RoMo: A Robust Solver for Full-body Unlabeled Optical Motion Capture ——— Supplementary Document ———

ANONYMOUS AUTHOR(S)

1 ADDITIONAL EXPERIMENTS

1.1 Robustness on Different Noise Levels

We compare RoMo to other labeling methods on datasets with different levels of noise. We synthesize data by introducing random occlusions into the Production dataset. Figure 1 shows the quantitative results. RoMo consistently outperforms other methods, showing robustness even in the presence of significant noise, such as a 25% occlusion rate. This can be attributed to two key factors: whole-body point cloud segmentation and tracklet-based labeling.

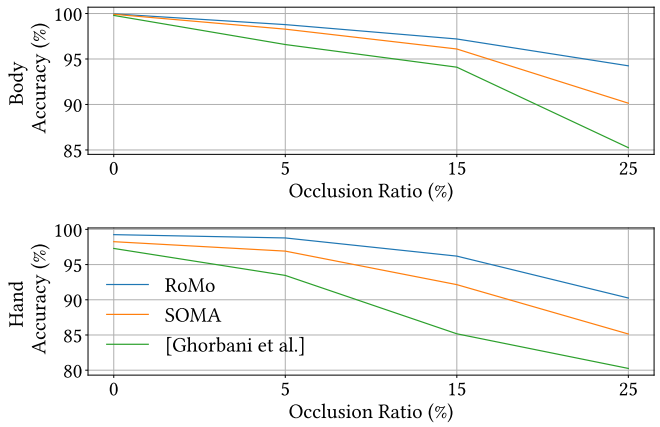

Fig. 1. Labeling accuracy on datasets with different noise scales.

1.2 Convergence Speed of Labeling Network

We present the training loss curves on the training split of Production dataset for body marker feature extraction with and without point cloud alignment on Fig. 2, noting a faster convergence speed when global transformation removal is applied.

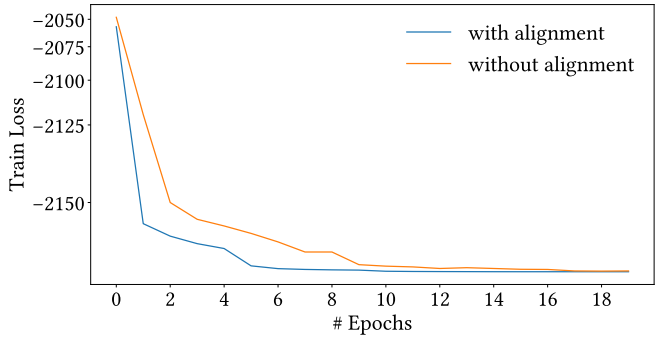

Fig. 2. Training loss curves with or without point cloud alignment.

1.3 Visualization of Attention and Feature Distance

We visualize the different layers' attentions and feature distances of different markers on the body in Fig. 4. In the first layer, the attention concentrate on one marker. In contrast, in deeper layers of the network, the attention span is wider.

1.4 Hyper-parameter Search

Hyper-parameters play an important role in the the hand marker tracklet generation pipeline, and further impacts their labeling performance. To evaluate their impact, we conduct tests on the Production dataset using various parameter values and analyzed their effects on the results. The numerical outcomes of these tests are depicted in Fig. 3. The accuracy of body marker labeling is high enough and changing the hyper-parameters only has marginal effect, thus we only set the position threshold th_{pos} to 0.3 and keep the rest hyper-parameters the same.

We also test different values of q in the confidence function $C_{L_i} = (\sum_{p_l^i \in Tr} |c_{p_l^i, L_j}|^q)^{\frac{1}{q}}$. For $q = 0$, this score mimics a simple voting mechanism, whereas for $q = 1$, it serves as the sum of the confidence towards label L_i. The quantitative results are illustrated on Table 1. We select $q = 2$ as an optimal hyper-parameter.

