A ROBUST KINEMATIC LAYER FOR HAND POSE ESTIMATION

SHICHENG CHEN

A THESIS SUBMITTED FOR THE DEGREE OF MASTER

DEPARTMENT OF COMPUTER SCIENCE
NATIONAL UNIVERSITY OF SINGAPORE

2021

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Shicheng Chen
Oct 2021

Acknowledgments

I want to first thank my supervisor, Professor Angela Yao, whose expertise, enthusiasm, and patience were invaluable to me for formulating the research questions and methodology.

I would like to thank Linlin Yang for guiding me through the various phases of our research. He has helped me to organize my research and focus on solving novel and critical issues.

Thanks to Abhinav Rai, Yuanzhi Yue, and Huan Jiang for polishing the paper.

Abstract

Estimating 3D hand pose from monocular RGB is complex because of the depth ambiguity. State-of-the-art results can be achieved by training deep networks with 3D hand pose ground truth. However, labeling 3D hand poses on a multi-view camera system is highly laborious. Obtaining 2D hand poses is easier than annotating 3D keypoints. However, 3D hand poses estimated by neural networks with 2D supervision usually do not follow biomechanical constraints due to losing depth information. We propose a differentiable kinematic layer with biomechanical rectifications to fix the estimated 3D hand pose. The kinematic layer can convert invalid hand poses to one that follows the biomechanical constraints. We conduct extensive experiments to show that the layer can alleviate depth ambiguity by making the estimated hand poses follow the biomechanical constraints. Quantitative results show that our layer can decrease the depth error of hand poses estimated by 2D supervised neural networks by 22 percent for the hand sequence dataset.

Contents

List of Figures vii
List of Tables viii
List of Algorithms ix

1 Introduction **1**
1.1 Contributions 2

2 Related Works **4**
2.1 3D Hand Pose Estimation 4
2.1.1 Supervised learning 4
2.1.2 Weakly and Self-supervised Learning 4
2.2 Biomechanical constraints 5
2.3 Inverse kinematics 5
2.3.1 Learning-based method 5
2.3.2 Model-based methods 6
2.4 VAE 7

3 Dataset and Evaluation metrics **8**
3.1 Dataset 8
3.1.1 RHD 8
3.1.2 STB 8
3.1.3 FreiHand 9
3.2 Evaluation metrics 9

4 Hand Sequence Dataset **10**
4.1 Dataset overall 10
4.2 Fine tune annotation process 10
4.3 Iterative approach for annotation 10

5 Method **12**
5.1 Notation 12
5.1.1 Variables 12
5.1.2 Functions 15
5.2 Template 16
5.3 Greedy Kinematic Layer 16
5.3.1 Palm Registration 17

5.3.2 Finger Registration 18
5.3.3 Full Process of Greedy Registration 21
5.4 Planarization 21
5.5 Abduction and Adduction Rectification 22
5.6 Twist Rectification 23
5.7 Flexion and Extension Rectification 24
5.7.1 Registration 24
5.7.2 Rectification 24
5.8 Kinematic Layer with Biomechanical Rectification 25
5.9 Framework 26
5.9.1 Input Data 26
5.9.2 Neural Network Structure 26
5.9.3 Training Strategy 27
5.9.4 Inference 27

6 Implementation 28
6.1 Data Pre-processing 28
6.2 Data Augmentation 28
6.3 Hyper-parameters 28

7 Experiment 30
7.1 Effect of Kinematic Layer with Biomechanical Rectification 30
7.2 Effects of Templates 32
7.3 Ablation Study 32
7.4 Comparison to other kinematic layers 33

8 Conclusion 35

References 36

List of Figures

5.1 Notations. 13
5.2 Joint rotation types and rest position. 13
5.3 Our proposed four hand templates. 17
5.4 Keypoint registration. The yellow and gray lines represent template bone and target bone (estimated bone), respectively. The red line denotes that the template bone is rotated with matrix $\boldsymbol{G}$. The blue line indicates the template bone is rotated and transited by $\boldsymbol{G}$ and $\boldsymbol{t}$. 18
5.5 two coordinate systems 19
5.6 Finger Registration includes three keypoint registrations. The gray and yellow solid lines are target (estimated) finger and template finger, respectively. If our template is rigid, the red lines will represent the registered template. The blue lines denote the no-rigid template registration result. 19
5.7 Differentiate between abduction and adduction 22
5.8 Overview of our framework. Our model is pre-trained on synthetic data denoted by blue arrows and fine-tuned on a real-world dataset indicated by green arrows. 27

7.1 Qualitative results. 31

List of Tables

3.1 Overview of datasets that are used for evaluating our work. 9

5.1 Variables 12
5.2 Functions 15

6.1 We list the angle range for each joint, and the degree is the unit of angle. 29

7.1 The effect of ground-truth 2D pose and KLBR. TR and TE represent training set and test set, respectively. 31
7.2 The effect of the kinematic layer on the test dataset of HSD. Only training on RHD leads to poor accuracy on HSD (real-world dataset). Adding $2D_{HSD}$ reduces 3D error because of better 2D alignment. Adding our kinematic layer can further reduce the 3D prediction error because of better d accuracy. TR and TE are short for the training set and the test set, respectively. 32
7.3 The effect of different templates 33
7.4 The effect of KLBR components. TR and TE represent training set and test set, respectively. The P, AR, and FER denote planarization, abduction and adduction rectifications, and flexion and extension rectifications. BL is short for Baseline. 34
7.5 Effects of KLBR, VAE, and BMC loss[SIM+20] for training $3D_{RHD}$+$2D_{HSD}$ 34

List of Algorithms

5.1 Kinematic Layer with Biomechanical Rectification Overview 25

Chapter 1

Introduction

Vision-based 3D hand pose estimation is a complex and interesting problem. There are some useful applications for hand pose estimation, such as sign language recognition, human-computer interaction. However, hand pose estimation is challenging because of the dexterity and flexibility of hands. Additionally, different lighting conditions, self-occlusion, and hand object interactions make the problem harder. Even if we can estimate 2D points from the image plane, there are a large number of potential 3D points that correspond to the estimated 2D points. To make the problem solvable, we just predicted the relative 3D coordinates to the hand wrist (or metacarpophalangeal joint of the middle finger). We also assume that we know the hand scale. Even with the above assumptions, there are still lots of candidate 3D points. Estimating 3D coordinates from a single RGB image still are an ill-posed problem due to depth ambiguity.

Most recent works use deep neural networks to estimate 3D hand poses (e.g. [BKK19, CGCY18, HVT+19]). For training neural networks properly, they need a large number of 3D annotated real-world hand pose datasets. Labeling 3D keypoint for RGB images is very complex due to the complicated multi-view setup and laborious manual 2D annotations for each view ([HROL20, ZCY+19]). To avoid 2D annotating multiple views manually, we label each keypoint in one depth image in a multi-camera calibrated setup while building our RGBD hand sequence dataset (HSD). We do not need to label all key points in a single depth image. Instead, we label a key point in one of the best quality depth images among four views. The corresponding key points on other views can be computed with camera intrinsic and extrinsic parameters. However, RGB images are much easier to obtain and better quality than depth images. Additionally, depth cameras have relatively lower resolution than RGB cameras for long distances. Synthetic datasets are incorporated for training to alleviate the pressure of 3D annotation demand. Due to domain gaps, the deep learning model trained on synthetic datasets usually cannot generalize well to real-world datasets.

[BBT19, IMGK18] try to alleviate the domain gap by adding extra 2D annotated RGB images. Such 2D weak supervision is easier to obtain even for real-world RGB images compared to 3D annotation. We can use OpenPose [CHS+19] to annotate 2D hand poses initially and then fix some wrong annotation manually in a semi-automatic annotation way.

Why could additional 2D hand poses improve the accuracy of the deep learning model? Because we can align projected 3D poses to 2D hand poses. However, we lose the depth information of 3D poses during the process of 3D projection. Therefore, the deep learning model trained by 2D hand poses may generate implausible 3D hand poses. 3D annotated training data can be used for estimating plausible 3D hand poses, whereas we want to remove the requirement of 3D hand pose annotations. Our goal is to train 3D hand poses estimation models by 2D annotated hand poses of real-world datasets and 3D annotated hand poses of synthetic datasets.

1.1 Contributions

We propose a kinematic layer with biomechanical rectifications (KLBR). It can be incorporated into a well-established deep neural network to generate anatomically valid 3D hand keypoints even the training data is 2D annotated hand poses. We assume that the human hand consists of a series of biomechanical limitations. [SIM+20] models the limitations as a series of differentiable soft constraints and obtains decent results. Our KLBR captures more biomechanical constraints than theirs. Instead of extracting different constraints separately and integrating these constraints with a deep neural network, we fit a hand model to the estimated 3D key points. Our work is designed carefully to be fully differentiable while working under different situations, and we make lots of efforts to avoid numeric errors in any edge cases. We propose to combine five different interpretable constraints into one single differentiable layer, KLBR, which can be integrated into the 3D pose estimation deep learning architectures. To be noticed here, it is not a post-process step since it is differentiable.

Concretely, we discuss five constraints in our KLBR here. The first one is planarization, which is the first operation before other operations. We make all four key points in one finger onto the same plane. The second one is abduction and adduction rectifications, which ensure that fingers stay in a valid angle range. Twist, flexion, and extension rectifications are the other two components to ensure valid intervals of the fingers. The final operation is a greedy kinematic layer, which controls the valid bone lengths. Our proposed method enjoys several advantages. All hyperparameters in our layer are interpretable. We can set the hyperparameters by daily experience or statistically from a subset of annotated 3D hand poses. As for the backbone, we use the ConvNet to predict 3D hand poses in the camera coordinate system [SXW+18], which consists of 2D pixel level and depth components.

To sum up, our contribution is

1. We propose a differentiable kinematic layer with biomechanical rectifications to fix the estimated 3D hand pose.
2. The kinematic layer can convert invalid hand poses to one that follows the biomechanical constraints.

3. Quantitative results show that our layer can improve the accuracy of 3D hand poses estimated by 2D supervised deep neural networks from 10 percent to 57 percent for different real-world datasets.
4. We introduce a new real-world hand sequence dataset (HSD).

Chapter 2

Related Works

2.1 3D Hand Pose Estimation

2.1.1 Supervised learning

Most methods propose specific neural network architectures or decent training strategies. [IMGK18, MBS+18, SSPH18] use different convolutional neural networks to regress the 3D keypoints of the hand directly. However, [BKK19, BBT19] regress the rotation matrices and shape parameters of a hand model named MANO[RTB17]. [ZB17] is the first used deep neural network on regular RGB images for 3D hand pose estimation. Theirs proposed neural network includes three blocks. The first is a segmentation network, which can crop out the hand from the input RGB image. The second structure localizes key points of hand indicated by score maps. Finally, conditioned on the score maps, the last network predicts the most likely 3D hand poses. [ZB17] also introduces a very well-known synthetic rendered hand pose dataset. [YY19] proposes disentangled variational autoencoder (dVAE). The learned latent space of dVAE can disentangle camera viewpoint, hand poses, and background.

2.1.2 Weakly and Self-supervised Learning

All the methods, as mentioned earlier, require lots of labeled images. However, 3D hand pose annotations are expensive to acquire. [CGCY18] proposes a neural network, which uses the RGBD images during training to alleviate the annotation problem since it is easy to obtain by commodity cameras. They only use RGB images for 3D keypoints predictions during the testing. And besides, their neural network is aided in training by a fully annotated synthetic dataset. [WPGY19] introduces a self-supervised method to estimate 3D hand poses from multiview depth images. They pre-train the neural network with synthetic data and fine-tune the model on real-world depth images by minimizing the distance between depth maps and a hand model crafted by 41 spheres. Additionally, [WPGY19] designs a differentiable render that can help to align rendered hand model to depth maps. They also apply various priors to help generate plausible hand poses. [WPVGY20] proposes a network for hand mesh vertices estimation and transformation matrices for each joint from depth maps. The objective function is to minimize the distance between depth maps and mesh

vertices. [IMGK18] propose a 2.5D pose representation for 3D hand pose estimate from RGB images, which represents 3D keypoints by 2D pixel level and root-relative depth components. They also mentioned that their model could be improved significantly if the additional 2D pose annotations were provided.

