Global Position Prediction for Interactive Motion Capture

PAUL SCHREINER, University of Copenhagen, Denmark and Rokoko Electronics, Denmark
MAKSYM PEREPICHKA, Concordia University, Canada
HAYDEN LEWIS, Clemson University, USA
SUNE DARKNER, University of Copenhagen, Denmark
PAUL G. KRY, McGill University, Canada
KENNY ERLEBEN, University of Copenhagen, Denmark
VICTOR B. ZORDAN, Clemson University, USA

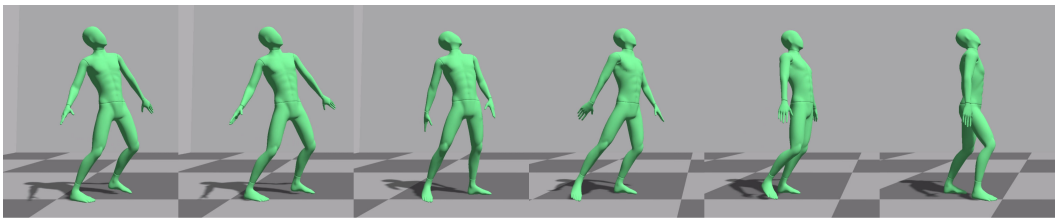

Fig. 1. Example of captured motion where our method can create global position for unseen motion capture data. Because we train our u-nets with a large corpus of motion capture data, we are able to reconstruct global position for a wide variety of behaviors, even this unusual zombie-style walk.

We present a method for reconstructing the global position of motion capture where position sensing is poor or unavailable. Capture systems, such as IMU suits, can provide excellent pose and orientation data of a capture subject, but otherwise need post processing to estimate global position. We propose a solution that trains a neural network to predict, in real-time, the height and body displacement given a short window of pose and orientation data. Our training dataset contains pre-recorded data with global positions from many different capture subjects, performing a wide variety of activities in order to broadly train a network to estimate on like and unseen activities. We compare training on two network architectures, a universal network (u-net) and a traditional convolutional neural network (CNN) - observing better error properties for the u-net in our results. We also evaluate our method for different classes of motion. We observe high quality results for motion examples with good representation in specialized datasets, while general performance appears better in a more broadly sampled dataset when input motions are far from training examples.

CCS Concepts: • **Computing methodologies** → **Motion capture**; **Motion processing**; **Animation**.

Additional Key Words and Phrases: motion capture, neural networks, IMU

Authors' addresses: Paul Schreiner, University of Copenhagen, Denmark and Rokoko Electronics, Denmark, paul.schreiner@di.ku.com; Maksym Perepichka, Concordia University, Canada, maksym@perepichka.com; Hayden Lewis, Clemson University, USA, whlewis@g.clemson.edu; Sune Darkner, University of Copenhagen, Denmark, darkner@di.ku.dk; Paul G. Kry, McGill University, Canada, kry@cs.mcgill.ca; Kenny Erleben, University of Copenhagen, Denmark, kenny@di.ku.dk; Victor B. Zordan, Clemson University, USA, vbz@g.clemson.edu.

2577-6193/2021/9-ART $15.00
https://doi.org/10.1145/3479985

ACM Reference Format:
Paul Schreiner, Maksym Perepichka, Hayden Lewis, Sune Darkner, Paul G. Kry, Kenny Erleben, and Victor B. Zordan. 2021. Global Position Prediction for Interactive Motion Capture. *Proc. ACM Comput. Graph. Interact. Tech.* 4, 3 (September 2021), 16 pages. https://doi.org/10.1145/3479985

1 INTRODUCTION

Motion capture is making the transition from the studio to the home and consumer markets with virtual reality (VR) game consoles and related hardware demanding lower cost, less cumbersome, and interactive/real-time performance capture technologies. As the go-to commercial technology today, camera-based motion capture systems are quite common and offer attractive solutions for both marker-based and markerless capture solutions. However, consumer motion capture solutions, such as IMU suits, have the advantages of being untethered, do not suffer from occlusion, and avoid the need for dedicated space with carefully calibrated cameras. Additionally, in most cases, the equipment, such as inertial measurement units (IMUs), is considerably less expensive than camera-based hardware.

The problem with IMU-based motion capture is that it does not provide a direct measurement of position. IMUs typically include accelerometers, magnetometers, and gyroscopes, which allow for an excellent measurement of rotation that can be used to reconstruct the pose of limbs as well as the orientation of the capture subject. In contrast, it is prohibitive to calculate useful position estimates through integration of the accelerometer signal due to noise and bias in their measurements. To the best of our knowledge, standard commercial (proprietary) solutions apply heuristics, such as reconstructing from assumed foot-ground contacts for interactive playback, and otherwise assume that errors can be corrected with post processing. But we note that this problem is also not unique to IMU capture, it exists for many measurement systems which focus on joint angle measurements, such as exoskeletons, strain sensors, and monocular camera systems.

In this paper, we propose a learning-based solution to compute the global position of joint or "pose" based motion capture by exploiting a large collection of previously recorded (optical) motion capture data. To this end, we train a neural network offline to predict the global body displacement from pose data based on a short-horizon trajectory of current pose data. We hypothesize that the information present in this short trajectory contains sufficient detail about the dynamics being captured that a short temporal window of such data will provide key information to predict the character's global motion. Namely, following training, the network predicts the vertical position of the center of mass and its horizontal displacements per frame. The latter is integrated to reconstruct the global motion. The use of a fixed temporal window makes the solution history independent, in contrast to, for instance, a recurrent neural network or other global optimization solution which considers a full trajectory. Further, because our solution requires only a short window of data, it is ideal for real-time use. Specifically, we showcase the use of a universal network (u-net) [Ronneberger et al. 2015] to accomplish this effort, while we contrast it to other network architectures as well as their sensitivity to training data, window length, and a number of other system and network hyper-parameters. Once trained, the u-net prediction is much faster than real-time with our unoptimized code running at around 200 fps (further discussion on the timing appears later in the paper.)

Figure 1 shows a preview of our results. In addition to the qualitative evaluation of our animations, we compute errors for our reconstructions compared to (held-out) test data and independent reconstructions. We evaluate our design decisions through ablation studies, and choice of data representations. Finally, we present results and comparisons on multiple data subsets, and discuss the limitations and advantages as well as future improvements.