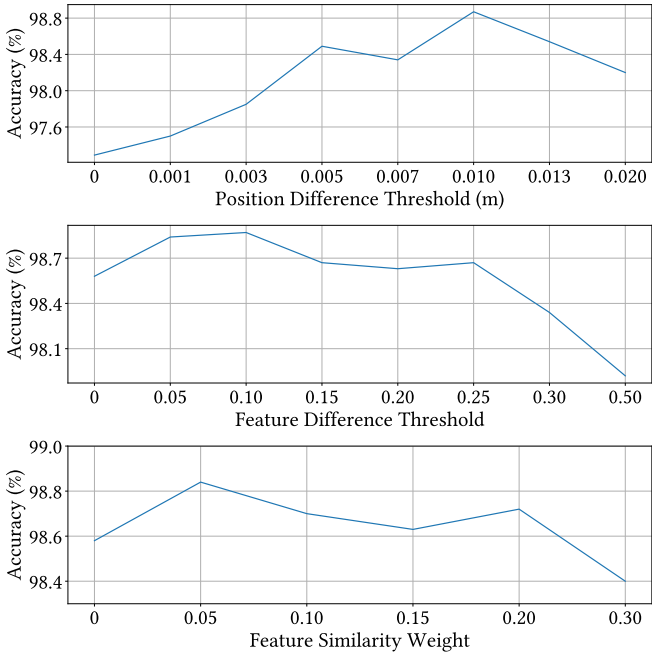

Fig. 3. Ablation study for tracklet generation. The accuracy of hand labeling of Production dataset under different tracklet generation hyper-parameters.

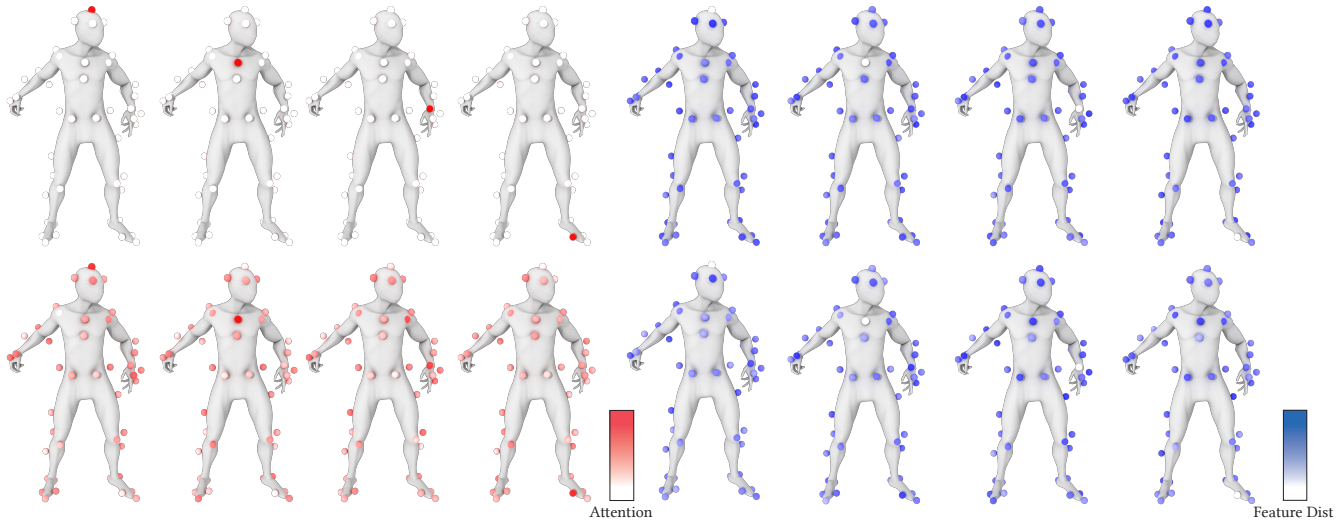

Fig. 4. The first (top row) and last (bottom row) layer's attention and feature distance of different markers on the body.

	F1		Acc.	
$q = 2$	99.94	98.62	99.96	98.87
$q = 1$	99.90	98.40	99.95	98.52
$q = 0$	99.92	97.90	99.96	98.15
No tracklet labeling	99.88	96.21	99.86	96.79

Table 1. The performance of tracklet labeling with different values of q in the confidence function of tracklet.

