


# Example-based Motion Synthesis via Generative Motion Matching

**Journal Article****Author(s):**

Li, Weiyu; Chen, Xuelin; Li, Peizhuo; [Sorkine-Hornung, Olga](#) ; Chen, Baoquan

**Publication date:**

2023-08

**Permanent link:**

<https://doi.org/10.3929/ethz-b-000626984>

**Rights / license:**

[In Copyright - Non-Commercial Use Permitted](#)

**Originally published in:**

ACM Transactions on Graphics 42(4), <https://doi.org/10.1145/3592395>

**Funding acknowledgement:**

101003104 - Sustainable Algorithmic Modeling of Personalized Garments (EC)

# Example-based Motion Synthesis via Generative Motion Matching

WEIYU LI<sup>\*†</sup>, Shandong University, China

XUELIN CHEN<sup>\*‡</sup>, Tencent AI Lab, China

PEIZHUO LI, ETH Zurich, Switzerland

OLGA SORKINE-HORNUNG, ETH Zurich, Switzerland

BAOQUAN CHEN, Peking University, China



Fig. 1. Our generative framework enables a variety of example-based motion synthesis tasks, that usually require long offline training for existing data-driven methods. Given a single or few examples, even with a highly complex skeletal structure (middle), our framework can (a) synthesize a high-quality novel motion, within a fraction of a second; (b) complete a partial motion (lower-body motion) with example motion patches; (c) synthesize a coherent sequence guided by a sparse set of keyframes (in blue clothes); (d) generate an infinitely looping animation that starts and ends with a specified pose (in blue clothes).

We present GenMM, a generative model that “mines” as many diverse motions as possible from a single or few example sequences. In stark contrast to existing data-driven methods, which typically require long offline training time, are prone to visual artifacts, and tend to fail on large and complex skeletons, GenMM inherits the training-free nature and the superior quality of the well-known *Motion Matching* method. GenMM can synthesize a high-quality motion within a fraction of a second, even with highly complex and large skeletal structures. At the heart of our generative framework lies

the generative motion matching module, which utilizes the bidirectional visual similarity as a generative cost function to motion matching, and operates in a multi-stage framework to progressively refine a random guess using exemplar motion matches. In addition to diverse motion generation, we show the versatility of our generative framework by extending it to a number of scenarios that are not possible with motion matching alone, including motion completion, key frame-guided generation, infinite looping, and motion reassembly.

<sup>\*</sup>Joint first authors

<sup>†</sup>Work done during an internship at Tencent AI Lab

<sup>‡</sup>Corresponding author

Authors' addresses: Weiyu Li, weiyu.li.cn@gmail.com, Shandong University, China; Xuelin Chen, xuelin.chen.3d@gmail.com, Tencent AI Lab, China; Peizhuo Li, peizhuo.li@inf.ethz.ch, ETH Zurich, Switzerland; Olga Sorkine-Hornung, sorkine@inf.ethz.ch, ETH Zurich, Switzerland; Baoquan Chen, baoquan@pku.edu.cn, Peking University, China.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from [permissions@acm.org](mailto:permissions@acm.org).

© 2023 Copyright held by the owner/author(s). Publication rights licensed to ACM.

0730-0301/2023/8-ART1 \$15.00

<https://doi.org/10.1145/3592395>

CCS Concepts: • **Computing methodologies** → **Motion processing**.

Additional Key Words and Phrases: motion synthesis, generative model, motion matching

## ACM Reference Format:

Weiyu Li, Xuelin Chen, Peizhuo Li, Olga Sorkine-Hornung, and Baoquan Chen. 2023. Example-based Motion Synthesis via Generative Motion Matching. *ACM Trans. Graph.* 42, 4, Article 1 (August 2023), 12 pages. <https://doi.org/10.1145/3592395>

## 1 INTRODUCTION

The generation of natural, varied, and detailed motions is a core problem in computer animation. Acquiring large volumes of motion data via a motion capture (mocap) system or manually authoring sophisticated animations is known to be costly and tedious. As such, motion datasets are generally limited, especially in terms of

the diversity of style, skeletal structures, or creature types, which hamper the effectiveness of existing data-driven motion synthesis methods. Advancing generative abilities of synthesizing diverse and extensive motions from limited example sequences has therefore become an important research problem.

In recent years, deep learning has taken the field of computer animation by storm. Deep learning methods have demonstrated the ability to synthesize diverse and natural motions when training on large and comprehensive datasets [Holden et al. 2016, 2017; Henter et al. 2020; Tevet et al. 2022a; Raab et al. 2023a; Tseng et al. 2022]. More encouragingly, the success was recently reproduced in an extremely reduced setting [Li et al. 2022], where only one sequence is provided for training, yet, the neural network learns the sample’s internal distribution, and demonstrates the ability to synthesize diverse variants of the input example sequence. Nevertheless, neural motion synthesis methods carry several drawbacks that limit their applicability in practice: (i) they require long training time; (ii) they are prone to visual artifacts such as jittering or over-smoothing; (iii) they do not scale well to large and complex skeleton structures.

In this paper, we explore an alternative approach to the problem. We revisit the classical idea in computer animation – *motion nearest neighbors* [Lee et al. 2010], which dates back long before the deep learning era and on which the state-of-the-art industrial solution for character animation – *motion matching* – was founded [Büttner and Clavet 2015], delivering exceptionally high-quality motion synthesis. Motion matching produces character animations that appear natural and respond to varying local contexts. Using a large mocap database as a local approximate of the entire natural motion space, motion matching simply searches for a motion patch that best fits a given local context. The dependence on a large dataset is, however, at odds with our goal: we are after a generative model that “mines” as many *diverse* motions as possible from a *single* or *few* examples. Inspired by the work of Granot et al. [2022] in image synthesis, we take the following insights for casting motion matching into our generative model and yield *generative motion matching* (GenMM, pronounced “gem”). First, to retain the motion quality of motion matching and inject generative capabilities, we exploit bidirectional similarity introduced in [Simakov et al. 2008] as a new generative cost function for motion matching. The bidirectional similarity serves the purpose of comparing the patch distribution between the example and the synthesized sequence. Specifically, it encourages the synthesized sequence to contain only motion patches from the example sequences, and vice versa, the examples should only contain motion patches from the synthesis. Consequently, no artifacts are introduced in the synthesis, and importantly, no critical motion patches are lost either. Second, we use a multi-stage framework to progressively synthesize a motion sequence that has minimal patch distribution discrepancy with the example, capturing patch distributions from varying temporal resolutions. Lastly, we utilize the observation that the generative diversity of GAN-based methods stems primarily from the unconditional noise input [Granot et al. 2022]: we input noise to the coarsest synthesis stage, and achieve highly diverse synthesis results.

We demonstrate that GenMM is more than competent in producing diverse motions from only a small set of input examples. Notably, compared to existing works, GenMM offers several advantages:

- GenMM runs very fast, without any pre-training. A motion sequence can be synthesized within a fraction of a second.
- GenMM inherits the appealing nature of motion matching, producing motions of high quality and fidelity.
- GenMM scales smoothly to highly complex skeletons (see the character with 433 joints in Figure 1), where neural networks struggle [Li et al. 2022].
- It is easy to extend GenMM to inputs with multiple sequences and encourage the synthesis to cover all examples, which is non-trivial for GAN-based methods [Li et al. 2022].

In addition to diverse motion generation, we also demonstrate the versatility of our generative framework by extending to an array of scenarios that are unachievable with motion matching alone, such as motion completion, key frame-guided generation, infinite looping, and motion reassembly, all enabled by the shared foundation of generative motion matching.

## 2 RELATED WORK

We review the most related work on kinematics-based motion synthesis. We also briefly cover recent advancements in image synthesis, particularly patch-based ones, from which we take inspiration.

*Motion Synthesis.* Generating novel motions via diversifying existing ones can date back to the work of Perlin and Goldberg [1996], where the Perlin noise [Perlin 1985] is added to motion clips for obtaining variants with local diversity. Pullen and Bregler [2002] show that mocap data can be used to enhance a coarse key-framed motion, by matching low-frequency patches and blending high-frequency details. Li et al. [2002] construct a graph model by matching similar patches in the dataset, and create a stochastic model for generating random samples with local and structural variations. Due to the use of a linear dynamics model, the model requires a large training dataset to achieve satisfactory results and faces a dilemma of quality and diversity. Contemporarily, motion graphs [Kovar et al. 2002; Lee et al. 2002; Arikian and Forsyth 2002] use a similar discrete graph model while keeping it deterministic, namely a state machine, and demonstrate characters that respond interactively to the user input. However, the discrete space inherently limits their agility and responsiveness. Hence, efforts on summarizing large datasets into statistical models have also been made since then [Pullen and Bregler 2000; Brand and Hertzmann 2000; Bowden 2000; Grochow et al. 2004; Chai and Hodgins 2007; Wang et al. 2007]. Instead of sorting the dataset into an organized but discrete structure or a statistical model, motion nearest neighbors [Lee et al. 2010; Levine et al. 2012] operate directly on contiguous *motion fields* to learn a control policy that interpolates the nearest neighbors of the current pose. Following that, [Büttner and Clavet 2015] introduce Motion Matching, which is a method searching a large database of animations for the animation which best fits the given context. This method has quickly been adopted by many studios due to its simplicity, flexibility, controllability, and the quality of the motion it produces [Harrower 2018; Büttner 2019]. Motion matching plays back the animation data stored in the database as-is, rendering it the de facto state-of-the-art in the industry. Nevertheless, its goal differs significantly from ours, as we target a generative model that synthesizes diverse motions from examples.

