Real-time Controllable Motion Transition for Characters

XIANGJUN TANG, State Key Lab of CAD&CG, Zhejiang University; ZJU-Tencent Game and Intelligent Graphics Innovation Technology Joint Lab, China
HE WANG, University of Leeds, United Kingdom
BO HU, Tencent Technology (Shenzhen) Co., Ltd., China
XU GONG, Tencent Technology (Shenzhen) Co., Ltd., China
RUIFAN YI, Tencent Technology (Shenzhen) Co., Ltd., China
QILONG KOU, Tencent Technology (Shenzhen) Co., Ltd., China
XIAOGANG JIN*, State Key Lab of CAD&CG, Zhejiang University; ZJU-Tencent Game and Intelligent Graphics Innovation Technology Joint Lab, China

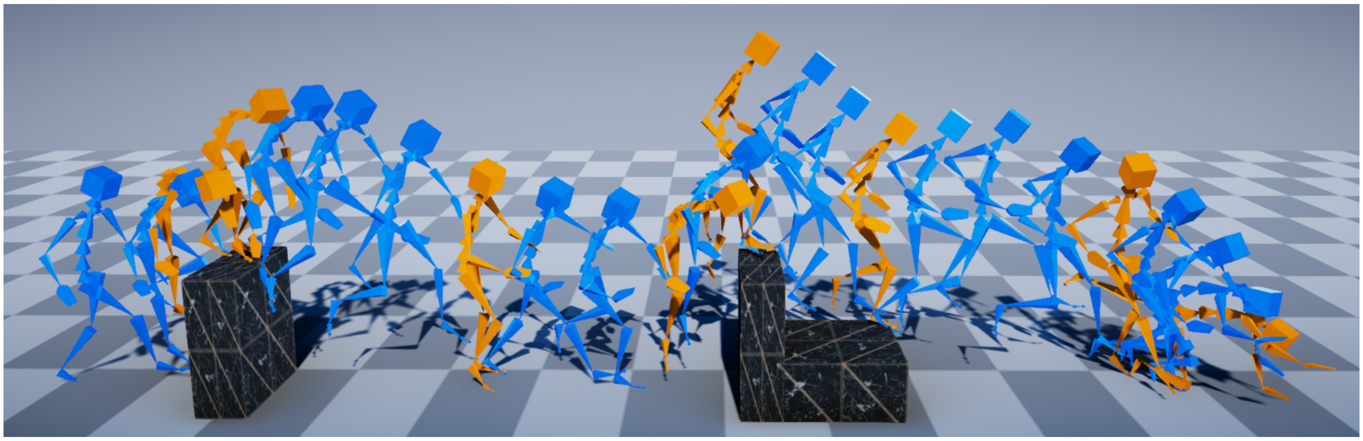

Fig. 1. In-between motion sequences (blue) between target frames (orange) generated by our method. Given a target frame and a desired transition duration, the controlled character can dynamically adjust strategies, e.g., different step sizes, velocities, or motion types, to reach the target without visual artifacts.

Real-time in-between motion generation is universally required in games and highly desirable in existing animation pipelines. Its core challenge lies in the need to satisfy three critical conditions simultaneously: *quality*, *controllability* and *speed*, which renders any methods that need offline computation (or post-processing) or cannot incorporate (often unpredictable) user control undesirable. To this end, we propose a new real-time transition method to address the aforementioned challenges. Our approach consists of two key components: motion manifold and conditional transitioning. The former learns the important low-level motion features and their dynamics; while the latter synthesizes transitions conditioned on a target frame and the desired transition duration. We first learn a motion manifold that explicitly models the intrinsic transition stochasticity in human motions via a multi-modal mapping mechanism. Then, during generation, we design a transition model which is essentially a sampling strategy to sample from the learned manifold, based on the target frame and the aimed transition duration. We validate our method on different datasets in tasks where no post-processing or offline computation is allowed. Through exhaustive evaluation and comparison, we show that our method is able to generate *high-quality* motions measured under multiple metrics. Our method is also *robust* under various target frames (with extreme cases).

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

Additional Key Words and Phrases: Animation, real-time, locomotion, motion manifold, conditional transitioning, in-betweening, deep learning

ACM Reference Format:
Xiangjun Tang, He Wang, Bo Hu, Xu Gong, Ruifan Yi, Qilong Kou, and Xiaogang Jin*. 2022. Real-time Controllable Motion Transition for Characters. *ACM Trans. Graph.* 41, 4, Article 137 (July 2022), 10 pages. https://doi.org/10.1145/3528223.3530090

*Corresponding author.

Authors' addresses: Xiangjun Tang, State Key Lab of CAD&CG, Zhejiang University; ZJU-Tencent Game and Intelligent Graphics Innovation Technology Joint Lab, China, fcsx1tf@163.com; He Wang, University of Leeds, United Kingdom, H.E.Wang@leeds.ac.uk; Bo Hu, Tencent Technology (Shenzhen) Co., Ltd., China, corehu@tencent.com; Xu Gong, Tencent Technology (Shenzhen) Co., Ltd., China, xugong@tencent.com; Ruifan Yi, Tencent Technology (Shenzhen) Co., Ltd., China, ryanfyi@tencent.com; Qilong Kou, Tencent Technology (Shenzhen) Co., Ltd., China, rambokou@tencent.com; Xiaogang Jin*, State Key Lab of CAD&CG, Zhejiang University; ZJU-Tencent Game and Intelligent Graphics Innovation Technology Joint Lab, China, jin@cad.zju.edu.cn.

0730-0301/2022/7-ART137 $15.00
https://doi.org/10.1145/3528223.3530090

1 INTRODUCTION

In-between motion generation has been a long-standing problem in computer graphics/animation [Witkin and Kass 1988], and recently revived [Harvey and Pal 2018; Kaufmann et al. 2020; Zhang and van de Panne 2018] under the context of deep learning. It has been heavily relied upon in both offline animation pipelines and online motion synthesis in games. Speedy generation of high-quality motions without post-processing or offline computation is highly desirable in the former, and is often a must in the latter.

Early methods formulate in-between motions as motion planning problem [Wang et al. 2015, 2013; Ye and Liu 2010], which requires solving complex optimizations and are prohibitively slow for real-time applications. Data-driven methods have also been developed [Kovar et al. 2008; Min and Chai 2012; Shen et al. 2017]. However, to handle arbitrary in-between motions and target frames, the size of needed data in memory grows exponentially [Harvey et al. 2020]. In the era of deep learning, in-between motions can be interpreted as a motion manifold learning problem [Chen et al. 2020; Holden et al. 2016; Wang et al. 2021], or a control problem [Ling et al. 2020] if dense temporal control signs are available. Compared with previous data-driven methods, deep neural networks can leverage compressed data representations, but cannot be easily converted into in-between motion generators [Harvey et al. 2020]. Very recently, this classic problem has been revived [Duan et al. 2021; Kaufmann et al. 2020], but there is still a lack of model generality when facing arbitrary target frames, which is often the case especially in real-time games where the user input is unpredictable.

