TransPose: Real-time 3D Human Translation and Pose Estimation with Six Inertial Sensors

XINYU YI, BNRist and school of software, Tsinghua University, China
YUXIAO ZHOU, BNRist and school of software, Tsinghua University, China
FENG XU, BNRist and school of software, Tsinghua University, China

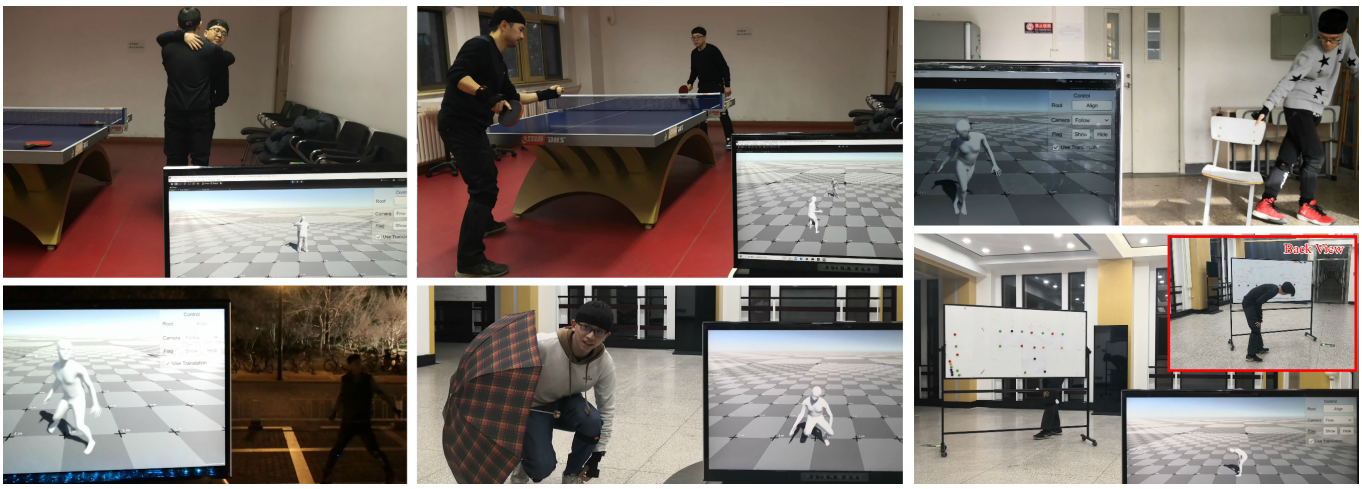

Fig. 1. We present a real-time motion capture approach that estimates poses and global translations using only six inertial measurement units. As a purely inertial sensor-based approach, our system does not suffer from occlusion, challenging environment or multi-person ambiguities, and achieves long-range capture with real-time performance.

Motion capture is facing some new possibilities brought by the inertial sensing technologies which do not suffer from occlusion or wide-range recordings as vision-based solutions do. However, as the recorded signals are sparse and quite noisy, online performance and global translation estimation turn out to be two key difficulties. In this paper, we present TransPose, a DNN-based approach to perform full motion capture (with both global translations and body poses) from only 6 Inertial Measurement Units (IMUs) at over 90 fps. For body pose estimation, we propose a multi-stage network that estimates leaf-to-full joint positions as intermediate results. This design makes the pose estimation much easier, and thus achieves both better accuracy and lower computation cost. For global translation estimation, we propose a supporting-foot-based method and an RNN-based method to robustly solve for the global translations with a confidence-based fusion technique. Quantitative and qualitative comparisons show that our method outperforms the state-of-the-art learning- and optimization-based methods with a large margin in both accuracy and efficiency. As a purely inertial sensor-based approach, our method is not limited by environmental settings (e.g., fixed cameras), making the capture free from common difficulties such as wide-range motion space and strong occlusion.

CCS Concepts: • **Computing methodologies** → **Motion capture**.

Additional Key Words and Phrases: IMU, Pose Estimation, Inverse Kinematics, Real-time, RNN

ACM Reference Format:
Xinyu Yi, Yuxiao Zhou, and Feng Xu. 2021. TransPose: Real-time 3D Human Translation and Pose Estimation with Six Inertial Sensors. *ACM Trans. Graph.* 40, 4, Article 86 (August 2021), 13 pages. https://doi.org/10.1145/3450626.3459786

1 INTRODUCTION

Human motion capture (mocap), aiming at reconstructing 3D human body movements, plays an important role in various applications such as gaming, sports, medicine, VR/AR, and movie production. So far, vision-based mocap solutions take the majority in this topic. One category requires attaching optical markers on human bodies and leverages multiple cameras to track the markers for motion capture. The marker-based systems such as Vicon[1] are widely applied and considered accurate enough for industrial usages. However, such approaches require expensive infrastructures and intrusive devices, which make them undesirable for consumer-level usages.

Authors' addresses: Xinyu Yi, BNRist and school of software, Tsinghua University, Beijing, China, yixy20@mails.tsinghua.edu.cn; Yuxiao Zhou, BNRist and school of software, Tsinghua University, Beijing, China, yuxiao.zhou@outlook.com; Feng Xu, BNRist and school of software, Tsinghua University, Beijing, China, xufeng2003@gmail.com.

0730-0301/2021/8-ART86 $15.00
https://doi.org/10.1145/3450626.3459786

[1] https://www.vicon.com/

Recently, another category focuses on pose estimation using a few RGB or RGB-D cameras [Chen et al. 2020; Habibie et al. 2019; Mehta et al. 2020; Tome et al. 2018; Trumble et al. 2016; Xiang et al. 2019]. Although these methods are much more lightweight, they are sensitive to the appearance of humans since distinguishable features need to be extracted from images. In consequence, these methods usually work poorly for textureless clothes or challenging environment lighting. Furthermore, all vision-based approaches suffer from occlusion. Such a problem can sometimes be solved by setting dense cameras (which further makes the system heavy and expensive), but it is often impractical in some applications. For example, it is very difficult to arrange cameras in a common room full of furniture or objects, which may occlude the subject from any direction. Another limitation of vision-based approaches is that the performers are usually restricted in a fixed space volume. For daily activities such as large-range walking and running, carefully controlled moving cameras are required to record enough motion information, which is very hard to achieve [Xu et al. 2016]. These disadvantages are fatal flaws for many applications and thus lead to limited usability of the vision-based solutions.

In contrast to vision-based systems, motion capture using body-worn sensors is environment-independent and occlusion-unaware. It does not require complicated facility setups and lays no constraint on the range of movements. Such characteristics make it more suitable for customer-level usages. As a result, motion capture using inertial sensors is gaining more and more focus in recent years. Most related works leverage Inertial Measurement Units (IMUs) to record motion inertia, which are economical, lightweight, reliable, and commonly used in a fast-growing number of wearables such as watches, wristbands, and glasses. The commercial inertial mocap system, Xsens[2], uses 17 IMUs to estimate joint rotations. Although accurate, the dense placement of IMUs is inconvenient and intrusive, preventing performers from moving freely. On the other hand, the facility requirements for such a system are still beyond the acceptance of normal customers. The work of SIP [Marcard et al. 2017] demonstrates that it is feasible to reconstruct human motions from only 6 IMUs. However, as an optimization-based method, it needs to access the entire sequence and takes a long time to process. The following state-of-the-art work, DIP [Huang et al. 2018], achieves real-time performance with better quality by leveraging a bidirectional RNN, also using 6 IMUs. However, it still fails on challenging poses, and the frame rate of 30 fps is not sufficient to capture fast movements, which are very common in practical applications. More importantly, it only estimates body poses without global movements which are also important in many applications such as VR and AR. The IMU itself is incapable of measuring distances directly. Some previous works [Liu et al. 2011; Vlasic et al. 2007] use additional ultrasonic sensors to measure global translations, which are expensive and subject to occlusion. Other possible solutions use GPS localization, which is not accurate enough and only works in outdoor capture. As a result, super-real-time estimation of both body poses and global translations from sparse worn sensors is still an open problem.

To this end, we introduce our approach, TransPose, which estimates global *translations* and body *poses* from only 6 IMUs at unprecedented 90 fps with state-of-the-art accuracy. Performing motion capture from sparse IMUs is extremely challenging, as the problem is severely under-constrained. We propose to formulate the pose estimation task in a multi-stage manner including 3 subtasks, each of which is easier to solve, and then estimate the global translations using the fusion of two different but complementary methods. As a result, our whole pipeline achieves lower errors but better runtime performance. Specifically, for *multi-stage pose estimation*, the task of each stage is listed as follows: *1) leaf joint position estimation* estimates the positions of *5 leaf joints* from the IMU measurements; *2) joint position completion* regresses the positions of *all 23 joints* from the leaf joints and the IMU signals; *3) inverse kinematics solver* solves for the joint rotations from positions, i.e. the inverse kinematics (IK) problem. For *fusion-based global translation estimation*, we combine the following two methods: *1) foot-ground contact estimation* estimates the foot-ground contact probabilities for both feet from the leaf joint positions and the IMU measurements, and calculates the root translations on the assumption that the foot with a larger contact probability is not moving; *2) root velocity regressor* regresses the local velocities of the root in its own coordinate frame from all joint positions and the IMU signals.

The design of using separate steps in the pose estimation is based on our observation that estimating joint positions as an intermediate representation is easier than regressing the rotations directly. We attribute this to the nonlinearity of the rotation representation, compared with the positional formulation. On the other hand, performing IK tasks with neural networks has already been well investigated by previous works [Holden 2018; Zhou et al. 2018, 2020a,b]. However, these works neglect the importance of motion history. As the IK problem is ill-posed, the motion ambiguity is commonly seen, which can only be eliminated according to motion history. Here, in each stage, we incorporate a bidirectional recurrent neural network (biRNN) [Schuster and Paliwal 1997] to maintain motion history. This is similar to [Huang et al. 2018], but due to our 3-stage design, our network is significantly smaller with higher accuracy.

As for the global translation estimation, the intuition behind is to predict which foot contacts the ground and fix the contacting foot to deduce the root movement from the estimated pose. Besides, to cope with the cases where no foot is on the ground, we further use an RNN to regress the root translation as a complement. Finally, our method merges these two estimations to predict the final translation using the fusion weight determined by the foot-ground contact probability. The inspiration comes from [Shimada et al. 2020; Zou et al. 2020] which leverage foot-ground contact constraints to reduce foot-sliding artifacts in vision-based mocap, but we have a different purpose which is to solve for the translation from the estimated pose. As shown in the experiments, this method can handle most movements like walking, running, and challenging jumping.

