ELMO: Enhanced Real-time LiDAR Motion Capture through Upsampling

DEOK-KYEONG JANG*, MOVIN Inc., South Korea
DONGSEOK YANG*, MOVIN Inc., South Korea
DEOK-YUN JANG*, MOVIN Inc., South Korea
BYEOLI CHOI*, MOVIN Inc., South Korea
DONGHOON SHIN, MOVIN Inc., South Korea
SUNG-HEE LEE†, KAIST, South Korea

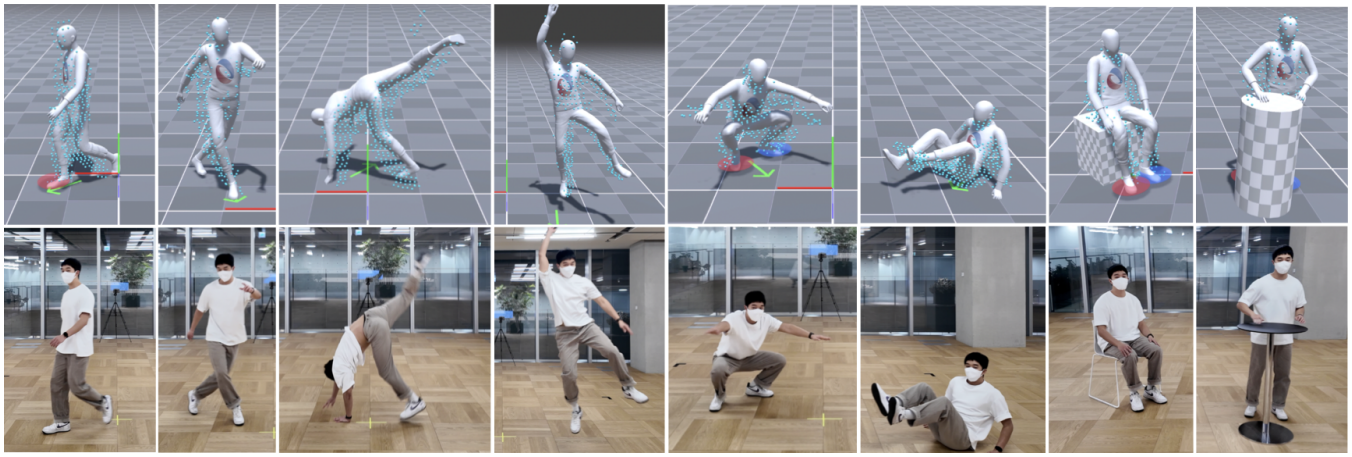

Fig. 1. Our ELMO framework enables upsampling motion capture (60 fps) from LiDAR point cloud (20 fps) in real-time.

This paper introduces ELMO, a real-time upsampling motion capture framework designed for a single LiDAR sensor. Modeled as a conditional autoregressive transformer-based upsampling motion generator, ELMO achieves 60 fps motion capture from a 20 fps LiDAR point cloud sequence. The key feature of ELMO is the coupling of the self-attention mechanism with thoughtfully designed embedding modules for motion and point clouds, significantly elevating the motion quality. To facilitate accurate motion capture, we develop a one-time skeleton calibration model capable of predicting user skeleton offsets from a single-frame point cloud. Additionally, we introduce a novel data augmentation technique utilizing a LiDAR simulator, which enhances global root tracking to improve environmental understanding. To demonstrate the effectiveness of our method, we compare ELMO with state-of-the-art methods in both image-based and point cloud-based motion capture. We further conduct an ablation study to validate our design principles. ELMO's fast inference time makes it well-suited for real-time applications, exemplified in our demo video featuring live streaming and interactive gaming scenarios. Furthermore, we contribute a high-quality LiDAR-mocap synchronized dataset comprising 20 different subjects performing a range of motions, which can serve as a valuable resource for future research. The dataset and evaluation code are available at https://movin3d.github.io/ELMO_SIGASIA2024/

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

Additional Key Words and Phrases: Motion capture, Motion synthesis, Character animation, Point cloud, Deep learning

ACM Reference Format:
Deok-Kyeong Jang, Dongseok Yang, Deok-Yun Jang, Byeoli Choi, Donghoon Shin, and Sung-Hee Lee. 2024. ELMO: Enhanced Real-time LiDAR Motion Capture through Upsampling. *ACM Trans. Graph.* 43, 6, Article 237 (December 2024), 14 pages. https://doi.org/10.1145/3687991

1 INTRODUCTION

With the increasing need for virtual avatars in 3D content creation and metaverse platforms, real-time motion capture technology is experiencing a notable surge in demand. Despite the strides made by inertial and optical sensors-based solutions with great accuracy, they remain costly and cumbersome for average consumers. Various approaches address this limitation by reducing the number of body-worn sensors [Von Marcard et al. 2017] and exploring more accessible devices such as RGB cameras [Wei and Chai 2010]. However, RGB cameras and inertial sensors lack explicit information on global translation, leading to drifting in the output. Research utilizing depth sensors, such as RGB-D cameras [Zollhöfer et al. 2014] and LiDARs [Jang et al. 2023a], demonstrates enhanced global

*Equal contribution.
†Corresponding author

Authors' addresses: Deok-Kyeong Jang, dk.jang1014@gmail.com, MOVIN Inc., Seoul, South Korea; Dongseok Yang, ds.yang@movin3d.com, MOVIN Inc., Seoul, South Korea; Deok-Yun Jang, dy.jang@movin3d.com, MOVIN Inc., Seoul, South Korea; Byeoli Choi, by.choi@movin3d.com, MOVIN Inc., Seoul, South Korea; Donghoon Shin, dh.shin@movin3d.com, MOVIN Inc., Seoul, South Korea; Sung-Hee Lee, sunghee.lee@kaist.ac.kr, KAIST, Daejeon, South Korea.

ACM 0730-0301/2024/12-ART237
https://doi.org/10.1145/3687991

tracking. On the other hand, the measurements from depth cameras suffer from noise and LiDARs' low framerate introduces pose discontinuities in real-time applications, typically requiring higher framerates over 60 frames per second (fps).

To address these limitations, we propose ELMO, a novel real-time upsampling motion capture framework designed to derive 60 fps mocap data from a 20 fps point cloud sequence captured by a single LiDAR sensor. The core concept involves using a sequence of sampled point clouds from the past to the current and one future frame to generate three upsampled poses. Our motion generator adopts a conditional autoregressive transformer-based architecture considering past inferred motion and acquired point cloud to establish the relationship between the input point cloud and the output upsampled poses. To overcome self-occlusions in single LiDAR setups, our framework includes a mechanism for sampling a latent vector from a motion prior. This vector is then processed by the generator to predict plausible poses, particularly in scenarios involving occlusions.

To effectively capture the correlation between the LiDAR point cloud and the human body joints, we design motion and point cloud embedding modules such that joint features preserve the skeletal graph structure, a root feature captures global coordinate information, and point features find characteristics of local regions in the point cloud, dubbed *body-patch groups*. Leveraging the self-attention mechanism in our transformer generator, we facilitate the learning of attention between individual body-patch groups and human joints for each embedding feature. This approach significantly enhances the quality of the output motions.

Real-time motion capture relies on accurately tracking global translation, crucial for seamless interaction between avatars and their environment or objects. Acknowledging the challenge of gathering comprehensive data across the capture space, we present a novel data augmentation technique leveraging a LiDAR simulator. We apply global rotations to each original motion clip and fit the SMPL body model [Loper et al. 2015] to compute collision points with simulated lasers. Implementing this augmentation technique on our training dataset resulted in a noticeable enhancement in the quality of the mocap results.

Furthermore, we developed a one-time skeleton calibration model that infers user skeleton offsets from a single-frame point cloud acquired while the user is in the A-pose. Skeleton calibration is a fundamental step in motion capture, determining initial joint offsets, global trajectory, and joint rotations.

Figure 1 presents snapshots of real-time mocap results from ELMO. To demonstrate the effectiveness of our framework, we conduct thorough comparisons with state-of-the-art image-based and point cloud-based methods, along with an ablation study to validate our design choices. Additionally, we conduct various experiments, such as testing for global drifting, to verify the essential elements required for accurate motion capture.

To the best of our knowledge, our work is the first real-time upsampling motion capture framework using a single LiDAR. By maintaining low latency, ELMO is well-suited for live application scenarios. Our demo video provides example use cases including live streaming and interactive gaming. The major contributions of our work can be summarized as follows:

- We present the first real-time upsampling motion capture using a single LiDAR, offering low-latency performance for diverse real-time applications.
- Our novel embedding and generator architecture effectively constructs attention maps between body-patch point groups and human joints, enabling precise upsampling in motion capture. Additionally, we propose a one-time skeleton calibration model to predict user skeleton offsets from a single-frame point cloud.
- We introduce a new LiDAR-simulation-based data augmentation technique that leverages the unique characteristics of LiDAR sensors to enhance global translation tracking performance. Furthermore, we release a high-quality LiDAR-mocap synchronized dataset featuring 20 subjects performing various actions.

2 RELATED WORK

2.1 Motion Capture

Optical and inertial systems stand out in the professional market for their high accuracy. However, a shared challenge across these mocap techniques is body-worn sensors that may restrict user motion or shift from their initial positions. Additionally, a time-consuming setup and calibration are required for the quality of captured data. A prominent research focus involves reducing the number of sensors and reconstructing full-body motion from a sparse setup [Huang et al. 2018; Jiang et al. 2022; Lee et al. 2023b; Ponton et al. 2023; Winkler et al. 2022; Yang et al. 2024]. While more accessible than previous methods, they still grapple with inherent issues of wearable sensors.

Consequently, markerless methods [Aguiar et al. 2008; Bregler and Malik 1998] have garnered significant attention for their notable advancement in eliminating the need for body-worn sensors. Moreover, they enhance the accessibility of mocap by using widely available devices such as webcams and RGB cameras. Simultaneously, research is underway on both mono [Bazarevsky et al. 2020; Bogo et al. 2016; Huang et al. 2022; Kocabas et al. 2020; Kolotouros et al. 2019; Pavlakos et al. 2017; Shetty et al. 2023; Wei et al. 2022; Ye et al. 2023; Zhu et al. 2022] and multi-view camera [Amin et al. 2013; Burenius et al. 2013; Dong et al. 2022] methods. Mono-camera setups offer simplicity while multi-view systems excel in accuracy. While offering supplementary depth information, RGBD solutions [Baak et al. 2011; Mehta et al. 2017; Ying and Zhao 2021] face challenges due to their limited field of view (FoV) and resolutions compared to RGB cameras. This leads to noisy and unstable output poses.

2.2 LiDAR-based Pose Tracking

The latest image-based human pose tracking methods [Goel et al. 2023; Li et al. 2023a] have shown impressive quality through a two-step process involving 2D keypoint extraction and SMPL body model [Loper et al. 2015] fitting. However, the accuracy of 3D keypoints and global translation is compromised during pose fitting in 2D image space.

