DOI: 10.1111/cgf.14920

COMPUTER GRAPHICS *forum*
Volume 42 (2023), number 6, e14920

Recurrent Motion Refiner for Locomotion Stitching

Haemin Kim,[1] Kyungmin Cho,[2] Seokhyeon Hong[1] and Junyong Noh[1]

[1]KAIST, Visual Media Lab, Daejeon, South Korea
{spot1223, ghd3079, junyongnoh}@kaist.ac.kr
[2]Anigma Technologies, Daejeon, South Korea
ckm@anigma-ai.com

Abstract

Stitching different character motions is one of the most commonly used techniques as it allows the user to make new animations that fit one's purpose from pieces of motion. However, current motion stitching methods often produce unnatural motion with foot sliding artefacts, depending on the performance of the interpolation. In this paper, we propose a novel motion stitching technique based on a recurrent motion refiner (RMR) that connects discontinuous locomotions into a single natural locomotion. Our model receives different locomotions as input, in which the root of the last pose of the previous motion and that of the first pose of the next motion are aligned. During runtime, the model slides through the sequence, editing frames window by window to output a smoothly connected animation. Our model consists of a two-layer recurrent network that comes between a simple encoder and decoder. To train this network, we created a sufficient number of paired data with a newly designed data generation. This process employs a K-nearest neighbour search that explores a predefined motion database to create the corresponding input to the ground truth. Once trained, the suggested model can connect various lengths of locomotion sequences into a single natural locomotion.

Keywords: character animation, motion processing, motion stitching

CCS Concepts: • Computing methodologies → Computer graphics; Animation; Motion processing

1. Introduction

Preparing every animation clip required in a project is a highly resource consuming process. Therefore, artists often reuse a few captured or manually created animation sequences by cutting out necessary frames and concatenating them in a desired way. Here, motion stitching plays an important role in connecting two different motions. Through motion stitching, users can attach the start of the second motion at the end of the first motion and interpolate the inbetween poses for a smooth transition. Many animation software resorts to this automatic stitching method equipped with various interpolation schemes to control or edit character animations.

While individual character motions may be of high-quality, the concatenated motion sequence may not be as natural as the original input. This is because the interpolation process in the current motion stitching methods fails to consider the complex nature of the motion data, often accompanied with visual artefacts. For instance, if the first motion ends with the right foot in front and the second motion starts with the left foot in front, applying interpolation will lead to a severe foot sliding artefact, unless the walking pattern of the character is properly considered. Therefore, additional manual work is typically performed to improve the quality of the results, which is undesirable in many applications that require automatic motion stitching.

In this study, we propose a novel motion stitching method that connects two locomotion clips of a character with possibly large pose differences as depicted in Figure 1. Our method preserves the locomotive pattern of the character while modifying the poses for continuous motion, significantly reducing visual artefacts even when a random phase of the character locomotion is given as input. Our framework begins by simply aligning the roots of given motions in order followed by refining the poses to make a seamless transition using a recurrent motion refiner (RMR), a recurrent neural network (RNN)-based network. Inspired by the encoder-recurrent-decoder (ERD) model [FLFM15], our network architecture consists of an encoder, RNN-based refiner, and decoder. We chose this architecture to make the refiner learn sequential information within the learned motion manifold. Furthermore, we use bidirectional gated

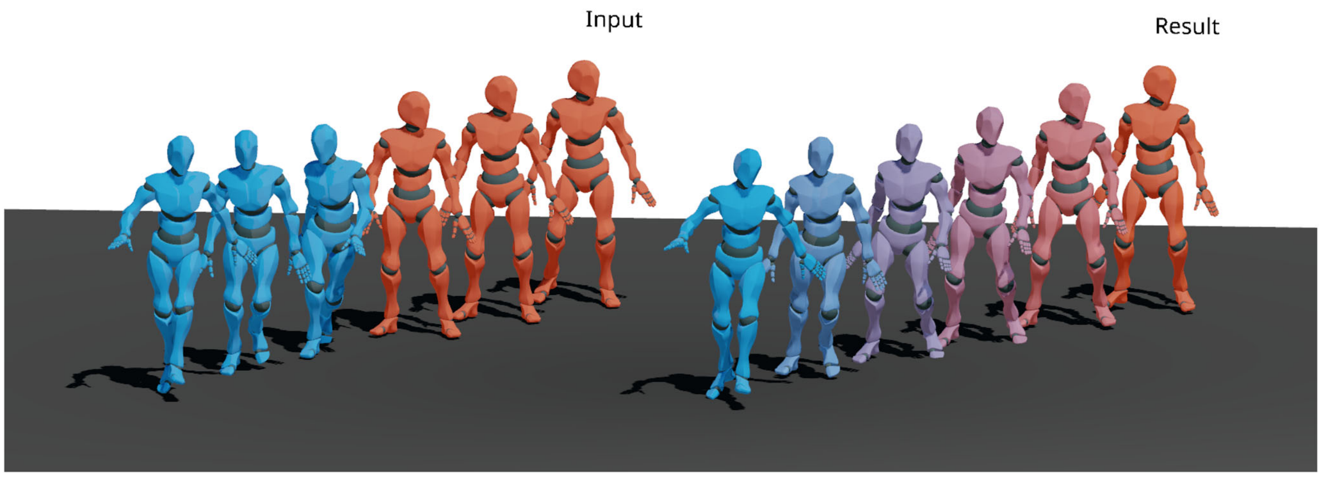


Figure 1: *Motion stitching conducted with our method. The left sequence indicates two different locomotion clips and the right sequence indicates the stitched result. The root position is translated for clear visualization.*

recurrent unit (BGRU) layers for our refiner to take advantage of the forward and backward information and effectively smooth out possible discontinuities present between the input motions.

For our network to effectively handle various stitching combinations of locomotion data, we designed a new data pair generation method. We prepared a large-scale dataset of two or three motions and a corresponding stitched motion that represents the ground truth. The stitched motions must be as natural as real human motion to be used as ground truth data. This is extremely difficult, however, to acquire large-scale data with such quality, because it requires carefully plotted motion capture with intense manual edits. Therefore, we instead started from the ground truth and exploited it to obtain corresponding input motion clips using an algorithm inspired by motion matching [BC15]. We employed a K-nearest neighbour (KNN) search to acquire two or three input motion clips that resemble the ground truth. For this process, by controlling parameters, we generated various combinations of input motions, including samples that are likely to produce artefacts when connected with current methods. With this data, our network learns to generate high-fidelity motions by eliminating the pose discontinuity that may occur when independent locomotion clips are concatenated.

Our method can stitch multiple motions during runtime. First, we align a few input motion clips into one long sequence. Then, our network edits the frames within the short window, while repeatedly shifting the window along the timeline. All input motion clips are fused into one natural locomotion when the network updates the last pose. By inferring one short segment at a time, we are able to combine multiple motions in a unified way and acquire a resulting motion with minimal artefacts. In addition, we expect that the proposed technique can be applied to generation methods such as motion matching[BC15]. When connecting reference motions, our method can refine possible pose differences and mitigate foot sliding artefacts that may occur in the process.

We demonstrated how the suggested method can stitch various input locomotion clips. We validated our design choices by comparing the results produced with different RNN cells and conducting an ablation study on the loss terms. We also compared the output with the result from an interpolation-based method and motion inbetweening method. These experiments verified that our method can be used in many applications that require automatic motion stitching with improved quality.

Our technical contributions can be summarized as follows:

- Novel data pair generation for motion stitching using KNN search. The KNN search encourages data variety.
- Employment of BGRU to incorporate the intricate nature of locomotion when performing motion stitching. No assumption is required for a specific phase of locomotion due to the use of BGRU.
- Window-based motion refinement algorithm that effectively stitches multiple motions at runtime. The employment of the adjustable inference window allows for stitching diverse lengths of motions.