We evaluate our approach on public datasets, where we outperform the previous works DIP and SIP by a large margin both qualitatively and quantitatively with various metrics, including positional and rotational error, temporal smoothness, runtime, etc. We also present live demos that capture a variety of challenging motions where the performer can act freely regardless of occlusion or range. These experiments demonstrate our significant superiority to previous works. In conclusion, our contributions are:

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

- A very fast and accurate approach for real-time motion capture with global translation estimation using only 6 IMUs.
- A novel structure to perform pose estimation which explicitly estimates joint positions as intermediate subtasks, resulting in higher accuracy and less runtime.
- The first method to explicitly estimate global translations in real-time from as sparse as 6 IMUs only.

2 RELATED WORK

The research on human motion capture has a long history, especially for the vision-based motion capture. Since this is not the category of this work, we refer readers to the surveys [Moeslund and Granum 2001; Moeslund et al. 2006] for more information. In this section, we mainly review the works using inertial sensors (or combined with other sensors) which are closely related to our approach.

2.1 Combining IMUs with Other Sensors or Cameras

Typically, a 9-axis IMU contains 3 components: an accelerometer that measures accelerations, a gyroscope that measures angular velocities, and a magnetometer that measures directions. Based on these direct measurements, the drift-free IMU orientations can be solved [Bachmann et al. 2002; Del Rosario et al. 2018; Foxlin 1996; Roetenberg et al. 2005; Vitali et al. 2020] leveraging Kalman filter or complementary filter algorithms. However, reconstructing human poses from a sparse set of IMUs is an under-constrained problem, as the sensors can only provide orientation and acceleration measurements which are insufficient for accurate pose estimation. To cope with this difficulty, one category of works [Liu et al. 2011; Vlasic et al. 2007] propose to utilize additional ultrasonic distance sensors to reduce the drift in joint positions. In the work of [Liu et al. 2011], database searching is used for similar sensor measurements, and the results help to construct a local linear model to regress the current pose online. Although the global position can be determined by leveraging the distance sensors, subjects can only act within a fixed volume to keep that the distance can be measured by ultrasonic sensors. Another category of works propose to combine IMUs with videos [Gilbert et al. 2018; Henschel et al. 2020; Malleson et al. 2019, 2017; Marcard et al. 2016; Pons-Moll et al. 2011, 2010; Zhang et al. 2020], RGB-D cameras [Helten et al. 2013; Zheng et al. 2018], or optical markers [Andrews et al. 2016]. Gilbert et al. [Gilbert et al. 2018] fuse multi-viewpoint videos with IMU signals and estimate human poses using 3D convolutional neural networks and recurrent neural networks. Global positions can also be determined in some works [Andrews et al. 2016; Malleson et al. 2019, 2017] due to the additional information from vision. Although these works achieve great accuracy, due to the need of vision, the capture suffers from occlusion and challenging lighting conditions, and can only be performed within a restricted area to keep the subject inside the camera field of view. To this end, Marcard et al. [von Marcard et al. 2018] propose to use a moving camera to break the limitation on the space volume. Nonetheless, all the methods that require visual or distance sensor inputs are substantially limited by the capture environment and occlusion.

2.2 Methods Based on Pure Inertial Sensors

As opposed to methods based on the fusion of IMUs and other sensors, approaches using pure IMUs do not suffer from occlusion and restricted recording environment and space. Commercial inertial motion capture systems such as Xsens MVN [Schepers et al. 2018] use 17 body-worn IMUs which can fully determine the orientations of all bones of a kinematic body model. However, the large number of sensors are intrusive to the performer and suffer from a long setup time. Though reducing the number of IMUs can significantly improve user experience, reconstructing human poses from a sparse set of IMUs is a severely underconstrained problem. Previous works take use of sparse accelerometers [Riaz et al. 2015; Slyper and Hodgins 2008; Tautges et al. 2011] and leverage a prerecorded motion database. They search in the database for similar accelerations when predicting new poses using a lazy learning strategy [Aha 1997], and demonstrate that learning with pure accelerometers is feasible. However, the searching-based methods cannot fully explore the input information. Due to the fundamental instability of accelerometer measurements and their weak correlations to the poses, the performance of these works is limited. Also, a trade-off between runtime performance and the database size is inevitable. In [Schwarz et al. 2009], support vector machine (SVM) and Gaussian processes regression (GPR) are used to categorize the actions and predict full-body poses using sparse orientation measurements. With the development of the sensing technique, inertial measurement units that measure both accelerations and orientations become common and popular. Instead of using only accelerations or orientations, recent works leverage both to make full use of IMUs and achieve better accuracy. Convolutional neural network (CNN) is used in [Hannink et al. 2016] to estimate gait parameters from inertial measurements for medical purposes. The pioneering work, SIP [Marcard et al. 2017], presents a method to solve for the human motions with only 6 IMUs. As an iterative optimization-based method, it has to operate in an offline manner, making real-time application infeasible. The state-of-the-art work, DIP [Huang et al. 2018], proposes to use only 6 IMUs to reconstruct full body poses in real-time. The authors adopt a bidirectional recurrent neural network (biRNN) [Schuster and Paliwal 1997] to directly learn the mapping from IMU measurements to body joint rotations. DIP achieves satisfying capture quality and efficiency, but there is still space to further improve it. In our approach, we demonstrate that decomposing this task into multiple stages, i.e. estimating joint positions as an intermediate representation before regressing joint angles, can significantly improve the accuracy and reduce the runtime. Equally importantly, DIP cannot estimate the global movement of the subject, which is an indispensable part of motion capture. In this paper, we propose a novel method to estimate the global translation of the performer *without any direct distance measurement*, which is a hybrid of supporting-foot-based deduction and network-based prediction.

3 METHOD

Our task is to estimate poses and translations of the subject in real-time using 6 IMUs. As shown in Figure 3, the 6 IMUs are mounted on the pelvis, the left and right lower leg, the left and right forearm, and the head. In the following, we refer to these joints as *leaf joints* except

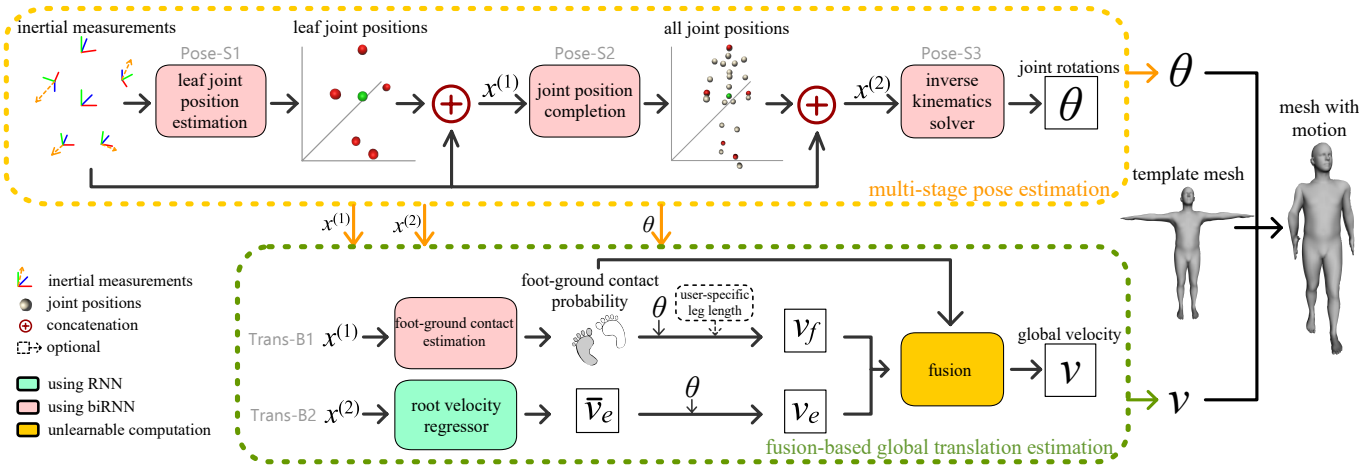

Fig. 2. Overview of our pipeline. We divide our task into 2 subtasks: *multi-stage pose estimation* and *fusion-based global translation estimation*. The pose estimation subtask is formulated into 3 stages, where: *1)* Pose-S1 estimates 5 leaf joint positions relative to the root from IMU signals, and its output is concatenated to the inertial measurements as $\boldsymbol{x}^{(1)}$; *2)* Pose-S2 estimates all joint positions from $\boldsymbol{x}^{(1)}$, and the output is again concatenated to the inertial measurements as $\boldsymbol{x}^{(2)}$; *3)* Pose-S3 regresses the joint rotations $\boldsymbol{\theta}$ from $\boldsymbol{x}^{(2)}$. The translation estimation subtask is addressed by two parallel branches, where: *1)* Trans-B1 estimates foot-ground contact probabilities from $\boldsymbol{x}^{(1)}$ and calculates the root velocity $\boldsymbol{v}_f$ using forward kinematics assuming that the foot with a higher probability is still; *2)* Trans-B2 estimates the local velocity of the root joint $\bar{\boldsymbol{v}}_e$ in its own coordinate frame from $\boldsymbol{x}^{(2)}$, which is then transformed to the world space as $\boldsymbol{v}_e$ according to the root rotation measured by the sensor mounted on the pelvis. Such two estimated velocities are then fused to form the final global translation $\boldsymbol{v}$ according to the foot-ground contact probabilities.

Fig. 3. IMU placement. The 6 IMUs are mounted on the left and right forearm, the left and right lower leg, the head, and the pelvis. We require the sensors to be tightly bounded around the joints with arbitrary orientations.

the pelvis, which is named as *root joint*. We divide this task into two subtasks: *multi-stage pose estimation* (Section 3.2) and *fusion-based global translation estimation* (Section 3.3). The system is illustrated in Figure 2. Detailed structures and hyper-parameters for all networks are presented in Appendix B.

3.1 System Input

We take the rotation and acceleration measurements of each IMU as the overall input of the system. We align these measurements into the same reference frame and normalize them to obtain the concatenated input vector as $\boldsymbol{x}^{(0)} = [\boldsymbol{a}_{\text{root}}, \cdots, \boldsymbol{a}_{\text{rarm}}, \boldsymbol{R}_{\text{root}}, \cdots, \boldsymbol{R}_{\text{rarm}}] \in \mathbb{R}^{72}$ where $\boldsymbol{a} \in \mathbb{R}^3$ is the acceleration and $\boldsymbol{R} \in \mathbb{R}^{3\times3}$ is the rotation matrix. We use $\boldsymbol{x}^{(0)}(t)$ to refer to the measurements of the tth frame, and the superscript $^{(0)}$ means it is the overall input. Please refer to Appendix A for more details on the sensor preprocessing.