A promising solution is using LiDAR sensors, which provide accurate 3D point cloud data. This method also offers a comprehensive view of the subject's full-body information, not possible with sparse sensor setups. A seminal study by Li et al. [Li et al. 2022b]

has demonstrated that LiDAR sensors can enhance the quality of captured poses from distances. The following research explores the fusion of LiDAR with IMUs [Ren et al. 2023] and RGB cameras [Cong et al. 2023] as a complement for capturing detailed 3D human poses. Recently, Human3D [Takmaz et al. 2023] has demonstrated remarkable performance in body part segmentation, relying solely on LiDAR point cloud data. Similarly, MOVIN [Jang et al. 2023a] performed skeletal motion capture with a single LiDAR sensor. Our approach addresses the limitations present in prior works, such as motion jitters and a low frame rate, enhancing the feasibility of LiDAR motion capture frameworks for real-time applications.

2.3 Neural Generative Models for Human Motion

Generating natural human motion while minimizing laborious and time-consuming tasks has been among the central focuses in the field of computer graphics. Upon the widespread integration of deep neural networks, researchers developed technologies to generate human motion from various inputs, encompassing low-dimensional control signals, navigation goals, and text prompts.

Within the realm of neural networks, generative models such as GANs [Goodfellow et al. 2014] and VAEs [Kingma and Welling 2013] have demonstrated notable success in producing high-quality, natural motion. GANs find application in diverse areas, including character control [Wang et al. 2021], speech-driven gesture generation [Ferstl et al. 2019], and motion generation from a single clip [Li et al. 2022a]. Conditional VAEs [Sohn et al. 2015] are employed in motion generation methods that utilize additional constraints such as past motion sequences [Ling et al. 2020], motion categories [Petrovich et al. 2021], and speech [Lee et al. 2019; Li et al. 2020]. Recent works have focused on creating a motion embedding space [Lee et al. 2023a] and leveraging the latent space for tasks like motion in-betweening [Tang et al. 2023], retargeting [Li et al. 2023b], and style transfer [Jang et al. 2023b]. An alternative approach involves using normalizing flow [Aliakbarian et al. 2022; Henter et al. 2020], enabling exact maximum likelihood. In parallel, building on recent advancements in diffusion models, researchers have extended their application to language [Tevet et al. 2023; Zhang et al. 2022] and music [Tseng et al. 2023]-driven motion synthesis. Methods utilizing deep reinforcement learning frameworks also incorporate VAEs to establish prior distributions for character control [Won et al. 2022], skills [Dou et al. 2023], unstructured motion clips [Zhu et al. 2023], and muscle control [Feng et al. 2023].

3 ELMO FRAMEWORK

Figure 2 (b) illustrates our upsampling motion capture framework from a single-front LiDAR sensor at runtime. Our framework effectively transforms 20fps LiDAR point cloud input into 60fps motion in real-time, with a latency of one 20fps frame[1]. From the inference point of view, the input to our framework is a previously inferred sequence of 60 frames (1 second) of motion, from past frame $i-(6s-1)$ to current frame i (where $s = 12$), as well as a sequence of sampled point clouds from the past to the current at timestamps $i-5s$, $i-4s$, $i-3s$, $i-2s$, $i-s$, i, and a newly captured point cloud at a future frame $i+3$. For the output, it generates the next three upsampled poses for frames $i+1$, $i+2$, and $i+3$ at a rate of 60fps.

Our model is based on an autoregressive conditional transformer-based architecture. At each time frame, we initially extract the features of joints, the root, and points. During the training phase, given the motion sequence as input, the motion distribution encoder E generates a latent vector z, which is trained to shape the latent variable z into a Gaussian distribution (Figure 3). At the same time, the motion generator G takes the past and current motion sequence, a sampled point cloud sequence of past to one future frame, and the latent vector z as inputs, and generates upsampled poses of the three future frames. During inference, the encoder E is discarded. Instead, we pass a randomly sampled latent vector z through the motion generator at each time frame, enabling the generation of plausible poses.

3.1 Data Representation

The input and output of our framework consist of the 3D point cloud and poses. We start by defining a pose vector for a single frame i, which contains a joint vector and root vector. Following [Jang et al. 2023a], which suggests the importance of accurate global (world) coordinate information of the character's root for high-quality motion capture, we handle the joint vector and root vector distinctly. The joint vector $\mathbf{x}_i$ includes all joint local information including joint local positions, rotations, velocity, and angular velocity with respect to the parent joint, denoted as $\mathbf{x}_i = [x^t, x^r, \dot{x}^t, \dot{x}^r] \in \mathbb{R}^{n_j \times 15}$, where $x^t \in \mathbb{R}^{n_j \times 3}$, $x^r \in \mathbb{R}^{n_j \times 6}$, $\dot{x}^t \in \mathbb{R}^{n_j \times 3}$, and $\dot{x}^r \in \mathbb{R}^{n_j \times 3}$. Here, n_j denotes the number of joints. The root vector $\mathbf{r}_i$ represents the global coordinates of each character, including the character's global root position, rotation, velocity, and angular velocity. Additional vectors, such as foot contact, are also incorporated. Specifically, we define the root vector as $\mathbf{r}_i = [r^t, r^r, \dot{r}^t, \dot{r}^r, c] \in \mathbb{R}^{17}$, where $r^t \in \mathbb{R}^3$, $r^r \in \mathbb{R}^6$, $\dot{r}^t \in \mathbb{R}^3$, $\dot{r}^r \in \mathbb{R}^3$, and $c \in \mathbb{R}^2$ for foot contact label. Lastly, the input point cloud vector, representing the global position of human body points, is denoted as $\mathbf{p}_i \in \mathbb{R}^{n_p \times 3}$, where n_p is the number of points.

Note that the captured motion is 60 fps and the LiDAR input is 20 fps, to prevent confusion, all frames described from now on are in 60 fps.

3.2 Motion and Point Cloud Embedding

The embedding process consists of two components: motion embedding and point embedding. The former extracts the spatial and temporal features of the input motion sequence, while the latter groups each point cloud into several local body-patch features.

Motion embedding. For motion, we utilize distinct embeddings for joint features and root features, as shown in the left part of Figure 2 (a). For embedding inputs at frame i, we consider frames from $i-(s-1)$ to i, denoted as $[\mathbf{x}_i]^i_{i-(s-1)} = [\mathbf{x}_{i-(s-1)}, \ldots, \mathbf{x}_i] \in \mathbb{R}^{s \times n_j \times 15}$ for joint vectors and $[\mathbf{r}_i]^i_{i-(s-1)} = [\mathbf{r}_{i-(s-1)}, \ldots, \mathbf{r}_i] \in \mathbb{R}^{s \times 17}$ for root vectors. Note that we handle motions of length s because our approach involves utilizing point cloud sequences sampled at the interval of s as inputs.

[1]Since the input LiDAR operates at 20 fps, when considering a 1 future frame latency, the timestamp will be $i+3$ from the perspective of a 60 fps system.

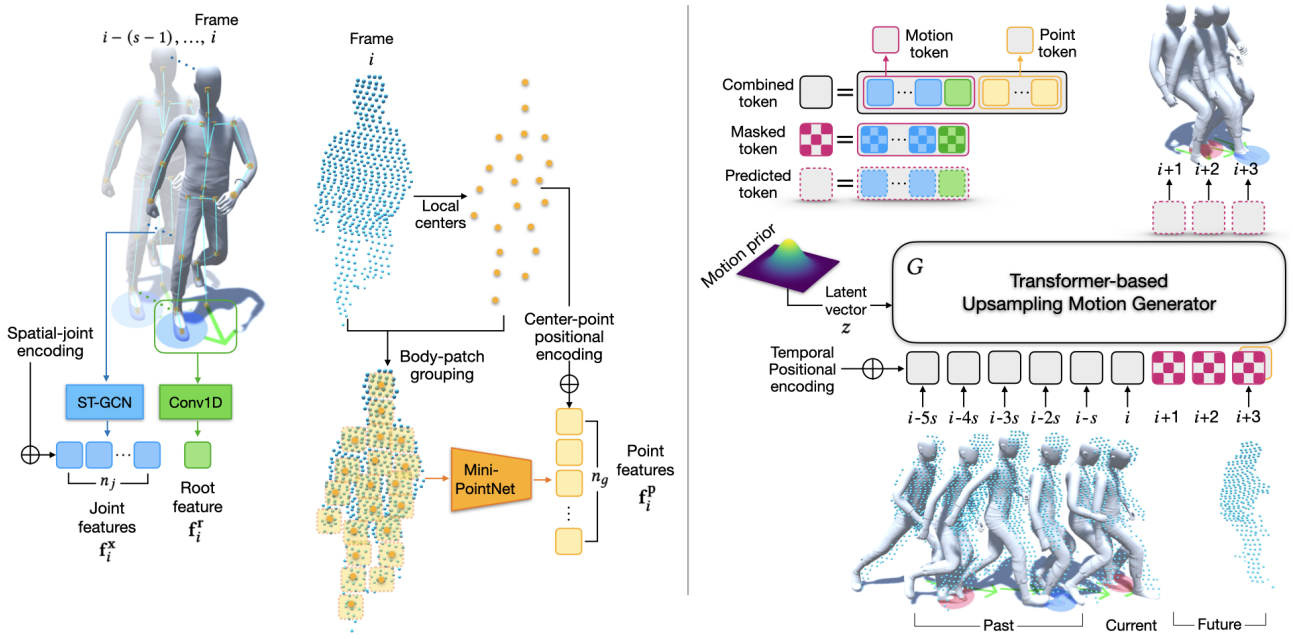

Fig. 2. Overall network architectures. (a) Detail of the feature extraction pipeline. (b) Overview of generator for real-time upsampling LiDAR motion capture in run-time.

For the joint feature embedding, we use spatial-temporal graph convolutional blocks (STGCN) [Yan et al. 2018] to maintain the local joint graph information of the human skeleton as much as possible. The STGCN blocks embed an input sequential joint vectors $[\mathbf{x}_i]_{i-(s-1)}^{i}$ into a joint features $\mathbf{f}_i^{\mathbf{x}}$ after temporal average pooling as follows:

$$\mathbf{f}_i^{\mathbf{x}} = \text{TempAvgPool}(\text{ST-GCN}([\mathbf{x}_i]_{i-(s-1)}^{i})) \in \mathbb{R}^{n_j \times C}, \quad (1)$$

where C is a feature dimension.