2. Related Work

2.1. Blending-based motion synthesis

Synthesizing a new motion from existing motion clips has been extensively studied by many researchers. Early studies produced a new motion that corresponds to the point in parameterized space through the interpolation of example motions. For the underlying interpolation mechanism, linear [WH97, GR96], radial basis function (RBF) [RCB98], KNN [KG04], geostatistics [MK05], optimization [HK10], and hand-crafted tetrahedron [FXS12]-based blending have been adopted. RBF-based motion interpolation was further improved by inserting pseudo-examples to the interpolation space [RISC01] or combining incremental time warping [PSS02, PSKS04] to blend motions of different speeds.

To obtain the best results through these methods, it is crucial to select motions that are structurally similar. Therefore, many researchers suggested methods that can compare and choose motions to interpolate. Rose et al. [RCB98] manually labelled the constraint information of example motions and Kovar et al. [KG03] automated this process by creating registration curves. Other methods based on directed graphs adopted various distance metrics to compute the similarity between motions. Joint positions and velocities with respect to the torso [AF02], or a point cloud of the character [KGP02, HG07] were used during this process. These metrics pre-computed the closeness of the example motions and automatically labelled the most probable transition points. In contrast, our method does not directly measure the distance between each data. We train a neural network on the dataset that encompasses various situations of connecting motions and let the network figure out the phase or pose information needed for seamless transitions.

Inertialization blending (IB) [Bol16, Bol17] computes the displacement between the source and target poses while keeping the velocity of the source. This displacement is interpolated and then added to the target poses for the number of frames to blend. Unless the source and target phases significantly differ in given locomotions, these methods generally produce smooth transitions. Our model goes further than interpolating the motion data by employing a neural network to edit the given poses and combine different locomotion clips into one natural locomotion.

2.2. Data-driven motion denoising

In the context of smoothing discontinuity in the motion data, our problem can be interpreted as motion denoising. Early studies aimed at finding the bases that best represent the motion or applying a filter to it to remove noise in the data. Tangkuampien et al. [TS06] employed a variant of principal component analysis (PCA) to take advantage of the denoising ability of kernel PCA. Lou et al. [LC10] learned multiple filter bases from available motion capture data. Akhter et al. [ASK*12] constructed a bilinear spatiotemporal basis model. Other studies attempted to use sparse representation to reconstruct clean motion data [XFJ*15, FJX*14, XSZF16].

The employment of deep learning made it possible to use large-scale training data and acquire a compact model that can be generalized to unseen sequences. Fully connected layers were used to construct hierarchical temporal encoders to recover the missing joints [BBKK17]. Convolutional Neural Networks (CNN) were also employed to learn the valid motion manifold and denoise a motion by projecting it onto the manifold and converting it back to the original representation [HSKJ15]. Later studies combined RNN and fully connected layers to model sequential information[MLCC17, LZZ*19, LZZL20]. Recently, Cui et al. [CS21] adopted a combination of graph convolutional and temporal convolution networks to exploit spatial information between joints. The goal of these studies and ours is similar. While they assume the input data from one natural sequence, we consider the input as separate sequences and recreate realistic poses to naturally concatenate the pieces of motions.

2.3. RNN-based motion prediction

A motion prediction task that synthesizes the frames following the past frames has been developed with recent advances in deep learning. Among the diverse network architectures, RNN-based networks have gained attention from many researchers. Fragkiadaki et al. [FLFM15] suggested an ERD model that places fully connected layers before and after the recurrent layers to make the recurrent part learn within a valid motion manifold. To alleviate the ambiguity that occurs when inferring long sequences, Habibie et al. [HHS*17] added user control signals and combined a Variational Autoencoder (VAE) with an RNN. Chiu et al. [CAW*19] utilized a multi-scale RNN to encode temporal dependencies. Other studies sought to improve the performance by incorporating spatial information. Diverse architectures such as a separate autoencoder [GSAH17], spatial encoder and decoder [WHSZ19], or spatiotemporal attention mechanism [SZQ*21] are used to encode correlation between the joints. The transformer architecture [VSP*17] was also employed with a spatial and temporal attention module [AKCH21]. In our model, the proposed RNN-based network is less affected by the ambiguity problem because the model is aware of the last frame of the given sequence and produces a result by traversing the given sequence back and forth.

RNN-based approaches can also suffer from error accumulation due to the use of an autoregressive inference scheme. Some studies added Gaussian noise to the input to reduce this error [JZSS16, WCX19]. Other studies mitigated this issue by feeding the previous output of the network along with the ground truth data during training [LZX*17, GMK*19, PGA18]. To avoid extensive parameter tuning associated with these techniques, some researchers formulated this problem as a sequence-to-sequence task and always input predicted values followed by penalizing the error through separate loss terms [MBR17] or discriminators [GWLM18]. Unlike these autoregressive models, our method updates frames at each time-step based on the hidden vectors and corresponding input frames. During runtime, our model predicts the frames using the previous output. To prevent error accumulation by preserving the input poses, we designed a data generation process and added loss terms.

2.4. Motion in-betweening

Many motion in-betweening techniques employ RNN-based networks to explicitly model the sequential features of the motion data. Harvey et al. [HP18] augmented the ERD model[FLFM15] and suggested a Recurrent Transition Network (RTN) which uses separate encoders for the current frame, offset, and target. Geleijin et al. [GRvSD21] made the RTN lightweight and used quaternion values for joint rotations. Similarly, Zhang et al.[ZvdP18] used both linear and recurrent layers to generate in-between frames of the Luxo character. Harvey et al. [HYNP20] extended RTN [HP18] and introduced additive embedding modifiers to make the network produce temporal variations when given the same input keyframes. The approach introduced in Tang et al. [TWH*22] combined the conditional VAE with an RNN-based network similar to that used in Harvey et al. [HYNP20]. This network learned the motion manifold to sample the lower body movement, which plays a critical role when determining the motion quality.

CNN has also been employed for the motion in-betweening task. Hernandez et al. [HGMN19] mainly used the Generative Adversarial Network (GAN) architecture with one- or two-dimensional CNN to complete occlusions in the motion data. Kaufmann et al. [KAS*20] interpreted this problem as an image infilling task and built a deep CNN-based autoencoder to reflect a wide range of

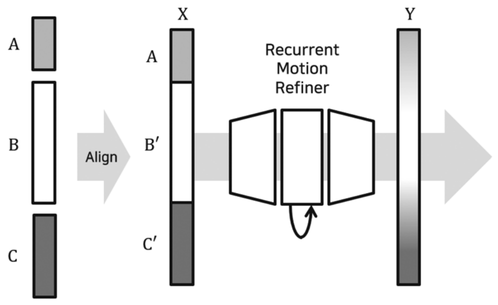


Figure 2: *System overview.*

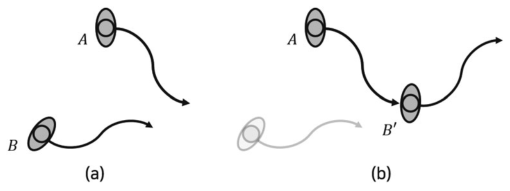


Figure 3: *Top view of input motions A and B. The black arrow represents the root trajectory. (a) Before alignment, the root trajectories of motion A and B are disconnected from each other. (b) After transforming motion B to start at the end of motion A, the two root trajectories form one continuous curve.*

frames. Zhou et al. [ZLB*20] generated local joint movements first, followed by predicting the corresponding root trajectory. Instead of using RNN or CNN, Li et al. [LVC*21] suggested a skeleton-aware architecture that includes skeleton convolution, pooling, and unpooling. Transformer-based methods have also been explored by using the BERT [DCLT18] architecture [DSZ*21], incorporating additional pose conditions [KBS*22], and predicting delta motion [OVH*22]. Because these approaches receive sparse poses as input, it is important to provide the most probable pose for a specific time-step for satisfactory results. In contrast, if less plausible poses are provided in the input sequence at a given time-step, our network can modify the poses into a natural motion.

3. System Overview

As shown in Figure 2, our method receives multiple locomotion clips, A, B and C, with various lengths as input. Because a mismatch in the root trajectory would degrade the motion quality, we first align motion B to motion A, and motion C to B′ to ensure the continuity in the root trajectory, as depicted in Figure 3. Here, we consider the aligned motions as a single motion X with length T. Note that X may have abrupt changes in the speed of root transformation or large differences in pose within a frame where motion A, B′, and C′ meet. These artefacts are removed by the use of RMR, which receives input X and outputs a smooth and natural motion Y.