Although adding more weakly-labeled samples for training improves the generalization, there are still depth ambiguities, or the output hand poses may not be plausible. Because RGB images do not have depth information or depth images are lousy quality or hand self-occlusion. Therefore, we propose a kinematic layer with biomechanical rectification (KLBR). The input of the layer is a 3D hand pose, and the output is an anatomical valid hand pose. We also try to make the output close to the input. Our layer is a differentiable layer so that it can be incorporated into existing deep learning architecture easily. Additionally, our layer is written by multiple Pytorch tensor operations to avoid extra time in Python for loops.

2.2 Biomechanical constraints

BMC loss[SIM+20] can be applied to the predicted 3D key points from ConvNet directly. BMC loss includes three parts. The first loss is bone length loss. Valid bone length intervals are defined for each bone. If any bones lie outside of the range, they will add penalization to the bone length loss. The second loss is palm loss, which includes palm curvature loss and angles between two neighbor palm bones. The finger angle loss is their last one. They define local coordinate systems for each finger bone. For each joint, they define a 2D intervals for flexion and abduction. They penalize any angles that lie outside the 2D intervals. The difference between their work and our work has three points. We introduce planarization operation, which can eliminate abductions or adductions for PIP and DIP joints. We have considered twist situations, but they do not add a constraint for the finger twist. The last is that we can directly produce a plausible hand pose without parameters, but they need to add loss to help network training.

[MBS+18] introduces joint angle constraints loss and palm registration loss. [WPGY19, WPVGY20] mentioned collision loss, bone length loss. [DWOG17] applied angle loss and collision loss. They are self-supervised learning. Input and output are depth images and hand poses, respectively. However, our work cannot ensure that our outputs are collision-free. We put this as our future work.

2.3 Inverse kinematics

2.3.1 Learning-based method

[ZHX+20, YLX+20] proposed an Inverse Kinematics Network (IKNet). IKNet is a neural network with seven fully connected layers and batch normalization layers. The input for the network includes hand joints, hand bone directions, and hand shape information. Rotation

matrices for each joint are the output of the IKNet. Our work has some similarities with IKNet. Input for the KLBR layer includes the hand joints, a hand template, which also includes the information of bone directions and hand shape. The most significant difference between the two works is that IKNet needs to be trained from some hand datasets. However, our KLBR layer is a plug-and-play layer, which is not needed to be trained before use.

[CGL+19] uses two types of 3D pose representations. One is 3D coordinates of key points in the world coordinate system. The other is the UV coordinates generated by 2D detectors. Two representations can be converted to each other by the camera intrinsic matrix. To balance accurate 2D pose (second representation) and valid 3D pose structure (first representation), they use two fully connected neural network layers to make decisions. Input to the network is two representations of 3D hand poses, and the output is the weight for two representations. The final results are calculated by weighted sum between two representations.

[ZLM+19, BKK19, BBT19] regress MANO shape and poses parameters directly from images. MANO is an off-the-shelf hand model, which has shape and poses parameters. Shape parameters control hand bones' length and palm shape. Poses parameters control rotation matrices for each joint. To be noticed here, each joint in MANO has 3 DOFs. However, MANO does not provide enough constraint compared with our work.

2.3.2 Model-based methods

To solve a problem of computing 3D hand poses from 2D hand poses, [POA18] formulates it as an inverse kinematics (IK) problem. They used a hand model with 26 degrees of freedom (DOF) represented by 26 parameters. The global rotation and translation of hands need 6 DOFs used by the hand palm since they view the hand palm as a rigid structure. The metacarpophalangeal (MCP) joints have two DOFs, and other finger joints obtain 1 DOF. However, they did not model the valid angle range for each joint. They apply a Levenberg-Marquardt optimizer to minimize an objective function to get transformations of the hand model for each joint. The objective function is the distance between estimated 2D hand poses, and 2D hand poses from the rendered hand model. They can calculate plausible 3D hand poses by applying the transformations to the hand model. It is an iterative model-fitting method. Therefore, it will be slow to incorporate directly into a deep learning framework compared to our method.

[XC13] estimates 3D hand poses from depth images. They use a hand model to get plausible 3D hand poses. Their hand model is similar to [POA18] except that their hand model has 27 DOFs. One more DOF is for palm arching. They also estimate initial transformations to make the objective function converge faster. [TSLP14] predicted 2D and 3D hand pose from depth images. They use a similar hand model [XC13, POA18] to estimate the hand model transformation matrices, and then 3D plausible hand poses can be calculated by these transformation matrices. However, they did not model the valid angle intervals for each joint

either.

2.4 VAE

[WPGY19] does not have a hand skeleton model, so they cannot apply kinematic constraints such as joint angle to their hand poses. As an alternative, one well-known data-driven approach can encourage the predicted key points from ConvNet to form kinematically plausible hand poses. More specifically, a variational auto-encoder(VAE) [KW13] layer can be used to replace the KLBR. Because the hand key points inherently populate in a subspace, we can train a VAE over feasible hand poses from various online public hand datasets to learn this latent space. There are several candidate datasets such as RHD[ZB17], STB[ZJC+16], HSD, Freihand[ZCY+19], GHD[TWX+18]. The hand poses from these datasets can be VAE training samples, and it is unsupervised learning. More learning and inference details are shown in the paper [KW13].

Chapter 3

Dataset and Evaluation metrics

Table 3.1 gives an overview of the datasets that we used. Each dataset provides both camera's intrinsic and 3D key points. Therefore, we can easily get 2D hand poses from the 3D poses. Fig 5.1 shows that each sample is annotated with 21 key points skeleton model. Each finger has four key points, from the points located on a finger's base to the finger's tip. In addition, a key point is placed at the hand wrist.

3.1 Dataset

3.1.1 RHD

The rendered hand pose dataset (RHD) is a robust synthetic dataset[ZB17]. They used the software Blender [1] to render 3D models of humans to images. RHD images are saved in JPEG format with different quality factors. The RHD is created with twenty different characters and thirty-nine actions. There are 16 characters and 31 actions in the training set. The other 4 characters and 8 actions are in the test set. The camera position is randomly located for each frame around any one of the hands—the distance between the camera and hands is around 40cm to 65 cm. The background images are scenes of landscapes and cities and are downloaded from Flickr [2].

3.1.2 STB

STB includes twenty sequences of one person's left hand with 3D poses annotation. There are six different backgrounds for the dataset. We follow [CGCY18] and modify the palm joint position to the wrist position [ZB17] in order to make STB consistent with other datasets. Here, we use ten sequences for training and two sequences for testing. [ZJC+16] uses a hand pose tracking system to label the dataset. Therefore, some of the key points are on the edge of the hand since edges have more features for tracking.

[1] https://www.blender.org/

[2] https://www.flickr.com/

Table 3.1: Overview of datasets that are used for evaluating our work.

Name	type	# joints	# train/test	resolution
RHD	syn	21	42k / 2.7k	320×320
STB	real	21	15k / 3k	480×640
FH	real	21	33k / 4k	224×224
HSD	real	21	40k / 40k	480×640

3.1.3 FreiHand

Freihand is an RGB dataset that contains 32 different persons in a multi-view setup with 3D pose annotation. [ZCY+19] recorded the training set against green screens, whereas the background of the test set includes several outdoor and indoor environments. [ZCY+19] annotated 2D key points from different views and used the calibrated cameras to lift the annotation to 3D key points. However, in such a way, the 3D key points are on the surface of hands.

The test set of Freihand is only available by an online judge system with limited submission numbers. Hence, we divide the training set of Freihand into a new validation and training split.

3.2 Evaluation metrics

We use a common metric mean end-point error (EPE) to measure the average Euclidean distance between two sets of points. 3D points have two forms. The first one is that 3D points xyz are in the world coordinates system, and the other is that 3D points uvd are in the camera coordinates system. If we have camera parameters, we can transfer between two forms of 3D points easily.

We use EPE to measure the average distance separately for two sets of key points in 3D coordinates xyz, 2D pixel-level uv, or 1D depth component d. EPE for the d component means the average value of the d between two set points. We provide 2D key points for training but no d component. We want to check how the d and uv component errors change individually using the kinematic layer.

Chapter 4

Hand Sequence Dataset

4.1 Dataset overall

Here, we introduce a new real-world hand sequence dataset (HSD) to evaluate the effectiveness of our proposed differentiable layer. The dataset consists of four sequences. Each video is annotated in a semi-automated, similar to [ZCY+19]. In contrast to Freihand, we annotate the 3D coordinate by calibrated RGBD cameras [1]. Each sequence includes 20K frames and is performed by a single actor. The actors are from different countries with different hand sizes. Two sequences are used for the training set, and others are for the test set.

4.2 Fine tune annotation process

We decide to use depth images to calculate the 3D world coordinates among four different cameras since the system error is acceptable. Additionally, we do not want to label the key points on the surface of hands. Instead, we want the key points can be penetrated the skin since key points should represent hand joints instead of hand skin.

To get more accurate 3D key points, we need to measure the thickness of each finger and hand wrist to know the penetration distance and calculate the hand surface's normal vector direction. We can only label the key points on the surface of the hand. By the information of penetration distance and the normal vector direction of the surface, we can estimate the position of hand joints.

4.3 Iterative approach for annotation

We used a semi-supervised annotation method inspired by a few previous works [WPVGY20, WPGY19, ZCY+19]. Firstly, we only use synthetic depth maps to warm up the Dual Grid Net [WPVGY20], and then train the model by both unlabelled multiview depth images and manually labeled depth images iteratively. As for multiview manually labeling, we use the 3D key points that are initialized by the prediction from Dual Grid Net, and it is easy to

[1] Intel RealSense D415

project the 3D key points to each view's image coordinates by camera parameters. After obtaining the 2D key points in each view's images, we refine these 2D labels manually in any view, and the 3D key points and the 2D key points in other views will be updated accordingly. Finally, we feed these manually labeled depth images to the Dual Grid Net.

In each iteration, we visualize the prediction of the Dual Grid Net and refine the worst 50 to 100 3D key points based on the model loss. Then, we add one-quarter of the manually refined data into the test set for further evaluate the model. Before the model gets the desired performance, we continue to train and test the model iteratively. We will stop our iteration until the mean EPE of the model is less or equal to 5 millimeters in the test data since the 5 mm EPE almost hits the upper bound of the system error.

Chapter 5

Method

Our goal is to refine the noisy 3D hand poses to feasible poses with a given hand template. In this case, we propose a kinematic layer with biomechanical rectification(KLBR). We will introduce notation in section 5.1. Then, we introduce hand templates in section 5.2. We consider the bone length constraint and introduce a greedy kinematic layer in section 5.3. Next, we consider the angle constraint and introduce planarization in section 5.4, abduction, adduction, twist, flexion, extension in sections 5.5, 5.6, and 5.7. We will discuss how to combine these all constraints together in section 5.8. Specifically, we embed our proposed KLBR into a weakly supervised learning framework, which is introduced in section 5.9.

5.1 Notation

5.1.1 Variables

In this subsection, we introduce our definition used in the thesis. We use the common **21 keypoint right-hand skeleton** as default. Since using relative 3D hand coordinates, for the left hand, we can reflect all left-hand coordinates over the x-axis to convert them to right-hand coordinates. Our right-hand skeleton is shown in Fig. 5.1.

