Iterative Motion Editing with Natural Language

Purvi Goel
Stanford University
USA
pgoel2@cs.stanford.edu

Kuan-Chieh Wang
Snap, Inc.
USA
jwang23@snapchat.com

C. Karen Liu
Stanford University
USA
karenliu@cs.stanford.edu

Kayvon Fatahalian
Stanford University
USA
kayvonf@cs.stanford.edu

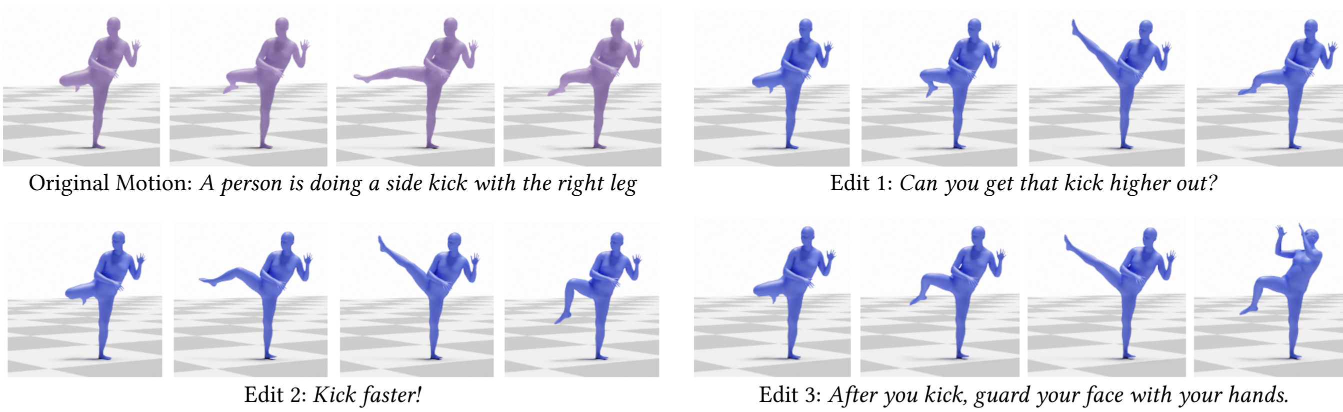


Figure 1: Our system supports iterative refinement of character motion using natural language. Here, the user has a vision for modifying an original kicking motion (top, left). Through a sequence of prompts, the user "coaches" the character to better match their artistic vision, progressively refining the motion by adjusting kinematic details. First, the user requests the character to kick higher (edit 1), and then decides the kick should also be faster (edit 2). Finally, the user has the character raise its hands in anticipation of a return attack (edit 3). Edited motions largely preserve the structure of the original motion while complying with the provided instructions. Retained conversation history allows the system to build upon previous edits.

ABSTRACT

Text-to-motion diffusion models can generate realistic animations from text prompts, but do not support fine-grained motion editing controls. In this paper, we present a method for using natural language to iteratively specify local edits to existing character animations, a task that is common in most computer animation workflows. Our key idea is to represent a space of motion edits using a set of kinematic motion editing operators (MEOs) whose effects on the source motion is well-aligned with user expectations. We provide an algorithm that leverages pre-existing language models to translate textual descriptions of motion edits into source code for programs that define and execute sequences of MEOs on a source animation. We execute MEOs by first translating them into keyframe constraints, and then use diffusion-based motion models to generate output motions that respect these constraints. Through a user study and quantitative evaluation, we demonstrate that our system can perform motion edits that respect the animator's editing intent, remain faithful to the original animation (it edits the original animation, but does not dramatically change it), and yield realistic character animation results.

CCS CONCEPTS

• **Computing methodologies → Animation**.

KEYWORDS

Character animation, motion editing, large language models, motion diffusion.

ACM Reference Format:
Purvi Goel, Kuan-Chieh Wang, C. Karen Liu, and Kayvon Fatahalian. 2024. Iterative Motion Editing with Natural Language. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers*

SIGGRAPH Conference Papers '24, July 27-August 1, 2024, Denver, CO, USA

ACM ISBN 979-8-4007-0525-0/24/07…$15.00
https://doi.org/10.1145/3641519.3657447

'24 (SIGGRAPH Conference Papers '24), July 27-August 1, 2024, Denver, CO, USA. ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3641519.3657447

1 INTRODUCTION

A common task in character animation workflow is to edit existing animation sequences to match a particular creative vision. For example, an animator might start with a martial arts kicking sequence downloaded from a stock animation library (or generated by a modern generative model, or estimated from video using recent human motion reconstruction techniques), and wish to make the character kick higher or raise their arms to guard their face during the kick.

In most traditional animation systems, performing these precise edits requires tedious keyframing of joints. Conversely, emerging systems based on generative text-to-motion models require only modifying an input prompt, e.g., changing the prompt "a side kick" to "a high side kick". However, it can be hard to predict how these systems will interpret a prompt, and animators have no guarantee the modified motion will retain any correspondence with the original. Some recent generative models offer limited editing control, but require additional multi-modal input, such as dense target joint trajectories or editing masks [Guo et al. 2023; Karunratanakul et al. 2023; Shafir et al. 2023], rather than a simple text-based interface.