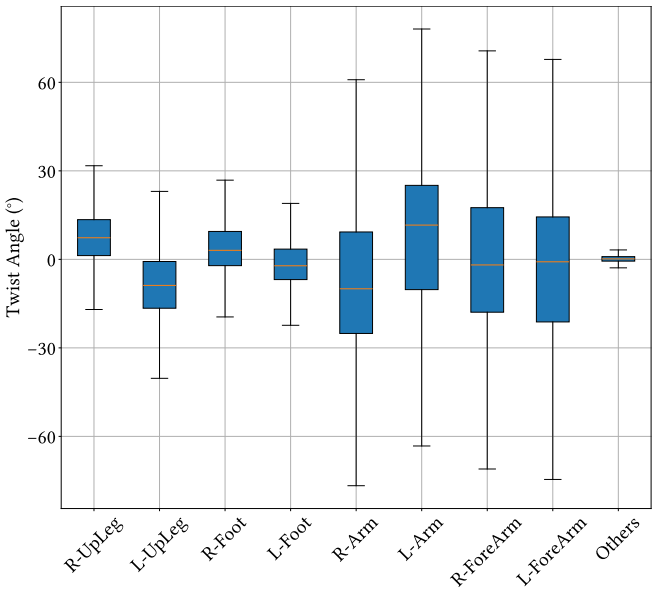

Fig. 5. Distribution of the twist angle in the Production dataset. Due to the physical limitation, only joints on limbs (8 in 27) have a wide range of variations, while other joints have a limited twist angle range.

1.5 Ablation Study of Labeling

Table 2 shows how the various RoMo components affect the Production dataset's validation split. We present quantitative results for body and hand markers separately. The most important factors influencing marker labeling performance are point cloud segmentation and tracklet-based labeling. Other components marginally improve the model's performance.

1.6 Analysis of Twist Angle

To showcase the effectiveness of the twist-and-swing decomposition, we present the distribution of joints' twist angles in Fig. 5. Only a few joints on the limbs exhibit a wide range of variations, while others have a more limited range of twist angles. This observation suggests that the decomposition approach can mitigate the challenge of network training by simplifying the estimation task. Instead of endeavoring to predict full rotation angles, the network can concentrate on estimating joint positions and twist angles, which represents a comparatively simpler problem.

1.7 Physical Accuracy

In addition to motion-based metrics, we evaluate physics-based metrics such as penetration and foot sliding. For penetration, we compute the height of the foot joints using the formula $M_{foot,z} \cdot [M_{foot,z} < 0]$, where the latter term is a mask. To calculate foot sliding, we use the formula $\dot{M}_{foot,xy} \cdot [M_{foot,z} < th_{height}]$. The combined metric is the average of the two foot joints. Table 3 shows the quantitative results. RoMo's global joint position significantly reduces foot-sliding and ground penetration artifacts.

1.8 Ablation of Inverse Kinematics Method

We compare the effectiveness of the inverse kinematics used in RoMo to the method proposed in [Holden 2018]. The network's output is set to the local rotation matrix, global position of the

	F1				Accuracy			
Base	99.94		98.62		99.96		98.87	
- Local aggregation layers	99.87	-0.07	98.59	-0.03	99.93	-0.03	98.32	-0.55
- Point cloud segmentation	99.86	-0.08	95.20	**-4.74**	99.89	-0.07	95.31	**-3.56**
- Global transformation removal	99.93	-0.01	98.10	-0.52	99.94	-0.02	98.23	-0.64
- Tracklet-based labeling	99.88	-0.06	96.21	**-2.41**	99.86	-0.10	96.79	**-2.08**
- Feature similarity in edge weights	99.95	+0.01	98.01	-0.61	99.95	-0.01	98.38	-0.49

Table 2. Ablation study of RoMo's components on the Production dataset. We take the full model as the baseline and remove one component at a time. Cells with white background display metrics for body, those with gray background represent metrics for hand.

Method	Foot Sliding	Penetration
Ground Truth	0.0659	0.00469
RoMo	**0.0714**	**0.00642**
RoMo (w/o global joint position)	0.1452	0.01649
LocalMoCap	0.1393	0.01752

Table 3. Physics-based metrics for motion solving on the Production dataset.

Method	MJMRE	MJMPE
RoMo	1.09	0.43
RoMo (with IK of [Holden 2018])	1.24	0.46
LocalMoCap	1.22	0.61

Table 4. Ablation of inverse kinematics method on the Production dataset.

joints, and bone lengths. We use Maya [Autodesk 2020] to implement inverse kinematics. Specifically, we designate the limbs' end joints as the IK handles and assign the resulting global positions to the handles. Table 4 displays quantitative results. Compared to LocalMoCap, the alternative inverse kinematics method improves global joint position accuracy but not rotation accuracy.