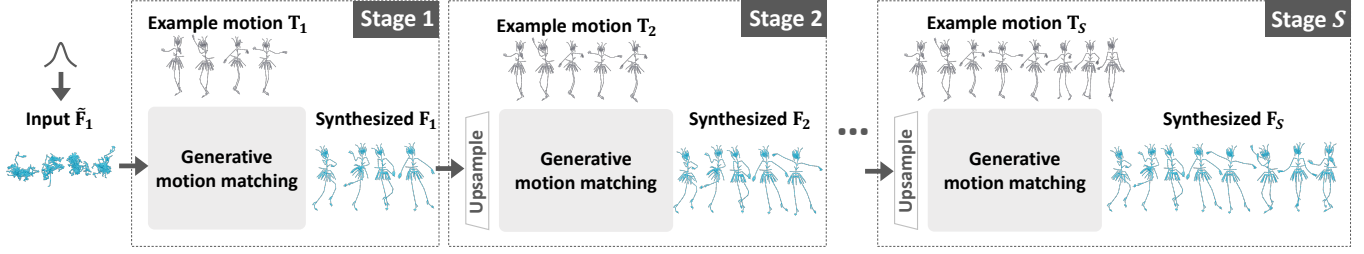


Fig. 2. Multi-stage motion synthesis. Starting from the coarsest stage, the generative motion matching at each stage  $s$  takes in an upsampled version of the output from the preceding stage as the initial guess, refines it with motion patches in the example motion  $T_s$ , and outputs a finer motion sequence  $F_s$ . Note the coarsest stage is purely generative, as the input is merely a Gaussian noise.

Recent advancements in deep learning also greatly impact the motion synthesis field. Early attempts [Holden et al. 2015, 2016] use deep neural networks to learn from animation data. Deep neural networks can learn a strong prior from a large dataset [Rempe et al. 2021; He et al. 2022], solving many ill-posed generative tasks including motion prediction [Fragkiadaki et al. 2015; Pavlo et al. 2018], motion in-betweening [Harvey et al. 2020; Duan et al. 2022; Qin et al. 2022], motion reassembling [Jang et al. 2022; Lee et al. 2022], text-guided motion synthesis [Tevet et al. 2022a,b; Zhao et al. 2023], etc. Holden et al. [2015, 2016] apply modern deep learning techniques for learning from animation data. Meanwhile, combining motion matching with deep learning has also resulted in variants that are computationally less expensive [Holden et al. 2020] and more versatile [Habibie et al. 2022]. Notably, all these works require a large and comprehensive dataset for training. Another noticeable line of work adapts deep reinforcement learning techniques to train a physically simulated character with a small set of example motions [Peng et al. 2018, 2021]. More recently, [Li et al. 2022], the most related work to us, proposes to use a patch GAN-based [Isola et al. 2017; Shaham et al. 2019] approach to train a generative model with a single example. Concurrently, Raab et al. [2023b] introduce a diffusion-based model that learns the internal motifs of a single motion clip for producing diverse outputs. Nonetheless, these two methods struggle to produce results with sharp motions, and are not suitable for training with multiple examples due to the discontinuous underlying latent space presented to it. We show the superiority of our method over GANimator by an in-depth comparison in Section 4.1.

**Image Synthesis.** Our work adopts several algorithmic designs from texture image synthesis, that shares a similar goal with motion synthesis. For an in-depth survey of this extensive body we refer readers to surveys [Wei et al. 2009; Barnes and Zhang 2017]. The image pyramid, also known as progressive generation [Karras et al. 2018] in deep learning, had been used in texture synthesis long ago. Heeger and Bergen [1995]; De Bonet [1997] use a Laplacian pyramid [Burt and Adelson 1987] for texture synthesis, realizing progressive generation on the spatial frequency domain. Wei and Levoy [2000] use a Gaussian pyramid for a similar purpose. Han et al. [2008] push the multi-scale generation to a new height, where gigapixel-sized images with great details can be synthesized. We also adopt the progressive synthesis, allowing our generative motion matching module to capture details of different levels. Progressive

synthesis has also become popular in today’s era of deep learning, leading to impressive generative models that learn to progressively refine random noise into images resembling a single natural image. Specifically, a series of GANs [Goodfellow et al. 2014] are trained to capture the patch distribution of the example at varying scales. Following that, Granot et al. [2022] show that bidirectional visual similarity [Simakov et al. 2008] can serve the purpose of measuring the patch distribution discrepancy between the example and the synthesized image, leading to diverse images of much higher quality and fast synthesis, compared to GAN-based methods.

### 3 METHOD

We elaborate details of our generative framework, that can synthesize high-quality motions resembling given examples, in large quantities and varieties. Although our method can take as input *multiple* exemplar motions, in our coverage, to ease the understanding of the algorithm, we mainly describe in the *single* input setting. Moreover, the synthesized motion does not need to match exactly the length of the example and can be of arbitrary length.

#### 3.1 Motion Representation

A motion sequence is defined by a temporal set of  $T$  poses that each consists of root joint displacements  $\mathbf{O} \in \mathbb{R}^{T \times 3}$  and joint rotations  $\mathbf{R} \in \mathbb{R}^{T \times J \times Q}$ , where  $J$  is the number of joints and  $Q$  is the number of rotation features. Instead of directly using the global root displacements  $\mathbf{O}$ , we convert  $\mathbf{O}$  to local root displacements  $\mathbf{V}$ , that are temporal-invariant and calculated as the difference between every two consecutive poses. The joint rotations are defined in the coordinate frame of their parent in the kinematic chain, and we use the 6D rotation representation (i.e.,  $Q = 6$ ) proposed by Zhou et al. [2019].

As human eyes are rather sensitive to implausible interactions between the end-effector and the ground, existing neural-based methods usually establish geometric losses on the locations and velocities of the end-effectors, i.e., the foot contact loss [Shi et al. 2020; Li et al. 2022; Tevet et al. 2022b], whereas our method does not demand such a design as the internal structure of exemplar motion patches, such as the high correlation between the end-effectors and the root motion, are inherently preserved in the synthesis. That said, it is also trivial to incorporate foot contact labels as in [Li et al. 2022] into our representation for improvements in *rare* cases, where



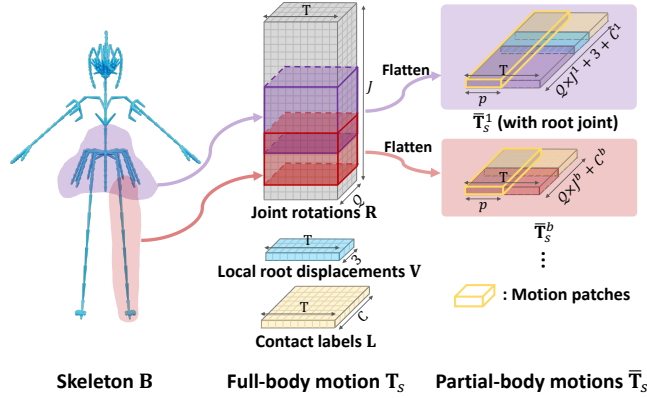


Fig. 3. Skeleton-aware motion patch extraction. The skeleton is partitioned into several overlapping skeletal parts (two coloreds on the left), with which the full-body motion can be split into a set of partial-body motions accordingly. Then motion patches (yellow boxes) with a temporal size of  $p$  frames can be extracted from each partial-body motion.

sliding feet could occur in the synthesis with large root-motion examples. The contact label enables the IK post-process to avoid floating feet. Specifically, the contact labels  $L$  can be easily retrieved from the input motion by setting a threshold of the magnitude of the velocity. Assume the number of foot joints is  $C$ , for each foot joint at a timestamp, we calculate a binary vector and append it to the per-frame feature (See Figure 3 and 4).

For convenience, we let  $M_H \equiv \mathbb{R}^{H \times (JQ+3+C)}$  denote the metric space of concatenated motion features of  $H$  frames,  $T \equiv [R, V] \in M_T$  the original input motion features,  $T_i \in M_{T_i}$  a corresponding downsampled version of the input,  $F \in M_F$  the synthesized motion features of  $F$  frames, and  $F_s$  a corresponding downsampled version of  $F$ .

