Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time

YINGHAO HUANG∗, Max Planck Institute for Intelligent Systems, Tübingen, Germany
MANUEL KAUFMANN∗, Advanced Interactive Technologies Lab, ETH Zürich, Switzerland
EMRE AKSAN, Advanced Interactive Technologies Lab, ETH Zürich, Switzerland
MICHAEL J. BLACK, Max Planck Institute for Intelligent Systems, Tübingen, Germany
OTMAR HILLIGES, Advanced Interactive Technologies Lab, ETH Zürich, Switzerland
GERARD PONS-MOLL, Max Planck Institute for Informatics, Saarbrücken, Germany

Fig. 1. We demonstrate a novel learning-based method for reconstructing 3D human pose in real-time using only six body-worn IMUs. Our method learns from a large synthetic dataset and at runtime predicts pose parameters from real IMU inputs in real-time while only requiring minimal user instrumentation. This brings 3D motion capture to new scenarios that are difficult for camera-based methods such as heavy occlusions and fast motion.

We demonstrate a novel deep neural network capable of reconstructing human full body pose in real-time from 6 Inertial Measurement Units (IMUs) worn on the user’s body. In doing so, we address several difficult challenges. First, the problem is severely under-constrained as multiple pose parameters produce the same IMU orientations. Second, capturing IMU data in conjunction with ground-truth poses is expensive and difficult to do in many target application scenarios (e.g., outdoors). Third, modeling temporal dependencies through non-linear optimization has proven effective in prior work but makes real-time prediction infeasible. To address this important limitation, we learn the temporal pose priors using deep learning. To learn from sufficient data, we synthesize IMU data from motion capture datasets. A bi-directional RNN architecture leverages past and future information that is available at training time. At test time, we deploy the network in a sliding window fashion, retaining real time capabilities. To evaluate our method, we recorded DIP-IMU, a dataset consisting of 10 subjects wearing 17 IMUs for validation in 64 sequences with 330,000 time instants; this constitutes the largest IMU dataset publicly available. We quantitatively evaluate our approach on multiple datasets and show results from a real-time implementation. DIP-IMU and the code are available for research purposes.

CCS Concepts: • Computing methodologies → Motion capture;
Additional Key Words and Phrases: Real-Time, IMU, Deep Learning, RNN

1 INTRODUCTION

Many applications such as gaming, bio-mechanical analysis, and emerging human-computer interaction paradigms such as Virtual and Augmented Reality (VR/AR) require a means to capture a user’s 3D skeletal configuration. Such applications impose three challenging constraints on pose reconstruction: (i) it must operate in real-time, (ii) it should work in everyday settings such as sitting at a desk, at home, or outdoors, and (iii) it should be minimally invasive in terms of user instrumentation.

Most commonly, the task of recording human motion is achieved via commercial motion capture (Mocap) systems such as Vicon, but...
these require expensive infrastructure and markers placed on the user. Marker-less multi-camera approaches are becoming more accurate and can sometimes provide dense surface reconstructions but also require controlled camera setups and can be computationally very expensive. Recently, the use of a single RGB or RGB-D camera for human pose estimation has become feasible and remains a very active area of research in computer vision [Alldieck et al. 2018; Kanazawa et al. 2018; Mehta et al. 2018; Omran et al. 2018; Tao et al. 2018]. Single camera methods are still less accurate than multi-view methods but more importantly, like all vision-based methods, they require an external camera with the full body visible in the image. This limitation presents a practical barrier for many applications, in particular those where heavy occlusions can be expected such as sitting at a desk or where the user moves around, such as outdoors.

Mounting sensors directly on the user’s body overcomes the need for direct line-of-sight. The most prominent choice for pose reconstruction tasks are inertial measurement units (IMUs), which can record orientation and acceleration, are small, and can hence be easily worn on the body. Commercial systems rely on dense placement of IMUs, which fully constrain the pose space, to attain accurate skeletal reconstructions [Roetenberg et al. 2007]. Placing 17 or more sensors on the body can be intrusive, time consuming, and prone to errors such as swapping sensors during mounting. However, it has been recently demonstrated, that full body pose can be recovered from a small set of sensors (6 IMUs) [von Marcard et al. 2017], albeit with a heavy computational cost, requiring offline optimization of a non-convex problem over the entire sequence; this can take hours per recording.

Given that emerging consumer products such as smart-watches, fitness trackers and smart-glasses (e.g., HoloLens, Google Glass) already integrate IMUs, reconstructing 3D body pose from a small set of sensors in real-time would enable many applications. In this paper we introduce DIP: Deep Inertial Poser, the first deep learning method capable of estimating 3D human body pose from only 6 IMUs in real time. Learning a function that predicts accurate poses from a sparse set of orientation and acceleration measurements alone is a challenging task because (i) the whole pose is not observable from just 6 measurements, (ii) previous work has shown that long-range temporal information plays an important role [von Marcard et al. 2017], and (iii) capturing large datasets for training is time-consuming and expensive.

To overcome these issues we leverage the following observations and insights: (i) Large datasets of human mocap such as CMU [2008] or the H3.6M [2014] exist. Specifically, we leverage AMASS [MoS 2018], a large collection of MoCap datasets with data provided in the form of SMPL [Loper et al. 2015] model parameters. We leveraged this to synthesize IMU data. Specifically, we place virtual sensors on the SMPL mesh and use the mocap sequences to obtain virtual orientations via forward kinematics, and accelerations via finite differences. We leverage this synthetic data for training a deep neural network model. (ii) To model long-range temporal dependencies, we leverage recurrent neural networks (RNNs) to map from orientations and accelerations to SMPL parameters. However, making full use of acceleration information proved to be difficult. This leads to systematic errors for ambiguous mappings between sensor orientation (measured at the lower-extremities) and pose. In particular knee and arm bending is problematic. To alleviate this issue we introduce a novel loss-term that forces the network to reconstruct accelerations during training, which preserves information throughout the network stack and leads to better performance at test time. (iii) Offline approaches for the same task leverage both past and future information [von Marcard et al. 2017]. We propose an extension of our architecture that leverages bi-directional RNNs to further improve the reconstruction quality. At training time, this architecture has access to the same information as [von Marcard et al. 2017], and propagates information from the past to the future and vice versa. To retain the real-time regime, we deploy this architecture at test time in a sliding-window fashion. We experimentally show that only 5 future frames are sufficient for high-quality predictions, while only incurring a modest latency penalty of 85ms.

Using only synthetic data for training already provides decent performance. However, real IMU data contains noise and drift. To close the gap between synthetic and real data distributions, we fine-tune our model using a newly created DIP-IMU dataset containing approximately 90 minutes of real IMU data.