1.9 Performance

We evaluate the time consumption of different methods during the labeling and solving stages. The quantitative results are presented in Table 5. RoMo is marginally slower than previous neural-based methods in both stages. The additional time required for labeling is due to tracklet generation, and for solving, it stems from the hybrid inverse kinematics process. Nevertheless, the execution time complies with real-time requirements, as the tasks of labeling and solving are performed in a pipeline fashion, where RoMo simultaneous labels current frame and solves previously labeled frame.

Models	Marker Labeling	Motion Solving
[Ghorbani et al. 2019]	16 ms	-
SOMA [Mahmood et al. 2019]	19 ms	105 ms
[Holden 2018]	-	19 ms
MoCap-Solver [Chen et al. 2021]	-	26 ms
LocalMoCap [Pan et al. 2023]	-	29 ms
RoMo	25 ms	33 ms

Table 5. Time for labeling and solving of different methods.

1.10 Failure Cases

RoMo may produce significant errors when joints exhibit large twist angles, as illustrated in Fig. 6. This can be attributed to an imbalanced distribution of twist angles, with only a few training data points possessing large twist angles. Consequently, this imbalance limits network's capacity to generalize to extreme motions.

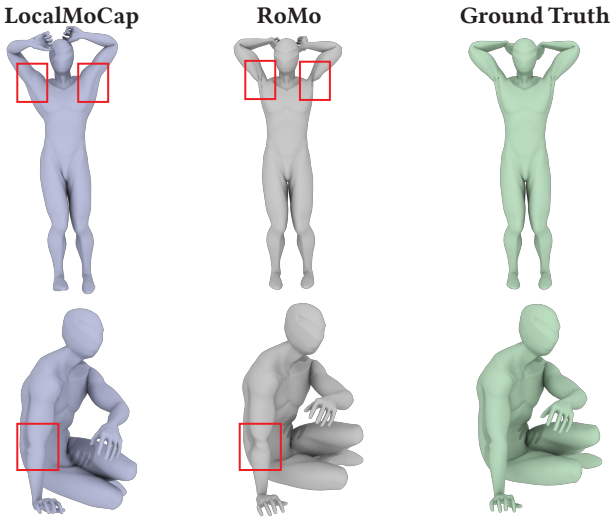

Fig. 6. RoMo may produce significant errors when joints exhibit large twist angles, such as shoulder and wrist joints.

2 IMPLEMENTATION DETAILS

2.1 Neural Network

We train two neural networks in RoMo: the marker labeling network and the motion solving network. We implement networks with PyTorch [Paszke et al. 2019] and PyTorch Lightning [Falcon et al. 2024]. We optimize their parameters with the RMSProp optimizer [Tieleman and Hinton 2012]. All the networks are trained using an NVIDIA GeForce RTX 2080Ti GPU.

For the marker labeling network, we employ both local aggregation and global attention layers. The local aggregation layer operates similar with EdgeConv [Wang et al. 2019], utilizing 4 closest neighbor points. Meanwhile, the global attention layer [Vaswani et al. 2017] adopts a self-attention scheme with 5 multi-heads. The dimensions of preceding layers are fixed at 125, with linear layers

applied before and after the input/output to adjust the feature dimensions. Following these layers, a linear layer with ReLU activation is applied. RoMo's alignment and segmentation network consist of alternatively stacked 2 layers of local aggregation and global attention layers. As for the feature extraction network, it comprises 4 layers. During training, we utilize batches of size 512 and set the learning rate to 1e-3, which decays by 70% every 30 epochs. We set the number of Sinkhorn normalization iterations to 20.

For the motion solving network, we follow the settings in [Pan et al. 2023]. The code also benefit from the hybrid inverse kinematics network implementation from [Li et al. 2021]. We set the loss weights $\lambda_{pos}, \lambda_{tw}, \lambda_{skel}$ to 1, 2 and 100, respectively. In the Production and Front-waist datasets, we observe that some joints, such as the leg and end joints of the feet, have no twist angle. Therefore, we consistently set them to zero in the network output.