In this paper, we seek the best of both worlds: precise editing control, delivered via an accessible text-based interface. As shown in Figure 1, given an existing character animation sequence, we hope to allow an animator to use natural language to *engage in a conversation* with the animation system, specifying precise edits that control changes to the character's motion at specific points in the animation ("after you kick, guard your face with your hands"). By iteratively refining the motion over the course of the conversation, we aim to allow an animator to produce a modified character animation that matches their artistic vision.

An important principle underlying the design of effective creative tools is predictability [Agrawala 2023]. An animator should be able to build a mental model of what a system will do in response to a control input. Inspired by this principle, our key idea is to constrain the space of animation edits to a small set of kinematic motion editing operators (MEOs) that are sufficiently simple that their effects on the source motion are well-aligned with user expectations. For example, MEOs can express constraints on a pose at a particular time ("wrist joint in front of head") or specify that a segment of motion should be slowed down or sped up. As a result, MEOs can be robustly translated into low-level joint edits that yield realistic output. Further, the edited motion is likely to be consistent with user expectations. At the same time, MEOs raise the level of abstraction of motion editing commands from keyframes to calls to programmatic operations. This makes it feasible to leverage LLM-based program synthesis techniques to automatically translate a natural language prompt, which may contain imprecise or ambiguous motion editing descriptions, into an executable program that makes API calls to create valid sequences of MEOs. Specifically we make the following contributions:

- We propose a set of **kinematic motion editing operators** (MEOs) that express fine-grained control similar to keyframes, but present a higher level of editing abstraction by modeling edits as spatial and temporal constraints expressed relative to poses (or events) in the source motion (e.g., a hand *above* a head, or a pose-change *after* a foot contact). MEOs serve as a useful intermediate representation for bridging high-level motion editing intent and low-level motion editing operations in an iterative editing context.
- We provide an algorithm, based on using LLMs for program synthesis, that translates natural language motion editing directions into *Python programs* that consist of MEOs.
- We provide an algorithm for applying motion edits described by MEOs to a source character animation sequence. Our approach translates MEOs into keyframes that constrain the output motion, and leverages diffusion-based generative motion models to modify the source motion to adhere to these constraints while maintaining realistic human motion.

Through qualitative and quantitative evaluation and a user study we demonstrate that our system provides an intuitive natural language interface for iterative character motion editing. Our system supports a range of motion edits, and produces motions that are visually realistic, respect the intent of the user, and preserve the fundamental structure of the original motion. Code and data for this paper can be found on our project webpage.

2 OVERVIEW AND DESIGN GOALS

Our goal is to support iterative editing of an existing character animation sequence. Our system takes as input a starting motion $\mathbf{X}_\mathrm{S}$ (the character's root position and joint angles for each frame in the sequence), a plain-text description of that motion (E_ctx), and a plain-text editing instruction E. It generates an edited motion $\mathbf{X}_\mathrm{E}$ adhering to the following desiderata:

(1) *High edit fidelity.* $\mathbf{X}_\mathrm{E}$ should reflect the intended edit E. For example, if the edit is "after you kick, guard your face with your hands" (Figure 1), the character's hands should be up after the kick in the animation $\mathbf{X}_\mathrm{E}$ (but not at the beginning).
(2) *Non destructive.* $\mathbf{X}_\mathrm{E}$ should minimally change aspects of the motion that should not be impacted by E. For example, in the above example of a kick, ideally the character's kick would be minimally impacted by raising the hands.
(3) *Realistic.* $\mathbf{X}_\mathrm{E}$ should be globally harmonized, meaning that the result should be a plausible character motion.

Our goals can conflict in complex ways. For example, to preserve realism when adding a higher kick, it might be necessary to add additional transitional movement, running against the goal of being as non-destructive to the original sequence as possible.

After producing $\mathbf{X}_\mathrm{E}$, if the user has not achieved the desired motion, they may continue to iterate, repeating the process using a prior $\mathbf{X}_\mathrm{E}$ as $\mathbf{X}_\mathrm{S}$, and providing a new E. For the example in Figure 1, the iterative process involves editing the last generated $\mathbf{X}_\mathrm{E}$; for other editing scenarios, it may also involve the user backing up to a prior step in the session and modifying that motion instead.

3 RELATED WORK

Human motion editing is a well-studied and challenging problem. Early work explored modifying motions with spacetime constraints [Gleicher 1997, 2001; Lee and Shin 1999], but producing realistic, coordinated motion edits typically requires the user to manually provide dense constraints. In recent years, deep learning (DL) has explored automating the process of motion editing with sparser user input. Key approaches include motion stylization [Aberman et al. 2020; Yin et al. 2023], pose editing [Oreshkin et al. 2022], and in-betweening [Qin et al. 2022; Shafir et al. 2023; Tevet et al. 2023; Tseng et al. 2022]. However, the above approaches do not use text-based control. Works like [Delmas et al. 2023] keep the natural-language interface we desire, but do not extend to motion.