We experimentally evaluate DIP using TotalCapture [Trumble et al. 2017], a benchmark dataset including IMU data and reference ("ground truth") poses, and on the DIP-IMU dataset. We show that DIP achieves an accuracy of 15.85° angular error, which is lower than the competing offline approach, SIP [von Marcard et al. 2017]. This is significant as our method runs in real-time, whereas SIP requires the full motion sequence as input. To further demonstrate the real-time capabilities of DIP, we integrate our approach in a simple VR proof-of-concept demonstrator; we take raw IMU data as input and our pipeline predicts full body poses, without any temporal filtering or post-hoc processing. The resulting poses are then visualized via Unity. In summary, DIP occupies a unique place in the pose estimation literature as it satisfies all three aforementioned constraints: it is real time, minimally intrusive, and works in everyday places Fig. 1.

2 RELATED WORK

The literature on human pose estimation from images and video is vast. Here we briefly review camera-based methods, those that integrate multiple sensor signals using tracking and optimization, and learning based methods that recover pose from sparse sensors.

2.1 Camera-based motion capture:

Commercial camera-based Mocap solutions require that subjects wear many markers and depend on multiple calibrated cameras mounted in the environment. To overcome these constraints, much research has been devoted to developing marker-less approaches from multiple cameras (cf. [Sarafianos et al. 2016]). Often such methods require offline processing to achieve high quality results [Ballan et al. 2012; Bregler and Malik 1998; Huang et al. 2017; Starck and Hilton 2003] but recently, real-time approaches [de Aguiar et al. 2008; Elhayek et al. 2017; Rhodin et al. 2015; Stoll et al. 2011] have been proposed. Such approaches typically fit a skeletal model to image data or represent the human as a collection of Gaussians [Stoll et al.
which makes the pose reconstruction problem much harder but Tompson et al. 2014; Toshev and Szegedy 2014; Wei et al. 2016]. How-

subject (using standard skeletal models). However, 17 IMUs are very
drift, the 17 IMU orientations completely define the full pose of the 

Filter. Assuming the measurements are noise-free and contain no 

gyroscopes and magnetometers, fused together using a Kalman

berg et al. 2007] use 17 IMUs equipped with 3D accelerometers, 

used in settings where traditional Mocap is not possible.

offers the the potential of an infrastructure-free system that can be 

intrusive for the user, but future miniature cameras could make 

the approach more practical; such visual data from body-mounted
cameras could be complementary to our system. Monocular depth 
cameras provide additional information and have been shown to 
aid robust skeletal tracking [Ganapathi et al. 2012; Pons-Moll et al. 2015b; Shotton et al. 2013; Taylor et al. 2012; Wei et al. 2012] and enable dense surface reconstruction even under non-rigid deformation [Dou et al. 2016; Newcombe et al. 2015; Tao et al. 2018; Zollhöfer et al. 2014]. Specialized scanners have been used to capture high-fidelity dense surface reconstructions of humans [Collet et al. 2015; Pons-Moll et al. 2017, 2015a].

In contrast to camera-based work, our approach does not rely on calibrated cameras mounted in the environment. Instead, we leverage a sparse set of orientation and acceleration measurements, which makes the pose reconstruction problem much harder but offers the the potential of an infrastructure-free system that can be used in settings where traditional Mocap is not possible.

2.2 Optimization based sensor fusion methods

Inertial trackers. Commercial inertial tracking solutions [Roetensberg et al. 2007] use 17 IMUs equipped with 3D accelerometers, gyroscopes and magnetometers, fused together using a Kalman Filter. Assuming the measurements are noise-free and contain no drift, the 17 IMU orientations completely define the full pose of the subject (using standard skeletal models). However, 17 IMUs are very

intrusive for the subject, long setup times are required, and errors such as placing a sensor on the wrong limb are common. To compen-
sate for IMU drift, the pioneering work of Vlasic et al. [2007] uses 
a custom system with 18 boards equipped with acoustic distance 
sensors and IMUs. However, the system is also very intrusive and 
difficult to reproduce.

Video-inertial trackers. Sparse IMUs have also been combined with video input [Malleson et al. 2017; Pons-Moll et al. 2011, 2010; von Marcard et al. 2016], or with sparse optical markers [Andrews et al. 2016] to constrain the problem. Similarly, sparse IMUs have been combined with a depth camera [Helten et al. 2013]; IMUs are only used to query similar poses in a database, which constrain the depth-based body tracker. While powerful, hybrid approaches that use video suffer from the same drawbacks as pure camera-based methods including occlusions and restricted recording volumes. Recent work uses a single moving camera and IMUs to estimate the 3D pose of multiple people in natural scenes [von Marcard et al. 2018], but the approach requires a camera that follows the subjects around.

Optimization from sparse IMUs. Von Marcard et al. [2017] compute accurate 3D poses using only 6 IMUs. They take a generative approach and place synthetic IMUs on the SMPL body model [Loper et al. 2015]. They solve for the sequence of SMPL poses that pro- duces synthetic IMU measurements that best match the observed sequence of real measurements by optimizing over the entire se-
quence. Like [von Marcard et al. 2017] we also use 6 IMUs to recover full body pose, and we also leverage SMPL. Our approach is however conceptually very different: instead of relying on computationally expensive offline optimization, we learn a direct mapping from sen-

sor data to the full pose of SMPL, resulting in real-time performance and good accuracy despite using only 6 IMUs.

2.3 Learning based methods

Sparse accelerometers and markers. An alternative to sensor fu-
sion and optimization is to learn the mapping from sensors to full body pose. Human pose is reconstructed from 5 accelerometers by retrieving pre-recorded poses with similar accelerations from a data-

base [Slyper and Hodgins 2008; Tautges et al. 2011]. The mapping from acceleration alone to position is however very difficult to learn,
Motion sensors. Alternatively, position and orientation can be obtained from motion sensors based on inertial-ultrasonic technology. Full pose can be regressed from 6 such sensors [Liu et al. 2011], which provide global orientation and position. While global position sensors greatly simplify the inverse problem since measurements are always relative to a static base-station; consequently, capture is restricted to a pre-determined recording volume. Furthermore, such sensors rely on a hybrid inertial-ultrasonic technology, which is mostly used for specialized military applications. Our method uses only commercially available IMUs—providing orientation and acceleration but no position, and does not require a fixed base-station.

Sparse IMUs. Learning methods using sparse IMUs as input have also been proposed [Schwarz et al. 2009], where full pose is regressed using Gaussian Processes. The models are trained on specific movements of individual users for each activity of interest, which greatly limits its applicability. Furthermore, Gaussian Processes scale poorly with the number of training samples. Generalization to new subjects and un-constrained motion patterns is not demonstrated.

Locomotion and gait. IMUs are often used for gait analysis and activity recognition, recently in combination with deep learning approaches (cf. [Wang et al. 2017]), for example to extract gait parameters via a CNN for medical purposes [Hannink et al. 2016]. Prediction of locomotion has been shown using a single IMU [Mousas 2017] using a hierarchical Hidden Markov Model. Deep learning has been used to produce locomotion of avatars that adapt to irregular terrain [Holden et al. 2017], or that avoid obstacles and follow a trajectory [Peng et al. 2017]. These approaches are suited to cyclic motion patterns, where cycle phase plays a central role.