### 3.2 Multi-stage Motion Synthesis

Figure 2 presents the overall pipeline of our approach, which consists of  $S$  stages to progressively synthesize a motion of  $F$  frames. Specifically, given an input motion, we build an exemplar pyramid  $\{T_1, \dots, T_S\}$ , where  $T_S = T$  is the original input sequence and  $T_s \in M_{T_s}$  is  $T_{s+1}$  downsampled by a factor  $r > 1$ . Then, a synthesis pyramid  $\{F_1, \dots, F_S\}$ , where  $F_S \in M_F$  is the final synthesized sequence of  $F$  frames and  $F_s \in M_{F_s}$  is an intermediate sequence of  $F \cdot r^{s-S}$  frames, will be synthesized in a coarse-to-fine manner, starting from the coarsest stage and up to the finest. At each stage  $s$ , the generative motion matching module (Section 3.3) takes in an up-sampled version of the output from the preceding stage as the initial guess,  $\tilde{F}_s = F_{s-1} \uparrow^r$ , refines it with exemplar motion patches in  $T_s$ , and outputs a finer motion sequence  $F_s$ . Note that the synthesis at the coarsest stage is purely generative, as the input is merely a noise drawn from a Gaussian distribution, i.e.,  $\tilde{F}_1 \sim \mathcal{N}(\mu, \sigma^2) \in M_{F_1}$ .

### 3.3 Generative Motion Matching

Typically, patch-based image synthesis consists of three steps, namely the patch extraction, nearest neighbor matching, and blending, that

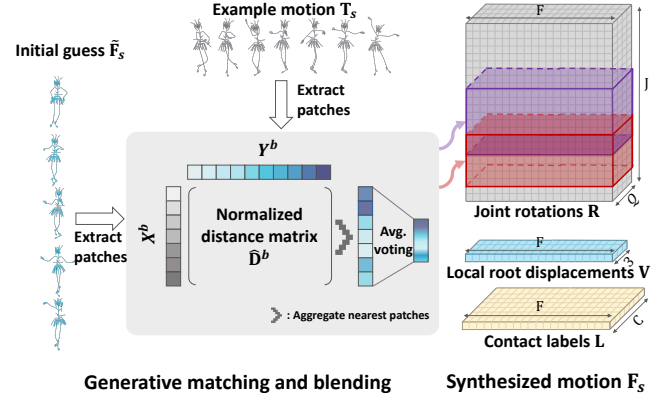


Fig. 4. Generative matching and blending. Each motion patch in the initial guess finds the best-matched motion patch in the example motion, according to the normalized distance matrix. Then we blend the overlapping matched patches to form a novel partial motion. Finally, we blend multiple resultant partial motions to get the final full-body motion (see right).

work in sequence to produce a converging result in multiple iterations. Our method follows a similar approach but with algorithmic designs specific to our task. Specifically, at each stage  $s$ , the following steps are invoked sequentially during  $E$  iterations.

**Skeleton-aware Motion Patch Extraction.** A motion patch can be defined trivially as a sub-sequence of  $p$  consecutive frames in the example sequence, which is a common practice in motion synthesis [Li et al. 2022; Büttner and Clavet 2015]. While our approach can simply work with this definition, we further propose to extract skeleton-aware motion patches from the motion sequence, which decomposes the skeleton into multiple sub-sets, i.e., skeletal parts, instead of treating it as a whole, and eventually leads to more *diverse poses*. Specifically, let  $B$  denote the skeletal tree used by the example full-body motion  $T_s$ , a set of skeletal parts  $\{B_1, \dots, B_B\}$ , where  $B_b \subset B$  is a sub-tree of the whole skeleton and has  $J^b$  joints, can be defined to divide the full-body motion into a set of partial-body motions  $\{\bar{T}_s^1, \dots, \bar{T}_s^B\}$ , from which we crop sub-sequences of  $p$  frames with stride size 1 as our motion patches (See Figure 3).

Usually, skeletons across different animations do not necessarily follow specific rules and can be of extremely high variability, for example, bipeds vs. hexapods, a pure biological skeleton vs. one with more artistic joints, etc. Hence, our approach allows the user to manually divide the whole skeletal structure into sub-parts with overlapping joints, similar as in [Jang et al. 2022; Lee et al. 2022].

**Generative Matching.** Let  $X$  denote the set of motion patches extracted from  $\tilde{F}_s$ ,  $Y$  the set of motion patches extracted from the example motion  $T_s$ . We calculate pairwise patch distance matrices using squared- $L_2$  distance, which provides the foundation for the measurement of the similarity between each exemplar motion patch and each synthesized motion patch. Note that the patch distance matrix is calculated *per* skeletal part:

$$D_{i,j}^b = \|X_i^b - Y_j^b\|_2^2, \quad (1)$$

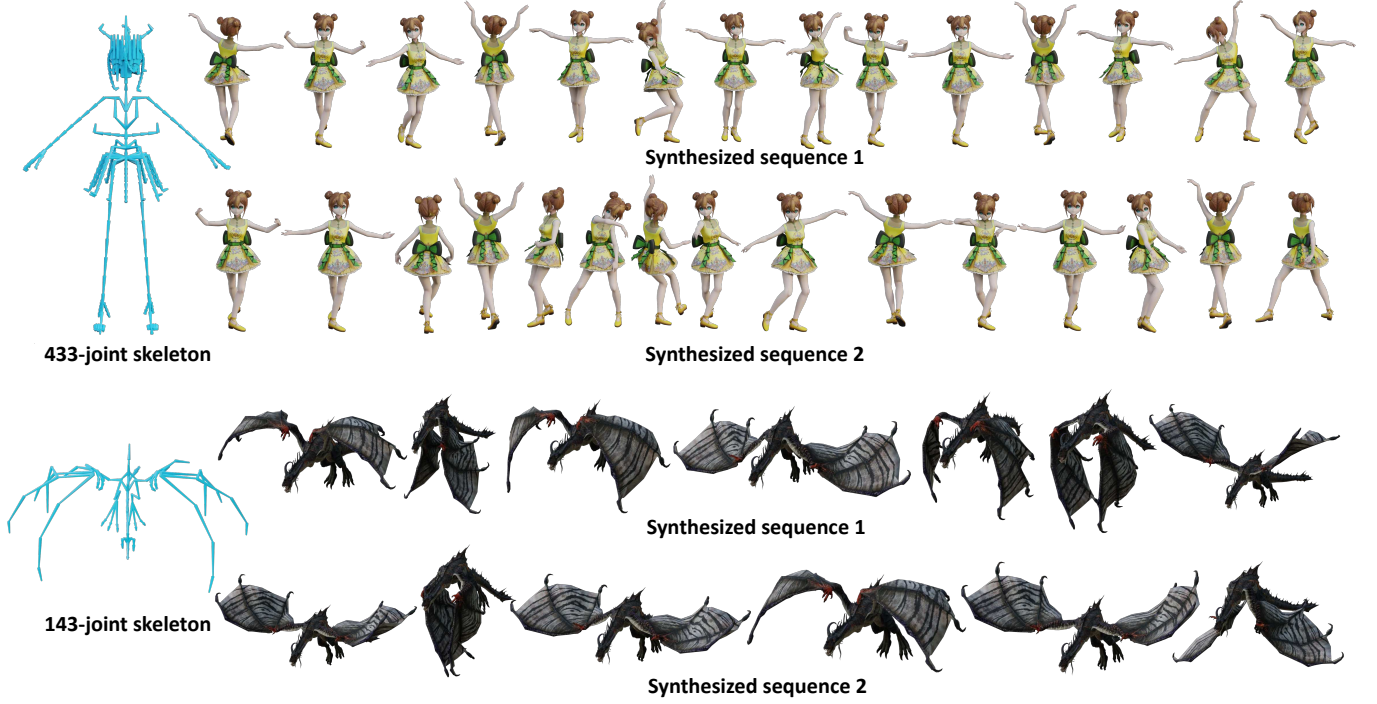


Fig. 5. From a single example, our framework generates, within a second, diverse motion sequences for even highly complex and large skeletons, including the animation of the clothes and the wings. Please refer to the accompanying video for more animation results.

where  $X^b$  and  $Y^b$  denote the set of motion patches extracted from corresponding partial-body motions. Then, the bidirectional similarity as in [Simakov et al. 2008; Granot et al. 2022] is introduced to encourage that all exemplar motion patches appear in the synthesis and all motion patches in the synthesis do not deviate from the example, i.e., high completeness and coherence. This is achieved by normalizing the distance matrices using a per-example-patch factor:

$$\hat{D}_{i,j}^b = \frac{D_{i,j}^b}{(\alpha + \min_{\ell} (D_{\ell,j}^b))}, \quad (2)$$

where  $\alpha$  controls the degree of completeness, and smaller  $\alpha$  encourages completeness. An in-depth study on the effect of  $\alpha$  is conducted in Section 4.3.

**Blending.** For each motion patch in  $X^b$ , we find its nearest (as defined by Equation 2) motion patch in  $Y^b$ , and then blend the values of collected motion patches using average voting, forming a synthesized partial-body motion  $\bar{F}^b$ . Finally, we average the values over overlapping joints between skeletal parts to assemble all synthesized partial-body motions into the final result  $F_s$  (See Figure 4).

