# A Unified Framework for Real Time Motion Completion

**Yinglin Duan[1], Yue Lin[1], Zhengxia Zou[2*], Yi Yuan[3], Zhehui Qian[3], Bohan Zhang[3]**

[1] NetEase Games AI Lab, Guangzhou, China,
[2] Beihang University,
[3] Netease, Inc., Hangzhou, China
{duanyinglin, gzlinyue, yuanyi, qianzhehui, hzzhangbohan}@corp.netease.com, zhengxiazou@buaa.edu.cn

## Abstract

Motion completion, as a challenging and fundamental problem, is of great significance in film and game applications. For different motion completion application scenarios (in-betweening, in-filling, and blending), most previous methods deal with the completion problems with case-by-case methodology designs. In this work, we propose a simple but effective method to solve multiple motion completion problems under a unified framework and achieve a new state-of-the-art accuracy on LaFAN1 (+17% better than the previous SoTA) under multiple evaluation settings. Inspired by the recent great success of self-attention-based transformer models, we consider the completion as a sequence-to-sequence prediction problem. Our method consists of three modules - a standard transformer encoder with self-attention that learns long-range dependencies of input motions, a trainable mixture embedding module that models temporal information and encodes different key-frame combinations in a unified form, and a new motion perceptual loss for better capturing high-frequency movements. Our method can predict multiple missing frames within a single forward propagation in real-time without post-processing. We also introduce a novel large-scale dance movement dataset for exploring the scaling capability of our method and its effectiveness in complex motion applications.

## Introduction

Motion completion is an important and challenging problem that provides fundamental technical support for animation authoring of 3D characters and has been recently successfully applied in film production and video games (Thomas 2009; Ciccone, Öztireli, and Sumner 2019). In recent years, deep learning methods have greatly promoted the research progress of motion completion. With recent advances in this field, manpower can now be greatly saved, where high-quality motion can be generated from a set of historical or sparse key-frames by learning from large-scale motion capture data sets (Kaufmann et al. 2020; Harvey et al. 2020).

We consider the motion completion in the following three scenarios:

* *In-betweening.* Animators are required to complement the motion between the past given frames and a provided further keyframe (Harvey et al. 2020).
* *In-filling.* As an extension of in-betweening, in-filling poses the characters on specific positions of the timeline (Ciccone, Öztireli, and Sumner 2019) and complements the rest of the frames (Kaufmann et al. 2020).
* *Blending.* Blending focuses on the automatic generation of the transition between a pair of pre-defined motions. Blending is widely used in video games, such as in-game choreography systems.

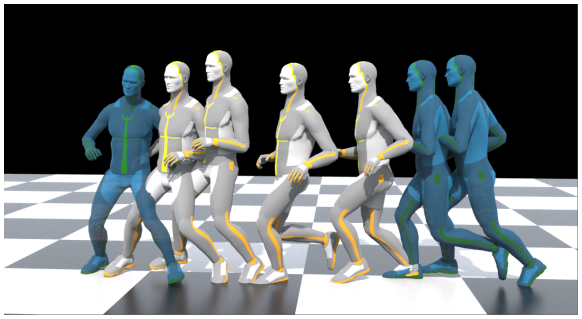

Figure 1: Motion completion results by our method based on input key frames (blue ones). We propose a unified framework that can solve multiple motion completion problems and achieve a new SoTA on a high-quality motion completion dataset – LaFAN1 (Harvey et al. 2020).

However, most previous motion completion methods deal with different completion scenarios (in-betweening, in-filling, and blending) with case-by-case designs. In this paper, we propose a novel completion method that deals with all the above processing scenarios under a unified framework. In our method, we leverage the recent popular deep learning architecture called Transformer (Vaswani et al. 2017), which is built based on the self-attention mechanism and is widely used in neural language processing, computer vision, and reinforcement learning (Tay et al. 2020; Dosovitskiy et al. 2020; Parisotto et al. 2020). We adopt BERT (Devlin et al. 2018), a recent well-known transformer encoder as our backbone, where known frames (key-frames) and unknown frames (need to be complemented) are fed together as input, and thus all the frames can be predicted in a sin-

*Corresponding author.

gle propagation at inference. Our method works in a non-autoregressive manner and can be easily accelerated with parallelization.

Based on the standard transformer architecture, we also introduce a mixture embedding module that further integrates temporal knowledge to the transformer encoder and makes the proposed framework compatible with the above three different motion completion scenarios. Our embedding module contains two types of learnable embeddings: position embedding and keyframe embedding. *Position embedding* is a widely studied technology in recent transformer literature, where a set of pre-defined sinusoidal signals are usually used to introduce temporal knowledge to the transformer model (Harvey and Pal 2018; Harvey et al. 2020). In our method, we further make this embedding trainable to deal with different motion completion scenarios. In addition to the encoding of input frame orders, the *Keyframe embedding* is introduced to annotate the input frames so that the model knows which parts of the input frames are keyframes (already known) and which parts need to be predicted (unknown). Since the keyframe embedding may have different forms, our method can be easily applied to different completion scenarios regardless of how the input keyframes are arranged. Besides, different from the previous work (Harvey et al. 2020) that predicts contact information for post-processing, we introduce a new motion perceptual loss, where we directly regularize the velocity of generated results on the ground contact points and further adopt wavelet transformation to dynamically capture the structure information of the complex movements. Our method can therefore get rid of the post-processing and can run in real-time.

Our contributions are summarized as follows:

1. We propose a transformer-based method to solve the motion completion of different application scenarios under a unified framework. A *mixture embedding* module and a *motion perceptual loss* are introduced to unify different completion tasks and introduce ground contact constraints with high-frequency representation.

2. We achieve a new benchmark accuracy on the in-betweening task under multiple evaluation settings, with +17% better than previous state of the art methods[1]. Results on other completion tasks (in-filling and blending) are also reported as baselines for fostering the research community.