In the following sections, we first explain how the training data for the motion stitching task is generated from random locomotion data (Section 4). We then describe how the RMR is trained (Section 5) and used to stitch multiple motions at inference time (Section 6).

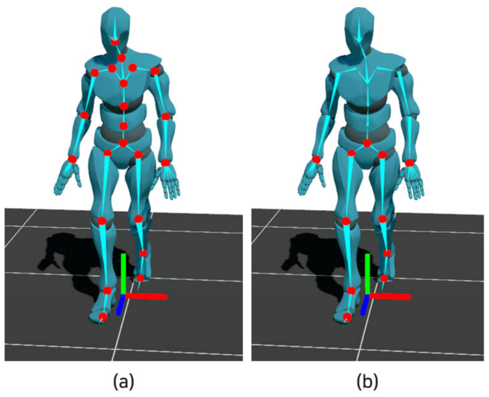


Figure 4: *Joints and the root used in pose representation. The joints are indicated with red circles and the root is shown with the three axes. (a) A total of $J = 22$ joints are used for the full body. (b) For the lower body, $J_{lower} = 11$ joints are used.*

4. Training Data Generation

The goal of generating the training data is to create two or three motion segments that sum up to length T. These segments are expected to be the ground truth Y^{gt} with length T when they are stitched and edited by experts. Using two or three motion segments prepares our model to deal with cases in which there are more than two motions in a single inference window. It is possible to use more than three motion segments, but using two or three segments produced natural results in all of our experiments. The detailed description about the inference window is provided in Section 6.

The ground truth clips of length T were created by sliding through motion capture sequences, shifting by $T/2$ frames. We project the hip joint of the character to the xz-plane and use it as the root transformation of the character throughout our work. Let $Y^{gt} = [x_1^{gt} x_2^{gt} \dots x_T^{gt}] \in \mathbb{R}^{d\times T}$ be the matrix representing locomotion data where x_t^{gt} is the pose at time-step t, which is defined as follows:

$$x_t^{gt} = \{r_t, p_t, q_t, v_t, c_t\} \in \mathbb{R}^d.$$

Here, $r_t = \{r_x, r_\theta, r_z\} \in \mathbb{R}^3$ is the root displacement vector, $p_t \in \mathbb{R}^{3J_{lower}}$ is the root-relative joint positions in the lower body, $q_t \in \mathbb{R}^{4J}$ is the parent-relative local rotations for all joints in quaternion, $v_t \in \mathbb{R}^{3J_{lower}}$ is the translational joint velocities with respect to the root of the previous frame, and $c_t \in \mathbb{R}^4$ is the binary foot contact vector. For the root displacement vector r_t, r_x and r_z each indicate the x-axis and z-axis translational displacement of the root, and r_θ indicates the rotational root displacement around the y-axis in radian. The foot contact vector was extracted from the left heel, left toe, right heel, and right toe joints with empirically selected thresholds. For the y position of the foot joints and toe joints, we used a threshold value of 2.5 and 1.6 cm, respectively. The joints used for the input data are shown in Figure 4.

First, we determine the length of motion segment A, B, and C by randomly sampling multiples of 10 that add up to the total frame number T. We denote these lengths as T_A, T_B, and T_C. While T_A and T_C are always larger than zero, the size of T_B can be zero. T_B

being zero means only two motion segments are generated. This sampling process allows our network to encounter diverse situations during training, so that it can effectively handle input motions of various lengths.

The first segment A is simply copied from the beginning of the ground truth $[\mathrm{x}_1^{gt} \dots \mathrm{x}_{T_A}^{gt}]$. This decision is based on the observation that artists often mainly edit the latter motion B when B is stitched to A. Although this may not always hold true, we found that this strategy leads to generating plausible data, as well as stabilizing the training by suggesting consistent tasks to the neural network.

For the second (B) and third (C) input motion, we search for the motion segments that are similar to $[\mathrm{x}_{T_A+1}^{gt} \dots \mathrm{x}_{T_A+T_B}^{gt}]$ and $[\mathrm{x}_{T_A+T_B+1}^{gt} \dots \mathrm{x}_{T}^{gt}]$, respectively. These segments are obtained from the reference motion database, under the assumption that the attributes in the input motion are preserved in the stitched motion. In the following, we briefly explain how the reference motion database is constructed and the KNN search is performed to generate motion segments B and C [BC15].

Let m be a motion feature that is defined as follows:

$$\mathrm{m} = \{w_1 \mathrm{j}_1, w_{T_l} \mathrm{j}_{T_l}, w_u \mathrm{u}\},$$

where $\mathrm{j}_t \in \mathbb{R}^{3J}$ denotes the joint positions at time-step t with respect to the root of the first frame of the motion, T_l is the length of motion, and $\mathrm{u} \in \mathbb{R}^{4T}$ is the root trajectory that consists of the 2D position and forward direction vector on the xz-plane for all frames. Here, we set weights w_1, w_{T_l}, and w_u as 0.1, 1.0, and 1.0, respectively. We use a smaller weight for j_1 than for j_{T_l} to find a motion with different poses from the ground truth in the first few frames while gradually becoming similar toward the end. More poses could be used to define the motion feature, but using the first and last pose worked well for all of our experiments.

A separate reference motion database M_l is created for every possible motion segment length T_l, which is determined by T. Each database is composed of n_l motion segments with a length of T_l. We query the motion feature of the subsequence of Y^{gt} to the database $M_l = \{\mathrm{m}_i\}_{i=1}^{n_l}$ where m_i is the motion feature of the i-th motion segment. To encourage input variety, we randomly sample k from 5 to 10 and select the k-th nearest neighbour with features that resemble but are not identical to the ground truth. The motion segment that corresponds to m_k is used as B or C.

Finally, we process motions A, B, and C to obtain input motion clip X. The root trajectory is copied from Y^{gt} to B and C to match the root trajectory of the input with that of the ground truth. Through this process, the network preserves the given trajectory during training because it is one of the important requirements of motion stitching. We align motion B to motion A by making the root transformation of the first frame in B equal to the root transformation of the last frame in A. The process is repeated to align C to B′: the motion B that has been aligned to A. Then, we simply interpolate the upper body of Y^{gt} to the aligned sequence. We define this sequence as input motion X for our network. As a result, we have a set of input motions $\{\mathrm{X}_i\}_{i=1}^{n_x}$ and ground truth motions $\{\mathrm{Y}_i^{gt}\}_{i=1}^{n_x}$ where n_x is the number of generated data pairs.

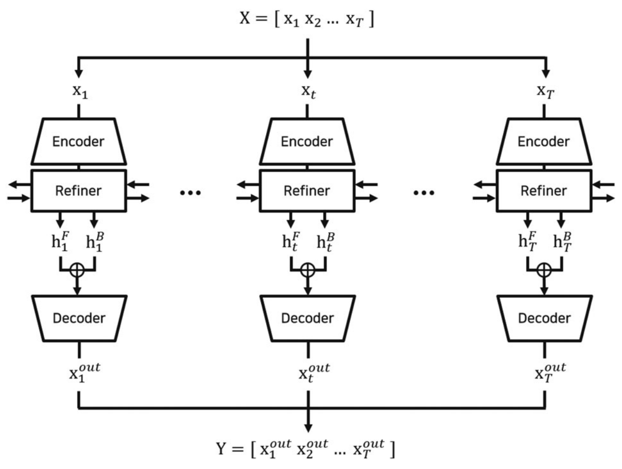


Figure 5: *Recurrent motion refiner.*

5. Recurrent Motion Refiner

As shown in Figure 5, the RMR consists of an encoder, refiner, and decoder [FLFM15]. The encoder E and decoder D each consist of two fully connected layers, and the refiner R includes two BGRU layers. Each layer in BGRU has a forward layer R^F and backward layer R^B. We exploit the most out of the given sequence by going through the input back and forth and providing valid context information at all time-steps. We use the hidden size of $hs = 256$ for every hidden layer and rectified linear unit (ReLU) activation for all linear layers except for the last one in the decoder, which does not use activation.