3.2 Multi-stage Pose Estimation

In this section, we introduce our multi-stage method to estimate body poses, i.e. the rotation of each joint, from sparse IMU measurements. This task is very difficult due to the ambiguity of the mapping from extremely sparse inertial data to full body joint angles. The key to address this challenging task is the use of *1)* the prior knowledge on human poses and *2)* the temporal information. Because of the diversity of human motions and the anatomic restrictions, the pose prior is hard to model or learn. While the previous work, DIP [Huang et al. 2018], proposes to adopt a bidirectional recurrent neural network (biRNN) to learn the direct mapping from IMU measurements to joint rotations, we demonstrate that using joint positions as an intermediate representation is significantly helpful for the model to learn the complex motion prior. More specifically, we first predict joint positions as an intermediate task in Pose-S1 (short for *pose estimation stage 1*) and Pose-S2 (Section 3.2.1), and then solve for the joint angles in Pose-S3 (Section 3.2.2). Further, to fully utilize the temporal information, we adopt biRNNs to leverage the past and future frames. As shown in Section 5.2, our approach achieves superior capture quality and runtime performance.

3.2.1 Pose-S1&S2: estimating joint positions. In Pose-S1&S2, we estimate joint positions from IMU measurements. We explicitly separate this task into two different stages to take the hierarchy of the human kinematic tree into consideration. Specifically, we divide the human joints into two sets: the leaf joints and the non-leaf joints. The leaf joints are mounted with sensors and have direct

measurements. Usually, they have larger movements. The inner non-leaf joints are without any direct inertial data and have relatively smaller motions. Due to the correlation between body joints, the information of the leaf joints is helpful for predicting the coordinates of inner ones. We first regress the positions of the leaf joints in Pose-S1, and then localize the other joints in Pose-S2.

The input to Pose-S1 is the inertial measurement vector $\boldsymbol{x}^{(0)}(t)$, and the output is the root-relative positions of the five leaf joints $\boldsymbol{p}_{\text{leaf}}(t) = [\boldsymbol{p}_{\text{lleg}}(t), \cdots, \boldsymbol{p}_{\text{rarm}}(t)] \in \mathbb{R}^{15}$. We use a standard biRNN [Schuster and Paliwal 1997] with Long Short-Term Memory (LSTM) cells [Hochreiter and Schmidhuber 1997] to learn the mapping from IMU measurements to leaf joint positions. Due to the very sparse measurements, it is common that the IMUs show identical signals while the subject is performing different animations. For example, when sitting still or standing still, the acceleration and orientation measurements of all the IMUs are almost the same. In such cases, the temporal information is the key to resolve this ambiguity, thus using RNNs is a natural choice. We choose to use biRNNs instead of vanilla RNNs because the future information is also of great help in this task, as stated in DIP. The loss function used to train the network is defined as:

$$\mathcal{L}_{\text{S1}} = \|\boldsymbol{p}_{\text{leaf}}(t) - \boldsymbol{p}^{\text{GT}}_{\text{leaf}}(t)\|_2^2, \tag{1}$$

where superscript $^{\text{GT}}$ denotes the ground truth.

The input of Pose-S2 is the concatenation of Pose-S1's output and the inertial measurements: $\boldsymbol{x}^{(1)}(t) = [\boldsymbol{p}_{\text{leaf}}(t), \boldsymbol{x}^{(0)}(t)] \in \mathbb{R}^{87}$. Based on the input information, Pose-S2 outputs the root-relative coordinates of all joints: $\boldsymbol{p}_{\text{all}}(t) = [\boldsymbol{p}_j(t) | j = 1, 2, \cdots, J-1] \in \mathbb{R}^{3(J-1)}$, where J is the number of joints in the human kinematic tree. Similar to Pose-S1, we use a biRNN for Pose-S2 with L2 loss:

$$\mathcal{L}_{\text{S2}} = \|\boldsymbol{p}_{\text{all}}(t) - \boldsymbol{p}^{\text{GT}}_{\text{all}}(t)\|_2^2. \tag{2}$$

3.2.2 Pose-S3: estimating joint rotations. In Pose-S3, we estimate joint rotations from joint positions. We concatenate the joint coordinates and the inertial measurements as $\boldsymbol{x}^{(2)}(t) = [\boldsymbol{p}_{\text{all}}(t), \boldsymbol{x}^{(0)}(t)] \in \mathbb{R}^{3(J-1)+72}$, which is the input of Pose-S3. The input vector is fed into a biRNN, which predicts the rotations of all non-root joints relative to the root in the 6D representation [Zhou et al. 2018]: $\boldsymbol{R}^{(6\text{D})}_{\text{all}}(t) = [\boldsymbol{R}^{(6\text{D})}_j(t) | j = 1, 2, \cdots, J-1] \in \mathbb{R}^{6(J-1)}$. We choose the 6D rotation representation over other common representations such as quaternions as the output for better continuity, as demonstrated in [Zhou et al. 2018]. We should note that the IMU measurements are fed into the networks as rotation matrices, while the only use of the 6D representation is in the output of Pose-S3. The loss function is defined as:

$$\mathcal{L}_{\text{S3}} = \|\boldsymbol{R}^{(6\text{D})}_{\text{all}}(t) - \boldsymbol{R}^{\text{GT},(6\text{D})}_{\text{all}}(t)\|_2^2. \tag{3}$$

The rotation of the root joint $\boldsymbol{R}_{\text{root}}$ is directly measured by the sensor placed on the pelvis. Combining all these rotations, we convert them to the rotation matrix formulation and obtain the full body pose as $\boldsymbol{\theta} = [\boldsymbol{R}_j(t) | j = 0, 1, \cdots, J] \in \mathbb{R}^{9J}$.

3.3 Fusion-based Global Translation Estimation

In this section, we explain our method to estimate global translations from the IMU measurements and the estimated body poses. This task is even more challenging due to the lack of direct distance measurements, and the acceleration measurements are too noisy to be used directly [Marcard et al. 2017]. Previous works address this task by introducing additional vision inputs [Andrews et al. 2016; Henschel et al. 2020; Malleson et al. 2019, 2017] or distance measurements [Liu et al. 2011; Vlasic et al. 2007], which increase the complexity of the system. While the work of SIP [Marcard et al. 2017] estimates global translations from IMUs only, it has to run in an offline manner. To our best knowledge, we are the first method that addresses the task of real-time prediction of global translations from sparse IMUs. To do so, we need to estimate the per-frame velocity of the root joint, i.e. the translation between the current frame and the previous frame. We propose a fusion-based approach that comprises two parallel branches. In Trans-B1 (short for *translation estimation branch 1*) (Section 3.3.1), we infer the root velocity based on the sequential pose estimation, in combination with the prediction of foot-ground contact probabilities. In Trans-B2 (Section 3.3.2), we use an RNN to predict the root velocity from joint positions and inertial measurements. These two branches run in a parallel style, and the final estimation is a fusion of the two branches based on the foot state (Section 3.3.3). The intuition behind is that by assuming the foot on the ground is not moving, the per-frame velocity can be deduced from the motion of the subject. However, this estimation is not totally accurate and inevitably fails when both feet are not on the ground, e.g., jumping or running. Therefore, a neural network is used here for complementary estimation. We demonstrate in Section 5.3 that such a hybrid approach gives more accurate results than using either branch alone.

3.3.1 Trans-B1: supporting-foot-based velocity estimation. In Trans-B1, we estimate the velocity of the root joint based on the estimation of body poses. First, we use a biRNN to estimate the foot-ground contact probability for each foot, i.e. the likelihood that the foot is on the ground, formulated as $\boldsymbol{s}_{\text{foot}} = [s_{\text{lfoot}}, s_{\text{rfoot}}] \in \mathbb{R}^2$. The input of this network is the leaf joint positions and the inertial measurements, $\boldsymbol{x}^{(1)}$. We assume that the foot with a higher on-ground probability is not moving between two adjacent frames, referred to as the supporting foot. The corresponding probability is denoted as $s = \max\{s_{\text{lfoot}}, s_{\text{rfoot}}\}$. Then, we apply the estimated pose parameters $\boldsymbol{\theta}$ on the kinematic model with optional user-specific leg lengths $\boldsymbol{l}$, and the root velocity is essentially the coordinate difference of the supporting foot between two consecutive frames, denoted as $\boldsymbol{v}_f$:

$$\boldsymbol{v}_f(t) = \text{FK}(\boldsymbol{\theta}(t-1); \boldsymbol{l}) - \text{FK}(\boldsymbol{\theta}(t); \boldsymbol{l}), \tag{4}$$

where $\text{FK}(\cdot)$ is the forward kinematics function that calculates the position of the supporting foot from the pose parameters. To train the model, we use a cross-entropy loss defined as:

$$\begin{aligned}\mathcal{L}_{\text{B1}} = &- s^{\text{GT}}_{\text{lfoot}} \log s_{\text{lfoot}} - (1 - s^{\text{GT}}_{\text{lfoot}}) \log(1 - s_{\text{lfoot}}) \\ &- s^{\text{GT}}_{\text{rfoot}} \log s_{\text{rfoot}} - (1 - s^{\text{GT}}_{\text{rfoot}}) \log(1 - s_{\text{rfoot}}).\end{aligned} \tag{5}$$

3.3.2 Trans-B2: network-based velocity estimation. While the pose-based method in Trans-B1 is straightforward, it is substantially incapable of the cases where both feet are off the ground simultaneously. Further, the errors in the pose estimation will also affect the accuracy. To this end, we additionally adopt a neural network to

estimate the velocity based on the predicted joint positions and the IMU measurements, $\boldsymbol{x}^{(2)}$, in parallel to Trans-B1. Instead of using a biRNN as in other steps, here we choose to use an RNN. This is because we find that estimating per-frame velocity accurately requires more previous frames, and a biRNN with a sufficiently large window size cannot satisfy our runtime requirements (see the experiments in Section 5.3). The output of the RNN is the velocity $\bar{\boldsymbol{v}}_e$ in the coordinate system of the root joint, and we transform it to the global system using the root rotation $\boldsymbol{R}_{\text{root}}$ in $\boldsymbol{\theta}$ as:

$$\boldsymbol{v}_e = \boldsymbol{R}_{\text{root}} \bar{\boldsymbol{v}}_e. \tag{6}$$

The loss function is defined as:

$$\mathcal{L}_{\text{B2}} = \mathcal{L}_{\text{vel}}(1) + \mathcal{L}_{\text{vel}}(3) + \mathcal{L}_{\text{vel}}(9) + \mathcal{L}_{\text{vel}}(27), \tag{7}$$

where:

$$\mathcal{L}_{\text{vel}}(n) = \sum_{m=0}^{\lfloor T/n \rfloor - 1} \left\| \sum_{t=m}^{mn+n-1} (\bar{\boldsymbol{v}}_e(t) - \bar{\boldsymbol{v}}_e^{\text{GT}}(t)) \right\|_2^2. \tag{8}$$

Here, T is the total number of frames in the training sequence. This loss measures the differences between the predicted and the ground truth translations in every consecutive 1, 3, 9, and 27 frames. We find that using $\mathcal{L}_{\text{vel}}(1)$ alone makes the network only focus on neighboring frames, resulting in unstable estimation and large errors. On the contrary, the supervision on larger frame ranges helps to reduce the accumulative error.