2.2 Dataset

Our experiments are carried out on three different real datasets: Production, Front-waist and GRAB, whose statistics are shown in Table 6. The three datasets contain three characters with distinct marker configurations and skeleton structures. For the Production and Front-waist dataset, they are captured from the real world in a game studio, clean up by artists and solved by Vicon [Vicon 2023].

	# Body Marker	# Hand Marker	# Joint	# Anim	# Frame
Production	57	10	73	1,862	531,2743
Front-waist	53	20	73	640	234,3248
GRAB	49	36	52	1,334	162,2459

Table 6. Detailed statistics for datasets, with numbers representing the quantities of the corresponding column headers.

2.3 Comparing Methods

During the experiments, we found that some of the baseline methods' hyper-parameters and settings did not match our dataset, resulting in lower quality results on specific poses when compared to the original paper's results. We tune the baseline methods' hyper-parameters to achieve the best results for a fair comparison. We made the following changes:

- For [Ghorbani et al. 2019], we set the dimensions of network's dense layers to 125.
- For SOMA [Ghorbani and Black 2021], we set the number of self-attention layers to 6 and number of Sinkhorn normalization steps to 50.

For baseline methods of motion solving, i.e. [Chen et al. 2021; Loper et al. 2014; Pan et al. 2023], we follow the settings in the supplementary of [Pan et al. 2023].

REFERENCES

Autodesk. 2020. *Maya.*

Kang Chen, Yupan Wang, Song-Hai Zhang, Sen-Zhe Xu, Weidong Zhang, and Shi-Min Hu. 2021. MoCap-Solver: A Neural Solver for Optical Motion Capture Data. *ACM Transactions on Graphics (TOG)* 40, 4, Article 84 (2021), 11 pages.

William Falcon, Jirka Borovec, Adrian Wälchli, Nic Eggert, Justus Schock, Jeremy Jordan, Nicki Skafte, Ir1dXD, Vadim Bereznyuk, Ethan Harris, Tullie Murrell, Peter Yu, Sebastian Præsius, Travis Addair, Jacob Zhong, Dmitry Lipin, So Uchida, Shreyas Bapat, Hendrik Schröter, Boris Dayma, Alexey Karnachev, Akshay Kulkarni, Shunta Komatsu, Martin.B, Jean-Baptiste SCHIRATTI, Hadrien Mary, Donal Byrne, Cristobal Eyzaguirre, cinjon, and Anton Bakhtin. 2024. *PyTorchLightning.* https://doi.org/10.5281/zenodo.3828935

Nima Ghorbani and Michael J Black. 2021. Soma: Solving optical marker-based mocap automatically. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV).* 11117–11126.

Saeed Ghorbani, Ali Etemad, and Nikolaus F Troje. 2019. Auto-labelling of markers in optical motion capture by permutation learning. In *Advances in Computer Graphics: 36th Computer Graphics International Conference, CGI 2019.* 167–178.

Daniel Holden. 2018. Robust solving of optical motion capture data by denoising. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–12.

Jiefeng Li, Chao Xu, Zhicun Chen, Siyuan Bian, Lixin Yang, and Cewu Lu. 2021. Hybrik: A hybrid analytical-neural inverse kinematics solution for 3d human pose and shape estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR).* 3383–3393.

Matthew M. Loper, Naureen Mahmood, and Michael J. Black. 2014. MoSh: Motion and Shape Capture from Sparse Markers. *ACM Transactions on Graphics (TOG)* 33, 6 (2014), 220:1–220:13.

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *The IEEE International Conference on Computer Vision (ICCV).* 5442–5451.

Xiaoyu Pan, Bowen Zheng, Xinwei Jiang, Guanglong Xu, Xianli Gu, Jingxiang Li, Qilong Kou, He Wang, Tianjia Shao, Kun Zhou, and Xiaogang Jin. 2023. A Locality-based Neural Solver for Optical Motion Capture. *SIGGRAPH Asia 2023 Conference Papers*, Article 117 (2023), 10 pages.

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. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32 (NIPS).* 8024–8035.

T. Tieleman and G. Hinton. 2012. RmsProp: Divide the Gradient by a Running Average of its Recent Magnitude. COURSERA: Neural Networks for Machine Learning.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems (NIPS)* 30 (2017).