In summary, existing learning methods either rely on global joint positions as input—which requires external cameras or specialized technology—or are restricted to pre-defined motion patterns. DIP is the first deep-learning method capable of reconstructing full-body motion from a sparse set of sensors in real time.

3 METHOD OVERVIEW

Our goal is to reconstruct articulated human motion in unconstrained settings from a sparse set of IMUs (6 sensors) in real-time. This problem is extremely difficult since many parts of the body are not directly observable from the sensor data alone. To overcome this problem, we leverage a state-of-the-art statistical model of human shape and pose, and regress its parameters using a deep recurrent neural network (RNN). In our implementation, we use the SMPL model [Loper et al. 2015] both to synthesize training data and as the output target of the LSTM architecture. This approach ensures that sufficient data is available for training, and encourages that the resulting predictions lie close to the subspace spanned by natural human motion. We now briefly introduce the most salient aspects of the data generation process (Sec. 3.1, Sec. 3.2), the accumulated dataset used for training (Sec. 3.3), and our proposed network architecture (Sec. 3.4). An overview of the entire pipeline can be found in Fig. 2.

3.1 Background: SMPL body model

SMPL is a parametrized model of 3D human body shape and pose that takes 72 pose, and 10 shape, parameters, $\theta$ and $\beta$ respectively, and returns a mesh with $N = 6890$ vertices. Shape and pose deformations are applied to a base template, $T_0$, that corresponds to the mean shape a training 3D scans. We summarize [Loper et al. 2015] here for completeness:

$$M(\beta, \theta) = W(T(\beta, \theta), J(\beta, \theta), W)$$

(1)

$$T(\beta, \theta) = T_0 + B_\theta(\beta) + B_p(\theta),$$

(2)

where $W$ is a linear blend-skinning (LBS) function applied to the template mesh in the rest pose, to which pose- and shape-dependent deformations, $B_\theta(\beta)$ and $B_p(\theta)$, are added. The resulting mesh is then posed using LBS with rotations about the joints, $J(\beta)$, which depend on body shape. The shape-dependent deformations model subject identity while the pose-dependent ones correct LBS artifacts and capture deformations of the body with pose.

3.2 Synthesizing training data

Our approach is learning-based and hence requires a sufficiently large dataset for training. Compared to the camera or marker-based cases, there are very few publicly available datasets including IMU data and ground-truth poses. To the best of our knowledge the only such dataset is TotalCapture [Trumble et al. 2017], including typical day-to-day activities. The dataset contains synchronized IMU, mocap and RGB data. However, due to the limited size and types of activities, models trained on this dataset alone do not generalize well, e.g., common interaction tasks in VR/AR such as reaching for and selecting objects in a seated position are not represented at all.

However, given the capability of fitting SMPL parameters to inputs of different modalities (17 IMUs, marker data), it becomes feasible to generate a much larger and more comprehensive training dataset by synthesizing pairs of IMU measurements and corresponding SMPL parameters from a variety of input datasets.

To obtain synthetic IMU training data, we place virtual sensors on the SMPL mesh surface. The orientation readings are then directly retrieved using forward kinematics, whereas we obtain accelerations via finite differences. Assuming the position of a virtual IMU is $p_t$ for time $t$, and the time interval between two consecutive frames is $dt$, the simulated acceleration is computed as:

$$a_t = \frac{p_t - p_{t-1} + p_{t+1} - 2 \cdot p_t}{dt^2},$$

(3)

3.3 Datasets

Our final training data is a collection of pairs of synthetic IMU sensor readings and corresponding SMPL pose parameters. We use a

3http://www.intersense.com/pages/20/14
Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time

3.4 Deep Inertial Poser (DIP)

Given the training dataset $D = \{(x^j, y^j)\}_{j=1}^N$ consisting of $N$ training sequences, our task is to learn a function $f : x \rightarrow y$ that predicts SMPL pose parameters $y$ from sparse IMU inputs $x$ (acceleration and orientation for each sensor). This mapping poses a severely under-constrained problem, since there exist potentially many SMPL poses corresponding to the same IMU inputs. For example, consider the case of knee raises while standing in place. Here the orientation data will remain mostly unchanged and only transient accelerations will be recorded throughout the sequence. This observation led to the use of strong priors and a global optimization formulation in [von Marcard et al. 2017], consisting of orientation, acceleration and anthropometric terms. This approach is computationally expensive and offline, with run-times of several minutes to hours depending on the sequence length. To overcome this limitation, we adopt a data-driven approach and model the mapping with neural networks by using a log-likelihood loss function, implicitly learning the space of valid poses from sequences directly.

Both IMU inputs and corresponding SMPL pose targets are highly structured and exhibit strong correlations due to the articulated nature of human motion. Recurrent neural networks are capable of modeling temporal data and have been previously used in modeling of human motion, typically attempting to predict the next frame of a sequence [Fragkiadaki et al. 2015; Ghosh et al. 2017; Martinez et al. 2017]. Although we use a different input modality, our model needs to learn similar motion dynamics. In order to exploit temporal coherency in the motion sequence we use recurrent neural networks (RNN) and bi-directional recurrent neural networks (BiRNN) [Schuster and Paliwal 1997]—with long short-term memory (LSTM) cells [Hochreiter and Schmidhuber 1997].

RNNs summarize the entire motion history via a fixed-length hidden state vector and require the current input $x_t$ in order to predict the pose vector $y_t$. While standard RNNs are sufficient in many real-time applications, we experimentally found that having access to both future and past information significantly improves the predicted pose parameters. BiRNNs take all temporal information into account by running two cells in the forward and backward directions, respectively. Compared to RNNs (i.e., unidirectional),
BiRNNs exhibit better qualitative and quantitative results by accessing the whole input sequence. This is in-line with the findings of [von Marcard et al. 2017] where optimizing over the entire sequence was found to be necessary.

Before arriving at the proposed bidirectional architecture, we experimented with simple feed-forward networks and WaveNet [van den Oord et al. 2016] variants. We found that these models either perform worse quantitatively or produce unacceptable visual jitter. The appendix A contains more details on these experiments. We assume that RNNs can make better use of the inherent temporal properties of the data and hence produce smoother predictions than non-recurrent variants, especially if they have access to future and past information.

While we train our BiRNN model by using all time-steps, it is important to note that at test time, we use only input sub-sequences consisting of past and future frames in a sliding-window fashion. In our real-time processing pipeline, we only permit a short temporal look ahead to keep the latency penalty minimal. Our evaluations show that using only 5 future frames provides the best compromise between performance and latency. Fig. 6 summarizes the impact of window size on the reconstruction quality. We note that in settings with strict low-latency requirements, such as AR, it may be desirable to use no future information at the cost of roughly 1° lower accuracy.

