Supplementary Material

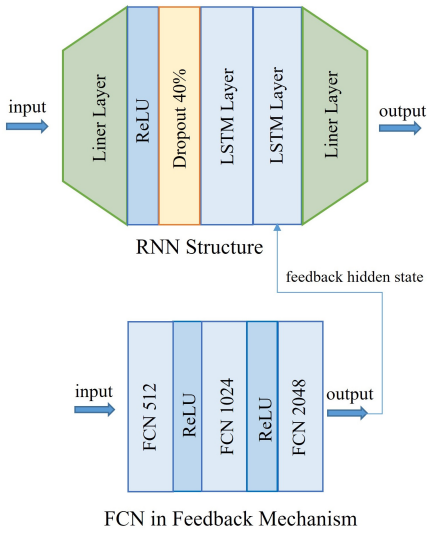

Figure 1: Detailed network structure. FCN means fully-connected network. The output of FCNs will be used as feedback for the hidden states of RNN-P1.

Figure 2: Training samples after synthetic occlusion is applied.

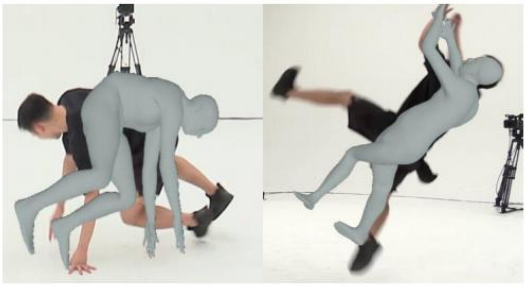

Figure 3: Examples of non-aligned ground-truth motion with the image in the AIST++ [Li et al. 2021] dataset.

A IMPLEMENTATION DETAILS

A.1 Network Sructure

Our system has multiple modules that use a similar network structure. As shown in Fig. 1, each network consists of one linear layer with ReLU activation function, two layers of LSTM [Hochreiter and Schmidhuber 1997] (with the width of 512 except RNN-P2 whose width is 1280, and RNN-T3 whose width is 1024), and one linear output layer. The LSTM does not take the future frames as its input. We use a 40% dropout. In addition, for RNN-P1, a fully-connected network (FCN) is used for the feedback mechanism, consisting of 2 linear layers with ReLU activation function and one linear output layer, as shown in Fig. 1. The output of FCNs will be used as feedback for the hidden states of RNN-P1.

A.2 Datasets

We use the AIST++ dataset [Li et al. 2021] and the AMASS dataset [Mahmood et al. 2019] for training. We perform our evaluations on TotalCapture [Trumble et al. 2017], AIST++ test split, 3DPW test split [von Marcard et al. 2018], and 3DPW-OCC [von Marcard et al. 2018; Zhang et al. 2020].

Data Synthesis. For AIST++, AMASS, 3DPW, and 3DPW-OCC without IMU input, we synthesize the IMU input using ground-truth SMPL [Loper et al. 2015] parameters. We synthesize the IMU acceleration as follows:

$$\boldsymbol{a}^{(t)} = \left(\boldsymbol{p}_{\mathrm{v}}^{(t-1)} + \boldsymbol{p}_{\mathrm{v}}^{(t+1)} - 2\boldsymbol{p}_{\mathrm{v}}^{(t)}\right)/t^2, \tag{1}$$

where $\boldsymbol{p}_{\mathrm{v}}$ denotes the synthesized IMU positions corresponding to the predefined vertices of SMPL mesh. And we use the same smooth factor of $n = 4$ as [Yi et al. 2021]. Because AMASS has no image input, we synthesize the 2D key points using ground-truth SMPL [Loper et al. 2015] parameters. For each frame, we first sample a confidence value c from the detected key points' confidence value in AIST++. Then we synthesized some random camera poses. Finally, we sample the 2D key points from:

$$\boldsymbol{x} \sim N\left(\Pi(\mathrm{FK}(\boldsymbol{\theta}^{\mathrm{GT}}, \boldsymbol{t}^{\mathrm{GT}})), \lambda_{\mathrm{conf}}(1-c)\right), \tag{2}$$

where Π denotes the projection onto $Z = 1$ plane from 3D space. We experimentally set the parameter $\lambda_{\mathrm{conf}} = 0.003$.