Given a sequence of poses $\mathrm{X} = [\mathrm{x}_1 \dots \mathrm{x}_T] \in \mathbb{R}^{d \times T}$ as input, the RMR outputs $\mathrm{Y} = [\mathrm{x}_1^{out} \dots \mathrm{x}_T^{out}] \in \mathbb{R}^{d \times T}$, where each output pose x_t^{out} is computed through the following process. All pose vectors x_t are converted to latent vectors with the dimension of hs by E. This series of vectors is input to R as an input sequence. For each time-step, the forward hidden state $\mathrm{h}_t^F \in \mathbb{R}^{hs}$ and backward hidden state $\mathrm{h}_t^B \in \mathbb{R}^{hs}$ of the last layer of R are concatenated into a vector $\mathrm{h}_t \in \mathbb{R}^{2hs}$. Then, D converts h_t back to original representation $\mathrm{x}_t^{out} \in \mathbb{R}^d$. This process is illustrated in Figure 5.

Note that all poses x_t in X are normalized using the mean and standard deviation of the generated data from Section 4. To apply the output pose to the character, we first denormalize x_t^{out} and apply the rotation values. Because the forward kinematics accumulates the error to the joints at ends, we further process the pose by locking both feet in the output positions using inverse kinematics.

5.1. Training

Using the generated data from Section 4, our network is trained with the following loss function:

$$L(\mathrm{X}, \mathrm{Y}, \mathrm{Y}^{gt}) = \lambda_{pose} L_{pose} + \lambda_{trj} L_{trj} + \lambda_{foot} L_{foot} + \lambda_{begin} L_{begin},$$

where L_{pose}, L_{trj}, L_{foot}, and L_{begin} are the poss loss, root trajectory loss, foot contact loss, and beginning loss, respectively. The weight for each term is $\lambda_{pose} = 1.0$, $\lambda_{trj} = 1.0$, $\lambda_{foot} = 0.5$, and $\lambda_{begin} = 1.0$.

The poss loss L_{pose} forces our network to produce a valid motion output:

$$L_{pose} = \frac{1}{T}\sum_{t=1}^{T}\left(\left\|\mathrm{r}_t^{out} - \mathrm{r}_t^{gt}\right\|_1 + \left\|\mathrm{p}_t^{out} - \mathrm{p}_t^{gt}\right\|_1 + \left\|\mathrm{q}_t^{out} - \mathrm{q}_t^{gt}\right\|_1 + \left\|\mathrm{v}_t^{out} - \mathrm{v}_t^{gt}\right\|_1\right),$$

where r_t^{out}, p_t^{out}, q_t^{out}, and v_t^{out} are the output root displacement, root-relative joint positions, parent-relative joint rotations, and joint velocity, respectively; r_t^{gt}, p_t^{gt}, q_t^{gt}, and v_t^{gt} are their corresponding ground truth values.

The root trajectory loss makes the output motion follow the ground truth root trajectory and alleviates the error accumulation in predicting the root displacement:

$$L_{trj} = \frac{1}{T}\sum_{t=1}^{T}\left(\|\mathrm{g}_t^{out} - \mathrm{g}_t^{gt}\|_1\right),$$

where $\mathrm{g}_t^{out} \in \mathbb{R}^3$ and $\mathrm{g}_t^{gt} \in \mathbb{R}^3$ represent the output and ground truth root transformation, respectively. These vectors consist of the 2D position on the xz-plane and the rotation around the y-axis, which are computed by accumulating the predicted root displacement to the initial identity matrix.

The foot contact loss guides the network to predict the correct contact labels which will be used in the post-processing:

$$L_{foot} = \frac{1}{T}\sum_{t=1}^{T}\left(\|\phi(\mathrm{c}_t^{out}) - \mathrm{c}_t^{gt}\|_1\right),$$

where ϕ indicates the sigmoid non-linearity.

Lastly, the beginning loss is added to minimize the artefacts found during runtime. This loss keeps the first few frames of the output motion close to the input motion at the same time-step, minimizing the discontinuity that may occur when the output is updated to the original sequence:

$$L_{begin} = \frac{1}{T_{begin}}\sum_{t=1}^{T_{begin}} sig\left(\frac{t}{T_{begin}}\right)\left(\left\|\mathrm{r}_t^{out} - \mathrm{r}_t\right\|_1 + \left\|\mathrm{p}_t^{out} - \mathrm{p}_t\right\|_1 + \left\|\mathrm{q}_t^{out} - \mathrm{q}_t\right\|_1 + \left\|\mathrm{v}_t^{out} - \mathrm{v}_t\right\|_1\right),$$

where $sig(x) = -1/(1 + exp(5 - 10x)) + 1$ is the transformed sigmoid function and T_{begin} is the number of frames used to compute the loss. We impose the largest loss on the first frame and gradually decrease the value, so that the network attempts to preserve the first few frames while creating the best answer for the remaining frames. We empirically found that using $T_{begin} = 5$ frames for this loss led to the best results. A further experiment on L_{begin} is introduced in Section 7.4.

6. Runtime

After training, our method can stitch multiple locomotion sequences of various lengths into a single smooth motion. First, the input motions are arranged and aligned in the desired order. For best results, each input motion should be longer than 10 frames, which is the minimum length of motion segment created during data generation.

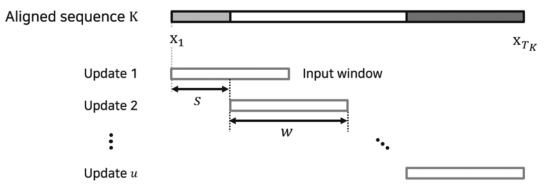


Figure 6: *Process of stitching multiple motions in runtime.*

Suppose the aligned input motion sequence is $\mathrm{K} = [\mathrm{x}_1\mathrm{x}_2 \ldots \mathrm{x}_{T_K}]$, where x_t is a pose vector at time-step t, and T_K is the total length of K. Our method updates the motion within the window of w frames while repeatedly shifting the window along the K by s frames per shift, updating the motion in total of u times, as depicted in Figure 6. Note that next motion should be known in advance because we set w to include the boundary between adjacent motions. We also set s to be smaller than w to overlap the windows. After the update, the motion is cleaned up by blending the small pose differences between the previous pose of the input window and the first pose of the output window [Bol16]. For our experiments, we set the blending weight to decrease to zero over 10 frames.

7. Experiments

In this section, we explain how we implemented our method in detail. Then, we present various locomotion stitching results and the result from an ablation study that shows the effects of the root trajectory loss and the beginning loss. The results obtained with different types of RNN are also compared in the following experiment. Finally, we compare our method with IB[Bol16] and the motion inbetweening method (ERD-QV) [HYNP20]. For the runtime experiments of the RNN comparison and ablation study on the beginning loss, we used $w = 60$ and $s = 30$. For all other experiments, we used $w = 180$ and $s = 90$. Please refer to our supplementary video for animation results.

7.1. Implementation details

Our data pairs were created using the LaFAN1 dataset[HYNP20] with 60 frames per second. The generated dataset consists of 3435 training data and 381 validation data, each sample being $T = 60$ frames long. The test data for all experiments were created with motion clips in the LaFAN1 dataset using combinations unseen during training. The model was trained for 5000 epochs with a batch size of 32, which required about four hours for training. An Adam optimizer was used with a learning rate of 0.0001. For the refiner, a dropout of 0.5 was used in the second layer. All training and experiments were conducted with an AMD Ryzen 7 3700X CPU, 32GB memory, and Nvidia GeForce RTX 3080 GPU.