For the root feature, a 1D temporal convolution block is utilized to embed the sequential root vectors $[\mathbf{r}_i]_{i-(s-1)}^{i}$ to get root feature $\mathbf{f}_i^{\mathbf{r}}$ with equal feature dimension C:

$$\mathbf{f}_i^{\mathbf{r}} = \text{TempAvgPool}(\text{Conv1D}([\mathbf{r}_i]_{i-(s-1)}^{i})) \in \mathbb{R}^{C}. \quad (2)$$

After the motion embedding process, we get a motion feature $[\mathbf{f}_i^{\mathbf{x}}, \mathbf{f}_i^{\mathbf{r}}]$ at frame i, which achieves temporal alignment with the point cloud data.

Point cloud embedding by body-patch. The right part of Figure 2 (a) illustrates the points body-patch embedding strategy. Inspired by PointBERT [Yu et al. 2022], we group each point cloud into several body part patches for each frame i. Given an input point cloud of the human body $\mathbf{p}_i \in \mathbb{R}^{n_p \times 3}$, we employ farthest point sampling (FPS) to select n_g central points from the point cloud. By grouping the k-nearest neighbors (k-NN) around these central points, we create n_g distinct local point patches, essentially forming a body-patch cloud with k elements each. To make these local patches unbiased, we subtract their center coordinates. This step effectively disentangles the structural patterns from the spatial coordinates within the local patches. We then utilize Mini-PointNet [Qi et al. 2017] to project these sub-body-patch clouds into point embeddings.

The Mini-PointNet involves the following steps: Initially, each point within a patch is mapped to a feature vector via a shared multilayer perceptron (MLP). Subsequently, max-pooled features are concatenated to each feature vector. These vectors are then processed through a second shared MLP and a final max-pooling layer, resulting in the body-patch embedding. The overall point cloud embedding process to extract point features $\mathbf{f}_i^{\mathbf{p}}$ is formalized as follows:

$$\mathbf{f}_i^{\mathbf{p}} = \text{Mini-PointNet}(\text{body-patch grouping}(\mathbf{p}_i)) \in \mathbb{R}^{n_g \times C} \quad (3)$$

3.3 Upsampling Motion Generator

Our motion generator utilizes a conditional autoregressive model, processing past inferred motion and acquired point cloud data as inputs. This establishes a relationship between the current point cloud input and upsampled output poses. By leveraging self-attention within this transformer-based architecture, our approach effectively learns the attention between body-patch point group features and motion features.

Tokenization. We implement a tokenization process for input into a Transformer-based Motion Upsampling Generator, utilizing three distinct token types as illustrated in Figure 2: the Combined Token $\mathbf{T}_i^{comb}$, Masked Token $\mathbf{T}_i^{mask}$, and Predicted Token $\mathbf{T}_i^{pred}$.

The Combined Token $\mathbf{T}_i^{comb}$ integrates the Motion Token ($\mathbf{T}_i^{mot}$) and Point Token ($\mathbf{T}_i^{point}$). The Motion Token consists of joint and

root features, added with learnable spatial joint encodings. Conversely, the Point Token, representing the point features of n_g body patches, lacks positional information. Therefore, a two-layer MLP is used to assign positional encodings to each center point of the body-patch groups, which are then added to the body-patch point features:

$$\begin{aligned} \mathbf{T}_i^{mot} &= [\mathbf{f}_i^{\mathrm{x}}, \mathbf{f}_i^{\mathrm{r}}] + \mathcal{P}^{\mathrm{spat}}, \quad \mathbf{T}_i^{point} = \mathbf{f}_i^{\mathrm{p}} + \mathcal{P}^{\mathrm{cent}} \\ \mathbf{T}_i^{comb} &= [\mathbf{T}_i^{mot}, \mathbf{T}_i^{point}], \end{aligned} \tag{4}$$

where $\mathcal{P}^{\mathrm{spat}}$ denotes spatial joint encoding with learnable parameters, and $\mathcal{P}^{\mathrm{cent}}$ represents center point positional encoding for body-patch groups.

Next, the Masked Token $\mathbf{T}_i^{mask}$ is a learnable masking token added with the same spatial positional encoding $\mathcal{P}^{\mathrm{spat}}$ that, upon passing through the upsampling motion generator, becomes the Predicted Token $\mathbf{T}_i^{pred}$. The Predicted Token, or pose feature, is employed in reconstructing the upsampled poses. Notably, the Predicted Token (pose feature) corresponding to the masked token frame i represents a pose feature for a single frame, as opposed to the motion feature that deals with the temporal sequence of $i-(s-1)$ to i as used in the combined token. The pose feature also comprises joint features and the root feature.

Transformer-based generator. After the tokenization process, as illustrated in Figure 2 (b), we create an input sequence by concatenating the combined tokens $\mathbf{T}_i^{comb}$ corresponding to frames $i-5s$, $i-4s$, $i-3s$, $i-2s$, $i-s$, and i along with the Point token $\mathbf{T}_{i+3}^{point}$ for frame $i+3$. Lastly, we pad the masked tokens corresponding to future frames $i+1$, $i+2$ and $i+3$. The temporal positional encoding $\mathcal{P}^{\mathrm{temp}}$ derived from sinusoidal functions for each time frame is then added to the input sequence.

Given the sampled latent vector z, the Upsampling Motion Generator G is an autoregressive model that generates future $i+1$, $i+2$, and $i+3$ pose features (predicted token) at the target frame rate (60fps), conditioned on the input sequence. These 3 pose features are further expanded using expanding modules to obtain the joint vectors $[\tilde{\mathbf{x}}_{i+1}, \tilde{\mathbf{x}}_{i+2}, \tilde{\mathbf{x}}_{i+3}]$ and root vectors $[\tilde{\mathbf{r}}_{i+1}, \tilde{\mathbf{r}}_{i+2}, \tilde{\mathbf{r}}_{i+3}]$. We adopt the standard vision transformer for the generator G, consisting of multi-headed self-attention layers and FFN blocks. The expanding modules utilize inverse forms of the motion embedding modules. However, a difference is that the expansion is performed separately for the pose of each frame. The overall upsampling generator process is formalized as follows:

$$\begin{aligned} &[\mathbf{T}_{i+1}^{pred}, \mathbf{T}_{i+2}^{pred}, \mathbf{T}_{i+3}^{pred}] = \\ &\quad G([\mathbf{T}^{comb}, \mathbf{T}_{i+3}^{point}, \mathbf{T}_{i+1}^{mask}, \mathbf{T}_{i+2}^{mask}, \mathbf{T}_{i+3}^{mask}] + \mathcal{P}^{\mathrm{temp}}, z), \\ &[\tilde{\mathbf{x}}_{i+1}, \tilde{\mathbf{x}}_{i+2}, \tilde{\mathbf{x}}_{i+3}] = \text{De-GCN}(\left[\mathbf{T}_i^{pred}[:n_j]\right]_{i+1}^{i+3}), \\ &[\tilde{\mathbf{r}}_{i+1}, \tilde{\mathbf{r}}_{i+2}, \tilde{\mathbf{r}}_{i+3}] = \text{De-Conv1D}(\left[\mathbf{T}_i^{pred}[n_j:n_j+1]\right]_{i+1}^{i+3}), \end{aligned} \tag{5}$$

where $\mathbf{T}^{comb} = [\mathbf{T}_{i-5s}^{comb}, \mathbf{T}_{i-4s}^{comb}, \mathbf{T}_{i-3s}^{comb}, \mathbf{T}_{i-2s}^{comb}, \mathbf{T}_{i-s}^{comb}, \mathbf{T}_i^{comb}]$.

3.4 Motion Prior

The motion prior used to sample the latent vector z is constructed via the motion distribution encoder E as depicted in Figure 3. Due to self-occlusions among body parts commonly encountered by a single LiDAR sensor, the latent vector z, derived from a motion prior, assists the generator in accurately predicting plausible poses.

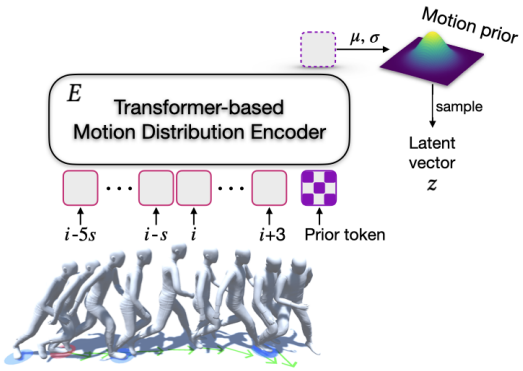

Fig. 3. Constructing the motion prior in the training phase.

To effectively capture the spatial-temporal dependencies between past and current poses, we use a transformer architecture, akin to the generator G. The motion distribution encoder E processes a learnable prior token $\mathbf{T}^{prior}$ along with concatenated motion tokens $[\mathbf{T}_{i-5s}^{mot}, \ldots, \mathbf{T}_{i+3}^{mot}]$ as its inputs. These inputs facilitate encoding the parameters of a Gaussian distribution $\mathcal{N}(\mu, \sigma)$. The reparameterization trick is then applied to transform these parameters and obtain the latent vector $z \in \mathbb{R}^C$:

$$E(z|[\mathbf{T}_{i-5s}^{mot}, \ldots, \mathbf{T}_{i+3}^{mot}] + \mathcal{P}^{\mathrm{temp}}, \mathbf{T}^{prior}) = \mathcal{N}(z; \mu, \sigma) \tag{6}$$

3.5 Point Cloud Processing

When acquiring point clouds from LiDAR in each frame, extraneous noise points that do not correspond to the human body are often captured due to the surrounding environment. Thus, as a preliminary step, we employ a *Statistical Outlier Removal* algorithm to filter out these irrelevant points. This method uses the mean distance of each point to its neighbors. Points that have a distance significantly larger than the average are considered outliers.

After filtering, our framework is designed to handle an input point cloud with $n_p = 384$ points. In cases where the number of points acquired from LiDAR exceeds 384, we apply farthest point sampling (FPS) to reduce the count. Conversely, if the number of points is fewer than 384, we randomly select points and add small noise to generate synthetic points, effectively padding the dataset to the required size.

4 TRAINING THE ELMO FRAMEWORK

We train the entire framework end-to-end, optimizing for the loss terms detailed in Section 4.2. Our training process leverages all available data, both captured and augmented using the techniques outlined in Section 4.1.

4.1 Data Augmentation

The primary goal of the augmentation is to make the training dataset cover the entire capture space (4×4 meters for our dataset acquisition and experiments), as we use the global coordinate for the root.

