one simple alternative is the split objective which is proposed in the regression forests [54]:

$$Q_{reg} = - \sum_{m\in\{L,R\}} \sum_{i\in D_m} \left\| K_i - \frac{1}{|D_m|} \sum_{j\in D_m} K_j \right\|^2, \tag{5.8}$$

where K_i, K_j denote the kinematic-layout for the i-th and the j-th sample, respectively. Using the Q_{reg}, left and right child nodes are enforced to have samples with similar K values in the Euclidean space. As an alternative, we think of the graph cut [170, 229]-based split objective as

follows:

$$\mathcal{Q}_{cut} = -\sum_{i \in D_L} \sum_{j \in D_R} w(i, j; K), \tag{5.9}$$

where

$$w(I_j, I_k; K) = \exp\left(\frac{-\|K_i - K_j\|_2}{\sigma^2}\right). \tag{5.10}$$

Using the pairwise similarity w enforces that samples i and j are grouped together when the corresponding kinematic-layout K are similar. Using the $\mathcal{Q}_{cut}$, we can measure the smoothness of data points in each child nodes.

However, our preliminary experiments have revealed that these two methods do not work well. We think this is due to the fact that the decision boundary of actions in the kinematic-layout K space is still non-linear and more than the smoothness of the kinematic-layout K is required. Therefore, instead of using the raw values of kinematic-layouts K, we propose to use the posterior probability for the actions given kinematic-layouts learned by pre-trained RFs, $P(Y|K)$ as in the proposed $\mathcal{Q}_k$.

KLRFs training. To train KLRFs $\mathcal{F}^+$ hierarchically as in Eq. 5.3 and Fig. 5.4 (a), we propose three types of split objectives: $\mathcal{Q}_s$, $\mathcal{Q}_c$ and $\mathcal{Q}_k$, which are called as switching, appearance and kinematic-layout term, respectively. $\mathcal{Q}_s$ first measures the usefulness of K as the *if-statement* of Eq. 5.3. Then, $\mathcal{Q}_c$, $\mathcal{Q}_k$ selectively performs either $A \mapsto Y$ or $A \overset{K}{\mapsto} Y$ depending on the node characteristics. The three split objectives are combined into a $\mathcal{Q}$ by variables α, β as:

$$\mathcal{Q}(\Psi) = \alpha \mathcal{Q}_s + (1 - \alpha)\{\beta \mathcal{Q}_c + (1 - \beta)\mathcal{Q}_k\} \tag{5.11}$$

where variables α and β controls KLRFs to first select $\mathcal{Q}_s$ until certain number of data samples remain in a node and then select either $\mathcal{Q}_c$ or $\mathcal{Q}_k$ to perform further classification according to the node characteristics. As in our preliminary experiments, we find that some samples are better classified using appearance A rather than using kinematic-layouts K. This phenomenon could be observed in the Fig. 5.5 (In the figure, we visualize the usefulness of the kinematic-layouts for each action class label.) Thus, we try to use the kinematic-layouts adaptively by first clustering

samples according to the usefulness of the kinematic-layouts by Q_s. Then, we further classify samples using either kinematic layout terms Q_k (if kinematic-layout is useful) or appearance term Q_c (if kinematic-layout is less useful). Towards the objective, α and β are defined as follows:

$$\alpha = \begin{cases} 1 & \text{, if } |D| > \eta \\ 0 & \text{, otherwise} \end{cases}, \beta = \begin{cases} 1 & \text{, if } \zeta > \Delta \\ 0 & \text{, otherwise} \end{cases}$$

where $|D|$ is the number of samples in a current node, η is empirically set to 0.1 times total number of training samples, Δ is the ratio of samples having positive usefulness score $U(V)$ (Eq. 5.5) and $\zeta \in [0,1]$ is a randomly sampled value at each node. If Δ is high, it implies that K is useful in a current node. At the same time, the probability for $\zeta > \Delta$ becomes low and nodes tend to select Q_k more frequently than Q_c. If Δ is low, the opposite happens. Each tree's diversity obtained by this random configuration makes the KLRF ensemble become robust [47]. As in Fig. 5.4 (a), thanks to the hierarchical nature of trees, we are able to utilize different split objectives within a tree: nodes near the root select Q_s while nodes in the bottom gradually select either Q_c or Q_k. In our further analysis shown in the Fig. 5.6, we performed the ablative experiments on the configuration of Q_s and Q_c combined with different configuration of cues: A, $A + C_t^J$ of K and (full) K. We observed that given same cues, performing the Q_s at first consistently improves the accuracy.

5.3.3 Inference by KLRFs

Inference of KLRFs. At testing, as in Fig. 5.4 (b), Appearance feature extracted from V, $A(V)$ is passed down the KLRFs $\mathcal{F}^+$ by learned split functions $\{\Psi(A^\gamma(\cdot), \tau)\}$ until it reaches the leaf nodes, which store both the class distribution $P(y|V)$ and the kinematic vectors $K(V)$. Split nodes decide its input V goes either to the left child (if $A(V)^\gamma < \tau$) or to the right child (otherwise) according to learned split functions $\{\Psi(A(\cdot)^\gamma, \tau)\}$. The responses are averaged to output the final $P(y|V)$ and $\hat{K}(V)$ for each V.

Cross-view setting. Cross-view setting is challenging: the model is testified for unseen camera views, which have much impact on the depth appearance [151, 152]. For example, in the right-most column of the Fig. 5.2, we visualized depth frames with new viewpoints. Depth appearance A by [152] is view-invariant to a certain degree. To further help, we augment depth maps by synthetic rotations and translations as in [223], and consider their coherency using Q_v at training and kinematic consistency filter (KCF) at testing. Though both Q_v and KCF are designed for cross-view, we also apply them for single-view experiment and report their results (see Fig. 5.7 (e)).

View clustering term Q_v: This term enforces Ψ to cluster data samples according to the value of $K(V)$ at training: $Q_v = \left[1 + \sum_{m\in\{l,r\}} \frac{|D_m|}{|D|}\Lambda\left(\{K(V)|V \in D_m\}\right)\right]^{-1}$, where $\Lambda = \text{trace}(\text{var}(\cdot))$ is defined as trace of a variance operator. Augmented data (*i.e.* translated, rotated) share the same kinematic-layout K; thus they are clustered together by Q_v. As a result, it enhances the view-invariance. Either Q_v or $\mathcal{Q}$ in Eq. 5.11 is randomly selected at each node, where random selection is known effective to mix up split objectives in a forest [47].

KCF: At testing, after obtaining both $P(y|V)$ and $\hat{\mathcal{K}}(V)$ from the leaf nodes, we smooth out noisy predictions by applying the KCF to $P(y|V)$. KCF exploits pairwise similarities of inferred kinematic-layout $\hat{\mathcal{K}}(V)$ to smooth the result by: $P^*(y|V) = \frac{1}{W_p}\sum_{J'\in\mathcal{S}(V)} P(y|J)g(||\hat{K}(J) - \hat{K}(J')||)$ where $W_p = \sum_{J'\in\mathcal{S}(V)} g(||\hat{K}(J) - \hat{K}(J')||)$ is a normalizing factor, $g(\cdot)$ is a Gaussian kernel and $S(V)$ is the augmented dataset of V. $P^*(y|V)$ is the final class distribution.

5.4 Experiments

We perform both single-view (on PATIENT, CAD-60 [192] datasets) and cross-view (on PATIENT, SHUWA3D [151] datasets) experiments to validate our methods. The "Baseline

Method	PATIENT			UWA3D
	View 1	View 2	View 3	Cross View
DCSF [230]	18.7	6.7	16.0	–
HON4D [132]	21.1	6.3	13.8	28.9
HOPC [151]	28.2	15.4	23.1	52.2
DMM [223]	29.3	19.3	24.0	–
Novel View [152]	43.8	23.8	32.5	76.9
Baseline (RFs)	47.8	21.5	27.2	77.1
Ours (KLRFs)	**53.2**	**27.5**	**36.2**	**80.4**

Table 5.2: *Results for PATIENT (single-view (View 1), cross-view (View 2, 3)) and UWA3D (cross-view) datasets.*

Method	Accuracy	Precision	Recall
Testing Input: Depth			
HON4D [132]	72.7	–	–
Zhu *et al.* [255]	75.0	–	–
Baseline (RFs)	81.6	93.2	78.6
Ours (KLRFs)	87.1	92.3	85.7
Testing Input: Skeleton			
GI *et al.* [134]	–	91.9	90.2
Shan *et al.* [167]	91.9	93.8	**94.5**
Cippitelli *et al.* [27]	–	93.9	93.5
Testing Input: Depth+Skeleton			
Actionlet Ensemble [222]	74.7	–	–
Zhu *et al.* [255]	87.5	93.2	84.6
Baseline (RFs+Skeleton)	89.7	92.9	89.3
Ours (KLRFs+Skeleton)	**94.1**	**97.5**	92.7

Table 5.3: *Results for CAD60 dataset.*

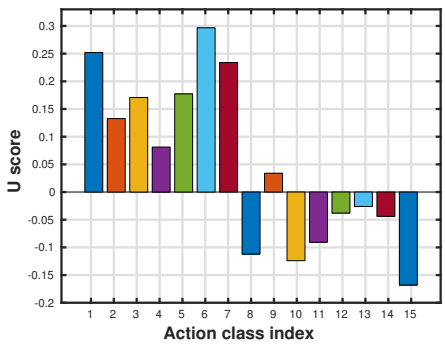
(RFs)" is the combination of depth appearance from [152] and standard RFs using additional translational, rotational data augmentation as in [223]. "Ours (KLRFs)" replaces RFs of "Baseline (RFs)" to KLRFs and consider the kinematic-layout K at training.

Same-view. We first evaluate our method for single-view action recognition using PATIENT and CAD60 datasets and each result is shown in Table 5.2 "View 1" column and Table 5.3, respectively. The classification accuracy is averaged over all classes, which corresponds to the mean of the confusion matrix diagonal. For PATIENT, we use the first 5 subjects as

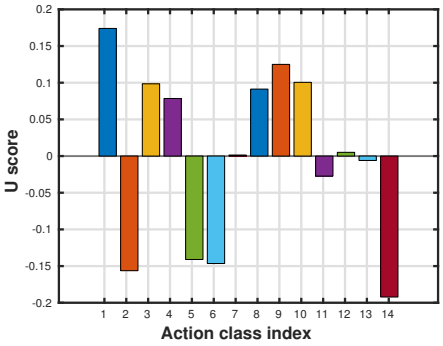
training and others as testing samples. We evaluate the recent state-of-the-art depth-based methods [132, 151, 152, 223, 230] using their publicly available codes. Our method produces a significant accuracy gain ($6-10\%$) over these methods. For CAD60, we follow the cross-person experimental settings in [68, 222]. We also use two more measures (*i.e.* Precision/Recall) to compare with various state-of-the-arts for this dataset. KLRFs show good accuracy compared to depth-based approaches [132, 255]. Since this conventional dataset contains mostly upright humans with frontal views, most state-of-the-arts use real-time obtained skeleton joints at testing to obtain their results. Thus, for fair comparison, we combine skeleton cues to our method at testing: We train half of trees as RFs using pure skeletons and half of trees as KLRFs (denoted as "Ours (KLRFs+Skeleton)"). Also, "Baseline (RFs+Skeleton)" consists with half skeleton-based RFs and half depth-based RFs. Showing 5% accuracy gain to "Baseline (RFs+Skeleton)", "Ours (KLRFs+Skeleton)" shows the best result in Table 5.3.