3.4.1 Training with uncertainty. During training, we model target poses \( \mathbf{y}_t \) with a Normal distribution with diagonal covariance and use a standard log-likelihood loss to train our network:

\[
\log(p(\mathbf{y}) = \sum_{t=1}^{T} \log(N(\mathbf{y}_t|\mathbf{\mu}_t, \mathbf{\sigma}_t)),
\]

where \( \mathbf{f} \) stands for either a unidirectional or bidirectional RNN being trained on sequences of \( T \) frames. In other words, our model \( \mathbf{f} \) outputs \( \mathbf{\mu} \) and \( \mathbf{\sigma} \) parameters of a Gaussian distribution at every time step. In-line with the RNN literature, we found that this log-likelihood loss leads to slightly better performance than, for example, mean-squared error (MSE).

3.4.2 Reconstruction of acceleration. The input vector for a single frame, \( \mathbf{x}_t = [\mathbf{o}_t, \mathbf{a}_t] \), contains orientation \( \mathbf{o}_t \) and acceleration \( \mathbf{a}_t \) data as measured by the IMUs. We represent orientations as \( 3 \times 3 \) rotation matrices in the SMPL body frame. Before feeding orientations and accelerations into the model, we normalize them w.r.t. the root sensor (cf. Section 4.2-4.4). This results in 5 input rotation matrices that are all stacked and vectorized into \( \mathbf{o}_t = \text{vec}(\mathbf{R}_1, \ldots, \mathbf{R}_5) \in \mathbb{R}^{45} \). Similarly, the normalized accelerations are stacked into \( \mathbf{a}_t \in \mathbb{R}^{15} \).

The acceleration data \( \mathbf{a}_t \) is inherently noisy and much less stable than the orientations. This issue is further complicated by the subtle differences between real and synthesized accelerations in the training data. In our experiments, we found that different network architectures displayed the tendency to discard most of the acceleration data already at the input level (almost zero weights on the acceleration inputs). For certain motions, the lack of acceleration information causes the model to underestimate flexion and extension of joints. In order to alleviate this problem, we introduce an auxiliary task during training. Our model is asked to reconstruct the input acceleration in addition to pose at training time. This additional loss forces the model to propagate the acceleration information to the upper layers.

Analogous to the main pose task, we model the auxiliary acceleration loss via a Normal distribution with diagonal covariance.

\[
\log(p(a_t)) = \sum_{t=1}^{T} \log(N(\mathbf{a}_t|\mathbf{\mu}_a, \mathbf{\sigma}_a)), \quad (\mathbf{\mu}_a, \mathbf{\sigma}_a)^T_{t=1} = \mathbf{f}(\mathbf{x} = [\mathbf{o}, \mathbf{a}]).
\]

The pose prediction loss Eq. (4) and acceleration reconstruction loss Eq. (5) are complementary to each other and are back-propagated through the architecture with all weights and other network parameters being shared. Only a minimal number of additional trainable network parameters is required to predict \( \mathbf{\mu}_a \) and \( \mathbf{\sigma}_a \) with sufficient accuracy.

We experimentally show that adding the auxiliary acceleration loss improves pose predictions quantitatively.

3.4.3 Regularization. We train on primarily synthetic data. While the data is sufficiently realistic, slight differences relative to real data are unavoidable. As a consequence, we observed indications of overfitting, and testing on real data yielded less accurate and jerky predictions. To counteract overfitting, we regularize models via dropouts directly on the inputs with a keep probability of 0.8, which randomly filters out 20% of the inputs during training. Randomly masking inputs helps the model to better generalize to the real data and to make smoother temporal predictions.

3.4.4 Fine-tuning with real data. To reduce the gap between real and synthetic data further, we fine-tune the pre-trained models, using the training split of the new dataset (see Sec. 4.5). We found fine-tuning particularly effective in situations where specific usage scenarios or motion types were underrepresented in the training data. Hence, this procedure is an effective means of adapting our method to novel situations.

4 IMPLEMENTATION DETAILS

4.1 Network architecture

We implemented our network architecture in TensorFlow [Abadi et al. 2015]. Fig. 12 in the appendix summarizes the architecture details. We used the Adam optimizer [Kingma and Ba 2014] with an initial learning rate of 0.001, which is exponentially decayed with a rate of 0.96 and decay step 2000. In order to alleviate the exploding
Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time

4.2 Sensors and calibration

Sensors. We use Xsens IMU sensors containing 3-axis accelerometers, gyroscopes and magnetometers; the raw sensor readings are in the sensor-local coordinate frame $F^S$. Xsens also provides absolute orientation of each sensor relative to a global inertial frame $F^I$. Specifically, the IMU readings that we use are orientation, provided as a rotation $R^{IS}: F^S \rightarrow F^I$, which maps from the sensor-local frame to the inertial frame, and acceleration which is provided in local sensor coordinates.

Calibration. Before feeding orientation and acceleration to our model, we must transform them to a common body-centric frame, in our case the SMPL body frame $F^B$. Concretely, we must find the map $R^{TI}: F^I \rightarrow F^T$, relating the inertial frame to the SMPL body frame. To this end, we place the head sensor onto the head such that the sensor axes align with the SMPL body frame. Consequently, in this configuration, the mapping from head sensor to SMPL frame $F^T$ is the identity. This allows us to set $R^{TI}$ as the inverse of the orientation $R_{\text{Head}}$ read from the head sensor at calibration time. All IMU readings can then be expressed in the SMPL body frame:

$$R^{IS}_t = R^{TI}_t R^{IS}_{\text{Head}} R^{IS}_0.$$  

Lastly, due to surrounding body tissue, the sensors are offset from the corresponding bones. We denote this constant offset by $R^{BS}: F^S \rightarrow F^B$, where $F^B$ is the respective bone coordinate frame. In the first frame of each sequence, each subject stands in a known straight pose with known bone orientation $R^{BT}_0$, and we compute the per-sensor bone offset as:

$$R^{BS}_t = \text{inv}(R^{TB}_0) R^{TS}_0,$$  

where $\text{inv}(\cdot)$ denotes matrix inverse. This lets us transform the sensor orientations to obtain virtual bone orientations at every frame

$$R^{TB}_t = R^{BS}_t R^{IS}_t,$$  

which we use for training and testing. The interpretation of virtual bone orientations is straightforward: they are the bone orientations as measured by the IMU. The acceleration data is transformed to the SMPL coordinate frame after subtracting gravity, and is denoted as $a$. Calibration only requires the subject to hold a straight pose for a couple of seconds at the beginning of the recording. Fig. 4 provides an overview of the different coordinate frames involved in our calibration process.

4.3 Normalization

For better generalization, the input data should be invariant to the facing direction of the person (e.g. a running motion while the subject is facing north or south should produce the same inputs for our learning model). To this end, we normalize all bone orientations with respect to the root sensor, mounted at the base of the user’s spine. With $R_{\text{Root}}(t)$ denoting the orientation of the root at time step $t$, and $R^{TB}_s(t)$, the orientation of the bone corresponding to sensor $s$ at time step $t$, we compute the normalized orientations and accelerations as follows:

$$\bar{R}^{TB}_s(t) = R^{TB}_s(t) R^{-1}_{\text{Root}}(t),$$  

$$\bar{a}_s(t) = R^{TB}_s(t) (a_s(t) - a_{\text{Root}}(t)).$$

This is performed per time-step, both for training and testing. We also experimented with other normalization schemes, such as (i) normalizing the root only w.r.t. the first frame in each sequence and (ii) removing only the heading. We found no significant advantages in either of these schemes compared to the one proposed here. Please refer to the appendix C for details.