2 RELATED WORK

Motion capture data recording and reuse has received a tremendous amount of attention in the research of human computer animation and analysis. We refer readers to additional sources, such as the book by Menache [2011], for the basics of the topic as a research domain, and instead focus here on the specific competing technology and approaches related to this work. Namely, we see our effort as an alternative to the common marker-based optical recording technologies, which require controlled environments and a relatively cumbersome set up [Thewlis et al. 2013]. While optical systems provide precision data, they require expert operators and are (still) rather expensive, relegating them to specialized and high-end studios or labs that are well funded and have ample space to devote to motion capture.

Because of their low cost, and relative ease-of-use, inertial motion recording systems in particular have attracted the attention of practitioners in commercial applications including sports, medicine, and entertainment. Research to address the position problem, the focus of our paper, also appears in sports and medicine, where the aim is high accuracy under very specific conditions. A number of solutions have been proposed to meet the needs of the specific domain [Coyte et al. 2013; Kok et al. 2017; Lapinski et al. 2009; Li et al. 2020; Suh 2014; Widagdo et al. 2017]. For example, biomedical researchers have proposed to correct special cases of errors for rehabilitation [Coyte et al. 2013]. Others have focused on reducing drift in specialized highly dynamic behaviors of interest [Fasel et al. 2017].

In the animation research community, the requirement for accuracy may be relaxed in favor of lower cost and more flexible solutions. For example, in support of recording alternatives, animation researchers have offered a variety of solutions for systems that compete with marker-based optical systems [Shiratori et al. 2011; Slyper and Hodgins 2008; Vlasic et al. 2007].

Alternative motion capture systems have been commercially available for many years, e.g., IMU suits [Roetenberg et al. 2009], and researchers have been offering solutions for computing and improving collected data from such for nearly as long [Floor-Westerdijk et al. 2012; Roetenberg et al. 2009; Schwarz et al. 2012]. However, to date, non-optical motion capture solutions have not become mainstream due to their limitations.

Finding global positions from pose information has also been studied extensively in the field of computer vision, where global trajectories are estimated from images [Mehta et al. 2017; Pavllo et al. 2019; Shimada et al. 2020; Véges and Lőrincz 2019; Zhou et al. 2018]. These techniques rely on 2D pose representations in an image space, i.e., with respect to a (normalized) camera focal point. This representation implicitly encodes 3D space in a 2D projection and is intrinsically different to our problem where we try to estimate 3D global coordinates from local 3D pose parameters.

2.1 Inertial motion capture

For inertial systems, motion capture data is extracted readily to provide body orientations [Roetenberg et al. 2009] and, from these, joint angles and body orientation for a given skeleton. While one can estimate sensor position displacement by integration of the acceleration from the IMUs, the calculation is prone to errors [Floor-Westerdijk et al. 2012]. Some researchers have proposed methods, for example particle filters, to extract better position data [Schwarz et al. 2012]. Others suggest the addition of complementary sensors, such as global positioning systems (GPS) [Roetenberg et al. 2009] or ultraWide band (UWB) sensors for indoor positioning [Corrales et al. 2008].

Clearly, an attractive option is to avoid additional hardware and instead employ a software solution derived from the input. Several solutions have been proposed in this area as well. The most straightforward approach is to employ the knowledge of the kinematics, for example extracting foot contact phases and then using this as a root to dictate forward motion through kinematics [Yuan

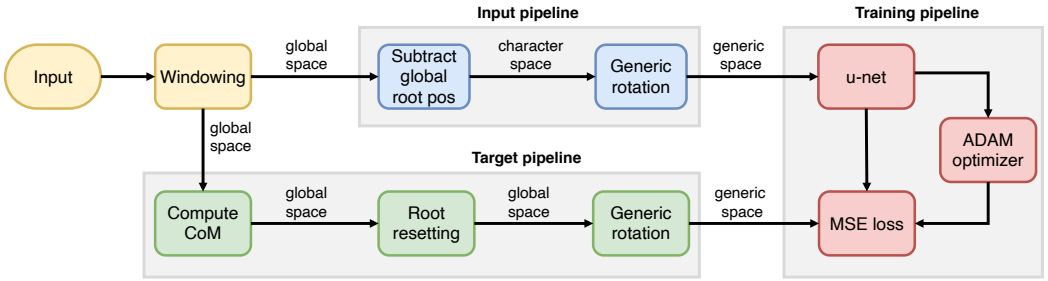

Fig. 2. Input data preprocessing pipeline for motion capture data imported from Rokoko's Motion Library and targets for training a neural network, u-net, to predict center of mass positions given a time-window of relative joint data input.

et al. 2011; Zheng et al. 2014]. While we cannot be certain, we believe proprietary solutions for commercial packages use heuristics, kinematics, and Kalmann filtering, to extract foot contacts. Furthermore, various methods have been proposed to determine locomotion phase including special sensors to detect foot and heal strikes [Foxlin 2005; Ju et al. 2015]. Unfortunately, both heuristic and sensor techniques are prone to errors when contact changes occur and in behaviors that include flight phases, e.g., running [Suh 2014]. Our technique aims to follow the current trends in machine learning (ML) to address the root positioning problem of such systems, especially for real-time applications, such as gaming and direct playback. While some IMU-based research has employed ML, for example to handle noisy IMU placement [Xiao and Zarar 2018], to our knowledge, no research has been published concerning use of ML in position estimation for real-time capture to date. Zhou et al. [2020] present a synthesis tool for keyframing which includes a global path predictor with some like characteristics to our own, although they solve a synthesis problem for "in-betweening" while we solve a reconstruction problem in comparison. Notably,

they report a precision of 0.7 cm/frame which is on the order of 100 times larger than our results.

2.2 Neural networks for motion capture

The fields of computer animation and motion capture have had a multitude of ML techniques employed in both academia and industry. Recurrent neural networks (RNNs) are an evident choice for dealing with time-series data and have been employed for motion learning [Fragkiadaki et al. 2015; Martinez et al. 2017]. However, RNN-based approaches often suffer from noisy outputs, a problem that has been addressed by a multitude of authors. Ghosh et al. [2017] combine dropout autoencoders with LSTM (DAE-LSTM), i.e., using LSTMs to predict motion poses and denoising autoencoders (DAEs) to filter output reducing accumulative error. Recently, Wang et al. [2019] propose spatio-temporal recurrent neural networks (STRNN) consisting of three networks: a network to encode temporal dependencies, a spatial network to learn body-part dependencies, and a residual component to smooth out high frequency noise.