There are two major challenges in real-time in-between motion generation. The foremost is the *motion quality*. Since motions need to be generated fast, post-processing is highly undesirable. Also, offline computation and any human intervention are strictly ruled out. One possible solution is a motion model which can capture the fine-grained dynamics of diverse actions and act as a source of motion generation. Designing such a model needs to consider the intrinsic transition ambiguity of human motions, i.e. multiple frames or actions could follow a given one. This leads to the second challenge: *controllability*. While capturing and disambiguating the transitions can achieved by relying on continuous control signals [Holden et al. 2017], our problem setting only involves sparse target frames. The control sparcity differentiates our problem from those with similar key frames or dense control signals. In addition, the generated motion needs to satisfy the target frame and the aimed transition duration simultaneously. Failing in transition disambiguation will lead to 'averaged' motions [Fragkiadaki et al. 2015], while failing in controllability will break the constraints imposed by the user.

In this paper, we propose a novel method which can generate high-quality in-between motions in real-time, given the starting and end frame with the desired period of transition. Our method consists of two components designed to address the aforementioned challenges. We start by representing the natural *motion manifold* and focus on modeling the multi-modality of motion transitions under a Markov assumption. To incorporate the target frame and the desired transition period, we further propose a new *sampler* which samples from the learned motion manifold, under the constraints imposed by the initial, target frame and the desired transition period.

The motion manifold employs a Conditional Variational Autoencoder (CVAE) architecture. Instead of learning a conditioned latent distribution of original data as traditional CVAEs, our CVAE learns a conditional distribution of transitions between frames [Ling et al. 2020]. Further, we explicitly model the transition ambiguity as a multi-modal mapping between frames, by utilizing a Conditional Mixture of Experts (CMoEs) in the latent space and the decoding phase. As a result, our motion manifold can act as a high-quality representation which provides a reliable source for online motion synthesis. The other key component is a transition sampler which samples one frame at a time. The sampler is realized as a deep neural network which models the dynamics of the generated motion by a Recurrent Neural Network (RNN). It conditions the next frame on the current frame, the target frame, the desired transition period and the remaining motion, through a multi-step residual network architecture.

We test our method on two popular datasets, under a variety of conditions, e.g. action types, generation lengths, the spatio-temporal aspects of the target frame and transition period. We employ both qualitative and quantitative evaluation, with multiple metrics including reconstruction errors, foot skating, and Normalized Power Spectrum Similarity (NPSS). After exhaustive ablation studies among different alternative architectures and representations, and comparisons with the state-of-the-art methods, we show that our method can generate high-quality motions in real-time, is robust across action types and dynamics, generalizes well to extreme user inputs, and outperforms existing methods under multiple criteria.

Our main contributions can be summarized as follow:

- We present a novel online framework for high-quality real-time in-between motion generation without post-processing.
- We propose a natural motion manifold model which is able to condition motion transitions on control variables for transition disambiguation, simultaneously providing controllability and ensuring motion quality.

2 RELATED WORK

Existing methods formulate in-between motions in various fashions. In the early days, in-between motions were often formulated as a motion planning problem [Arikan and Forsyth 2002; Beaudoin et al. 2008; Levine et al. 2012; Safonova and Hodgins 2007; Wang and Komura 2011], where fairly sophisticated motions can be synthesized. Complex optimization problems [Chai and Hodgins 2007] are formed with respect to various constraints such as contact and control input, leading to slow computation which is impractical for the animators and impossible for real-time applications. Alternatively, data-driven methods can avoid slow optimizations by searching in structured data, e.g. motion graphs [Kovar et al. 2008; Min and Chai 2012; Shen et al. 2017]. However, since the control or constraints can be diverse, the size of needed data in memory to cover all situations grows exponentially [Harvey et al. 2020], leading to unaffordable space complexity. Recently, in-between motions have been interpreted as a motion manifold learning problem [Chen et al. 2020; Holden et al. 2016; Li et al. 2021; Petrovich et al. 2021; Rempe et al. 2021; Wang et al. 2021], or a control problem [Ling et al. 2020] in deep learning. Compared with previous data-driven methods, deep neural networks can leverage compressed data representation [Holden et al. 2020]. However, they cannot be easily converted into an in-between motion generator [Harvey et al. 2020].

One attempt to convert the deep neural network to an in-between motion generator is to add constraints as regularization in the loss function. Examples include RNN-based models [Chiu et al. 2019; Martinez et al. 2017] which can generate motions with constraints to reduce motion ambiguity [Harvey and Pal 2018]. However, simply adding constraints cannot achieve high-quality results when different transition duration is needed. Subsequently, a time-to-arrival condition is proposed [Harvey et al. 2020] and a generative adversarial network (GAN) is employed to safe-guard the quality of the generated motion. However, without explicitly extracting different hierarchies in human dynamics [Chiu et al. 2019] or utilizing the relation of joints [Jain et al. 2016], the diversity of generated sequence is limited, leading to poor generalizability to unseen constraints such as extreme user control.

If real-time performance is not a requirement, offline methods can be employed in motion completion such as in-between motion generation or joints filling. Motion completion can be solved by optimizing the sampling of the motion manifold [Li et al. 2021], or considered as an analogy to the image infilling problem [Hernandez et al. 2019; Kaufmann et al. 2020]. A convolutional network can be employed to infill the missing parts of the sequence. The missing parts do not have to be whole frames. They could be just the position or orientation of a single joint. Besides, separating local motions and global trajectories enables convolutional networks to focus on generating realistic local poses [Zhou et al. 2020]. The infilling problem can also be solved by Transformers [Duan et al. 2021]. Instead of padding the missing frames, Transformer-based motion infiller [Yuan et al. 2021] can restrict its attention to visible frames to achieve effective temporal modeling. Time efficiency is normally not the primary goal of offline methods, so it is acceptable to utilize post-processing to improve the motion quality. Nevertheless, we aim for real-time generation.

3 METHODOLOGY

Our method consists of two main components: a natural motion manifold model and a sampler for motion generation. We first introduce the natural motion manifold that learns the low-level short-horizon motion dynamics. We then introduce a sampling strategy to generate motions from the learned manifold satisfying the target frame and the aimed transition duration.

3.1 The Motion Manifold

To generate a motion $M = \{S^1, \ldots, S^{n-1}\}$ with $n-1$ frames, each frame is denoted by $S^i = \{\mathbf{p}_L^i, \mathbf{p}_h^i, \mathbf{p}_U^i, \mathbf{r}_L^i, \mathbf{r}_h^i, \mathbf{r}_U^i, \mathbf{v}_L^i, \mathbf{v}_h^i, \mathbf{v}_U^i\}$ where $\mathbf{p}$, $\mathbf{r}$ and $\mathbf{v}$ are the joint position, rotation and velocity, and the subscript L, h and U indicate the lower body, the hip and the upper body joints respectively. Given a starting frame S^0, a target frame S^t, and the aimed transition duration z_{dt}, the joint probability of M can be represented as:

$$P(M) = \int\int\int P(M|S^0, S^t, z_{dt}) P(S^0, S^t, z_{dt}) dS^0 dS^t dz_{dt}, \quad (1)$$

where we assume the independence among S^0, S^t, and z_{dt} and omit them for later. Under a Markov assumption, $P(M)$ can be decomposed into:

$$P(M) = \prod_{i=1}^{n-2} P(S^{i+1}|S^i), \quad (2)$$

where $P(S^{i+1}|S^i)$ can be learned in many ways, e.g. through recurrent models [Wang et al. 2021] or co-embedding of consecutive frames [Ling et al. 2020]. Here, we choose to employ the co-embedding strategy as it easily allows conditional variables to be introduced. We introduce a latent variable z to encode the co-embedding of two consecutive frames and also use the next frame hip velocity v_h^{i+1} as a conditional variable. While z encodes the transition probability of two consecutive frames (a.k.a the dynamics), v_h^{i+1} can help disambiguate the next frame, for which we will give details when introducing the transition sampling. Introducing z and v_h^{i+1} into $P(S^{i+1}|S^i)$ gives:

$$P(S^{i+1}|S^i) = \int P(S^{i+1}|S^t, z, v_h^{i+1}) P(z) P(v_h^{i+1}), \quad (3)$$