4.4 Inputs and targets

The inputs to DIP are the normalized orientations and accelerations. Using 6 IMUs, the input for one frame,

$$s_t = [\bar{a}_1(t), \bar{a}_2(t), \ldots, \bar{a}_5(t), \bar{a}_6(t)]^T,$$

is a vector of dimension $d = (3 \cdot 3 + 3) \cdot 5 = 60$. We experimented with more compact representations of orientation than rotation matrices, such as exponential maps or quaternions; these performed significantly worse than using rotation matrices directly. Rotation matrices elements are bounded between $[-1, 1]$ which is good for training neural networks. Similarly for the targets $y$, instead of regressing to the pose parameters $\theta$ of SMPL in axis-angle space, we transform them to rotation matrices and regress them directly. This may seem counter-intuitive because the representation is redundant, but we found empirically that performance is better.

4.5 Data collection

To overcome discrepancies between the sensor data characteristics of the synthetic and real data and to complement the activities portrayed in existing Mocap-based datasets, we recorded an additional dataset of real IMU data, which we call DIP-IMU.

We recorded data from 10 subjects (9 male, 1 female) wearing 17 Xsens sensors (see Fig. 4). All the subjects gave informed consent to share their IMU data. To attain ground-truth, we ran SIP [von Marcard et al. 2017] on all 17 sensors. To compensate for different magnetic offsets across IMUs, a heading reset is performed first. The sensors are aligned in a known spatial configuration, after which the heading is reset. Subsequently, the sensors are mounted on the subject and the calibration procedure (cf. Section 4.2) is performed.

https://www.xsens.com/
Participants were then asked to repeatedly carry out motions in five different categories, including controlled motion of the extremities (arms, legs), locomotion, and also more natural full-body activities (e.g., jumping jacks, boxing) and interaction tasks with everyday objects. In total, we captured approximately 90 minutes of additional data resulting in the largest dataset of real IMU data (appendix F).

5 EXPERIMENTS

To assess the proposed method, we performed a variety of quantitative and qualitative evaluations. We compare our method to the offline baselines SIP [von Marcard et al. 2017] and SOP (reduced version of SIP not leveraging accelerations), and perform self-comparisons between the variants of our architecture. Here we distinguish between two distinct settings. First, we compare performance in the offline setting. That is, we use test sequences from existing real datasets (TotalCapture and DIP-IMU). Second, one of the main contributions of our work is the real-time (online) capability of our system. To demonstrate this, we implemented an end-to-end live system, taking IMU data as input and predicting SMPL parameters as output.

5.1 Quantitative evaluation

In this section we show how our approach performs on several test datasets. We report both mean joint angle error, computed as in [von Marcard et al. 2017], and positional error.

5.1.1 Offline evaluation. First, we report results from the offline setting, in which all models have access to the whole sequence. This setting is a fair comparison between our model and the baselines since SIP and SOP solve an optimization problem over the whole sequence. Table 2 summarizes the results with our models performing close to or better than the SIP baseline. The best configuration (BiRNN (Acc+Dropout)) outperforms SIP by more than one degree. Fig. 5 (left) shows the angular error distribution over the entire TotalCapture dataset. The peak is around 8° error.

The combination of dropouts on the inputs and use of the acceleration loss improve both RNN and BiRNN models. Note that, due to its access to the future steps, BiRNNs perform qualitatively better and produce smoother predictions than the uni-directional RNNs.

5.1.2 Fine-tuning on real data. While the techniques shown in the previous section perform reasonably well on TotalCapture, a significant performance drop is evident on DIP-IMU. This is due to the difference in motions in our new dataset and the aforementioned gap between real and synthetic data distributions. However, the results we have analyzed so far stem from models trained without access to the DIP-IMU data and hence have not seen the types of poses and motions contained in DIP-IMU. We now report the results from our best configurations after fine-tuning on the DIP-IMU data. We fine tune the network on the training split and test it on the held-out set. Tables 2 and 3 show results from the offline and online setting. We find a clear performance increase on DIP-IMU, which is now comparable to TotalCapture. This is further illustrated by the error histogram on DIP-IMU before and after fine-tuning (cf. Fig. 5). Note that the performance on TotalCapture decreases only minimally, indicating that no catastrophic forgetting takes place.

Next, we select our best performing configuration and evaluate it in the online setting. We do not evaluate performance of SIP and SOP since these baselines can not run online. Note that now the RNN configurations no longer have access to the entire sequence, but only to past frames in the case of the unidirectional RNN, and to a sliding window of past and a few future frames in the case of the BiRNN. Table 3 indicates that the networks obtain good pose accuracy in the online setting. Notably, the BiRNN with access to 50 past frames even slightly outperforms the offline setting on TotalCapture. This may be due to the accumulation of error in the hidden state of the RNN and the stochasticity of human motion over longer time spans.

In the online setting, the influence of the acceleration loss is most evident. If evaluated on 20 past and 5 future frames on TotalCapture, a BiRNN without acceleration loss performs worse (16.26° ± 13.54°) than one using the acceleration loss (15.88° ± 13.57°). For 50 past and 5 future frames the error increases to 16.10° ± 13.42° (compared to 15.77° ± 13.41°).

5.1.4 Influence of future window length. Our final implementation leverages a BiRNN to learn motion dynamics from the data. At training time, the entire sequence is processed, but at runtime, only a subset of frames is made available to the network. Fig. 6 summarizes the performance of the network as function of how many frames of past and future information are available at test time. We experimentally found that using 5 future and 20 past frames is the best compromise between prediction accuracy and latency penalty; we use this setting in our live system.
Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time

Table 2. Offline evaluation of SOP, SIP, RNN and BiRNN models on TotalCapture [Trumble et al. 2017] and DIP-IMU. Errors reported as joint angle errors in degrees and positional errors in centimeters. Models with Dropout are trained by applying dropout on input sequences. Acc corresponds to acceleration reconstruction loss. SOP, SIP and BiRNN have access to the whole input sequence while RNN models only use inputs from the past. BiRNN (after fine-tuning) is BiRNN (Acc+Dropout) fined-tuned on DIP-IMU using acceleration reconstruction loss and dropout as well.