Cross-view. We also applied our method to cross-view experiments using PATIENT and SHUWA3D datasets. For SHUWA3D, we follow the same experimental setting as in [152]. Averaged accuracy for all cross-views is reported in the "SHUWA3D Cross View" column of Table 5.2. For PATIENT dataset, we applied the same model trained in the single-view setting to View 2 and View 3 for cross-view tesing. The results are summarized in "View 2", "View 3" columns of PATIENT in Table 5.2, respectively. "Baseline (RFs)" often performs worse than [152] in cross-view experiments, while "Ours (KLRFs)" shows consistent accuracy improvement.

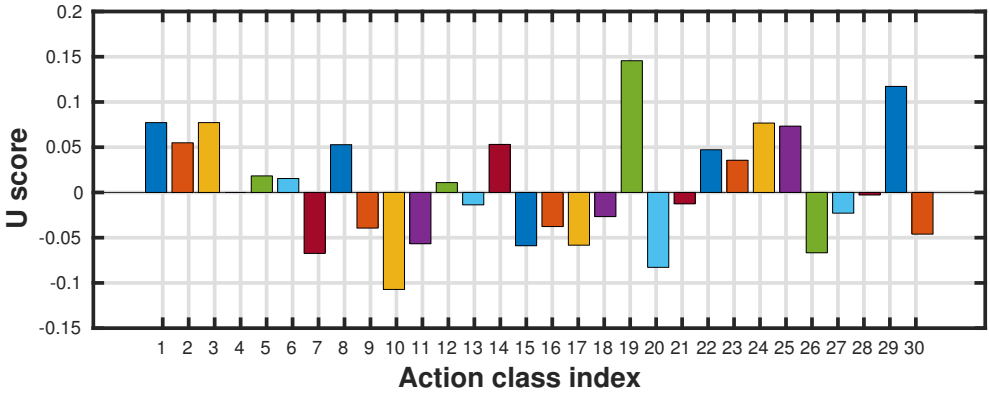
Usefulness score U vs. classes. In Fig. 5.7 (a), (b) and (d), We plot the averaged usefulness score $U(V) \in [-1,1]$ for samples in each action classes. Positive $U(V)$ means that K is more useful than A, while negative $U(V)$ means the opposite. The results imply that in PATIENT dataset, static actions (*i.e.* (1)-(7)) are well explained by K rather than A while dynamic actions (*i.e.* (8)-(15)) are well classified by using only A without K. In CAD60 and SHUWA3D datasets, we also report the usefulness scores per each class, showing variations. Class index is



(a) U score vs. class in PATIENT.



(b) U score vs. class in CAD60.



(c) U score vs. classes in UWA3D.

Figure 5.5: *Further analysis 1: Usefulness score U vs. classes in the databases.*

given in Sec. 5.2 for PATIENT and in the supplementary page for other datasets.

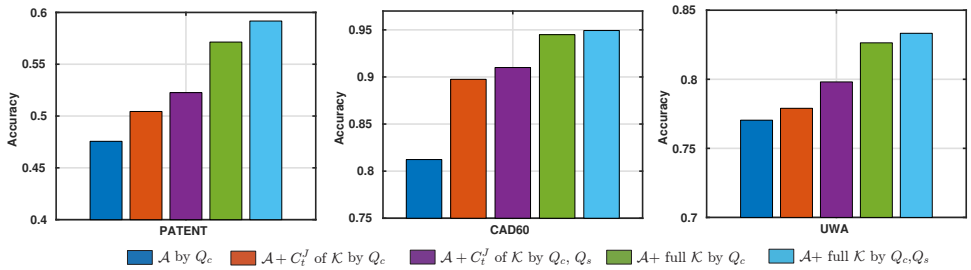


Figure 5.6: *Further analysis 2: Utilizing K at testing for each database*

Utilizing K at testing. To test the strength of kinematic-layout K, we report the classification accuracy explicitly using ground-truths of K as input features in Fig. 5.7 (c). Note that utilizing K at testing is not realistic in our scenario; we conducted the experiments only for evaluation

purpose using ground-truths of K. We configure 3 different features $\{A, A + \mathbf{C}_t^J$ of $K, A + K\}$ and two classifiers by standard RFs (*i.e.* Q_c) and KLRFs using $Q_c + Q_s$ terms. The graph shows that K offers $5 - 10\%$ accuracy gain, when combined with A.

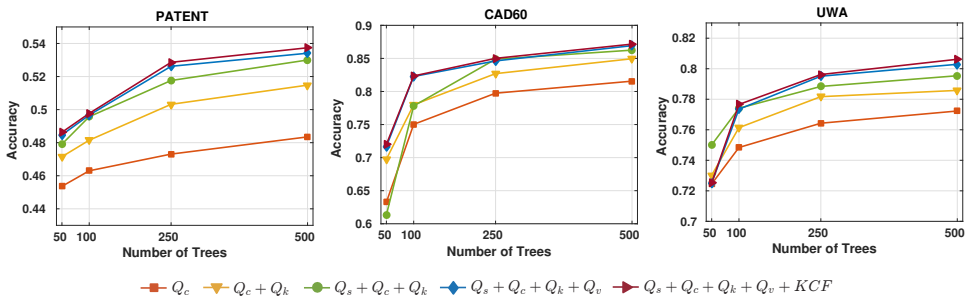


Figure 5.7: *Parameter sensitivity: PATIENT(left), CAD60 (mid), UWA3D (right). Action classification accuracy according to the different number of trees in each databases. The accuracy saturates around* 500 *trees.*

Sensitivity to parameters. We evaluate the sensitivity of our model depending on tree numbers in Fig. 5.7 (e). The performance increases as tree numbers increase and saturates around 500 trees. Component analysis is further reported in the same figure.

5.5 Conclusion

In this chapter, we study the problem of depth-based action recognition in a 24 hours-monitoring patient actions in a ward, with the goal of effectively recognizing human actions in the hospital scenario. We try to improve the overall action recognition accuracy by exploiting the scene layout and skeleton information in the training process. We propose the KLRF to encode this prior information, thereby capturing more geometry that provides greater discriminant power in action classification. Even though this method encodes the 3D geometry information rather implicitly only at the training stage, the obtained accuracy improvement was significant, showing that the 2D-to-3D ambiguity could be relieved by this method.

CHAPTER 6

CONCLUSION

Contents

We have proposed methods to exploit the 3D geometry in the context of estimating 3D poses and actions for especially human bodies and hands. This chapter concludes this thesis by summarizing key achievements and describing the future work.

6.1 Summary of thesis achievements

As mentioned in the chapter 1, estimating poses and actions of human bodies and hands has many applications while it also involves many in-the-wild challenges. Among challenges, we try to focus on the two, which is closely tied with the incorporation of the 3D geometry:

- **2D-to-3D mapping ambiguity**: While the interaction between human-human, hand-hand, hand/human-objects occurs in the 3D space, captured inputs (i.e. RGB images, Depth

maps) are projected into the 2D space. 3D information (i.e. multi-viewed appearance, cross-view invariance) is missing in the obtained inputs; while the lost information is often critical to reason about poses and actions of human bodies and hands.

- **Insufficient data with quality 3D annotations**: There have been no principled way of collecting 1) large-scale datasets 2) with high-quality 3D pose annotations 3) in an automatic way yet. Furthermore, planning and collecting a new dataset requires huge amount of manual efforts.

We designed chapters to deal with challenges we mentioned above:

In chapter 3, we proposed to reconstruct the new data samples in the form of 2.5D depth maps using the conditional GAN framework. The attributes of the new samples are manipulated in the aspects of a shape as well as a viewpoint, to jointly tackle both challenges. The synthesized 2.5D depth maps are used as the new training samples where the hand pose model is trained on. By this data augmentation scheme, we demonstrated that the state-of-the-art accuracy is obtained in the depth-based hand pose estimation task.

In chapter 4, we proposed to reconstruct the full 3D meshes of hands. Since there is no proper dataset to train the dense hand pose estimation network that can estimates 3D meshes based on RGB images, we formulate our framework to use the indirect coarse supervision whose input is RGB images and outputs are 3D skeletons and 2D segmentation masks using the differetiable renderer and skeleton regressors. Furthermore, during the training, as the estimated 3D meshes are in the hand pose distribution, we generate new samples by manipulating their shape and viewpoint parameters and re-projecting them into the new RGB images. We observe that the proposed framework and methods are able to generate 3D meshes and the pose the 3D meshes are better than previous skeleton-based coarse approaches.

In chapter 5, we have demonstrated that the 3D geometry can be used as context for the action recognition problem. Our algorithm is developed based on the random forest classifier and on top of it, we developed the method that can use 3D geometry during the training stage, to obtain

the more robust model. We have demonstrated that the information used is able to improve the overall action recognition accuracy by 5-7%.

In the appendix, we proposed a new multimodal synthetic database named BigHandMesh, which has complete articulation space of BigHand2.2M database while having wide range of viewpoint/shape variation capability with the aid of fitted 3D deformable mesh model. Also, we could obtain multimodal cues: 2D/3D skeletons, 3D meshes, RGB images, Depth maps, 2D segmentation masks, 2D part-based segmentation masks having 6 hand parts. We expect this database could be usefully used in the multi-modal learning in the hand domain , combining RGB-based and depth-based approaches which was tackled separately as in chapter 3 and 4.

Overall, we have found the method to compensate losing 3D information, insufficiency of data, coarseness of the representation using the full 3D mesh representation and obtained improved results in each sub-tasks.

6.2 Future Work

Although estimating poses and actions of human bodies and hands have achieved great improvements in past years, it is still challenging and on-going research problem in computer vision community which have many remaining issues:

- **Multiple subject pose estimation**: Most pose estimation frameworks are based on detection modules exploiting the simple heuristics (hands are the foremost objects) in the depth domain or external hand detector [256] in the RGB domain. 1) If multiple hands/humans are in the scene, the development of high-quality detection methods are required. Exploiting interactions between pose estimator and subject detector will be informative. 2) Also, the new poses like “shaking hands”, “praying” will be introduced which were ignored in the single hand pose estimation scenario. New dataset and new methods need to be developed to tackle this new challenges.

- **Multiple subject action recognition**: It is hard to deal with multiple subjects in action recognition problem since most of actions are annotated for the main subjects in each video. If multiple subjects are involved in the video and they are performing different actions, new dataset need to collected and methods based on local subject bounding boxes need to be developed.

The 3D information is very useful and properly incorporating dense hands and humans are not well established in the action recognition framework yet. In the future, more methods need to be developed for this direction to verify the helpfulness of the 3D geometry in both recognizing poses and actions:

- **Joint end-to-end learning of multi-modal tasks (i.e. estimating full 3D meshes, full 3D scenes and action/gesture recognition)**: Multiple tasks in each individual domains, for example, 2D/3D skeleton and 3D mesh estimation from depth/RGB maps, 2D segmentation and etc. are challenging tasks by itself. These multimodal tasks can be learned together to improve each individual task's performance in the similar aspect with the work of "taskonomy" [245].

‘

Appendix

BIGHANDMESH: A NOVEL MULTIMODAL SYNTHETIC DATASET

Contents

In this chapter, a novel multimodal synthetic benchmark that can be used for different computer vision tasks involving human hands, including 3D mesh information, RGB-D hand pose annotations and 2D/3D hand/part segmentation masks is proposed. This is intended to encourage the development of pose estimation methods exploiting more diverse and complete attribute challenges, and at the same time encourage the future aggregation of multi-modal tasks in the hands domain (i.e. RGB-based, depth-based, segmentation, pose estimation and etc).