Note that we divide all joints into three groups: upper-body, hip, and lower-body. This is because we empirically find they have different importance in the generation. The hip velocity gives a strong indication of the next frame (e.g. distinguishing between motions with high and low velocities). The lower-body joints significantly influence the visual quality due to potential foot sliding. The upper-body joints are less constrained comparatively. Therefore, we focus on learning the lower-body and the hip in $P(S^{i+1}|S^i)$:

$$P(S^{i+1}|S^i) = \int P(v_L^{i+1}, \dot{r}_L^{i+1}|c_{h,L}, v_h^{i+1}, z) P(z), \quad (4)$$

where $c_{h,L}^i = \{v_h^i, v_L^i, r_h^i, r_L^i\}$ consists of the lower-body and the hip joints of the current frame. We assume that v_h^{i+1} is given during prediction and hence its prior distribution can be removed. $\dot{r}$ is the angular velocity. Since z should encode two consecutive frames, it can be independently learned via $P(z) = P(z|c_{h,L}^i, c_{h,L}^{i+1})$ or expanding it to $P(z) = P(z|c_{h,L}^i, v_h^{i+1}, v_L^{i+1}, r_h^{i+1}, r_L^{i+1})$. If we assume $z \sim N(0, \mathbf{I})$, where N is a normal distribution, $P(z)$ can then be considered as the encoder of a Conditional VAE or CVAE where the condition is $\{c_{h,L}^i, v_h^{i+1}\}$ and the latent space distribution is constrained to be a Normal, as shown in Figure 2.

Next, $P(v_L^{i+1}, \dot{r}_L^{i+1}|c_{h,L}^i, v_h^{i+1}, z)$ can be considered as the decoder of the CVAE. Instead of reconstructing r_L^{i+1} directly, we compute it by $r_L^{i+1} = \dot{r}_L^{i+1} + r_L^i$. The conditional variables of the decoder are specifically designed to capture the ambiguous transitions that are intrinsic to human motions. It aims to learn discriminative transitions via a multi-modal mapping. Specifically, given a non-discriminative embedding z, the decoding is conditioned on the current frame $(v_h^i, v_L^i, r_h^i, r_L^i)$ and the future hip velocity (v_h^{i+1}). Such a decoding process requires the decoder to learn a multi-modal mapping that is similar to incorporating different dense control signals [Zhang et al. 2018] [Holden et al. 2017]. Therefore, we employ a Conditioned Mixture of Experts (CMoEs) model in the decoder, as shown in Figure 2. During learning, the CMoEs can learn discriminative mappings where each expert network tends to focus on learning

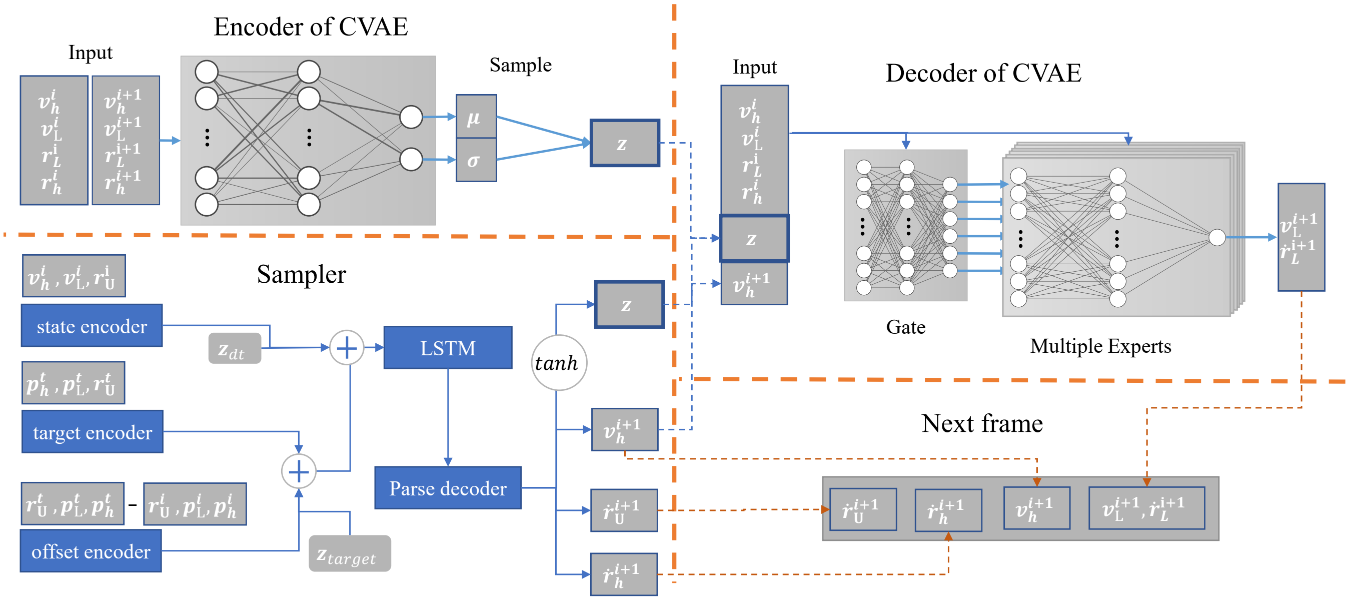

Fig. 2. When training the CVAE, the encoder takes frames i and $i+1$ as input and generates the mean value μ and log variance σ of the normal distribution. The motion manifold model (the decoder of CVAE) takes the current frame $(v_h^i, \mathbf{v}_L^i, \mathbf{r}_L^i, r_h^i)$, the character movement v_h^{i+1} at the next frame, and a latent vector z to generate the rotation and velocity of the lower-body joints. When training the transition sampler, we first remove the encoder of the CVAE, then fix the decoder and connect the transition sampler to the fixed decoder to train the sampler. The target frame, current frame, and offset are encoded by the target, state, and offset encoders, respectively. Subsequently, the embeddings z_{dt} and z_{target} are added to the encoded vectors, and an LSTM network takes the encoded vectors to produce the next state. Finally, the parse decoder takes the state and outputs the upper joints $\dot{r}_U^{i+1}$ and the sample (z, v_h^{i+1}).

one phase of motions. We also add a gating network which learns a weighting scheme for experts given a specific input. The final output is a weighted sum of all expert outputs.

3.1.1 Losses. To train the CVAE, we minimize a loss function:

$$L = L_{foot} + L_{bone} + L_{rec} + L_{kl}, \quad (5)$$

where several loss terms are proposed. L_{foot} is a foot skating loss in the joint position space:

$$L_{foot} = \hat{v}_{end} + v_h, \quad (6)$$

where $\hat{v}_{end}$ is the predicted relative velocity of the contacting foot with respect to the ground. When the velocity is less than $0.2cm/s$, we assume a foot contact with the ground. Although a joint angle representation is also theoretically possible with forward kinematics, the relation to be learned would become unnecessarily non-linear.