The **key points or joints** $\boldsymbol{J} \in R^{21\times 3}$ are represented by $\boldsymbol{j}_i^k$. Here, $i \in \{1, ..., 4\}$ for metacarpophalangeal joint (MCP), proximal interphalangeal joint (PIP), distal interphalangeal joint (DIP) and TIP and $k \in \{0, ..., 4\}$ for **little, ring, middle, index and thumb finger** respectively. The wrist joint are represented by $\{\boldsymbol{j}_0^k\}^{k\in\{0...4\}}$ for simplicity purposes. But

Table 5.1: Variables

key point: $\boldsymbol{j}_i^k$	bone: $\boldsymbol{b}_i^k$	Rotation: $\boldsymbol{R}_i^k$	transition: $\boldsymbol{t}_i^k$
key points: $\boldsymbol{J}$	finger: $\boldsymbol{F}^k$	Palm: $\boldsymbol{M}$	normal vector: $\boldsymbol{n}^k$
palm normal: $\boldsymbol{e}^k$	standard right direction: $\boldsymbol{r}^k$		

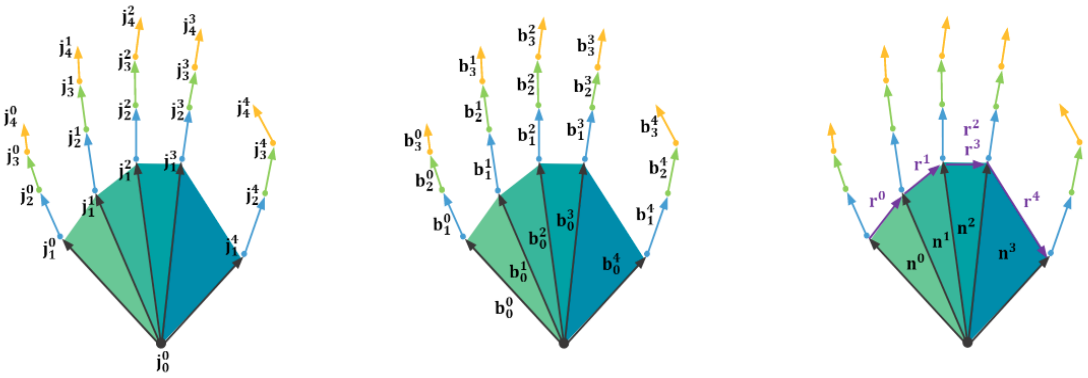
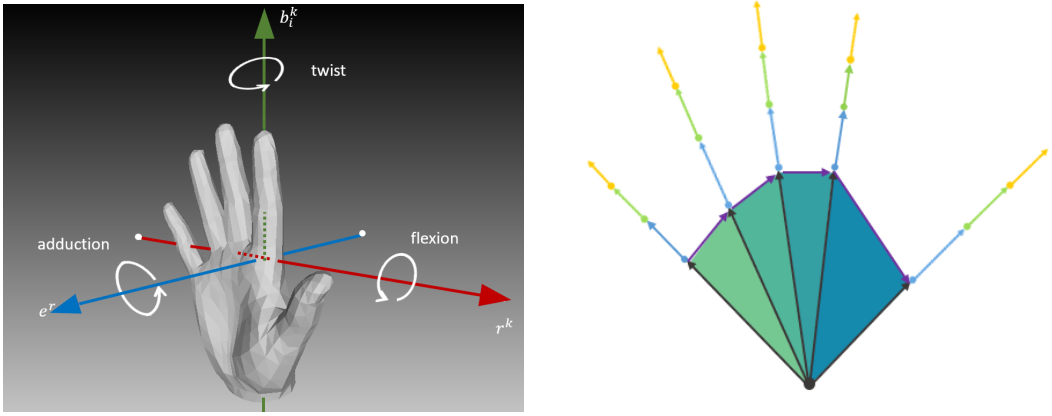


Figure 5.1: Notations.



(a) Three types of rotations

(b) Rest position.

Figure 5.2: Joint rotation types and rest position.

note that $\{\boldsymbol{j}_0^k\}^{k\in\{0...4\}}$ are the same joint, *i.e.* , $\boldsymbol{j}_0^0 = ... = \boldsymbol{j}_0^4$. Based on above definition, we can find the parent keypoint of one keypoint easily. As shown in Fig 5.1, we can denote a hand as a tree by viewing bones as edges and joints as nodes. The wrist is the root of the tree. Each key point in the tree has a father key point except the root. The **father of a joint** $\boldsymbol{j}_i^k$ is denoted by $\boldsymbol{j}_{i-1}^k$, where $i >= 1$.

The **bone** is defined by a vector $\boldsymbol{b}_i^k = \boldsymbol{j}_{i+1}^k - \boldsymbol{j}_i^k$, pointing from the father joints $\boldsymbol{j}_i^k$ to their child joints $\boldsymbol{j}_{i+1}^k$. We name the bones by the child joint. For example, one of the MCP bones is a vector pointing from the wrist (root) to MCP. More specifically, MCP bones includes $\boldsymbol{b}_0^k$ where $k \in \{0, ..., 4\}$.

We further define the **Normal vectors** $\boldsymbol{n}^k$ for $k \in \{0, 1, 2, 3\}$ in a palm by bones vectors $\boldsymbol{b}_i^k$. Specifically, $\boldsymbol{n}^k = \boldsymbol{b}_0^k \times \boldsymbol{b}_0^{k+1}$. Based on the vectors [SIM+20], we define four plane with five palm bones as shown in 5.1. We denote $\boldsymbol{e}^k$ as the palm normal. Specifically,

$$\boldsymbol{e}^k = \begin{cases} unit(\boldsymbol{n}^0 + \boldsymbol{n}^1) & \text{if k=0 or 1} \\ unit(\boldsymbol{n}^1 + \boldsymbol{n}^2) & \text{if k=2} \\ \boldsymbol{n}^2 & \text{if k=3} \\ \boldsymbol{n}^3 & \text{if k=4} \end{cases} \tag{5.1}$$

To measure the angle of twist for each finger, we also define a reference direction named **standard right direction** as shown in Fig. 5.1. For each finger, the standard right direction $\boldsymbol{r}$ is from its MCP to its neighbor MCP as below,

$$\boldsymbol{r}^k = \begin{cases} \boldsymbol{j}_1^1 - \boldsymbol{j}_1^0 \text{ if } k = 0 \\ \boldsymbol{j}_1^2 - \boldsymbol{j}_1^1 \text{ if } k = 1 \\ \boldsymbol{j}_1^3 - \boldsymbol{j}_1^2 \text{ if } k = 2, 3 \\ \boldsymbol{j}_1^4 - \boldsymbol{j}_1^3 \text{ if } k = 4. \end{cases} \tag{5.2}$$

Fig 5.2a shows how we define flexion, twist, and abduction by vectors r^k, b_i^k, and e^k. Please notice here, three vectors r^k, b_i^k, and e^k may not be perpendicular to each other. White arrows denote clockwise rotations.

For the sake of simplicity, we also define **finger matrix** $\boldsymbol{F}^k \in R^{4\times3}$,$\boldsymbol{F}^k = \{\boldsymbol{j}_i^k\}_{i\in\{1,...,4\}}$. For example, $\boldsymbol{F}^1$ is the ring finger. Similarly, we define a **palm** $\boldsymbol{M} \in R^{5\times3}$ by wrist and four MCP joints, *i.e.* , $\boldsymbol{M} = \{\boldsymbol{j}_0^0, \boldsymbol{j}_1^0, \boldsymbol{j}_1^1, \boldsymbol{j}_1^2, \boldsymbol{j}_1^3\}$ directly.

Based on the above definition, we define the **rest pose** as shown in Fig 5.2b. If five joint $\{\boldsymbol{j}_i^k\}_{i\in0..4}$ are in the same line for all k, the corresponding hand pose is a **rest pose**.

We show all these notations in Tab. 5.1. Each bone $\boldsymbol{b}_i^k$ has a **rotation matrix** $\boldsymbol{R}_i^k$ and a

Table 5.2: Functions

projection: $Proj_{\boldsymbol{a},\boldsymbol{o}}(\boldsymbol{d})$	unit: $unit(\boldsymbol{x})$	cosine: $\cos(\boldsymbol{v}_0, \boldsymbol{v}_1)$
maximum: $max(\boldsymbol{v}_0, \boldsymbol{v}_1)$	mean: $mean(\boldsymbol{X}, dim)$	determinant: $det(\boldsymbol{V})$
minimum: $min(\boldsymbol{v}_0, \boldsymbol{v}_1)$	SVD: $SVD(\boldsymbol{V})$	absolute value: $abs(v)$
arccos: $\arccos(\boldsymbol{v}_0, \boldsymbol{v}_1)$	rotate: $rot_{\boldsymbol{v}}(\boldsymbol{F}_i, \theta)$	cross product: $\times$

transition vector $\boldsymbol{t}_i^k$ where $\boldsymbol{t}_i^k$ is worked for no-rigid templates. However, we have a palm concept so that $\boldsymbol{R}_i^k, \boldsymbol{t}_i^k, k \in \{0, 1, 2, 3\}$ have the same value since we view the palm as a rigid body. Later, we will release this concept in section 5.2.

5.1.2 Functions

In this subsection, we introduce some commonly used functions in our paper. We define a projection operator,

$$Proj_{\boldsymbol{a},\boldsymbol{o}}(\boldsymbol{d}) = \boldsymbol{a} - (\boldsymbol{a} \cdot \boldsymbol{d} - \boldsymbol{a} \cdot \boldsymbol{o})\boldsymbol{d} \tag{5.3}$$

which can project 3D point $\boldsymbol{d}$ onto a plane. The plane is represented by the plane normal vector $\boldsymbol{a}$ and a vector on the plane $\boldsymbol{o}$. $Proj_{\boldsymbol{a},\boldsymbol{o}}(\boldsymbol{F})$ can project multiple points onto the plane.

We define the normalization operator for a vector as

$$unit(\boldsymbol{x}) = \frac{\boldsymbol{x}}{||\boldsymbol{x}||_2} \tag{5.4}$$

The cosine value between $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ is denoted by

$$\cos(\boldsymbol{v}_0, \boldsymbol{v}_1) = unit(\boldsymbol{v}_0) \cdot unit(\boldsymbol{v}_1) \tag{5.5}$$

The angle θ between $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$ is indicated by

$$\theta = \arccos(\cos(\boldsymbol{v}_0, \boldsymbol{v}_1)) \tag{5.6}$$

Moreover, we define $rot_{\boldsymbol{v}}(\boldsymbol{F}^k, \theta)$ as rotating each vector (joints) in $\boldsymbol{F}^k$ counterclockwise for θ degree around $\boldsymbol{v}$ in the 3D space followed by [TK94]. To simplify, we also introduce numpy-like [HMvdW+20] function in our paper as following. $max(\boldsymbol{v}_0, \boldsymbol{v}_1)$ and $min(\boldsymbol{v}_0, \boldsymbol{v}_1)$ return new vectors containing the element-wise maxima and minima between two vectors $\boldsymbol{v}_0$ and $\boldsymbol{v}_1$, respectively. $mean(\boldsymbol{A}, dim)$ calculates average values of a matrix at a given dimension dim [HMvdW+20]. $det(\boldsymbol{A})$ returns determinant of $\boldsymbol{A}$. $SVD(\boldsymbol{V})$ Computes the

singular value decomposition of matrix $\boldsymbol{V}$[PGM+19]. $abs(v)$ returns the absolute value of scalar v. $\times$ is a cross product operator. We summarize the functions in Tab. 5.2.

5.2 Template

As the templates become more accurate, it will be harder to achieve. Here, we introduce four types of templates based on the access difficulty as shown in Fig 5.3. Three triangles represent hand palms (Fig. 5.3a). To craft templates of palm, we need to calculate the unit vectors and lengths for each triangle or use $\boldsymbol{M}^{temp}$ directly, which is defined in section 5.1. Thin white lines denote that we know hand bone lengths for each sample of the dataset. However, red lines indicate that we know only one set of bone lengths for a hand from the whole dataset. Thick white lines denote bones length are calculated by mean and standard deviation values from all samples from the training set.