3.3.3 Fusing two branches. Lastly, we fuse the estimated velocity $\boldsymbol{v}_f$ and $\boldsymbol{v}_e$ to get the final global translation $\boldsymbol{v}$ based on the foot-ground contact likelihood. We set an upper threshold $\bar{s}$ and a lower threshold $\underline{s}$ for the predicted foot-ground contact probability s. The output global velocity is then computed as:

$$\boldsymbol{v} = \begin{cases} \boldsymbol{v}_e & 0 \le s < \underline{s} \\ \frac{s-\underline{s}}{\bar{s}-\underline{s}} \boldsymbol{v}_e + \frac{s-\underline{s}}{\bar{s}-\underline{s}} \boldsymbol{v}_f & \underline{s} \le s < \bar{s} \\ \boldsymbol{v}_f & \bar{s} \le s \le 1 \end{cases}. \tag{9}$$

We empirically set $\underline{s} = 0.5$ and $\bar{s} = 0.9$ in our model. The linear interpolation provides a smooth and automatic transition between the two branches.

4 IMPLEMENTATION

In this section, we provide more details on our implementation. Specifically, we introduce the kinematic model in Section 4.1, the training data in Section 4.2, and other related details in Section 4.3.

4.1 Kinematic Model

We use the SMPL [Loper et al. 2015] skeleton as our kinematic model, and use the corresponding mesh for visualization. The model is defined as:

$$M(\boldsymbol{\theta}) = W(\bar{\mathbf{T}}, \boldsymbol{J}, \boldsymbol{\theta}, \mathcal{W}), \tag{10}$$

where $\bar{\mathbf{T}}$ is the template mesh in the rest pose, $\boldsymbol{J}$ is 24 body joints, $\boldsymbol{\theta}$ is the pose parameters in terms of joint angles, $\mathcal{W}$ is the blend weights, and $W(\cdot)$ is the linear blend skinning function. As the body shape is not our concern, we leave out the shape- and pose-blendshape terms in the original paper. We should note that the IMU measurements do not contain any information about the subject's body shape, and the pose estimation task is also shape-agnostic. However, in Trans-B1, leg lengths will directly affect the prediction of global translations which are deduced from foot movements. Here, we assume that the leg lengths of the subject can be measured in advance, and the global translations are calculated based on this measurement. Otherwise, we use the mean SMPL model, which will give a less accurate estimation due to the disparity in leg lengths, but still being plausible.

Table 1. Dataset overview. We use DIP-IMU [Huang et al. 2018], TotalCapture [Trumble et al. 2017], and AMASS [Mahmood et al. 2019] dataset for our training and evaluation. The table shows pose parameters, IMU measurements, global translations, foot-ground contact states, and the total length in minutes for each dataset. "Y" means that the dataset contains such information. "N" means that the dataset does not contain such information. "S" means that the data is synthesized from other information.

Dataset	Pose	IMU	Translation	Contact	Minutes
DIP-IMU[a]	Y	Y	N	N	80
TotalCapture	Y[b]	Y	Y	S	49
AMASS[c]	Y	S	Y	S	1217

[a]Some training sequences that have too many "nan" IMU measurements are discarded.
[b]The ground truth SMPL pose parameters are provided by DIP [Huang et al. 2018].
[c]We select and sample the original AMASS motions [Mahmood et al. 2019] in 60 fps.

4.2 Training Data

In our implementation, each network requires different types of data and is trained individually. The relevant datasets include: *1)* DIP-IMU [Huang et al. 2018], which consists of IMU measurements and pose parameters for around 90 minutes of motions performed by 10 subjects wearing 17 IMUs; *2)* TotalCapture [Trumble et al. 2017], which consists of IMU measurements, pose parameters, and global translations for around 50 minutes of motions performed by 5 subjects wearing 13 IMUs; *3)* AMASS [Mahmood et al. 2019], which is a composition of existing motion capture (mocap) datasets and contains pose parameters and global translations for more than 40 hours of motions performed by over 300 subjects. The overview of the datasets is shown in Table 1. As TotalCapture is relatively small, we use it only for evaluation as an examination for cross-dataset generalization. In the following, we introduce how to use the training data for each subtask, i.e. pose estimation (Section 4.2.1) and global translation estimation (Section 4.2.2).

4.2.1 Training data for pose estimation. To train the networks in Pose-S1 and Pose-S2, we need sequential IMU measurements (including rotations and accelerations) and root-relative joint coordinates. While DIP-IMU contains such data, it is not diverse enough to train a model with sufficient generalization ability. Hence, we synthesize inertial data for AMASS dataset which is much larger and contains more variations. To do so, we place virtual IMUs on the corresponding vertices of the SMPL mesh, and then the sequential positions and rotations of each IMU can be inferred from the pose parameters using Equation 10. We use the following method to synthesize the acceleration measurements:

$$\boldsymbol{a}_i(t) = \frac{\boldsymbol{x}_i(t-n) + \boldsymbol{x}_i(t+n) - 2\boldsymbol{x}_i(t)}{(n\Delta t)^2}, i = 1, 2, \cdots, 6, \tag{11}$$

where $\boldsymbol{x}_i(t)$ denotes the coordinate of the ith IMU at frame t, $\boldsymbol{a}_i(t) \in \mathbb{R}^3$ is the acceleration of IMU i at frame t, and Δt is the time interval between two consecutive frames. To cope with jitters in the data, we do not compute the accelerations based on adjacent frames (i.e. $n = 1$), but use relatively apart frames for smoother accelerations by setting $n = 4$. Please refer to Appendix C for more details. We synthesize a subset of AMASS dataset which consists of about 1217 minutes of motions in 60 fps, which sufficiently cover the motion variations. To make Pose-S2 robust to the prediction errors of leaf-joint positions, during training, we further add Gaussian noise to the leaf-joint positions with $\sigma = 0.04$. We use DIP-IMU and synthetic AMASS as training datasets for the pose estimation task, and leave TotalCapture for evaluation. To train Pose-S3, we need inertial measurements and mocap data in the form of joint angles. We again use DIP-IMU and synthetic AMASS dataset for Pose-S3 with additional Gaussian noise added to the joint positions, whose standard deviation is empirically set to $\sigma = 0.025$.

4.2.2 Training data for translation estimation. In Trans-B1, a biRNN is used to predict foot-ground contact probabilities, thus we need binary annotations for foot-ground contact states. To generate such data, we apply the 60-fps pose parameters and root translations on the SMPL model to obtain the coordinates of both feet, denoted as $\boldsymbol{x}_{\text{lfoot}}(t)$ and $\boldsymbol{x}_{\text{rfoot}}(t)$ for frame t. When the movement of one foot between two consecutive frames is less than a threshold u, we mark it as contacting the ground. We empirically set $u = 0.008$ meters. In this way, we automatically label AMASS dataset using:

$$s^{\text{GT}}_{\text{lfoot}}(t) = \begin{cases} 1 & \text{if } \|\boldsymbol{x}_{\text{lfoot}}(t) - \boldsymbol{x}_{\text{lfoot}}(t-1)\|_2 < u \\ 0 & \text{otherwise} \end{cases}, \tag{12}$$

$$s^{\text{GT}}_{\text{rfoot}}(t) = \begin{cases} 1 & \text{if } \|\boldsymbol{x}_{\text{rfoot}}(t) - \boldsymbol{x}_{\text{rfoot}}(t-1)\|_2 < u \\ 0 & \text{otherwise} \end{cases}, \tag{13}$$

where $s^{\text{GT}}_{\text{lfoot}}(t)$ and $s^{\text{GT}}_{\text{rfoot}}(t)$ are the foot-ground contact state labels for frame t. For better robustness, we additionally apply Gaussian noise to the input of Trans-B1 during training, with the standard deviation set to $\sigma = 0.04$. Since DIP-IMU does not contain global translations, we only use synthetic 60-fps AMASS as the training dataset and leave TotalCapture for evaluation.

In Trans-B2, an RNN is used to predict the velocity of the root joint in its own coordinate system. Since the root positions are already provided in AMASS dataset, we compute the velocity and convert it from the global space into the root space using:

$$\bar{\boldsymbol{v}}^{\text{GT}}_e(t) = (\boldsymbol{R}^{\text{GT}}_{\text{root}}(t))^{-1}(\boldsymbol{x}^{\text{GT}}_{\text{root}}(t) - \boldsymbol{x}^{\text{GT}}_{\text{root}}(t-1)), \tag{14}$$

where $\boldsymbol{R}^{\text{GT}}_{\text{root}}(t)$ is the ground truth root rotation at frame t and $\boldsymbol{x}^{\text{GT}}_{\text{root}}(t)$ is the ground truth root position in world space at frame t. Note that the frame rate is fixed to be 60 fps for training and testing, and the *velocity* is defined as the translation between two consecutive 60-fps frames in this paper. We intend to use Trans-B2 mainly for the cases where both feet are off the ground, thus we only use such kind of sequences from AMASS to construct the training data. Specifically, we run the well-trained Trans-B1 network and collect the sequence clips where the minimum foot-ground contact probability is lower than $\bar{s}$, i.e. $\min_{t \in \mathcal{F}} s(t) < \bar{s}$, where $\mathcal{F}$ is the set containing all frames of the fragment and $|\mathcal{F}| \leq 300$. During training, Gaussian noise of $\sigma = 0.025$ is added to the input of Trans-B2.

4.3 Other Details

All training and evaluation processes run on a computer with an Intel(R) Core(TM) i7-8700 CPU and an NVIDIA GTX1080Ti graphics card. The live demo runs on a laptop with an Intel(R) Core(TM) i7-10750H CPU and an NVIDIA RTX2080 Super graphics card. The model is implemented using PyTorch 1.7.0 with CUDA 10.1. The front end of our live demo is implemented using Unity3D, and we use Noitom[3] Legacy IMU sensors to collect our own data. We separately train each network with the batch size of 256 using an Adam [Kingma and Ba 2014] optimizer with a learning rate $\text{lr} = 10^{-3}$. We follow DIP to train the models for the pose estimation task using synthetic AMASS first and fine-tune them on DIP-IMU which contains real IMU measurements. To avoid the vertical drift due to the error accumulation in the estimation of translations, we add a gravity velocity $v_G = 0.018$ to the Trans-B1 output $\boldsymbol{v}_f$ to pull the body down.

5 EXPERIMENTS

In this section, we first introduce the data and metrics used in our experiments in Section 5.1. Using the data and metrics, we compare our method with previous methods qualitatively and quantitatively in Section 5.2. Next, we evaluate our important design choices and key technical components by ablative studies in Section 5.3. Finally, to further demonstrate the power of our technique, we show various real-time results containing strong occlusion, wide range motion space, dark and outdoor environment, close interaction, and multiple users in Section 5.4. Following DIP [Huang et al. 2018], we adopt two settings of inference: the *offline* setting where the full sequence is available at the test time, and the *online* (*real-time*) setting where our approach accesses 20 past frames, 1 current frame, and 5 future frames in a window sliding manner, with a tolerable latency of 83ms. Please note that all the results are estimated from the real IMU measurements, and no temporal filter is used in our method. Following DIP, we do not give rotation freedoms to the wrists and ankles because we have no observation to solve them.