Programmatic representations of human motion are a long-standing way to summarize motion sequences, such as with smaller clips or motifs [Aristidou et al. 2018; Kovar et al. 2023], learned concepts [Endo et al. 2023], or combinations of primitives [Kulal et al. 2021]. Unlike these works, we propose an intermediate representation (IR) that is specific to motion editing. In that vein, our representation is similar to that of [Ha and Liu 2015], which introduces an IR to edit dynamic controllers in a physics-based setting, but we focus our IR on text-driven *kinematic* motion edits.

Plans as programs is a strategy that represents problem solving plans as code. Our design of MEOs is inspired by recent systems that perform advanced reasoning about visual environments by using LLMs to translate high-level, plain-text problem descriptions into *executable programs* that make calls to a pre-defined, domain-specific API to carry out a precise reasoning strategy. This strategy has been used for task planning in virtual environments [Huang et al. 2022; Liang et al. 2023; Singh et al. 2023; Wang et al. 2023b] and for visual question answering [Surís et al. 2023]. By representing plans as executable code, these approaches simultaneously leverage the common sense reasoning and program synthesis capabilities of LLMs, and ground plans in the environment by providing APIs for the resulting programs to query for environment-specific information (e.g., nearby objects). We follow a similar design for expressing high-level motion editing intent and grounding edits in a target animation sequence.

Motion diffusion models can generate high-quality 3D motion from text [Ren et al. 2023; Tevet et al. 2023; Zhang et al. 2022, 2023]. In addition to generation, motion diffusion models can learn a strong prior for tasks like constrained trajectory infilling [Li et al. 2023; Rempe et al. 2023], motion reconstruction from video [Jiang et al. 2024], and multi-person reconstruction [Müller et al. 2023]. The prior makes the produced motions more realistic and plausible. Our work incorporates this strong motion prior to ensure the realism of the edited motion. While there is a plethora of diffusion-based editing methods in the image domain [Brooks et al. 2023; Hertz et al. 2022; Meng et al. 2022; Sarukkai et al. 2023], they are not applicable as they depend on the architecture of the diffusion model. Specifically, they rely on manipulation of the cross attention layers which interface the input text encoder. The widely used text-conditioned MDM [Tevet et al. 2023] does not have this architecture. For the task of motion editing, work like [Shafir et al. 2023] modifies MDM's motion inpainting process to control and edit end-effector trajectories, but requires dense multi-modal input (e.g, the entire joint trajectory). [Karunratanakul et al. 2023] similarly supports edits to motion trajectories, but does not support fine-grained edits nor a text-based interface for specifying corrections.

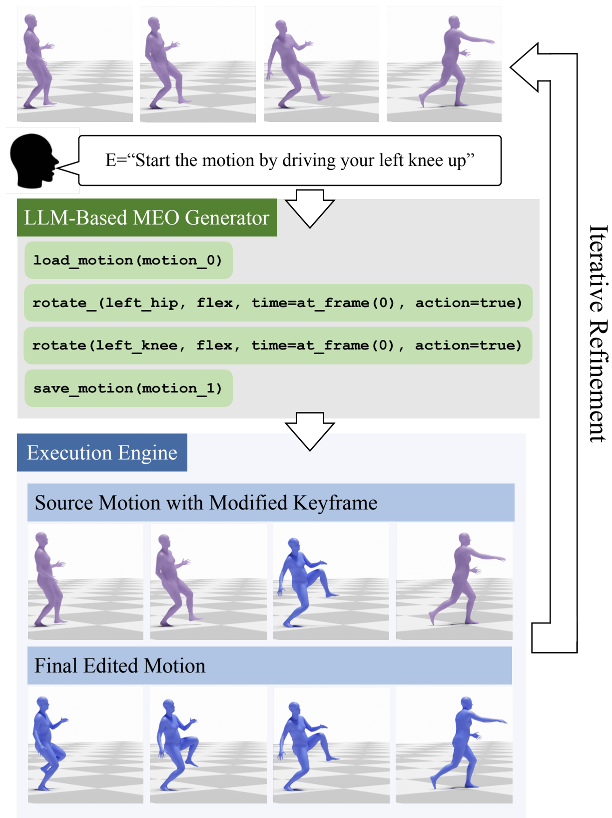


Figure 2: System overview: Our system uses a LLM to translate a natural language editing instruction (E) into source code for a Python program that executes motion editing operations (MEOs). Our MEO execution engine applies MEOs to the source motion by first generating motion constraints (e.g., keyframes, retiming constraints). In the case shown above, E describes a sub-movement that should *start* at the beginning of the motion and lead to a pose in the future; the engine determines the explicit frame requiring editing. A diffusion-based motion infilling step then produces output motions that embody the desired edit, preserve the original motion when possible, and look realistic. Our system can be used in an iterative fashion.