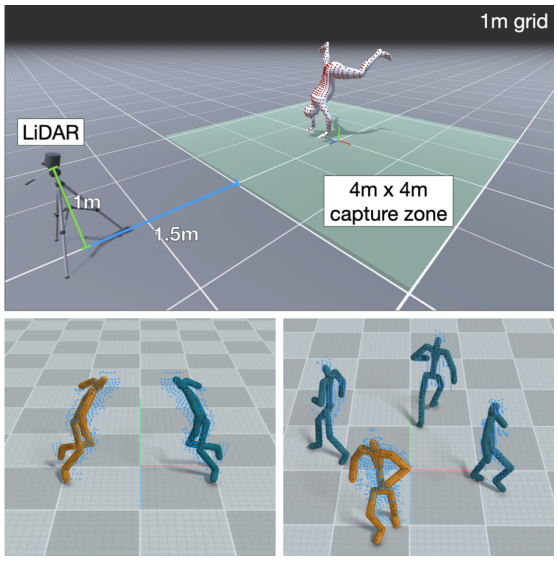

Fig. 4. Top: Snapshot of our LiDAR simulator. Red dots represent collision points between the simulated lasers and the body mesh animated with the augmented motion clips. Bottom: Augmentation results using mirroring and simulation for 90°, 180°, and 270° global rotations. The yellow character represents the original data, while the blue characters represent the augmented data.

We employ two augmentation strategies: mirroring and rotating, as described at the bottom of Figure 4. For mirroring, we double the number of original data by flipping the skeleton and point cloud data. Furthermore, we augment each motion clip by applying global rotations of 90°, 180°, and 270°. However, unlike mirroring, rotating the point cloud around the global origin poses a challenge as a fixed LiDAR would capture different sides of the subject for rotated motion clips. To address this issue, we use a point cloud simulator.

The simulator is implemented with the Unity3D engine and the virtual LiDAR follows the Hesai QT128 specifications [hes 2023]. To compute collision points with simulated lasers, we use the SMPL body mesh, with its shape parameters manually adjusted to match the subject's skeleton. The top image of Figure 4 shows a snapshot of the resulting simulated point clouds using our simulator. Following our LiDAR placement guidelines to cover a 4m x 4m x 2.5m capture volume, the virtual LiDAR is positioned 3.5 meters from the center of the capture zone (global origin), 1 meter above the ground, and angled 20 degrees downward. During simulation, motion clips run at 60 fps, and point cloud data are captured every 3 frames (20 fps).

4.2 Losses

The overall framework is trained by minimizing the reconstruction $\mathcal{L}^{rec}$, velocity loss $\mathcal{L}^{vel}$, and KL-divergence $\mathcal{L}^{kl}$ losses. The reconstruction loss comprises joint feature loss on both local and global coordinates and root feature loss. The velocity loss is the difference between consecutive features. The reconstruction and velocity loss are computed for frames $i+1$, $i+2$, and $i+3$. In addition, the KL-divergence loss regularizes the distribution of latent vector z to be near the prior distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$.

The total loss function is thus:

$$\begin{aligned}\mathcal{L}^{total} &= \mathcal{L}^{rec}|_{i+1}^{i+3} + w_{vel}\mathcal{L}^{vel}|_{i+1}^{i+3} + w_{kl}\mathcal{L}^{kl} \\ \mathcal{L}_i^{rec} &= \|\tilde{\mathbf{x}}_i - \mathbf{x}_i\|_1 + \|FK(\tilde{\mathbf{x}}_i) - FK(\mathbf{x}_i)\|_1 + \|\tilde{\mathbf{r}}_i - \mathbf{r}_i\|_1 \\ \mathcal{L}_i^{vel} &= \|V(\tilde{\mathbf{x}}_i) - V(\mathbf{x}_i)\|_1 + \|V(FK(\tilde{\mathbf{x}}_i)) - V(FK(\mathbf{x}_i))\|_1 \\ &\quad + \|V(\tilde{\mathbf{r}}_i) - V(\mathbf{r}_i)\|_1,\end{aligned} \tag{7}$$

where $V(\mathbf{x}) = \frac{\mathbf{x}^0 - \mathbf{x}^1}{h}$, $V(FK(\mathbf{x})) = \frac{FK(\mathbf{x}^0) - FK(\mathbf{x}^1)}{h}$, h is time step, and w_{kl}, w_{delta} are relative weights. FK represents forward kinematics.

4.3 Implementation details

The AdamW optimizer was used over 30 epochs with a learning rate of 10^{-4}. The loss weights w_{vel} and w_{kl} were both set to 1. In the embedding module, the ST-GCN and 1D convolution comprise one layer along with temporal pooling to extract joint features and the root feature. We split the input point cloud into 32 body-patch groups, which are input to Mini-PointNet, composed of one set abstraction layer. The upsampling motion generator G comprises 3 vision transformer layers with 384 channels and 8 heads. The motion distribution encoder E has the same architecture as the generator. The expanding module has an architecture symmetric to the embedding module. To prevent covariate shifts during autoregressive inference, we set the prediction length to 20 frames for training. Scheduled sampling was also utilized to make the model robust to its own prediction errors, enabling long-term generation. Training took around three days using two 24GB RTX4090 GPUs.

5 SKELETON CALIBRATION MODEL

Commercial motion capture systems measure bone lengths directly [xse 2011] or optimize from pre-programmed marker sets [opt 2009]. Prior data-driven methods continuously predict body shape and motion together from input sequences [Jiang et al. 2023; Ren et al. 2024]. In line with commercial setups, we devise a one-time skeleton calibration model to precede the capture session. The model predicts the user skeleton offsets from a single-frame point cloud, acquired while the user is in the A-pose.

5.1 Dataset Synthesis

To accommodate diverse body shapes among users, we generate a synthetic dataset comprising 50,000 pairs of an A-pose point cloud and initial joint offsets using our LiDAR simulator (Sec.4.1). Each data pair was generated through a randomized process to ensure the model's efficacy in real-world scenarios, despite being trained and validated solely on synthetic data. SMPL shape parameters of each subject are sampled in [-2, 2], covering 95% of the SMPL shape space. To enhance robustness against deviations in LiDAR placements in practice, we introduce random positional and rotational offsets to the virtual LiDAR, with maximums of 10cm and 5°, respectively. To further improve the model's robustness against variations in user A-poses, we apply random joint rotations: shoulder joints z-axis [65°, 45°], shoulder joints x-axis [-30°, 30°], hip joints y-axis [-30°, 20°], and hip joints z-axis [-20°, 4°]. In the final stage, the obtained SMPL joint offsets are retargeted to our skeleton hierarchy. Figure 5 presents example images of SMPL body mesh with random shape parameters

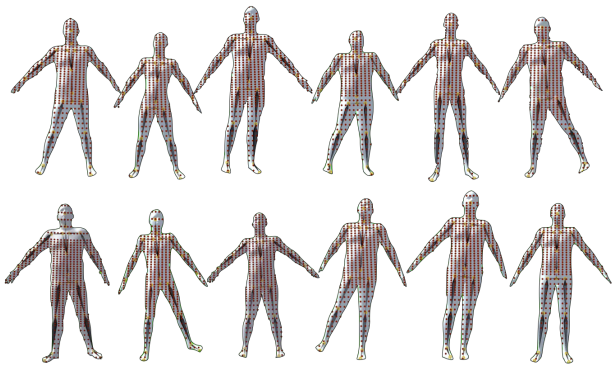

Fig. 5. Samples of random SMPL body meshes in A-poses with corresponding skeletons and simulated point clouds.

in random A poses, corresponding skeletons, and simulated point clouds. The dataset was split for training and validation in an 80:20 ratio, resulting in 40,000 and 10,000 pairs, respectively.

5.2 Calibration Model & Training

Our skeleton calibration model is a simple 6-layer multi-layer perceptron with bias disabled. It takes as input a flattened vector of 384 3D points sorted by their global height, $x \in \mathbb{R}^{384\times3}$. A preprocessing step, detailed in Sec. 3.5, is applied to sampled A-pose point clouds to ensure a fixed number of input points. Additionally, we apply noise ($X \sim \mathcal{N}(1\text{cm}, 1^2)$) to each position channel (x, y, z) of simulated points to enhance the robustness of our model in handling real-world noise induced by factors such as user outfits and hairstyles. Given this input, our model predicts skeleton offsets, including initial hip height and 20 joint offsets, represented as $y \in \mathbb{R}^{1+20\times3}$. Training is conducted for 220 epochs to minimize mean squared error, utilizing the Adam optimizer with a learning rate of 5.0×10^{-6}, and employing a batch size of 2048.

6 EVALUATION AND EXPERIMENTS

To assess the effectiveness of our framework, we conduct both quantitative and qualitative evaluations, comparing it with state-of-the-art (SOTA) methods in human motion tracking. Furthermore, we examine how the self-attention mechanism operates between body-patch point groups and joint features by constructing attention maps. Lastly, to showcase the real-time capability of ELMO, we present live-streaming scenarios with both single and multi-person setups and an interactive shooting game. For visual animation results, please refer to the supplementary video.

6.1 Datasets

We construct the ELMO dataset, a high-quality, synchronized single LiDAR-Optical Motion Capture-Video dataset featuring 20 subjects (12 males / 8 females, height_{cm} $h \sim \mathcal{N}(170.66,\ 7.90^2)$, $155 \leq h \leq 180$). Our objective is to capture a wide range of motions, styles, and body shapes. We utilize a 4×4 meter space, Hesai QT128 LiDAR, and an Optitrack system equipped with 23 cameras. The point cloud and mocap data were recorded at 20 and 60 fps, respectively. We split the 20 subjects into training and test sets with 17 and 3 subjects.

Table 1. Motion categories in ELMO dataset.

Range of Motion
Freely rotating an individual joint/joints in place & while moving
Static Movements
T-pose, A-pose, Idle, Look, Roll head
Elbows bent up & down, Stretch arms
Bow, Touch toes, Lean, Rotate arms
Hands on waist, Twist torso, Hula hoop
Lunge, Squat, Jumping Jack, Kick, Lift knee
Turn in place, Walk in place, Run in place
Sit on the floor
Locomotion
Normal walk, Walk with free upper-body motions
Normal Jog, Jog with free upper-body motions
Normal Run, Run with free upper-body motions
Normal Crouch, Crouch with free upper-body motions
Transitions with changing pace
Moving backward with changing pace
Jump (one-legged, both-legged, running, ...)

To capture point clouds from diverse distances and angles of a single LiDAR, we subdivided the capture zone into four separate subspaces and defined four viewing directions: forward, backward, left, and right (+z, -z, +x, -x). For each action category, participants executed the same action in 16 combinations of zone and viewing directions. To ensure our model accommodates diverse human poses, we initially captured a Range of Motion (ROM), including individual movements of each joint, along with their free combinations. Subsequently, we recorded in-place motions, followed by diverse locomotions involving different velocities, directions, and styles. Table 1 presents details on action labels comprising the ELMO dataset. The captured LiDAR point cloud and .bvh mocap files were precisely synchronized by our preprocessing pipeline.

We additionally tested our algorithm on the MOVIN dataset [Jang et al. 2023a] for quantitative evaluation. It consists of 10 subjects including 4 males and 6 females performing actions such as Walking, Jogging, Jumping, and Sitting on the floor. For the MOVIN dataset, we use 8 subjects for the training and 2 subjects for the test set.