The generated dataset covers diverse hand pose variations by leveraging the existing large scale depth hand pose dataset BigHand2.2M, and the deformable 3D hand mesh model MANO. First,

the large scale dataset is strategically subsampled to reduce articulation redundancy and the 3D mesh model is fitted to the sampled hand skeletons. After distilling the articulation space, shape and viewpoint parameters of the 3D meshes are swept to cover the natural variability of the human hand in images. Finally, with the use of realistic textures and background RGB images along with mesh-to-skeleton regressors and 2D projection renderers, the multiple modality data are generated. Data coverage in terms of hand articulations, shapes and viewpoints is compared with those of existing benchmarks, showing that the proposed benchmark fills the gaps of previous work. Furthermore, experiments on a real RGB-based hand pose estimation dataset by using the proposed dataset in training demonstrate the quality and quantity of the generated data. To conclude, we expect our proposed dataset to accelerate research on hand tasks that have been previously hindered by lack of a quality large benchmark, such as RGB-based 3D hand pose estimation and semantic hand part segmentation.

A.1 Motivation

One of of the most important challenges in understanding hands from a computer vision perspective is the access to a significant amount of annotated data to be able to train modern machine learning systems, usually requiring the learning of non-linear and high dimensional mappings of noisy, occluded, and highly variable images. Typical applications of hands require estimating the similar number of joint parameters as in human body pose. However, compared to body pose, hands often present severe self-occlusions making hard to define canonical views, while human bodies usually appear isolated and in upright postures in images. Therefore, the ability to operate reliably under varying viewpoints, poses, and shapes is crucial to accomplish the tasks of interest. A simple yet most promising approach to face such challenge is by training e.g. CNN on a large-scale dataset which covers such variations. However, to the best of our knowledge, most existing datasets are limited in the coverage of camera viewpoint, shape, and pose variations. The main difficulty to obtain a large scale dataset lies in procurement of ground-truth annotations in the form of either skeleton joints or segmentation masks, which often

Table A.1: *Comparison of related RGB-based hand pose datasets: DO [185], STB [248], RHD [256], SH [118], GANHAND [117].*

	[185]	[248]	[256]	[118]	[117]	Ours
Depth	✓	✓	✓	✓	✗	✓
RGB	✓	✓	✓	✓	✓	✓
(Part-based) hand mask	✗	✗	✓	✗	✗	✓
3D mesh	✗	✗	✗	✗	✗	✓
# Shape (subjects)	2	1	16	2	7	31 and in-between
Viewpoints	3rd	3rd	3rd	Ego	Ego	Complete
Skeleton annotations	5 tips	21	21	21	21	21
Real (R)/Synth. (S)	R	R	S	S	S	S
Number of Frames	3,014	18k	41k+2.7k	63k	330k	100k×(#view)×(#shape)

involves laborious efforts. To relieve this, data augmentation methods have been frequently used in hand pose domain, and have observed major accuracy improvements [126, 128]. Synthetic datasets have also been used to augment the data space [117, 176]; random sampling is usually adopted to generate the dataset and there is, nonetheless, no guarantee of capturing diverse variations of hands. The gap between synthetic and real data also has been the issue in this approach, and several methods were proposed [148, 174] to tackle this. Yuan et al. [244] proposed a systematic method to collect real database in the depth domain, equipping magnetic sensors on hands and proposed the biggest dataset up to date, BigHand2.2M. Baek et al. [6] tried to fill the remaining gap in data space by synthesizing data using the cyclic consistency and changing viewpoints and shapes of existing data entries. However, note the variation ranges were limited to, i.e. ± 45 standard deviation for rotation angles, ± 0.5 standard deviation for bone length change ratio, due to limited extrapolation capability of GAN. Also, the approach for shape variations is relatively simple over more sophisticated models such as MANO [161], a parametric deformable 3D hand model.

In this chapter, we propose a pipeline to complete the three main variations of hands, i.e shapes, viewpoints and articulations, and transfer them to computer vision tasks involving hands. With the aid of the annotated 3D skeletons of BigHand2.2M, we first fit the hand model MANO to these skeletons. As the dataset was collected densely there are redundant articulations among different subjects, we cluster data in the articulation angle space first. We select $K = 100,000$ distinct articulation samples and fit the hand model to the selected data.

Shape and viewpoint parameters are manipulated to complete the data coverage. Finally, RGB-D images, 2D/3D skeletons, 2D segmentation masks and part-based segmentation masks are automatically generated using a Neural renderer and the 3D skeleton regressor provided by the MANO model. After generating the dataset, we compare the data coverage of the proposed to existing RGBD-based hand pose datasets. Furthermore, we perform a comparative quantitative experiment on a RGB-based hand pose estimation with and without the use of the proposed dataset in training, showing state-of-the-art performance on a real RGB dataset without the use of real data in training. In summary, this work has the following major contributions:

• **A pipeline to systematically generate a complete and multimodal dataset**: A deformable 3D hand mesh model is fitted to an existing real depth benchmark that has satisfactorily complete articulation space. Shape and viewpoint are generated by changing corresponding parameters of the model. Finally, using 2D projection methods, multiple modality data from the 3D mesh model are generated. The pipeline is shown in Figure A.1 .

• **A large scale synthetic dataset**: In computer vision tasks such as 3D mesh reconstruction, 2D/3D pose estimation and segmentation tasks, the lack of a complete dataset has been often pointed out by the community. This work attempts to fill the gap and encourages multimodal research on hands by making this dataset publicly accessible.

• **Analyses on the dataset quality**: The proposed dataset is compared with existing datasets in terms of the three main hand domain variations: shape, viewpoint and articulation. Furthermore, the completeness of the dataset is empirically shown, from which the state-of-the-art hand pose estimation accuracy is obtained.

A.2 BigHandMesh Generation

In this section, we will explain how we generate the BigHandMesh benchmark exploiting the MANO deformable 3D mesh model [161] and BigHand2.2M benchmark [182].

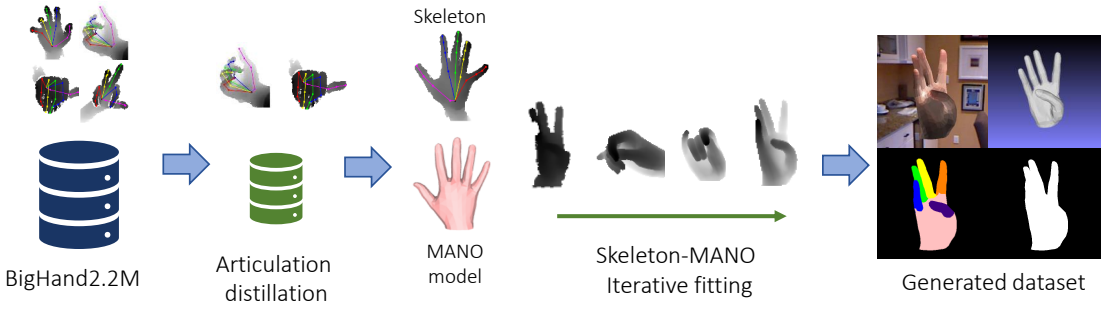
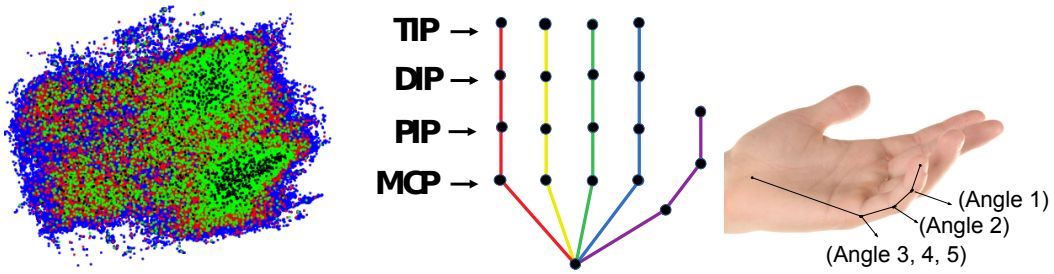


Figure A.1: *Schematic diagram of the proposed pipeline for generating the hand benchmark. We first select distinct articulations from BigHand2.2M database, then fit the MANO hand model to their skeletons, finally RGB-D, skeletons, segmentation masks are generated.*

A.2.1 Distilling hand articulations from BigHand2.2M

As described in Yuan et al. [182], during their dataset collection 10 subjects were requested to perform $2^5 = 32$ extreme articulations by completely folding or stretching fingers to cover ${}_{32}C_2 = 496$ hand postures and thus continuous transitions in-between 32 extreme articulations were collected. By using this systematic scheme, the database could be regarded as *complete* in the hand articulation space. However, since the dataset was collected in a continuous and dense way, several redundant postures were captured. Utilizing all redundant articulations is inefficient considering the time taken for a deep neural network training. This leads us to the first step: reducing the redundancy in the articulation space from a total $957,032$ training samples publicly available via the ICCV'17 HANDS workshop challenge [243] and then fit the MANO model to the sampled skeletons.

Selecting distinct hand articulations. The 63-dimensional raw skeleton vectors composed of the 3D coordinate values x, y and z of the 21 joints are affected by the main three hand variations, i.e. viewpoint, shape and articulation. To distill articulations only, we propose to extract 25-dimensional angle features, 5 angles for each finger as depicted in Fig. A.2(c)). K-means algorithm is applied on top of these angular feature vectors. A cluster size of $K = 100,000$ is empirically set, as it can be observed in Fig. A.2(a), the angular space containing the 496 continuous transitions in-between 32 extreme articulations could be sufficiently covered by this number. K skeletons closest to each cluster mean are selected for further processing.



(a) *(b)* *(c)*

Figure A.2: *(a) 2-dimensional PCA plot for 25-dimensional angle feature space depending on different K values: 1,000, 10,000, 100,000 and 957,032. We select 100,000 skeletons from total 957,032 skeletons to reduce the redundancy, (b) Our hand model having 21 joints, (c) Angles used extracting 25-dimensional angular features.*

Gradient-based MANO model fitting. We fit the MANO model [161]'s shape $\mathbf{s} = \{s_j\}_{j=1}^{10}$, camera $\mathbf{c} = \{c_j\}_{j=1}^{8}$ and articulation $\mathbf{a} = \{a_j\}_{j=1}^{45}$ parameters to the i-th raw skeletons of selected articulations $\mathbf{z} = \{z_i\}_{i=1}^{K}$, by solving the following equation:

$$(\mathbf{s}^{i*}, \mathbf{c}^{i*}, \mathbf{a}^{i*}) = \arg\min_{(\mathbf{s},\mathbf{c},\mathbf{a})} O(\mathbf{s}, \mathbf{c}, \mathbf{a}, \mathbf{z}^i)), \forall i \in [1, K], \tag{A.1}$$

where our proposed objective function $O(\mathbf{s}, \mathbf{c}, \mathbf{a}, \mathbf{z}^i)$ for the sample i is defined as follows:

$$O(\mathbf{s}, \mathbf{c}, \mathbf{a}, \mathbf{z}^i) = ||f^{reg}(V(\mathbf{s}, \mathbf{c}, \mathbf{a})) - \mathbf{z}^i||_2^2 + \sum_{j=1}^{10} \|\mathbf{s}_j\|_2^2 + R_{Lap}(V(\mathbf{s}, \mathbf{c}, \mathbf{a})), \tag{A.2}$$

where $V(\mathbf{s}, \mathbf{c}, \mathbf{a})$ denotes the parametric 3D mesh with three parameters $\mathbf{s}, \mathbf{c}, \mathbf{a}$. Eq. A.2 is composed of the following terms: i) the Euclidean distance between 3D skeleton ground-truths $\mathbf{z}^i$ and the current MANO mesh model's 3D skeleton values $f^{reg}(V(\mathbf{s}, \mathbf{c}, \mathbf{a}))$[1]; ii) A shape regularizer enforcing the shape parameters $\mathbf{s}$ to be close to their MANO model's mean values, normalized to 0 as in [161], to maximize the shape likelihood; and iii) A Laplacian regularizer $R_{Lap}(V(\mathbf{s}, \mathbf{c}, \mathbf{a}))$ to obtain the smooth mesh surfaces similar to [78]. Eq. A.1 is solved iteratively by using the gradients from Eq. A.2 as follows:

$$(\mathbf{s}_{t+1}, \mathbf{c}_{t+1}, \mathbf{a}_{t+1}) = (\mathbf{s}_t, \mathbf{c}_t, \mathbf{a}_t) - \gamma \cdot \nabla O(\mathbf{s}_t, \mathbf{c}_t, \mathbf{a}_t, \mathbf{z}^i), \forall t \in [1, T] \tag{A.3}$$

[1] f^{reg} geometrically regresses the 3D skeleton from the 3D mesh vertices coordinates. This regressor is provided with the MANO model and its weights are fixed during the process. The provided regressor does not have finger tips, which we define similar to [7, 16].