Vicon. 2023. *Vicon Shogun.*

Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. 2019. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)* 38, 5 (2019), 1–12.

Symbol	Size	Practical Value	Meaning
P^i	$\mathbb{R}^{n^i \times 3}$	-	The input sparse and unordered point cloud at time i.
L_i	-	-	A valid label representing a special position on the body.
$null$	-	-	A special label representing noise or outliers.
M	$\mathbb{R}^{T \times N \times 3}$	-	The ordered positions of markers span a length of T with N valid markers.
O	$\{0,1\}^{T \times N}$	-	The visibility of markers.
R	$\mathbb{R}^{T \times (3+K \times 9)}$	-	The solved motion, which spans a length of T with K joints.
S	$\mathbb{R}^{3 \times K}$	-	The solved template skeleton with K joints.
J	$\mathbb{R}^{T \times K \times 3}$	-	The estimated joint positions.
A^i	$\mathbb{R}^{4 \times 4}$	-	The estimated point cloud alignment matrix of P^i.
λ_{reg}	$\mathbb{R}^1$	0.001	The regularization weight in the loss of alignment network training.
C^i_{init}	$[0,1]^{n^i \times N}$	-	The initial confidence matrix at time i.
C^i_{aug}	$[0,1]^{(n^i+1) \times (N+1)}$	-	The augmented confidence matrix at time i.
C^i	$[0,1]^{(n^i+1) \times (N+1)}$	-	The augmented confidence matrix at time i after k Sinkhorn normalization iterations.
k	$\mathbb{N}^1$	20	The number of Sinkhorn normalization iterations.
$\mathcal{G}$	-	-	The graph for tracklet construction, consisting of $\mathcal{V}$, $\mathcal{E}$ and $\mathcal{W}$.
$\mathcal{V}$	-	-	The node set, where each node represents a point in point cloud.
$\mathcal{E}$	$\{0,1\}^{edge\_num}$	-	The edge set, linking the nodes.
$\mathcal{W}$	$\mathbb{R}^{edge\_num}$	-	The weight set, where each weight is associated with an edge.
e^{ij}_{mn}	$\{0,1\}^1$	-	The edge between the two nodes.
w^{ij}_{mn}	$\mathbb{R}^1$	-	The weight associated with edge e_{mn}, quantifying the similarity between the two nodes.
$w^{ij}_{pos,mn}$	$\mathbb{R}^1$	-	The position similarity between the two nodes.
$w^{ij}_{fet,mn}$	$\mathbb{R}^1$	-	The feature similarity between the two nodes.
λ_{fet}	$\mathbb{R}^1$	0.05	The weight of feature similarity.
th_{pos}	$\mathbb{R}^1$	0.01	The threshold of position similarity.
th_{fet}	$\mathbb{R}^1$	0.05	The threshold of feature similarity.
Tr	-	-	The set of markers belonging to a tracklet.
R_{sw}	$\mathbb{R}^{3 \times 3}$	-	The swing rotation, whose axis is perpendicular to the bone direction.
R_{tw}	$\mathbb{R}^1$	-	The twist rotation, whose axis is parallel to the bone direction.
$\vec{j}_i$	$\mathbb{R}^3$	-	The relative position of i-th joint and its child in the estimated joint positions.
$\vec{t}_i$	$\mathbb{R}^3$	-	The relative position of i-th joint and its child in the template skeleton.
$\vec{j}'_i$	$\mathbb{R}^3$	-	The refined relative position of i-th joint and its child.
J'	$\mathbb{R}^{T \times K \times 3}$	-	The joint positions of the template skeleton rotated with estimated joint rotations.
α_i	$\mathbb{R}^1$	-	The intermediate variable of Rodrigues formula, indicating the rotation angle of i-th joint.
$\vec{n}_i$	$\mathbb{R}^{3 \times 3}$	-	The intermediate variable of Rodrigues formula, indicating the rotation axis of i-th joint.
λ_{pos}	$\mathbb{R}^1$	1.0	The joint position weight in the loss of solving network training.
λ_{tw}	$\mathbb{R}^1$	100.0	The twist angle weight in the loss of solving network training.
λ_{skel}	$\mathbb{R}^1$	2.0	The skeleton weight in the loss of solving network training.

Table 7. Table of Notations.