For qualitative comparison, we also created a wild test dataset consisting of 3 subjects. To compare the performance of markerless motion capture without a suit, we built a synchronized single LiDAR-IMU-based Motion Capture-Video dataset. The IMU-based Motion Capture system used Xsens Awinda [awi 2011].

6.2 Quantitative Evaluation

We quantitatively compare our results with SOTA image-based pose tracking methods including VIBE [Kocabas et al. 2020], MotionBERT [Zhu et al. 2022], and NIKI [Li et al. 2023a]. Additionally, MOVIN [Jang et al. 2023a] serves as a primary comparison, utilizing the same LiDAR input as our framework. The quantitative evaluation assesses the methods using both MOVIN and our new ELMO dataset.

Quantitative measures are defined in terms of the mean joint (J) and pelvis (P) position/rotation/linear velocity/angular velocity errors (M*PE, M*RE, M*LVE, M*AVE). We measure joint errors for the fixed pelvis and separately measure pelvis errors, as image-based methods cannot explicitly track global trajectory. The accuracy of

Table 2. Quantitative comparison between baseline and ELMO models on the test splits of the ELMO and MOVIN datasets. "w/ dup" signifies output upsampled via duplication, while "w/ interp" denotes via interpolation. MOVIN† indicates the model retrained with the ELMO dataset. To ensure a fair comparison on the MOVIN dataset, results from ELMO (Ours) are downsampled to 20 fps.

	MJPE$_{cm}$	MJRE°	MJLVE	MJAVE
NIKI	14.30	18.04	1.41	2.35
MOVIN† w/ dup	7.03	11.87	1.32	1.65
MOVIN† w/ interp	7.05	11.87	1.08	1.45
ELMO (Ours)	**4.86**	**10.41**	**0.38**	**0.77**
	MPPE$_{cm}$	MPRE°	MPLVE	MPAVE
MOVIN† w/ dup	8.88	6.27	1.55	1.31
MOVIN† w/ interp	8.73	6.56	0.67	1.54
ELMO (Ours)	**4.08**	**5.08**	**0.20**	**0.38**

(a) ELMO dataset.

	MJPE$_{cm}$	MJRE°	MJLVE	MJAVE
VIBE	10.86	18.39	2.39	3.16
MotionBERT	10.62	18.05	1.75	2.24
NIKI	12.32	17.21	1.90	3.32
MOVIN	6.21	**10.12**	1.89	2.75
ELMO (Ours)	**4.77**	11.00	**0.92**	**1.53**
	MPPE$_{cm}$	MPRE°	MPLVE	MPAVE
MOVIN	4.42	11.64	2.46	4.94
ELMO (Ours)	**4.42**	**6.49**	**0.49**	**0.91**

(b) MOVIN dataset.

pelvis prediction significantly influences overall motion quality, as errors at the root joint propagate along the kinematic chain of the joint hierarchy. For comparison, we retarget the SMPL output of NIKI to our skeleton hierarchy [2].

SOTA comparison on ELMO dataset. The upper section of Table 2 compares our model's quantitative measures with baselines on the ELMO dataset. Given that MOVIN operates at 20 fps, we retrain it with a downsampled ELMO dataset, and its outputs are upsampled to 60 fps using duplication (w/ dup) and interpolation (w/ interp) for comparison.

For both joint and pelvis measures, our ELMO significantly outperformed the baselines. Compared to MOVIN upsampled with interpolation, the best among the baselines, the improvements are particularly notable in position measures, with a decrease of 2.19 cm in MJPE and 4.65 cm in MPPE. Additionally, our model showed performance increases ranging from a minimum of 47% (MJAVE) to a maximum of 75% (MPAVE), demonstrating its strength in capturing natural, non-linear pose transitions in both temporal and spatial spaces. This is further illustrated by the example outputs in panel (a) of Figure 12, where ELMO generates poses with greater accuracy (sitting, upper row) and faster reactions to input point clouds (running, bottom row).

SOTA comparison on the MOVIN dataset. Since the MOVIN dataset only provides 20 fps motion capture (mocap) data, we retrain ELMO on a synthetic 60 fps MOVIN dataset using interpolation, and the outputs are then downsampled to 20 fps for comparison.

The bottom part of Table 2 compares quantitative measures of our model with baselines on the MOVIN dataset. Overall, our ELMO outperformed the baselines in joint measures, except for MJRE. Despite a slight increase in MJRE by 0.88° compared to MOVIN, ELMO showed a notable improvement in MJPE by 1.44cm, indicating superior accuracy in joint positions. We observed that ELMO especially performed better during occlusions, as shown in the bottom row of the panel (b) of Figure 12, where the MOVIN output incorrectly lifted the opposite arm. Moreover, ELMO significantly outperformed MOVIN in MPRE, surpassing it by 5.15°; the upper row of Figure 12 shows an instance where MOVIN exhibited a global y-axis flip in the output.

Table 3. Quantitative measures of ablation models of future frame input and dataset augmentation.

	MJPE$_{cm}$	MJRE°	MJLVE	MJAVE
ELMO$_{20}$ w/ interp	5.05	11.45	0.54	0.91
ELMO baseline	6.33	12.48	0.48	0.84
+ 1 future frame	5.33	11.61	0.39	0.78
+ Augmentation (Ours)	**4.86**	**10.41**	**0.38**	**0.77**
	MPPE$_{cm}$	MPRE°	MPLVE	MPAVE
ELMO$_{20}$ w/ interp	4.10	5.13	0.19	0.82
ELMO baseline	4.97	6.39	0.28	0.45
+ 1 future frame	5.05	5.20	**0.19**	0.39
+ Augmentation (Ours)	**4.08**	**5.08**	0.20	**0.38**

(a) ELMO dataset.

	MJPE$_{cm}$	MJRE°	MJLVE	MJAVE
ELMO baseline	6.14	11.31	1.02	1.58
+ 1 future frame	5.39	11.65	0.98	1.57
+ Augmentation (Ours)	**4.77**	**11.00**	**0.92**	**1.53**
	MPPE$_{cm}$	MPRE°	MPLVE	MPAVE
ELMO baseline	5.03	8.66	0.59	1.01
+ 1 future frame	5.08	6.69	0.55	0.95
+ Augmentation (Ours)	**4.42**	**6.49**	**0.49**	**0.91**

(b) MOVIN dataset.

Ablation study. To assess the impact of our novel upsampling motion generator (Sec. 3.3) and data augmentation (Sec. 4.1), we conducted an ablation study in (a) ELMO and (b) MOVIN datasets. The baseline ELMO corresponds to the basic setup of our model , which predicts the poses of 3 future frames from the point cloud input of the current frame. In addition, we include a 20 fps output model without the upsampling scheme. For comparison within the ELMO dataset, the 20 fps outputs are upsampled using interpolation, denoted as ELMO$_{20}$ w/ interp.

Table 3 presents metrics from ablation models. Testing on the ELMO dataset revealed that incorporating a future frame input generally decreased errors, except for MPPE, which maintained comparable performance with a 0.08cm gap. This enhancement notably improved the model's accuracy in local body pose transitions compared to the baseline. Dataset augmentation yielded modest improvements in velocity errors but notably enhanced position and rotation errors for both body joints and pelvis. In the case of ELMO$_{20}$

[2]Retargeting is performed to enable comparison within the same skeleton topology. However, this process might introduce some unfairness.

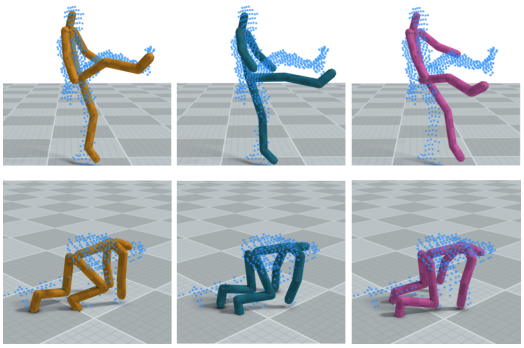

Fig. 6. Samples of offline outputs of ablation models. From left to right: ELMO with future frame input and data augmentation (Yellow), only with future frame input (Blue), and the baseline (Pink).

w/ interp., since the output poses are generated to match the point cloud input fps, the position and rotation errors are relatively low, but the overall velocity error is high.

Figure 6 displays examples from the ablation models. In the first row, adding a future frame enhanced leg pose accuracy during a kicking motion, and augmentation further improved performance, aligning the output with the point cloud input. In the bottom row, data augmentation increased robustness during occlusions, enhancing leg details while crawling.

Inference time comparison. For real-time applications, achieving a minimum of 60 fps is crucial. Table 4 outlines the elapsed times for methods on a Windows Laptop PC (i9-13900HX, RTX4080: TGP 175W). NIKI's inference time of 14 ms falls within the acceptable limit for 60 fps. However, the total time for bounding box detection, a prerequisite step before pose generation, is 454 ms, rendering it unsuitable for real-time applications. MOVIN demonstrates an inference time of 48 ms. When combined with the point cloud capture and processing time, the total time increases to 54 ms, barely meeting the 20 fps framerate. In contrast, although ELMO generates poses from the previous two frames to the current frame at a 60Hz rate, resulting in an input latency of 33ms, the inference time is only 5ms, bringing the total end-to-end elapsed time to just 44ms.

The reason our inference time is significantly faster compared to MOVIN lies in the overall structure of our model architecture. Specifically, in the point embedding section, MOVIN simply utilizes PointNet++, whereas our method employs body-patch grouping and Mini-PointNet to efficiently extract point features for each group.

Table 4. Total elapsed times for different frameworks.

Framework	Time_{ms}
NIKI inference	14
+ bounding box detection	454
MOVIN inference	48
+ point cloud capture & process	54
ELMO inference	5
+ latency	38
+ point cloud capture & process	44

Skeleton offset error. Table 5 displays the average bone length and direction errors of our skeleton calibration model (Sec. 5) evaluated on a test set comprising 7 subjects with heights of 155, 160, 168, 171, 177, and 179cm sampled from ELMO dataset. Our model achieves low errors, with an average joint length error of 1.52cm and direction error of 0.22°. Figure 7 and supplementary video demonstrate the calibration model's robust performance across users with diverse body shapes, outfits, and hairstyles. The model accurately predicts initial offsets, facilitating precise fitting of the scaled mesh to the captured point cloud.

Table 5. Average offset length and direction errors by body parts (number of joints in brackets).

	Hips (1)	Spine (4)	Arm(8)	Leg (8)	Total (21)
length error_{cm}	2.41	1.22	1.68	1.39	1.52
direction error°	–	0.11	0.34	0.17	0.22

6.3 Qualitative Evaluation

We qualitatively compare our real-time output of ELMO with Xsens Awinda [awi 2011], MOVIN, and NIKI [3] for the wild test dataset (Sec. 6.1). Please refer to the supplementary video for a visual comparison.