Figure A.3: *Fitting process: Visualized with depth images. (Col. 1) Targetted hand pose in BigHand2.2M, (Col. 2-7) Fitting results in different iterations, (Col. 8) Final fitted hand pose obtained. Note that even though there is slightly difference between original one and the final result, we only use the final output and its self-data generation capability to label it.*

where $\gamma = 10^{-3}$ and $T = 3,000$ are empirically set. This process is similar to the refinement step from [7, 209], which refines estimated meshes by using the gradients from the loss.

In Fig. A.3, both the target and the fitted depth images during the process described by Eq. A.3 are depicted. The obtained meshes are slightly different from their original inputs, however this is not a problem for our purpose given that we will generate input and output pairs of the fitted model by exploiting fitted meshes' self-data generation capability while ignoring original depth and skeletons. Here the aim of fitting the hand model is to obtain a plausible and complete articulation space.

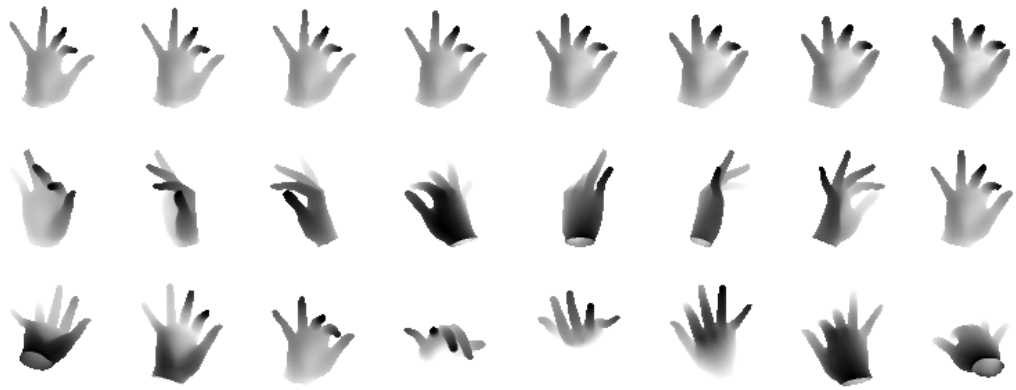


Figure A.4: *Example viewpoint/shape variation results: (Row 1) Shape parameter variation, (Row 2) Viewpoint (Azimuth) variation, (Row 3) Viewpoint (Elevation) variation. With the 3D mesh model, we can represent complete viewpoint spaces and diverse shape spaces from* thin *to* fat *hands.*

A.2.2 Varying shape and viewpoint parameters of the fitted hand model

Although BigHand2.2M dataset is complete in the articulation space, the combinatorial with viewpoints and shapes is not, given that only five subjects are available in the training data and viewpoint variation is randomly directed during the data collection. To overcome this issue we further perform viewpoint and shape parameters variations. The fitted hand meshes could be rotated with rotation matrices constructed by angles relative to x and y axes. The 10 shape PCA coefficients can also be manipulated to change the mesh shapes. For the shape variation, we bounded shape entries within three times of the standard deviation to avoid implausible hand poses. Varying shape and viewpoint examples are shown in Fig. A.4.

A.2.3 Data generation for heterogeneous tasks

After generating meshes covering a relatively complete space of articulation, shape, and viewpoint spaces, we use these meshes to generate diverse modality data with the help of a 3D skeleton regressor and rendering engines. The resultant examples of six modalities are plotted

in Fig. A.5.

2D and 3D skeletons. The 3D skeleton is obtained by directly applying the 3D skeleton regressor provided by the MANO model. Since the provided 3D skeleton regressor geometrically infers the 16 skeleton joints from the mesh vertices, we additionally define the finger tip vertices to obtain a total of 21 skeleton joints. This 21 skeletal joint model is shared by recent benchmarks in hand pose estimation [182]. 2D skeleton is obtained by projecting the 3D skeletons to the 2D space.

RGB maps. We use the neural renderer proposed in [79] to 2D raycast the 3D mesh into 2D images with textures. Textures are recovered similarly to the work of [16] by geometrically comparing the MANO original scans and registered meshes. Since original scans have textures on the mesh vertices, we retrieve the closest vertices to 3 vertices of each face and average them in the face. To improve the reality of the texture, we apply the following post processing steps: 1) Since ambient and light components are not separated in the original MANO scan, we convert RGB values to YUV space and increase Y values within the dark regions, while decreasing Y values within bright regions. 2) We apply a bilateral filter with 9 pixel neighborhood diameter, 60 spatio-color sigma values to smooth out “blocky" textures due to triangular faces. The generated RGB images are of size 224×224.

Depth maps. We use the same neural renderer as above to generate 96×96 sized depth maps with intensity values in the range of $[0, 255]$.

Hand and part segmentation masks. Vertice-to-part labels with 6 part labels, i.e. palm and 5 finger regions, are defined. This label map is attached to the mesh faces and projected to the 2D space using the neural renderer. Hand segmentation masks are generated by gathering all the part segmentation masks.

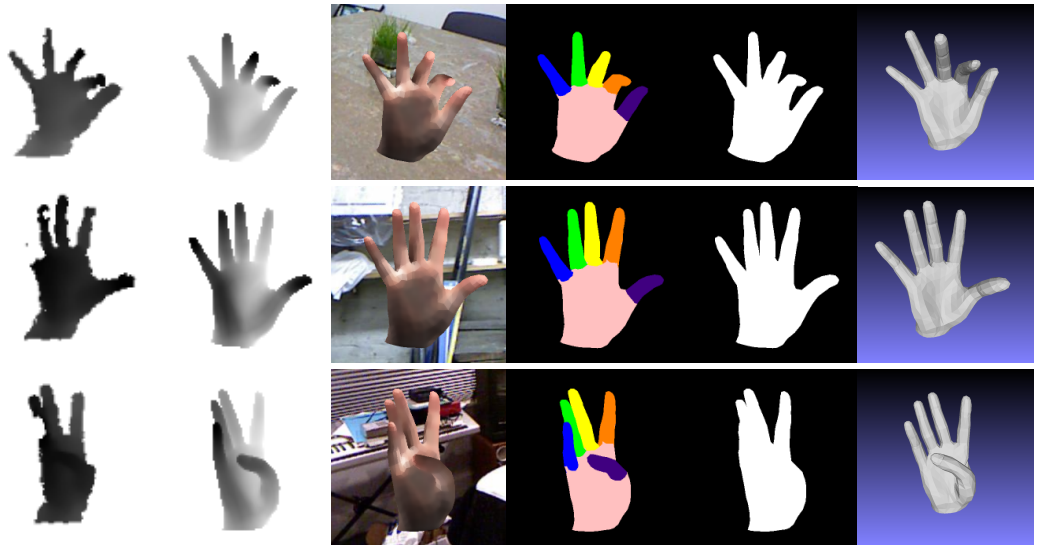


Figure A.5: *Example RGB-D maps, segmentation masks, mesh and skeletons obtained: (Col. 1) Real depth maps* $\mathbf{x}$ *and corresponding (Col. 2) synthesized depth maps at iteration* $3,000$*, (Col. 3) RGB maps, (Col. 4) Part-segmentation maps, (Col. 5) Segmentation maps, (Col. 6) Full 3D mesh.*

A.3 Experiments: Dataset quality analysis

In this section we analyze the quality of proposed BigHandMesh dataset. First, we compare our dataset with other publicly available related datasets, i.e. STB [248], RHD [256], and SH [118], in terms of articulation space by using a PCA projection. Second, we compare our texture model to those of the mentioned datasets. Finally, to quantitatively evaluate the benefit the proposed dataset, we train a hand pose estimator that maps RGB images to 3D skeletons with and without our dataset.

A.3.1 Related datasets comparison

As presented in Table A.1, shape and viewpoint spaces of our BigHandMesh dataset are more complete compared to conventional RGB-based hand pose databases, i.e. STB [248], RHD [256], and SH [118]. Shape and viewpoint variations of BigHandMesh dataset are visualized in the Fig. A.4. In Fig. A.6 (a) textures and articulations are compared.

Textures. In the top of Fig. A.6 (a), we qualitatively compare our generated textures with those of the other datasets. We can see that our texture is moderately closer to real dataset STB compared to other synthetic datasets RHD and SH.

Articulations. In the bottom of Fig. A.6 (a), the articulation space of the four compared datasets is shown. As expected, the articulation space of our data is more dense compared to the related datasets. This could be explained by the fact that STB contains simple counting hand gestures, while RHD and SH contain only few synthetic action poses.

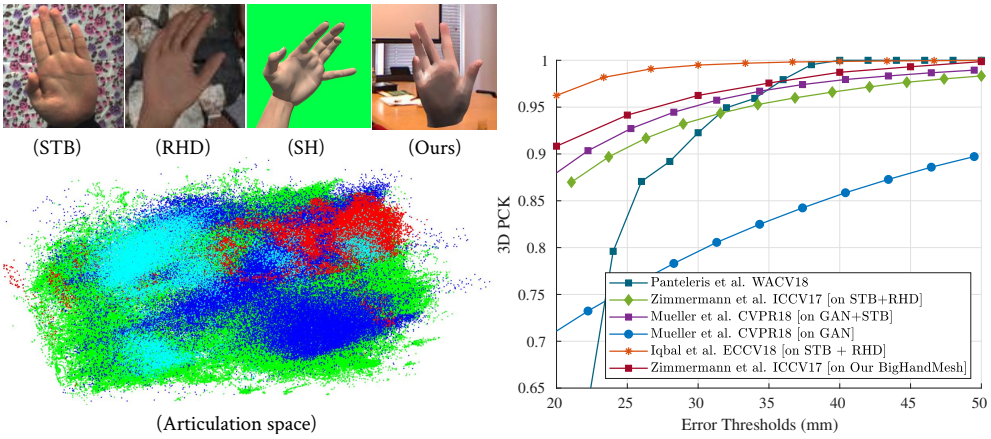
A.3.2 RGB-to-3D skeleton experiments

In this section we quantitatively evaluate the benefit the proposed dataset. For this, we train a RGB-based hand pose estimator [256] with and without the use of our dataset and compare the obtained results with the state-of-the-art on STB dataset [248].