### 3.4 Extension to More Settings

Our method can also be easily extended for various settings.

**Skeleton Partition for Motion Patches.** In addition to the skeleton-aware motion patch defined above, our method can also work with

the traditional definition of a motion patch, i.e., treating the skeleton as a whole and then extracting  $p$  consecutive poses.

**Multiple Examples.** As aforementioned, our method can not only be applied to a single motion input but also works with multiple sequences of different numbers of frames. This can be achieved by simply extracting motion patches from all input motions to form the set of exemplar patches used in Equation 1. In this setting, the completeness control knob  $\alpha$  plays a crucial role in ensuring that all motion patches across examples are utilized in the output.

**Heterogeneous Skeletons.** Interestingly, under the setting of multiple examples, the skeletons across different motions do not necessarily share the same one. For example, we can take motion clips of a monster and a zombie, and synthesizes a moving Frankenstein, via harmonizing different partial-body motions extracted from these two creatures. To achieve this, the user can split the skeleton of each input with overlap and manually specify the skeletal parts of interest to be used in the generative motion matching and blending. The overlapping region plays a crucial role in bridging different skeletal parts during the generation process. Then, our method can synthesize the novel motion for the new creature by combining the partial-body motions as discussed in Section 3.3.

## 4 EXPERIMENTS

We evaluate the effectiveness of our method on example-based motion synthesis, compare to other motion generation techniques, and

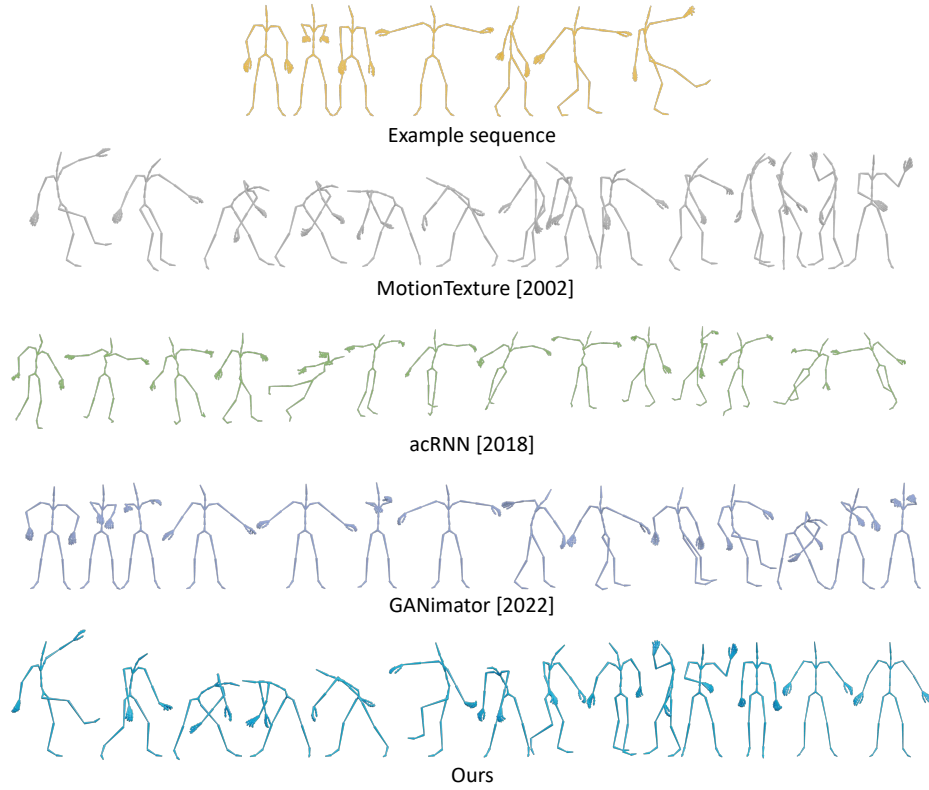


Fig. 6. Visual comparisons. MotionTexture [2002] generates motions with unnatural transitions. acRNN [2018] produces noisy motions or sometimes quickly converges to a static pose. GANimator [2022] struggles to handle complex skeletons, producing over-smoothing results. Our method outperforms these methods with diverse and high-quality results, where highly dynamic motions are well preserved. Please refer to the accompanying video for animation results.

demonstrate its versatility by applying it in various settings and applications. We highly recommend readers refer to the accompanying video for more qualitative evaluations. *Code and data will be released to ease the understanding of our implementation and facilitate future studies.*

**Data.** We collected a diverse set of example animations featuring varied motion styles and highly complex and large skeletal structures from Mixamo [2022] and Truebones [2022]. The motion styles we experimented with include sharp motions of a popping dance, subtle motions of fanning wings, etc. Some examples are authored with highly sophisticated skeletal structures, such as the 433-joint and 143-joint skeletons as visualized in Figure 5. The number of frames ranges from 140 to 1000 frames at 30 fps.

**Implementation Details.** Our framework is lightweight and does not require any training. Due to its simplicity and efficiency, we simply implement our method with Python. We also develop an add-on in the open source software Blender [Blender Online Community 2023], which is ready to take animations from users and synthesizes diverse and high-quality variants. In our implementation, a motion sample with around 1000 frames can be generated in  $\sim 0.2s$  with an Apple M1 CPU or  $\sim 0.05s$  with a modern GPU (NVIDIA V100). By default, we run experiments using an Apple M1 CPU, except

that the comparison experiments are conducted using an NVIDIA V100 GPU for fair comparisons with neural network-based methods. We set the length of  $T_1$  at the coarsest stage to  $K$  times the patch size  $p$ . Thus the receptive field (a similar concept as in the image) always occupies the same proportion of example motions with different lengths. Then,  $T_1$  is gradually upsampled using the factor  $r$  until it reaches the final length of  $T_5$ . Unless otherwise specified, we use a patch size  $p = 11$ ,  $K = 4$ , a completeness control knob  $\alpha = 0.01$  and a number of iterations  $E = 5$ . These are the empirically best hyper-parameters we have found. For more discussions on the hyper-parameters, please refer to the supplementary material.

#### 4.1 Novel Motion Synthesis

We first evaluate the performance of our framework on novel motion synthesis, and compare it to a classical statistical model and recent neural-based models, namely MotionTexture [Li et al. 2002], acRNN [Zhou et al. 2018] and GANimator [Li et al. 2022].

**Settings.** Although our method can use multiple inputs, for fair comparisons, we conduct the evaluation on diverse motion synthesis from a *single* example and disable the skeleton-aware motion patch extraction. We use three example sequences containing highly dynamic and agile movements for evaluation. The character consists

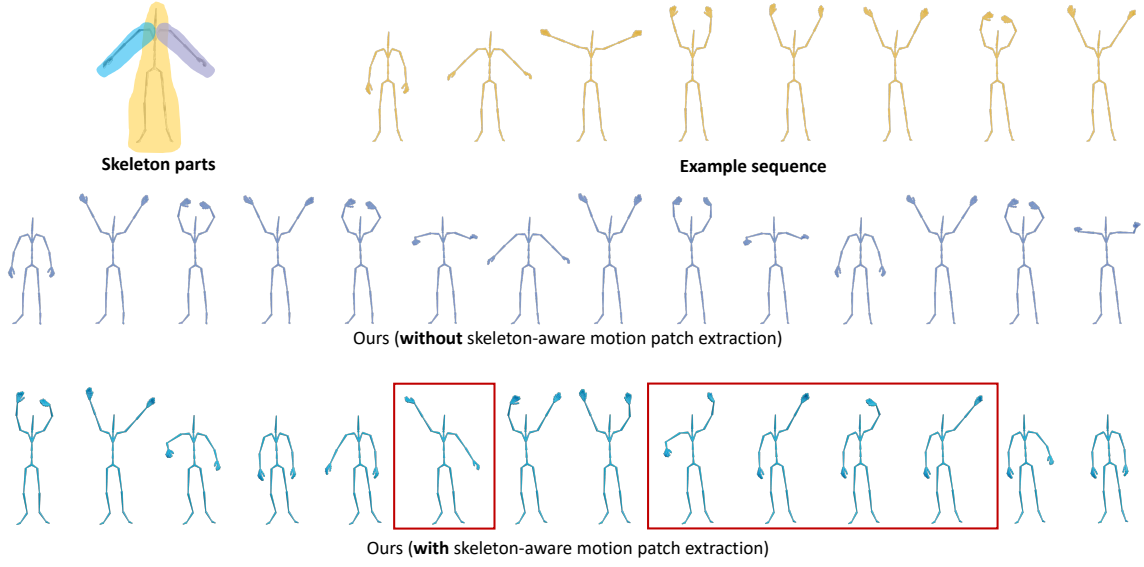


Fig. 7. The effectiveness of skeleton-aware motion patch extraction. An artist manually divides the skeleton into three overlapping parts (top left). Given an example of a character waving two hands simultaneously, only a sequence with two waving hands can be synthesized without the skeleton-aware component. However, with the skeleton-aware motion patch extraction, a more diverse sequence, including waving with only one hand (in red boxes), can be generated.

of 65 joints, and each sequence has around 500 frames. For each example sequence, we use all methods to synthesize a novel sequence that doubles the length of the example.