5.1 Data and Metrics

The pose evaluations are conducted on the test split of DIP-IMU [Huang et al. 2018] and TotalCapture [Trumble et al. 2017] dataset. The global translation evaluations are conducted on TotalCapture alone, as DIP-IMU does not contain global movements. We use the following metrics for quantitative evaluations of poses: *1) SIP error* measures the mean global rotation error of upper arms and upper legs in degrees; *2) angular error* measures the mean global rotation error of all body joints in degrees; *3) positional error* measures the mean Euclidean distance error of all estimated joints in centimeters with the root joint (Spine) aligned; *4) mesh error* measures the mean Euclidean distance error of all vertices of the estimated body mesh also with the root joint (Spine) aligned. The vertex coordinates are calculated by applying the pose parameters to the SMPL [Loper et al. 2015] body model with the mean shape using Equation 10. Note

[3] https://www.noitom.com/

that the errors in twisting (rotations around the bone) cannot be measured by positional error, but will be reflected in mesh error. *5) Jitter error* measures the average *jerk* of all body joints in the predicted motion. Jerk is the third derivative of position with respect to time and reflects the smoothness and naturalness of the motion [Flash and Hogan 1985]. A smaller average jerk means a smoother and more natural animation.

5.2 Comparisons

Quantitative and qualitative comparisons with DIP [Huang et al. 2018] and SIP/SOP (SOP is a simplified version of SIP that only uses orientation measurements) [Marcard et al. 2017] are conducted and the results are shown in this section. We should note that our test data is slightly different from [Huang et al. 2018] for both DIP-IMU and TotalCapture, resulting in some inconsistency with the values reported by DIP. More specifically, the DIP-IMU dataset contains both raw and calibrated data, and the results in the DIP paper are based on the latter. However, the calibrated data removes the root inertial measurements which are required by our method to estimate the global motions, so we have to perform the comparisons on the raw data. As for the TotalCapture dataset, the ground truth SMPL pose parameters are acquired from the DIP authors. To evaluate DIP on the test data, we use the DIP model and the code released by the authors. No temporal filtering technique is applied.

5.2.1 Quantitative comparisons. The quantitative comparisons are shown in Table 2 for the offline setting and Table 3 for the online setting. We report the mean and standard deviation for each metric in comparison with DIP (both offline and online) and SIP/SOP (only offline). As shown in the tables, our method outperforms all previous works in all metrics. We attribute our superiority to the multi-stage structure that first estimates joint coordinates in the positional space and then solves for the angles in the rotational space. We argue that human poses are easier to estimate in the joint-coordinate space, and with the help of the coordinates, joint rotations can be better estimated. A fly in the ointment is that we can see an accuracy gap between online and offline settings. The reason is that the offline setting uses much more temporal information to solve the pose of the current frame, while the online setting does not do this to avoid noticeable delay. Since our approach fully exploits such temporal information to resolve the motion ambiguities, there is an inevitable accuracy reduction when switching from the offline setting to the online setting. Nevertheless, we still achieve the state-of-the-art online capture quality which is visually pleasing as shown in the qualitative results and the live demos.

5.2.2 Qualitative comparisons. Here, we visually compare offline and online results for pose estimation between our method and the state-of-the-art work DIP. Some selected frames from the two datasets are shown in Figure 4 and more comparisons are shown in our video. From the top three rows in Figure 4 picked from DIP-IMU dataset, we see that our approach reconstructs the upper and lower body well, while DIP does not accurately estimate the poses of the upper arms and legs. Since inner joint rotations are not directly measured by any sensor, high ambiguities exist in the mapping from leaf joint rotations to poses of the whole body. The explicit estimation of joint positions in our pipeline helps the network to make better use of the acceleration information, which is the key to solve such ambiguities. Thus, our method shows greater accuracy in inner joint rotations. The following three rows are some challenging poses selected from the TotalCapture dataset. Our method performs slightly better for the side-bending pose as shown in the top row. In the middle row, our method correctly estimates the rotations of upper arms where DIP fails, while both have similar forearm rotations. We show a challenging case in the last row where DIP fails but we still give a satisfying estimation. Again, we attribute such superiority of our method to the intermediate joint position estimation tasks which have a regularization effect on the final result.

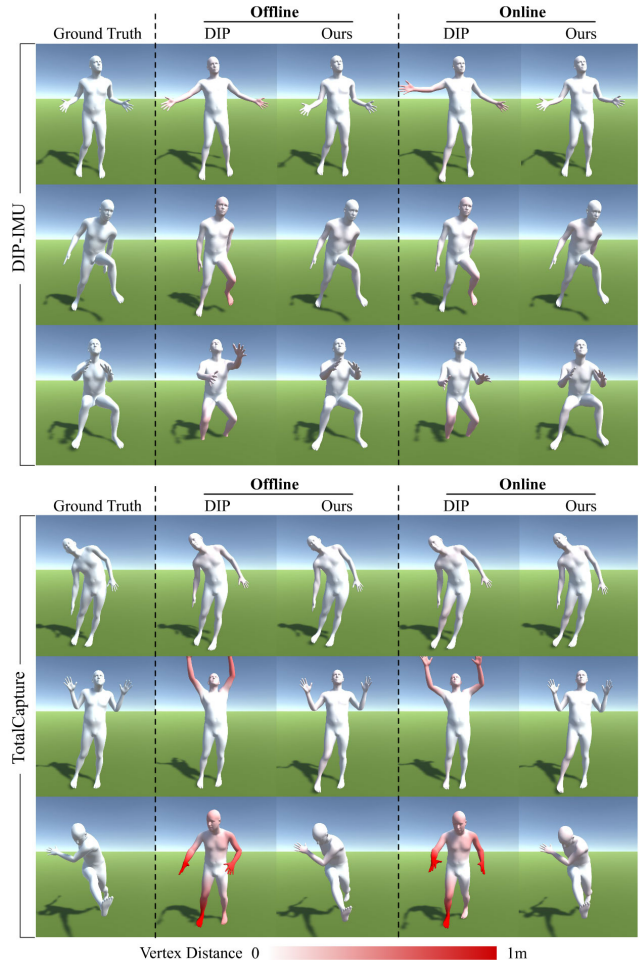

Fig. 4. Qualitative comparisons between our method and DIP. We perform both offline and online comparisons on DIP-IMU (top-three rows) and TotalCapture (bottom-three rows) dataset, and pick some results here. More comparisons can be found in our video. Each vertex is colored by its distance to the ground truth location. A complete-red vertex means a distance greater than 1m.

Table 2. Offline comparison results for body poses. We compare our method with SIP/SOP [Marcard et al. 2017] and DIP [Huang et al. 2018] on TotalCapture [Trumble et al. 2017] and DIP-IMU [Huang et al. 2018] dataset. The mean and standard deviation (in parentheses) of the SIP error, angular error, positional error, mesh error, and jitter error are reported.

	TotalCapture					DIP-IMU				
	SIP Err (deg)	Ang Err (deg)	Pos Err (cm)	Mesh Err (cm)	Jitter (10^2m/s^3)	SIP Err (deg)	Ang Err (deg)	Pos Err (cm)	Mesh Err (cm)	Jitter (10^2m/s^3)
SOP	23.09 (±12.37)	17.14 (±8.54)	9.24 (±5.33)	10.58 (±6.04)	8.17 (±13.55)	24.56 (±12.75)	9.83 (±5.21)	8.17 (±4.74)	9.32 (±5.27)	5.66 (±9.49)
SIP	18.54 (±9.67)	14.84 (±7.26)	7.65 (±4.32)	8.60 (±4.83)	8.27 (±17.36)	21.02 (±9.61)	8.77 (±4.38)	6.66 (±3.33)	7.71 (±3.80)	3.86 (±6.32)
DIP	18.79 (±11.85)	17.77 (±9.51)	9.61 (±5.76)	11.34 (±6.45)	28.86 (±29.18)	16.36 (±8.60)	14.41 (±7.90)	6.98 (±3.89)	8.56 (±4.65)	23.37 (±23.84)
Ours	**14.95 (±6.90)**	**12.26 (±5.59)**	**5.57 (±3.09)**	**6.36 (±3.47)**	**1.57 (±2.93)**	**13.97 (±6.77)**	**7.62 (±4.01)**	**4.90 (±2.75)**	**5.83 (±3.21)**	**1.19 (±1.76)**

Table 3. Online comparison results for body poses. We compare our method with DIP [Huang et al. 2018] in the real-time setting on TotalCapture [Trumble et al. 2017] and DIP-IMU [Huang et al. 2018] dataset.

	TotalCapture					DIP-IMU				
	SIP Err (deg)	Ang Err (deg)	Pos Err (cm)	Mesh Err (cm)	Jitter (10^2m/s^3)	SIP Err (deg)	Ang Err (deg)	Pos Err (cm)	Mesh Err (cm)	Jitter (10^2m/s^3)
DIP	18.93 (±12.44)	17.50 (±10.10)	9.57 (±5.95)	11.40 (±6.87)	35.94 (±34.45)	17.10 (±9.59)	15.16 (±8.53)	7.33 (±4.23)	8.96 (±5.01)	30.13 (±28.76)
Ours	**16.69 (±8.79)**	**12.93 (± 6.15)**	**6.61 (± 3.93)**	**7.49 (± 4.35)**	**9.44 (±13.57)**	**16.68 (±8.68)**	**8.85 (± 4.82)**	**5.95 (± 3.65)**	**7.09 (± 4.24)**	**6.11 (±7.92)**

Table 4. Evaluation of the multi-stage design for pose estimation using DIP-IMU and TotalCapture dataset. "I" stands for IMU measurements; "LJ" means leaf joint positions; "AJ" denotes full joint positions; and "P" indicates body poses. We demonstrate the superiority (judging from the SIP error and jitter error) of our multi-stage method which estimates leaf and full joint positions as intermediate tasks.