4 METHOD

The key idea of our approach, illustrated in Figure 2, is to cast motion editing as a two-step process: first, converting natural language editing instructions into a sequence of discrete motion editing operations (MEOs), then executing resulting operations using a keyframe generation and diffusion-based motion infilling process. We first describe the MEOs supported by the system. (Section 4.1). Then we describe how we use an LLM, prompted using in-context learning, to translate a plain-text motion editing instruction into an *executable Python program* comprised of MEOs (Section 4.2). Finally we describe how we implement the motion edits described by MEOs to produce new motions (Section 4.3).

4.1 Motion Editing Operators (MEOs)

A common representation for kinematic motion specification is a keyframe, which defines the location or orientation of character's joints at a frame. Our system is inspired by traditional workflows for keyframe editing, but raises the level of abstraction with MEOs.

Like keyframes, the majority of MEOs supported by our system define a joint to modify, a spatial constraint (rotation/translation) on that joint, and a time interval during which the constraint applies. Target joints can in principle be any of the joints in the SMPL model [Bogo et al. 2016]; we focus on the end effectors, knees, elbows, head, shoulders, hips, chest, and waist.

Unlike keyframes, the spatial and temporal constraints of MEOs can be expressed *relative* to properties of the source motion they are being applied to. Rotation/translation constraints may be defined relative to the joint's current configuration (e.g., moving the right hand *higher*, or *abducting* the left hip) or relative to another joint (e.g., moving the right hand *above* the right shoulder). Spatial constraints are applied over a time range that is specified by temporal MEO parameters. Temporal MEO parameters can be explicit references to frame indices, e.g., `at_frame`. We also include implicit references, e.g., `when_joint`, to ground the MEO program in $\mathbf{X}_S$. For example, an MEO might reposition the right knee when the waist is highest. See Figs. 2 and 3 for examples. Additionally, MEOs designate if the edit describes a change in pose that should occur at the specified time, or describes a sub-movement that should begin at the stated time and lead to an edited pose in the future.

To simplify the space of edits that must be accurately generated from natural language instructions, we limit translation and rotation spatial constraints to a small set of discrete directions (e.g, higher/lower, above/below, abduct/adduct), rather than specific numerical values or vectors (e.g, 10.2 cm). It is common for humans to use these coarse descriptions when talking about motion. Our system also supports non-keyframe-based MEOs, such as operations that speed up or slow down a segment of motion.

We implement MEO abstractions as a Python API, containing methods for constructing MEOs and querying the source motion for specific times of extrema events. We provide full details of this API in the supplemental. An example of usage is given in Figure 3.

API for MEO Construction

```
from actions import translate_joint, rotate_joint,...
from timing import when_joint, at_frame,...
from motion_io import load_motion, save_motion
```

Available Parameters

```
relative_moments=["highest", "lowest",...]
translate_directions=["forward", "backward", "up",...]
rotation_directions=["abduct", "adduct", "extend",...]
joints_that_rotate=["right_knee", "left_knee",…]
```

In-context Learning Example(s)

```
# the person is jumping. Bring the right knee to chest
# as you jump
def right_knee_to_chest():
  # load the motion that needs to be edited
  load_motion("motion_0")
  ...
  # bend the right knee
  rotate_joint("right_knee", "flex",
    time=when_joint("waist", "highest"))
  # flex the right hip to bring the knee higher
  rotate_joint("right_knee", "flex",
    time=when_joint("waist", "highest"))
  ...
  # save the edited motion
  save_motion("motion_1")
```

Editing Instruction *E*

```
# A person is doing a side kick with the right leg.
# Can you get that kick higher out?
```

LLM completes code here

Figure 3: LLM Prompt Specification. An abridged LLM prompt that contains MEO API information, an editing prompt E: "Can you get that kick higher out?" (with context E_{ctx} "A person is doing a side kick with the right leg"), and an example MEO program for the task: "lift the right knee to the chest during a jump.", which serves to teach the LLM agent how to use the API. In practice, we provide several examples. The example program here makes API calls to create a plan for completing the editing task, by using MEO construction methods from our API and lists of joints/directions. We ask the LLM agent to write a program that performs the motion edit by combining E and E_{ctx} into a function header comment. The LLM completes the code by writing an MEO program under the header comment.

4.2 Generating MEOs from Natural Language

Given a plain-text motion editing instruction E, and a description of the source motion to modify, we prompt an LLM to generate Python using the MEO API to perform the editing task specified by E.