<table>
<thead>
<tr>
<th></th>
<th>TotalCapture</th>
<th>DIP-IMU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu_{\text{ang}}$ [deg]</td>
<td>$\sigma_{\text{ang}}$ [deg]</td>
</tr>
<tr>
<td>SOP</td>
<td>22.18</td>
<td>17.34</td>
</tr>
<tr>
<td>SIP</td>
<td>16.98</td>
<td>13.26</td>
</tr>
<tr>
<td>RNN (Dropout)</td>
<td>16.83</td>
<td>13.41</td>
</tr>
<tr>
<td>RNN (Acc)</td>
<td>16.07</td>
<td>13.16</td>
</tr>
<tr>
<td>RNN (Acc+Dropout)</td>
<td>16.08</td>
<td>13.46</td>
</tr>
<tr>
<td>BiRNN (Dropout)</td>
<td>15.86</td>
<td>13.12</td>
</tr>
<tr>
<td>BiRNN (Acc)</td>
<td>16.31</td>
<td>12.28</td>
</tr>
<tr>
<td>BiRNN (Acc+Dropout)</td>
<td>15.85</td>
<td>12.87</td>
</tr>
<tr>
<td>BiRNN (after fine-tuning)</td>
<td>16.84</td>
<td>13.22</td>
</tr>
</tbody>
</table>

Table 3. Online evaluation of BiRNN models on TotalCapture [Trumble et al. 2017] and DIP-IMU. We select the best performing model from our offline evaluation, i.e., (Acc+Dropout). Numbers in brackets ($x$, $y$) mean that this model is evaluated in online mode using $x$ past and $y$ future frames. (fine-tuning) means that the model was fine-tuned on DIP-IMU.

<table>
<thead>
<tr>
<th></th>
<th>TotalCapture</th>
<th>DIP-IMU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$\mu_{\text{ang}}$ [deg]</td>
<td>$\sigma_{\text{ang}}$ [deg]</td>
</tr>
<tr>
<td>BiRNN (20, 5)</td>
<td>15.88</td>
<td>13.57</td>
</tr>
<tr>
<td>BiRNN (50, 5)</td>
<td>15.77</td>
<td>13.41</td>
</tr>
<tr>
<td>BiRNN (fine-tuning)</td>
<td>16.84</td>
<td>13.22</td>
</tr>
<tr>
<td>BiRNN (20, 5) (fine-tuning)</td>
<td>16.90</td>
<td>13.83</td>
</tr>
<tr>
<td>BiRNN (50, 5) (fine-tuning)</td>
<td>16.74</td>
<td>13.64</td>
</tr>
</tbody>
</table>

5.2 Qualitative evaluation

We now further assess our approach via qualitative evaluations. Based on the above quantitative results, we only report results from our best model (BiRNN). We use the model in offline mode to produce the results in this section. Section 5.3 discusses the results when using it in online mode. Please see the accompanying video.

5.2.1 Playground (real). First, we compare to SIP and SOP on the Playground dataset [von Marcard et al. 2017]. Playground is challenging because it is captured outdoors and contains uncommon poses and motions. Since the dataset contains no ground truth, we provide only qualitative results. Fig. 7 shows selected frames from a sequence where the subject climbs over an obstacle. We find that SOP has a lot of trouble in reconstructing the leg motion and systematically underestimates arm and knee bending. Our results are comparable to SIP although sometimes the limbs are more outstretched than in the baseline. However, note that SIP optimizes over the whole sequence and is hence computationally very expensive, whereas ours produces predictions in milliseconds.

5.2.2 TotalCapture (real). Here we provide a qualitative comparison of our method and the baseline on the TotalCapture dataset [Trumble et al. 2017]. Fig. 8 summarizes three different sample frames from the dataset. We note that for challenging motions such as back bending (bottom row) and leg raises (middle row), our model outperforms both SIP and SOP and is very close to the reference. Fig. 8 also shows a case where our model successfully reconstructs a leg-raise when SIP and SOP fail. Note however, that this difficult motion also fails to be reconstructed by our model at times (cf. Section 6.2).

5.2.3 DIP-IMU (real). Lastly, we illustrate results on our own DIP-IMU dataset (cf. Sec. 3.3 and Appendix F). Here the reference ("ground truth") is obtained by running SIP using all 17 IMUs. At test time, however, we only use 6 IMUs as input for our method, and
To demonstrate that our model runs in real-time, we implemented an on-line system that streams the sensor data directly into the model and displays the output poses. The raw IMU readings are retrieved via the Xsens SDK, the model’s output is displayed via Unity and all communication is performed via the network. Example frames from the live demo are shown in Fig. 1 and Fig. 10. The results are best viewed in the supplementary video. For the live demo we use the online version of the fine-tuned BiRNN model as explained in Section 5.2, i.e. using 20 frames from the past and 5 frames from the future. The system runs at approximately 29 fps while still producing faithful results (cf. appendix E).

6 DISCUSSION AND LIMITATIONS

6.1 Generalization

In this paper we have shown that our model is able to generalize well to unseen data based on the following observations. (i) We achieve good results on a held-out dataset with real IMU recordings (Total Capture) despite training on synthetic data only (cf. Section 5.1). (ii) We show good qualitative performance on another held-out dataset, Playground, and in the live demo (cf. Section 5.2). This is challenging to achieve due to differences in motions, sensors, and data preprocessing across the datasets. (iii) The system is robust w.r.t. different root orientations (cf. Fig. 1). However, robustness to even more poses, datasets and settings is still the subject of future work. We hypothesize that one of the main limitations is the difficulty of modeling accelerations (synthetic and real) effectively. This is the main reason for fine-tuning on DIP-IMU, which improves generalization but certainly does not have to be the final solution to this problem. In the following we report additional experiments to provide insight into these issues.

Synthetic vs. real. We first train a BiRNN on a smaller, real dataset (DIP-IMU) as opposed to a large, synthetic one (AMASS). We subsequently evaluate this model on TotalCapture, where we notice a drop in performance of around 5.2° (21.03° ±16.35°). Testing on the DIP-IMU held-out set yields an error of (18.84° ±14.08°), which is comparable to the performance when training with synthetic and fine-tuning with DIP-IMU (17.54°). However, the latter yields better performance on TotalCapture. These results illustrate the benefits of a large synthetic motion database.

Re-Synthesis. To analyze the impact of differences between synthetic and real data, we synthesize the IMU measurements for the real datasets (TotalCapture and DIP-IMU). We then evaluate our best BiRNN on these synthetic versions of TotalCapture and DIP-IMU and compare it with the performance on real data. The model performs better on the synthetic version of both TotalCapture (improvement by 2.71° to 13.14° ±10.50°) and DIP-IMU (improvement by 8.84° to 22.86° ±15.70°), highlighting domain differences that need to be addressed. Appendix D provides a more detailed discussion and additional experiments. In summary, we hypothesize that differences in accelerations lie at the core of this problem.