	DIP-IMU		TotalCapture	
	SIP Err (deg)	Jitter (10^2m/s^3)	SIP Err (deg)	Jitter (10^2m/s^3)
I→P	14.43 (±7.77)	2.50 (±3.42)	23.16(±9.00)	3.34 (±5.72)
I→LJ→P	14.35 (±7.75)	2.22 (±3.32)	17.71 (±7.89)	2.90 (±5.09)
I→AJ→P	14.29 (±7.30)	1.23 (±1.82)	19.76 (±8.05)	1.60 (±2.94)
I→LJ→AJ→P	**13.97 (±6.77)**	**1.19 (±1.76)**	**14.95 (±6.90)**	**1.57 (±2.93)**

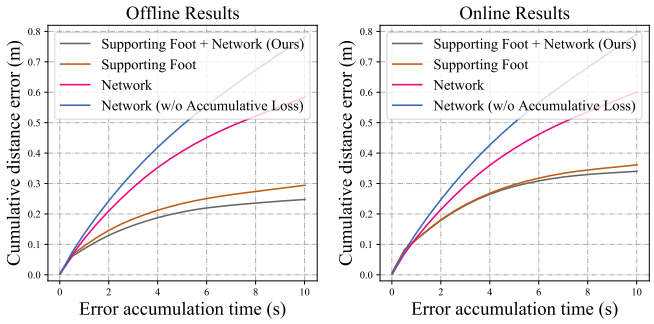

Fig. 5. Evaluation of global translation estimation. Here we evaluate cumulative distance errors with respect to time. The error of each solution is plotted as a curve. A lower curve means better translation accuracy. We demonstrate the superiority of our fusion-based method and also the effectiveness of the accumulative loss terms (Equation 7) used in the training of *Trans-B2*.

5.3 Evaluations

5.3.1 Multi-stage pose estimation. We demonstrate the superiority of our three-stage pose estimation structure here. We evaluate the following three variants of our pipeline: *1)* directly regressing joint rotations from inertial inputs, without any intermediate task; *2)* estimating joint rotations with the help of intermediate *leaf* joint coordinates regressed from inertial measurements, i.e. combining Pose-S2 and Pose-S3; *3)* estimating joint rotations with the help of intermediate *all* joint positions, which are directly regressed from inertial measurements, i.e. combining Pose-S1 and Pose-S2. We compare these variations with our original three-stage method on DIP-IMU and TotalCapture dataset in the offline setting. As shown in Table 4, estimating joint positions greatly helps with the pose estimation task, which leads to better accuracy and a significant reduction of jitters. We attribute this to the nonlinearity of human poses in the joint rotation representation, which makes it hard to learn how to extract useful information from the linear accelerations of body joints, resulting in an automatic discard of acceleration data as reported in DIP [Huang et al. 2018]. Hence, by estimating joint positions as an intermediate task, we make better use of acceleration information due to its linear correlation with positions, and as a result, we do not need any special loss for accelerations where DIP does. Another observation from Table 4 is that the leaf joint positions are helpful for the estimation of other joints. We think this is because estimating leaf joint positions is easier than inner ones as their inertia is directly measured by IMUs. Then, with the help of the leaf joints, it is easier to estimate the inner ones, as the leaf joints are affected by inner ones. So breaking down such a complex task into easier ones helps to achieve better results.

5.3.2 Fusion-based global translation estimation. We conduct an evaluation on the global translation estimation task to demonstrate the advantages of our hybrid supporting-foot- and network-based approach. Each alternative method is evaluated and compared with our fusion-based implementation. Specifically, we evaluate: *1)* the supporting-foot-based method in Trans-B1, which means that we only estimate foot-ground contact states and compute the global translations from poses; *2)* using the neural network to regress translations in Trans-B2, which means that we only estimate global translations from an RNN; *3)* running both branches and fusing them according to the estimated foot-ground contact probabilities, i.e. our

Table 5. Evaluation of the network of Trans-B2 on TotalCapture dataset. RNN and biRNN are compared using the cumulative distance errors with 1s (left) and 5s (right) accumulation time. We demonstrate that RNN is better suited for the velocity estimation task.

Subject	Dist Err@1s (cm)		Dist Err@5s (cm)	
	biRNN	RNN	biRNN	RNN
s1	14.20 (±7.51)	**11.34 (±5.49)**	47.74 (±20.29)	**40.07 (±16.73)**
s2	12.75 (±7.41)	**9.38 (±7.09)**	41.64 (±17.41)	**30.04 (±17.23)**
s3	15.76 (±8.38)	**12.15 (±6.79)**	52.45 (±21.05)	**41.32 (±18.12)**
s4	22.18 (±17.13)	**20.09 (±14.54)**	72.18 (±42.15)	**68.40 (±34.31)**
s5	18.54 (±10.53)	**13.31 (±6.59)**	55.42 (±27.53)	**46.93 (±17.81)**
total	15.50 (±9.05)	**12.18 (±7.36)**	50.77 (±22.80)	**41.49 (±19.28)**

hybrid method. Additionally, we evaluate *4)* running only the RNN in Trans-B2, but trained with a simple L2 loss that measures the mean square error of the predicted velocities, i.e. using only $\mathcal{L}_{\text{vel}}(1)$ rather than the accumulative loss (Equation 7). All the methods above are evaluated on TotalCapture dataset in both offline and online settings. We draw the cumulative error curve with respect to time in Figure 5, i.e. the mean Euclidean distance between the estimated and ground truth global positions with predefined error accumulation time ranging from 0 to 10 seconds. This mean distance is calculated by first aligning the root of the estimated and ground truth body in one frame, then recording the distance errors in the following t (ranging from 0 to 10) seconds, and finally averaging the errors over all frames in the test dataset. We should note that there are five subjects in TotalCapture dataset and their leg lengths are acquired by averaging the ground truth lengths of all frames. As shown in Figure 5, our hybrid approach outperforms all the other methods in both offline and online tests. It indicates that our hybrid method, which computes global motions majorly using the supporting foot and deals with the remaining unsolved movements using a network trained on complementary data, makes up the flaws of both branches and thus shows better results. However, this benefit is not significant comparing with the method in Trans-B1 because the supporting foot can be detected in most of the test datasets. So the result is dominated by Trans-B1. In addition, the RNN trained without the accumulative loss performs worse than the one with such a loss term. It demonstrates the effectiveness of the accumulative terms which benefit the estimation of long-range movements. Finally, the errors caused by jittering will be canceled out somehow in the long run, resulting in the flattening trend of the curves.

5.3.3 The network for velocity estimation. We conduct an evaluation of the network structure used for global translation estimation. We compare *1)* a *unidirectional* RNN and *2)* a biRNN using 20 past, 1 current, and 5 future frames on TotalCapture dataset in the online setting. As shown in Table 5, RNN outperforms biRNN on distance errors for both 1 and 5 seconds of accumulation time. Due to the long-term historical dependency of human movements, RNN can utilize all historical information while biRNN can only acquire a limited number of past frames when running in an online manner. On the other hand, RNN only takes one forward pass for a new frame, while biRNN needs to process all the 26 frames in the window forward and backward, which is about 50 times slower. To increase the estimation accuracy, more frames are needed for biRNN, which further reduce the runtime performance. Hence, RNN is a more suitable choice for the global translation estimation.

Table 6. Evaluation of the cross-layer connections of IMU measurements using DIP-IMU and TotalCapture dataset. We compare our original pose estimation pipeline with removing the inertial inputs in Pose-S2 and Pose-S3. We demonstrate the superiority of the cross-layer connections of IMU measurements which help with full joint position estimation and IK solving.

	DIP-IMU		TotalCapture	
	SIP Err (deg)	Mesh Err (cm)	SIP Err (deg)	Mesh Err (cm)
S2 w/o IMUs	17.06 (±7.29)	6.44 (±3.38)	18.51 (±7.31)	6.84 (±3.61)
S3 w/o IMUs	15.66 (±7.53)	6.50 (±3.51)	15.75 (±7.18)	6.83 (±3.67)
Ours	**13.97 (±6.77)**	**5.83 (±3.21)**	**14.95 (±6.90)**	**6.36 (±3.47)**

5.3.4 Cross-layer connections of IMU measurements. We demonstrate the effectiveness of the cross-layer connections of IMU measurements in the pose estimation task, i.e. the impact of the inertial inputs in Pose-S2 and Pose-S3. Specifically, we evaluate two variations of our pose estimation pipeline: *1)* removing the IMU measurements in the input of Pose-S2, i.e. estimating full joint positions only from the leaf joint positions; *2)* removing the IMU measurements in the input of Pose-S3, i.e. solving the IK problem only from full joint positions. As shown in Table 6, knowing leaf joint inertia helps with the estimation of inner joint positions and rotations. We attribute this to the kinematic structure of human bodies, where leaf joints are affected by the movements of inner joints. Thus, by leveraging the leaf joint inertia via skip connections, Pose-S2 and Pose-S3 can achieve better accuracy.

5.4 Live Demo

We implement a real-time live mocap system and show its results in the accompanying video. Concretely, we attach six IMUs onto the corresponding body parts and read the raw inertial measurements via wireless transmission. We calibrate the received data (see Appendix A) and predict the poses and global translations, which are finally used to drive a 3D human model in real-time. From the live demos, we can see that our system handles strong occlusion, wide range motion space, dark and outdoor environment, close interaction, and multiple users with high accuracy and real-time performance.

5.5 Limitations

5.5.1 On the hardware side. In the inertial sensor, a magnetometer is used to measure directions. However, the magnetometer is easily affected by the magnetic field of the environment. As a result, it cannot work in an environment with spatially or temporally varying magnetic fields. Our current sensor uses wireless transmission to deliver signals to our laptop, which does not allow the sensor to be far from the laptop (10 meters at most in our experiments). Hence, to record wide range motions, we still need to carry the laptop and follow the performer. The transmission problem should also be solved to extend the usage of the techniques based on inertial sensors.

5.5.2 On the software side. As a data-driven method, our approach also suffers from the generalization problem. It cannot handle motions that are largely different from the training dataset, e.g., splits and other complex motions that can only be performed by professional dancers and players. Our supporting-foot-based method relies on the assumption that the supporting foot has no motion in the world coordinate space, which is not true for motions like skating and sliding.

6 CONCLUSION

This paper proposes TransPose, a 90-fps motion capture technique based on 6 inertial sensors, which reconstructs full human motions including both body poses and global translations. In this technique, a novel multi-stage method is proposed to estimate body poses based on the idea of reformulating the problem with two intermediate tasks of leaf and full joint position estimations, which leads to a large improvement over the state-of-the-art on accuracy, temporal consistency, and runtime performance. For global translation estimation, a fusion technique is used to leverage a purely data-driven method and a method involving the motion rules of the supporting foot and a kinematic body model. By combining data-driven and motion-rule-driven, the challenging problem of global translation estimation from noisy sparse inertial sensors is solved in real-time for the first time to the best of our knowledge. Extensive experiments with strong occlusion, wide range motion space, dark and outdoor environment, close interaction, and multiple users demonstrate the robustness and accuracy of our technique.

ACKNOWLEDGMENTS

We would like to thank Yinghao Huang and the other DIP authors for providing the SMPL parameters for TotalCapture dataset and the SIP/SOP results. We would also like to thank Associate Professor Yebin Liu for the support on the IMU sensors. We also appreciate Hao Zhang, Dong Yang, Wenbin Lin, and Rui Qin for the extensive help with the live demos. We thank Chengwei Zheng for the proofreading, and the reviewers for their valuable comments. This work was supported by the National Key R&D Program of China 2018YFA0704000, the NSFC (No.61822111, 61727808) and Beijing Natural Science Foundation (JQ19015). Feng Xu is the corresponding author.