4.2.1 Context strings for grounding. Interpreting a motion editing instruction requires understanding the context of $\mathbf{X}_S$. For example, without knowledge of the contents of $\mathbf{X}_S$, it is unclear to the LLM which leg the instruction "kick higher" intends to modify. While LLMs are capable of handling some multi-modal tasks [Feng et al. 2023; Gong et al. 2023; Yan et al. 2021], these models cannot yet interpret or produce 3D motion. To address this grounding problem, in addition to the corrective motion editing instruction E (e.g., "kick higher") our system requires a motion context string E_{ctx}: a short description of the current motion (e.g. "a person is doing a side kick with the right leg."). The latter can be provided by the user, automatic captioning [Jiang et al. 2023], or, if the original motion was generated by a text-conditioned model, the original prompt.

4.2.2 Prompt structure. Like [Singh et al. 2023], we inform the LLM agent about the MEOs and time query functions available in our API via `import` statements, and provide the set of valid MEO parameters as a list of strings at the top of the file (e.g., valid joints, relative translation/rotation options). This has been shown to encourage the LLM to use only the methods and parameters it has available. In addition to these inputs we follow standard in-context learning practice and include a small collection of examples of valid MEO programs and their corresponding motion editing prompts [Wei et al. 2023b]. These programs demonstrate how to use the MEO API functions and, in the case of iterative editing sessions (discussed below), how to access the correct motions in the motion undo stack and summarize a new E_{ctx} from the session's history.

At inference time, E and E_{ctx} are provided as code comments and the LLM agent is prompted to "complete the code" to satisfy E. When doing so, we ask the LLM to generate code comments that justify its choices of MEOs and MEO parameters. This form of self-reflection is known to improve the quality of LLM output [Shinn et al. 2023; Yao et al. 2023]. See Fig 3 for an abridged example of a prompt containing one in-context example. If the generated program is invalid (e.g., the LLM uses a invalid function or parameter) the system reports the error message to the LLM agent, which tries generation again.

4.2.3 Iterative Editing Support. Editing motion is an iterative process, and a key goal in our system is to support iterative editing via an extended conversation between the user and the system. Iterative edits can be necessary to clarify or disambiguate the goal movement, or to break larger editing intents into sub-goals. For example, a human might correct, "Kick higher", "Higher", "Now finish in a squat." Alone, edits like "higher" are ambiguous, but gain context from knowledge of previous instructions. So, we provide previous editing instructions and MEO program outputs from the entire session as part of the input prompt in each step. Thus the LLM agent can reference earlier conversation points without their explicit mention in a new E, can correct programs in the case of human-computer miscommunication, and build upon previous edits.

4.2.4 Undo Stack. During iterative editing sessions, the system must determine which motion an editing instruction refers to. For example, the instruction "Keep your hand in front of your waist", followed by "Now add a kick at the start", implies the second edit should modify the output of the first. Conversely, if the second instruction is "No, your other hand", the edit should be applied to the original motion. Our runtime maintains a cache of motions produced during a session, and allows access to each using `load_motion("motion_N")` and `save_motion("motion_N")` API methods. Given a prompt E, the LLM agent must choose which prior motion to load (what the parameter to `load_motion()` should be); the next result is always saved as motion N + 1.

4.3 Execution Engine

To execute the MEO progam, we need identify the specific frames $\mathbf{x}_\text{S}^{\text{key}}$ to operate on. Then, we mechanically edit $\mathbf{x}_\text{S}^{\text{key}}$ to produce edited frames $\mathbf{x}_\text{E}^{\text{key}}$. Finally, we integrate $\mathbf{x}_\text{E}^{\text{key}}$ back into $\mathbf{X}_\text{S}$ while retaining plausibility by leveraging the powerful generative prior of a diffusion model. Our motion notation is illustrated in Fig 4.

4.3.1 Frame Identification. We first identify the frame indices specified by each MEO. Explicit references require no processing of $\mathbf{X}_\text{S}$, but for implicit references, we analyze joint trajectories of $\mathbf{X}_\text{S}$ and compute explicit frame indices using heuristics. For example, if the operation should occur *when* a joint reaches an extremum, we identify the frame index containing the extremum.

4.3.2 Spatial Constraints. Spatial edits are directly executed on $\mathbf{x}_\text{S}^{\text{key}}$ to produce $\mathbf{x}_\text{E}^{\text{key}}$. If an MEO specifies a rotational or translational

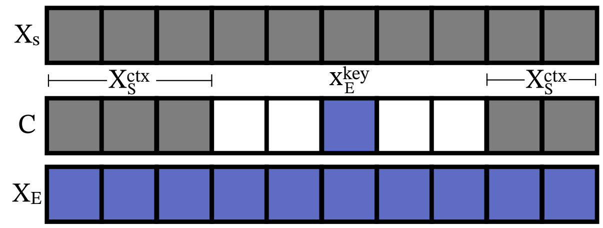


Figure 4: Motion notation. $\mathbf{X}_\text{S}$ is the source motion; condition C comprises $\mathbf{X}_\text{S}^{\text{ctx}}$(context from $\mathbf{X}_\text{S}$) and edited keyframe(s) $\mathbf{x}_\text{E}^{\text{key}}$. Our diffusion-based execution engine outputs $\mathbf{X}_\text{E}$. Gray squares represent components of $\mathbf{X}_\text{S}$; blue squares represent components or desired components of $\mathbf{X}_\text{E}$.

edit, we directly apply it with forward/inverse kinematics, respectively. Edit magnitudes are procedural, grounded in the current articulation of $\mathbf{x}_\text{S}^{\text{key}}$; for example, rotation edits take joints a fraction of the way from the current angle to the joint limit. We use spline-based time warping to change speed [Witkin and Popovic 1995]; the start/end of the identified time range act as time-warp constraints, manipulated programmatically based on the desired change. See our Supplemental for implementation details.