**Qualitative Comparison.** (i) To accomplish diverse motion synthesis, *MotionTexture* organizes similar motion patches from training motions into linear dynamics models known as textons, and models the probability of transition between textons using a Markov Chain. However, it faces the challenge of balancing diversity and quality, particularly when there is only one example sequence due to the choice of linear dynamics model. We follow the procedure as done in [Li et al. 2022] to apply *MotionTexture* to a single example. As a result, *MotionTexture* produces unnatural transitions between textons. (ii) *acRNN* uses an RNN-based network structure. The lack of data leads to a model that is prone to overfitting and is not robust to perturbation or error accumulation. Consequently, *acRNN* can only stably generate a limited number of frames. (iii) *GANimator* utilizes a series of GANs to capture the distribution of motion patches at different scales, in order to progressively synthesize motions that closely resemble the input. In our experiments with complex and large skeletons, as shown in Table 3, *GANimator* struggles to produce high-quality results, often resulting in jittery or over-smoothed motions. Additionally, it requires a significant amount of training time, typically from several hours to a day. In contrast, our method can adapt to these complex skeletal structures and various motion styles, and synthesize diverse and high-quality variations as shown Figure 6. Notably, highly dynamic motions, in particular sharp and agile movements, are well preserved in our synthesized results. For more qualitative results, please refer to the accompanying video.

Table 1. Quantitative comparisons on single example-based generation.

	Coverage	Set Div.	Global Patch Dist.	Local Patch Dist.	Training Time	Inference Time
<i>MotionTexture</i> [2002]	84.12	0.05	1.12	1.13	32.3s	0.03s
<i>MotionTexture</i> (Single)	100.00	0.01	0.45	0.44	0.08s	0.07s
<i>acRNN</i> [2018]	5.13	0.75	13.62	13.55	25 hrs	0.21s
<i>GANimator</i>	49.07	0.24	2.18	2.08	6 hrs	0.12s
Ours	99.89	0.28	0.22	0.18	N/A	0.08s

**Quantitative Comparison.** Measuring the quality of generated results against a few examples is known to be difficult [Li et al. 2022]. As one of the pioneers, *GANimator* uses a combination of established metrics, namely coverage, diversity, and reconstruction loss, to rate the performance, since a single metric does not suffice the need of measuring the overall quality. However, the reconstruction loss is only suitable as a quality indicator for neural network-based methods. The diversity is measured with the average distance between generated motion patches and their nearest neighbor in the examples, and different motion patch size corresponds to local and global diversity. As a result, they tend to increase if the generated motion becomes unnatural, and favor results with minor perturbation or over-smoothed results generated by neural networks. We thus use neutral names for the diversity metrics, namely *local patch distance* and *global patch distance* in our experiments. We refer readers to [Li et al. 2022] for more details about these metrics. In addition, following the well-established metric in 2D image synthesis [Shaham et al. 2019], we also report the *set diversity*, which measures the diversity among the generated results and is calculated as the averaged standard deviation of the rotation of each joint over 200 synthesized motions and normalized by the standard deviation of



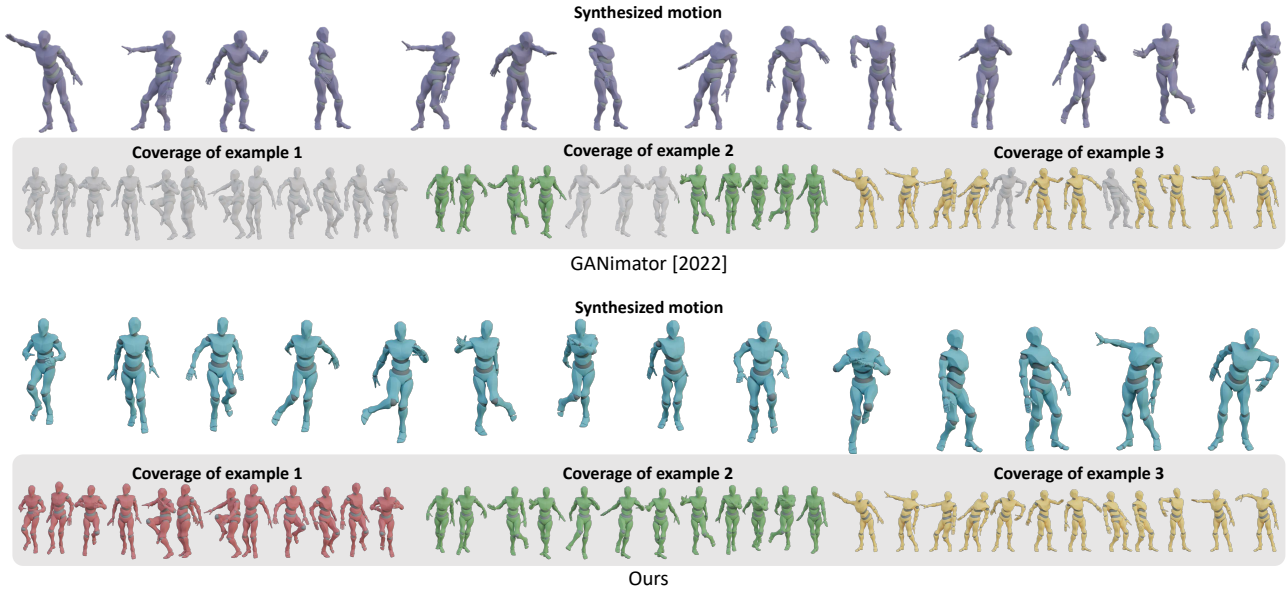


Fig. 8. Comparison under the multi-example setting. GANimator struggles to incorporate all examples, resulting in the loss of a significant portion of exemplar motion patches in the synthesis (marked in gray) and a low coverage score. In contrast, our method effectively covers all examples (marked as colored), resulting in high coverage score.

all joint rotations of the input example. Note that, while this metric also has a preference for noisy output, we mainly rate the methods using the combination of the coverage and set diversity.

The quantitative comparison results are shown in Table 1. Notably, our method produces a significantly high coverage score, while still exhibiting sufficiently diverse results (evidenced by a high set diversity score). For a more comprehensive comparison of the quality, we refer the readers to the accompanying video. Note, we also report the computation time in Table 1, where we can see that our method is highly efficient, as it is both training-free and extremely fast during inference.

#### 4.2 More Generation Settings

In addition to the basic setting used above, we further evaluate our method in the following aspects.

*Skeleton-aware Motion Patches.* In addition to the temporal axis, our method can also extract motion patches from examples along the skeletal axis, thus allowing obtaining diversity also on the spatial dimension as shown in Figure 7 and the accompanying video.

*Multiple Examples.* Unlike existing methods, which struggle when presented with multiple example sequences due to the lack of explicit encouragement of completeness, our method can handle multiple examples with the completeness control knob in Equation 2. We collect five dancing sequences ranging from 120-220 frames at 30 fps. It can be seen in Table 2 that when more examples are given, existing methods generate results with lower coverage while our method remains a high coverage. MotionTexture produces unnatural transitions, similar to its results in the single-example setting. acRNN fails on the task and produces noisy motions due to the diverse but scarce

Table 2. Coverage rates of different numbers of example sequences.

	Number of Example Sequences			
	2	3	4	5
MotionTexture [2002]	100	29.19	10.03	27.69
acRNN [2018]	7.34	4.02	1.38	0.41
GANimator [2022]	63.55	23.90	17.30	16.22
Ours	99.71	99.95	99.91	99.64

Table 3. Coverage rate on skeletons with different complexity.

	Number of Joints		
	24	65	433
GANimator [2022]	92.10	44.10	2.38
Ours	97.82	99.84	86.90

motion data. GANimator requires a corresponding pre-defined latent variable for each sequence. However, the structure of the given sequences is not taken into consideration for defining these latent variables, hindering the network from generating various and complete samples. In contrast, our method produces motions that cover a large portion of the examples with a properly set completeness control knob as shown in Figure 8 and the accompanying video.

*Complex Skeletons.* Our method can work with skeletons of high complexities (See Figure 5), on which the GAN-base method GANimator fails to produce reasonable results as demonstrated in the accompanying video. Specifically, we experiment with skeletons consisting of 24, 65, and 433 joints. It can be seen in Table 3 that GANimator performs normally on the 24-joint skeletons, while its performance drops dramatically when presented with complex

Table 4. The different settings of hyperparameters.