3. Our method can work in a parallel prediction manner with high efficiency. On a single CPU desktop (I7-8700K @ 3.70GHz), our method can run in real time (40 motion sequences per second, each with 50 frames long), 4x faster than previous methods.

4. A new high-quality and large-scale dance movement dataset is built for advanced motion completion tasks.

# Related Works

## Motion Completion

Motion completion is a research hotspot in the field of computer graphics and multimedia. Some early researches of motion completion can be traced back to the late 1980s (Witkin and Kass 1988; Ngo and Marks 1993). Motion completion can be considered as a conditional motion sequence generation problem. Different from those unconditional motion synthesis tasks that focus on the generation of unconstrained motion sequences by directly sampling from their posterior distribution (Sidenbladh, Black, and Sigal 2002; Wang, Fleet, and Hertzmann 2007; Taylor, Hinton, and Roweis 2007), motion completion aims at filling the missing frames in a temporal sequence based on a given set of keyframes. Early works of motion completion typically adopt inverse kinematics to generate realistic transitions between keyframes. For example, space-time constraints and searching-based methods were proposed in the 1980s-1990s (Witkin and Kass 1988; Ngo and Marks 1993) to compute optimal physically-realistic motion trajectories. By using such techniques, transitions between different motions can be smoothly generated (Rose et al. 1996). Also, probabilistic models like the maximum a posterior methods (Chai and Hodgins 2007; Min, Chen, and Chai 2009), the Gaussian process (Wang, Fleet, and Hertzmann 2007), Markov models (Lehrmann, Gehler, and Nowozin 2014) were introduced to motion completion tasks and were commonly used after 2000s.

Recently, deep learning methods have greatly promoted the research and the performance of motion completion, where the recurrent neural network is the most commonly used framework for motion completion of the deep learning era (Zhang and van de Panne 2018; Harvey and Pal 2018). For example, Harvey *et al*. introduce a novel framework named Recurrent Transition Networks (RTN) to learn a more complex representation of human motions with the help of LSTM (Harvey and Pal 2018). Besides, generative adversarial learning has been also introduced to the motion completion to make the output motion more realistic and naturalistic (Hernandez, Gall, and Moreno-Noguer 2019). Some non-recurrent motion completion models are also proposed very recently. Kaufmann *et al*. propose an end-to-end trainable convolutional autoencoder to fill in missing frames (Kaufmann et al. 2020). Another recent progress by Harvey *et al*. introduces time-to-arrival embeddings and scheduled target-noise to further enhance the performance of RTN and achieve impressive completion results (Harvey et al. 2020). Different from the previous RNN-based or convolution-based method, we propose a transformer-based model - a more unified solution to this task that can deal with arbitrary forms of missing frames in parallel with only a single-shot prediction.

## Motion Control

Motion control is a typical conditional motion generation task, which is also highly related to motion completion. In motion control, the control signal comes from a pre-defined temporal sequence, e.g. root trajectory, rather than a set of keyframes in motion completion.

Graph-based motion control is the most common type of method in this field before the deep learning era (Lee et al. 2002; Safonova and Hodgins 2007; Kovar, Gleicher, and Pighin 2008; Arikan and Forsyth 2002; Beaudoin et al.

[1]Our project is available at: https://github.com/SilvanDuan/MotionCompletion

2008). For example, Arikan *et al*. formulate the motion generation as a randomized search of the motion graph, which allows complex motions editing (Arikan and Forsyth 2002). Beaudoin *et al*. propose a string-based motif-finding algorithm named Motion-Motif Graphs, which further considers the motif length and the number of motions in a motif (Beaudoin et al. 2008). Besides, there are many statistical methods proposed to avoid searching from predefined motion templates (Grochow et al. 2004; Wang, Fleet, and Hertzmann 2007; Min, Chen, and Chai 2009; Min and Chai 2012). Chai *et al*. propose a statistical dynamic model to generate motions with motion prior (e.g. user-defined trajectory) and formulate the constrained motion synthesis as a maximum a posterior problem (Chai and Hodgins 2007). Ye *et al*. introduce a nonlinear probabilistic dynamic model that can handle perturbations (Ye and Liu 2010). Levine *et al*. propose a probabilistic motion model that learns a low-dimensional space from example motions and generates character animation based on user-specified tasks (Levine et al. 2012).

Recently, some popular deep learning architectures like convolutional autoencoder and recurrent neural networks are widely used in the motion control problems (Holden et al. 2015; Holden, Saito, and Komura 2016; Holden, Komura, and Saito 2017; Lee, Lee, and Lee 2018; Aberman et al. 2020b,a; Gomes et al. 2021). Adversarial training has also played an important role in this problem and can help generate more realistic motion sequences (Barsoum, Kender, and Liu 2018; Gui et al. 2018; Harvey et al. 2020). Besides, some recent approaches also leverage reinforcement learning to further incorporate physical rules to improve the quality of the generation results (Coros, Beaudoin, and Van de Panne 2009; Yin, Loken, and Van de Panne 2007; Baram, Anschel, and Mannor 2016; Peng et al. 2017, 2018; Bergamin et al. 2019). As far as we know, the transformer-based prediction architecture proposed in this paper is still rarely studied in both motion completion and motion control.

## Methods

In this work, we consider motion completion as a sequence-to-sequence prediction problem. We choose BERT, an off-the-shelf transformer architecture (Devlin et al. 2018) as our network backbone with minimum modifications. In this way, the subsequent variations can be easily introduced without impediment. The unknown motion frames can be generated in a single forward propagation conditioned by those input keyframes at the inference stage.