Besides, we add a bone length loss. For each joint j and its neighbor joints in $n(j)$, the loss is:

$$L_{bone} = ||\hat{p}_j - \hat{p}_k||_2 - ||p_j - p_k||_2, \forall k = n(j), j \in \mathbf{L}, \quad (7)$$

where $\hat{p}_j$ is the predicted position of joint j.

The reconstruction loss is defined as the mean squared error (MSE) between the predicted pose and the ground-truth:

$$L_{rec} = ||\hat{\mathbf{p}}_L - \mathbf{p}_L||_2^2 + ||\hat{\mathbf{r}}_L - \mathbf{r}_L||_2^2. \quad (8)$$

Finally, a KL-divergence loss is employed to constrain the distribution of the latent vector to be a standard Gaussian distribution:

$$L_{kl} = -0.5 \cdot (1 + \sigma - \mu^2 - e^{\sigma}), \quad (9)$$

where μ and σ are the mean and log variances.

3.2 Transition Sampling

Although the CVAE can learn a natural manifold, it can only perform uncontrolled generation. This is because S^0 can be easily used to start the generation in Equation 4, but the distributions are not conditioned on S^t and z_{dt}. Explicitly learning $P(M)$ conditioned on S^t and z_{dt} requires learning the reverse Markov chain across all possible duration, which is not trivial. Therefore, we use a neural network to learn them implicitly. To be able to generate motions continuously, we need to sample z and v_h^{i+1} to generate S^{i+1} given S^i, under the constraints of S^t and z_{dt}. So the network is essentially a sampler for sampling frames from the learned manifold.

The architecture of the network is shown in Figure 2. The sampler considers the constraints by the target encoder and the offset encoder, which encode the target frame and the offset between the current and the target frame, respectively. The key output is the next-frame condition z and v_h^{i+1}. In addition, when used for decoding the next frame, z and v_h^{i+1} will be pulled through the decoder of our CVAE, where essentially a manifold projection is conducted to refine the pose. We also add a time-varying noise z_{target} to the encoded vector, sampled from a zero-centered Gaussian distribution with variance equal to 0.5. Its amplitude λ decreases as it approaches the target frame so that the sampler's attention only focuses on the target when close to it. It also helps to improve the robustness to new conditioning information [Harvey et al. 2020]. The amplitude of the noise decreases by the function:

$$\lambda = clamp\left(\frac{dt - t_{zero}}{t_{period} - t_{zero}}, 0, 1\right), \tag{10}$$

where dt is the frame difference between the current time and the target, t_{zero} is the frame duration without noise, and t_{period} is the period of linear decrease of the noise. We empirically set $t_{zero} = 5$ and $t_{period} = 30$ in our experiments.

The sampler takes the current state of the pose via the state encoder. The constraint of z_{dt} is represented by the time embedding $\mathbf{z}_{dt}$ [Harvey et al. 2020], added to the latent vector of all encoders. The time embedding vector is similar to the positional encoding in [Vaswani et al. 2017]:

$$\mathbf{z}_{dt,2i} = sin\left(\frac{dt}{10000^{2i/d}}\right), \quad \mathbf{z}_{dt,2i+1} = cos\left(\frac{dt}{10000^{2i/d}}\right), \tag{11}$$

where d represents the dimension of $\mathbf{z}_{dt}$ and $i \in [0, ..., d/2]$ represents the dimension index.

Next, the recurrent neural network takes all latent vectors to predict the next state. A decoder parses the state to generate the sample (z, v_h^{i+1}) and the upper joints $\dot{r}_U^{i+1}$.

When passing the frame into encoders, we represent S by the hip velocity, the lower joints' velocity, and the upper joints' rotation to reduce dimensionality (v_h, v_L, r_U) compared to the full state, and calculate the offset using the lower joints' position p_L. To balance the attention on the lower joints and upper joints, we apply z-score normalization on p_L before passing it into the offset encoder.

All three encoders are two-layer feed-forward networks with 512 units in the first hidden layer and 256 units in the second layer. Each layer is followed by PLU activation. The parse decoder has three layers with 512 units in the first hidden layer and 256 units in the second layer, followed by ELU activation. To compute z, which is a key input to our CVAE to sample the next frame, we apply a tanh function and scale the output by 4.5 to ensure a good coverage of the normal distribution.

3.2.1 Losses. To train the sampler, we propose the following loss function:

$$L = L_{rot} + L_{leg} + L_{pos,rot}, \tag{12}$$

where L_{rot} is a L1 norm rotation loss for all joints and L_{leg} is a position loss for lower-body joints:

$$\begin{aligned} L_{rot} &= ||\hat{\mathbf{r}}_L - \mathbf{r}_L||_1 + ||\hat{\mathbf{r}}_U - \mathbf{r}_U||_1, \\ L_{leg} &= ||\hat{\mathbf{p}}_L - \mathbf{p}_L||_1. \end{aligned} \tag{13}$$

Besides, similar to Harvey *et al.* [2020], we employ Forward Kinematics (FK) to obtain the position $\hat{\mathbf{p}}_{rot}$ of all joints from their predicted rotation. The loss between $\hat{\mathbf{p}}_{rot}$ and the ground truth position $\mathbf{p}$ helps to implicitly weigh the rotation of the bone's hierarchy for better results [2020]:

$$L_{pos,rot} = ||\hat{\mathbf{p}}_{rot} - \mathbf{p}||_1. \tag{14}$$

In addition, the foot skating loss (see Eq. 6) and the bone length loss (see Eq. 7) are also used for training the sampler.

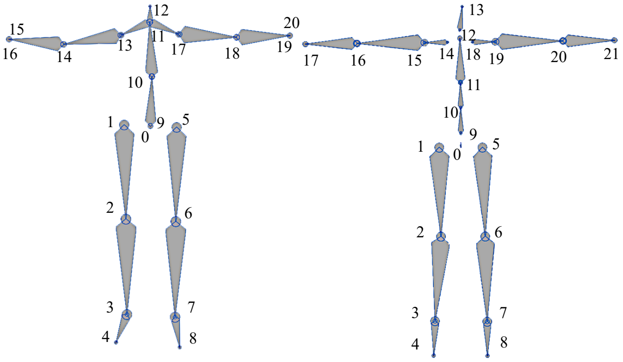

Fig. 3. The left character has 21 joints from the Human3.6M. The right character from the Lafan1 dataset has 22 joints. In our setup, the lower joints set is $L = \{1, 2, 3, 4, 5, 6, 7, 8\}$, the joint 0 is hip joint, and the other joints are the upper joints.

4 IMPLEMENTATION

4.1 Data formatting

We use the LaFAN1 dataset [Harvey et al. 2020] and the Human3.6M dataset [Ionescu et al. 2011] [Ionescu et al. 2014]. We remove the wrist and thumb joints from the Human3.6M dataset, which leaves us with 21 joints. The character from the Lafan1 dataset has 22 joints. We employ different representations for different joints. As shown in Fig. 3, we use the position-based representation for 8 lower joints and the rotation-based representation for upper joints. All lower joints connect less than two other joints to determine their orientation.