Holden et al. [2016, 2015] introduced the usage of convolutional neural nets (CNNs) to the domain of computer animation, using temporal convolutional layers to learn motion manifolds on the CMU motion capture dataset. Overlapping windows of motion are computed on animation data and provided as input to the network. CNNs have the advantage of being typically easy to train and producing smooth motion output, but suffer from their inability to deal with long-term dependencies. Recently, specialized models designed for animation data have achieved state-of-the-art results, such as phase function neural networks (PFNN) [Holden et al. 2017], which explicitly use footstep

phase information to dynamically change network weights to drive character controllers. However, this method requires prior explicit manual labeling of motion phase information. Mode adaptive neural networks (MANN) [Zhang et al. 2018] address this issue by automatically extracting phase information. Starke et al. [2020] improve upon this by considering local motion phases, determined by joint contacts with external objects.

Processing of time-series (e.g. motion capture) with neural networks has primarily been performed using RNNs which are specifically designed for this purpose.

Although their application has been successful in the context of text and speech [Chiu and Nichols 2016; Graves et al. 2013] and EEG series in sleep classification [Biswal et al. 2017], they are notoriously hard to train [Pascanu et al. 2013]. To overcome the challenges of RNNs Perslev et al. [2019] propose the u-net architecture [Ronneberger et al. 2015] for time-series data as an alternative, and report superior performance, significantly increased stability, and ability to be trained on very large data sets [Perslev et al. 2021]. The scale-space convolutional structure of the u-net enables the modeling of temporal-spatial correlation over a fixed time-window and by modifying the u-net architecture to a regression output provides us with an ensemble of predictors. Ensembles are know for robustness and accuracy [Hastie et al. 2009] and the average the set of predictions from our ensemble provides the final prediction. The choice of u-net offers faster training compared RNNs, increased stability during training and real-time inference [Perslev et al. 2019].

3 METHOD

In this section, we introduce the u-net architecture and present the data pre- and post-processing pipeline that we use to optimize the learning capacity of the network. A core idea behind the use of this type of network architecture is the ability to learn correlations between pose data and its spatial-temporal correlation structure at multiple temporal scales. Figure 2 provides an overview of different parts of our training pipeline that we describe throughout the remainder of this section.

3.1 U-net architecture

Through empirical experimentation we opt to employ the u-net architecture for our specialized problem. To adapt this tool for our needs, our u-net is modified for regression and acts as an ensemble of regression models from which we construct our prediction. Its layout is summarized in Figure 3. This network consists of an encoder stage and a decoder stage with skip connections relaying information at different temporal scales. In the encoder stage, the input data is encoded in the temporal dimension while being expanded in the feature dimension using convolutional layers. The input to the network is a 2D tensor, with time in the vertical dimension and features in the horizontal dimension.

We use T to denote the time-window size and N for the dimension of the combined feature vectors. A description of the layers, sizes, and features of the network architecture are as follows:

- The u-net operates at 3 different scales in the encoding and 3 scales in the decoding. At each scale 2 consecutive convolutions of the input to that scale are performed.
- The first convolution is 2 dimensional with a kernel spanning the entire feature dimension N. In the temporal dimension we have experimented with kernels of size 3 and 5 and found that a kernel of size 5 generally provides the best results.
- With an input of [Batch $\times$ Channels in $\times$ T $\times$ N] the output of the first convolution looks like [Batch $\times$ Channels out $\times$ T $\times$ 1]. *Channel in* is generally 1 in our use case.
- The number of output channels of the first convolution doubles for each up sampling layer and is halved for each down sampling layer.

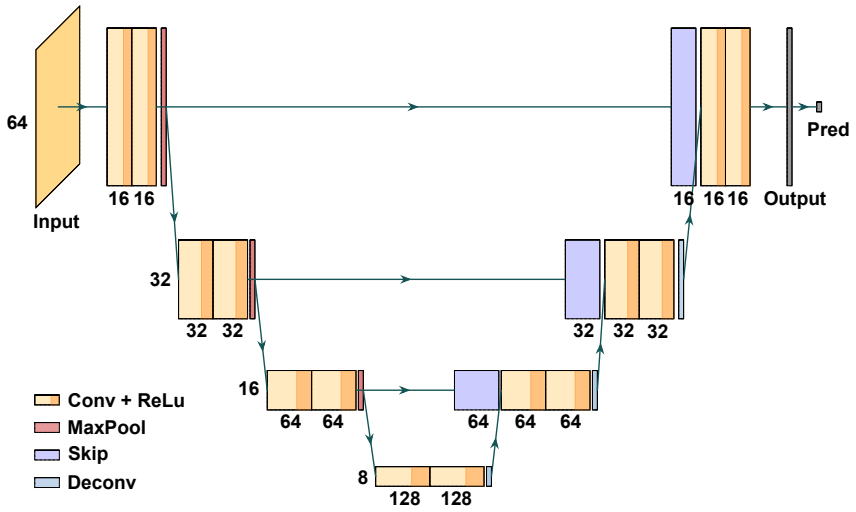

Fig. 3. Illustration of our u-net layout, observe that skip connections and up/down sampling allow the u-net to handle time-series data and perform analysis of data at different frequency levels.

- The second convolution is exclusively in the temporal dimension, over all the output channels from the first convolution.
- The activation functions used throughout the network are rectified linear units (ReLU).
- After each set of convolutions the output of that step is reshaped so that the input to the next layer is again of the form [Batch $\times$ 1 $\times$ T $\times$ F]. Here F = *Channels out* can be seen as a new abstract feature dimension.
- At the end of the layer the current output is stored for later use in the skip connections. Then the output is down sampled in the temporal dimension using a maxpool operation with a length of 2. The feature dimension is kept constant during this step.
- Between each down and up sampling layer of the same temporal scale, there is a skip connection which passes the output of the encoder directly to its temporal counter part in the decoder side of the network. This ensures that the network can extract information and process it in the output for multiple timescales.
- The decoder structure follows an inverse description of the encoding process, where the up sampling is performed using linear interpolation.

In our implementation, we use the ADAM optimizer with a weight decay of $1e^{-5}$ as the only non-default parameter. We use the MSE loss function from the PyTorch library with all parameters set to default values.

3.2 Source data

Our raw data is a rich collection of assets each containing the performance of a single motion or a short sequence of motions. Our training motion set comes from commercially available databases for human motion data, with 577 different assets from 61 unique actors, totalling 629,093 frames. Note, we purposefully include data from different motion capture studios (authors) and individuals (actors) to support diversity with respect to capture variance, and character size, shape, and gender. Each asset also may have individually distinct bone dimensions. Further, the range of motion types contained is also diverse, examples include walking, dance, jumps, martial arts, idle motion, and more. The animation data is encoded in a hierarchical format using a skeleton consisting of 19

bones plus a root bone (the pelvic bone). Each bone's motion is stored as an offset from its parent and a rotational trajectory. The root's motion is represented by positional and rotational trajectories with respect to a global frame.