Implementation details. For the RGB-based hand pose estimator we use the architecture presented in [256] whose code is publicly available online, by exploiting three types of networks: hand detector, heat map-based 2D skeleton estimator and 3D skeleton estimation network. We initialized our architecture using pre-trained weights provided on the author's website. We fine-tune our architecture using $K = 100,000$ articulations combined with random viewpoint and shape parameters at each epoch. We used Adam optimizer with a learning rate of 10^{-3} and default β parameters.

Result analysis. We compare the trained hand pose estimator [256] with the use of our proposed dataset with four state-of-the-art RGB-based 3D skeleton estimation algorithms [18, 117, 133, 256]. These approaches are trained using slightly different combinations of datasets and we detail specify this in the legend of Fig. A.6 (b). In the case of [133] given that it is a generative fitting method does not require a training dataset.

In Fig. A.6 (b) we observe the benefit of using our synthetic dataset to train a RGB-based hand pose estimator even when the test data is real. Training [256] with BigHandMesh leads to a



(a) Texture/Articulation comparison *(b) Accuracy on STB database*

Figure A.6: *Comparison of databases: (a) (Top) Texture comparison, (Bottom) PCA plot for "articulation" space, Different colors (RHD, STB, SH and Ours) denote samples from different databases. (b) RGB-based hand pose estimation results trained on different databases.*

significant improvement in performance compared to training with STB and RHD. The approach of Iqbal et al. [18] is the best performing approach among all the evaluated ones. However, it uses a sophisticated RGB-to-Depth reconstruction module within its network architecture, making it difficult to draw any conclusions with respect to the use of our dataset. Mueller et al. [117] also proposed and to use a synthetically generated data by enhancing SH dataset using CycleGAN. Compared to Mueller et al. [117]'s work, our trained hand pose estimator achieves a higher accuracy. Considering that this approach uses a deeper ResNet architecture, using our dataset with stronger architectures could lead to a better performance. Furthermore, Mueller et al. [117]'s work shows a significant accuracy drop when the real dataset STB is not included in the training stage. Our proposed dataset leads to a higher performance even without the aid of real data.

A.4 Conclusion

In this chapter, a novel multimodal synthetic benchmark including 3D mesh, RGB-D hand pose annotations and 2D/3D hand/part segmentation masks is proposed. In chapters 3 and 4, we tackle the depth-based and RGB-based hand pose estimation, respectively. In this chapter, we

try to combine the largest-scale “depth”-based dataset (Big Hand 2.2M) and 3D model (MANO) to generate more complete synthetic datasets. This dataset could be used for both “RGB”-based and “depth”-based hand pose estimation works. We expect our proposed dataset to accelerate research on hand tasks that have been previously hindered by lack of a quality large benchmark, such as RGB-based 3D hand pose estimation and semantic hand part segmentation.

Finally, in order to encourage research on these directions, we are organizing the “HANDS 2019 Challenge” (https://sites.google.com/view/hands2019/challenge) in 5th International Workshop on HANDS 2019 (https://sites.google.com/view/hands2019/) held in conjunction with the ICCV’19. We encourage participants to augment their training pose datasets using initial 3D mesh model we fitted.

REFERENCES

[1] IntelSR300. http://click.intel.com/intelrrealsensetm-developer-kit-featuring-sr300.html. Accessed: 2017-11-12. 53

[2] T. Alldieck, M. Magnor, B. Bhatnagar, C. Theobalt, and G. Pons-Moll. Learning to reconstruct people in clothing from a single RGB camera. In *CVPR*, 2019. 5

[3] B. Allen, B. Curless, and Z. Popovic. The space of human body shapes: Reconstruction and parameterization from range scans. *ACM Transactions on Graphic*, 2003. 14

[4] D. Anguelov, P. Srinivasan, D. Koller, S. Thrun, J. Rodgers, and J. Davis. Scape: Shape completion and animation of people. *ACM Transactions on Graphic*, 2005. 14, 16

[5] S. Baek, K. I. Kim, and T.-K. Kim. Real-time online action detection forests using spatio-temporal contexts. In *WACV*, 2017. 1, 64

[6] S. Baek, K. I. Kim, and T.-K. Kim. Augmented skeleton space transfer for depth-based hand pose estimation. In *CVPR*, 2018. 7, 111

[7] S. Baek, K. I. Kim, and T.-K. Kim. Pushing the envelope for RGB-based dense 3D hand pose estimation via neural rendering. In *CVPR*, 2019. 114, 115

[8] S. Baek, Z. Shi, M. Kawade, and T.-K. Kim. Kinematic-layout-aware random forests for depth-based action recognition. In *BMVC*, 2017. 1, 64

[9] L. Ballan, A. Taneja, J. Gall, L. V. Gool, and M. Pollefeys. Motion capture of hands in action using discriminative salient points. In *ECCV*, 2012. 15

[10] M. Blank, L. Gorelick, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. In *ICCV*, 2005. 28

[11] V. Blanz and T. Vetter. A morphable model for the synthesis of 3D faces. In *SIGGRAPH*, 1999. 15

[12] V. Bloom, D. Makris, and V. Argyriou. G3D: A gaming action dataset and real time action recognition evaluation framework. In *CVPR Workshop*, 2012. 6

[13] A. F. Bobick and J. W. Davis. The recognition of human movement using temporal templates. *TPAMI*, 2001. 28

[14] F. Bogo, A. Kanazawa, C. Lassner, P. Gehler, J. Romero, and M. J. Black. Keep it SMPL: Automatic estimation of 3D human pose and shape from a single image. In *ECCV*, 2016. 14, 16

[15] J. Booth, A. Roussos, S. Zafeiriou, A. Ponniah, and D. Dunaway. A 3D face model for pose and illumination invariant face recognition. In *CVPR*, 2016. 15

[16] A. Boukhayma, R. Bem, and P. H. S. Torr. 3D hand shape and pose from images in the wild. In *CVPR*, 2019. 21, 114, 117

[17] A. Brunton, A. Salazar, T. Bolkart, and S. Wuhrer. Review of statistical shape spaces for 3D data with comparative analysis for human faces. *CVIU*, 2014. 15

[18] Y. Cai, L. Ge, J. Cai, and J. Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *ECCV*, 2018. 1, 7, 8, 23, 66, 69, 71, 74, 79, 81, 119, 120

[19] C. Cao, Y. Weng, S. Zhou, Y. Tong, and K. Zhou. Facewarehouse: A 3D facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 2014. 15

[20] Z. Cao, T. Simon, S.-E. Wei, , and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *CVPR*, 2017. 14, 22

[21] J. Carreira and A. Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, 2017. 33

[22] J. Charles, T. Pfister, D. Magee, D. Hogg, and A. Zisserman. Upper body pose estimation with temporal sequential forests. In *BMVC*, 2014. 32

[23] C. Chen, R. Jafari, and N. Kehtarnavaz. UTD-MHAD: A multimodal dataset for human action recognition utilizing a depth. In *ICIP*, 2015. 92

[24] C. Chen and D. Ramanan. 3d human pose estimation = 2d pose estimation + matching. In *CVPR*, 2017. 23

[25] J. Chen, H. M. Le, P. Carr, Y. Yue, and J. J. Little. Learning online smooth predictors for realtime camera planning using recurrent decision trees. In *CVPR*, 2016. 32

[26] C. Choi, S. H. Yoon, C.-N. Chen, and K. Ramani. Robust hand pose estimation during the interaction with an unknown object. In *ICCV*, 2017. 64

[27] E. Cippitelli, S. Gasparrini, E. Gambi, and S. Spinsante. A human activity recognition system using skeleton data from RGBD sensors. In *Computational Intelligence and Neuroscience*, 2016. 101

[28] A. M. Clerke, J. P. Clerke, and R. D. Adams. Effects of hand shape on maximal isometric grip strength and its reliability in teenagers. *Journal of Hand Therapy*, 2005. 5, 18, 38, 43

[29] H. Dai, N. Pears, W. Smith, and C. Duncan. A 3d morphable model of craniofacial shape and texture variation. In *ICCV*, 2017. 15

[30] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, 2005. 28

[31] M. Dantone, J. Gall, G. Fanelli, and L. V. Gool. Real-time Facial Feature Detection using Conditional Regression Forests . In *CVPR*, 2012. 20

[32] V. Delaitre, D. F. Fouhey, I. Laptev, J. Sivic, A. Gupta, and A. A. Efros. Scene semantics from long-term observation of people. In *ECCV*, 2012. 27, 31, 94

[33] M. Ding and G. Fan. Articulated gaussian kernel correlation for human pose estimation. In *CVPR Workshop*, 2015. 20

[34] P. Dollár, , V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, 2005. 27, 28

[35] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *CVPR*, 2015. 33

[36] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015. 31

[37] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *CVPR*, 2015. 89

[38] A. A. Efros, A. C. Berg, G. Mori, and J. Malik. Recognizing actions at a distance. In *ICCV*, 2003. 28

[39] A. A. Fallahi and A. A. Jadidian. The effect of hand dimensions, hand shape and some anthropometric characteristics on handgrip strength in male grip athletes and non-athletes. *Journal of Human Kinetics*, 2011. 5, 18, 38, 43

[40] A. Fathi, X. Ren, and J. M. Rehg. Learning to recognize objects in egocentric activities. In *CVPR*, 2011. 34

[41] C. Feichtenhofer, H. Fan, J. Malik, and K. He. SlowFast Networks for Video Recognition. In *ICCV*, 2019. 6

[42] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *CVPR*, 2016. 27, 28, 32, 34

[43] S. Fothergill, H. Mentis, P. Kohli, and S. Nowozin. Instructing people for training gestural interactive systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2012. 32, 33

[44] D. F. Fouhey, V. Delaitre, A. Gupta, A. A. Efros, I. Laptev, and J. Sivic. People Watching: Human Actions as a Cue for Single View Geometry. *IJCV*, 2014. 31, 94

[45] J. Gall, A. Yao, and L. V. Gool. 2D Action Recognition Serves 3D Human Pose Estimation. In *ECCV*, 2010. 20, 30

[46] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *TPAMI*, 2011. 32

[47] J. Gall, A. Yao, N. Razavi, L. V. Gool, and V. Lempitsky. Hough forests for object detection, tracking, and action recognition. *TPAMI*, 2011. 99, 100

[48] G. Garcia-Hernando, H. Chang, I. Serrano, O. Déniz, and T.-K. Kim. Transition Hough forest for trajectory-based action recognition. In *WACV*, 2016. 30

[49] G. Garcia-Hernando and T.-K. Kim. Transition forests: learning discriminative temporal transitions for action recognition and detection. In *CVPR*, 2017. 1, 30

[50] G. Garcia-Hernando, S. Yuan, S. Baek, and T.-K. Kim. First-person hand action benchmark with RGB-D videos and 3D hand pose annotations. In *CVPR*, 2018. x, 1, 3, 7, 21, 26, 35, 64

[51] L. Ge, Y. Cai, J. Weng, and J. Yuan. Hand PointNet: 3D hand pose estimation using point sets. In *CVPR*, 2018. 23, 24