The first template is a fixed-length template as shown in Fig 5.3a. To craft the template, we need to know hand palm $\boldsymbol{M}^{temp}$ and bone length from each bone vector $\boldsymbol{b}^{temp}$. Such information needs to be collected for each sample. The DOF number of $\boldsymbol{M}^{temp}$ is six, which includes 3D rotations and transitions. Each joint have 3 DOFs, including joint $\boldsymbol{j}_0^{4,temp}$ while aligning $\boldsymbol{j}_1^{4,temp}$ to $\boldsymbol{j}_1^4$. Please notice here, the DOF number we mentioned above is for the templates in the greedy kinematic layer in section 5.3. If we add planarization introduced in section 5.4, the DOF number for the template will nearly follow biomechanical constraints.

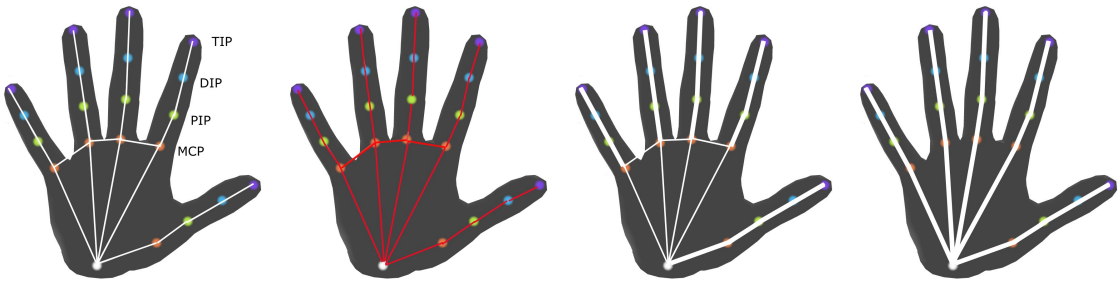
The second template is a single fix-length template (5.3b). We still need to know the palm $\boldsymbol{M}^{temp}$ and all bone lengths. The DOF for each joint is the same as the first template. However, Only one or two templates for the whole training set is enough for us.

To relax the constraint for the above two templates, we introduce an adjustable finger template as shown in Fig 5.3c. The palm $\boldsymbol{M}^{temp}$ for each sample is needed. As for finger bone length, we no longer need the exact value. We only need to calculate mean and standard deviation values for each hand bone from the whole training set to craft an adjustable finger template.

To further relax the constraint for the hand template, we finally introduce an adjustable bone template. For each bone in Fig 5.3d, we estimate a mean and standard deviation value. We no longer need to collect complete palm information such as vectors. In other words, we lose the constraint among MCP joints. Bones $\boldsymbol{b}_0^{0,temp}, ...\boldsymbol{b}_4^{0,temp}$ stemmed from joint $\boldsymbol{j}_0^{0,temp}$ have 3 DOFs while do alignment.

5.3 Greedy Kinematic Layer

To ensure the bone length feasibility of the hand pose, we introduce a greedy kinematic layer. The layer works by doing registration between the template hand and the 3D hand pose.



(a) Fixed-length template. (b) Single fix-length template. (c) Adjustable finger template. (d) Adjustable bone template.

Figure 5.3: Our proposed four hand templates.

The registered template hand is the output of the layer. Our proposed registration includes palm registration and finger registration, which consists of PIP, DIP, TIP registration.

5.3.1 Palm Registration

Here, we introduce hand palm registration. We use $\boldsymbol{M}^{temp}$ and $\boldsymbol{M}$ to denote the template palm and a palm from the estimated hand pose. Since the correspondence between palms is fixed, to align those two palms, we need to estimate 3D palm rotations $\boldsymbol{R}_0^0$ and 3D palm translation $\boldsymbol{t}_0^0$ to as follows:

$$\boldsymbol{R}_0^0, \boldsymbol{t}_0^0 = \underset{\boldsymbol{R},\boldsymbol{t}}{\operatorname{argmin}} H(\boldsymbol{R}, \boldsymbol{t}), \tag{5.7}$$

where $H(\boldsymbol{R}, \boldsymbol{t}) = ||\boldsymbol{R}\boldsymbol{M}^{temp} + \boldsymbol{t} - \boldsymbol{M}||_2^2$. We estimate $\boldsymbol{R}_0^0$ and $\boldsymbol{t}_0^0$ based on Kabsch algorithm [Kab76]. Specifically, to calculate the optimal $\boldsymbol{t}_0$, we set the derivative of H w.r.t $\boldsymbol{t}$ equal to zero as below:

$$\frac{\partial H(\boldsymbol{R}, \boldsymbol{t})}{\partial \boldsymbol{t}} = 2(\boldsymbol{R}\boldsymbol{M}^{temp} + \boldsymbol{t} - \boldsymbol{M}) = 0, \tag{5.8}$$

we have

$$\boldsymbol{t}_0^0 = \overline{\boldsymbol{M}} - \boldsymbol{R}\overline{\boldsymbol{M}}^{temp}, \tag{5.9}$$

where $\overline{\boldsymbol{M}}^{temp} = mean(\boldsymbol{M}^{temp}, 0)$ and $\overline{\boldsymbol{M}} = mean(\boldsymbol{M}, 0)$ are their corresponding centroid. To calculate the $\boldsymbol{R}_0^0$, we

$$\boldsymbol{R}_0^0 = \underset{\boldsymbol{R}}{\operatorname{argmin}} ||\boldsymbol{M}^{temp,\prime}\boldsymbol{R} - \boldsymbol{M}'||_2^2 \tag{5.10}$$

where $\boldsymbol{M}^{temp,\prime} = \boldsymbol{M}^{temp} - \overline{\boldsymbol{M}_x}^{temp}$ and $\boldsymbol{M}' = \boldsymbol{M} - \overline{\boldsymbol{M}}$. This can be solved by singular value decomposition (SVD) as below:

$$SVD(\boldsymbol{M}^{temp,\prime T}\boldsymbol{M}') = \boldsymbol{U}\Sigma\boldsymbol{V}^T \tag{5.11}$$

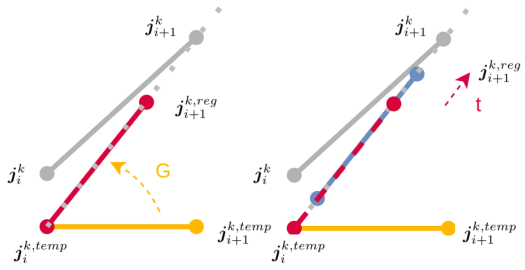


Figure 5.4: Keypoint registration. The yellow and gray lines represent template bone and target bone (estimated bone), respectively. The red line denotes that the template bone is rotated with matrix $\boldsymbol{G}$. The blue line indicates the template bone is rotated and transited by $\boldsymbol{G}$ and $\boldsymbol{t}$.

$$\boldsymbol{R}_0^0 = \boldsymbol{V} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & sign(det(\boldsymbol{V}\boldsymbol{U}^T)) \end{bmatrix} \boldsymbol{U}^T \tag{5.12}$$

Here, $sign(\cdot)$ is a function of computing sign for a scalar. For more details, we refer the reader to the paper [Kab76].

5.3.2 Finger Registration

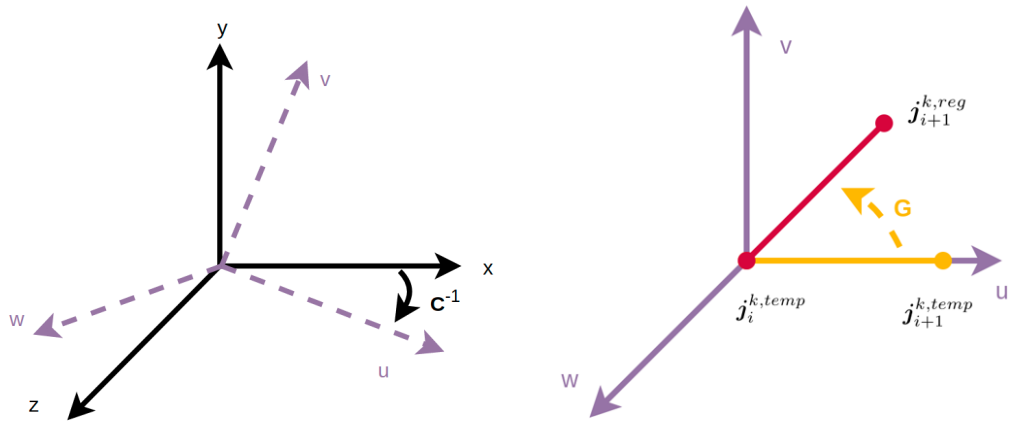
keypoint registration

Finger registrations include PIP, DIP, TIP registrations. To better understand finger registration, we first introduce keypoint (*i.e.* , PIP, DIP, TIP) registrations.

Figs 5.4 show a keypoint registration between estimated keypoint $\boldsymbol{j}_{i+1}^k$ and $\boldsymbol{j}_{i+1}^{k,temp}$ from template. Here, we assume that the keypoint $\boldsymbol{j}_{i+1}^k$ is the target, and keypoint $\boldsymbol{j}_i^{k,temp}$ is a fixed keypoint. $||b_i^k||_2$, $||b_i^{k,temp}||_2$ are given. To minimize the distance between keypoints $\boldsymbol{j}_{i+1}^{k,temp}$ and $\boldsymbol{j}_{i+1}^k$ by rotating $b_i^{k,temp}$, we use a greedy strategy that rotating bone $b_i^{k,temp}$ to the direction of $\boldsymbol{j}_{i+1}^k - \boldsymbol{j}_i^{k,temp}$. To achieve this, we first build coordinate system defined by 3 three normalized vector $[\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}]$ to put $\boldsymbol{j}_{i+1}^k - \boldsymbol{j}_i^{k,temp}$ and $\boldsymbol{b}_i^{k,temp} = \boldsymbol{j}_{i+1}^{k,temp} - \boldsymbol{j}_i^{k,temp}$ in the same plane:

$$\begin{cases} \boldsymbol{u} = \boldsymbol{u}_0 \\ \boldsymbol{v} = unit(\boldsymbol{u}_1 - (\boldsymbol{u}_0 \cdot \boldsymbol{u}_1)\boldsymbol{u}_0) \\ \boldsymbol{w} = \boldsymbol{u} \times \boldsymbol{v} \end{cases} \tag{5.13}$$

Here $\boldsymbol{u}_0 = unit(\boldsymbol{j}_{i+1}^{k,temp} - \boldsymbol{j}_i^{k,temp}), \boldsymbol{u}_1 = unit(\boldsymbol{j}_{i+1}^k - \boldsymbol{j}_i^{k,temp})$. As we can see in Figs 5.5b



(a) We can transform from the default xyz-coordinate to uvw-coordinate by the rotation matrix $\boldsymbol{C}^{-1}$.

(b) Rotation $\boldsymbol{G}$ is computed in uvw-coordinate.