	Coverage	Set Diversity	Global Patch Dist.	Local Patch Dist.
Ours (not use $\alpha$ )	87.45	0.27	0.18	0.17
Ours ( $\alpha = 5$ )	87.89	0.27	0.18	0.17
Ours ( $\alpha = 0.5$ )	88.47	0.27	0.18	0.17
Ours ( $\alpha = 0.05$ )	93.80	0.27	0.18	0.17
Ours ( $\alpha = 0.005$ )	99.96	0.27	0.30	0.23
Ours ( $\alpha = 0.0$ )	36.78	0.25	2.17	1.60
Ours ( $K = 20$ )	87.74	0.25	1.03	0.71
Ours ( $K = 15$ )	92.04	0.26	0.73	0.54
Ours ( $K = 10$ )	96.07	0.27	0.42	0.36
Ours ( $p = 23$ )	99.85	0.26	0.27	0.24
Ours ( $p = 17$ )	99.88	0.26	0.23	0.21
Ours ( $p = 5$ )	99.66	0.27	0.41	0.32
Ours ( $r = 2$ )	99.61	0.27	0.22	0.20
Ours ( $r = 4$ )	99.16	0.27	0.34	0.28
Ours ( $r = 8$ )	97.97	0.26	0.57	0.42

skeletons. Whereas our method maintains a consistent performance for different skeletons, evidenced by the fluttering effects of the skirt and dragon wings in the accompanying video.

#### 4.3 Effects of Hyper-parameters

Our framework involves several hyper-parameters during the synthesis process. In this section, we discuss the effects of these hyper-parameters. The quantitative results are presented in Table 4.

*Effects of  $\alpha$ .* As rarely-appearing patches have a larger minimal distance, the completeness of the synthesis is encouraged by normalizing the distance of patches extracted from examples with their minimal distance to the initial guess. Therefore, the  $\alpha$  in Equation 2 serves as a control knob for the completeness of exemplar patches in the synthesized result. As it restricts the lower bound of the normalizing denominator, a smaller  $\alpha$  value encourages more preservation of the example content in the synthesis. As shown in Table 4, when  $\alpha$  decreases to a certain level, a higher coverage score is achieved. However, if the value  $\alpha$  is too small, an excessive emphasis on completeness (especially for patches with almost zero distance to the generated motion) can overwhelm the similarity measure used for the matching process, resulting in unstable generation and low-quality motion (evidenced by low coverage and high patch distances of the corrupted results)."

*Effects of  $K$ .* The ratio of the patch size to the length of input example motion at the coarsest stage controls the receptive field for synthesis, similar to the concept in image domain. A larger  $K$  causes a smaller receptive field, leading to more diverse results. In particular, a large  $K$  allows only capturing fine-level movements and leads to some unnatural transitions, while a small  $K$  leads to overfitting of the original sequence. Table 4 shows the global and local patch distance increase as  $K$  increases. This is because the generated result deviates further from the input sequence when the receptive field is smaller.

*Effects of Patch Size  $p$ .* The patch size  $p$  defines the temporal length of patches used in the generative matching and blending. Patch size controls the receptive field jointly with  $K$ , and a smaller patch size

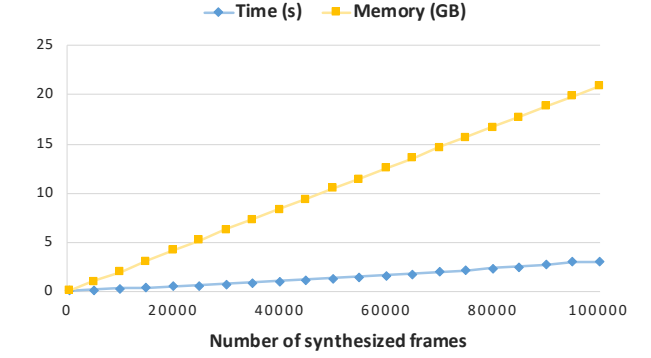


Fig. 9. Time and memory consumption with respect to increasing numbers of generated frames.

leads to a smaller receptive field, which creates less coherent result as shown by the increase of global and local patch distance in Table 4.

*Effects of  $r$ .* The factor  $r$  controls the step size of transition between stages. A large step size, controlled by a large  $r$ , may result in unstable generation due to big gaps between consecutive scales. On the other hand, a small step size causes unnecessary running time.

#### 4.4 Time and Memory Consumption

The memory footprint of the distance metrics described in Section 3.3 increases as the number of generated frames,  $F$ , grows. To further investigate the time and memory consumption, we stress-test our method under extreme conditions, where  $E_N$  comprises 522 motion frames of a 65-joint character and we set  $F$  ranging from 1,000 to 100,000. These tests are conducted using an NVIDIA V100 GPU equipped with 32GB memory. Figure 9 illustrates that both time and memory consumption exhibit a linear growth pattern with respect to the number of generated frames. Owing to the highly paralleled computation of the distance metrics in the GPU, our method takes only around 3 seconds to synthesize a high-quality sample even consisting of 100,000 frames.

### 5 APPLICATIONS

In this section, we demonstrate the versatility of our framework by adapting it to various applications, such as motion completion, key frame-guided generation, infinite looping, and motion reassembly. The results are presented in Figure 10. A more detailed demonstration is available in the accompanying video.

*Motion Completion.* Our framework, which utilizes skeleton-aware motion patch extraction, enables the completion of partial motions that contain only the movement of specific body parts. For example, when a lower-body motion sequence  $\mathbf{T}^{\text{lower}}$  is provided, the upper-body motion can be completed using the example motion. Specifically, we build a pyramid for the partial-body motion  $\{\mathbf{T}_1^{\text{lower}}, \dots, \mathbf{T}_S^{\text{lower}}\}$ , and the corresponding partial motion in the output  $\mathbf{F}_s$  is fixed to  $\mathbf{T}_s^{\text{lower}}$  at each stage  $s$ . Our framework then automatically synthesizes the movements of the rest by parts, completing the partial constraints with a coherent and natural motion.



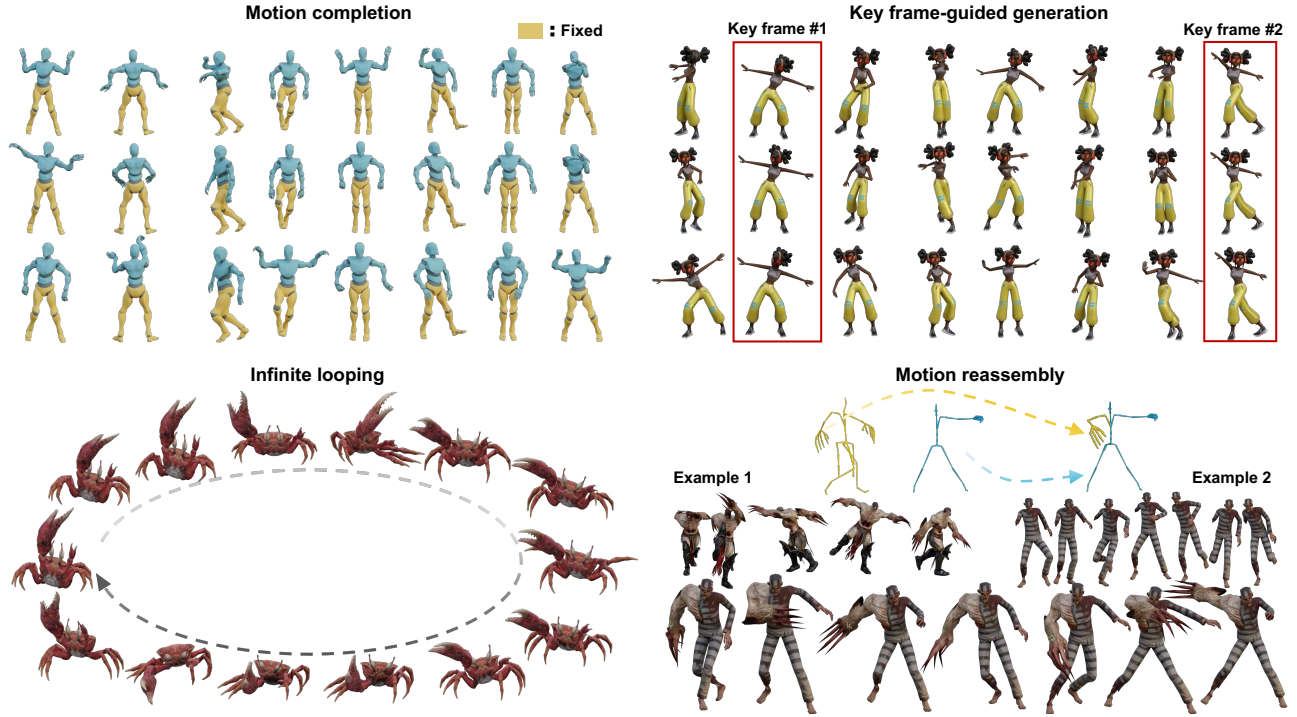


Fig. 10. Applications. (1) Motion completion. Users can provide the lower-body motion (marked in yellow), and our method completes with diverse motions. (2) Key frame-guided generation. Given a set of key frames (marked in red boxes), we can generate diverse novel motion sequence that follow the key frame poses. (3) Infinite looping. By simply specifying the starting and ending pose to be identical, our method can generate a infinitely looping animation, which can be useful in crowd simulation. (4) Motion reassembly. Given two motion sequences with heterogeneous skeletons, our method can combine them to form a new creature with coherent and natural motion.