[52] L. Ge, H. Liang, J. Yuan, and D. Thalmann. Robust 3D Hand pose estimation in single depth images: from single-view CNN to multi-view CNNs. In *CVPR*, 2016. 23, 53

[53] L. Ge, H. Liang, J. Yuan, and D. Thalmann. 3D convolutional neural networks for efficient and robust hand pose estimation from single depth images. In *CVPR*, 2017. 23, 53, 54

[54] R. Girshick, J. Shotton, P. Kohli, A. Criminisi, and A. Fitzgibbon. Efficient regression of general-activity human poses from depth images. In *ICCV*, 2011. 20, 97

[55] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 6

[56] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014. 40

[57] R. A. Güler, N. Neverova, and I. Kokkinos. DensePose: Dense human pose estimation in the wild. In *CVPR*, 2018. 16, 18

[58] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011. 94

[59] T. Gupta, A. Schwing, and D. Hoiem. No-frills human-object interaction detection: factorization, layout encodings and training techniques. In *ICCV*, 2017. 31

[60] A. Haquea, B. Peng, Z. Luo, A. Alahi, S. Yeung, and L. Fei-Fei. Towards viewpoint invariant 3D human pose estimation. In *ECCV*, 2016. 89, 90, 91

[61] N. Hasler, H. Ackermann, B. Rosenhahn, T. Thormhlen, and H. P. Seidel. Multilinear pose and body shape estimation of dressed subjects from image sets. In *CVPR*, 2010. 16

[62] N. Hasler, C. Stoll, M. Sunkel, B. Rosenhahn, and H.-P. Seidel. A statistical model of human pose and body shape. *Computer Graphics Forum*, 2009. 14

[63] Y. Hasson, G. Varol, D. Tzionas, I. Kalevatykh, M. J. Black, I. Laptev, and C. Schmid. Learning joint reconstruction of hands and manipulated objects. In *CVPR*, 2019. 16

[64] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 21, 27, 69, 76

[65] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the spatial layout of cluttered rooms. In *ICCV*, 2009. 31

[66] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural compu- tation*, 1997. 32

[67] H. Hu, X. Gao, J. Li, J. Wang, and H. Liu. Calibrating human hand for teleoperating the hit/dlr hand. In *ICRA*, 2004. 17

[68] J.-F. Hu, W.-S. Zheng, J. Lai, and J. Zhang. Jointly learning heterogeneous features for RGB-D activity recognition. In *CVPR*, 2015. 89, 102

[69] Z. Huang, C. Wan, T. Probst, and L. V. Gool. Deep learning on lie groups for skeleton-based action recognition. In *CVPR*, 2017. 31

[70] U. Iqbal, P. Molchanov, T. Breuel, J. Gall, and J. Kautz. Hand pose estimation via latent 2.5D heatmap regression. In *ECCV*, 2018. 1, 7, 8, 23, 66, 69, 71, 73, 74, 79, 80, 81

[71] T. Ishihara, K. M. Kitani, W.-C. Ma, H. Takagi, and C. Asakawa. Recognizing hand-object interactions in wearable camera videos. In *ICIP*, 2015. 34, 35

[72] A. Jain, J. Tompson, M. Andriluka, G. W. Taylor, and C. Bregler. Learning human pose estimation features with convolutional networks. In *ICLR*, 2014. 22

[73] Y. Jang, S.-T. Noh, H. J. Chang, T.-K. Kim, and W. Woo. 3D Finger Cape: Clicking action and position estimation under self-occlusions in egocentric viewpoint. *TVCG*, 2015. 32

[74] V. John, E. Trucco, and S. Ivekovic. Human articulated tracking using hierarchical particle swarm optimization. *Image and Vision Computing*, 2010. 17

[75] H. Joo, H. Liu, L. Tan, L. Gui, B. Nabbe, I. Matthews, T. Kanade, S. Nobuhara, and Y. Sheikh. Panoptic studio: a massively multiview system for social motion capture. In *ICCV*, 2015. 22

[76] H. Joo, T. Simon, and Y. sheikh. Total capture: A 3D deformation model for tracking faces, hands, and bodies. In *CVPR*, 2018. 15, 16

[77] A. Kanazawa, M. J. Black, D. W. Jacobs, and J. Malik. End-to-end recovery of human shape and pose. In *CVPR*, 2018. 16, 18, 66, 68, 71, 72, 81

[78] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 69, 75, 114

[79] H. Kato, Y. Ushiku, and T. Harada. Neural 3D mesh renderer. In *CVPR*, 2018. 16, 18, 19, 65, 68, 73, 78, 117

[80] Q. Ke, M. Bennamoun, S. An, F. Sohel, and F. Boussaid. A new representation of skeleton sequences for 3d action recognition. In *CVPR*, 2017. 31

[81] J. Kennedy and R. Eberhart. Particle Swarm Optimization. In *IEEE International Conference on Neural Networks*, 1995. 17, 79

[82] C. Keskin, F. Kirac, Y. E. Kara, and L. Akarun. Hand pose estimation and hand shape classification using multi-layered randomized decision forests. In *ECCV*, 2012. 20

[83] S. Khamis, J. Taylor, J. Shotton, C. Keskin, S. Izadi, and A. Fitzgibbon. Learning an efficient model of hand shape variation from depth images. In *CVPR*, 2015. 15, 17

[84] A. U. Khan and A. Borji. Analysis of hand segmentation in the wild. In *CVPR*, 2018. xiv, 6, 69, 81, 82, 83

[85] H. Kim, M. Zollöfer, A. Tewari, J. Thies, C. Richardt, and C. Theobalt. Inversefacenet: Deep single-shot inverse face rendering from a single image. In *CVPR*, 2018. 78

[86] D. Kingma and J. B. Adam. A method for stochastic optimization. In *ArXiv:1412.6980*, 2014. 51

[87] A. Klaser, M. Marszalek, and C. Schmid. A Spatio-Temporal Descriptor Based on 3D-Gradients. In *BMVC*, 2008. 28

[88] M. Kocabas, S. Karagoz, and E. Akbas. Multiposenet: Fast multiperson pose estimation using pose residual network. In *ECCV*, 2018. 22

[89] Y. Kong and Y. Fu. Bilinear Heterogeneous Information Machine for RGB-D action recognition. In *CVPR*, 2015. 89

[90] P. Krejov, A. Gilbert, and R. Bowden. Combining discriminative and model based approaches for hand pose estimation. *FG*, 2015. 67

[91] A. Krizhevsky, Sutskever, Ilya, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *NIPS*, 2012. 21, 27, 32

[92] I. Laptev. On Space-Time Interest Points. *IJCV*, 2005. 28, 29

[93] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 28

[94] P. Li, H. Liang, X. Li, and C. Liao. 3D hand pose estimation using randomized decision forest with segmentation index points. In *ICCV*, 2015. 21

[95] T. Li, T. Bolkart, M. J. Black, H. Li, and J. Romero. Learning a model of facial shape and expression from 4D scans. In *ACM Transactions on Graphics*, 2017. 15

[96] T.-M. Li, M. Aittala, F. Durand, , and J. Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *ACM Transactions on Graphics*, 2018. 19

[97] W. Li, Z. Zhang, and Z. Liu. Action recognition based on a bag of 3d points. In *CVPR Workshop*, 2010. 33

[98] Z. Li, K. Gavrilyuk, E. Gavves, M. Jain, and C. G. Snoek. Videolstm convolves, attends and flows for action recognition. *CVIU*, 2018. 33

[99] C. Lien. A scalable model-based hand posture analysis system. In *MVA*, 2005. 17

[100] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3D reasoning. In *ICCV*, 2019. 18, 19

[101] M. Loper, N. Mahmood, and M. J. Black. MoSh: Motion and shape capture from sparse markers. *TOG*, 2014. 16

[102] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black. SMPL: A skinned multi-person linear model. *ToG*, 2015. 2, 14, 15, 18, 70

[103] M. M. Loper and M. J. Black. OpenDR: An approximate differentiable renderer. In *ECCV*, 2014. 18, 19

[104] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI*, 1981. 29

[105] J. Luo, W. Wang, and H. Qi. Group Sparsity and Geometry Constrained Dictionary Learning for Action Recognition from Depth Maps. In *ICCV*, 2013. 30

[106] M. Ma, H. Fan, and K. M. Kitani. Going deeper into first-person activity recognition. In *CVPR*, 2016. 34, 35

[107] B. Mahasseni and S. Todorovic. Latent multitask learning for view-invariant action recognition. In *ICCV*, 2013. 7

[108] B. Mahasseni and S. Todorovic. Regularizing long short term memory with 3D human-skeleton sequences for action recognition. In *CVPR*, 2016. 20, 30

[109] J. Malik, A. Elhayek, F. Nunnari, K. Varanasi, K. Tamaddon, A. Heloir, and D. Stricker. DeepHPS: End-to-end estimation of 3D hand pose and shape by learning from synthetic depth. In *3DV*, 2018. 65

[110] V. Mansinghka, T. D. Kulkarni, Y. N. Perov, and J. Tenenbaum. Approximate bayesian image interpretation using generative probabilistic graphics programs. In *NIPS*, 2013. 19

[111] J. Martinez, R. Hossain, J. Romero, and J. J. Little. A simple yet effective baseline for 3d human pose estimation. In *ICCV*, 2017. 23

[112] P. Matikainen, M. Hebert, and R. Sukthankar. Trajectons: Action recognition through the motion analysis of tracked features. In *ICCV Workshop*, 2009. 29

[113] S. Melax, L. Keselman, and S. Orsten. Dynamics based 3D skeletal hand tracking. In *i3D*, 2013. 15, 17

[114] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV*, 2009. 29

[115] K. Mikolajczyk and H. Uemura. Action Recognition with Motion-Appearance Vocabulary Forest. In *CVPR*, 2008. 32

[116] G. Moon, J. Y. Chang, and K. M. Lee. V2V-PoseNet: Voxel-to-voxel prediction network for accurate 3D hand and human pose estimation from a single depth map. In *CVPR*, 2018. 23

[117] F. Mueller, F. Bernard, O. Sotnychenko, D. Mehta, S. Sridhar, D. Casas, and C. Theobalt. GANerated hands for real-time 3D hand tracking from monocular RGB. In *CVPR*, 2018. xviii, 26, 27, 79, 80, 81, 111, 119, 120

[118] F. Mueller, D. Mehta, O. Sotnychenko, S. Sridhar, D. Casas, and C. Theobalt. Real-time hand tracking under occlusion from an egocentric RGB-D sensor. In *ICCV*, 2017. xviii, 26, 111, 118

[119] A. Mujika, F. Meier, and A. Steger. Fast-Slow recurrent neural networks. In *NIPS*, 2017. 27

[120] A. Newell, K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. In *ECCV*, 2016. 22

[121] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *CVPR*, 2015. 33

[122] B. X. Nie, P. Wei, and S.-C. Zhu. Monocular 3d human pose estimation by predicting depth on joints. In *ICCV*, 2017. 23

[123] B. X. Nie, C. Xiong, and S.-C. Zhu. Joint Action Recognition and Pose Estimation From Video. In *CVPR*, 2015. 30

[124] X. Nie, J. Feng, J. Xing, and S. Yan. Pose partition networks for multi-person pose estimation. In *ECCV*, 2018. 22

[125] J. C. Niebles, H. Wang, and L. Fei-Fei. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 28

[126] M. Oberweger and V. Lepetit. DeepPrior++: Improving fast and accurate 3D hand pose estimation. In *ICCV Workshop*, 2017. 5, 21, 111