Zhang et al. [2018] proposed to represent joint rotation by a 2-axis rotation matrix $r \in \mathbf{R}^6$, containing a 3D vector for the up direction and a 3D vector for the forward direction. The joint position j contains a 3D vector $v_j \in \mathbf{R}^3$ to represent the velocity of the joint and a 3D vector to represent the up direction. We replace the 3D up direction with the 2-axis rotation matrix for uniformity. So the lower joints X_L^i of frame i can be represented by eight lower joints' rotation $\mathbf{r}_L^i \in \mathbf{R}^{8\times6}$ and six lower joints' position $\mathbf{p}_L^i \in \mathbf{R}^{6\times3}$ with the velocity $\mathbf{v}_L^i \in \mathbf{R}^{6\times3}$. Notice that we discard the velocity of joints 1 and 5 because they are determined by joint 0's rotation.

Both datasets contain multiple subjects. The subjects in the test set are different from the training set in our experiments, which ensures that our motion model can generalize to different subjects after training. The Lafan1 dataset contains 496,672 motion frames performed by 5 motion subjects. Similar to [Duan et al. 2021; Harvey et al. 2020], we use subject 5 as the test set. For the Human3.6M dataset, we choose to use a subset containing the walk-related actions (walking, walkingdog, walkingtogether), the same as [Harvey et al. 2020] did. We work with a 25HZ sampling rate and take Subject 5 as the test subject.

For both datasets, we split the training set into multiple 50-frames windows. Similar to [Harvey et al. 2020] [Duan et al. 2021], two consecutive windows have 25 overlapped frames.

When training, the input of the current step is the output from the last step so that the error accumulates as the generated sequence grows. This effect increases the robustness because the network learns a pose not only from the dataset.

4.2 Training of Motion Manifold

The 50-frames sequence is still too long for learning motion manifold to convergence efficiency. We equally split the 50-frames sequence into two 25-frames sequences before training the architecture.

The bone length loss constrains the joints' velocity to zero, generating a weird motion sequence. To avoid this, we train the architecture twice. We first use the reconstruction loss and KL-divergence loss, teaching the network to predict an approximate pose. Secondly, we add foot skating loss and bone length loss.

Inspired by Ling et al. [2020], we pass the latent variable z and the future hip velocity v_h^{i+1} to every layer of the expert network to avoid posterior collapse. The encoder of the CVAE has two hidden layers with 256 units followed by ELU activation. The gating network has two hidden feed-forward layers followed by ELU activation. The output layer of the gating network uses Softmax activation. Each of expert network is a three-layer feed-forward network with 256 units in the hidden layers followed by ELU. In our preliminary experiments, six expert networks achieve good results and fewer than six lead to worse accuracy. We therefore empirically set the expert number to six in our implementation.

The scheduled sampling strategy is employed the first time. The network takes the predicted pose the last timestep as input with probability p. Otherwise, it takes the ground truth from the dataset as input. The probability starts at 0 for the first k epochs and then increases to 1 linearly for another k epochs. We set $k = 5$ for the Lafan1 dataset and $k = 20$ for the Human3.6M dataset.

We use AMSgrad optimizer with adjusted parameters ($\beta_1 = 0.5, \beta_2 = 0.9$). At the first training time, the learning rate is initialized to 1e-4 and linear decreases to 1e-5 by 50,000 iterations. The learning rate starts at 0 for the second time and increases to 1e-4 for ten epochs so that the added losses do not significantly change the network's parameters. After ten epochs, the learning rate decreases with the same decreasing rate as the first time. We scale all losses to be approximately equal to 1 for an untrained network without employing extra weights.

For the transition sampler, the target frame, current frame, and offset are encoded by the target, state and offset encoder, respectively. Subsequently, the embeddings z_{dt} and z_{target} are added to the encoded vectors. Then an LSTM takes the encoded vectors to produce the next state. Finally, the parse decoder takes the state and outputs the upper joints $\dot{r}_U^{i+1}$ and the sample (z, v_h^{i+1}).

4.3 Training of Transition Sampler.

After training the CVAE, we remove its encoder, fix its decoder, and connect the transition sampler to the fixed decoder to train the sampler.

All encoders are feed-forward networks with a hidden layer of 512 units and an output layer of 256 units. All layers use PLU as the activation. The LSTM has 1024 units in the hidden layer. The parse decoder has two hidden feed-forward layers with 512 units in the first layer and 256 units in the second layer. Both layers are followed by ELU.

During training, we sample a transition length from 5 to 30 frames from a window in each learning step so that the network can learn from different transition lengths and the target frames.

The AMSgrad optimizer is also employed for training the transition architecture. The learning rate equals 1e-3, the weights for L_{rot}, L_{leg} are 1 and the weights for $L_{pos,rot}, L_{bone}, L_{foot}$ equals 0.5. We train the transition architecture for 300,000 iterations costing approximately one day.

5 EXPERIMENTS AND RESULTS

We conduct our experiments on a PC with an Nvidia RTX 2080 graphics card, with an AMD 3950x CPU and 32G memory. Our method takes on average 2.1 ms to synthesize one frame, which is sufficient for real-time applications. As in similar research, real-time in-between motion generation requires high-quality data. Therefore, we mainly use the Lafan1 dataset [Harvey et al. 2020] for its good quality and diversity in motion styles. To further test the generalizability of our method, we also validate it on the Human3.6M dataset [Ionescu et al. 2011] [Ionescu et al. 2014] and compare it with previous methods. Unless specified otherwise, the following experiments are conducted on the Lafan1 dataset and all models are trained with transition lengths of 5 to 30 frames (see supplemental material for details).

Our data split for training/testing is similar to [Harvey et al. 2020]. Each test window contains 65 frames, sampled from Subject 5 of both datasets. Two consecutive windows have 25 overlapped frames. Our evaluation focuses on the motion quality, transition quality and model generalizability under unseen control signals. We employ both qualitative visual evaluation and quantitative metrics. The quantitative metrics include reconstruction accuracy given a seen target frame and transition duration, evaluated by Normalized Power Spectrum Similarity (NPSS) in the joint angle space and the average L2 distance of global joint position between the predicted results and the ground truth. These metrics are good indicators of transition quality, i.e., testing whether multi-modal transitions are captured in detail. Note that although we add a bone-length loss term during training, it cannot keep the lengths of the bones constant. For a 30-frames transition motion, the average bone length error is 0.64 cm. For a fair comparison with RTN, we first transform the joint positions to joint rotations, and then obtain their joint positions by FK before we compare their joint position accuracy. In addition, we also employ a foot skating metric to evaluate the motion quality [Zhang et al. 2018]. This metric checks whether the motion manifold learns a reasonable pose under a given velocity. The foot skating metric averages the foot velocity v_f over the ground if the foot height h is within a threshold H. Since there is foot skating in the ground-truth, we empirically set H to 2.5 cm. The metric is defined as:

$$L_f = v_f \cdot \text{clamp}(2 - 2^{h/H}, 0, 1). \tag{15}$$

5.1 Ablation study

Pose Representation Previous research uses joint positions, joint angles or both [Holden et al. 2020, 2017]. To test which representation works the best for our manifold model, we conduct an ablation study and focus on the foot skating, shown in Table 1. The results

Table 1. Comparisons of foot skating between position-based representation and rotation-based representation.