**Key Frame-guided Generation.** Our method also allows users to manually specify a sparse set of key frames to guide the content of the synthesized motion. Our method can then effectively handle these sparse pose constraints distributed throughout the sequence and generate smooth, highly-detailed motion. Given a set of key frames at the coarsest stage, we simply realize it by replacing corresponding frames in  $F_1$  with the specified frames, and fixing them through the whole generation process. Note that these manually specified key frames should not deviate significantly from the distribution of the poses in the example. In practice, they can be obtained by simply selecting existing poses in the example, possibly with slight manual modifications by the user.

**Infinite Looping.** Our framework can easily synthesize endless looping motion by fixing the ending pose to be identical to the beginning pose at every stage in the synthesis. This allows for the seamless looping of the entire motion sequence. It can be useful in animation production, such as creating repetitive crowds like spectators cheering outside an arena.

**Motion Reassembly.** As aforementioned in Section 3.4, our method has the ability to synthesize a Frankenstein. We demonstrate an example that stitches the right arm of a monster to a zombie; See Figure 10 and the accompanying video. Note the example sequence of these two characters is different and the zombie does not have any

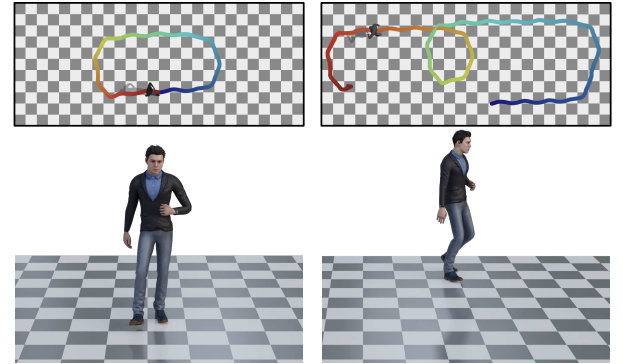


Fig. 11. Random locomotion generation. Given an example locomotion sequence of a character walking in a circular path (left), we show a high-quality novel motion sequence generated by our method, in which the character walks along a different trajectory (right).

movement in its partially missing right arm, yet our method is still able to successfully synthesize a natural and meaningful motion.

**Random Locomotion Generation.** Our method can also generate high-quality novel motion sequences when given a locomotion clip. As can be seen in Figure 11, while the example sequence contains a person walking in a circular path, our method can generate novel

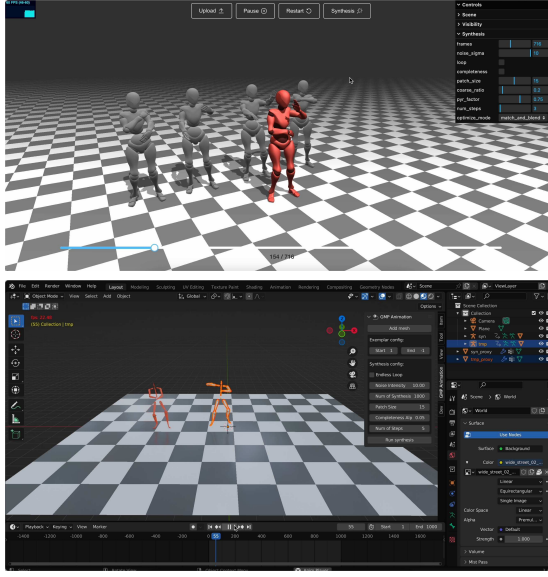


Fig. 12. Top: screenshot of our web-based interface, characters in grey denote synthesized motions. Bottom: screenshot of our Blender add-on, the synthesized motion is highlighted in the middle.

outputs with a different trajectory (See the difference between the corresponding trajectories at the top row). More animation results can be found in the accompanying video.

## 6 USER INTERFACE

Our framework is general, lightweight, and easy to integrate into many production tools. For novice users, we build a user-friendly website where users can upload their motion files and then synthesize diverse novel motions with a single click (See the top in Figure 12). We also develop a Blender add-on for professional artists, which seamlessly integrates into their existing workflow as demonstrated at the bottom of Figure 12. Note both interfaces can run efficiently on a consumer-level laptop. Please refer to the accompanying video for the results.

## 7 DISCUSSION AND CONCLUSION

We presented a generative framework for synthesizing diverse motion sequences from only a small set of examples. We achieve this via injecting generative capabilities into the industry state-of-the-art technique for character animation – motion matching. As a result, our framework inherits the training-free nature and superior quality, and is able to produce a high-quality sample within just a fraction of a second, even with highly complex and large skeletons. We demonstrate the utility of our framework on a variety of applications. Despite its advantages, our method in its current form has a few shortcomings: It uses a discrete patch distribution, whereas GANimator [Li et al. 2022] learns a continuous distribution. Therefore, GANimator can generate novel poses with high likelihood from the learned distribution. Although the skeleton-aware component can be a remedy, this capability is missing in our method. Nevertheless, we argue that such generalization can be disadvantageous in motion synthesis, as sequences formed by novel poses often contain

visual artifacts such as jittering and incoherence, which are highly noticeable to human eyes. We prioritized the motion quality at the outset, which led us to the motion matching approach.

Our method seeks to synthesize as many variants as can be mined from the examples, rather than struggle to balance quality with novelty of motion. As a consequence, although diverse results of our method are shown, the generative diversity of our method is lower than that of GANimator. Hence, a future work direction is to inject the high quality of motion matching into generative neural models, possibly with discrete neural representation learning techniques [Van Den Oord et al. 2017], and thus obtain the best of both worlds.

Our method favors example motions with sufficient intrinsic periodicity, which has been increasingly recognized as an important property of common human motion [Holden et al. 2017; Starke et al. 2022], to generate highly diverse novel variations. It seeks to exploit such patterns in a single example for mining as many coherent variations as possible. In extreme cases where the example only involves a single pose change, it may be meaningless to create temporal variations based solely on such input. Nonetheless, our skeleton-aware component may introduce variations along the skeletal axis, as evidenced by the asynchronized waving hands in the supplementary video.

Regarding the manual constraints required in the key frame-guided application, the manually specified key frames cannot differ significantly from those example poses as aforementioned, otherwise the generated sequence may not faithfully follow those constraining poses due to the lack of ability to generate completely novel poses as discussed above.

Last, our method cannot deal with overly long example sequences, as the normalized similarity matrices grow excessively large. Adopting approximate nearest neighbors search, such as [Barnes et al. 2009], may help alleviate this issue.

## ACKNOWLEDGMENTS

We thank the anonymous reviewers for their constructive comments. This work was supported in part by National Key R&D Program of China 2022ZD0160801, and the European Research Council (ERC) under the European Union’s Horizon 2020 Research and Innovation Programme (ERC Consolidator Grant, agreement No. 101003104, MYCLOTH). We would also like to thank Han Liu from Tencent AI Lab for providing the motion data of the avatar Ailing (Figure 1).

## REFERENCES

- Adobe Systems Inc. 2022. Mixamo. <https://www.mixamo.com> Accessed: 2022-03-25.
- Okan Arikan and David A Forsyth. 2002. Interactive motion generation from examples. *ACM Transactions on Graphics (TOG)* 21, 3 (2002), 483–490.
- Connelly Barnes, Eli Shechtman, Adam Finkelstein, and Dan B Goldman. 2009. Patch-Match: A randomized correspondence algorithm for structural image editing. *ACM Transactions on Graphics (TOG)* 28, 3 (2009), 24.
- Connelly Barnes and Fang-Lue Zhang. 2017. A survey of the state-of-the-art in patch-based synthesis. *Computational Visual Media* 3, 1 (2017), 3–20.
- Blender Online Community. 2023. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam.
- Richard Bowden. 2000. Learning statistical models of human motion. In *IEEE Workshop on Human Modeling, Analysis and Synthesis, CVPR*, Vol. 2000. Citeseer.
- Matthew Brand and Aaron Hertzmann. 2000. Style machines. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 183–192.
- Peter J Burt and Edward H Adelson. 1987. The Laplacian pyramid as a compact image code. In *Readings in computer vision*. Elsevier, 671–679.