For training the data was split into 3 subsets: a training set, a validation set and a left-out test set. The training set is used to train the model, the validation set is used to calculate a validation loss at the end of each training epoch. Finally, the left-out test set is used after training has converged to determine error statistics and all other results in the result section of this paper.

3.3 Pre-processing of input data

In preparation for training, we re-sample the data to a uniform frame rate of 100 Hz as is typical with IMU motion capture systems (similar to Perslev et al. [2019]). This is necessary because the input to the u-net requires a consistent temporal frequency, that is, each pose window must be the same size and span the same period. However, our training motion assets come from different sources with different frame rates. Subsequently, we preprocess the data by extracting short temporal windows, and mapping data into a generic forward facing reference frame (see also the pipeline diagram in Figure 2).

The data is passed to the network in short sequences of frames which we call *windows*. Each window is conceptually a short (less than one second) animation, with a fixed length of time. We chose a window size of $T = 64$ throughout this work. This windowing is performed online at training time, and has the advantage that we do not need to store duplicate frame data, hence reducing memory usage during training. This has negligible impact on training time as it is simply an array of pointers to memory. The windows overlap with a stride of 1 causing every frame in the data to be passed to the network in T consecutive windows. During training, the windows are shuffled in order to avoid bias from temporal correlation.

In addition, rather than using the local hierarchical joint angles, we preprocess each asset based on its skeletal dimensions by calculating 3D position vectors that indicate joint positions with respect to the root (the pelvic bone) and use those to represent poses. For our pose data, we assume the root that has a fixed position at the origin of a world frame. Further, as motion within the physical world is invariant to facing direction in the horizontal plane, we also define a *generic rotation* in which our model is to be trained. We align the vertical axis of our reference frame to match the the global vertical, opposite the direction of gravity. Next, the axes of the horizontal plane of our reference frame is set from the orientation of the root at the first frame of each window. That is, the root forward axis is projected to the global forward direction and the lateral axis is orthogonal to both the forward and the vertical.

The 19 joint position vectors are stacked and zero padded to form a 1D array $f \in \mathbb{R}^{64}$ for each frame. The root is not included in the input tensor because it is assumed to be poor or missing in the input data. The padding is done to maintain identical dimensions across the down- and up-sampling layers of the u-net. Finally, each series of 64 consecutive frames are concatenated to form $[64 \times 64]$ tensors which are fed to the network in batches of 16 windows, thus $[16 \times 64 \times 64]$ is the size of the input tensors to our networks.

3.4 Processing of training targets

To compute the training targets, we adjust the input samples by: 1) computing the center of mass and treating this signal as the target; and 2) zeroing the root displacement at the start of the temporal window.

The root of our character is defined as the pelvic body which is subject to its own oscillations, for example, as the hips shift left and right in forward walking. Instead of estimating the root, we predict the center of mass (CM) positions, as it is less oscillatory and more indicative of the

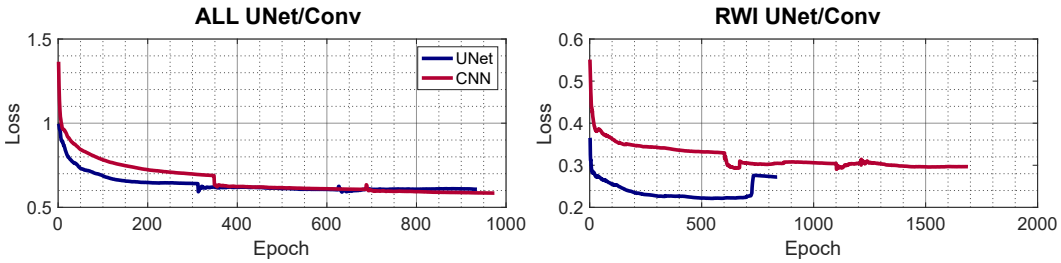

Fig. 4. Validation loss plots for all data (ALL) or run walk idle (RWI), and different network architectures. Note the discontinuities in most plots caused by restarting the ADAM optimizer. The jump in the curve of the of the u-net in the RWI plot is caused by over fitting. Note that the blue (u-net) and red (CNN) curves are consistently in the same loss range within experiments.

dynamics of the behavior. The CM estimates of the training targets are computed by summing a weighted approximation of each limb's center of mass. The weighting of each limb was performed using a re-targeting of the parameters from Dumas and Wojtusch [2017]. In our experiments, we found this simple approach of computing the CM to be sufficient. In addition, to make the network invariant to the starting position of a time window, the trajectory in the horizontal plane of each window is reset to start at the origin. In the result, the training target is a time series representing the displacement of the character over the time period of the window.

In post processing, to recover global root data, the system performs an inverse of the target pre-processing. As the same frame in time is present in multiple target windows (due to the windowing), the network provides multiple predictions for the CM target of the same frame. So, we opt to collect all estimations and compute a mean value for our final CM position prediction.

4 IMPLEMENTATION AND RESULTS

Our u-net framework is implemented in Python, using PyTorch 1.7.0 for training the neural networks and NumPy/SciPy for data pre-processing. The experiments were all trained on a cluster using NVIDIA TITAN RTX GPUs. The memory use on the GPU during training is approximately 3 GB. The focus of the experiments is to show that we can solve the global positioning problem using a neural network in real time. After training, the u-net runs above speeds needed for real time, in excess of 200 fps. However, there is an inherent delay due to the window size, as follows. If a window has size T, then the first T frames the u-net makes no estimation. Between frames T and $2T$ it can make a suboptimal estimation, and from frame $2T$ it can make the reported estimate - in real time - with a T frame delay. In our results we chose a value of T as 64 frames, or approximately 0.64 s.

We compare the u-net with conventional CNNs to demonstrate its benefits and drawbacks. A 4 layer CNN with Batch Normalization and Leaky ReLU activations is used for comparison. One-dimensional convolution is performed along the time dimension, with sliding windows identical to those of our u-net. The input/output channels of the 4 layers are as: (1) in: $T \times N$, out: 40; (2) in: 40, out: 20; (3) in: 20, out: 10; and (4) in: 10, out: 3.

We conducted the following experiments and discuss their results later in this section:

- **All data vs specialized dataset.** We investigate use of a more specialized dataset than the whole set (ALL), expecting that this would result in a better performance for selected motion types. To this end, we produce a reduced dataset containing only assets that are either *Run*, *Walk*, or *Idle* (RWI).