### Motion Completion Transformer

Fig.2 shows an overview of our method. Our network consists of 1) a mixture embedding module that converts the motion to a set of sequential tokens, and 2) a standard transformer encoder used to process sequential features. Our method supports multiple input coded format, e.g. [local positions & rotations] or [global positions & rotations] or [positions only]. Without loss of generality, we assume that the input contains both positions ($x$, $y$, $z$) and rotations ($q_0$, $q_1$, $q_2$, $q_3$) variables (no matter local or global), and therefore, a single pose can be represented as two matrices - a position coordinate matrix $\boldsymbol{P} \in \mathbb{R}^{J\times 3}$ and a quaternion matrix

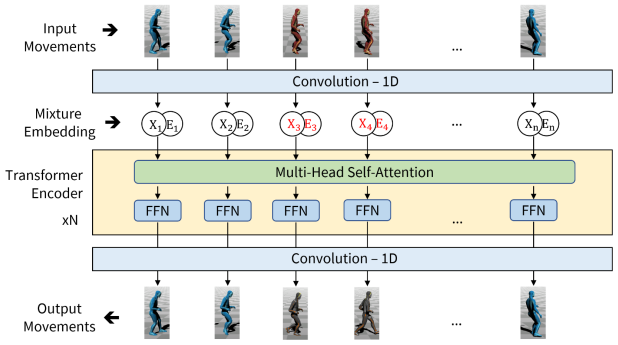


Figure 2: An overview of our method. Our method consists of a standard transformer encoder, a mixture embedding layer, and input/output convolutional heads. In motion completion, the unknown input frames are first generated by using linear interpolation (shown in red characters) before being fed to the model. Our method takes in a whole masked sequence and completes the prediction within only a single forward propagation.

$\boldsymbol{Q} \in \mathbb{R}^{J\times 4}$, where $J$ represents the joint number of the input pose.

For each input pose, we firstly flatten $\boldsymbol{P}$ and $\boldsymbol{Q}$ into 1-D vectors $\boldsymbol{p}$ and $\boldsymbol{q}$, and then concatenate the two vectors together into a long vector:

$$\boldsymbol{x} = [\boldsymbol{p}, \boldsymbol{q}] \in \mathbb{R}^{J\times(3+4)}. \tag{1}$$

For those unknown input frames, we use linear interpolation to fill in their missing values along the temporal dimension before feeding them to the transformer model. We then use a 1-D temporal convolution layer to transform those pose vectors to tokens:

$$\boldsymbol{Z} = \text{Conv1D}([\boldsymbol{x}^1; \boldsymbol{x}^2; ...; \boldsymbol{x}^T]), \tag{2}$$

where $T$ is the length of the input sequence, $\boldsymbol{Z} \in \mathbb{R}^{T\times F}$ is the formated temporal feature of the tokens, and $F$ is the output dimension of the Conv1D layer. The convolution is performed in the joint dimension. Note that different from the previous transformer-based models in computer vision (Dosovitskiy et al. 2020) that use a linear projection to generate the embeddings, here we use a convolution layer for better capturing the temporal information, e.g. velocity and acceleration.

Considering that the transformer does not know the order of the keyframes in the input sequence as well as their exact locations, we introduce a mixture embedding $\boldsymbol{E}$ to annotate these frames before feeding their features into the transformer. For each group of input configuration of keyframes, an embedding $\boldsymbol{E}$ is learned as a global variable on the training data and will not change along with the pose feature $\boldsymbol{Z}$. We represent the final annotated features $\boldsymbol{Z}$ as follows:

$$\hat{\boldsymbol{Z}} = [\boldsymbol{z}^1 + \boldsymbol{e}^1; \boldsymbol{z}^2 + \boldsymbol{e}^2; ...; \boldsymbol{z}^T + \boldsymbol{e}^T], \tag{3}$$

where $\boldsymbol{z}^t$ and $\boldsymbol{e}^t$ are the sub-vectors of input feature $\boldsymbol{Z}$ and mixture embedding $\boldsymbol{E}$ at the time $t$.

The BERT transformer we used consists of multiple encoder layers (Devlin et al. 2018). Each encoder layer further consists of a multi-head self-attention layer (MHSA) and a feed-forward network (FFN). A residual connection (He et al. 2016) is applied across the two layers. The forward mapping of the transformer can be written as follows:

$$\begin{aligned}\hat{\boldsymbol{H}}^{l} &= \text{Norm}(\boldsymbol{H}^{l-1} + \text{MHSA}(\boldsymbol{H}^{l-1})) \\ \boldsymbol{H}^{l} &= \text{Norm}(\hat{\boldsymbol{H}}^{l} + \text{FFN}(\hat{\boldsymbol{H}}^{l})),\end{aligned} \tag{4}$$

where $\boldsymbol{H}$ is the output of hidden layers. $l = 1, ..., L$ are the indices of encoder layers. "Norm" represents the layer normalization (Ba, Kiros, and Hinton 2016) placed at the output end of the residual connections. We use $\boldsymbol{H}^0 = \text{Norm}(\hat{\boldsymbol{Z}})$ as the hidden representation of the input layer.

In the multi-head self-attention layer (MHSA), a dense computation between each pair of input frames are conducted, and thus a very long-range temporal relations between frames can be captured. The processing of a single head can be represented as follows:

$$\boldsymbol{f}_{att} = \text{Softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\alpha})\boldsymbol{V}, \tag{5}$$

where $\boldsymbol{Q} = \boldsymbol{W}_q\boldsymbol{H}$ represents a query matrix, $\boldsymbol{K} = \boldsymbol{W}_k\boldsymbol{H}$ represents a key matrix, and $\boldsymbol{V} = \boldsymbol{W}_v\boldsymbol{H}$ represents a value matrix. $\boldsymbol{W}_q$, $\boldsymbol{W}_k$ and $\boldsymbol{W}_v$ are all learnable matrices. We follow the multi-head attention configuration in BERT and set the dimension of $\boldsymbol{Q}$, $\boldsymbol{K}$ and $\boldsymbol{V}$ to $\frac{1}{M}$ of the input $\boldsymbol{H}$, $M$ represents the number of heads in the attention layer. Finally, the outputs from different heads are collected, concatenated and projected by a matrix $\boldsymbol{W}_{mhsa}$ as the output of the MHSA layer:

$$\text{MHSA}(\boldsymbol{H}) = \boldsymbol{W}_{mhsa}[\boldsymbol{f}_{att}^{(1)}; \boldsymbol{f}_{att}^{(2)}; ...; \boldsymbol{f}_{att}^{(M)}]. \tag{6}$$