- Michael Buttner. 2019. Machine learning for motion synthesis and character control in games. *Proc. of I3D 2019* (2019).
- Michael Büttner and Simon Clavet. 2015. Motion Matching - The Road to Next Gen Animation. [https://www.youtube.com/watch?v=z\\_wpgHFSWss&t=658s](https://www.youtube.com/watch?v=z_wpgHFSWss&t=658s)
- Jinxiang Chai and Jessica K Hodgins. 2007. Constraint-based motion optimization using a statistical dynamic model. In *ACM SIGGRAPH 2007 papers*. 8–es.
- Jeremy S De Bonet. 1997. Multiresolution sampling procedure for analysis and synthesis of texture images. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*. 361–368.
- Yinglin Duan, Yue Lin, Zhengxia Zou, Yi Yuan, Zhehui Qian, and Bohan Zhang. 2022. A Unified Framework for Real Time Motion Completion. (2022).
- Katerina Fragkiadaki, Sergey Levine, Panna Felsen, and Jitendra Malik. 2015. Recurrent network models for human dynamics. In *Proceedings of the IEEE International Conference on Computer Vision*. 4346–4354.
- Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *NIPS*.
- Niv Granot, Ben Feinstein, Assaf Shocher, Shai Bagon, and Michal Irani. 2022. Drop the gan: In defense of patches nearest neighbors as single image generative models. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 13460–13469.
- Keith Grochow, Steven L Martin, Aaron Hertzmann, and Zoran Popović. 2004. Style-based inverse kinematics. In *ACM SIGGRAPH 2004 Papers*. 522–531.
- Ikhshanul Habibie, Mohamed Elgharib, Kripasindhu Sarkar, Ahsan Abdullah, Simbarashe Nyatsanga, Michael Neff, and Christian Theobalt. 2022. A Motion Matching-based Framework for Controllable Gesture Synthesis from Speech. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–9.
- Charles Han, Eric Risser, Ravi Ramamoorthi, and Eitan Grinspun. 2008. Multiscale texture synthesis. In *ACM SIGGRAPH 2008 papers*. 1–8.
- Geof Harrower. 2018. Real player motion tech in 'ea sports ufc 3'. *Proc. of GDC 2018* (2018).
- Félix G Harvey, Mike Yurick, Derek Nowrouzezahrai, and Christopher Pal. 2020. Robust motion in-betweening. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 60–1.
- Chengnan He, Jun Saito, James Zachary, Holly Rushmeier, and Yi Zhou. 2022. NeMF: Neural Motion Fields for Kinematic Animation. In *Advances in Neural Information Processing Systems*.
- David J Heeger and James R Bergen. 1995. Pyramid-based texture analysis/synthesis. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. 229–238.
- Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. 2020. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Transactions on Graphics (TOG)* 39, 6 (2020), 1–14.
- Daniel Holden, Oussama Kanoun, Maksym Perepichka, and Tiberiu Popa. 2020. Learned motion matching. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 53–1.
- Daniel Holden, Taku Komura, and Jun Saito. 2017. Phase-functioned neural networks for character control. *ACM Transactions on Graphics (TOG)* 36, 4 (2017), 1–13.
- Daniel Holden, Jun Saito, and Taku Komura. 2016. A deep learning framework for character motion synthesis and editing. *ACM Transactions on Graphics (TOG)* 35, 4 (2016), 1–11.
- Daniel Holden, Jun Saito, Taku Komura, and Thomas Joyce. 2015. Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 technical briefs*. 1–4.
- Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).
- Deok-Kyeong Jang, Soomin Park, and Sung-Hee Lee. 2022. Motion Puzzle: Arbitrary Motion Style Transfer by Body Part. *ACM Transactions on Graphics (TOG)* (2022).
- Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive Growing of GANs for Improved Quality, Stability, and Variation. In *International Conference on Learning Representations*.
- Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2002. Motion Graphs. In *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques* (San Antonio, Texas) (*SIGGRAPH '02*). Association for Computing Machinery, New York, NY, USA, 473–482. <https://doi.org/10.1145/566570.566605>
- Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. 2002. Interactive control of avatars animated with human motion data. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 491–500.
- Seyoung Lee, Jiye Lee, and Jehee Lee. 2022. Learning Virtual Chimeras by Dynamic Motion Reassembly. *ACM Trans. Graph.* 41, 6, Article 182 (2022).
- Yongjoon Lee, Kevin Wampler, Gilbert Bernstein, Jovan Popović, and Zoran Popović. 2010. Motion fields for interactive character locomotion. In *ACM Transactions on Graphics (TOG)*. 1–8.
- Sergey Levine, Jack M Wang, Alexis Harauz, Zoran Popović, and Vladlen Koltun. 2012. Continuous character control with low-dimensional embeddings. *ACM Transactions on Graphics (TOG)* 31, 4 (2012), 1–10.
- Peizhuo Li, Kfir Aberman, Zihan Zhang, Rana Hanocka, and Olga Sorkine-Hornung. 2022. GANimator: Neural Motion Synthesis from a Single Sequence. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 138.
- Yan Li, Tianshu Wang, and Heung-Yeung Shum. 2002. Motion texture: a two-level statistical model for character motion synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 465–472.
- Dario Pavlo, David Grangier, and Michael Auli. 2018. Quaternet: A quaternion-based recurrent model for human motion. *arXiv preprint arXiv:1805.06485* (2018).
- Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. 2018. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics (TOG)* 37, 4 (2018), 1–14.
- Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. 2021. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics (TOG)* 40, 4 (2021), 1–20.
- Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.
- Ken Perlin and Athomas Goldberg. 1996. Improv: A system for scripting interactive actors in virtual worlds. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*. 205–216.
- Katherine Pullen and Christoph Bregler. 2000. Animating by multi-level sampling. In *Proceedings Computer Animation 2000*. IEEE, 36–42.
- Katherine Pullen and Christoph Bregler. 2002. Motion capture assisted animation: Texturing and synthesis. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*. 501–508.
- Jia Qin, Youyi Zheng, and Kun Zhou. 2022. Motion In-betweening via Two-stage Transformers. *ACM Transactions on Graphics (TOG)* 41, 6 (2022), 1–16.
- Sigal Raab, Inbal Leibovitch, Peizhuo Li, Kfir Aberman, Olga Sorkine-Hornung, and Daniel Cohen-Or. 2023a. MoDi: Unconditional Motion Synthesis from Diverse Data. (2023).
- Sigal Raab, Inbal Leibovitch, Guy Tevet, Moab Arar, Amit H Bermanto, and Daniel Cohen-Or. 2023b. Single Motion Diffusion. *arXiv preprint arXiv:2302.05905* (2023).
- Davis Remppe, Tolga Birdal, Aaron Hertzmann, Jimei Yang, Srinath Sridhar, and Leonidas J Guibas. 2021. Humor: 3d human motion model for robust pose estimation. In *Proceedings of the IEEE/CVF international conference on computer vision*. 11488–11499.
- Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. 2019. Singan: Learning a generative model from a single natural image. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 4570–4580.
- Mingyi Shi, Kfir Aberman, Andreas Aristidou, Taku Komura, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2020. Motionet: 3d human motion reconstruction from monocular video with skeleton consistency. *ACM Transactions on Graphics (TOG)* 40, 1 (2020), 1–15.
- Denis Simakov, Yaron Caspi, Eli Shechtman, and Michal Irani. 2008. Summarizing visual data using bidirectional similarity. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 1–8.
- Sebastian Starke, Ian Mason, and Taku Komura. 2022. Deepphase: Periodic autoencoders for learning motion phase manifolds. *ACM Transactions on Graphics (TOG)* 41, 4 (2022), 1–13.
- Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermanto, and Daniel Cohen-Or. 2022a. MotionCLIP: Exposing Human Motion Generation to CLIP Space. *arXiv preprint arXiv:2203.08063* (2022).
- Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermanto. 2022b. Human motion diffusion model. *arXiv preprint arXiv:2209.14916* (2022).
- Truebones Motions Animation Studios. 2022. Truebones. <https://truebones.gumroad.com/> Accessed: 2022-9-2.
- Jonathan Tseng, Rodrigo Castellon, and C Karen Liu. 2022. EDGE: Editable Dance Generation From Music. *arXiv preprint arXiv:2211.10658* (2022).
- Aaron Van Den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems* 30 (2017).
- Jack M Wang, David J Fleet, and Aaron Hertzmann. 2007. Gaussian process dynamical models for human motion. *IEEE transactions on pattern analysis and machine intelligence* 30, 2 (2007), 283–298.
- Li-Yi Wei, Sylvain Lefebvre, Vivek Kwatra, and Greg Turk. 2009. State of the art in example-based texture synthesis. *Eurographics 2009, State of the Art Report, EG-STAR* (2009), 93–117.
- Li-Yi Wei and Marc Levoy. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 479–488.
- Mengyi Zhao, Mengyuan Liu, Bin Ren, Shuling Dai, and Nicu Sebe. 2023. Modiff: Action-Conditioned 3D Motion Generation with Denoising Diffusion Probabilistic Models. *arXiv preprint arXiv:2301.03949* (2023).
- Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. 2019. On the continuity of rotation representations in neural networks. In *Conference on Computer Vision and Pattern Recognition (CVPR)*. 5745–5753.
- Yi Zhou, Zimo Li, Shuangjiu Xiao, Chong He, Zeng Huang, and Hao Li. 2018. Auto-conditioned recurrent networks for extended complex human motion synthesis. In *International Conference on Learning Representations*.