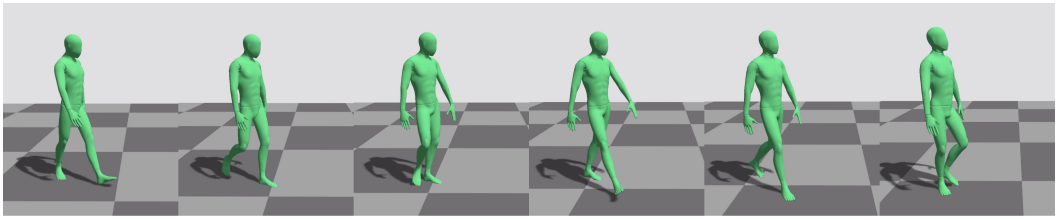

Fig. 5. This walk motion shows solid foot plants for this walking data that does not need any clean up as it is well represented in our database and the u-net is able to predict the motion with minimal error.

- **Absolute vertical vs vertical displacement.** An inherent problem with estimating displacement is that integration is needed. Any bias in the estimation, however small, will therefore eventually grow without bounds. In this experiment, we estimated the absolute height of the character against vertical displacements to avoid drift on the vertical axis.
- **Center of mass vs root position estimation.** We proposed the use of the CM in the estimation as a more stable signal than the root position for global displacement prediction. We compare the two to ascertain the impact of this decision.

To improve the visual quality of the results, we optionally post process the results with inverse kinematics (as noted in the video). To this end, foot ground contact labels are extracted from the motion using a method based on Lee et al. [2002], comparing each foot joint's positions and velocities against predefined thresholds. Ground contact labels are subsequently cleaned using thresholds to avoid false positives and false negatives by removing short contacts less than 5 frames and filling gaps between contacts that are less than 15 frames. Using the contact labels, feet target positions are generated. For frames where contact is detected, the target is set to a static point where the horizontal plane coordinates correspond to the position of the foot, and vertical coordinate corresponds to the previously determined floor height. Cubic Hermite spline interpolation is used to blend in and out from this contact point. Finally, the feet are made to track the target positions using analytical two joint inverse kinematics. Note that this is not a full foot skate cleanup, but provides a moderate improvement to our final results.

4.1 Validation

The typical training times for all experiments were in the order one to two weeks. Figure 4 shows the validation loss for both conventional nets and u-nets, and serves as a sanity check that training converges in all cases. Table 1 provides an overview of the error statistics of the different experiments. The u-net version of the ALL dataset outperformed the other scenarios. Mean errors result in drift when integration is performed. While all models had some degree of mean error, albeit low, our u-net performed best for all parameters that contribute to drift, with mean errors in the sub millimeter range. We did see a larger mean error in the vertical direction, however this does not contribute to any drift as these estimates are not integrated. Observe how the standard deviation is always an order of magnitude larger than the mean error, indicating that the errors are predominantly local and less systematic.

To take a closer look at the somewhat large mean errors of the height estimates, we refer to the plots in Figure 6. Here probability density functions are shown for the errors of both networks. We can see that while the main body of errors for the u-net model in the forward direction is centered around zero, the errors from the CNN model are more biased. For the vertical axis we see a bias in

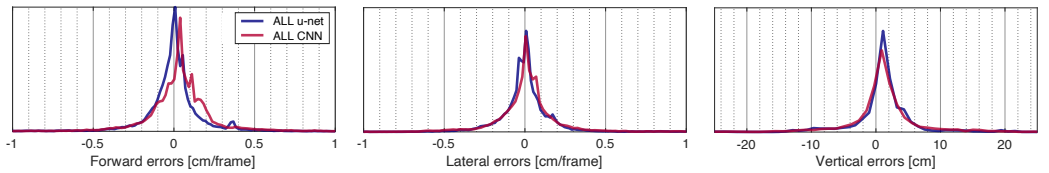

Fig. 6. Probability density functions showing the distribution of the per frame error on each axis for the all data experiment.

Table 1. Comparison between different networks and datasets, either ALL, or run, walk and idle (RWI). The error mean μ and standard deviation σ is shown for forward, lateral, and vertical directions as denoted by subscripts. All units are cm per frame except for those where the vertical output is an absolute position estimate, in which case the units are cm.

Model	Data	μ_f	μ_l	μ_v	σ_f	σ_l	σ_v
u-net	ALL	**-0.004**	**0.002**	**-4.062**	**0.191**	**0.185**	**23.123**
CNN	ALL	0.034	0.007	-4.202	0.211	0.219	24.274
u-net	RWI	-0.026	0.034	-5.886	3.331	0.203	29.010
CNN	RWI	-0.017	-0.040	-14.582	3.349	0.303	78.551

the distribution of both models, however this bias is significantly smaller than μ_v in Table 1 and has an opposite sign.

4.2 Robustness and generalizations

For a comprehensive view of what our results look like in final render we refer readers to the supplementary video. However, the snapshots in Figure 5 give an impression of what a walk motion predicted by our u-net looks like. Note the solid foot positioning as an indicator that the global position is estimated with high precision and minimal visible drift.

A comparison between the walking results of the u-net model and the CNN model can be seen in Figure 7. The trajectory estimated by both models as well as the reference trajectory can be seen. The u-net predicts estimates very close to the original motion while the CNN estimate displays significant drift over the range of motion.

A more complex motion asset of a character's motion is plotted in Figure 8 (for the u-net alone). The character is initially standing still, then, at approximately frame 130, starts accelerating to full-speed walking, reached around frame 350. The character maintains a constant cyclic walking motion for a duration, until frame 2050 where the character decelerates and come to full stand still. Our system proves capable of accurately reconstructing the entire motion sequence. Both displacements in the horizontal plane as well as the position on the vertical axis are estimated to a high precision.

4.3 Ablation studies

Figure 9 shows plots of a character's height during a run together with the absolute height estimates and integrated displacement estimates (both u-net). A natural approach to train a network to predict motion would be to make it learn the same type of parameter on all axes. However, the kinematic constraints present in the human morphology and an assumption of ground contact

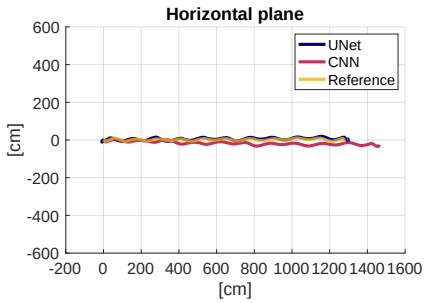

Fig. 7. Top view of a trajectory of a character walking in a straight line. Notice how the u-net estimate is close to the reference while the CNN shows significant drift.