For the feed-forward network (FFN), it consists of two linear layers and a GeLU layer (Hendrycks and Gimpel 2016). FFN processes each of the frame individually:

$$\text{FFN}(\boldsymbol{H}) = \boldsymbol{W}_{ffn}^{(1)}(\text{GeLU}(\boldsymbol{W}_{ffn}^{(2)}(\boldsymbol{H}))), \tag{7}$$

where $\boldsymbol{W}_{ffn}^{(1)}$ and $\boldsymbol{W}_{ffn}^{(2)}$ are the learnable linear projection matrices. Finally, we apply another 1D-convolution layer at the output end of the transformer, and final completion output $\boldsymbol{Y}$ can be written as follows:

$$\boldsymbol{Y} = \text{Conv1D}(\boldsymbol{H}^N). \tag{8}$$

## Mixture Embeddings

The mixture embedding $\boldsymbol{E}$ we used consists of a positional embedding $\boldsymbol{E}_{pos} \in \mathbb{R}^{T\times F}$ and a keyframe embedding $\boldsymbol{E}_{kf} \in \mathbb{R}^{T\times F}$, where $T$ is the length of temporal sequence and $F$ is the input feature dimension of transformer. Fig.3 gives an illustration of the mixture embedding module.

The position embedding $\boldsymbol{E}_{pos}$ is a matrix that contains $T$ sub-vectors, each for a single time step:

$$\boldsymbol{E}_{pos} = [\boldsymbol{e}_{pos}^1, \boldsymbol{e}_{pos}^2, ..., \boldsymbol{e}_{pos}^T]. \tag{9}$$

The keyframe embeddings $\boldsymbol{E}_{kf}$ are selected from a learnable dictionary $\mathcal{D}$ that contains three types of embedding vectors

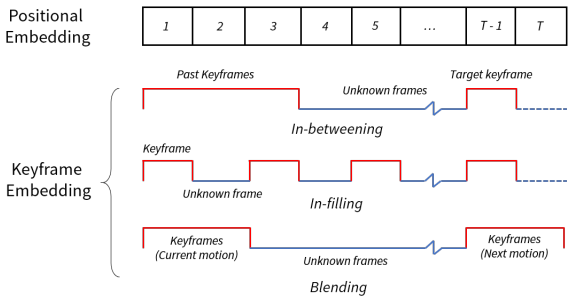


Figure 3: An illustration of mixture embedding module. We design a position embedding and a keyframe embedding, where the former one integrates position information and the latter one annotates whether the frame is the keyframe or not. (Blue dash-lines represent ignored frames that exceed the prediction range)

$\mathcal{D} = \{\hat{\boldsymbol{e}}_0, \hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2\}$, which annotate the keyframes, unknown frames, and ignored frames, respectively (Keyframe = 0, Unknown = 1, Ignored = 2). These types of frames can be configured in any forms according to different completion tasks and scenarios (e.g., in-betweening, in-filling, and blending). The keyframe embeddings are written as follows:

$$\boldsymbol{E}_{kf} = [\boldsymbol{e}_{kf}^1, \boldsymbol{e}_{kf}^2, ..., \boldsymbol{e}_{kf}^T], \tag{10}$$

where $\boldsymbol{e}_{kf}^m \in \{\hat{\boldsymbol{e}}_0, \hat{\boldsymbol{e}}_1, \hat{\boldsymbol{e}}_2\}$. Finally, the two types of embeddings are mixed by adding together position-by-position:

$$\boldsymbol{E} = \boldsymbol{E}_{pos} + \boldsymbol{E}_{kf}. \tag{11}$$

It is worth noting that our keyframe embedding is not limited to the above three configurations shown in Fig.3. It can be in any format with practical meaning. As a special case, if the keyframes are randomly specified, then our keyframe embedding turns out to be the random token mask in the original BERT paper (Devlin et al. 2018).

## Motion Perceptual Loss

We train our transformer and the embeddings with multi-task losses, including basic reconstruction loss, kinematic constraint, and motion perceptual loss.

**Reconstruction loss**: Given a set of predicted motions and their ground truth, we design our pose reconstruction loss $\mathcal{L}_{rec}$ as follows:

$$\mathcal{L}_{rec} = \frac{1}{NT}\sum_{n=1}^{N}\sum_{t=1}^{T}(\left\|\boldsymbol{p}_n^t - \hat{\boldsymbol{p}}_n^t\right\|_1 + \left\|\boldsymbol{q}_n^t - \hat{\boldsymbol{q}}_n^t\right\|_1), \tag{12}$$

where $\boldsymbol{p}_n^t$ and $\boldsymbol{q}_n^t$ are the position coordinates and rotation quaternions of the predicted motion sequence $n$ at the time step $t$. $\hat{\boldsymbol{p}}_n^t$ and $\hat{\boldsymbol{q}}_n^t$ are their ground truth. $N$ represents the length of motion sequences and $T$ is the total number of training sequences in the data set. Note that the above losses can be used in both global and local coordinate systems.