7.2. Results on locomotion stitching

Here, we describe in detail how locomotion sequences are refined by RMR using three different experiments. In the first experiment, as shown in Figure 7a, two motion clips with a length of 180 frames each are aligned and provided as input to RMR. Between the

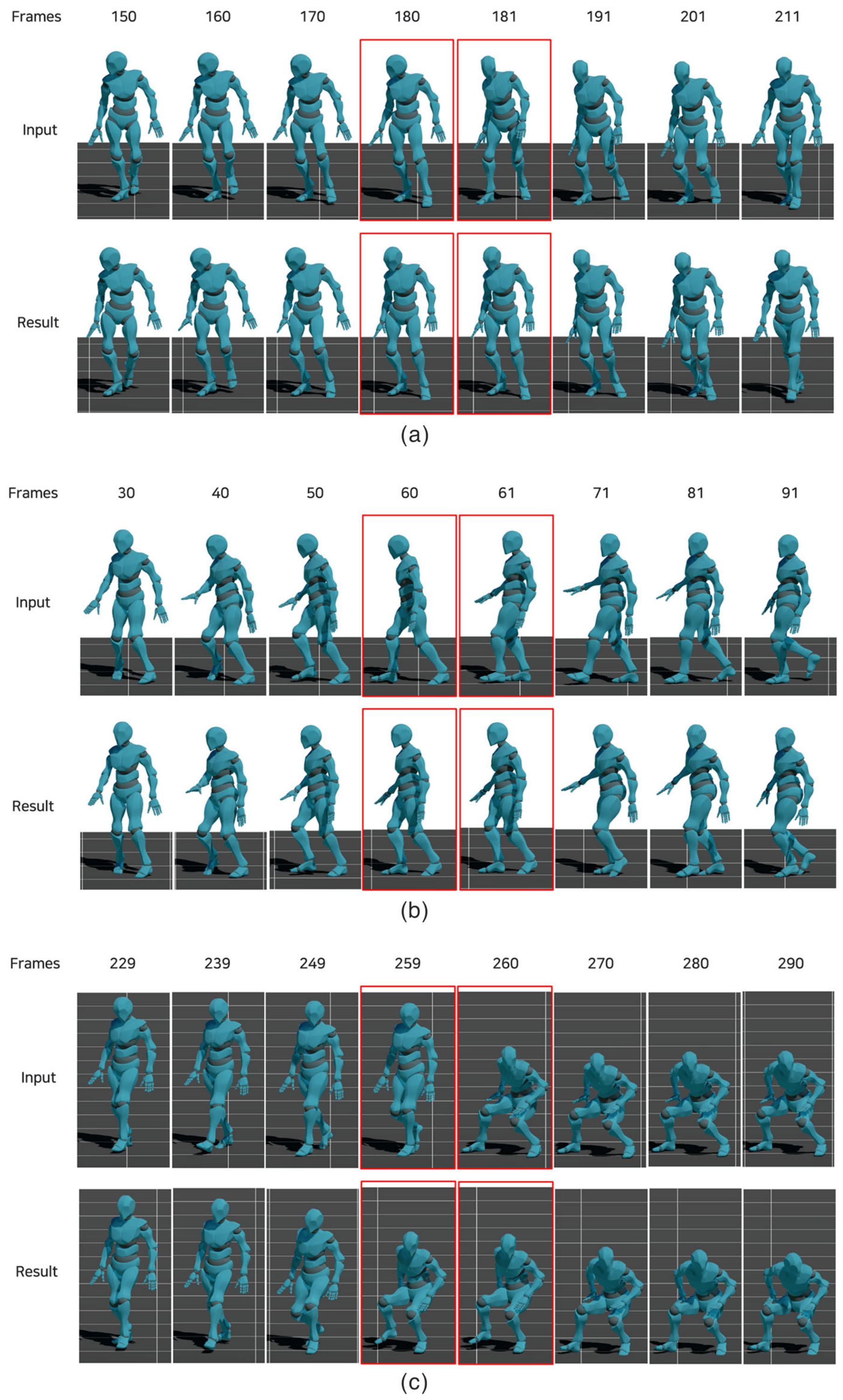


Figure 7: *Locomotion stitching results. A, B, and C are from three different experiments. The frames highlighted in red indicate the boundary where two different locomotion clips meet. Other than the boundary, the poses are shown every 10 frames for clear visualization.*

Table 1: *Comparison to other recurrent neural network cells.*

Architecture	MSE ↓ ($\times 10^{-3}$)	NPSS ↓
LSTM	0.668008	0.165049
GRU	0.688607	0.17621
BLSTM	0.422105	0.149565
BGRU (Ours)	**0.418925**	**0.14885**

The results using long short-term memory (LSTM), gated recurrent unit (GRU), bidirectional long short-term memory (BLSTM), and bidirectional gated recurrent unit (BGRU) are compared.

boundary frames 180 and 181, the input motion has abrupt changes in the feet position and head rotation. In contrast, the refined motion holds the feet of the character and gradually matches the locomotion phase of the input, as shown in frame 211. We also observed that RMR eliminates the discontinuity in head rotation and makes it smoothly converge to the input motion as the frames progress.

In the second experiment, the input motion is composed of five locomotion clips with the length of 60-frames each. Figure 7b shows several frames from the second experiment. As shown in the input at frames 60 and 61, there are sudden switches in the feet position within a frame. The refined pose at frame 60 makes a smaller step compared to the input, preparing for a smooth transition to the next frames. Then, the character takes another small step in frame 81 to catch up on the walking phase of the input motion. This shows that our method can change the timing and size of footsteps as necessary.

In the third experiment, we tested three motion clips of different lengths: 106, 154, and 179. Figure 7c visualizes frames around the second boundary. The character in the input pose at frames 259 and 260 abruptly lowers its body, with the head being rotated and feet positions changed. In the resulting motion, the body of the character is gradually lowered over several frames, as shown from frame 249 to frame 270. The character also starts to turn its head before the boundary and adjusts its feet positions for a continuous transition. This experiment proves that our method can successfully stitch multiple locomotion clips with different lengths. For more various results, please refer to our supplementary video.

7.3. Comparison to other RNNs

We validated our choice of BGRU for the refiner by comparing the mean squared error (MSE) and NPSS [GMK*19] values acquired with other types of RNN. Only the architecture of the refiner is replaced for each experiment and the scores are computed on the validation set. As shown in Table 1, the network using BGRU performs the best. There is a significant difference between using only the forward direction and both the forward and backward directions. This difference in performance can also be observed visually when the models slide through the test samples longer than the training data. As can be seen in the supplementary video (01:47–02:08), the results produced using the gated recurrent unit (GRU) exhibit noticeable popping artefacts at the start of each window due to the lack of context information in the initial hidden vector. Furthermore, as shown in Figure 8, the root trajectory of the output motion from GRU frequently deviates from the input. In contrast, our method

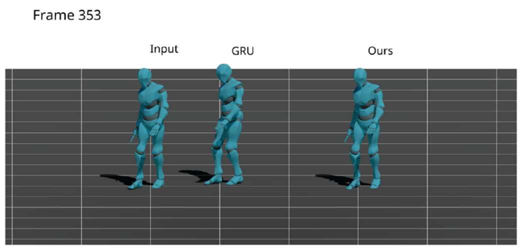
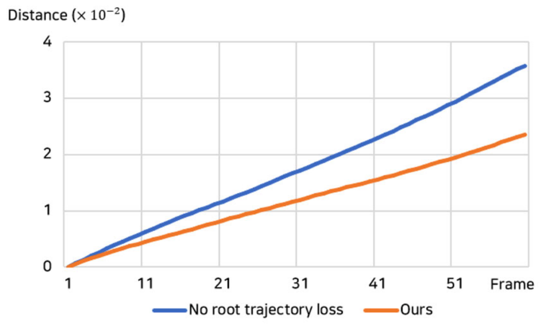
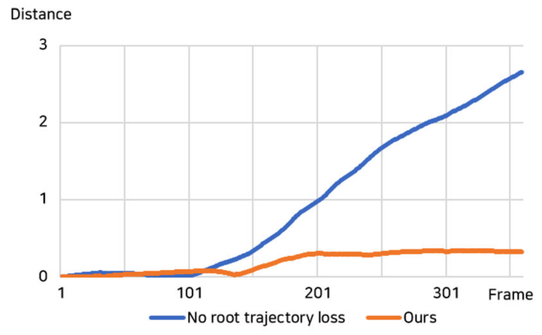


Figure 8: *Comparison to GRU-based network.*



(a) The distance is the averaged value computed over the validation set.