Figure 5.5: two coordinate systems

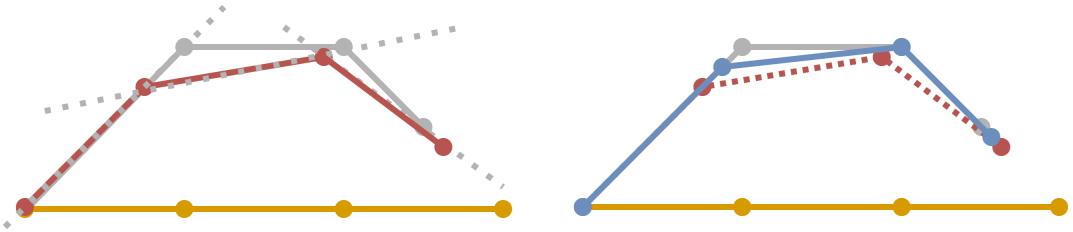


Figure 5.6: Finger Registration includes three keypoint registrations. The gray and yellow solid lines are target (estimated) finger and template finger, respectively. If our template is rigid, the red lines will represent the registered template. The blue lines denote the no-rigid template registration result.

and 5.4, $\boldsymbol{j}_i^{k,temp}$, $\boldsymbol{j}_{i+1}^{k,temp}$, $\boldsymbol{j}_{i+1}^{k,reg}$ (registered keypoint) and $\boldsymbol{j}_{i+1}^{k}$ are in the same u-v plane and we only need to rotate with θ in the uvw-coordinate. To rotate $\boldsymbol{j}_{i+1}^{k,temp}$ to $\boldsymbol{j}_{i+1}^{k,reg}$ in the uvw-coordinate, we use a rotation matrix $\boldsymbol{G}$,

$$\boldsymbol{G} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{5.14}$$

Here, $\cos(\theta) = \boldsymbol{u}_0 \cdot \boldsymbol{u}_1$ and $\sin(\theta) = \boldsymbol{u}_0 \times \boldsymbol{u}_1$. Therefore, to achieve rotation matrix $\boldsymbol{R}$ between $\boldsymbol{j}_{i+1}^{k,temp}$ and $\boldsymbol{j}_{i+1}^{k}$ in the original coordinate, we need a coordinate system rotation and a plane rotation. As such, the rotation and translation for keypoint is shown below:

$$\boldsymbol{R} = \boldsymbol{C}\boldsymbol{G}\boldsymbol{C}^{-1} \tag{5.15}$$

where $\boldsymbol{C} = [\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{w}]$. As shown in Fig 5.5a, we can use $\boldsymbol{C}^{-1}$ and $\boldsymbol{C}$ to transfer to uvw-coordinate and transfer back to default coordinate.

No translation will be needed for keypoint if the template is rigid. However, we allow finger bones to be extended or shortened for some non-rigid templates, which means that $\boldsymbol{t}$ does not need to be zero anymore. We ensure that the length of $\boldsymbol{t}$ is shorter than a predefined constant number t^c as shown in Fig 5.4. We define $t^c = 0$ for rigid templates.

$$\boldsymbol{t} = min(||\boldsymbol{j}_{i+1}^{k} - \boldsymbol{j}_{i+1}^{k,reg}||_2, t^c) unit(\boldsymbol{j}_{i+1}^{k} - \boldsymbol{j}_{i+1}^{k,reg}) \tag{5.16}$$

To make a better alignment, we use $\boldsymbol{t}$ ($||\boldsymbol{t}||_2 \leq t^c$) to move bone $\boldsymbol{j}_{i+1}^{k,reg}$ along the line $\boldsymbol{b}_i^k = \boldsymbol{j}_{i+1}^{k} - \boldsymbol{j}_i^k$.

We combine rotation and transition for each keypoint and get the following equation.

$$\begin{cases} \boldsymbol{R} = \boldsymbol{C}\boldsymbol{G}\boldsymbol{C}^{-1} \\ \boldsymbol{t} = min(||\boldsymbol{j}_{i+1}^{k} - \boldsymbol{j}_{i+1}^{k,reg}||_2, t^c) unit(\boldsymbol{j}_{i+1}^{k} - \boldsymbol{j}_{i+1}^{k,reg}) \end{cases} \tag{5.17}$$

finger registration

Here, we consider registration for each finger based on keypoint registration. To reduce the accumulated errors, we align key points (*i.e.* , PIP, DIP, TIP) in sequence greedily based on the kinematic chain of the hand.

Fig 5.6 shows one example of finger registration given a fixed hand template. The yellow line indicates the template finger, and the gray line indicates the target finger.

For the fixed bone length of the hand template, firstly, we select the parent-child keypoint

pair from root to leaf based on the kinematic chain. Then we fix the predicted parent keypoint and estimate rotation matrices based on the keypoint registration to get child keypoint prediction which nearest to the target child keypoint. We repeat this procedure until all the parent-child pairs are handled. Specifically, we get $\{\boldsymbol{R}_i^k\}_{i=1..3}^{k=0..4}$. Here, $i \in \{1, .., 3\}$ for PIP, DIP and TIP and $k \in \{0, .., 4\}$ for little, ring, middle, index and thumb finger respectively based on Eq. 5.17.

The red line indicates the predicted finger. We align PIP, DIP, and TIP bones in sequence. Although the bone length between a template and that between target is different, the greedy algorithm still can find a locally optimal solution.

To relieve the bone length constraints, we also explore finger registration with an adjustable finger template in Fig 5.6. With an adjustable finger template, we can get a better-predicted finger, as shown blue line. The only difference between the adjustable and fix-length templates is that we can change finger bone length within an interval while finger registration for the adjustable template. Specifically, we need an extra $\boldsymbol{t}$ to record the transition and set a predefined parameter t^c to limit the length of $\boldsymbol{t}$. More details of the adjustable finger template can be found in Sec. 5.2. The adjustable finger templates show more accurate registration than fix-length templates.

5.3.3 Full Process of Greedy Registration

After introducing palm registration and finger registration independently, we introduce the complete process of greedy registration in this subsection. The entire process is a greedy forward kinematics method. Specifically, we get $\boldsymbol{R}_0^0$ and $\boldsymbol{t}_0^0$ based on palm registration, $\boldsymbol{R}_i^k$ and $\boldsymbol{t}_i^k$ based on finger registration. We can calculate the k-th finger of a hand based on forward kinematics $\boldsymbol{D}_k$ by using $\boldsymbol{R}_i^k, \boldsymbol{t}_i^k$ as following:

$$\boldsymbol{D}_i^k = \prod_{j=0}^{i} \boldsymbol{T}_j^k. \tag{5.18}$$

Here $\boldsymbol{T}_i^k = \begin{bmatrix} \boldsymbol{R}_i^k & \boldsymbol{t}_i^k \\ 0 & 1 \end{bmatrix}$.

5.4 Planarization

Based on the feasibility of hand, we know that PIP and DIP joints only have one DOF, TIP does not have DOF. Therefore, we assume that four key points (*i.e.* , MCP, PIP, DIP, and TIP) of one finger are in the same plane. This is to say, for k-th finger, its four joints $\boldsymbol{F}^k = \{\boldsymbol{j}_i^k\}_{i=1..4} \in R^{4\times3}$ should be in the same plane. To solve this, we calculate the vector

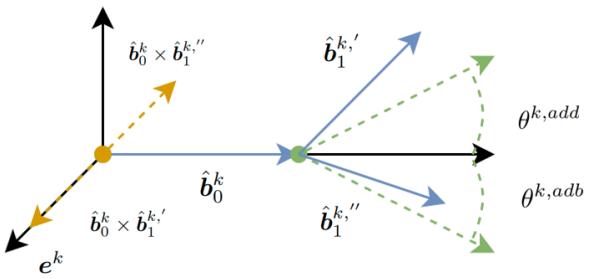


Figure 5.7: Differentiate between abduction and adduction

of the plane $\boldsymbol{f}^k \in R^{3\times1}$ and offset $\boldsymbol{o}^k \in R^{4\times1}$ as below

$$(\boldsymbol{f}^k, \boldsymbol{o}^k) = \underset{\boldsymbol{f},\boldsymbol{o}}{\operatorname{argmin}} ||\boldsymbol{F}^k\boldsymbol{f} + \boldsymbol{o}||_2^2. \tag{5.19}$$

To estimate this plane from multiple 3D points, we use SVD to solve the problem like [Kab76]. Specifically, with the centroid point $\overline{\boldsymbol{F}}^k = mean(\boldsymbol{F}^k, 0)$ in one finger. By similar method shown in 5.8, we get $\boldsymbol{o}^k = -\overline{\boldsymbol{F}}^k\boldsymbol{f}^k$. We apply SVD on $\boldsymbol{F} - \overline{\boldsymbol{F}}_i$ and get

$$SVD(\boldsymbol{F}^k - \overline{\boldsymbol{F}}^k) = \boldsymbol{U}\Sigma\boldsymbol{V}^T. \tag{5.20}$$

Based on [Kab76], we get plane vector $\boldsymbol{f}^k$, which is equal to the third column of matrix $\boldsymbol{V}$.

$$\boldsymbol{j}_i^{k,p} = Proj_{\boldsymbol{f}^k,\overline{\boldsymbol{F}}^k}(\boldsymbol{j}_i^k) \tag{5.21}$$

With $\boldsymbol{f}^k$ and $\overline{\boldsymbol{F}}^k$, we can project keypoints $\boldsymbol{j}_i^k$ onto the plane using the projection function. Especially, if the equation $abs(cos(\boldsymbol{j}_i^k, \boldsymbol{j}_{i+1}^k)) > 0.95 \ \forall \ i \in \{1, 2, 3\}$, we will consider that four keypoints for k-th finger are in the same line and not move these points.

5.5 Abduction and Adduction Rectification

This section will introduce how to limit fingers only to adduct/abduct invalid intervals. If abduction and adduction angles lie outside of the valid interval for fingers, we should correct them. We call this abduction and adduction rectification, AR for short. As shown in Fig 5.2a, Rotating the index finger around the $\boldsymbol{e}^k$ clockwise or counterclockwise represents abduction or adduction.

PIP, DIP, and TIP do not have the DOF for abduction and adduction based on the feasibility of the hand. Therefore, we only need to consider the abduction and adduction for the MCP joints. For k-th finger, we assume the valid interval is $[-\theta^{k,add}, \theta^{k,abd}]$ and know the $\boldsymbol{j}_i^{k,p}$

after planarization. Based on this, we get

$$\hat{\boldsymbol{j}}_i^k = Proj_{\boldsymbol{e}^k, \boldsymbol{j}_1^k}(\boldsymbol{j}_i^{k,p}), \tag{5.22}$$

where $\hat{\boldsymbol{j}}_i^k$ is a point on the plane $\boldsymbol{e}^k$ projected by the joint $\boldsymbol{j}_i^{k,p}$.

Based on the directions of two bone $\hat{\boldsymbol{b}}_0^k = \hat{\boldsymbol{j}}_1^k - \hat{\boldsymbol{j}}_0^k$ and $\hat{\boldsymbol{b}}_1^k = \hat{\boldsymbol{j}}_2^k - \hat{\boldsymbol{j}}_1^k$, we estimate the corrected angle of MCP as

$$\theta^k = \begin{cases} -max((\arccos(\hat{\boldsymbol{b}}_0^k, \hat{\boldsymbol{b}}_1^k)) - \theta^{k,add}, 0) & \text{if } c \geq 0. \\ max((\arccos(\hat{\boldsymbol{b}}_0^k, \hat{\boldsymbol{b}}_1^k)) - \theta^{k,abd}, 0) & \text{if } c < 0. \end{cases} \tag{5.23}$$

where $c = \cos((\hat{\boldsymbol{b}}_0^k \times \hat{\boldsymbol{b}}_1^k), \boldsymbol{e}^k)$. Figure 5.7 shows how we use the cross product to differentiate between abduction and adduction. If $\hat{\boldsymbol{b}}_0^k \times \hat{\boldsymbol{b}}_1^k$ has same direction with palm normal direction $\boldsymbol{e}^k$, *i.e.* , $c \geq 0$, the finger will be in an adduction state. Therefore, the adduction degrees can be calculated by $\arccos(\hat{\boldsymbol{b}}_0^k, \hat{\boldsymbol{b}}_1^k)$. If the abduction degrees are more than $\theta^{k,add}$ degrees, we will rotate the finger clockwise to make the finger in the valid range. If $\hat{\boldsymbol{b}}_0^k \times \hat{\boldsymbol{b}}_1^k$ and $\boldsymbol{e}^k$ are in the opposite direction, *i.e.* , $c < 0$, this finger is in an abduction state. Therefore, the abduction degrees can be calculated by $\arccos(\hat{\boldsymbol{b}}_0^k, \hat{\boldsymbol{b}}_1^k)$. If the abduction degrees are more than $\theta^{k,add}$ degrees, we will rotate the finger around $\boldsymbol{e}^k$ counterclockwise to make the finger in the valid range.

$$\boldsymbol{F}^{k,ar} = rot_{\boldsymbol{e}^k}(\boldsymbol{F}^k, \theta^k) \tag{5.24}$$

We rotate the whole finger $\boldsymbol{F}^k$ as $\boldsymbol{F}^{k,ar}$. Therefore, we get $\boldsymbol{f}^{k,ar}$, $\{\boldsymbol{b}_i^{k,ar}\}_{i=1..3}$ and $\{\boldsymbol{j}_i^{k,ar}\}_{i=1..3}$.