4.3.3 Generative Interpolation. When new keyframes $\mathbf{x}_\text{E}^{\text{key}}$ are added or edited, we need to modify neighboring frames such that transitions appear plausible. While traditional techniques like spline interpolation may to ensure motion smoothness, interpolated results may not appear natural. We cast the problem of integrating $\mathbf{x}_\text{E}^{\text{key}}$ into $\mathbf{X}_\text{S}$ as a motion-infilling problem, and leverage a motion diffusion model to solve for the transition.

Given the context frames $\mathbf{X}_\text{S}^{\text{ctx}}$ and edited keyframe $\mathbf{x}_\text{E}^{\text{key}}$, our model needs to generate a completion of the motion, $\mathbf{X}_\text{E}$. We extend diffusion models [Ho et al. 2020] thanks to their recent success.

The core component in diffusion models is a denoising network, G, trained to reverse the Markov noising process below:

$$q(\mathbf{X}_t|\mathbf{X}) = \mathcal{N}(\sqrt{\alpha_t}\mathbf{X}, (1-\alpha_t)I), \tag{1}$$

where $\alpha_t \in (0,1)$ decrease monotonically. We use a variant of the diffusion model that outputs the denoised motion at each step (the 'simple' loss in [Ho et al. 2020]); denoiser G takes as input the noised motion $\mathbf{X}_t$, the condition $\mathbf{C}$, and the current diffusion step t, and learns to output the denoised motion with objective:

$$\mathcal{L} = \mathrm{E}_{\mathbf{X},t}[\|\mathbf{X} - G(\mathbf{X}_t, \mathbf{C}, t)\|_2^2]. \tag{2}$$

The representation of $\mathbf{C}$ is a critical design choice. Since our goal is to teach the model to infill transitional motion around $\mathbf{x}_\text{E}^{\text{key}}$(which will be provided by MEOs at inference time) for a transition window of length W and motion length F, we represent $\mathbf{C}$ as a motion sequence composed of context frames from the source $\mathbf{X}_\text{S}^{\text{ctx}}$: $\mathbf{x}^{0:\text{key}-1-W}$ and $\mathbf{x}^{\text{key}+W:F-1}$, and with $\mathbf{x}_\text{E}^{\text{key}}$. Neighboring frames around $\mathbf{x}_\text{E}^{\text{key}}$, $\mathbf{x}^{\text{key}-W:\text{key}}$ and $\mathbf{x}^{\text{key}+1:\text{key}+W}$ are masked to zero (see Fig. 4).

During training, we randomly sample a keyframe index between 0 and $F-1$ and zero out W frames before/after the keyframe(s). The window is clamped at the start and end of the sequence.

4.3.4 A Diffusion-Based Architecture for Infilling Conditioning. We use an architecture based on the transformer-decoder, augmented with a conditioning branch to account for $\mathbf{C}$. To distinguish the original input branch from the conditioning branch, we use the following labels, '`input`' vs '`cond`' (Figure 5). In practice, since $\mathbf{C}$ can be thought of as a masked version of a full motion sequence $\mathbf{X}$, we introduce a binary mask $\mathbf{M}$ that zeros out attributes that need to be infilled. Like [Wei et al. 2023a], we feed context information in the original input branch rather than pure noise:

$$G(\texttt{input=}\mathbf{M} \odot \mathbf{X} + (1 - \mathbf{M}) \odot q(\mathbf{X}_t|\mathbf{X}), \\ \texttt{cond=}\mathbf{M} \odot \mathbf{X}, t). \tag{3}$$

4.3.5 Inference. At inference, $\mathbf{C}$ comprises $\mathbf{X}_S^{ctx}$and $\mathbf{x}_E^{key}$. $\mathbf{M}$ zeros out degrees of freedom that need to be infilled around $\mathbf{x}_E^{key}$. Similar to the image blending process proposed by [Avrahami et al. 2022], we observe more consistent generations for non-root edits when we seed the denoising process with an initial guess for the masked area; we use $\mathbf{X}_{\text{spline}}$, a naive solution for $\mathbf{X}_E$ which integrates $\mathbf{x}_E^{key}$into $\mathbf{X}_S$ using spline interpolation. At each diffusion step t, we spatially blend *(*lerp) the infilled regions of an appropriately noised $\mathbf{X}_{\text{spline}}$ with $\mathbf{X}_t$ using monotonically decreasing interpolant λ_t. Our insight is that $\mathbf{X}_{\text{spline}}$ guides the start of the denoising process, while later diffusion steps add detail.