(b) The distance is computed on a test sample that consists of 360 frames in total.

Figure 9: *Effect of the root trajectory loss L_{trj} on the result motion. The x-axis represents the frame and the y-axis represents the distance between the input and output root positions.*

using BGRU produces much less artefacts and better preserves the input trajectory than the method using GRU by utilizing backward information.

7.4. Ablation study

7.4.1. *Root trajectory loss*

An ablation study was conducted on the root trajectory loss to show its effect. We trained two separate models with and without L_{trj}, and measured the root position difference between the input and output on the validation set. As presented in Figure 9a, the model trained with L_{trj} incurs smaller distance errors than the model without L_{trj}. By constraining the network to produce the

Table 2: *Comparison of MSE resulting from the use of different T_{begin}.*

T_{begin}	with GT ($\times 10^{-3}$)	with input ($\times 10^{-3}$)
3	0.434587	3.20278
5 (Ours)	**0.418925**	**3.17028**
10	0.4275	3.17795

Table 3: *Comparison of the foot sliding distances produced by inertialization blending (IB), motion in-betweening (ERD-QV), and our method (Ours)*

Method	Foot sliding distance
IB	0.0092
ERD-QV	0.0094
Ours	**0.0074**

correct global root position, this loss successfully mitigates the error that may be accumulated when continuously inferring the root displacement.

The impact of L_{trj} is apparent when refining the poses by the window during runtime. We tested the two models with unseen sequences longer than the training samples and measured the distance between the input and output root positions. The measured distances using one of the test samples are plotted in Figure 9b. In the figure, the distance produced without using L_{trj} increases dramatically, while that produced using L_{trj} remains relatively low. This suggests that L_{trj} effectively preserves the input trajectory even when refining sequences longer than the training data.

7.4.2. *Beginning loss*

We evaluated the impact of the beginning loss on the result during runtime. We trained the network without L_{begin} and compared the test results with those produced by the model that uses all losses. When tested on an input sequence longer than the training samples, the model without L_{begin} produced popping artefacts in the beginning of the update window, as shown in the accompanying video (02:29–02:48). To determine the T_{begin} that best preserves the first few input poses at the same time-step, we trained three different models that use different T_{begin}. We did not use more than 10 frames because the discontinuity in the input sequence could affect the training process. MSE was computed between the output and ground truth, and the output and input in all three cases. As shown in Table 2, ours using $T_{begin} = 5$ achieved the lowest error, producing poses close to the ground truth and minimizing the possible discontinuity that may appear during each window update.

7.5. Comparison to motion blending and in-betweening

To examine our approach with other methods, we compared our method with IB and ERD-QV. Given the task of connecting two different motions, we measured the foot sliding distance on the validation set. Among the validation data made of two motion segments, we selected 51 data samples whose first segment is longer than 10 frames to secure the input frames required by ERD-QV.

To compare the performance of each method in the motion stitching task, we applied IB and ERD-QV as follows. For IB, we interpolated the pose difference between the boundary frames and added it to the first 30 frames of the subsequent motion, decreasing the blending weight to zero. For ERD-QV, we trained it to fill in-between poses of the ground truth when given the first 10 frames as context and the last frame as the target. Then, for each validation input, we removed a window of poses up to 30 frames centred at the frames where the two motion segments meet. We let ERD-QV fill these deleted frames and connect the motion segments.

The resulting average foot sliding distances are compared in Table 3. We measured the foot positional displacement when the distance between the floor and each foot was less than 2 cm. No post processing was applied for fair comparison. As shown in the table, our method outperformed IB and ERD-QV by producing the smallest amount of foot sliding.

In addition, we qualitatively compared IB and ERD-QV to our method using the test samples. First, we compared IB to our method by utilizing five input motions of 60 frames each. Figure 10 shows several frames close to the second boundary. As can be seen from frames 120 to 141, IB failed to bend the character's knee and let the foot slide on the ground, while ours produced a consistent walking motion. This proves that employing RNN trained on a large amount of realistic motion data can connect different motions more naturally compared to the interpolation-based method.

Next, we compared ERD-QV to our method by utilizing three input motions of 120 frames each. Figure 11 presents the resulting poses close to the first boundary. As shown in frames 131 and 141, ERD-QV suffers from a floating character problem, which makes the output motion unnatural and different from the input. In contrast, our model refines the given sequences into a continuous motion while preserving the input poses. Through this experiment, we verified that our approach, which receives a sequence with no missing frames and exploits backward information, can perform better than ERD-QV when connecting different locomotions. The animated results of these experiments are provided in the supplementary video (02:49–03:24).

8. Discussion

During the data generation process, retrieving a motion segment using KNN search and forcing it to be same as the ground truth motion may lead to many-to-one mapping. Because it is important to match the phase of the feet when stitching locomotions, we retrieved motion segments with similar but not identical phases through KNN search as explained in Section 4. Our model was trained to mainly edit the latter part of the input to be identical to the ground truth, because we assumed that this approach would be suitable for real-time controllers such as motion matching [BC15]. In this process, motions with different phases can be edited and mapped to the same output phase. Although this might restrain the diversity in the output, our results show that our method succeeds in automatically refining different footsteps and naturally synchronizing the upper and

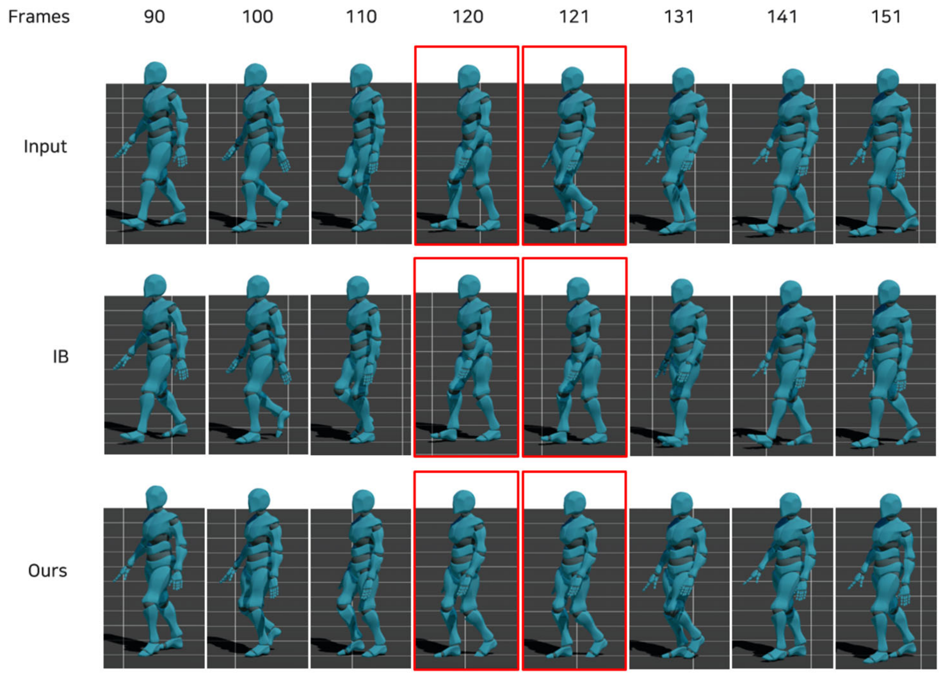


Figure 10: *Comparison of motion stitching test results obtained with inertialization blending (IB) and our method (Ours). Five input motion clips of 60 frames each were given. The poses in red highlights indicate the boundary frames. For IB, results were obtained by decreasing the blending weight to zero over 60 frames.*