Comparison with Xsens and SOTA methods. Figure 13 presents the ground truth sequence and real-time output sequence of ELMO, Xsens Awinda, MOVIN, and NIKI. Overall, NIKI performs poorly across all scenarios, containing severe jitters and inaccurate poses. Xsens shows weakness in height change and global drifting, exemplified in jumping (4^{th} row) and rapid turning (6^{th} row). Moreover, the inherent limitation of acceleration-based tracking results in inaccurate overall global positions.

MOVIN exhibits significant jitter in the output poses and frequently struggles to match the input point cloud (1^{st}, 2^{nd}, and 5^{th} rows). As the method runs at 20 fps, the output poses often lose continuity, especially for relatively fast and continuous motions, including rotating the foot (5^{th} row) and high kick (1^{st} row). In addition, the method sometimes suffers a y-axis flip for global orientation as in ballet turn (3^{rd} row), making the character face the wrong direction. Furthermore, since the output is at 20 fps, severe frame drop can be observed when visualized in a 60 Hz environment.

In contrast, ELMO achieves natural and continuous 60 fps motions akin to Xsens while accurately capturing the global root trajectory. Furthermore, ELMO excels in capturing precise details such as body part contacts and producing plausible motions even in the presence of occlusion.

Skeleton Calibration. For the qualitative evaluation of the skeleton calibration model, we conducted a wild test with three subjects: a 157 cm female, a 171 cm male, and an 182 cm male. After acquiring a single-frame point cloud while each subject was in the A-pose at origin, our model predicted the user skeleton offsets. Figure 7 demonstrates that our model successfully predicted the skeleton offsets for each subject.

[3]The outputs from NIKI lack root transformation, so we copy the root transformation from the Xsens results to the NIKI output.

Fig. 7. Skeleton calibration results for wild input. From left to right: 171 cm male, 157 cm female, and 182 cm male.

Global drifting. Global drifting in motion capture leads to inaccuracies over time, significantly affecting the precision and reliability of motion data. In long sequences, these errors accumulate, causing substantial positional deviations. To compare the global drifting issues, we conducted an experiment where the subjects initiated dynamic motion from the origin, continued for 1-2 minutes, and then returned to the origin. The objective was to determine whether they precisely returned to the same position.

As depicted in Figure 8, our model and the MOVIN model demonstrate consistent positions at the starting and ending points when returning to the origin. In contrast, Xsens exhibits significant global drifting, confirming its final position is notably distant from the starting point.

6.4 Attention between joints and body-patch point groups.

In Fig. 9, (a) shows an input point cloud and the corresponding pose output for a specific time frame. The left image of (b) visualizes the center point (highlighted in color) of 32 body-patch point groups within the input point cloud, while the right image illustrates the attention map between the right arm joints and the body-patch

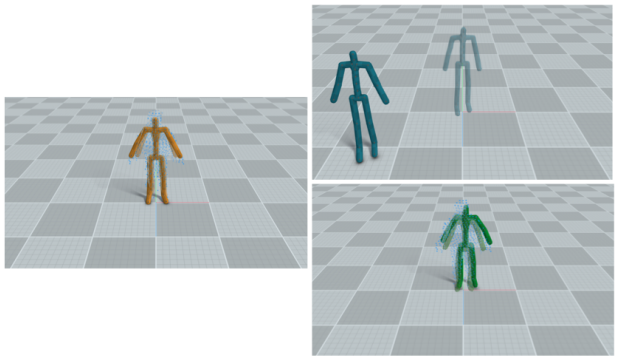

Fig. 8. Global Drifting results on ELMO (Yellow), Xsens (Blue), and MOVIN (Green). The initial character at the origin is depicted as transparent; upon returning to the origin, the character appears solid.

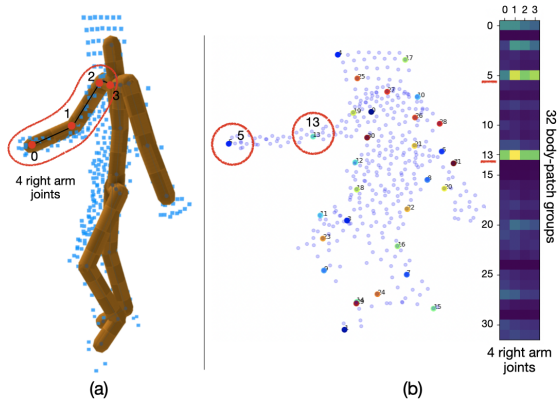

Fig. 9. (a) Input point cloud and output pose. (b) Attention map showing a higher correlation between right arm joints and groups indexed 5 and 13.

point groups. The x-axis of the attention map represents four joints in the right arm (grouped within the red border line in (a)), and the y-axis represents the 32 body-patch point groups.

Notably, the attention map shows high values on groups indexed 5 and 13, whose center points are located closer to the wrist and elbow joints. This suggests that our design choice of embedding the point cloud by body-patch, combined with the self-attention for Motion Tokens and Point Tokens, effectively captures the correlation between human joints and the body-patch groups from the input point cloud.

6.5 Real-time Motion Capture Applications

To highlight the practicality of ELMO, we seamlessly integrated it into real-time applications running at 60 fps using the Unity3D engine. For a comprehensive visual overview of the results, please refer to our supplementary video.

System setup. The process begins by positioning a single LiDAR sensor in front of the user. A laptop processes the incoming signals from the LiDAR sensor to generate motion data output. Next, the boundary or region of interest (ROI) is defined to specify where the motion capture will occur, ensuring focus on pertinent areas. The subject's skeleton offset is then registered using our calibration model. Finally, real-time motion capture is initiated through the ELMO framework, enabling seamless tracking of the user's movements. Figure 10 illustrates the overall system setup process.

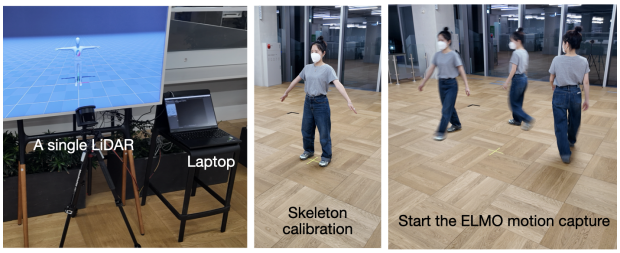

Fig. 10. System setup process for real-time motion capture.

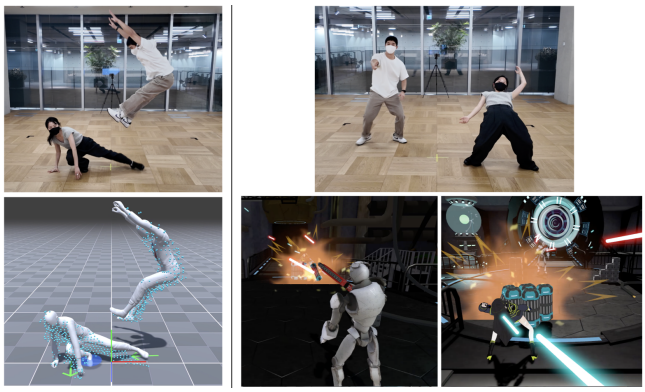

Fig. 11. Left: live streaming with two subjects engaged in martial arts. Right: Interactive shooting game.

Live streaming. We demonstrate our framework's ability to stream output motion in real-time for single-subject actions, object interactions, and two-subject interactions.

Figure 1 showcases ELMO's output poses across various actions, highlighting its versatility in capturing both common movements like walking, running, and jumping, as well as more intricate actions such as lying down, doing push-ups, and performing cartwheels. Our method accurately predicts foot contact from the output, facilitating the removal of foot-skating. Additionally, ELMO captures object interaction motions, such as sitting on a chair and placing a hand on a table. Notably, even with small occlusions, ELMO generates plausible motions, as demonstrated in the rightmost image featuring a narrow table. Moreover, ELMO's lightweight model ensures low latency when handling multiple subjects. The left image of Fig. 11 illustrates its proficiency in accommodating interactive dynamic actions like martial arts.

Interactive multi-player game. In the right image of Fig. 11, two subjects are depicted engaging in a shooting game, accompanied by in-game snapshots. Our method enables precise global tracking, facilitating immersive interactions within the virtual environment.

7 DISCUSSION ON LATENCY VS. ACCURACY

ELMO takes an in-betweening-based upsampling strategy by processing the point cloud input of frame $i+3$ and conducting a 3-frame upsampling for frames from $i+1$ to $i+3$. In contrast, the baseline ELMO (Sec. 6.2) predicts the poses of future frames $i+1$, $i+2$, and $i+3$ from the point cloud input of the current frame i, making it a prediction-based upsampling method.

While Baseline ELMO provides the benefit of zero-latency upsampling, it faces challenges when handling dynamic movements. Inaccuracies in predicting future frames can diminish accuracy, especially during acyclic rapid body motions. This can lead to accumulated errors and discontinuity in the output motions.

By utilizing the point cloud input of one future frame, ELMO introduces a latency of $33ms$ (equivalent to 2 frames at $60Hz$). However, employing in-betweening-based upsampling effectively mitigates the challenges faced by baseline ELMO and significantly improves accuracy, which is evident in our evaluation results in Section 6. Given the high quality of motion capture it provides, a latency of $50ms$ is considered acceptable in real-time scenarios. Hence, we have opted for the in-betweening-based upsampling approach.

8 LIMITATION AND FUTURE WORK

ELMO has several limitations that need to addressed in future studies. One challenge is self-occlusion, where the subject's body obstructs other body parts, particularly when standing sideways. This can result in ELMO generating plausible but imprecise poses that may not accurately align with the actual pose. Employing a multi-LiDAR system will be a promising solution by eliminating the occluded areas.

Additionally, when the user's action significantly deviates from the distribution of the training dataset, our motion prior generates only the closest plausible motion. As a result, it may not precisely track the exact pose while producing a convincing motion. Similarly, in challenging environments with degraded input (such as missing frames/points, sparse sampling, or self-occlusions), the model is expected to generate a plausible motion but may struggle to precisely track the actual pose.

To capture multi-person scenarios, we use a clustering algorithm to separate the input point clouds from each person and perform individual motion inference. Consequently, ELMO is restricted to non-overlapping dedicated zones for each subject, as the model fails when subjects' point clouds aggregate into a single cluster. We also handle occlusion from large objects like desks by placing the object in the desired position and filtering out its point cloud during background removal. This ensures that only the user's point cloud is captured. However, this approach cannot handle moving objects. A promising direction for future work is integrating point cloud segmentation with the motion capture framework to address these limitation.

For data augmentation, we does not reflect variations for hair and clothing. Since users in real-world scenarios may have diverse hairstyles and outfits, our current augmentation method may not be entirely appropriate. To overcome this limitation, Combining hair and clothing with the human model could yield higher quality synthetic data.