[127] M. Oberweger, G. Riegler, P. Wohlhart, and V. Lepetit. Efficiently creating 3D training data for fine hand pose estimation. In *CVPR*, 2016. 7

[128] M. Oberweger, P. Wohlhart, and V. Lepetit. Hands deep in deep learning for hand pose estimation. In *CVWW*, 2015. 5, 21, 111

[129] M. Oberweger, P. Wohlhart, and V. Lepetit. Training a Feedback Loop for Hand Pose Estimation. In *ICCV*, 2015. xiii, 24, 53, 59, 60

[130] I. Oikonomidis, N. Kyriazis, and A. A. Argyros. Efficient model-based 3D tracking of hand articulations using kinect. In *BMVC*, 2011. 17, 53

[131] D. Oneata, J. Verbeek, and C. Schmid. Action and event recognition with fisher vectors on a compact feature set. In *ICCV*, 2013. 29

[132] O. Oreifej and Z. Liu. HON4D: histogram of oriented 4D normals for activity recognition from depth sequences. In *CVPR*, 2013. 6, 92, 101, 102

[133] P. Panteleris, I. Oikonomidis, and A. Argyros. Using a single RGB frame for real time 3D hand pose estimation in the wild. In *WACV*, 2018. 16, 66, 79, 81, 119

[134] G. I. Parisi, C. Weber, and S. Wermter. Self-organizing neural integration of pose-motion features for human action recognition. In *Frontier in Neurobotics*, 2015. 101

[135] E. Park, J. Yang, E. Yumer, D. Ceylan, and A. C. Berg. Transformation-grounded image generation network for novel 3D view synthesis. In *CVPR*, 2017. 40

[136] G. Park, A. A. Argyros, and W. Woo. Efficient 3d hand tracking in articulation subspaces for the manipulation of virtual objects. In *CGI*, 2016. 17

[137] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros. Context encoders: Feature learning by inpainting. In *CVPR*, 2016. 38, 46, 49

[138] G. Pavlakos, V. Choutas, N. Ghorbani, T. Bolkart, A. A. A. Osman, D. Tzionas, and M. J. Black. Expressive body capture: 3D hands, face, and body from a single image. In *CVPR*, 2019. 15

[139] G. Pavlakos, L. Zhu, X. Zhou, and K. Daniilidis. Learning to estimate 3D human pose and shape from a single color image. In *CVPR*, 2018. 66, 68, 69

[140] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action recognition with stacked fisher vectors. In *ECCV*, 2014. 29

[141] F. Perronnin, J. Sanchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In *ECCV*, 2010. 29

[142] T. Pfister, J. Charles, and A. Zisserman. Flowing convnets for human pose estimation in videos. In *ICCV*, 2015. 22

[143] H. Pirsiavash and D. Ramanan. Detecting activities of daily living in first-person camera views. In *CVPR*, 2012. 34

[144] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, and P. Gehler. DeepCut: Joint subset partition and labeling for multi person pose estimation. In *CVPR*, 2016. 16

[145] G. Poier, K. Roditakis, S. Schulter, D. Michel, H. Bischof, and A. A. Argyros. Hybrid one-shot 3d hand pose estimation by exploiting uncertainties. In *BMVC*, 2015. 17

[146] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *CVPR*, 2017. 24

[147] C. Qian, X. Sun, Y. Wei, X. Tang, and J. Sun. Realtime and robust hand tracking from depth. In *CVPR*, 2014. 17, 24

[148] M. Rad, M. Oberweger, and V. Lepetit. Feature mapping for learning fast and accurate 3D pose inference from synthetic images. In *CVPR*, 2018. 1, 6, 8, 27, 111

[149] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. In *ICLR*, 2016. xi, 40, 46, 47

[150] H. Rahmani, A. Mahmood, D. Q. Huynh, and A. Mian. HOPC: histogram of oriented principal components of 3D pointclouds for action recognition. In *ECCV*, 2014. 6

[151] H. Rahmani, A. Mahmood, D. Q. Huynh, and A. Mian. Histogram of oriented principal components for cross-view action recognition. *TPAMI*, 2016. 30, 89, 90, 91, 92, 100, 101, 102

[152] H. Rahmani and A. Mian. 3D action recognition from novel viewpoints. In *CVPR*, 2016. 30, 33, 89, 92, 93, 100, 101, 102

[153] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You Only Look Once: Unified, real-time object detection. In *CVPR*, 2016. 6

[154] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. In *CVPR*, 2017. 6

[155] E. Remelli, A. Tkach, A. Tagliasacchi, and M. Pauly. Low-dimensionality calibration through local anisotropic scaling for robust hand model personalization. In *ICCV*, 2017. 16, 17

[156] K. M. Robinette, S. Blackwell, H. Daanen, M. Boehmer, S. Fleming, T. Brill, D. Hoeferlin, and D. Burnsides. Civilian american and european surface anthropometry resource (caesar) final report. In *Technical Report AFRL-HE-WP-TR-2002-0169, US Air Force Research Laboratory*, 2002. 14

[157] G. Rogez, M. Khademi, J. S. III, J. M. M. Montiel, and D. Ramanan. 3D hand pose detection in egocentric RGB-D images. In *ECCV*, 2014. 20, 34

[158] G. Rogez, J. Rihan, C. Orrite-Uruneula, and P. H. Torr. Fast human pose detection using randomized hierarchical cascades of rejectors. *IJCV*, 2012. 20

[159] G. Rogez, J. S. Supancic, and D. Ramanan. First-person pose recognition using egocentric workspaces. In *CVPR*, 2015. 35

[160] J. Romero, H. Kjellstrom, and D. Kragic. Monocular real-time 3D articulated hand pose estimation. *Humanoids*, 2009. 19

[161] J. Romero, D. Tzionas, and M. J. Black. Embodied hands: Modeling and capturing hands and bodies together. In *SIGGRAPH Asia*, 2017. xiv, 2, 15, 67, 68, 70, 72, 111, 112, 114

[162] M. Savva, A. X. Chang, P. Hanrahan, M. Fisher, and M. Niesner. SceneGrok: Inferring Action Maps in 3D Environments. *ACM Transactions on Graphics (TOG)*, 2014. 31

[163] P. Scovanner, S. Ali, and M. Shah. A 3-dimensional sift descriptor and its application to action recognition. In *ACM International Conference on Multimedia*, 2007. 28

[164] L. Seidenari, V. Varano, S. Berretti, A. Bimbo, and P. Pala. Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses. In *CVPRW*, 2013. 33

[165] I. Serrano, O. Deniz, G. Bueno, G. Garcia-Hernando, and T.-K. Kim. Spatio-temporal elastic cuboid trajectories for efficient fight recognition using hough forests. In *Machine Vision and Applications*, 2018. 32

[166] A. Shahroudy, J. Liu, T.-T. Ng, and G. Wang. NTU RGB+D: a large scale dataset for 3D human activity analysis. In *CVPR*, 2016. 33, 92

[167] J. Shan and S. Akella. 3D human action segmentation and recognition using pose kinetic energy. In *IEEE Workshop on Advanced Robotics and its Social Impacts*, 2014. 101

[168] A. Sharifi, A. Harati, and A. Vahedian. Marker based human pose estimation using annealed particle swarm optimization with search space partitioning. In *ICCKE*, 2014. 17

[169] T. Sharp, C. Keskin, D. Robertson, J. Taylor, J. Shotton, D. Kim, C. Rhemann, I. Leichter, A. Vinnikov, Y. Wei, D. Freedman, P. Kohli, E. Krupka, A. Fitzgibbon, and S. Izadi. Accurate, robust and flexible real-time hand tracking. In *CHI*, 2015. 24

[170] J. Shi and J. Malik. Normalized cuts and image segmentation. *TPAMI*, 2000. 97

[171] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from a single depth image. In *CVPR*, 2011. 19, 20, 34, 35

[172] J. Shotton, R. Girshick, A. Fitzgibbon, T. Sharp, M. Cook, M. Finocchio, R. Moore, P. Kohli, A. Criminisi, and A. Kipman. Efficient human pose estimation from single depth images. *TPAMI*, 2013. 29, 33, 88

[173] J. Shotton, M. Johnson, and R. Cipolla. Semantic texton forests for image categorization and segmentation. In *CVPR*, 2008. 96

[174] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, and R. Webb. Learning from simulated and unsupervised images through adversarial training. In *CVPR*, 2017. 6, 8, 27, 38, 53, 111

[175] N. Silberman, P. Kohli, D. Hoiem, and R. Fergus. Indoor segmentation and support inference from RGBD images. In *ECCV*, 2012. 78

[176] T. Simon, H. Joo, I. Matthews, and Y. Sheikh. Hand keypoint detection in single images using multiview bootstrapping. In *CVPR*, 2017. 8, 22, 26, 66, 111

[177] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 27, 28, 32, 34

[178] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. In *ICLR*, 2015. 21, 27

[179] B. Singh, T. K. Marks, M. Jones, O. Tuzel, and M. Shao. A multi-Stream bi-directional recurrent neural network for fine-grained action detection. In *CVPR*, 2016. 34

[180] S. Singh, C. Arora, and C. V. Jawahar. First person action recognition using deep learned descriptors. In *CVPR*, 2016. 35

[181] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. SurfNet: Generating 3D shape surfaces using deep residual networks. In *CVPR*, 2017. 65

[182] A. Sinha, A. Unmesh, Q. Huang, and K. Ramani. SurfNet: Generating 3D shape surfaces using deep residual networks. In *CVPR*, 2017. 112, 113, 117

[183] A. Spurr, J. Song, S. Park, and O. Hilliges. Cross-modal deep variational hand pose estimation. In *CVPR*, 2018. 79, 81

[184] S. Sridhar, F. Mueller, A. Oulasvirta, and C. Theobalt. Fast and robust hand tracking using detection-guided optimization. In *CVPR*, 2015. 24

[185] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from RGB-D input. In *ECCV*, 2016. xviii, 7, 20, 79, 111

[186] S. Sridhar, F. Mueller, M. Zollhöfer, D. Casas, A. Oulasvirta, and C. Theobalt. Real-time joint tracking of a hand manipulating an object from rgb-d input. In *ECCV*, 2016. 26

[187] S. Sridhar, A. Oulasvirta, and C. Theobalt. Interactive markerless articulated hand motion tracking using RGB and depth data. In *ICCV*, 2013. 15

[188] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller. Multi-view convolutional neural networks for 3D shape recognition. In *ICCV*, 2015. 4

[189] M. Sun, P. Kohli, and J. Shotton. Conditional Regression Forests for Human Pose Estimation. In *CVPR*, 2012. 20

[190] X. Sun, Y. Wei, S. Liang, X. Tang, and J. Sun. Cascaded hand pose regression. In *CVPR*, 2015. 20, 21, 24, 25, 38, 42, 53

[191] J. Sung, C. Ponce, B. Selman, and A. Saxena. Human activity detection from RGBD images. In *AAAI workshop on Pattern, Activity and Intent Recognition*, 2011. x, 3

[192] J. Sung, C. Ponce, B. Selman, and A. Saxena. Unstructured human activity detection from RGBD images. In *ICRA*, 2012. 92, 100

[193] D. J. Tan, T. Cashman, J. Taylor, A. Fitzgibbon, D. Tarlow, S. Khamis, S. Izadi, and J. Shotton. Fits like a glove: Rapid and reliable hand shape personalization. In *CVPR*, 2016. xv, 84