	Foot skate		
Frames	5	15	30
Rotation-based	0.934	1.035	1.161
Position-based	**0.356**	**0.373**	**0.401**

Table 2. Comparisons between the Full-body network and our method. Both models are trained with 300,000 iterations.

	L2 norm of global position		
Frames	5	15	30
Full-body	0.259	0.612	1.143
Our method	**0.196**	**0.562**	**1.124**
	NPSS		
Full-body	0.00671	0.07471	0.35385
Our method	**0.00554**	**0.07026**	**0.34549**

are similar to existing research [Wang et al. 2021] in that joint position representation can effectively mitigate the foot skating. Note that our model still explicitly models joint angles, which is normally required for animation purposes. Using joint positions here acts as a regularization term to facilitate learning.

Motion manifold focus on the lower-body joints. Our focus on the lower-body and the hip in natural motion modeling is a different design choice compared with recent deep learning research. This is because the lower-body motions are relatively simple in locomotion but highly important for motion quality due to foot skating. Therefore, we prioritize these joints in learning the manifold and leave the correlation learning between the upper-body and the lower-body to the transition sampler. Alternatively, we can also model the whole body directly but the relatively unconstrained upper-body motion can introduce ambiguity in learning.

To prove this, we add the upper-body joints to the CVAE and remove the upper joints' rotation difference $\dot{\mathbf{r}}_U$ from the output of the transition sampler. We refer to this network as the Full-body network. A comparison is shown in Table 2. Smaller reconstruction errors in both the joint angle and position space indicate that better transitions are learned. Note that in both networks, our method still predicts the full body. The larger errors in Full-body are likely to be caused by its predictions being closer to the 'averaged' motion. By separating the modeling of the lower-body and upper-body joints, our method manages to improve the learning.

5.2 Evaluation and Comparison

Table 3. Comparisons of reconstruction accuracy and foot skating of different methods. All models are trained with 300,000 iterations.

	L2 norm of global position		
Frames	5	15	30
Interpolation	0.37	1.24	2.31
RTN	0.22	0.59	1.16
+skating loss	0.28	0.68	1.27
Auto-Encoder	0.28	0.63	1.16
VAE	**0.20**	**0.56**	**1.11**
Our method	**0.20**	**0.56**	1.12
	NPSS		
Interpolation	0.0073	0.1135	0.5229
RTN	0.0056	0.0719	0.3495
+skating loss	0.0071	0.0799	0.3715
Auto-Encoder	0.0078	0.0835	0.3721
VAE	**0.0055**	**0.0701**	**0.3420**
Our method	**0.0055**	0.0702	0.3455
	Foot skate		
Ground Truth	0.162	0.141	0.143
Interpolation	1.708	2.081	2.144
RTN	0.483	0.698	0.930
+skating loss	0.249	0.349	**0.455**
Auto-Encoder	0.294	0.485	0.649
VAE	0.255	0.353	0.502
Our method	**0.244**	**0.343**	0.469

5.2.1 Motion quality. Foot sliding is an important metric for motion quality. Earlier research suffers from such problems due to 'averaged' motions and drifting issues [Fragkiadaki et al. 2015]. A common strategy is to have a post-processing stage [Wang et al. 2021]. More recent work tends to mitigate this problem e.g. by inducing contact patterns [Starke et al. 2020] when dense temporal control signal is available, or constraining the distribution of the generated motions to be similar to that of the data [Harvey et al. 2020]. However, while post-processing is highly undesirable in our application, predicting contact patterns is also not straightforward with arbitrary target frames. Constraining the distribution seems effective, but it is still not easy to mitigate the foot skating. To show this, we compare RTN [Harvey et al. 2020] and its variant with a foot skating loss (+skating loss). As a naive baseline, we also experiment with linear interpolation (Interpolation). Results are shown in Table 3. First, adding an additional foot skating loss to RTN mitigates foot skating to some extent (Table 3 Bottom). However, it also leads to worse reconstruction accuracy. During learning, the predicted contact step might be different from the step from the ground truth, so their loss induces an incorrect backward gradient and make the pose unnatural. The results in the video show that adding the foot skating loss also sometimes causes unsmoothed transitions near the start and the end frame. Although our method predicts the foot contact step, the inconsistent backward gradient doesn't affect the pose unnaturally but helps to adjust the hip velocity because the pose is sampled from the manifold.

Further, to also show the importance of controlling the latent distribution of the motion dynamics, we replace our CVAE with a plain autoencoder (Auto-Encoder) so that the distribution of z is not constrained (i.e., without the KL-divergence loss). Besides, to show the importance of using the hip velocity as a condition, we replace our CVAE with a VAE so that the latent distribution is unconditional. Autoencoder and VAE are widely used for learning motion manifolds [Harvey et al. 2020; Holden et al. 2020; Ling et al. 2020; Wang et al. 2021]. The results are shown in Table 3. By explicitly controlling z with conditioned decoding on the hip velocity, our method performs similarly to the VAE in reconstruction accuracy (i.e., L2 error and NPSS), significantly better than the Auto-encoder. Further, our method outperforms VAE in foot skating by large margins especially from the 30th frame, greatly improving the motion quality.

We compare our method with RTN on the Human3.6M dataset, shown in Table 4. While RTN achieves slightly better results in NPSS by as large as 7.6%, our method outperforms RTN in both the L_2 norm and the foot skate by 10.3% and 69.2%, respectively. When looking at the difference between the Human3.6M and the Lafan1, the most significant difference is the skeleton variation between subjects. The Human3.6M has a larger variation. We speculate this might be the reason for the slightly worse results in NPSS and also why at frame 5, Interpolation even outperforms both methods. However, since neither methods explicitly aim to generalize to different skeletons, we leave the analysis to future work.

Table 4. [Human3.6M]Comparisons of different frames between RTN and our method. Both models are trained with 300,000 iterations.

	L2 norm of global position		
Frames	5	15	30
Interpolation	0.78	1.85	2.59
RTN	0.53	0.98	1.50
Our method	**0.47**	**0.93**	**1.44**
	NPSS		
Interpolation	**0.0044**	0.0651	0.3132
RTN	0.0049	**0.0549**	**0.2298**
Our method	0.0054	0.0554	0.2386
	Foot skate		
Ground Truth	0.070	0.089	0.089
Interpolation	0.824	1.525	0.929
RTN	0.325	0.363	0.456
Our method	**0.100**	**0.198**	**0.317**

In addition, we also compare our method with the state-of-the-art offline motion completion method, which is based on a transformer [Duan et al. 2021]. The results reported in Table 5 shows that our method can generate comparable results.

Table 5. Comparisons between our method and the transformer.

	L2 norm of global position		
Frames	5	15	30
Transformer	0.22	**0.56**	**1.10**
Our method	**0.20**	**0.56**	1.12

Table 6. Comparisons of different kinds of actions. The number of frames of the generated sequences is 30. Both models are trained with 300,000 iterations.

	L2 norm of global position			
Actions	Walk	Dance	Jump	Obstacle
Interpolation	2.76	2.40	1.89	2.23
RTN	0.99	1.51	1.21	1.21
Our method	**0.95**	**1.48**	**1.18**	**1.15**
	NPSS			
Interpolation	0.6430	0.6405	0.4000	0.4513
RTN	0.3380	0.5197	**0.3123**	0.3158
Our method	**0.3306**	**0.5141**	0.3205	**0.3085**
	Foot skate			
Ground Truth	0.160	0.230	0.155	0.121
Interpolation	2.743	1.844	1.381	2.024
RTN	1.187	1.103	0.640	0.657
Our method	**0.589**	**0.571**	**0.326**	**0.293**

5.2.2 Generalization. Since the target frame and aimed transition duration can be used as control signals, evaluating models under unseen and extreme user control is crucial. We validate our method via multiple aspects: motion style, aimed transition duration and the distance between the start and end frame.