9 CONCLUSION

We introduced ELMO, an upsampling motion capture framework utilizing a single LiDAR sensor. ELMO achieves 60 fps motion capture from a 20 fps point cloud sequence with minimal latency ($< 44ms$) through its unique upsampling motion generator. Enhanced by a novel embedding module and attention coupling, ELMO delivers real-time capture performance comparable to off-the-shelf motion capture systems, eliminating the necessity for time-consuming calibration and wearable sensors. Additionally, our introduced LiDAR-Mocap data augmentation technique significantly enhances global tracking performance.

ACKNOWLEDGMENTS

We sincerely thank the MOVIN Inc. team for their insightful discussions and invaluable assistance in capturing and processing the

dataset. Sung-Hee Lee was partially supported by IITP, MSIT, Korea (2022-0-00566) and STEAM project, NRF, MIST (RS-2024-00454458).

REFERENCES

2009. *OptiTrack Motion Capture Systems.* https://www.optitrack.com/

2011. *Xsens Technologies B.V.* https://www.xsens.com/

2011. *Xsens Technologies B.V.* https://www.movella.com/products/motion-capture/xsens-mvn-awinda

2023. *Hesai QT128 Specification.* https://www.hesaitech.com/wp-content/uploads/QT128C2X_User_Manual_Q03-en-231210.pdf

Edilson Aguiar, Christian Theobalt, Carsten Stoll, Naveed Ahmed, Hans-Peter Seidel, and Sebastian Thrun. 2008. Performance Capture from Sparse Multi-View Video. *ACM Transactions on Graphics* (02 2008). https://doi.org/10.1145/1360612.1360697

Sadegh Aliakbarian, Pashmina Cameron, Federica Bogo, Andrew Fitzgibbon, and Thomas J Cashman. 2022. Flag: Flow-based 3d avatar generation from sparse observationszhang22motiondiffuse. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13253–13262.

S. Amin, Mykhaylo Andriluka, Marcus Rohrbach, and Bernt Schiele. 2013. Multi-view Pictorial Structures for 3D Human Pose Estimation. In *British Machine Vision Conference*.

Andreas Baak, Meinard Müller, Gaurav Bharaj, Hans-Peter Seidel, and Christian Theobalt. 2011. A data-driven approach for real-time full body pose reconstruction from a depth camera. In *2011 International Conference on Computer Vision*. 1092–1099. https://doi.org/10.1109/ICCV.2011.6126356

Valentin Bazarevsky, Ivan Grishchenko, Karthik Raveendran, Tyler Zhu, Fan Zhang, and Matthias Grundmann. 2020. Blazepose: On-device real-time body pose tracking. *arXiv preprint arXiv:2006.10204* (2020).

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. 2016. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. arXiv:1607.08128 [cs.CV]

C. Bregler and J. Malik. 1998. Tracking people with twists and exponential maps. In *Proceedings. 1998 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No.98CB36231)*. 8–15. https://doi.org/10.1109/CVPR.1998.698581

Magnus Burenius, Josephine Sullivan, and Stefan Carlsson. 2013. 3D Pictorial Structures for Multiple View Articulated Pose Estimation. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*. 3618–3625. https://doi.org/10.1109/CVPR.2013.464

Peishan Cong, Yiteng Xu, Yiming Ren, Juze Zhang, Lan Xu, Jingya Wang, Jingyi Yu, and Yuexin Ma. 2023. Weakly supervised 3d multi-person pose estimation for large-scale scenes based on monocular camera and single lidar. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 461–469.

Junting Dong, Qi Fang, Wen Jiang, Yurou Yang, Qixing Huang, Hujun Bao, and Xiaowei Zhou. 2022. Fast and Robust Multi-Person 3D Pose Estimation and Tracking From Multiple Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 10 (2022), 6981–6992. https://doi.org/10.1109/TPAMI.2021.3098052

Zhiyang Dou, Xuelin Chen, Qingnan Fan, Taku Komura, and Wenping Wang. 2023. C·ASE: Learning conditional adversarial skill embeddings for physics-based characters. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Yusen Feng, Xiyan Xu, and Libin Liu. 2023. MuscleVAE: Model-Based Controllers of Muscle-Actuated Characters. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Ylva Ferstl, Michael Neff, and Rachel McDonnell. 2019. Multi-Objective Adversarial Gesture Generation. In *Proceedings of the 12th ACM SIGGRAPH Conference on Motion, Interaction and Games* (Newcastle upon Tyne, United Kingdom) *(MIG '19)*. Association for Computing Machinery, New York, NY, USA, Article 3, 10 pages.

Shubham Goel, Georgios Pavlakos, Jathushan Rajasegaran, Angjoo Kanazawa, and Jitendra Malik. 2023. Humans in 4D: Reconstructing and Tracking Humans with Transformers. *arXiv preprint arXiv:2305.20091* (2023).

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems*, Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (Eds.), Vol. 27. Curran Associates, Inc.

Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. 2020. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–14.

Buzhen Huang, Liang Pan, Yuan Yang, Jingyi Ju, and Yangang Wang. 2022. Neural MoCon: Neural Motion Control for Physically Plausible Human Motion Capture. arXiv:2203.14065 [cs.CV]

Yinghao Huang, Manuel Kaufmann, Emre Aksan, Michael J Black, Otmar Hilliges, and Gerard Pons-Moll. 2018. Deep inertial poser: Learning to reconstruct human pose from sparse inertial measurements in real time. *ACM Transactions on Graphics (TOG)* 37, 6 (2018), 1–15.

Deok-Kyeong Jang, Dongseok Yang, Deok-Yun Jang, Byeoli Choi, Taeil Jin, and Sung-Hee Lee. 2023a. MOVIN: Real-time Motion Capture using a Single LiDAR. In *Computer Graphics Forum*. Wiley Online Library, e14961.

Deok-Kyeong Jang, Yuting Ye, Jungdam Won, and Sung-Hee Lee. 2023b. MOCHA: Real-Time Motion Characterization via Context Matching. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Jiaxi Jiang, Paul Streli, Manuel Meier, Andreas Fender, and Christian Holz. 2023. EgoPoser: Robust Real-Time Ego-Body Pose Estimation in Large Scenes. *arXiv preprint arXiv:2308.06493* (2023).

Yifeng Jiang, Yuting Ye, Deepak Gopinath, Jungdam Won, Alexander W Winkler, and C Karen Liu. 2022. Transformer Inertial Poser: Attention-based Real-time Human Motion Reconstruction from Sparse IMUs. *arXiv preprint arXiv:2203.15720* (2022).

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

Muhammed Kocabas, Nikos Athanasiou, and Michael J Black. 2020. Vibe: Video inference for human body pose and shape estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5253–5263.

Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. 2019. Convolutional Mesh Regression for Single-Image Human Shape Reconstruction. arXiv:1905.03244 [cs.CV]

Hsin-Ying Lee, Xiaodong Yang, Ming-Yu Liu, Ting-Chun Wang, Yu-Ding Lu, Ming-Hsuan Yang, and Jan Kautz. 2019. Dancing to Music. 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. https://proceedings.neurips.cc/paper/2019/file/7ca57a9f85a19a6e4b9a248c1daca185-Paper.pdf

Sunmin Lee, Taeho Kang, Jungnam Park, Jehee Lee, and Jungdam Won. 2023a. SAME: Skeleton-Agnostic Motion Embedding for Character Animation. In *SIGGRAPH Asia 2023 Conference Papers*. 1–11.

Sunmin Lee, Sebastian Starke, Yuting Ye, Jungdam Won, and Alexander Winkler. 2023b. QuestEnvSim: Environment-Aware Simulated Motion Tracking from Sparse Sensors. *arXiv preprint arXiv:2306.05666* (2023).

Jiefeng Li, Siyuan Bian, Qi Liu, Jiasheng Tang, Fan Wang, and Cewu Lu. 2023a. NIKI: Neural Inverse Kinematics with Invertible Neural Networks for 3D Human Pose and Shape Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12933–12942.

Jiaman Li, Yihang Yin, Hang Chu, Yi Zhou, Tingwu Wang, Sanja Fidler, and Hao Li. 2020. Learning to Generate Diverse Dance Motions with Transformer. (08 2020).

Jialian Li, Jingyi Zhang, Zhiyong Wang, Siqi Shen, Chenglu Wen, Yuexin Ma, Lan Xu, Jingyi Yu, and Cheng Wang. 2022b. Lidarcap: Long-range marker-less 3d human motion capture with lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 20502–20512.

Peizhuo Li, Kfir Aberman, Zihan Zhang, Rana Hanocka, and Olga Sorkine-Hornung. 2022a. GANimator: Neural Motion Synthesis from a Single Sequence. *ACM Trans. Graph.* 41, 4, Article 138 (jul 2022), 12 pages.

Tianyu Li, Jungdam Won, Alexander Clegg, Jeonghwan Kim, Akshara Rai, and Sehoon Ha. 2023b. ACE: Adversarial Correspondence Embedding for Cross Morphology Motion Retargeting from Human to Nonhuman Characters. *arXiv preprint arXiv:2305.14792* (2023).

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel Van De Panne. 2020. Character controllers using motion vaes. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 40–1.

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)* 34, 6 (2015), 1–16.

Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. 2017. VNect: Real-time 3D Human Pose Estimation with a Single RGB Camera. *ACM Transactions on Graphics* 36, 4, 14 pages. https://doi.org/10.1145/3072959.3073596

Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis. 2017. Coarse-to-Fine Volumetric Prediction for Single-Image 3D Human Pose. arXiv:1611.07828 [cs.CV]

Mathis Petrovich, Michael J Black, and Gül Varol. 2021. Action-conditioned 3d human motion synthesis with transformer vae. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10985–10995.

Jose Luis Ponton, Haoran Yun, Andreas Aristidou, Carlos Andujar, and Nuria Pelechano. 2023. SparsePoser: Real-time Full-body Motion Reconstruction from Sparse Data. *ACM Transactions on Graphics* 43, 1 (2023), 1–14.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. 2017. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. arXiv:1612.00593 [cs.CV]

Yiming Ren, Xiao Han, Chengfeng Zhao, Jingya Wang, Lan Xu, Jingyi Yu, and Yuexin Ma. 2024. LiveHPS: LiDAR-based Scene-level Human Pose and Shape Estimation in Free Environment. *arXiv preprint arXiv:2402.17171* (2024).

Yiming Ren, Chengfeng Zhao, Yannan He, Peishan Cong, Han Liang, Jingyi Yu, Lan Xu, and Yuexin Ma. 2023. Lidar-aid inertial poser: Large-scale human motion capture by sparse inertial and lidar sensors. *IEEE Transactions on Visualization and Computer Graphics* 29, 5 (2023), 2337–2347.