**Kinematic constraint**: We introduce two kinematics losses to improve the results in different coordinate systems:

1) Forward Kinematics (FK) loss $\mathcal{L}_{FK}$. We follow Harvey *et al.* (Harvey et al. 2020) and calculate the global position

coordinates by using local ones with forward kinematics and weight the local reconstruction loss on different joints:

$$\mathcal{L}_{FK} = \left\| \mathrm{FK}(\boldsymbol{r}, \boldsymbol{q}_{local}) - \boldsymbol{p}_{global} \right\|_1 . \tag{13}$$

2) Inverse Kinematics (IK) loss $\mathcal{L}_{IK}$. We also apply IK loss to constrain the T-pose when in global coordinate system. We first compute the local position from the global one with inverse kinematics, and then we compare the offsets between the inverse output and the original input:

$$\mathcal{L}_{IK} = \left\| \mathrm{IK}(\boldsymbol{p}_{global}, \boldsymbol{q}_{global}) - \boldsymbol{b} \right\|_1 , \tag{14}$$

where $\boldsymbol{b}$ represents the offset vector. We remove the root coordinate and keep predicted offsets only when computing the IK loss.

**Motion perceptual loss**: Inspired by the previous work (Harvey et al. 2020), we also introduce ground contact constraints to reduce the foot-skate in human motion. However, we do not predict the ground contact points but we suppress the velocity on these points to get rid of the post-processing requirements. Furthermore, we apply discrete wavelet transformation (DWT) for better capturing the high-frequency information, thus our perceptual loss can be written as follow:

$$\begin{aligned} \mathcal{L}_{per} = & \sum_{t \in \mathcal{T}_{contact}} \left\| \dot{\boldsymbol{p}}_n^t \right\|_1 \\ & + \frac{1}{L} \sum_{n=1}^{L} (\left\| DWT(\boldsymbol{p}) - DWT(\hat{\boldsymbol{p}}) \right\|_1), \end{aligned} \tag{15}$$

where $\mathcal{T}_{contact}$ is the contact points provided by datasets (e.g. LaFAN1 (Harvey et al. 2020)) and $L$ represents the level of extracted high-frequency information by DWT.

Our final training loss is written as follow:

$$\mathcal{L} = \alpha_{rec}\mathcal{L}_{rec} + \alpha_K \mathcal{L}_K + \alpha_{per}\mathcal{L}_{per}, \tag{16}$$

where $\alpha$s are the coefficients to balance different loss terms. $\mathcal{L}_K$ represents the kinematics loss in the global system ($\mathcal{L}_{IK}$) or the local system ($\mathcal{L}_{FK}$).

## Implementation Details

We adopt BERT (Devlin et al. 2018) as the backbone of our transformer with 8 encoder layers. In each encoder layer, we set the number of attention heads to $M = 8$. For our input and output Conv1D layers, the kernel size is set to 3 and the padding is set to 1. We set the dimension of the feature embedding in the MHSA layers to 256, and set those in the FFN layers to 512. In our training loss, we set $\alpha_{rec} = \alpha_{per} = 1.0$ and $\alpha_K = 0.01$. Consider that the quaternions $\boldsymbol{q} \in [0, 1]$ while the position coordinates $\boldsymbol{p}$ are in a much larger range, we scale the localization loss and the rotation loss to the same order of magnitude.

We train our network by using Adam optimizer (Kingma and Ba 2014). We set the maximum learning rate to $10^{-3}$. The whole framework is implemented using PyTorch (Paszke et al. 2019). For a more detailed training configuration, please refer to our experimental section.

# Experiments

## Motion Completion Tasks

In our experiment, we evaluate our method across three different motion completion tasks:

1. "In-betweening" on LaFAN1 (Harvey et al. 2020): LaFAN1 is a public high-quality general motion dataset introduced by Harvey *et al*. from Ubisoft. In this task, given the past 10 keyframes and another future keyframe, we aim to predict the motion of the rest frames.

2. "In-filling" on Anidance (Tang, Mao, and Jia 2018): Anidance is a public music-dance dataset proposed by Tang *et al*. (Tang, Mao, and Jia 2018). We test our method on this dataset for the in-filling task, where equally spaced keyframes are given.

3. "Blending" on our newly proposed dance dataset: We collect a new dance movement dataset, which contains high-quality dance movements performed by senior dancers and is much larger and is more challenging than the previous ones. We evaluate our method on this dataset for exploring the potential of our method in dance applications in-game environments.

Besides, since Kaufmann *et al*. 's method can also take in arbitrary input, we re-implement and apply this method across the above tasks as our comparison baseline (as there is no publicly available source code). To be fair, we adopt the same input (in the global coordinate system) and training strategy for the re-implemented version.

## Novel Dance Movement Dataset

Since dance is the most complex form of movements, we build a new challenging dance dataset to explore the potential of our method, where we invited four senior dancers to perform five types of dance movements (including Jazz dance, Street dance, J-pop dance, Indian dance, Folk dance). For better universality, we follow the classic theory of Doris Humphrey (Humphrey 1959) for choreography. We use motion capture devices (Vicon V16 cameras) to record the dance movements at 30Hz. As shown in Tab 3, our dataset is 2$\sim$4 times larger than the previous two datasets. Also, the choreography of our dataset is more professional and our dance style is more diverse. As a comparison, the LaFAN1 dataset (Harvey et al. 2020) only provided "free dance", the Anidance dataset (Tang, Mao, and Jia 2018) cannot be directly applied in the game environment since the rotation of the joints are not provided, and AIST++ dataset is built by fitting methods rather than Mocap devices (Loper et al. 2015; Kolotouros et al. 2019; Li et al. 2021).