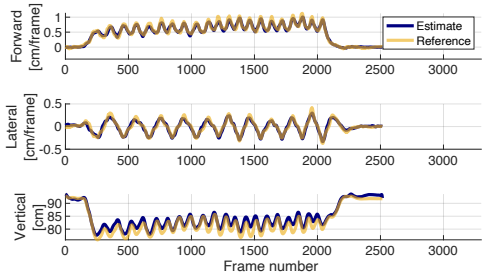

Fig. 8. Horizontal displacement and vertical position estimates for a compound motion using the u-net architecture. Observe how the system is able to estimate standstill as well as cyclic motion, acceleration and deceleration in all axes.

justify attempting to estimate absolute vertical positions instead. We compared the performance in both scenarios. While the absolute height estimate follows the reference trajectory, the integrated displacements introduce a downward drift caused by biased estimations of the displacements as the figure reveals.

To support our choice of the CM as the estimation value, we trained two networks: one predicting the global position of the root and one predicting that of the CM. We evaluate the two by reconstructing the root position of the character using the estimate of the CM and compare it with the directly estimated root position. In support of our proposed approach, we found that the root reconstruction from the CM network produces quantitatively better results in general. We show a representative plot in Figure 10. It should be noted that even though the root estimation seems to outperform the CM when it comes to the vertical axis prediction, these prediction errors are not integrated; i.e., the vertical *position* error does not grow in time. The forward and lateral predictions are velocities, which must be integrated in order to get a final position.

In the case of the animation in Figure 10, the center of mass predictions follow the reference more closely in several places in the motion for the lateral and forward displacement while it shows a constant offset with respect to the reference for the vertical direction. When integrating the forward displacements of this animation, the positional error at the end of the motion is 1.5 cm for the CM while it is 58 cm for the root. In comparison, the difference in mean absolute error of the vertical estimation is 3.4 cm for the CM prediction versus 1.6 cm for the root predictions. In light of this, we deem the CM estimation to be a better predictor than the root overall.

We highlight how the system performs differently for the RWI training motion in comparison to training on ALL the data in Figure 11. Here we showcase two assets, one a run and the second a dance. The first motion is that of a character accelerating from standstill to a running motion, and then after a period of constant motion slowing down again to end in standstill. We note the specialized model trained with the RWI dataset performs slightly better for running. For the second, a dancing motion, a similar comparison is performed. We note that for this motion the model trained with the ALL data performs better. In aggregate, from Table 1 we see that the networks trained using the ALL data set outperform the specialized RWI trained networks. From this result, we conclude that training with more data is generally preferable over training with specialized data. More experimentation would likely benefit specific applications.

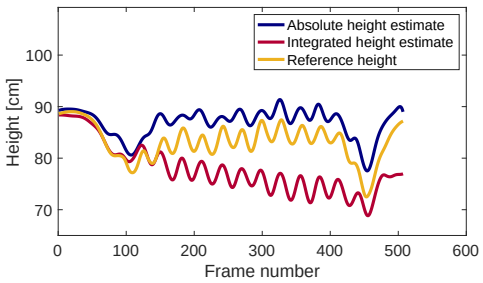

Fig. 9. Comparison between (u-net) estimates of the absolute height and height by integrating displacements. Note the drift in the integrated result due to the accumulation of the error.

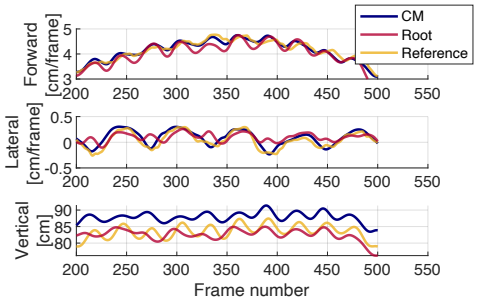

Fig. 10. Comparison between center of mass and root as the estimation target. The center of mass estimation has been projected back to the root (pelvis). Note that the unit of the forward and lateral predictions is cm/frame while the vertical prediction is in cm.

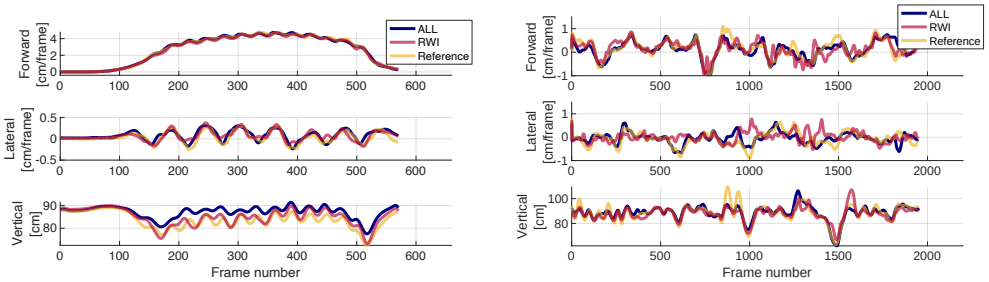

Fig. 11. Comparison between a u-net trained using the ALL data set and a u-net trained using the more specialized RWI data set. The left plot shows a character running. Notice the larger error in the estimation of the ALL trained model, especially in the vertical prediction. The right plot shows a character dancing, which is a motion type not available in the RWI data set. Observe how the RWI trained model has more difficulty predicting the motion, for example, in the lateral motion around frame 1000.

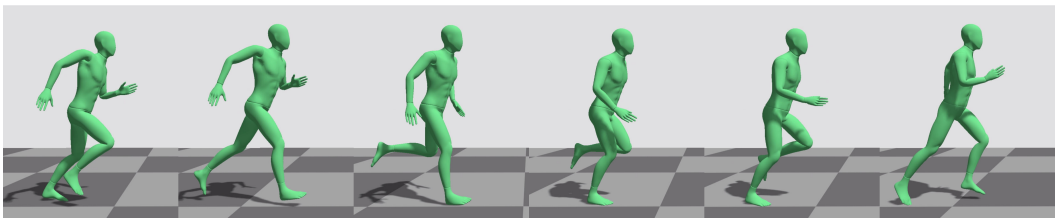

Fig. 12. Running motion with a flight phase is particularly difficult for heuristic based solutions. Our approach can estimate good lateral motion, as exhibited by the lack of foot skate, and predicts the vertical trajectory that is nearly imperceptible from ground truth. Please refer to the video for this and other comparisons.