Data Augmentation. Moreover, for AIST++ training, we synthesized the same number of occluded images with its origin dataset. These occluded images are fed into the 2D detector to extract occluded 2D keypoints, which are then used to train RNN-P2. We demonstrate the results of synthetic occlusion in Fig. 2.

Data Preprocessing. Because the ground-truth SMPL meshes in some sequences of the AIST++ dataset are not aligned with the images, we drop these sequences. We demonstrate some examples of lousy ground-truth overlay in Fig. 3. For the TotalCapture dataset, we dropped three sequences that are not aligned at different views, and we aligned the ground-truth motions from [Huang et al.

2018] with the origin videos. Note that in both 3DPW and 3DPW-OCC datasets, the camera undergoes movement. Consequently, the ground-truth translation does not represent the subject's translation. Therefore, we only evaluate pose accuracy in these datasets. The 3DPW and 3DPW-OCC datasets operate at 30 fps; thus, we perform interpolation on the 2D keypoints.

A.3 Other Details

All training and evaluation processes run on a computer with an Intel(R) Core(TM) i7-8700 CPU and an NVIDIA GTX2080Ti graphics card. We use PyTorch 1.7.1 with CUDA 11.0 to implement our network and optimizer. We use Xsens Dot IMUs in our demo. Both training and evaluation assume 60 fps IMU input. The training data is clipped into short sequences in 200-frame lengths for more effective learning. The batch size is 256, while the learning rate is 1×10^{-3}.

B LIVE DEMO

B.1 Calibration

Because of our dual coordinate strategy, we need to align the two input modalities (image domain 2D keypoints and inertial measurements) by transforming either one to the coordinate system of the other (body root coordinate system and camera coordinate system). This process is referred to as calibration.

For the calibration method, we refer to the method in TransPose [Yi et al. 2021]. Here we list the key steps and essential formulas, omitting the detailed proof process, which can be found in TransPose. There are 5 coordinate frames: 1) the sensor coordinate frame F^S, which is the coordinate frame of a single IMU, 2) the global inertial coordinate frame F^I, which is shared by all IMUs, 3) the coordinate frame of the SMPL model body [Loper et al. 2015], which is represented as F^M, 4) the coordinate frame of the camera, which is represented as F^C, 5) the body root coordinate frame of the subject, which is represented as F^R. The first step is adjusting the orientation of the IMU sensor so that the axes of its sensor coordinate frame F^S are aligned with the corresponding axes of the camera coordinate frame F^C. From this step we can calculate $\boldsymbol{P}^{IC}$, which is the transition matrix from F^I to F^C. The second step is to adjust the orientation of the IMU sensor so that the axes of its sensor coordinate frame F^S are aligned with the corresponding axes of the SMPL coordinate frame F^M. From this step, our system can automatically calculate P^{IM}, which is the transition matrix from F^I to F^M. Then, let the user put on the IMUs to the body and keep still in predefined poses (e.g., T-Pose) for a few seconds. Predefined means we know the rotation of the bone wearing the IMU (e.g., arm) relative to F^M, which is represented as $\boldsymbol{R}_M^{\text{bone}[i]}$ and $i(0, 1, ...5)$ denotes the serial number of IMU. Using these few seconds of IMU measurements, we calculate the acceleration $\boldsymbol{a}_S^{\text{sensor}[i]}$ relative to F_S and the orientation $\boldsymbol{R}_I^{\text{sensor}[i]}$ relative to F_I. Next, for the rotation, we calculate $\boldsymbol{R}_I^{\text{offset}[i]}$, which is rotation offsets between the IMUs and corresponding bones, which is constant after the devices are set up as follows:

$$\boldsymbol{R}_I^{\text{offset}[i]} = \left(\boldsymbol{R}_I^{\text{sensor}[i]}\right)^{-1} \boldsymbol{p}^{IM} \boldsymbol{R}_M^{\text{bone}[i]}. \tag{3}$$