6.2 Failure cases

Fig. 11 (top) shows typical failure cases, taken from an example sequence in TotalCapture. While the model is robust to various root orientations in the live demo, extreme cases, where the body is parallel to the floor (such as push-ups), are challenging. Finally, we compute the worst 5% poses on the test set of DIP-IMU, which results in a mean joint angle error of 43.68° ±8.53° degrees and show the worst poses in appendix B. Leg raises are among the most difficult motions as the sensors show very similar orientation readings while performing this motion. Hence, only acceleration can disambiguate these [von Marcard et al. 2017]. However, sensitivity for the baselines SIP and SOP. Fig. 9 summarizes several sequences from the dataset. It is evident that our model outperforms both SIP and SOP qualitatively in a consistent way. We see that SIP/SOP creates a lot of inter-penetrations between limbs and the torso. Our model more faithfully reproduces the arm motions over a large range of frames and poses. Interestingly, our model rarely produces inter-penetrations and produces smooth motion despite noise in the inputs (see video) and without any explicit smoothness or inter-penetration constraints. Hence, DIP learns a mapping from IMU data to the space of valid poses and motion. Smoothness may be explained by the regularization of training via dropouts.

5.3 Live demo

To demonstrate that our model runs in real-time, we implemented an on-line system that streams the sensor data directly into the model and displays the output poses. The raw IMU readings are retrieved via the Xsens SDK, the model’s output is displayed via Unity and all communication is performed via the network. Example frames from the live demo are shown in Fig. 1 and Fig. 10. The results are best viewed in the supplementary video. For the live demo we use the online version of the fine-tuned BiRNN model as explained in Section 5.2, i.e. using 20 frames from the past and 5 frames from the future. The system runs at approximately 29 fps while still producing faithful results (cf. appendix E).
to IMU placement, environmental factors, and different noise characteristics per sensor make it extremely challenging to integrate them effectively into a learning-based approach. Fig. 11 (bottom) shows how both our model and the baselines fail to reconstruct a leg raise. We believe that these problems arise from the fact that our model struggles to fully exploit the acceleration information. Hence, future work should focus on addressing this challenge by modeling the noise in acceleration and exploring new sensor placement.

7 CONCLUSION AND FUTURE WORK

We presented a new learning-based pose-estimation method that requires only 6 IMUs as input, runs in real-time, and avoids the direct line-of-sight requirements of camera-based systems. We leverage a large Mocap corpus to synthesize IMU data (orientation and acceleration) using the SMPL body model. From this synthetic data, we show how to learn a model (DIP) that generalizes to real IMU data, obtaining an accuracy of $15.85^\circ$ angular error on TotalCapture. We exploit temporal information by using a bi-directional RNN that propagates information forward and backwards in time; at training time DIP has access to full sequences, whereas at test time the model has access to the last 20 frames and only 5 frames in the future. This produces accurate pose estimates at a latency of only 85ms. Even satisfying the real-time requirement, DIP performs comparably to, or better than, the competing off-line approach, SIP. Furthermore, DIP produces results that are smooth and generally without interpenetrations. This demonstrates that DIP learns a mapping to the space of valid human poses without requiring explicit smoothness or joint angle limits.

Future work should address capturing multi-person interactions and people interacting with objects and the environment. While the focus of this work has been a system based purely on wearable sensors, some applications admit external, or body mounted cameras [Rhodin et al. 2016]. It would be interesting to integrate visual input with our tracker in order to obtain even better pose estimates, especially to capture contact points, knee bends and sitting-down poses, which are difficult to recover using only 6 IMUs. While our approach runs in real-time, transferring the motion data over the Internet may introduce latency, which is a problem for virtual social interaction. Hence, we will explore ways to predict into the future to reduce latency. Finally, unlike [von Marcard et al. 2017], no global translation is considered in our method. This limitation can be critical in some application scenarios. We see two possible solutions to this. First, a GPS signal, which is integrated into most phones, could be integrated into DIP to obtain reasonable global position. Another potential way is to regress the global translations directly from the temporal IMU inputs. We leave this for future work.

We have demonstrated the capabilities of DIP by displaying its pose predictions in real time. We believe that real-time pose estimation methods, which require only a small number of wearable sensors like DIP, will play a key role for emerging interactive technologies such as VR and AR.

ACKNOWLEDGMENTS

We would like to express our gratitude to Timo von Marcard for providing code and support to run SIP/SOP and for fruitful discussions throughout this project. We thank the reviewers for their valuable comments and all participants for their efforts spent in recording DIP-IMU. We are also grateful to Velko Vechev for his extensive help with the live demo and Jonathan Williams and Benjamin Pellkofer for web development. This work was supported in part by the ERC Grant OPTINT (StG-2016-717054). We thank the NVIDIA Corporation for the donation of GPUs used in this work. Disclosure: MJB has received research gift funds from Intel, Nvidia, Adobe, Facebook, and Amazon. While MJB is a part-time employee of Amazon, his research was performed solely at, and funded solely by, MPI.
REFERENCES


Xianjie Chen and Alan L Yuille. 2014. Articulated pose estimation by a graphical model with image dependent pairwise relations. In NIPS. 1736–1744.


ACM Trans. Graph., Vol. 37, No. 6, Article 185. Publication date: November 2018.
Deep Inertial Poser: Learning to Reconstruct Human Pose from Sparse Inertial Measurements in Real Time


Xiaolin Wei, Peizhao Zhang, and Jinxian Chai. 2012. Accurate realtime full-body motion capture using a single depth camera. ACM Transactions on Graphics (TOG) 31, 6 (2012), 188.


A ADDITIONAL ARCHITECTURES

Along with the models discussed in the paper, we experimented with other, non-recurrent structures. Specifically, we implemented a WaveNet architecture [van den Oord et al. 2016] and a simple feed-forward network (FFN). The FFN is composed of 5 fully-connected layers with 256, 512, 256, 256 and 256 units per layer respectively. At a single time step t the model is fed a temporal window of 20 past and 5 future frames. Table 4 summarizes the results in terms of mean joint angle error.

The FFN performs around 4.4° worse on TotalCapture than our best BiRNN evaluated on 20 past and 5 future frames. Additionally, the output is greatly corrupted by jerkiness and trembling artifacts. WaveNet performs better both in terms of the joint angle error and visual quality. Although WaveNet is able to considerably reduce the trembling artifacts, they are still apparent, resulting in displeasing visual output.

Furthermore, the BiRNN model proposed in this paper offers much greater flexibility. Increasing the input window size in a feed-forward network requires both retraining the model and incurs a large growth in trainable parameters. This is not the case for the BiRNN; the window length can be changed “on the fly” for the same model and hence does not affect the number of parameters.

Table 4. Performance of WaveNet and a feed-forward network (FFN) on TotalCapture and DIP-IMU in terms of the mean joint angle error in degrees. Both models were trained on the synthetic AMASS training set.

<table>
<thead>
<tr>
<th></th>
<th>TotalCapture</th>
<th>DIP-IMU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p(±std)</td>
<td>p(±std)</td>
</tr>
<tr>
<td>WaveNet</td>
<td>17.26 (±13.91)</td>
<td>30.79 (±17.98)</td>
</tr>
<tr>
<td>Feed-forward</td>
<td>20.32 (±15.67)</td>
<td>31.88 (±20.63)</td>
</tr>
</tbody>
</table>

B ADDITIONAL FIGURES

Fig. 12 shows the architecture details of the BiRNN as reported in the paper. In Fig. 13 we show the three poses with the highest mean joint angle error taken from the test set of DIP-IMU.