Running motion is typically a difficult type of motion for heuristic methods due to the highly dynamic nature of the run cycle, which includes a flight phase, making it difficult to predict displacements even if proper foot contacts are determined. In Figure 12, snapshots of a running motion are shown where the displacement and height were predicted using our method. Note how

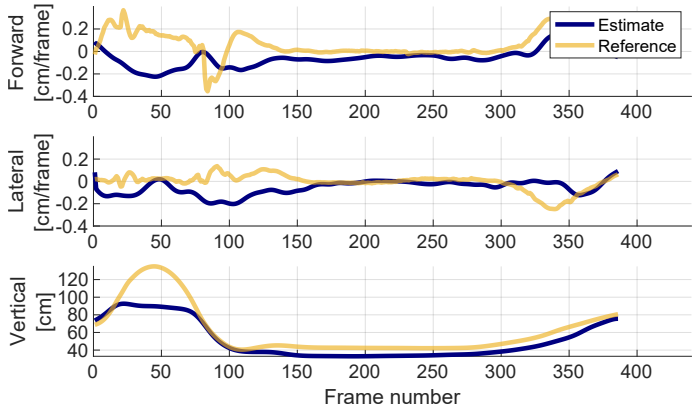

Fig. 13. Estimation plot of a character jumping at time $t = 0$. Note that the network is unable to track the height as the character lifts off but recovers as soon as the character touches down again.

the right foot does not move with respect to the tile borders once it is planted, indicating a solid stance phase (see also the supplementary video). Note, no inverse kinematics clean up is performed to obtain this result.

4.4 IMU data evaluation

We use real IMU based motion capture data recorded with Rokoko's Smartsuit Pro, to compare our trained u-net to Rokoko's heuristic approach (in the video). The supplemental video also shows position reconstruction for other motions recorded using an IMU suit. While our network was trained on optical motion capture data, we show we can also reconstruct global positions for motion captured with the IMU suit. We do note, however, that the raw IMU pose data was poorer quality in general and had larger errors, assumably because the IMU error accumulates at all joints from the root to the extremities.

5 DISCUSSION AND CONCLUSION

Our experiments show that a trained network can reasonably estimate the position of a humanoid character in a global coordinate frame interactively from local joint angle and orientation data alone. We propose to apply this estimation method for IMU motion capture but feel it is particularly valuable for interactive applications where position is needed, especially as it runs at faster than real-time rates. Within the scope of our investigations, we compare standard CNNs with the u-net architecture, and found that u-nets are capable of estimating motion with a higher level of precision. Notably the u-net both drifted significantly less and produced smaller per-frame errors in comparison to the CNN. In addition, our approach was able to eliminate drift in the vertical dimension by estimating an absolute position, instead of displacement, thereby eliminating the need for integration. While this relies on the assumption that motion appears on a horizontal plane, it could also be reverted to the displacement (with errors as reported) if elevation change is required for an application.

We compare networks trained using test data from a wide span of motion with models trained on a more specialized dataset. We found that the specialized network was slightly better at predicting motions similar to those present in the specialized training data. We see this as useful for in-depth studies where the motion type is known in advance, and the luxury of a dedicated dataset is justified. In contrast, when the trained system is presented with novel data, such as that of a

character dancing, the specialized model performed less well than the one trained on the more general dataset. From this we conclude that the model trained on the broader dataset is likely able to estimate new motion types better. This experimentation also spurs the pursuit of more complex systems where clusters of networks might compete for optimal prediction in future work.

We observe that our method fails in cases where the character is not in physical contact with the world over a prolonged period of time. The results for one such asset is plotted in Figure 13. In this asset, a character starts with a jump. The model initially tracks the motion until the feet lift off the ground and the model poorly estimates the motion in the air. Once the characters feet touch down, an accurate prediction is restored. As the capacity to estimate positions depends on the presence of like data in the database, in order to estimate general free fall, the model would need to incorporate additional training data or perhaps a model of the dynamics itself. For example, with parameters for gravity and momentum, we believe reconstructed flight phases could be improved. Incorporating these types of dynamic constraints within this scope is, to the best of our knowledge, unexplored and an additional interesting topic of future work.

Our method is meant to be used in real-time in order to get an on-the-fly estimate of global displacement. To this regard execution time has to be fast enough to run concurrently with a motion capture solution. Our solution is light enough to execute at 200 fps on a modern CPU, far faster than the 100 fps it is designed for. This is without any hardware acceleration such as GPU or code optimisations implemented, leaving room for far higher frame rates. Due to the windowing of data, estimates do include a short delay of less than one second.

Returning to our original motivation, we show that our solution is capable of estimating global placement for data from IMU systems alone but note the output quality is lower than of the optical examples, which is not surprising since the input data to the network is also lower quality. We expect that introducing IMU data to the training will improve the performance for this type of data. This is a key direction to make this approach practical for future commercialization with IMUs. However, as is, our solution still represents a significant step forward in the potential for global positioning from joint angle and orientation data, especially in real-time applications.

ACKNOWLEDGMENTS

This project was funded in part by Innovationsfonden Danmark (grant no. 9065-00085A) and NSERC Discovery. This project was conceived at the 2020 Bellairs Workshop on Computer Animation. We would like to thank Paul Kry for organizing the workshop and all participants for inspiring discussions. We thank Rasmus Lindhardt Møller for his help with the suplementary video. Finally, we would like to express our gratitude to Matias Søndergaard and Rokoko Electronics for providing us with access to the data from Rokoko's Motion Library.

REFERENCES

S. Biswal, J. Kulas, H. Sun, B. Goparaju, M. B. Westover, M. T. Bianchi, and J. Sun. 2017. SLEEPNET: automated sleep staging system via deep learning. *arXiv preprint arXiv:1707.08262* (2017).

J. P. Chiu and E. Nichols. 2016. Named entity recognition with bidirectional LSTM-CNNs. *Transactions of the Association for Computational Linguistics* 4 (2016), 357–370.

J. A. Corrales, F. Candelas, and F. Torres. 2008. Hybrid tracking of human operators using IMU/UWB data fusion by a Kalman filter. In *2008 3rd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 193–200.

J. L. Coyte, D. Stirling, M. Ros, H. Du, and A. Gray. 2013. Displacement profile estimation using low cost inertial motion sensors with applications to sporting and rehabilitation exercises. In *2013 IEEE/ASME International Conference on Advanced Intelligent Mechatronics*. IEEE, 1290–1295.

R. Dumas and J. Wojtusch. 2017. *Estimation of the Body Segment Inertial Parameters for the Rigid Body Biomechanical Models Used in Motion Analysis*. 1–31.

B. Fasel, J. Spörri, J. Chardonnens, J. Kröll, E. Müller, and K. Aminian. 2017. Joint inertial sensor orientation drift reduction for highly dynamic movements. *IEEE journal of biomedical and health informatics* 22, 1 (2017), 77–86.