REFERENCES

David Aha. 1997. *Lazy Learning*.

Sheldon Andrews, Ivan Huerta, Taku Komura, Leonid Sigal, and Kenny Mitchell. 2016. Real-time Physics-based Motion Capture with Sparse Sensors. 1–10.

E.R. Bachmann, Robert Mcghee, Xiaoping Yun, and Michael Zyda. 2002. Inertial and Magnetic Posture Tracking for Inserting Humans Into Networked Virtual Environments. (01 2002).

Long Chen, Haizhou Ai, Rui Chen, Zijie Zhuang, and Shuang Liu. 2020. Cross-View Tracking for Multi-Human 3D Pose Estimation at Over 100 FPS. 3276–3285.

Michael Del Rosario, Heba Khamis, Phillip Ngo, and Nigel Lovell. 2018. Computationally-Efficient Adaptive Error-State Kalman Filter for Attitude Estimation. *IEEE Sensors Journal* PP (08 2018), 1–1.

Tamar Flash and Neville Hogan. 1985. The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. *The Journal of neuroscience : the official journal of the Society for Neuroscience* 5 (08 1985), 1688–703.

Eric Foxlin. 1996. Inertial Head-Tracker Sensor Fusion by a Complementary Separate-Bias Kalman Filter. 185 – 194, 267.

Andrew Gilbert, Matthew Trumble, Charles Malleson, Adrian Hilton, and John Collomosse. 2018. Fusing Visual and Inertial Sensors with Semantics for 3D Human Pose Estimation. *International Journal of Computer Vision* (09 2018), 1–17.

Ikhsanul Habibie, Weipeng Xu, Dushyant Mehta, Gerard Pons-Moll, and Christian Theobalt. 2019. In the Wild Human Pose Estimation Using Explicit 2D Features and Intermediate 3D Representations. 10897–10906.

Julius Hannink, Thomas Kautz, Cristian Pasluosta, Jochen Klucken, and Bjoern Eskofier. 2016. Sensor-Based Gait Parameter Extraction With Deep Convolutional Neural Networks. *IEEE Journal of Biomedical and Health Informatics* PP (09 2016).

Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Christian Theobalt. 2013. Real-Time Body Tracking with One Depth Camera and Inertial Sensors. *Proceedings of the IEEE International Conference on Computer Vision*, 1105–1112.

Roberto Henschel, Timo Marcard, and Bodo Rosenhahn. 2020. Accurate Long-Term Multiple People Tracking using Video and Body-Worn IMUs. *IEEE Transactions on Image Processing* PP (08 2020), 1–1.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-term Memory. *Neural computation* 9 (12 1997), 1735–80.

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

Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael Black, Otmar Hilliges, and Gerard Pons-Moll. 2018. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics* 37, 1–15.

Diederik Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *International Conference on Learning Representations* (12 2014).

Huajun Liu, Xiaolin Wei, Jinxiang Chai, Inwoo Ha, and Taehyun Rhee. 2011. Realtime human motion control with a small number of inertial sensors. 133–140.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. 2015. SMPL: A Skinned Multi-Person Linear Model. *ACM Trans. Graphics (Proc. SIGGRAPH Asia)* 34, 6 (Oct. 2015), 248:1–248:16.

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)*.

Charles Malleson, John Collomosse, and Adrian Hilton. 2019. Real-Time Multi-person Motion Capture from Multi-view Video and IMUs. *International Journal of Computer Vision* (12 2019).

Charles Malleson, Andrew Gilbert, Matthew Trumble, John Collomosse, Adrian Hilton, and Marco Volino. 2017. Real-Time Full-Body Motion Capture from Video and IMUs. 449–457.

Timo Marcard, Gerard Pons-Moll, and Bodo Rosenhahn. 2016. Human Pose Estimation from Video and IMUs. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 38 (02 2016), 1–1.

Timo Marcard, Bodo Rosenhahn, Michael Black, and Gerard Pons-Moll. 2017. Sparse Inertial Poser: Automatic 3D Human Pose Estimation from Sparse IMUs. *Computer Graphics Forum 36(2), Proceedings of the 38th Annual Conference of the European Association for Computer Graphics (Eurographics), 2017* 36 (02 2017).

Dushyant Mehta, Oleksandr Sotnychenko, Franziska Mueller, Weipeng Xu, Mohamed Elgharib, Pascal Fua, Hans-Peter Seidel, Helge Rhodin, Gerard Pons-Moll, and Christian Theobalt. 2020. XNect: real-time multi-person 3D motion capture with a single RGB camera. *ACM Transactions on Graphics* 39 (07 2020).

Thomas B Moeslund and Erik Granum. 2001. A survey of computer vision-based human motion capture. *Computer vision and image understanding* 81, 3 (2001), 231–268.

Thomas B Moeslund, Adrian Hilton, and Volker Krüger. 2006. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding* 104, 2-3 (2006), 90–126.

Gerard Pons-Moll, Andreas Baak, Juergen Gall, Laura Leal-Taixé, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. 2011. Outdoor human motion capture using inverse kinematics and von mises-fisher sampling. *Proceedings of the IEEE International Conference on Computer Vision* 0, 1243–1250.

Gerard Pons-Moll, Andreas Baak, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bodo Rosenhahn. 2010. Multisensor-Fusion for 3D Full-Body Human Motion Capture. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 663–670.

Qaiser Riaz, Tao Guanhong, Björn Krüger, and Andreas Weber. 2015. Motion Reconstruction Using Very Few Accelerometers and Ground Contacts. *Graphical Models* (04 2015).

Daniel Roetenberg, Hendrik Luinge, Chris Baten, and Peter Veltink. 2005. Compensation of Magnetic Disturbances Improves Inertial and Magnetic Sensing of Human Body Segment Orientation. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on* 13 (10 2005), 395 – 405.

Martin Schepers, Matteo Giuberti, and G. Bellusci. 2018. Xsens MVN: Consistent Tracking of Human Motion Using Inertial Sensing. (03 2018).

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* 45, 11 (1997), 2673–2681.

Loren Schwarz, Diana Mateus, and Nassir Navab. 2009. Discriminative Human Full-Body Pose Estimation from Wearable Inertial Sensor Data. 159–172.

Soshi Shimada, Vladislav Golyanik, Weipeng Xu, and Christian Theobalt. 2020. PhysCap: physically plausible monocular 3D motion capture in real time. *ACM Transactions*

on Graphics 39 (11 2020), 1–16.
Ronit Slyper and Jessica Hodgins. 2008. Action Capture with Accelerometers. *ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 193–199.
Jochen Tautges, Arno Zinke, Björn Krüger, Jan Baumann, Andreas Weber, Thomas Helten, Meinard Müller, Hans-Peter Seidel, and Bernhard Eberhardt. 2011. Motion Reconstruction Using Sparse Accelerometer Data. *ACM Transactions on Graphics* 30 (05 2011), 18.
Denis Tome, Matteo Toso, Lourdes Agapito, and Chris Russell. 2018. Rethinking Pose in 3D: Multi-stage Refinement and Recovery for Markerless Motion Capture. 474–483.
Matthew Trumble, Andrew Gilbert, Adrian Hilton, and John Collomosse. 2016. Deep Convolutional Networks for Marker-less Human Pose Estimation from Multiple Views. 1–9.
Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. 2017. Total Capture: 3D Human Pose Estimation Fusing Video and Inertial Sensors.
Rachel Vitali, Ryan McGinnis, and Noel Perkins. 2020. Robust Error-State Kalman Filter for Estimating IMU Orientation. *IEEE Sensors Journal* (10 2020).
Daniel Vlasic, Rolf Adelsberger, Giovanni Vannucci, John Barnwell, Markus Gross, Wojciech Matusik, and Jovan Popovic. 2007. Practical motion capture in everyday surroundings. *ACM Trans. Graph.* 26 (07 2007), 35.
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. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part X (Lecture Notes in Computer Science)*, Vittorio Ferrari, Martial Hebert, Cristian Sminchisescu, and Yair Weiss (Eds.), Vol. 11214. 614–631.
Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. 2019. Monocular Total Capture: Posing Face, Body, and Hands in the Wild. 10957–10966.
Lan Xu, Lu Fang, Wei Cheng, Kaiwen Guo, Guyue Zhou, Qionghai Dai, and Yebin Liu. 2016. FlyCap: Markerless Motion Capture Using Multiple Autonomous Flying Cameras. *IEEE Transactions on Visualization and Computer Graphics* PP (10 2016).
Zhe Zhang, Chunyu Wang, Wenhu Qin, and Wenjun Zeng. 2020. Fusing Wearable IMUs With Multi-View Images for Human Pose Estimation: A Geometric Approach. 2197–2206.
Zerong Zheng, Yu Tao, Hao Li, Kaiwen Guo, Qionghai Dai, Lu Fang, and Yebin Liu. 2018. *HybridFusion: Real-Time Performance Capture Using a Single Depth Sensor and Sparse IMUs: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part IX.* 389–406.
Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2018. On the Continuity of Rotation Representations in Neural Networks. *CoRR* abs/1812.07035 (2018). arXiv:1812.07035
Yuxiao Zhou, Marc Habermann, Ikhsanul Habibie, Ayush Tewari, Christian Theobalt, and Feng Xu. 2020a. Monocular Real-time Full Body Capture with Inter-part Correlations.
Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. 2020b. Monocular Real-Time Hand Shape and Motion Capture Using Multi-Modal Data. 5345–5354.
Yuliang Zou, Jimei Yang, Duygu Ceylan, Jianming Zhang, Federico Perazzi, and Jia-Bin Huang. 2020. Reducing Footskate in Human Motion Reconstruction with Ground Contact Constraints. 448–457.

A SENSOR PREPROCESSING

Since each inertial sensor has its own coordinate system, we need to *1)* firstly transform the raw inertial measurements into the same reference frame, which is referred to as *calibration*, and then *2)* transform the leaf joint inertia into the root's space and rescale it to a suitable size for the network input, which is referred to as *normalization*. The sensors can be placed with arbitrary rotations during setup, and our method automatically computes the transition matrices for each sensor before capturing the motion. This process requires the subject to keep in T-pose for a few seconds. In this section, we explain the details of the sensor preprocessing in our method, including the calibration (Section A.1) and the normalization (Section A.2).