5.6 Twist Rectification

We will discuss rotating fingers to valid intervals $[-\theta^{k,twist}, \theta^{k,twist}]$ if they lie outside of the interval for fingers' twist. The abbreviation for twist rectification is TR. Rotating the index finger around the $\boldsymbol{b}_1^k$ clockwise or counterclockwise indicates twist, as shown in Fig 5.2a.

We only need to calculate the twist degrees of MCP joints. To measure the angle of twist for each finger, we use the standard right direction as reference directions. Based on those reference directions, we can get the corrected twist angle,

$$\theta^k = max((\arccos(\boldsymbol{r}^k, \boldsymbol{f}^{k,ar})) - \theta^{k,twist}, 0) \tag{5.25}$$

The angle between the normal vector $\boldsymbol{f}^{k,ar}$ of a finger plane and its standard right direction of the finger $\boldsymbol{r}^k$ should be less than $\theta^{k,twist}$. If it is out of range, we will correct it by rotating

the finger $\boldsymbol{F}^{k,ar}$ around $\boldsymbol{b}_1^{k,ar}$.

$$\boldsymbol{F}^{k,twr} = \begin{cases} rot_{\boldsymbol{b}_1^{k,ar}}(\boldsymbol{F}^{k,ar}, \theta^k) \text{ if } \arccos(rot_{\boldsymbol{b}_1^{k,ar}}(\boldsymbol{f}^{k,ar}), r^k) < \theta^{k,twist} \\ rot_{\boldsymbol{b}_1^{k,ar}}(\boldsymbol{F}^{k,ar}, -\theta^k) \text{ elsewise} \end{cases} \tag{5.26}$$

As per the rotation direction, we calculate results from equation 5.26 and simply select the result in the valid interval $[-\theta^{k,twist}, \theta^{k,twist}]$. Therefore, we correct the plane of finger and get $\boldsymbol{f}^{k,twr}$, $\{\boldsymbol{b}_i^{k,twr}\}_{i=1..3}$ and $\{\boldsymbol{j}_i^{k,twr}\}_{i=1..3}$.

5.7 Flexion and Extension Rectification

Planarization, abduction and adduction rectification, twist rectification introduced above three sections 5.4, 5.5, and 5.6 were applied on the estimated hand poses from a ConvNet. This section will first introduce registration 5.3 between a pre-defined hand template $\boldsymbol{j}_i^{k,temp}$ and an estimated hand pose $\boldsymbol{j}_i^{k,twr}$. Afterward, we will discuss how to apply flexion and extension rectification to the registered hand template $\boldsymbol{j}_i^{k,reg}$.

5.7.1 Registration

We first align hand palm $\boldsymbol{M}^{temp}$ to $\boldsymbol{M}^{twr}$ and get registered palm $\boldsymbol{M}^{reg}$. Then, to get $\boldsymbol{j}_i^{k,reg}$, we align keypoint from $\boldsymbol{j}_i^{k,temp}$ to $\boldsymbol{j}_i^{k,twr}$. After alignment of a finger bone, we apply flexion and extension rectification to $\boldsymbol{j}_i^{k,reg}$ and get $\boldsymbol{j}_i^{k,fer}$. We conduct the above process to each bone for $i = 2, 3, 4$ in sequence.

5.7.2 Rectification

As we get $\boldsymbol{j}_i^{k,reg}$ after registration, here, we apply flexion and extension rectification for $\boldsymbol{j}_i^{k,reg}$. For k-th finger, we dig into how to rotate each bone $\boldsymbol{b}_i^{k,reg}$ to valid intervals $[-\theta_i^{k,exten}, \theta_i^{k,flex}]$ if they are outside of the range. Rotating the index finger around the $\boldsymbol{r}^k$ clockwise or counterclockwise indicates flexion or extension in Fig 5.2a. We estimate the corrected angle for each bone.

$$\theta_i^k = \begin{cases} max((\arccos(\boldsymbol{b}_i^{k,reg}, \boldsymbol{b}_{i+1}^{k,reg})) - \theta_i^{k,flex}, 0) & \text{if } \cos(\boldsymbol{b}_i^{k,reg} \times \boldsymbol{b}_{i+1}^{k,reg}, \boldsymbol{r}^k) \geq 0 \\ max((\arccos(\boldsymbol{b}_i^{k,reg}, \boldsymbol{b}_{i+1}^{k,reg}) - \theta_i^{k,exten}, 0) & \text{if } \cos(\boldsymbol{b}_i^{k,reg} \times \boldsymbol{b}_{i+1}^{k,reg}, \boldsymbol{r}^k) < 0, \end{cases} \tag{5.27}$$

$$\boldsymbol{j}_i^{k,fer} = rot_{\boldsymbol{f}_i^k}(\boldsymbol{j}_q^k, \theta_i^k) \tag{5.28}$$

where $\boldsymbol{r}^k$ is the standard right direction in equation 5.2. If $\boldsymbol{b}_i^{k,reg} \times \boldsymbol{b}_{i+1}^{k,reg}$ has similar direction with $\boldsymbol{r}^k$, which is indicated by $\cos(\boldsymbol{b}_i^{k,reg} \times \boldsymbol{b}_{i+1}^{k,reg}, \boldsymbol{r}^k) \geq 0$, the two continuous bones of the finger will be flexed. We measure the flexion degrees by angle $\arccos(\boldsymbol{b}_i^{k,reg}, \boldsymbol{b}_{i+1}^{k,reg})$ between two bones $\boldsymbol{b}_i^{k,reg}$ and $\boldsymbol{b}_{i+1}^k$. If the flexion degrees is bigger than the limitation $\theta_i^{k,flex}$, we

Algorithm 5.1: Kinematic Layer with Biomechanical Rectification Overview

Input : Estimated Hand Pose $\boldsymbol{J}$, Hand Template $\boldsymbol{J}^{temp}$, Template Type $tt = \{0, 1, 2, 3\}$

Output : 3D Hand Keypoints following Biomechanical Constraints $\boldsymbol{J}^{fer}$

1 $\boldsymbol{J}^{p}$: fingers planarization for $\boldsymbol{J}$ based on Equation 5.21
2 $\boldsymbol{J}^{ar}$: abduction and adduction rectification for $\boldsymbol{J}^{p}$ based on Equation 5.24
3 $\boldsymbol{J}^{twr}$: twist rectification for $\boldsymbol{J}^{ar}$ based on Equation 5.26
4 **if** *the template includes palm concept (Figures 5.3a, 5.3b, and 5.3c)* **then**
5 $\quad \boldsymbol{J}^{reg}$:hand palm registration between $\boldsymbol{J}^{twr}$ and $\boldsymbol{J}^{temp}$ (Equation 5.7)
6 **end**
7 **for** $k = 0$ **to** 4 **do**
8 $\quad$ **if** *the template does not include palm concept (Fig 5.3d)* **then**
9 $\quad\quad \boldsymbol{j}_1^{k,reg}$: align keypoint between $\boldsymbol{j}_1^{k,twr}$ and $\boldsymbol{j}_1^{k,temp}$ based on Equation 5.17
10 $\quad$ **end**
11 $\quad$ **for** $i = 2$ **to** 4 **do**
12 $\quad\quad \boldsymbol{j}_i^{k,reg}$: align keypoint between $\boldsymbol{j}_i^{k,twr}$ and $\boldsymbol{j}_i^{k,temp}$ based on Equation 5.17
13 $\quad\quad \boldsymbol{j}_i^{k,fer}$: flexion and extension rectification for $\boldsymbol{j}_i^{k,reg}$ based on Equation 5.28
14 $\quad$ **end**
15 **end**

will rotate the joint and its children $\{\boldsymbol{j}_q^k\}_{q=i..4}$ around $\boldsymbol{f}_i^k$ for θ_i^k degrees by the function equation 5.28. If $\cos\left((\boldsymbol{b}_i^{k,reg} \times \boldsymbol{b}_{i+1}^{k,reg}, \boldsymbol{r}^k), \boldsymbol{r}^k\right) < 0$, the two continuous bones of the finger will be extended. we will rotate joints and its children $\{\boldsymbol{j}_q^k\}_{q=i..4}$ around $\boldsymbol{f}_i^k$ for θ_i^k degrees by equation 5.28. After flexion and extension rectification, we get our final out.

5.8 Kinematic Layer with Biomechanical Rectification

Algorithm 5.1 shows the complete algorithm combining greedy kinematic layer with four types of biomechanical rectifications (KLBR). Inputs for KLBR are a template $\boldsymbol{J}^{temp}$ and estimated joints $\boldsymbol{J}$ from ConvNet. The output of KLBR is a registered template $\boldsymbol{J}^{fer}$ that is close to the estimated joints $\boldsymbol{J}$ and following biomechanical constraints.

KLBR algorithm includes two paths. For the first path, we apply planarization, abduction, and adduction rectification, and twist rectification directly to input joints $\boldsymbol{J}$ to get joints $\boldsymbol{J}^{twr}$, as shown in the algorithm 5.1 lines from 4 to 6. For the second path, we do registration

between template $\boldsymbol{J}^{temp}$ and joints $\boldsymbol{J}^{twr}$. While aligning each bone, we apply flexion and extension rectification to the bones to follow flexion and extension constraints (lines from 12 to 13). Since each alignment step is done greedy, we assert that the final registered output joints $\boldsymbol{J}^{fer}$ are also close to the input joints $\boldsymbol{J}$.

5.9 Framework

5.9.1 Input Data

Let us discuss the input for the network. $\mathbb{X}_L = \{(\mathbf{I}_L, \mathbf{J}_L^{gt})\}$ denotes a labeled synthetic dataset, including n synthetic images $\mathbf{I}_L \in R^{n \times height \times width \times depth}$ and n 3D hand poses with twenty-one keypoints $\mathbf{J}_L^{gt} \in R^{n \times 21 \times 3}$. $\mathbb{X}_W = \{(\mathbf{I}_W, \mathbf{J}_W^{gt}\}$ indicates a weakly-labeled data, including m real-world images $\mathbf{I}_W \in R^{n \times height \times width \times depth}$ and m 2D hand poses with twenty-one keypoints $\mathbf{J}_W^{gt} \in R^{m \times 21 \times 2}$. We define 3D hand poses in a world coordinate as $\mathbf{J}_L^{gt}$ and 2D pixel-level hand poses in a camera coordinate as $\mathbf{J}_W^{gt}$.