C NORMALIZATION

In Section 4.3 we show how the data is normalized w.r.t. the root. We experimented with more sophisticated schemes, which we did not find to be beneficial and explain in the following.

Per-sequence normalization. Instead of normalizing the sensor measurements per frame, we experimented with normalizing them only to the root orientation in the initial frame of the sequence. In this case, we fed measurements of all 6 sensors into the model.
with a BiRNN configuration that does not use any accelerations in w.r.t. the heading of the root sensor. We implemented this strategy to better understand the influence of acceleration on final pose (instead of 5 in the current architecture), whereby the root orientation in the first frame is always the identity. In other words, the normalization is performed as follows for all sensors $s \in \{1, \ldots, 6\}$:

$$
R_{TB}^T(t) = R^{-1}_{\text{root}}(0) \cdot R_{TB}^T(t),
\hat{a}_s(t) = R^{-1}_{\text{root}}(0) \cdot (a_s(t) - \hat{a}_s(t))
$$

**Per-frame heading removal.** Another option is to normalize only w.r.t. the heading of the root sensor. We implemented this strategy as follows: We extract the yaw angle $\gamma$ from the root orientation and create a new rotation matrix $R_{\text{yaw}}$ that rotates around the y-axis by $\gamma$. $R_{\text{yaw}}$ then replaces $R_{\text{root}}$ in Equations (9) and (10) to perform the normalization for all sensors $s \in \{1, \ldots, 6\}$ and all time steps $t$. Training our BiRNN with this normalization scheme results in a mean angular error of $40.18^\circ \pm 21.74^\circ$ on TotalCapture, which is $24.41^\circ$ worse than the per-frame normalization we adopt.

**Zero-mean unit-variance.** Note that after we normalize orientations and accelerations to the root, we subtract the mean and divide by the standard deviation to rescale the inputs; i.e., the input to the model is $(x_s - \mu)/\sigma$, where the statistics $\mu, \sigma \in \mathbb{R}^{60}$ are computed over the entire training data set. The same normalization procedure is applied to the outputs.

## D. Influence of Acceleration

To better understand the influence of acceleration on final pose estimates we conduct a further experiment in which we compare our best network configuration (BiRNN, fine-tuning with accelerations) with a BiRNN configuration that does not use any accelerations in the input. In this experiment, we make two observations. First, the error on both TotalCapture and DIP-IMU is higher (both the mean and the standard deviations). Second, the error on TotalCapture increases to $19.45^\circ$ after fine-tuning. These findings indicate that both orientations and accelerations are indeed essential to produce accurate predictions, especially for the more challenging poses in DIP-IMU. Furthermore, the higher error on TotalCapture indicates that without accelerations the network overfits to DIP-IMU.

**Table 5. Performance of a BiRNN that does not use accelerations as inputs (no acc) compared to the best model as reported in the paper.**

<table>
<thead>
<tr>
<th>Model</th>
<th>TotalCapture $\mu \pm \sigma$</th>
<th>DIP-IMU $\mu \pm \sigma$</th>
</tr>
</thead>
<tbody>
<tr>
<td>BiRNN (fine-tuning)</td>
<td>$16.84 \pm 13.22$</td>
<td>$17.54 \pm 11.54$</td>
</tr>
<tr>
<td>BiRNN (fine-tuning, no acc)</td>
<td>$19.45 \pm 15.67$</td>
<td>$18.89 \pm 15.24$</td>
</tr>
</tbody>
</table>

## E. Hardware Specifications

We trained our models on a NVIDIA GTX Titan X (Pascal, 12 GB), which took roughly 3 hours for our best BiRNN model. In the live demo, both the visualization (in Unity) and the model inference (in Python) run on the same machine, i.e., both processes access the Titan X GPU. Because Unity and Python communicate through the network stack, it is possible to run the visualization component on a different, less potent machine. To test this, we run the visualization on a commodity laptop, an Asus Zenbook (CPU i7-3632QM @ 2.20 GHz, 8 GB RAM, NVIDIA GeForce GT 650M (2 GB)).

Table 7 summarizes the resulting FPS for these different settings and for 4 different models. Reported is the time it takes to grab the current measurements from the IMUs, send them to the model over the network and retrieve the model’s predictions for that time step; i.e., the FPS displayed by Unity might differ. Furthermore, the actual
runtime of the RNN model is technically capped by the update rate of the Xsens sensors (60 Hz).

F DATA COLLECTION

Table 6 summarizes the protocol we used for collecting our new real dataset, DIP-IMU. Particular attention was paid to recording activities that are underrepresented in existing Mocap datasets.

<table>
<thead>
<tr>
<th>Categories</th>
<th>Motions (# Repetitions)</th>
<th># Frames</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upper Body</td>
<td>Arm raises, stretches, and swings (10). Arm crossings on torso and behind head (10).</td>
<td>116,817</td>
<td>32.45</td>
</tr>
<tr>
<td>Lower Body</td>
<td>Leg raises (10). Squats (shoulder-width and wide) (5). Lunges (5).</td>
<td>70,743</td>
<td>19.65</td>
</tr>
<tr>
<td>Locomotion</td>
<td>Walking straight (3). Walking in circle (2). Sidesteps, crossing legs (1). Sidesteps, touching feet (1).</td>
<td>73,935</td>
<td>20.54</td>
</tr>
<tr>
<td>Freestyle</td>
<td>The subject can select one of the following activities: jumping jacks, tennis, kicking/boxing, pushups, basketball. Choice of jumping jacks is predominant.</td>
<td>18,587</td>
<td>5.16</td>
</tr>
<tr>
<td>Interaction</td>
<td>The subject sits at a table and interacts with everyday objects, such as keyboard, mobile device, toys, grabbing objects in front of them, touching mounted screen displays. Freestyle for 1 minute.</td>
<td>50,096</td>
<td>13.92</td>
</tr>
</tbody>
</table>

Table 7. Average running time in FPS for the model inference component of our live demo. Measured is the time it takes to grab the current measurements from the IMUs, send them to the model over the network and retrieve the predictions for that time step. Local means visualization and model inference run on the same machine (here the machine used for training). Remote Laptop means the visualization runs on a commodity laptop.

<table>
<thead>
<tr>
<th></th>
<th>Local [fps]</th>
<th>Remote Laptop [fps]</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNN</td>
<td>173.3</td>
<td>67.3</td>
</tr>
<tr>
<td>BiRNN (20, 5)</td>
<td>29.7</td>
<td>25.1</td>
</tr>
<tr>
<td>BiRNN (50, 5)</td>
<td>15.7</td>
<td>14.8</td>
</tr>
<tr>
<td>BiRNN (100, 5)</td>
<td>9.3</td>
<td>8.7</td>
</tr>
</tbody>
</table>