4.3.6 Implementation. In practice, we found that edits to the root joint, e.g., to make a character jump or crouch, were better handled if $\mathbf{C}$ included the root trajectory to help break the problem down. So, we trained two models in the manner described above: a regression model to infill the root trajectory, and G to infill the rest of the body. At inference, the first model generates the root trajectory, which the second model includes in $\mathbf{C}$ to infill the other degrees of freedom. We train both models on the AMASS dataset [Mahmood et al. 2019].

Masking frames with $\mathbf{M}$ can destroy important structural information from $\mathbf{X}_S$. So, at inference, we automatically detect important frames in $\mathbf{X}_S$ that should be preserved and include these in $\mathbf{C}$. Frames are important if they either contain significant extrema, or were edited in a previous iteration. $\mathbf{X}_E$ can optionally be postprocessed with, e.g., smoothing and foot-skate clean-up.

5 EVALUATION

5.1 Implementation Details

We use OpenAI's ChatGPT-4 as our LLM agent. We train diffusion model G and the trajectory infilling model on the AMASS dataset [Mahmood et al. 2019] using an NVIDIA Tesla V100 GPU. All motions are represented as 60-frame clips (2.5 seconds). Hyperparameters are included in the Supplemental.

5.2 Qualitative Evaluation

We used our system to edit a variety of motions using natural language. In Figures 2 and 6, and our Supplemental Materials, we demonstrate how on a per-edit basis, the system can handle a range of editing intents, and produce a variety of motions that are faithful to the edit, preserve qualities of the original motion, and are visually plausible. Our primary goal, though, is to provide a conversational interface supporting iterative editing; we show results of iterative

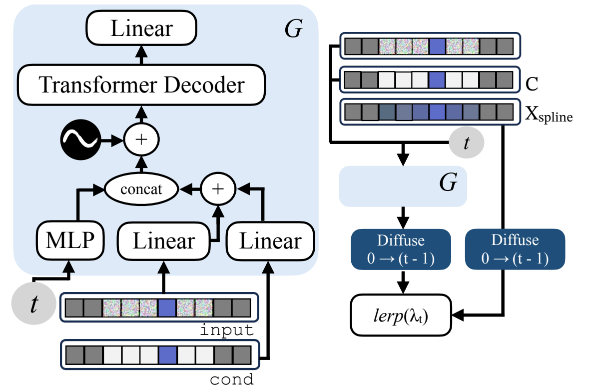


Figure 5: Infilling Diffusion Model. In training, our model *(left)* learns to infill motions. *G* takes `input`, a noisy sequence imputed with C, and `cond`, a masked verion of C. At inference *(right)*, we optionally integrate $\mathbf{X}_{\text{spline}}$ to guide inference. For each t we spatially *lerp* the infilled frames of $\mathbf{X}_{\text{spline}}$ with those progressively generated by *G* with interpolant $\lambda(t)$, which decreases monotonically as a function of t.

editing sessions in Figure 1 and in the accompanying video. In these examples, instructions are used to progressively refine the character's motion, break complex goals into step-by-step instructions, and also clarify or adjust editing intent during the refinement process.

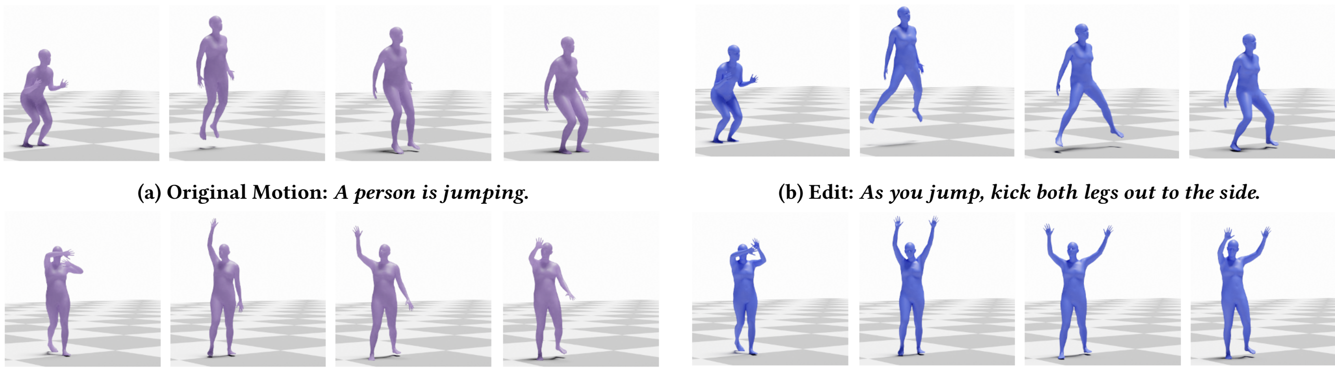
5.3 Quantitative Evaluation

5.3.1 Experimental Setup. To quantitatively evaluate our system, we compare its performance against two SOTA text-to-motion models: MDM [Tevet et al. 2023] and MoMask [Guo et al. 2023], using automated metrics and a user study.