Different motion styles. We split the Lafan1 dataset into multiple subsets by motion styles. All the walking and running sequences are categorized into a 'walk' set. The 'dance' set includes dance sequences, fight sequences and sports sequences. A 'jump' set collects all the jump sequences, and an 'obstacles' set contains all the character's motions across the obstacles. The results are shown in Table 6. Since the ground truth data also contains foot skating, we use it as a baseline too. The results show that our method can universally improve the results. While being able to achieve better reconstruction, our method provides far better foot skates. This is demonstrated across different motion styles.

Different transition duration. To test our method can handle long and short duration as the aimed transition time, we discard 1 second (30 frames) from each sample, and require the network to generate the sequences at speed scaled to 2x (15 frames), 4x (8 frames) and 0.5x (60 frames) respectively. Besides, we have made experiments with an extreme case, in which we slow down the speed 100 times (0.01x). The results are shown in Table 7. Interpolation performs best in this situation because the drifting is divided by 100 times. However, there is no valid pose change in interpolation results.

Slowing down motions is theoretically easier as foot contact can still be generated. Our method outperforms RTN in keeping foot contact. Comparatively, speeding up is more difficult because it might be impossible for a character to reach the target during such a short time. This is why the foot skating in 4x and 2x are in general worse. However, our method still provides the best results. Visually, we find that if the start and the end frame are within the same phase of a walking cycle, for example, when the left leg is behind the right leg, RTN motion tends to drift to the target without changing pose,

Table 7. Comparisons of foot skating of changing transition duration between RTN and our method. Both models are trained with 300,000 iterations.

	Foot skate			
Speed	4x	2x	0.5x	0.01x
Ground Truth	0.161	0.148	0.156	0.149
Interpolation	7.302	3.917	1.075	**0.004**
RTN	4.050	2.087	0.814	0.522
Our method	**3.363**	**1.350**	**0.438**	0.110

while our results make fast footsteps. These results can be seen in the accompanying video.

Different target locations. Another dimension is to vary the location of the target frame. We conduct two experiments. The first one has the target frame set further away, along the direction from the start to the target frame (Forwarding). In addition, we also move the target frame in the opposite direction (i.e., from the original target frame to the starting frame), to a location that is twice further away from the starting frame compared with the original target frame (Backwarding).

In Forwarding, our method generates fewer bigger steps or more small steps to fill the gap without visible artifacts. RTN usually performs at the same pace as the ground truth but fills the distance gap by drifting. Backwarding is an extreme testing case and is challenging for both methods. Since the poses of the start frame and the target frame have similar orientations and there is not enough time for turning twice, both methods try to achieve the target frame by walking backward. However, RTN always generates visible artifacts while our method does not. We speculate that this is because the data does not contain enough clips for the character to move backwards. However, the motion manifold captured the motions with backward velocity in the training set, which helps to synthesize the natural action successfully. Results of both Forwarding and Backwarding can be found in the video. Here, we show an extreme Forwarding case where the target frame is 10 meters away from the starting frame, where the longest distance in the training set is merely 5.79 meters. To generate a 60-frames motion, the character must have a speed exceeding $5m/s$ ($7.9m/s$ for the farthest distances) to reach the target.

Figure 4 shows a visual comparison among RTN, VAE and our method. RTN (the first row) generates motions that drift towards the goal because the target frame is too far (i.e., floating in the air in the middle of the motion). In addition, without the condition of the hip velocity, the poses synthesized by VAE do not conform to the movements, leading to unnatural motions (i.e., the poses are similar to the blending of crawling and running). Comparatively, our method generates the most natural motions under this extreme case.

6 LIMITATIONS

Similar to other data-driven methods, one major limitation of our method is that our generation results are limited by the training data. It cannot generate motions that are too different from the training data. One example is that motion generation will become quite challenging when we place the target frame behind the starting frame. The reason is that there is no quick turning or enough backward walking motion in the training data. Another example is that we cannot guarantee the target frame is 100% achieved if the target frame is too different from data, spatially, temporally, or both. However, we argue that our framework itself is still effective and the aforementioned problems can be easily overcome when more diverse data are introduced.

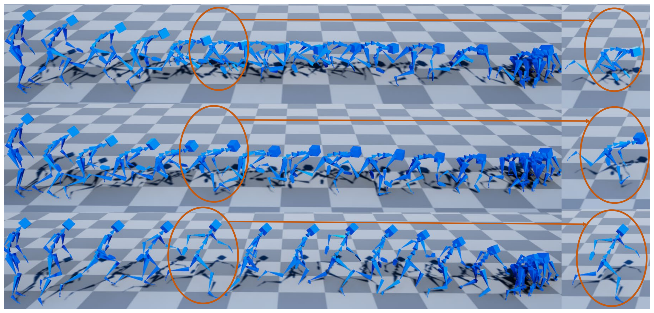

Fig. 4. We sample the pose every 5 frames. The first row shows the RTN's results, the second row shows the VAE's results, and the last row shows our results. For a clearer observation, we sample one frame from each sequence and put it at the rightmost row. The results indicate that our method generates the most natural running motion.

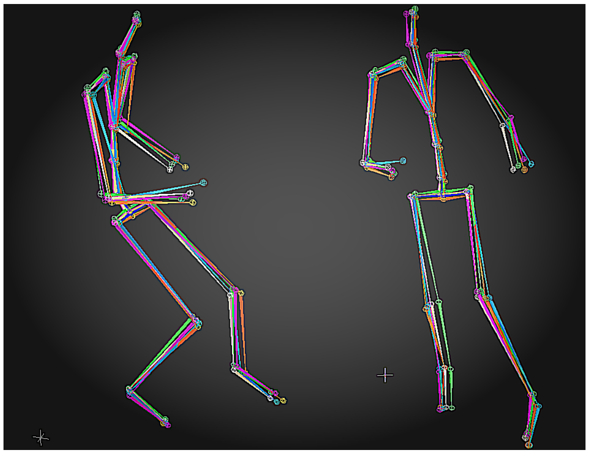

Fig. 5. The figure shows the midway in a 30-frames transition re-sampled 5 times. The upper body shows the diversity.

Lacking of motion diversity is another limitation of our method. As a CVAE-based network, our model can indeed generate different motions for the same control, but the differences of the generated motions are small, especially for the lower body (as shown in Figure 5). To generate high-quality motion under unseen control, we assume that the contact position with the ground of each step cannot change too much.

7 FUTURE WORK

Although specifying the motion duration is widely used in the animation/game pipelines to control the timing of motions and transitions, we will provide automated computation for desired motion duration in future. Given the robustness of our model under different timing requirements, we will compute a reasonable duration by modeling the distribution of possible timing requirements automatically. In addition, we will take more factors into considerations, such as motion styles, interactions with environments, and various skeletal topologies (e.g. quadrupeds).