5.9.2 Neural Network Structure

We use a ResNet-18 as the network backbone. The input of the ResNet is RGB images $\mathbf{I} \in R^{batch\ size \times 3 \times 256 \times 256}$, and the output of the ResNet image features, whose tensor shape is $[batch\ size, 512, 8, 8]$. There are three transposed convolution layers and one convolution layer. After the four layers, the output heatmap shape is $[batch\ size, joint\ number \times 64, 64, 64]$ which is the heat map representation of hand poses. With the help of an integral pose regression, we get 3D key points finally. The advantage of the integral pose regression function is that the gap between regression-based methods and heat maps is narrowed, as shown in Fig. 5.8. We first pre-train the model on a synthetic dataset,

$$\mathcal{L}_{pre-train} = \mathcal{L}_{3d}^{syn} \tag{5.29}$$

We add an L2 loss $\mathcal{L}_{3d}^{syn} = ||\mathbf{J}_L - \mathbf{J}_L^{gt}||_2^2$ between predicted 3D key points $\mathbf{J}_L$ and ground truth 3D key points $\mathbf{J}_L^{gt}$ from a synthetic dataset.

$$\mathcal{L}_{fine-tune} = \mathcal{L}_{2d}^{real} + \lambda \mathcal{L}_{KLBR} + \mathcal{L}_{3d}^{syn} \tag{5.30}$$

For real-world training data, we only have the weak labels, *i.e.* , 2D key points ground truth. Therefore, during fine-tuning, an L1 loss $\mathcal{L}_{2d}^{real} = |\mathbf{J}_W^{gt} - \mathbf{J}_W|$ is added between predicted 2D key points $\mathbf{J}_W$ and ground truth 2D key points $\mathbf{J}_W^{gt}$ from real-world dataset. Moreover, We apply a kinematic layer with biomechanical rectification to estimated 3D key points $\mathbf{J}_W$ to get new 3D key points $\mathbf{J}_W^{fer}$ following biomechanical constraints. An L2 loss $\mathcal{L}_{KLBR} = ||\mathbf{J}_W - \mathbf{J}_W^{fer}||_2^2$ is added directly between two sets of 3D key points.

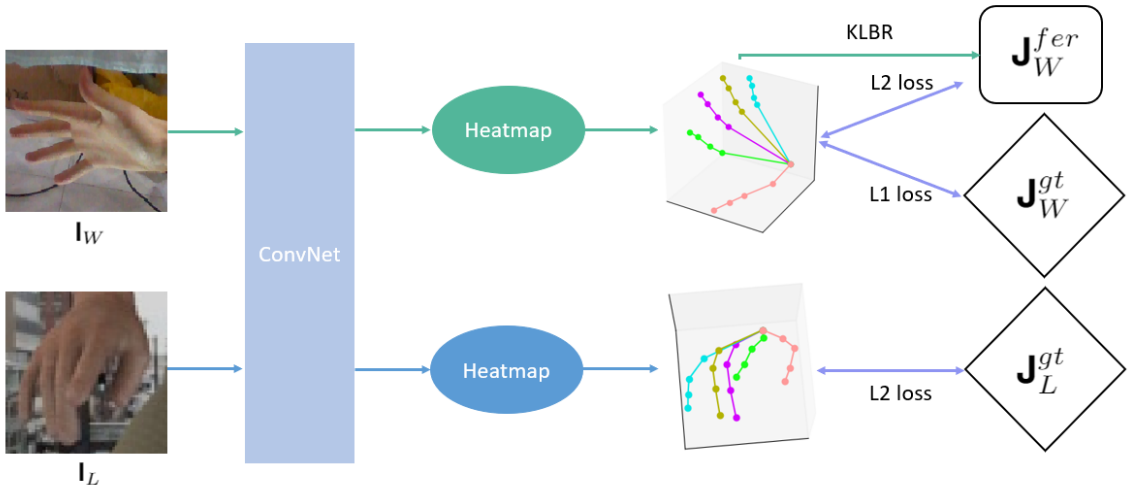


Figure 5.8: Overview of our framework. Our model is pre-trained on synthetic data denoted by blue arrows and fine-tuned on a real-world dataset indicated by green arrows.

5.9.3 Training Strategy

Firstly, we pre-train the model by equation 5.29 on the synthetic data (RHD) for 60 epochs. Blue lines in Fig 5.8 show the pre-train phase.

Then, we train the network by fully supervised synthetic data and weakly supervised real-world datasets for 80 epochs by equation 5.30. All blocks in Fig 5.8 is the fine-tune phase.

5.9.4 Inference

We use our pre-processing data method mentioned above but without data augmentation. The 3D keypoints $\mathbf{J}$ extracted from heatmaps are the normalized and transited results. By scaling back and decentralizing the key points, we finally get our results.

Chapter 6

Implementation

6.1 Data Pre-processing

We apply data pre-processing to each of the input images. Firstly, we find the bounding box of the hand based on the provided 2D ground-truth pose and crop it from the whole image like [ZB17], and then we normalize the input image by dividing 255, subtracting the mean pixel value, and dividing the standard deviation for each pixel to centralize the input images.

As for the 3D pose labels, we convert them from world coordinates to camera coordinates using camera parameters. Following [YLLY19, SSPH18], we assume the root of the hand and the reference bone length are given. Specifically, we use the MCP of the middle finger as the hand root keypoint and the phalangal proximal bone of the middle finger as the reference bone.

6.2 Data Augmentation

The deep neural networks tend to overfit due to the limited training data. Therefore, data augmentation is commonly adopted to create new training data from existing data and avoid overfitting. Like existing 3D supervised learning [IMGK18, SXW+18, MYW+20], we augment data by rotating the images randomly for any degree, rescaling the whole images randomly within the range 0.8-1.0, translating images up to 40 pixels, and changing the contrast, hue, brightness, and saturation of images. Specially, we also random rescale each channel separately [LWL21] and use random erase [ZZK+20] by selecting rectangle regions randomly and setting their pixels to zeros.

6.3 Hyper-parameters

We use PyTorch Adam optimizer with 1e-5 weight decay. For training synthetic data RHD, the batch size is 48. While training both synthetic and real-world data, we set batch size as 48 for combining RHD images and real-world images. The initial learning rate is 1e-4, and the learning rate decay is 0.1 for every thirty epochs. The total epoch number is 80. We set

Table 6.1: We list the angle range for each joint, and the degree is the unit of angle.

	MCP	PIP	DIP	Thumb MCP	Thumb PIP
Flexion	90	135	90	45	45
Extension	45	10	45	45	10
Adduction	30	4	4	30	4
Abduction	30	4	4	30	4
Twist	20	4	4	20	4

the hyper-parameters empirically, with $\lambda = 500$. The relaxation range of planarization is 2 millimeters. Table 6.1 shows the angle range for each joints.

We can find some nearly zero values in the table. They are not exactly zero due to the relaxation of planarization. Otherwise, there will be some nasty numeric errors during the training phase.

Chapter 7

Experiment

In this Chapter, we embed our kinematic layer into a weakly supervised learning framework and show the quantitative and qualitative results of our proposed kinematic layer (See Chapter 5) compared with existing kinematic layers [SIM+20, KW13]. The default setting is that we pre-train the model on labeled synthetic data (*i.e.* , RHD) and fine-tune it on weakly labeled data of a real-world dataset's training partition. The test data is withheld completely. Specifically, ground-truth 3D poses are given during pre-training, and ground-truth 2D poses are given during fine-tuning. In the following, we emphasize the effect of KLBR in Section 7.1, templates choosing in Section 7.2, ablation study 7.3 and comparing with exist kinematic layers in Section 7.4

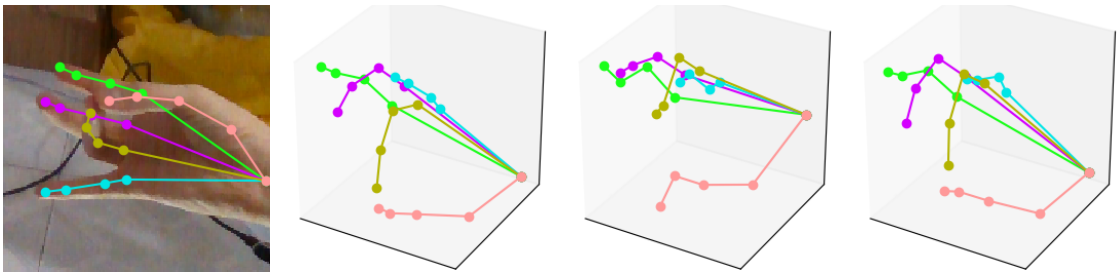
7.1 Effect of Kinematic Layer with Biomechanical Rectification

To start with, we first verify the effectiveness of our kinematic layer with biomechanical rectification (KLBR). We compare the following four settings: (1) training with full 3D supervision on RHD training set ($3D_{RHD}$), (2) training with full 3D supervision on RHD training set ($3D_{RHD}$) and the training partition of one real-world data, (3) training with full 3D supervision on RHD training set ($3D_{RHD}$) and fine-tuning on the training partition of one real-world data with weak supervision, (4) training and fine-tuning same to (3) but with our KLBR. We evaluate all the models on their corresponding training and testing partitions as shown in Tab. 7.1.

We can see the model trained on $3D_{RHD}$ achieves good performance on the RHD testing set with a mean EPE of 12.60 mm. However, when evaluating the model on the HSD testing set, it deteriorates to the mean 3D EPE 24.24 mm, which shows the significant domain gap between the synthetic RHD versus the real-world datasets and servers the lower bound in performance. If we mix and train the model on both one synthetic dataset and one real-world training set, we can lower the mean 3D EPE on their corresponding testing sets, which serves the upper bound in performance. When we introduce 2D poses as weak labels, the 3D mean EPE decreases. When adding our proposed layer, we further reduce 3D EPE.

Table 7.1: The effect of ground-truth 2D pose and KLBR. TR and TE represent training set and test set, respectively.

Training Method	STB TR	STB TE	HSD TR	HSD TE	FH TR	FH TE
$3D_{RHD}$	22.86	21.86	29.09	24.24	24.18	24.23
$3D_{RHD}$+2D	13.22	14.31	15.19	17.73	12.12	13.16
$3D_{RHD}$+2D+KLBR	7.13	9.08	9.95	14.85	10.64	11.88
$3D_{RHD}$+3D	7.21	6.52	5.73	13.16	8.31	9.50



(a) Ground-truth 2D hand pose (b) A ground-truth 3D pose. (c) 2D supervision. (d) KLBR layer.

Figure 7.1: Qualitative results.

Next, besides the 3D mean EPE, we conduct the experiments on RHD and HSD and show the 2D mean EPE and d component error separately in Tab. 7.2. We can see that when introducing an HSD training set with 2D pose as weak labels *i.e.* , $2D_{HSD}$, the 3D mean EPE decreases to 17.73 mm, and the 2D mean EPE decreases from 18.54 px to 8.94 px. However, note that the d component (Section 3.2) does not reduce much (14.73 mm to 14.03 mm). When adding our proposed layer, we further reduce 3D EPE to 14.85 mm due to the significant improvement of the d component (14.03 mm to 10.68 mm).

We also show qualitative results in Fig. 7.1. We can see that weak supervision with our proposed kinematic layer achieves more plausible and smooth hand poses 7.1d than only weak supervision 7.1c, which indicates that our proposed kinematic layer ensures the feasibility of hand. Even though the predictions from fully supervised learning and weakly supervised learning with the KLBR layer are feasible, they still have significant discrepancies. Moreover, the performance of fully supervised learning is still better than our KLBR layer with weak labels in most cases. This encourages us to explore further the connection between 2D mean EPE and d component error.

Table 7.2: The effect of the kinematic layer on the test dataset of HSD. Only training on RHD leads to poor accuracy on HSD (real-world dataset). Adding $2D_{HSD}$ reduces 3D error because of better 2D alignment. Adding our kinematic layer can further reduce the 3D prediction error because of better d accuracy. TR and TE are short for the training set and the test set, respectively.

training method	$3D_{RHD}$	$3D_{RHD}$+$2D_{HSD}$	$3D_{RHD}$+$2D_{HSD}$+KLBR
2D(px, TR)	19.63	2.99	3.06
depth(mm, TR)	17.97	14.45	8.88
3D(mm, TR)	29.09	15.19	9.95
2D(px, TE)	18.54	8.94	9.36
depth(mm, TE)	14.73	14.03	10.68
3D(mm, TE)	13.16	17.73	14.85

7.2 Effects of Templates

We discuss the effects of hand templates of our KLBR as shown in Fig 5.3.