## Metrics

We follow Harvey *et al*. (Harvey et al. 2020) and use L2Q, L2P, and NPSS as our evaluation metrics. The L2Q defines the average L2 distances of the global quaternions between the predicted motions and their ground truth. Similarly, the L2P defines the average L2 distances of the global positions.[2] The NPSS, proposed by Gopalakrishnan (Gopalakr-

[2] In Harvey *et al*. 's implementation, the predicted positions and their ground truth are normalized by mean and std of the training set. We also follow this setting for a fair comparison.

| | L2Q | | | L2P | | | NPSS | | |
|---|---|---|---|---|---|---|---|---|---|
| Length | 5 | 15 | 30 | 5 | 15 | 30 | 5 | 15 | 30 |
| Zero-Vel | 0.56 | 1.10 | 1.51 | 1.52 | 3.69 | 6.60 | 0.0053 | 0.0522 | 0.2318 |
| Interp | 0.22 | 0.62 | 0.98 | 0.37 | 1.25 | 2.32 | 0.0023 | 0.0391 | 0.2013 |
| Harvey *et al*. (LSTM-based) | 0.17 | 0.42 | 0.69 | 0.23 | 0.65 | 1.28 | 0.0020 | 0.0258 | 0.1328 |
| Kaufmann *et al*. (Conv-based) | 0.49 | 0.60 | 0.78 | 0.84 | 1.07 | 1.53 | 0.0048 | 0.0345 | 0.1454 |
| Ours (local w/ ME) | 0.18 | 0.47 | 0.74 | 0.27 | 0.82 | 1.46 | 0.0020 | 0.0307 | 0.1487 |
| Ours (local w/ ME & FK loss) | 0.17 | 0.44 | 0.71 | 0.23 | 0.74 | 1.37 | 0.0019 | 0.0291 | 0.1430 |
| Ours (global w/ ME) | 0.14 | 0.36 | 0.61 | **0.21** | 0.57 | 1.11 | 0.0016 | 0.0238 | 0.1241 |
| Ours (global w/ ME & IK loss) | **0.14** | **0.36** | 0.61 | 0.22 | 0.56 | 1.10 | **0.0016** | **0.0234** | 0.1222 |
| Ours (global transformer only) | 0.16 | 0.37 | 0.63 | 0.24 | 0.61 | 1.16 | 0.0018 | 0.0243 | 0.1284 |
| Ours (global full) | 0.18 | 0.37 | **0.61** | 0.23 | **0.56** | **1.06** | 0.0018 | 0.0238 | **0.1218** |
| Ours (noisy training data - 30 db) | 0.19 | 0.39 | 0.63 | 0.27 | 0.59 | 1.11 | 0.0020 | 0.0248 | 0.1259 |

Table 1: Experimental results on LaFAN1 dataset.Interpolation and zero-velocity are used as the naive baselines following (Harvey et al. 2020). A lower score indicates better performance. (*Note that for a fair comparison, the T-pose of our global results have been replaced by a standard one in local coordinate system.

| Method | 1 x 30 | 10 x 30 | CPU info |
|---|---|---|---|
| Harvey *et al*. | 0.31s | 0.40s | E5-1650 |
| Kaufmann *et al*. | 0.066s | 0.33s | I7-8700K |
| Ours(global full) | **0.025s** | **0.083s** | I7-8700K |

Table 2: Speed performance comparison on LaFAN1 dataset. CPU inference time are recorded in different batch sizes (1 & 10) where In-betweening length is set to 30 frames (i.e. 1 second).

| Dataset | Frames | Duration | Types | Format |
|---|---|---|---|---|
| LaFAN1* | 45,690 | 0.42h | 1 | pos + rot |
| Anidance | 101,390 | 1.13h | 4 (pro) | pos only |
| Ours | 228,165 | 2.11h | 5 (pro) | pos + rot |

* Only dance-related subset of LaFAN1 is reported.

Table 3: Comparison between three Mocap dance datasets.

ishnan et al. 2019), is a variant of L2Q, which computes the Normalized Power Spectrum Similarity and is based on angular frequency distance between the prediction and the ground truth.

Note that when we generate our results based on the global coordinate system, we replace the T-pose with the standard one under the local system (by using global to local coordinate transformation) for a fair comparison.

## Motion In-Betweening

We evaluate our method on the LaFAN1 dataset (Harvey et al. 2020) for the motion in-betweening task. The whole dataset contains 496,672 frames performed by 5 motion subjects and are recorded by using Mocap in 30Hz. The training set and the test set are clearly separated, where the test set contains motions from subject No.5 only. Since the originally captured motions are in very long sequences, we follow Harvey *et al*. (Harvey et al. 2020) and introduce motion windows to this dataset, where the width of the window is set to 50 (65) frames, and the offset is set to 20 (25) frames for training (test). Finally, there are 20,212 and 2,232 windows extracted for training and testing, respectively. Since our method can take in arbitrary-length inputs, we set the transition length of in-betweening to 5$\sim$39 (since the maximum length is 39). For the unknown frames, before feeding them to our model, we interpolate them based on the nearest two keyframes by linear interpolation (LERP) and spherical linear interpolation (SLERP).

During the evaluation, only the first 10 keyframes and another keyframe at the frame $10+L+1$ are given, where $L$ is a predefined transition length. We keep the same configuration with Harvery *et al.* , where transition lengths are set to 5, 15, and 30. We evaluate our method under both local and global coordinate systems.

| | L2P | | |
|---|---|---|---|
| Length | 5 | 15 | 30 |
| Zero-Vel | 2.34 | 5.12 | 6.73 |
| Interp | 0.94 | 3.24 | 4.68 |
| Kaufmann *et al*. | 3.57 | 3.69 | 3.93 |
| Ours (full) | **0.84** | **1.46** | **1.64** |

Table 4: In-filling results on Anidance dataset (Tang, Mao, and Jia 2018). Since only position coordinates are provided in this dataset, we cannot compute the L2Q and NPSS score. (A lower L2P score indicates a better performance)