8 CONCLUSIONS

We proposed a novel learning framework consisting of a new natural motion manifold model and a new transition sampler for real-time in-between motion generation. The motion manifold model treats different body parts separately and focuses on controllability and motion quality, while the transition sampler ensures natural motions are generated with respect to user control. Our model generates high-quality motions in mitigating foot skating and motion transitions so that it can be used for both offline animation and online games. Our method is also general under unseen control signals. It outperforms alternative solutions and the state-of-the-art methods.

ACKNOWLEDGMENTS

Xiaogang Jin was supported by the National Natural Science Foundation of China (Grant Nos. 62036010, 61972344), the Ningbo Major Special Projects of the “Science and Technology Innovation 2025” (Grant No. 2020Z007), and the Key Research and Development Program of Zhejiang Province (Grant No. 2020C03096).

REFERENCES

Okan Arikan and D. A. Forsyth. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics* 21, 3 (2002), 483–490.

Philippe Beaudoin, Stelian Coros, Michiel van de Panne, and Pierre Poulin. 2008. Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 117–126.

Jinxiang Chai and Jessica K. Hodgins. 2007. Constraint-based motion optimization using a statistical dynamic model. *ACM Transactions on Graphics* 26, 3 (2007), 8–es.

Wenheng Chen, He Wang, Yi Yuan, Tianjia Shao, and Kun Zhou. 2020. Dynamic future net: diversified human motion generation. In *Proceedings of the 28th ACM International Conference on Multimedia*. 2131–2139.

Hsu-kuang Chiu, Ehsan Adeli, Borui Wang, De-An Huang, and Juan Carlos Niebles. 2019. Action-agnostic human pose forecasting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 1423–1432.

Yinglin Duan, Tianyang Shi, Zhengxia Zou, Yenan Lin, Zhehui Qian, Bohan Zhang, and Yi Yuan. 2021. Single-Shot Motion Completion with Transformer. *arXiv:2103.00776 [cs]* (March 2021).

Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*. 4346–4354.

Félix G. Harvey and Christopher Pal. 2018. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs (SA '18)*. Association for Computing Machinery, 1–4.

Félix G. Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust motion in-betweening. *ACM Transactions on Graphics* 39, 4, Article 60 (2020).

Alejandro Hernandez, Jurgen Gall, and Francesc Moreno-Noguer. 2019. Human motion prediction via spatio-temporal inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 7134–7143.

Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. 2020. Learned motion matching. *ACM Transactions on Graphics* 39, 4, Article 53 (2020).

Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics* 36, 4 (2017), 1–13.

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

Catalin Ionescu, Fuxin Li, and Cristian Sminchisescu. 2011. Latent structured models for human pose estimation. In *2011 International Conference on Computer Vision*. 2220–2227.

Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. 2014. Human3.6M: large scale datasets and predictive methods for 3D human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 36, 7 (2014), 1325–1339.

Ashesh Jain, Amir R. Zamir, Silvio Savarese, and Ashutosh Saxena. 2016. Structural-RNN: deep learning on spatio-temporal graphs. In *2016 IEEE Conference on Computer Vision and Pattern Recognition*. 5308–5317.

Manuel Kaufmann, Emre Aksan, Jie Song, Fabrizio Pece, Remo Ziegler, and Otmar Hilliges. 2020. Convolutional autoencoders for human motion infilling. In *2020 International Conference on 3D Vision*. 918–927.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2008. Motion graphs. In *ACM SIGGRAPH 2008 Classes (SIGGRAPH '08)*.

Sergey Levine, Jack M Wang, Alexis Haraux, Zoran Popović, and Vladlen Koltun. 2012. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.

Jiaman Li, Ruben Villegas, Duygu Ceylan, Jimei Yang, Zhengfei Kuang, Hao Li, and Yajie Zhao. 2021. Task-generic hierarchical human motion prior using vaes. In *2021 International Conference on 3D Vision*. IEEE, 771–781.

Hung Yu Ling, Fabio Zinno, George Cheng, and Michiel van de Panne. 2020. Character controllers using motion VAEs. *ACM Transactions on Graphics* 39, 4, Article 40 (2020).

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

Jianyuan Min and Jinxiang Chai. 2012. Motion graphs++: a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics* 31, 6, Article 153 (2012), 12 pages.

Dario Pavllo, Christoph Feichtenhofer, Michael Auli, and David Grangier. 2020. Modeling human motion with quaternion-based neural networks. *International Journal of Computer Vision* 128 (2020), 855–872.

Mathis Petrovich, Michael J. Black, and Gül Varol. 2021. Action-Conditioned 3D Human Motion Synthesis with Transformer VAE. *arXiv:2104.05670 [cs]* (2021).

Davis Rempe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. 2021. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 11488–11499.

Alla Safonova and Jessica K. Hodgins. 2007. Construction and optimal search of interpolated motion graphs. *ACM Transactions on Graphics* 26 (2007).

Yijun Shen, He Wang, Edmond S. L. Ho, Longzhi Yang, and Hubert P. H. Shum. 2017. Posture-based and action-based graphs for boxing skill visualization. *Computers and Graphics* 69, Supplement C (2017), 104–115.

Sebastian Starke, Yiwei Zhao, Taku Komura, and Kazi Zaman. 2020. Local motion phases for learning multi-contact character movements. *ACM Transactions on Graphics* 39, 4, Article 54 (July 2020).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*. 5998–6008.

He Wang, Edmond SL Ho, and Taku Komura. 2015. An energy-driven motion planning method for two distant postures. *IEEE Transactions on Visualization and Computer Graphics* 21, 1 (2015), 18–30.

He Wang, Edmond S. L. Ho, Hubert P. H. Shum, and Zhanxing Zhu. 2021. Spatio-temporal manifold learning for human motions via long-Horizon modeling. *IEEE Transactions on Visualization and Computer Graphics* 27, 1 (2021), 216–227.

He Wang and Taku Komura. 2011. Energy-based pose unfolding and interpolation for 3D articulated characters. In *Motion in Games*. 110–119.

He Wang, Kirill A Sidorov, Peter Sandilands, and Taku Komura. 2013. Harmonic parameterization by electrostatics. *ACM Transactions on Graphics* 32, 5 (2013), 155.

Andrew Witkin and Michael Kass. 1988. Spacetime constraints. *ACM Siggraph Computer Graphics* 22, 4 (1988), 159–168.

Yuting Ye and C. Karen Liu. 2010. Synthesis of responsive motion using a dynamic model. *Computer Graphic Forum* 29, 2 (2010), 555–562.

Ye Yuan, Umar Iqbal, Pavlo Molchanov, Kris Kitani, and Jan Kautz. 2021. GLAMR: Global Occlusion-Aware Human Mesh Recovery with Dynamic Cameras. *arXiv preprint arXiv:2112.01524* (2021).

He Zhang, Sebastian Starke, Taku Komura, and Jun Saito. 2018. Mode-adaptive neural networks for quadruped motion control. *ACM Transactions on Graphics* 37, 4 (2018), 1–11.

Xinyi Zhang and Michiel van de Panne. 2018. Data-driven autocompletion for keyframe animation. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games*. 1–11.

Yi Zhou, Jingwan Lu, Connelly Barnes, Jimei Yang, Sitao Xiang, et al. 2020. Generative tweening: Long-term inbetweening of 3d human motions. *arXiv preprint arXiv:2005.08891* (2020).