We use our proposed weakly supervised learning framework and KLBR with four different templates (Sec. 5). The results are shown in Tab. 7.3. It is reasonable that fixed-length templates achieve the best performance since they are directly derived from their corresponding ground-truth 3D pose. However, the template of one hand should be consistent and ignore the annotation. Therefore, we also simply derive a hand template based on all the poses from one character. We denote this personalized template as single fix-length templates. The result based on single fix-length templates is slightly worse than that based on fixed-length templates. We believe this is because of the annotation error. Moreover, as we progressively release the bone length constraints and palm constraint to soft ranges for hand templates (see adjustable finger templates and adjustable bone templates), the template becomes much easier to get. However, the performance slightly deteriorates accordingly, which indicates the template-based methods achieve similar performance to the range constraints-based method. Apparently, the first type of template keeps more information. Therefore, it gives the best results. The last two types of templates are easier to get since we only need to provide range information for the hand bones.

7.3 Ablation Study

In Tab .7.4, we conduct several experiments to show the impact of each component. We finetune the pre-trained model on different real-world training sets (*i.e.* , HSD, STB, and

Table 7.3: The effect of different templates

templates(T)	fixed-length T	single fix-length T	adjustable finger T	adjustable bone T
HSD training set	9.95	11.69	10.65	11.70
HSD test set	14.85	15.96	15.72	15.96

Freihand) and show their performance on their corresponding training and validation sets. Specifically, We finetune the pre-trained model based on pose 2D weak labels without kinematic layer, only with greedy kinematic layer (with GKL), with abduction/adduction rectifications and planarization (with P and AR), with extension/flexion rectifications and planarization (with P and FER), with twist rectifications and planarization (with P and twist) and with all components (with all). The templates that we used for the ablation study are fixed-length templates, which are rigid templates with known bone lengths for each sample. We show the results in Tab. 7.4. We can see that almost every component improves the performance compared with the baseline, which means that each component (P+AR, P+FER, P+twist) is useful to make the hand poses follow biomechanical constraints. Combining all the components achieves the best performance except on the testing set of STB. We believe this is because the STB is easy and has been saturated.

We also find an interesting phenomenon in the Freihand dataset. The EPE arises when we add the greedy kinematical layer to the network trained by synthetic data and 2D supervised FH data. We can say that only bone length constraints from the greedy kinematical layer sometimes cannot provide enough information for the network to estimate 3D hand poses. After applying some biomechanical constraints to the greedy kinematical layer, we find that the EPE decreases again.

7.4 Comparison to other kinematic layers

The experiment setup is that we use weak supervision for training on HSD. We compared our proposed kinematic layer with two other common kinematic layers, *i.e.* , VAE layer and BMC layer.

We train a VAE over 4 or 5 different real-world or synthetic datasets (*i.e.* Freihand, RHD, STB, GHD[TWX+18], HSD) in advance, and its weights will not change as a kinematic layer while training our framework. The only difference between 4 and 5 is whether feed HSD to the VAE. To be noticed here, the training set for our ConvNet is also HSD. After training the VAE, we freeze the VAE parameters and replace the KLBR layer with VAE. If we use VAE 5 to replace the KLBR layer, the result will be as good as the training results of using 3D supervision for the HSD dataset. If we only use VAE 4, the result is a little better than

Table 7.4: The effect of KLBR components. TR and TE represent training set and test set, respectively. The P, AR, and FER denote planarization, abduction and adduction rectifications, and flexion and extension rectifications. BL is short for Baseline.

Ablation Study	STB TR	STB TE	HSD TR	HSD TE	FH TR	FH TE
BL_0=$3D_{RHD}$+2D	13.22	14.31	15.19	17.73	12.12	13.16
BL_1=BL_0+GKL	7.82	9.47	12.52	15.93	16.16	16.82
BL_1+P+AR	7.21	9.05	10.63	15.26	11.39	12.36
BL_1+P+FER	7.40	9.00	10.86	15.27	11.37	12.54
BL_1+P+twist	7.65	9.23	10.71	15.23	10.75	11.84
BL_1+all	7.13	9.08	9.95	14.85	10.64	11.51

Table 7.5: Effects of KLBR, VAE, and BMC loss[SIM+20] for training $3D_{RHD}$+$2D_{HSD}$

	$2D_{HSD}$(TR,3D EPE)	(TE,3D EPE)
KLBR fixed-length templates	9.95	14.85
BMC with exact bone length and palm	10.17	14.95
KLBR adjustable finger templates	10.77	15.89
BMC with soft bone length and exact palm	10.99	15.07
VAE_4 (STB,FH,GHD,RHD)	14.31	15.66

only 2D supervision of the HSD training set but worse than the other two methods.

For a fair comparison, we provide two setups for BMC loss. We provide exact palm curvature and bone length for the first setup to compare the fixed-length hand templates for our KLBR layers. We provided exact palm curvature and ranged hand bones' length for BMC loss to compare the adjustable finger hand templates for our proposed layer for the second setup. Tab. 7.5 shows that our two methods give similar results for a similar experimental setup. However, our proposed layer can generate plausible hand poses directly, and we also make the generated hand poses as close as the input pose. Instead, BMC can only apply the loss to the train networks to get plausible hand poses. Therefore, we can say that our method can be used in more ways, such as post processes.

Chapter 8

Conclusion

We propose a kinematic layer with biomechanical rectification, which considers the information on hand bone length, palm curvature, abduction and adduction, flexion and extension, twist degree of each finger joint. Our method can add biomechanical loss to the predicted hand and generate a plausible hand pose close to the original hand pose. Additionally, our kinematic layer is fully differentiable written by PyTorch. Therefore, our method helps make use of weakly supervised data.

Our method encourages our ConvNet backbone to predict anatomically correct hand pose via some novel procedures mentioned above. In the experiment chapter, we can find that our kinematic layer can improve the accuracy of the EPE prediction in several datasets in a weak supervision setup.

Now, we only consider the validation for each joint separately. In the future, we will explore the codependency among different hand joints within single finger or cross-fingers. We also need to explore how to generate or estimate suitable templates for the kinematic layer in the future. We also want to apply our method to different semi-supervised learning or self-supervised learning later.

References

[BBT19] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3d hand shape and pose from images in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10843–10852, 2019.

[BKK19] Seungryul Baek, Kwang In Kim, and Tae-Kyun Kim. Pushing the envelope for rgb-based dense 3d hand pose estimation via neural rendering. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1067–1076, 2019.

[CGCY18] Yujun Cai, Liuhao Ge, Jianfei Cai, and Junsong Yuan. Weakly-supervised 3d hand pose estimation from monocular rgb images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 666–682, 2018.

[CGL+19] Yujun Cai, Liuhao Ge, Jun Liu, Jianfei Cai, Tat-Jen Cham, Junsong Yuan, and Nadia Magnenat Thalmann. Exploiting spatial-temporal relationships for 3d pose estimation via graph convolutional networks. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2272–2281, 2019.

[CHS+19] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Openpose: realtime multi-person 2d pose estimation using part affinity fields. *IEEE transactions on pattern analysis and machine intelligence*, 43(1):172–186, 2019.

[DWOG17] Endri Dibra, Thomas Wolf, Cengiz Oztireli, and Markus Gross. How to refine 3d hand pose estimation from unlabelled depth data? In *2017 International Conference on 3D Vision (3DV)*, pages 135–144. IEEE, 2017.

[HMvdW+20] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[HROL20] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3196–3206, 2020.

[HVT+19] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11807–11816, 2019.

[IMGK18] Umar Iqbal, Pavlo Molchanov, Thomas Breuel Juergen Gall, and Jan Kautz. Hand pose estimation via latent 2.5 d heatmap regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 118–134, 2018.

[Kab76] Wolfgang Kabsch. A solution for the best rotation to relate two sets of vectors. *Acta Crystallographica Section A: Crystal Physics, Diffraction, Theoretical and General Crystallography*, 32(5):922–923, 1976.

[KW13] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[LWL21] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1954–1963, 2021.

[MBS+18] Franziska Mueller, Florian Bernard, Oleksandr Sotnychenko, Dushyant Mehta, Srinath Sridhar, Dan Casas, and Christian Theobalt. Ganerated hands for real-time 3d hand tracking from monocular rgb. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 49–59, 2018.

[MYW+20] Gyeongsik Moon, Shoou-I Yu, He Wen, Takaaki Shiratori, and Kyoung Mu Lee. Interhand2. 6m: A dataset and baseline for 3d interacting hand pose estimation from a single rgb image. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XX 16*, pages 548–564. Springer, 2020.

[PGM+19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural*

Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

[POA18] Paschalis Panteleris, Iason Oikonomidis, and Antonis Argyros. Using a single rgb frame for real time 3d hand pose estimation in the wild. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 436–445. IEEE, 2018.

[RTB17] Javier Romero, Dimitrios Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics (ToG)*, 36(6):1–17, 2017.

[SIM+20] Adrian Spurr, Umar Iqbal, Pavlo Molchanov, Otmar Hilliges, and Jan Kautz. Weakly supervised 3d hand pose estimation via biomechanical constraints. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XVII 16*, pages 211–228. Springer, 2020.

[SSPH18] Adrian Spurr, Jie Song, Seonwook Park, and Otmar Hilliges. Cross-modal deep variational hand pose estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 89–98, 2018.

[SXW+18] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 529–545, 2018.

[TK94] Camillo J Taylor and David J Kriegman. Minimization on the lie group so (3) and related manifolds. *Yale University*, 16(155):6, 1994.

[TSLP14] Jonathan Tompson, Murphy Stein, Yann Lecun, and Ken Perlin. Real-time continuous pose recovery of human hands using convolutional networks. *ACM Transactions on Graphics (ToG)*, 33(5):1–10, 2014.

[TWX+18] Hao Tang, Wei Wang, Dan Xu, Yan Yan, and Nicu Sebe. Gesturegan for hand gesture-to-gesture translation in the wild. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 774–782, 2018.

[WPGY19] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Self-supervised 3d hand pose estimation through training by fitting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10853–10862, 2019.

[WPVGY20] Chengde Wan, Thomas Probst, Luc Van Gool, and Angela Yao. Dual grid net: Hand mesh vertex regression from single depth maps. In *European Conference on Computer Vision*, pages 442–459. Springer, 2020.

[XC13] Chi Xu and Li Cheng. Efficient hand pose estimation from a single depth image. In *Proceedings of the IEEE international conference on computer vision*, pages 3456–3462, 2013.

[YLLY19] Linlin Yang, Shile Li, Dongheui Lee, and Angela Yao. Aligning latent spaces for 3d hand pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2335–2343, 2019.

[YLX+20] Lixin Yang, Jiasen Li, Wenqiang Xu, Yiqun Diao, and Cewu Lu. Bihand: Recovering hand mesh with multi-stage bisected hourglass networks. *arXiv preprint arXiv:2008.05079*, 2020.

[YY19] Linlin Yang and Angela Yao. Disentangling latent hands for image synthesis and pose estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9877–9886, 2019.

[ZB17] Christian Zimmermann and Thomas Brox. Learning to estimate 3d hand pose from single rgb images. In *Proceedings of the IEEE international conference on computer vision*, pages 4903–4911, 2017.

[ZCY+19] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. Freihand: A dataset for markerless capture of hand pose and shape from single rgb images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 813–822, 2019.

[ZHX+20] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. Monocular real-time hand shape and motion capture using multi-modal data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5346–5355, 2020.

[ZJC+16] Jiawei Zhang, Jianbo Jiao, Mingliang Chen, Liangqiong Qu, Xiaobin Xu, and Qingxiong Yang. 3d hand pose tracking and estimation using stereo matching. *arXiv preprint arXiv:1610.07214*, 2016.

[ZLM+19] Xiong Zhang, Qiang Li, Hong Mo, Wenbo Zhang, and Wen Zheng. End-to-end hand mesh recovery from a monocular rgb image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2354–2364, 2019.

[ZZK+20] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13001–13008, 2020.