We show in Tab 1 that our method can still achieve high accuracy on the LaFAN1 dataset even when the transition length $L = 30$, better than the results of Kaufmann *et al*. (Kaufmann et al. 2020) and Harvey *et al*. (Harvey et al. 2020). We can also see that there is a noticeable accuracy improvement when switching the generation mode from local

| | L2Q | | | L2P | | | NPSS | | |
|---|---|---|---|---|---|---|---|---|---|
| Length | 8 | 16 | 32 | 8 | 16 | 32 | 8 | 16 | 32 |
| Zero-Vel | 0.93 | 1.41 | 1.89 | 2.71 | 4.08 | 5.73 | 0.0221 | 0.1003 | 0.4875 |
| Interp | 0.40 | 0.92 | 1.44 | 1.16 | 2.34 | 3.70 | 0.0122 | 0.1095 | 0.4726 |
| Kaufmann *et al*. | 1.36 | 1.35 | 1.35 | 3.64 | 3.63 | 3.64 | 0.0392 | 0.1223 | 0.3944 |
| Ours | **0.35** | **0.60** | **0.98** | **0.79** | **1.35** | **2.32** | **0.0122** | **0.0631** | **0.3124** |
| Ours (enhanced) | **0.29** | **0.51** | **0.89** | **0.62** | **1.12** | **2.12** | **0.0098** | **0.0529** | **0.2817** |

Table 5: Blending results of our new dance dataset (A lower score indicates a better performance).

to global. This may be because of the accumulative errors of rotations in local coordinates, and the previous method proposes to use the FK loss to reduce this error (Harvey et al. 2020). Our results suggest that the motion completion can be better solved in the global coordinate system, although the T-pose predicted in the global coordinate system may not be accurate enough (can be further improved by IK loss). We further evaluate the Mixture Embedding (ME) and perceptual loss used in our method, where the former one improves the accuracy in all evaluation settings (L=5, 15, and 30) and the latter one can significantly improve the position accuracy (L2P) in long-term generation ($L = 30$). The noisy training data also shows that our method has good robustness.

Besides, Tab 2 indicates that our method can achieve a very high inference speed on CPU devices and can even run in real-time ($< 0.033$s), which benefits from its non-autoregressive design.

## Dance In-Filling

Next, we evaluate our method on the in-filling task of the Anidance dataset (Tang, Mao, and Jia 2018). In this dataset, four types of dance movements (Cha-cha, Tango, Rumba, and Waltz) are captured in the global coordinate system. This dataset was originally designed for music-to-dance generation and contains 61 independent dance fragments with 101,390 frames.

We apply similar evaluation settings on this dataset, where the time window is set to 128 frames and the offset is set to 64. 20% dance fragments are randomly selected as the test set, and thus there are 1,117 sequences in the training set and 323 sequences in the test set. We train our model on this dataset with 3000 epochs, and the rest of the configurations are kept the same with the in-betweening task. We set the interval of keyframes to 5$\sim$30 for in-filling, which means that there are only 5 keyframes given to the model when their interval is 30, and the initial transitions between keyframes are also interpolated by LERP.

Tab 4 shows the evaluation results of our method on the test set of Anidance. The transition lengths are set to 5, 15, and 30 respectively. We evaluate our method under the global coordinate system since the Anidance dataset contains global positions only. Similar to the results in the in-betweening task, our method can achieve the highest completion accuracy among all comparison methods on both long-term and short-term completion settings, while Kaufmann *et al*. 's results contain some dislocations of joints.

## Dance Blending

To further evaluate our method on a much more complex motion completion application, we test our method on the dance blending task using our own dance dataset. In this experiment, the time window is set to 64 and the offset is set to 32, and there are finally 5,651 sequences for training and 1,379 sequences for testing. We keep the same experimental configurations with the in-betweening task, but half the maximum learning rate.

We set the blending window to 5$\sim$32 frames. For example, when the blending window is set to 32, the 16 frames before and after this window are given as keyframes, and frames within this window are masked out. Tab 5 shows our evaluation results with the window width = 8, 16, 32 respectively. We can see our method can still generate very high-quality results on this challenging dataset.

## Scaling Capability

To further explore the scaling capability of our method, we collect an extra-large dance dataset that is 3x larger than our current version. We pre-train our method on this dataset by using the setting of in-betweening task and then fine-tuning on our own standard dance dataset for 200 epochs with $10^{-4}$ learning rate. From the last row of Tab 5, we can see the enhanced version of our method gets $\sim$ 10% relative improvement over our standard version when more training data is provided, which suggests that our model still has great potential for further improvement.

# Conclusion

In this paper, we propose a simple but effective method to solve motion completion problems under a unified framework. In our method, a standard transformer encoder, a mixture embedding, and a new motion perceptual loss are introduced to handle arbitrary motion sequence input. Our method can predict multiple missing frames in a single forward propagation rather than running in an auto-regressive manner and does not require any post-processing. Experimental results show that our method can be well applied to different motion completion tasks, including in-betweening, in-filling, and blending. Our method has better computational efficiency, good scaling capability, and also achieves a new state-of-the-art result on the LaFAN1 dataset.

## References


Aberman, K.; Li, P.; Lischinski, D.; Sorkine-Hornung, O.; Cohen-Or, D.; and Chen, B. 2020a. Skeleton-aware networks for deep motion retargeting. *ACM Transactions on Graphics (TOG)*, 39(4): 62–1.

Aberman, K.; Weng, Y.; Lischinski, D.; Cohen-Or, D.; and Chen, B. 2020b. Unpaired motion style transfer from video to animation. *ACM Transactions on Graphics (TOG)*, 39(4): 64–1.

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

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. *arXiv preprint arXiv:1607.06450*.

Baram, N.; Anschel, O.; and Mannor, S. 2016. Model-based adversarial imitation learning. *arXiv preprint arXiv:1612.02179*.

Barsoum, E.; Kender, J.; and Liu, Z. 2018. HP-GAN: Probabilistic 3D human motion prediction via GAN. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 1418–1427.

Beaudoin, P.; Coros, S.; van de Panne, M.; and Poulin, P. 2008. Motion-motif graphs. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, 117–126.

Bergamin, K.; Clavet, S.; Holden, D.; and Forbes, J. R. 2019. DReCon: data-driven responsive control of physics-based characters. *ACM Transactions on Graphics (TOG)*, 38(6): 1–11.

Chai, J.; and Hodgins, J. K. 2007. Constraint-based motion optimization using a statistical dynamic model. In *ACM SIGGRAPH 2007 papers*, 8–es.