And we calculate $\boldsymbol{R}_C^{\text{bone}[i]}$ per frame as:

$$\boldsymbol{R}_C^{\text{bone}[i]} = \left(\boldsymbol{P}^{IC}\right)^{-1} \boldsymbol{R}_I^{\text{sensor}[i]} \boldsymbol{R}_I^{\text{offset}[i]}. \tag{4}$$

Then, we calculate the global acceleration offset as follows (after subtracting the gravity):

$$\boldsymbol{a}_C^{\text{offset}[i]} = \left(\boldsymbol{p}^{IC}\right)^{-1} \boldsymbol{R}_I^{\text{sensor}[i]} \boldsymbol{a}_S^{\text{sensor}[i]}. \tag{5}$$

Finally, we use $\boldsymbol{a}_I^{\text{bone}[i]}$ and $\boldsymbol{a}_C^{\text{offset}[i]}$ to calculate $\boldsymbol{a}_C^{\text{bone}[i]}$ per frame.

Once the aforementioned information has been computed, we can proceed to transform inputs and outputs across various coordinate systems. Utilizing the orientation of root IMU in the SMPL coordinate system $\boldsymbol{R}_M^{\text{bone}[0]}$, we can convert data between the root and the SMPL coordinate systems. And we calculate calibrated camera extrinsic matrix $\boldsymbol{P}^{CM}$ as:

$$\boldsymbol{P}^{CM} = \left(\boldsymbol{P}^{IC}\right)^{-1} \boldsymbol{P}^{IM}, \tag{6}$$

By leveraging the calibrated camera extrinsic, we can transform the data between the camera and the SMPL coordinate system.

B.2 Other Details

Our hardware configuration for recording the live demo is as follows: two laptops are used. The first one is used to run the MediaPipe[Lugaresi et al. 2019] 2D detector to extract the 2D coordinates of the keypoints of the human body in each frame from the video stream captured by the camera. At the same time, this laptop receives the measurement from the IMU via Bluetooth and then sends the 2D keypoints together with the IMU data to the second laptop by the socket. This process is running at an Intel(R) Core(TM) i7-12700H CPU. After the second laptop receives the 2D coordinates and IMU data, it runs our model to estimate the human body's joint rotations and global translation. The second laptop renders human motion onto the screen using Unity3D. Since we render the motion only by Unity3D, we do not run the reprojection optimization. This process demo runs on the laptop with an Intel(R) Core(TM) i7-10750H CPU and an NVIDIA RTX2080 Super graphics card.

REFERENCES

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J. Black, Otmar Hilliges, and Gerard Pons-Moll. 2018. Deep Inertial Poser Learning to Reconstruct Human Pose from SparseInertial Measurements in Real Time. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)* 37 (nov 2018).

Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. 2021. Learn to Dance with AIST++: Music Conditioned 3D Dance Generation. arXiv:2101.08779 [cs.CV]

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: a skinned multi-person linear model. *international conference on computer graphics and interactive techniques* (2015).

Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. 2019. MediaPipe: A Framework for Building Perception Pipelines. arXiv:1906.08172 [cs.DC]

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

Matt Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. 2017. Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors. In *2017 British Machine Vision Conference (BMVC)*.

Timo von Marcard, Roberto Henschel, Michael J. Black, Bodo Rosenhahn, and Gerard Pons-Moll. 2018. Recovering Accurate {3D} Human Pose in the Wild Using {IMUs} and a Moving Camera. *european conference on computer vision* (2018).

Xinyu Yi, Yuxiao Zhou, and Feng Xu. 2021. TransPose: Real-time 3D Human Translation and Pose Estimation with Six Inertial Sensors. *ACM Transactions on Graphics* 40 (08 2021).

T. Zhang, B. Huang, and Y. Wang. 2020. Object-Occluded Human Shape and Pose Estimation From a Single Color Image. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*