M. J. Floor-Westerdijk, H. M. Schepers, P. H. Veltink, E. H. van Asseldonk, and J. H. Buurke. 2012. Use of inertial sensors for ambulatory assessment of center-of-mass displacements during walking. *IEEE transactions on biomedical engineering* 59, 7 (2012), 2080–2084.

E. Foxlin. 2005. Pedestrian tracking with shoe-mounted inertial sensors. *IEEE Computer graphics and applications* 25, 6 (2005), 38–46.

K. Fragkiadaki, S. Levine, P. Felsen, and J. Malik. 2015. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*. 4346–4354.

P. Ghosh, J. Song, E. Aksan, and O. Hilliges. 2017. Learning human motion models for long-term predictions. In *2017 International Conference on 3D Vision (3DV)*. IEEE, 458–466.

A. Graves, N. Jaitly, and A.-r. Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *2013 IEEE workshop on automatic speech recognition and understanding*. IEEE, 273–278.

T. Hastie, R. Tibshirani, and J. Friedman. 2009. *The elements of statistical learning: data mining, inference, and prediction*. Springer Science & Business Media.

D. Holden, T. Komura, and J. Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.

D. Holden, J. Saito, and T. Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.

D. Holden, J. Saito, T. Komura, and T. Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*. 1–4.

H. Ju, M. S. Lee, S. Y. Park, J. W. Song, and C. G. Park. 2015. A pedestrian dead-reckoning system that considers the heel-strike and toe-off phases when using a foot-mounted IMU. *Measurement Science and Technology* 27, 1 (2015), 015702.

M. Kok, J. D. Hol, and T. B. Schön. 2017. Using Inertial Sensors for Position and Orientation Estimation. *CoRR* abs/1704.06053 (2017).

M. Lapinski, E. Berkson, T. Gill, M. Reinold, and J. A. Paradiso. 2009. A distributed wearable, wireless sensor system for evaluating professional baseball pitchers and batters. In *2009 International Symposium on Wearable Computers*. IEEE, 131–138.

J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard. 2002. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 491–500.

T. Li, L. Wang, Q. Li, and T. Liu. 2020. Lower-body walking motion estimation using only two shank-mounted inertial measurement units (IMUs). In *2020 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)*. IEEE, 1143–1148.

J. Martinez, M. J. Black, and J. Romero. 2017. On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2891–2900.

D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, and C. Theobalt. 2017. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*. IEEE, 506–516.

A. Menache. 2011. *Understanding motion capture for computer animation*. Elsevier.

R. Pascanu, T. Mikolov, and Y. Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*. PMLR, 1310–1318.

D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli. 2019. 3d human pose estimation in video with temporal convolutions and semi-supervised training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 7753–7762.

M. Perslev, S. Darkner, L. Kempfner, M. Nikolic, P. J. Jennum, and C. Igel. 2021. U-Sleep: resilient high-frequency sleep staging. *npj Digital Medicine* 4, 1 (2021), 1–12.

M. Perslev, M. Jensen, S. Darkner, P. J. r. Jennum, and C. Igel. 2019. U-Time: A Fully Convolutional Network for Time Series Segmentation Applied to Sleep Staging. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc., 4415–4426.

D. Roetenberg, H. Luinge, and P. Slycke. 2009. Xsens MVN: Full 6DOF human motion tracking using miniature inertial sensors. *Xsens Motion Technologies BV, Tech. Rep* 1 (2009).

O. Ronneberger, P. Fischer, and T. Brox. 2015. U-Net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

L. A. Schwarz, D. Mateus, and N. Navab. 2012. Recognizing multiple human activities and tracking full-body pose in unconstrained environments. *Pattern Recognition* 45, 1 (2012), 11–23.

S. Shimada, V. Golyanik, W. Xu, and C. Theobalt. 2020. Physcap: Physically plausible monocular 3d motion capture in real time. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–16.

T. Shiratori, H. S. Park, L. Sigal, Y. Sheikh, and J. K. Hodgins. 2011. Motion Capture from Body-Mounted Cameras. *ACM Trans. Graph.* 30, 4, Article 31 (July 2011), 10 pages.

R. Slyper and J. K. Hodgins. 2008. Action capture with accelerometers. In *Symposium on Computer Animation*. 193–199.

S. Starke, Y. Zhao, T. Komura, and K. Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 54–1.

Y. S. Suh. 2014. Inertial sensor-based smoother for gait analysis. *Sensors* 14, 12 (2014), 24338–24357.

D. Thewlis, C. Bishop, N. Daniell, and G. Paul. 2013. Next-generation low-cost motion capture systems can provide comparable spatial accuracy to high-end systems. *Journal of applied biomechanics* 29, 1 (2013), 112–117.

M. Véges and A. Lőrincz. 2019. Absolute human pose estimation with depth prediction network. In *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–7.

D. Vlasic, R. Adelsberger, G. Vannucci, J. Barnwell, M. Gross, W. Matusik, and J. Popović. 2007. Practical motion capture in everyday surroundings. *ACM transactions on graphics (TOG)* 26, 3 (2007), 35–es.

H. Wang, E. S. Ho, H. P. Shum, and Z. Zhu. 2019. Spatio-temporal manifold learning for human motions via long-horizon modeling. *IEEE transactions on visualization and computer graphics* 27, 1 (2019), 216–227.

P. A. C. Widagdo, H.-H. Lee, and C.-H. Kuo. 2017. Limb motion tracking with inertial measurement units. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 582–587.

X. Xiao and S. Zarar. 2018. Machine learning for placement-insensitive inertial motion capture. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 6716–6721.

Q. Yuan, I.-M. Chen, and S. P. Lee. 2011. SLAC: 3D localization of human based on kinetic human movement capture. In *2011 IEEE International Conference on Robotics and Automation*. IEEE, 848–853.

H. Zhang, S. Starke, T. Komura, and J. Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics (TOG)* 37, 4 (2018), 1–11.

Y. Zheng, K.-C. Chan, and C. C. Wang. 2014. Pedalvatar: An IMU-based real-time body motion capture system using foot rooted kinematic model. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 4130–4135.

X. Zhou, M. Zhu, G. Pavlakos, S. Leonardos, K. G. Derpanis, and K. Daniilidis. 2018. Monocap: Monocular human motion capture using a cnn coupled with a geometric prior. *IEEE transactions on pattern analysis and machine intelligence* 41, 4 (2018), 901–914.

Y. Zhou, J. Lu, C. Barnes, J. Yang, S. Xiang, et al. 2020. Generative tweening: Long-term inbetweening of 3d human motions. *arXiv preprint arXiv:2005.08891* (2020).