Ciccone, L.; Öztireli, C.; and Sumner, R. W. 2019. Tangent-space optimization for interactive animation control. *ACM Transactions on Graphics (TOG)*, 38(4): 1–10.

Coros, S.; Beaudoin, P.; and Van de Panne, M. 2009. Robust task-based control policies for physics-based characters. In *ACM SIGGRAPH Asia 2009 papers*, 1–9.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Gomes, T. L.; Martins, R.; Ferreira, J.; Azevedo, R.; Torres, G.; and Nascimento, E. R. 2021. A Shape-Aware Retargeting Approach to Transfer Human Motion and Appearance in Monocular Videos. *International Journal of Computer Vision*, 1–19.

Gopalakrishnan, A.; Mali, A.; Kifer, D.; Giles, L.; and Ororbia, A. G. 2019. A neural temporal model for human motion prediction. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 12116–12125.

Grochow, K.; Martin, S. L.; Hertzmann, A.; and Popović, Z. 2004. Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers*, 522–531.

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

Harvey, F. G.; and Pal, C. 2018. Recurrent transition networks for character locomotion. In *SIGGRAPH Asia 2018 Technical Briefs*, 1–4.

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

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

Hernandez, A.; Gall, J.; and Moreno-Noguer, F. 2019. Human motion prediction via spatio-temporal inpainting. In *Proceedings of the IEEE International Conference on Computer Vision*, 7134–7143.

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

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

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

Humphrey, D. 1959. *The art of making dances*. Dance Horizons.

Kaufmann, M.; Aksan, E.; Song, J.; Pece, F.; Ziegler, R.; and Hilliges, O. 2020. Convolutional Autoencoders for Human Motion Infilling. In *8th international conference on 3D Vision (3DV 2020)(virtual)*, 263.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kolotouros, N.; Pavlakos, G.; Black, M. J.; and Daniilidis, K. 2019. Learning to reconstruct 3D human pose and shape via model-fitting in the loop. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2252–2261.

Kovar, L.; Gleicher, M.; and Pighin, F. 2008. Motion graphs. In *ACM SIGGRAPH 2008 classes*, 1–10.

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

Lee, K.; Lee, S.; and Lee, J. 2018. Interactive character animation by learning multi-objective control. *ACM Transactions on Graphics (TOG)*, 37(6): 1–10.

Lehrmann, A. M.; Gehler, P. V.; and Nowozin, S. 2014. Efficient nonlinear markov models for human motion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1314–1321.

Levine, S.; Wang, J. M.; Haraux, A.; Popović, Z.; and Koltun, V. 2012. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)*, 31(4): 1–10.

Li, R.; Yang, S.; Ross, D. A.; and Kanazawa, A. 2021. Ai choreographer: Music conditioned 3d dance generation with aist++. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 13401–13412.

Loper, M.; Mahmood, N.; Romero, J.; Pons-Moll, G.; and Black, M. J. 2015. SMPL: A skinned multi-person linear model. *ACM transactions on graphics (TOG)*, 34(6): 1–16.

Min, J.; and Chai, J. 2012. Motion graphs++ a compact generative model for semantic motion analysis and synthesis. *ACM Transactions on Graphics (TOG)*, 31(6): 1–12.

Min, J.; Chen, Y.-L.; and Chai, J. 2009. Interactive generation of human animation with deformable motion models. *ACM Transactions on Graphics (TOG)*, 29(1): 1–12.

Ngo, J. T.; and Marks, J. 1993. Spacetime constraints revisited. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, 343–350.

Parisotto, E.; Song, F.; Rae, J.; Pascanu, R.; Gulcehre, C.; Jayakumar, S.; Jaderberg, M.; Kaufman, R. L.; Clark, A.; Noury, S.; et al. 2020. Stabilizing transformers for reinforcement learning. In *International Conference on Machine Learning*, 7487–7498. PMLR.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, 8024–8035. Curran Associates, Inc.

Peng, X. B.; Abbeel, P.; Levine, S.; and van de Panne, M. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions on Graphics (TOG)*, 37(4): 1–14.

Peng, X. B.; Berseth, G.; Yin, K.; and Van De Panne, M. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)*, 36(4): 1–13.

Rose, C.; Guenter, B.; Bodenheimer, B.; and Cohen, M. F. 1996. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 147–154.

Safonova, A.; and Hodgins, J. K. 2007. Construction and optimal search of interpolated motion graphs. In *ACM SIGGRAPH 2007 papers*, 106–es.

Sidenbladh, H.; Black, M. J.; and Sigal, L. 2002. Implicit probabilistic models of human motion for synthesis and tracking. In *European conference on computer vision*, 784–800. Springer.

Tang, T.; Mao, H.; and Jia, J. 2018. AniDance: Real-Time Dance Motion Synthesize to the Song. In *Proceedings of the 26th ACM international conference on Multimedia*, 1237–1239.

Tay, Y.; Dehghani, M.; Bahri, D.; and Metzler, D. 2020. Efficient transformers: A survey. *arXiv preprint arXiv:2009.06732*.

Taylor, G. W.; Hinton, G. E.; and Roweis, S. T. 2007. Modeling human motion using binary latent variables. In *Advances in neural information processing systems*, 1345–1352.

Thomas, A. 2009. *Integrated graphic and computer modelling*. Springer Science & Business Media.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Wang, J. M.; Fleet, D. J.; and Hertzmann, A. 2007. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence*, 30(2): 283–298.

Witkin, A.; and Kass, M. 1988. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4): 159–168.

Ye, Y.; and Liu, C. K. 2010. Synthesis of responsive motion using a dynamic model. In *Computer Graphics Forum*, volume 29, 555–562. Wiley Online Library.

Yin, K.; Loken, K.; and Van de Panne, M. 2007. Simbicon: Simple biped locomotion control. *ACM Transactions on Graphics (TOG)*, 26(3): 105–es.

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