A.1 Calibration

An inertial measurement unit (IMU) outputs acceleration data relative to the sensor coordinate frame F^S and orientation data relative to the global inertial coordinate frame F^I. We define the coordinate frame of the SMPL [Loper et al. 2015] body model as F^M, and the basis matrix of F^S, F^I, F^M as $\boldsymbol{B}^S, \boldsymbol{B}^I, \boldsymbol{B}^M$ respectively, where each consists of three column basis vectors. Before capturing motions, we firstly put an IMU with the axes of its sensor coordinate frame F^S aligned with the corresponding axes of the SMPL coordinate frame F^M, i.e. to place the IMU with its x-axis left, y-axis up and z-axis forward in the real world. Then, the orientation measurement $\boldsymbol{P}^{IM}$ can be regarded as the transition matrix from F^I to F^M:

$$\boldsymbol{B}^M = \boldsymbol{B}^I \boldsymbol{P}^{IM}. \tag{15}$$

Next, we put each IMU onto the corresponding body part with arbitrary orientations and keep still in a predefined pose (such as the T-pose) with known leaf-joint and pelvis orientations $\boldsymbol{R}_M^{\text{bone}[i]} (i = 0, 1, \cdots, 5)$ (relative to F^M) for several seconds. We read the IMU measurements and calculate the average acceleration (relative to F^S) and orientation (relative to F^I) of each sensor as $\boldsymbol{a}_S^{\text{sensor}[i]}$ and $\boldsymbol{R}_I^{\text{sensor}[i]}$ respectively. We represent the rotation offsets between the sensors and the corresponding bones as $\boldsymbol{R}_I^{\text{offset}[i]}$ due to the arbitrarily oriented placement of the sensors and the assumption that the angles between muscles and bones are constants. We then have:

$$\boldsymbol{R}_I^{\text{bone}[i]} = \boldsymbol{R}_I^{\text{sensor}[i]} \boldsymbol{R}_I^{\text{offset}[i]}, \tag{16}$$

where $\boldsymbol{R}_I^{\text{bone}[i]}$ is the absolute orientation of bone i in the coordinate frame F^I. For any given pose, the absolute orientation of bone i is equivalent in two coordinate frames F^M, F^I:

$$\boldsymbol{B}^M \boldsymbol{R}_M^{\text{bone}[i]} = \boldsymbol{B}^I \boldsymbol{R}_I^{\text{bone}[i]}. \tag{17}$$

Combining Equation 15, 16, and 17, we can get:

$$\boldsymbol{R}_I^{\text{offset}[i]} = (\boldsymbol{R}_I^{\text{sensor}[i]})^{-1} \boldsymbol{P}^{IM} \boldsymbol{R}_M^{\text{bone}[i]}. \tag{18}$$

For accelerations, we first transform the measurements in the sensor local frame F^S to the global inertial frame F^I as:

$$\boldsymbol{a}_I^{\text{bone}[i]} = \boldsymbol{a}_I^{\text{sensor}[i]} = \boldsymbol{R}_I^{\text{sensor}[i]} \boldsymbol{a}_S^{\text{sensor}[i]}, \tag{19}$$

where $\boldsymbol{a}_I^{\text{bone}[i]}$ and $\boldsymbol{a}_I^{\text{sensor}[i]}$ are the accelerations of bone i and the IMU attached to it in F^I, respectively. They are equal because we assume the IMUs do not move relatively once mounted. Due to the sensor error and the inaccuracy of $\boldsymbol{P}^{IM}$, there is a constant offset in the global acceleration. Thus, we add an offset $\boldsymbol{a}_M^{\text{offset}[i]}$ to the coordinate frame F^M as:

$$\boldsymbol{B}^M (\boldsymbol{a}_M^{\text{bone}[i]} + \boldsymbol{a}_M^{\text{offset}[i]}) = \boldsymbol{B}^I \boldsymbol{a}_I^{\text{bone}[i]}. \tag{20}$$

Since the subject keeps still during calibration, $\boldsymbol{a}_M^{\text{bone}[i]} = \boldsymbol{0}$. We then combine Equation 15, 19, and 20 to get:

$$\boldsymbol{a}_M^{\text{offset}[i]} = (\boldsymbol{P}^{IM})^{-1} \boldsymbol{R}_I^{\text{sensor}[i]} \boldsymbol{a}_S^{\text{sensor}[i]}. \tag{21}$$

Leveraging the known $\boldsymbol{P}^{IM}$, $\boldsymbol{R}_I^{\text{offset}[i]}$, and $\boldsymbol{a}_M^{\text{offset}[i]}$ estimated in the pre-computation step, we can now perform motion capture. Specifically, we calculate $\boldsymbol{R}_M^{\text{bone}[i]}$ and $\boldsymbol{a}_M^{\text{bone}[i]}$ (denoted as $\bar{\boldsymbol{R}}_i$ and $\bar{\boldsymbol{a}}_i$ in short) per frame, and feed them into our model after the normalization process described in Section A.2.

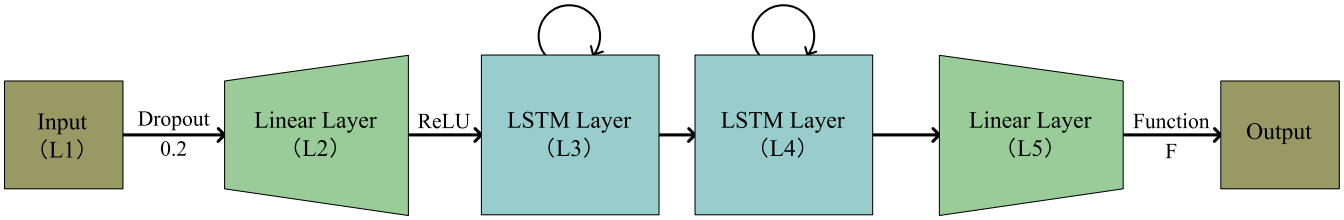

Fig. 6. Detailed structures for each network of our pipeline. "L1" · · · "L5" represent the output widths of the layers. "F" represents the additional function applied after the last linear layer. Values of these parameters for each network are shown in Table 7.

Table 7. Details of each network in our pipeline. "L1" · · · "L5" are the output widths of the corresponding layers as shown in Figure 6. "F" is the function applied after the output layer. "Bidirectional" represents whether the two LSTM layers are bidirectional.

	L1	L2	L3	L4	L5	F	Bidirectional
Pose-S1	72	256	256	256	15	-	true
Pose-S2	87	64	64	64	69	-	true
Pose-S3	141	128	128	128	90	-	true
Trans-B1	87	64	64	64	2	sigmoid	true
Trans-B2	141	256	256	256	3	-	false

A.2 Normalization

We explain our normalization process in this section. For each frame, the raw inputs are accelerations $[\tilde{\boldsymbol{a}}_{\text{root}}, \tilde{\boldsymbol{a}}_{\text{lleg}}, \tilde{\boldsymbol{a}}_{\text{rleg}}, \tilde{\boldsymbol{a}}_{\text{head}}, \tilde{\boldsymbol{a}}_{\text{larm}}, \tilde{\boldsymbol{a}}_{\text{rarm}}] \in \mathbb{R}^{3\times6}$ and rotations $[\tilde{\boldsymbol{R}}_{\text{root}}, \tilde{\boldsymbol{R}}_{\text{lleg}}, \tilde{\boldsymbol{R}}_{\text{rleg}}, \tilde{\boldsymbol{R}}_{\text{head}}, \tilde{\boldsymbol{R}}_{\text{larm}}, \tilde{\boldsymbol{R}}_{\text{rarm}}] \in \mathbb{R}^{3\times3\times6}$ measured by the IMUs. We transfer these measurements from their own coordinate frames to the SMPL reference frame, obtaining $\bar{\boldsymbol{a}}$ and $\bar{\boldsymbol{R}}$ as described in Section A.1. Then, we align leaf joint inertial measurements with respect to the root as:

$$\boldsymbol{a}_{\text{leaf}} = \bar{\boldsymbol{R}}_{\text{root}}^{-1}(\bar{\boldsymbol{a}}_{\text{leaf}} - \bar{\boldsymbol{a}}_{\text{root}}), \tag{22}$$

$$\boldsymbol{R}_{\text{leaf}} = \bar{\boldsymbol{R}}_{\text{root}}^{-1}\bar{\boldsymbol{R}}_{\text{leaf}}, \tag{23}$$

and normalize root joint measurements as:

$$\boldsymbol{a}_{\text{root}} = \bar{\boldsymbol{R}}_{\text{root}}^{-1}\bar{\boldsymbol{a}}_{\text{root}}, \tag{24}$$

$$\boldsymbol{R}_{\text{root}} = \bar{\boldsymbol{R}}_{\text{root}}. \tag{25}$$

Finally, we rescale the accelerations to a suitable size for neural networks.

B NETWORK DETAILS

We introduce the detailed structures for all the networks in our pipeline here. As shown in Figure 6, each network contains a direct 20% dropout on the input, followed by a linear and a ReLU operation that map the input to the dimension of the LSTM layers. Then two LSTM layers with the same width process the data and a linear operation maps it to the output dimension. Particularly, we apply a sigmoid function on the output in the network of Trans-B1 that estimates foot-ground contact probabilities. The output widths of the layers for each network are listed in Table 7.

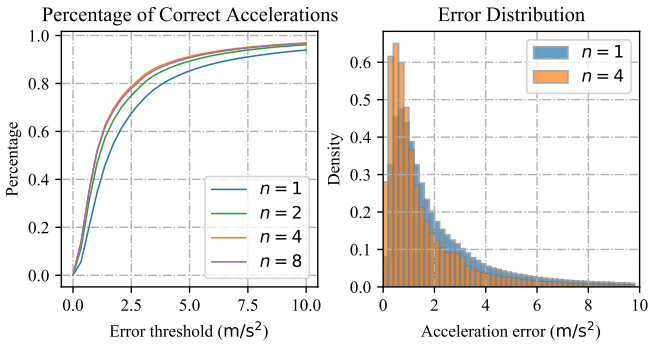

Fig. 7. Influences of the smoothing factor n in Equation 11. We evaluate the *percentage of correct synthetic accelerations* on TotalCapture dataset using different smoothing factors n and demonstrate that a value of $n = 4$ produces accelerations with the closest distribution to the measurements of real sensors.

C ACCELERATION SYNTHESIS

We need to synthesize leaf joint inertia for AMASS dataset as stated in Section 4.2. Here we show the influence of the smoothing factor n in Equation 11 and demonstrate our advantages to use $n = 4$ over other works [Huang et al. 2018; Malleson et al. 2019, 2017; Marcard et al. 2017] which assume $n = 1$. In order to demonstrate the purpose of the smoothing factor, we re-synthesize accelerations for TotalCapture dataset using the ground truth poses and translations, and compare the synthetic accelerations with real IMU measurements. We evaluate the *percentage of correct accelerations* under some selected error thresholds and plot the curves and the error distributions in Figure 7. The results show that a smoothing factor of $n = 4$ produces accelerations most similar to real sensor measurements, while the classical finite difference method where $n = 1$ produces less accurate results due to the jitters of leaf joints. Therefore, we take $n = 4$ in our work.