[194] D. Tang, H. Chang, A. Tejani, and T.-K. Kim. Latent regression forest: structural estimation of 3D articulated hand posture. *TPAMI*, 2016. 20, 21, 25, 38, 42, 53

[195] D. Tang, J. Taylor, P. Kohli, C. Keskin, T.-K. Kim, and J. Shotton. Opening the black box: hierarchical sampling optimization for estimating human hand pose. In *ICCV*, 2015. 21, 53, 54

[196] D. Tang, T.-H. Yu, and T.-K. Kim. Real-time articulated hand pose estimation using semi-supervised transductive regression forests. In *ICCV*, 2013. 20, 21

[197] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *CVPR*, 2012. 36, 54

[198] B. Tekin, F. Bogo, and M. Pollefeys. H+O: Unified egocentric recognition of 3D hand-object poses and interactions. In *CVPR*, 2019. 35

[199] B. Tekin, P. Marquez-Neila, M. Salzmann, and P. Fua. Learning to fuse 2D and 3D image cues for monocular body pose estimation. In *ICCV*, 2017. 23

[200] A. Tilmax and M. Shah. Actions sketch: A novel action representation. In *CVPR*, 2005. 28

[201] A. Tkach, M. Pauly, and A. Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. *ACM Transactions on Graphics*, 2016. 15

[202] A. Tkach, M. Pauly, and A. Tagliasacchi. Sphere-meshes for real-time hand modeling and tracking. In *SIGGRAPH Asia*, 2016. 18

[203] A. Tkach, A. Tagliasacchi, E. R. M. Pauly, and A. Fitzgibbon. Online generative model personalization for hand tracking. *ACM Trans. on Graphics*, 2017. 18

[204] D. Tome, C. Russell, and L. Agapito. Lifting from the deep: Convolutional 3D pose estimation from a single image. In *CVPR*, 2017. 23

[205] J. Tompson, M. Stein, Y. Lecun, and K. Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *TOG*, 2014. 21, 24, 25, 39, 42, 67

[206] J. J. Tompson, A. Jain, Y. LeCun, and C. Bregler. Joint training of a convolutional network and a graphical model for human pose estimation. In *NIPS*, 2014. 22

[207] A. Toshev and C. Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 22

[208] S. Tulsiani, S. Gupta, D. F. Fouhey, and A. A. Efros. Factoring shape, pose and layout from the 2D image of a 3D scene. In *CVPR*, 2018. 31

[209] H.-Y. F. Tung, H.-W. Tung, E. Yumer, and K. Fragkiadaki. Self-supervised learning of motion capture. In *NIPS*, 2017. 18, 66, 68, 69, 115

[210] M. Valstar, B. Martinez, X. Binefa, and M. Pantic. Facial point detection using boosted regression and graph models. In *CVPR*, 2010. 20

[211] G. Varol, I. Laptev, and C. Schmid. Long-term temporal convolutions for action recognition. *TPAMI*, 2017. 33

[212] V. Veeriah, N. Zhuang, and G. Qi. Differential recurrent neural networks for action recognition. In *ICCV*, 2015. 31

[213] R. Vemulapalli, F. Arrate, and R. Chellappa. Human action recognition by representing 3D skeletons as points in a lie group. In *CVPR*, 2014. 30, 89, 93

[214] C. Wan, T. Probst, L. V. Gool, and A. Yao. Crossing nets: dual generative models with a shared latent space for hand pose estimation. In *CVPR*, 2017. xiii, 53, 59, 60

[215] C. Wan, A. Yao, and L. V. Gool. Hand Pose Estimation from Local Surface Normals. In *ECCV*, 2016. 20, 21

[216] C. Wang, Y. Wang, and A. Yuille. An approach to pose-based action recognition. In *CVPR*, 2013. 30

[217] C. Wang, Y. Wang, and A. L. Yuille. An Approach to Pose-Based Action Recognition. In *CVPR*, 2013. 27, 29

[218] C. Wang, Y. Wang, and A. L. Yuille. Mining 3d key-pose-motifs for action recognition. In *CVPR*, 2016. 31

[219] H. Wang, S. Gould, and D. Koller. Discriminative Learning with Latent Variables for Cluttered Indoor Scene Understanding. In *ECCV*, 2010. 31

[220] H. Wang, A. Klaser, C. Schmid, and C.-L. Liu. Action Recognition by Dense Trajectories. In *CVPR*, 2011. 29

[221] H. Wang, M. M. Ullah, A. Klaser, and I. Laptev. Evaluation of local spatio-temporal features for action recognition. In *BMVC*, 2009. 29

[222] J. Wang, Z. Liu, Y. Wu, and J. Yuan. Learning actionlet ensemble for 3D human action recognition. *TPAMI*, 2014. 30, 33, 89, 92, 101, 102

[223] P. Wang, W. Li, Z. Gao, J. Zhang, C. Tang, and P. Ogunbona. Action Recognition from Depth Maps Using Deep Convolutional Neural Networks. In *IEEE Transactions on Human Machine Systems*, 2015. 29, 89, 100, 101, 102

[224] R. Y. Wang and J. Popovic. Real-time hand-tracking with a color glove. *ACM Transactions on Graphic*, 2009. 16, 19

[225] X. Wang, R. Girshick, A. Gupta, and K. He. Non-local neural networks. In *CVPR*, 2018. 33

[226] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *CVPR*, 2016. x, 3, 6, 16, 21, 22, 35, 69

[227] D. Wu and L. Shao. Leveraging hierarchical parametric networks for skeletal joints based action segmentation and recognition. In *CVPR*, 2014. 34

[228] M.-Y. Wu, Y. H. Tang, P.-W. Ting, and L.-C. Fu. Hand pose learning: combining deep learning and hierarchical refinement for 3D hand pose estimation. . In *BMVC*, 2017. 25

[229] Z. Wu and R. Leahy. An optimal graph theoretic approach to data clustering: theory and its application to image segmentation. *TPAMI*, 1993. 97

[230] L. Xia and J. K. Aggarwal. Spatio-Temporal Depth Cuboid Similarity Feature for Activity Recognition Using Depth Camera. In *CVPR*, 2013. 101, 102

[231] L. Xia, C.-C. Chen, and J. K. Aggarwal. View invariant human action recognition using histograms of 3D joints. In *CVPR Workshop on Human Activity Understanding from 3D Data*, 2012. 33

[232] D. Xiang, H. Joo, and Y. Sheikh. Monocular total capture: posing face, body, and hands in the wild. In *CVPR*, 2019. 15

[233] X. Yang, C. Zhang, and Y. Tian. Recognizing Actions Using Depth Motion Maps-based Histograms of Oriented Gradients. In *ACM Multimedia*, 2012. 29

[234] Y. Yang, C. Fermuller, Y. Li, and Y. Aloimonos. Grasp type revisited: a modern perspective on a classical feature for vision. In *CVPR*, 2015. 35

[235] A. Yao, J. Gall, G. Fanelli, and L. V. Gool. Does Human Action Recognition Benefit from Pose Estimation? In *BMVC*, 2011. 20, 30, 34

[236] A. Yao, J. Gall, G. Fanelli, and L. J. Van Gool. Does human action recognition benefit from pose estimation?. In *BMVC*, 2011. 19

[237] M. Ye and R. Yang. Real-time simultaneous pose and shape estimation for articulated objects using a single depth camera. In *CVPR*, 2014. 20

[238] Q. Ye, S. Yuan, and T.-K. Kim. Spatial attention deep net with partial PSO for hierarchical hybrid hand pose estimation. In *ECCV*, 2016. 22, 67

[239] R. Yu, H. Wang, A. Li, J. Zheng, V. I. Morariu, and L. S. Davis. Layout-induced video representation for recognizing agent-in-place actions. In *ICCV*, 2017. 31

[240] T.-H. Yu, T.-K. Kim, and R. Cipolla. Real-time action recognition by spatiotemporal semantic and structural forests. In *BMVC*, 2010. 32

[241] T.-H. Yu, T.-K. Kim, and R. Cipolla. Unconstrained monocular 3D human pose estimation by action detection and cross-modality regression forest. In *CVPR*, 2013. 6, 30

[242] S. Yuan, G. Garcia-Hernando, B. Stenger, G. Moon, J. Y. Chang, K. M. Lee, P. Molchanov, J. Kautz, S. Honari, L. Ge, J. Yuan, X. Chen, G. Wang, F. Yang, K. Akiyama, Y. Wu, Q. Wan, M. Madadi, S. Escalera, S. Li, D. Lee, I. Oikonomidis, A. Argyros, and T.-K. Kim. Depth-based 3D hand pose estimation: From current achievements to future goals. In *CVPR*, 2018. 1, 7, 8, 21

[243] S. Yuan, Q. Ye, G. Garcia-Hernando, and T.-K. Kim. The 2017 hands in the million challenge on 3d hand pose estimation. *arXiv preprint arXiv:1707.02237*, 2017. 113

[244] S. Yuan, Q. Ye, B. Stenger, S. Jain, and T.-K. Kim. Big hand 2.2M benchmark: hand pose data set and state of the art analysis. In *CVPR*, 2017. x, xi, xiii, 2, 3, 7, 8, 15, 21, 25, 26, 39, 41, 42, 46, 47, 52, 53, 54, 59, 60, 111

[245] A. R. Zamir, A. Sax, W. Shen, L. Guibas, J. Malik, and S. Savarese. Taskonomy: Disetangling task transfer learning. In *CVPR*, 2018. 7, 108

[246] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose: an efficienct 3D kinematics descriptor for low-latency action recognition and detection. In *ICCV*, 2013. 30

[247] M. Zanfir, M. Leordeanu, and C. Sminchisescu. The moving pose: an efficient 3D kinematics descriptor for low-latency action recognition and detection. In *ICCV*, 2013. 30, 93

[248] J. Zhang, J. Jiao, M. Chen, L. Qu, X. Xu, and Q. Yang. A hand pose tracking benchmark from stereo matching. In *ICIP*, 2017. xviii, 26, 79, 111, 118, 119

[249] S.-H. Zhang, R. Li, X. Dong, P. L. Rosin, Z. Cai, H. Xi, D. Yang, H.-Z. Huang, and S.-M. Hu. Pose2Seg: detection free human instance segmentation. In *CVPR*, 2019. 6

[250] X. Zhang, Y. Wang, M. Gou, M. Sznaier, and O. Camps. Efficient temporal sequence comparison and classification using gram matrix embeddings on a riemannian manifold. In *CVPR*, 2016. 31, 89

[251] S. Zhou, H. Fu, L. Liu, D. Cohen-Or, and X. Han. Parametric reshaping of human bodies in images. *TOG*, 2010. 16

[252] X. Zhou, Q. Wan, W. Zhang, X. Xue, and Y. Wei. Model-based deep hand pose estimation. In *IJCAI*, 2016. 21

[253] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *ICCV*, 2017. 49, 50

[254] X. Zhu, X. Jia, and K.-Y. K. Wong. Pixel-level hand detection with shape-aware structured forests. In *ACCV*, 2014. 20

[255] Y. Zhu, W. Chen, and G. Guo. Fusing spatiotemporal features and joints for 3D action recognition. In *CVPR Workshop*, 2013. 32, 93, 101, 102

[256] C. Zimmermann and T. Brox. Learning to estimate 3D hand pose from single RGB images. In *ICCV*, 2017. xviii, 22, 23, 26, 66, 69, 73, 74, 76, 79, 80, 81, 83, 107, 111, 118, 119