Karthik Shetty, Annette Birkhold, Srikrishna Jaganathan, Norbert Strobel, Markus Kowarschik, Andreas Maier, and Bernhard Egger. 2023. PLIKS: A Pseudo-Linear Inverse Kinematic Solver for 3D Human Body Estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 574–584.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. *Advances in neural information*

processing systems 28 (2015).

Ayça Takmaz, Jonas Schult, Irem Kaftan, Mertcan Akçay, Bastian Leibe, Robert Sumner, Francis Engelmann, and Siyu Tang. 2023. 3D Segmentation of Humans in Point Clouds with Synthetic Data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 1292–1304.

Xiangjun Tang, Linjun Wu, He Wang, Bo Hu, Xu Gong, Yuchen Liao, Songnan Li, Qilong Kou, and Xiaogang Jin. 2023. RSMT: Real-time Stylized Motion Transition for Characters. *arXiv preprint arXiv:2306.11970* (2023).

Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Amit H Bermano, and Daniel Cohen-Or. 2023. Human Motion Diffusion Model. In *ICLR*.

Jonathan Tseng, Rodrigo Castellon, and C Karen Liu. 2023. EDGE: Editable Dance Generation From Music. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Timo Von Marcard, Bodo Rosenhahn, Michael J Black, and Gerard Pons-Moll. 2017. Sparse inertial poser: Automatic 3d human pose estimation from sparse imus. In *Computer graphics forum*, Vol. 36. Wiley Online Library, 349–360.

Zhiyong Wang, Jinxiang Chai, and Shihong Xia. 2021. Combining Recurrent Neural Networks and Adversarial Training for Human Motion Synthesis and Control. *IEEE Transactions on Visualization and Computer Graphics* 27, 1 (jan 2021), 14–28.

Wen-Li Wei, Jen-Chun Lin, Tyng-Luh Liu, and Hong-Yuan Mark Liao. 2022. Capturing Humans in Motion: Temporal-Attentive 3D Human Pose and Shape Estimation from Monocular Video. arXiv:2203.08534 [cs.CV]

Xiaolin Wei and Jinxiang Chai. 2010. Videomocap: Modeling physically realistic human motion from monocular video sequences. In *ACM SIGGRAPH 2010 papers*. 1–10.

Alexander Winkler, Jungdam Won, and Yuting Ye. 2022. QuestSim: Human Motion Tracking from Sparse Sensors with Simulated Avatars. In *SIGGRAPH Asia 2022 Conference Papers*. 1–8.

Jungdam Won, Deepak Gopinath, and Jessica Hodgins. 2022. Physics-based character controllers using conditional vaes. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–12.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. 2018. Spatial Temporal Graph Convolutional Networks for Skeleton-Based Action Recognition. arXiv:1801.07455 [cs.CV]

Dongseok Yang, Jiho Kang, Lingni Ma, Joseph Greer, Yuting Ye, and Sung-Hee Lee. 2024. DivaTrack: Diverse Bodies and Motions from Acceleration-Enhanced Three-Point Trackers. In *Computer Graphics Forum*. Wiley Online Library, e15057.

Vickie Ye, Georgios Pavlakos, Jitendra Malik, and Angjoo Kanazawa. 2023. Decoupling human and camera motion from videos in the wild. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 21222–21232.

Jiaming Ying and Xu Zhao. 2021. Rgb-D Fusion For Point-Cloud-Based 3d Human Pose Estimation. In *2021 IEEE International Conference on Image Processing (ICIP)*. 3108–3112. https://doi.org/10.1109/ICIP42928.2021.9506588

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. 2022. Point-BERT: Pre-training 3D Point Cloud Transformers with Masked Point Modeling. arXiv:2111.14819 [cs.CV]

Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. 2022. MotionDiffuse: Text-Driven Human Motion Generation with Diffusion Model. *arXiv preprint arXiv:2208.15001* (2022).

Qingxu Zhu, He Zhang, Mengting Lan, and Lei Han. 2023. Neural Categorical Priors for Physics-Based Character Control. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–16.

Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. 2022. MotionBERT: Unified Pretraining for Human Motion Analysis. *arXiv preprint arXiv:2210.06551* (2022).

Michael Zollhöfer, Matthias Nießner, Shahram Izadi, Christoph Rehmann, Christopher Zach, Matthew Fisher, Chenglei Wu, Andrew Fitzgibbon, Charles Loop, Christian Theobalt, et al. 2014. Real-time non-rigid reconstruction using an RGB-D camera. *ACM Transactions on Graphics (ToG)* 33, 4 (2014), 1–12.

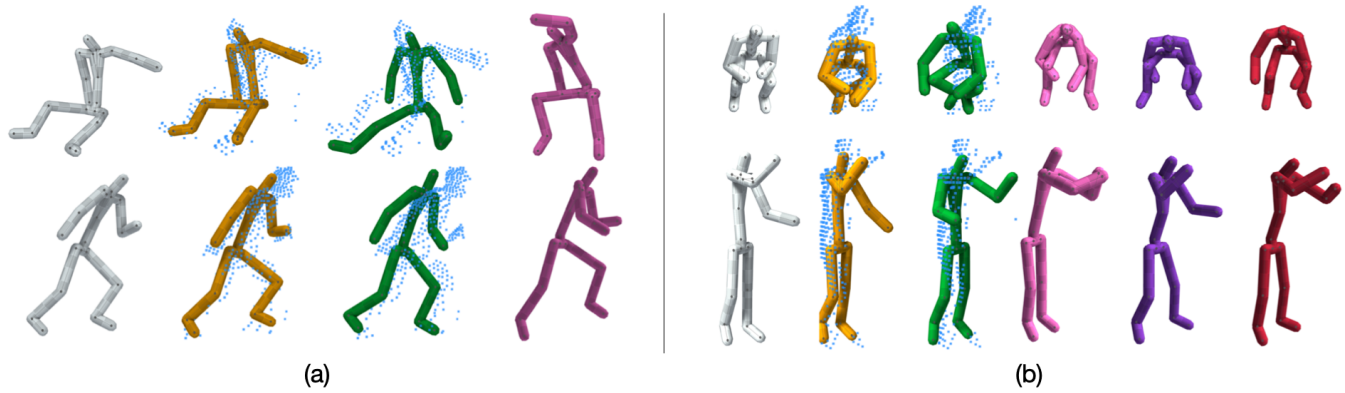

Fig. 12. **(a)** Samples of offline outputs on the ELMO dataset. From left to right: Ground Truth (Grey), ELMO (Yellow), MOVIN (Green), and NIKI (Pink). **(b)** Samples of offline outputs on the MOVIN dataset. From left to right: Ground Truth (Grey), ELMO (Yellow), MOVIN (Green), NIKI (Pink), MotionBert (Purple), and VIBE (Brown).

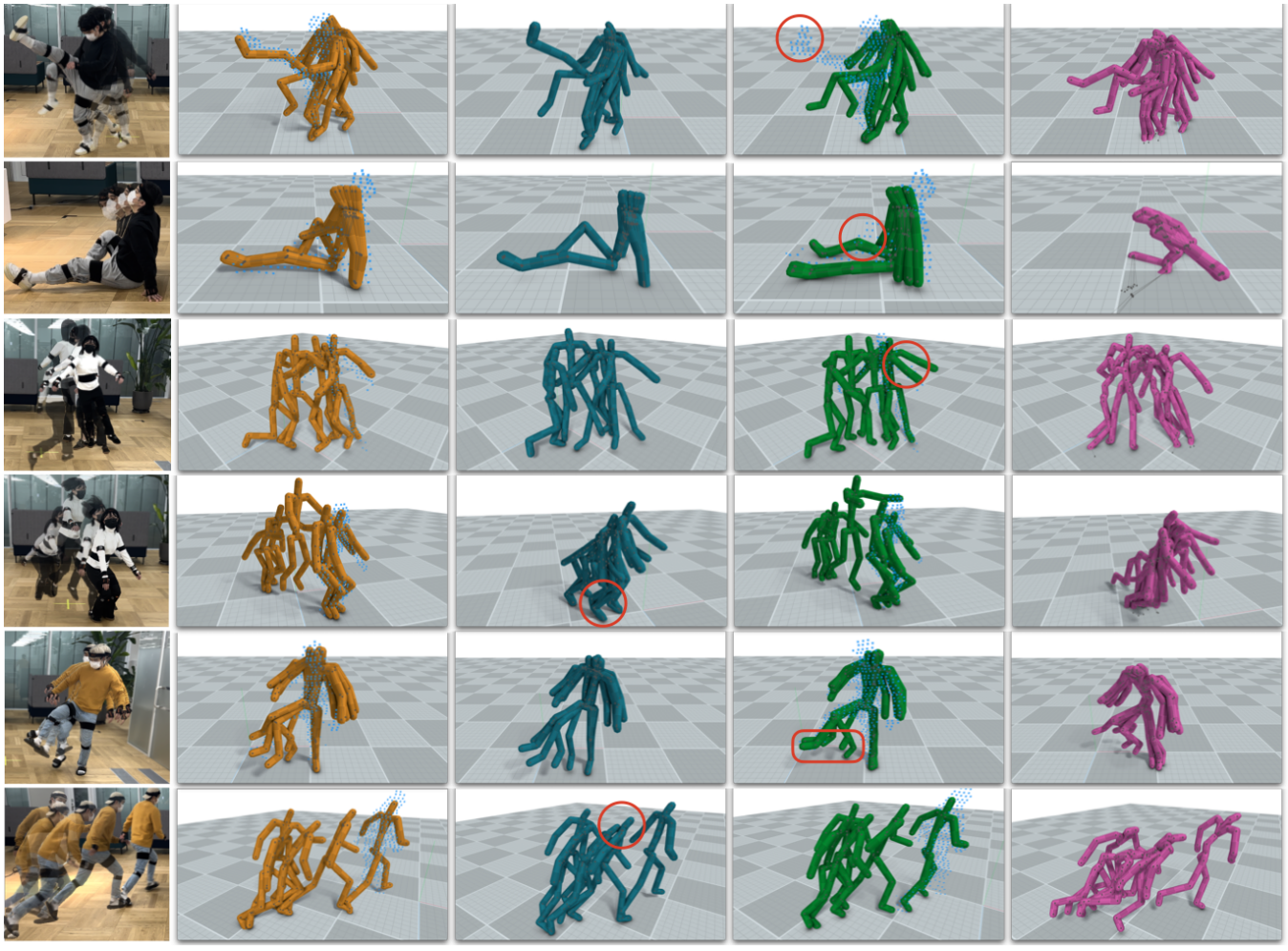

Fig. 13. Qualitative comparison of real-time motion capture performance, featuring ELMO and state-of-the-art methods. The sequence, from left to right, includes ground truth, ELMO (Yellow), Xsens Awinda (Blue), MOVIN (Green), and NIKI (Pink). Notable points are denoted by red circles.