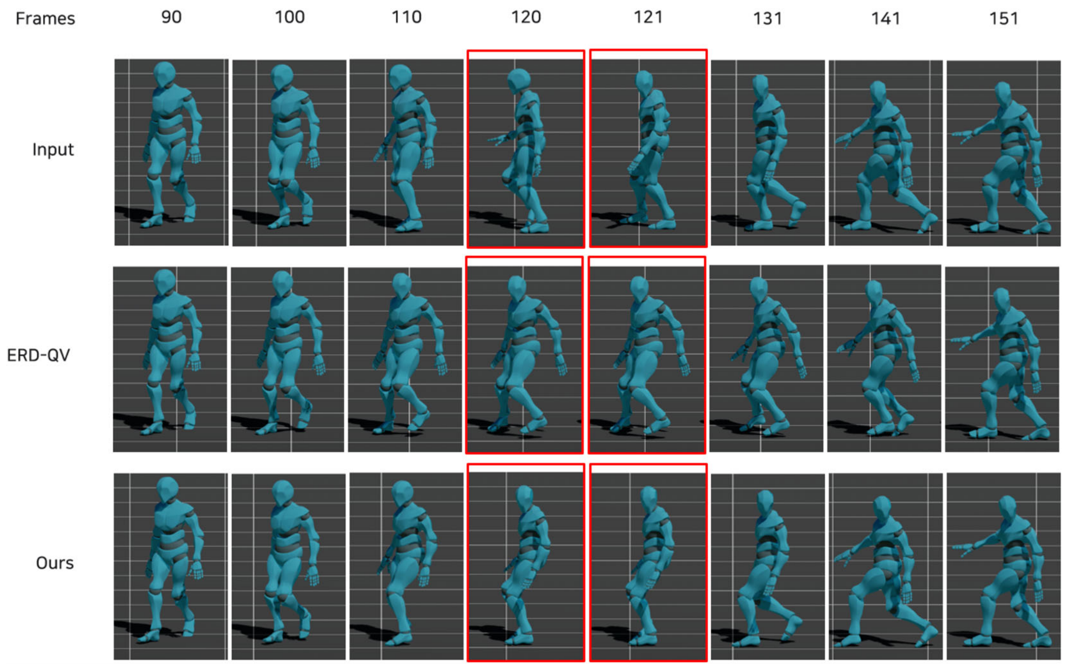


Figure 11: *Comparison of motion stitching test results obtained with motion in-betweening (ERD-QV) and our method (Ours). Three input motion clips of 120 frames each were given. The poses in red highlights indicate the boundary frames. For ERD-QV, results were obtained by removing 60 frames centred at the boundary and filling them with the generated poses.*

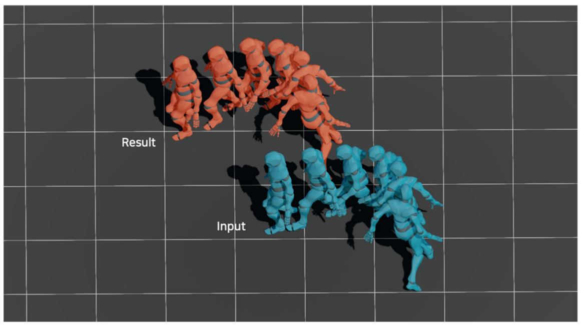


Figure 12: *Comparison of the input and result trajectories as a failure case. The root trajectory is not preserved well. The poses are visualized every 20 frames.*

lower body movement, which is difficult to achieve through previous methods.

While our method is able to cope with arbitrary combinations of locomotion clips, its quality of the output is data-dependent. Our model can be applied to unseen input motions from other dataset [SZKS19], but if the input motion largely differs from the training samples, the output may be overly smoothed out. Similarly, if the pose differences at the boundary are larger than those in the training samples, our model may produce unnatural transitions. To generalize our method for use in many applications, we plan to expand the dataset to include various categories and styles of motion. In addition, we could take advantage of a more complex network architecture than the current network to learn the generalized pattern of character motion.

In some cases, our model struggles to preserve the exact trajectory of the input motions, as shown in Figure 12. The main reason is that the error accumulates as the input is refined during runtime. To produce results that abide by the input trajectory, we could add another component to our method. One way would be to employ a two-step process in which the root trajectory of the raw output motion is blended to that of the input motion followed by refining the output again to remove foot sliding artefacts.

At runtime, inferring the given sequence by window may introduce discontinuity artefacts due to the pose difference between the first frame of each window and the previous frame that is not updated. However, the difference is small enough that applying simple interpolation in post-processing can easily remove it. Using a wider window during runtime can also reduce this artefact because we empirically found that using a larger window size smooths out poses in a wider range at the cost of losing some sharpness in the motion. In future work, we can include the previous frame in our training scheme and add a smoothing loss term to reduce the sudden pose differences.

9. Conclusion

In this work, we propose a novel neural network-based motion stitching method that connects multiple locomotion clips into a single natural animation. To acquire the data pairs, we construct a motion database based on selected motion features and create input data through KNN search. The generated dataset was used to train the suggested network, RMR, that consists of an encoder, refiner, and decoder. During runtime, we first align multiple input motions in order by matching the root transformation of the last frame in the preceding motion and that of the first frame in the subsequent motion. Then, our trained network refines the poses using a sliding window, until it reaches the end of the aligned sequence. We demonstrated that our method can combine many locomotions with various lengths and inconsistent phases. In the future, we plan to improve the method of balancing the naturalness of the output motion with the input pose preservation. We also hope to generalize our method to stitch a wide range of motions besides locomotion.

Acknowledgements

This research was supported by Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency grant funded by Ministry of Culture, Sports and Tourism in 2023 (Project Name: Development of Universal Fashion Creation Platform Technology for Avatar Personality Expression, Project Number: RS-2023-00228331, Contribution Rate: 100%)

References

[AF02] Arikan O., Forsyth D. A.: Interactive motion generation from examples. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 483–490.

[AKCH21] Aksan E., Kaufmann M., Cao P., Hilliges O.: A spatio-temporal transformer for 3d human motion prediction. In *2021 International Conference on 3D Vision (3DV)* (2021), IEEE, pp. 565–574.

[ASK*12] Akhter I., Simon T., Khan S., Matthews I., Sheikh Y.: Bilinear spatiotemporal basis models. *ACM Transactions on Graphics (TOG) 31*, 2 (2012), 1–12.

[BBKK17] Butepage J., Black M. J., Kragic D., Kjellstrom H.: Deep representation learning for human motion prediction and classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6158–6166.

[BC15] Büttner M., Clavet S.: Motion matching – The road to next gen animation. *Proceedings of Nucl.ai* (2015).

[Bol16] Bollo D.: Inertialization: High-performance animation transitions in 'gears of war'. *Proceedings of GDC 2018* (2016).

[Bol17] Bollo D.: High performance animation in gears of war 4. In *ACM SIGGRAPH 2017 Talks* (New York, NY, USA, 2017), SIGGRAPH '17, Association for Computing Machinery, Article 22, 2 pages URL: https://doi.org/10.1145/3084363.3085069.

[CAW*19] Chiu H.-k., Adeli E., Wang B., Huang D.-A., Niebles J. C.: Action-agnostic human pose forecasting. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)* (2019), IEEE, pp. 1423–1432.

[CS21] Cui Q., Sun H.: Towards accurate 3d human motion prediction from incomplete observations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 4801–4810.

[DCLT18] Devlin J., Chang M.-W., Lee K., Toutanova K.: Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[DSZ*21] Duan Y., Shi T., Zou Z., Lin Y., Qian Z., Zhang B., Yuan Y.: Single-shot motion completion with transformer. *arXiv preprint arXiv:2103.00776* (2021).

[FJX*14] Feng Y., Ji M., Xiao J., Yang X., Zhang J. J., Zhuang Y., Li X.: Mining spatial-temporal patterns and structural sparsity for human motion data denoising. *IEEE Transactions on Cybernetics 45*, 12 (2014), 2693–2706.

[FLFM15] Fragkiadaki K., Levine S., Felsen P., Malik J.: Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision* (2015), pp. 4346–4354.

[FXS12] Feng A. W., Xu Y., Shapiro A.: An example-based motion synthesis technique for locomotion and object manipulation. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (2012), pp. 95–102.

[GMK*19] Gopalakrishnan A., Mali A., Kifer D., Giles L., Ororbia A. G.: A neural temporal model for human motion prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 12116–12125.

[GR96] Guo S., Robergé J.: A high-level control mechanism for human locomotion based on parametric frame space interpolation. In *Computer Animation and Simulation'96*. Springer, 1996, pp. 95–107.

[GRvSD21] Geleijn R., Radziszewski A., van Straaten J. B., Debarba H. G.: Lightweight quaternion transition generation with neural networks. In *2021 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)* (2021), IEEE, pp. 579–580.

[GSAH17] Ghosh P., Song J., Aksan E., Hilliges O.: Learning human motion models for long-term predictions. In *2017 International Conference on 3D Vision (3DV)* (2017), IEEE, pp. 458–466.

[GWLM18] Gui L.-Y., Wang Y.-X., Liang X., Moura J. M.: Adversarial geometry-aware human motion prediction. In *Proceedings of the European Conference on Computer Vision (ECCV)* (2018), pp. 786–803.

[HG07] Heck R., Gleicher M.: Parametric motion graphs. In *Proceedings of the 2007 Symposium on Interactive 3D Graphics and Games* (2007), pp. 129–136.

[HGMN19] Hernandez A., Gall J., Moreno-Noguer F.: Human motion prediction via spatio-temporal inpainting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 7134–7143.

[HHS*17] Habibie I., Holden D., Schwarz J., Yearsley J., Komura T.: A recurrent variational autoencoder for human motion synthesis. In *28th British Machine Vision Conference* (2017).

[HK10] Huang Y., Kallmann M.: Motion parameterization with inverse blending. In *International Conference on Motion in Games* (2010), Springer, pp. 242–253.

[HP18] Harvey F. G., Pal C.: Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*. 2018, pp. 1–4.

[HSKJ15] Holden D., Saito J., Komura T., Joyce T.: Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*. 2015, pp. 1–4.

[HYNP20] Harvey F. G., Yurick M., Nowrouzezahrai D., Pal C.: Robust motion in-betweening. *ACM Transactions on Graphics (TOG) 39*, 4 (2020), 60–1.

[JZSS16] Jain A., Zamir A. R., Savarese S., Saxena A.: Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 5308–5317.

[KAS*20] Kaufmann M., Aksan E., Song J., Pece F., Ziegler R., Hilliges O.: Convolutional autoencoders for human motion infilling. In *2020 International Conference on 3D Vision (3DV)* (2020), IEEE, pp. 918–927.

[KBS*22] Kim J., Byun T., Shin S., Won J., Choi S.: Conditional motion in-betweening. *arXiv preprint arXiv:2202.04307* (2022).

[KG03] Kovar L., Gleicher M.: Flexible automatic motion blending with registration curves. In *Symposium on Computer Animation* (2003), vol. *2*, San Diego, CA, USA.

[KG04] Kovar L., Gleicher M.: Automated extraction and parameterization of motions in large data sets. *ACM Transactions on Graphics (ToG) 23*, 3 (2004), 559–568.

[KGP02] Kovar L., Gleicher M., Pighin F.: Motion graphs. *ACM Transactions on Graphics (TOG) 21*, 3 (2002), 473–482.

[LC10] Lou H., Chai J.: Example-based human motion denoising. *IEEE Transactions on Visualization and Computer Graphics 16*, 5 (2010), 870–879.

[LVC*21] Li J., Villegas R., Ceylan D., Yang J., Kuang Z., Li H., Zhao Y.: Task-generic hierarchical human motion prior using vaes. In *2021 International Conference on 3D Vision (3DV)* (2021), IEEE, pp. 771–781.

[LZX*17] Li Z., Zhou Y., Xiao S., He C., Huang Z., Li H.: Auto-conditioned recurrent networks for extended complex human motion synthesis. *arXiv preprint arXiv:1707.05363* (2017).

[LZZ*19] Li S., Zhou Y., Zhu H., Xie W., Zhao Y., Liu X.: Bidirectional recurrent autoencoder for 3d skeleton motion data refinement. *Computers & Graphics 81* (2019), 92–103.

[LZZL20] Li S.-J., Zhu H.-S., Zheng L.-P., Li L.: A perceptual-based noise-agnostic 3d skeleton motion data refinement network. *IEEE Access 8* (2020), 52927–52940.

[MBR17] Martinez J., Black M. J., Romero J.: On human motion prediction using recurrent neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 2891–2900.

[MK05] Mukai T., Kuriyama S.: Geostatistical motion interpolation. In *ACM SIGGRAPH 2005 Papers*. 2005, pp. 1062–1070.

[MLCC17] Mall U., Lal G. R., Chaudhuri S., Chaudhuri P.: A deep recurrent framework for cleaning motion capture data. *arXiv preprint arXiv:1712.03380* (2017).

[OVH*22] Oreshkin B. N., Valkanas A., Harvey F. G., Ménard L.-S., Bocquelet F., Coates M. J.: Motion inbetweening via deep δ-interpolator. *arXiv preprint arXiv:2201.06701* (2022).

[PGA18] Pavllo D., Grangier D., Auli M.: Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* (2018).

[PSKS04] Park S. I., Shin H. J., Kim T. H., Shin S. Y.: On-line motion blending for real-time locomotion generation. *Computer Animation and Virtual Worlds 15*, 3-4 (2004), 125–138.

[PSS02] Park S. I., Shin H. J., Shin S. Y.: On-line locomotion generation based on motion blending. In *Proceedings of the 2002 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (2002), pp. 105–111.

[RCB98] Rose C., Cohen M. F., Bodenheimer B.: Verbs and adverbs: Multidimensional motion interpolation. *IEEE Computer Graphics and Applications 18*, 5 (1998), 32–40.

[RISC01] Rose III C. F., Sloan P.-P. J., Cohen M. F.: Artist-directed inverse-kinematics using radial basis function interpolation. *Computer Graphics Forum 20*, (2001), 239–250.

[SZKS19] Starke S., Zhang H., Komura T., Saito J.: Neural state machine for character-scene interactions. *ACM Transactions on Graphics 38*, 6 (2019), 209–1.

[SZQ*21] Shu X., Zhang L., Qi G.-J., Liu W., Tang J.: Spatiotemporal co-attention recurrent neural networks for human-skeleton motion prediction. *IEEE Transactions on Pattern Analysis and Machine Intelligence 44*, 6 (2021), 3300–3315.

[TS06] Tangkuampien T., Suter D.: Human motion de-noising via greedy kernel principal component analysis filtering. In *18th International Conference on Pattern Recognition (ICPR'06)* (2006), vol. *3*, IEEE, pp. 457–460.

[TWH*22] Tang X., Wang H., Hu B., Gong X., Yi R., Kou Q., Jin X.: Real-time controllable motion transition for characters. *arXiv preprint arXiv:2205.02540* (2022).

[VSP*17] Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A. N., Kaiser Ł., Polosukhin I.: Attention is all you need. *Advances in Neural Information Processing Systems 30* (2017).

[WCX19] Wang Z., Chai J., Xia S.: Combining recurrent neural networks and adversarial training for human motion synthesis and control. *IEEE Transactions on Visualization and Computer Graphics 27*, 1 (2019), 14–28.

[WH97] Wiley D. J., Hahn J. K.: Interpolation synthesis for articulated figure motion. In *Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)* (USA, 1997), VRAIS '97, IEEE Computer Society, p. 156.

[WHSZ19] Wang H., Ho E. S., Shum H. P., Zhu Z.: Spatio-temporal manifold learning for human motions via long-horizon modeling. *IEEE Transactions on Visualization and Computer Graphics 27*, 1 (2019), 216–227.

[XFJ*15] Xiao J., Feng Y., Ji M., Yang X., Zhang J. J., Zhuang Y.: Sparse motion bases selection for human motion denoising. *Signal Processing 110* (2015), 108–122.

[XSZF16] Xia G., Sun H., Zhang G., Feng L.: Human motion recovery jointly utilizing statistical and kinematic information. *Information Sciences 339* (2016), 189–205.

[ZLB*20] Zhou Y., Lu J., Barnes C., Yang J., Xiang S., et al.: Generative tweening: Long-term inbetweening of 3d human motions. *arXiv preprint arXiv:2005.08891* (2020).

[ZvdP18] Zhang X., van de Panne M.: Data-driven autocompletion for keyframe animation. In *Proceedings of the 11th Annual International Conference on Motion, Interaction, and Games* (2018), pp. 1–11.

Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Supporting Video S2