MDM cannot take source motions as input; therefore, to generate edited motions, we first write ten captions that are plentiful in MDM's training data (HumanML3D [Guo et al. 2022]) like kicking and throwing, e.g., "A person is kicking with the right foot." Captions are fed into MDM to generate several motions to be $\mathbf{X}_S$. Next, we concatenate each source caption with different editing instructions, e.g., "A person is kicking with the right foot. As you kick, raise your arms out to the side." Editing instructions were inspired by kinematic motion descriptions that appeared often in HumanML3D. For the baseline, MDM-Edit, we fix MDM's generation seed and rerun it on the concatenated caption. We compare MDM-Edit with our system's editing of $\mathbf{X}_S$ using the same caption.

MoMask can perform mask-based editing, e.g., inpainting source motions within a specified mask given a new prompt, but cannot deduce mask frame indices. So, we generate a separate set of $\mathbf{X}_S$ with MoMask, then employ our LLM-based parser to determine the frame(s) associated with different editing prompts. We rerun MoMask with the editing prompts using masks centered around these frames, producing baseline MoMask-Edit.

5.3.2 User Study. We conduct a user study to compare a sample of edited motions. 19 users rated nine MDM-Edit motions versus our edited versions, and nine MoMask-Edit motions versus our



(a) Original Motion: ***A person is jumping.***

(b) Edit: ***As you jump, kick both legs out to the side.***

(c) Original Motion: ***A person is swinging their arms.***

(d) Edit: ***Synchronize your arms.***

Figure 6: Handling natural-language instructions. Starting from a source motion (left column, in purple) and editing instruction (italicized), our system produces plausible motions (right column, blue) that preserve the structure of the original motion and abide by the editing instruction.

Table 1: User study results. 19 participants rated faithfulness of the edited motion to the instruction (Fidelity), preservation of the source motion's structure (StrucSim), and motion quality (Qual). We report average scores over all users and motions; we score much higher on StrucSim and Fidelity, and similarly on Qual.

	StrucSim (↑)	Fidelity (↑)	Qual (↑)
MDM-Edit	2.77	1.56	3.79
Ours	4.33	4.48	4.07
MoMask-Edit	3.59	1.72	3.88
Ours	4.25	4.52	3.77

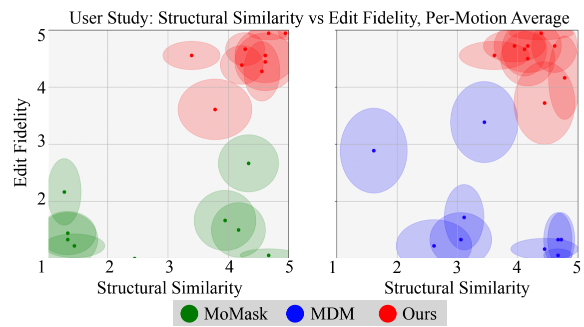


Figure 7: Per-motion average score for edit fidelity vs structural similarity in our user study. MDM-Edit (blue) and MoMask-Edit (green) struggle to achieve a high score on both axes at the same time; high scores in structural similarity are often at the cost of edit fidelity. In contrast, our system (red) simultaneously scores high on both.

edited versions. Users were asked to rate each result based on the overall quality of each motion, fidelity to the editing instruction, and structural similarity to the original motion from 1-5 (higher is better); details are in the Supplemental. We show average scores in Table 1. Our system's Fidelity scores far exceed both baselines, and were judged better on structural similarity. On a per-motion level, we observe that though baselines can sometimes maintain structural similarity, they often struggle to simultaneously maintain Fidelity. In contrast, our motions score high on both axes; see Fig. 7.

5.3.3 Metrics. We also evaluate our system against baselines using automated metrics for an additional 17 edited motion pairs for each baseline. We measure structural similarity using G-MPJPE, a common geometric distance metric in motion reconstruction. To measure edit fidelity, we author binary edit fidelity tests that use joint positions to assess whether changes fulfill the desired intent of a given MEO. We rate edit fidelity by the average number of tests passed (Fidelity-Auto). We measure quality using Frechet Distance to compare an empirical distribution against 1000 ground truth motions in the *fairmotion* [Gopinath and Won 2020] geometric feature space (FID_g) [Li et al. 2021]. See the Supplemental for more details.

Quantitative metrics reveal similar trends to our user study. Against MoMask-Edit, our edited motions score 140% higher on Fidelity-Auto (0.88 versus 0.6), and are structurally more similar to MoMask-Source. We show similar improvement over MDM-Edit–see Table 2.

We do not compare the motion quality of our system vs MDM-Edit or MoMask-Edit quantitatively here; all are editing motions that have been generated by MDM/MoMask, which already have some deviation from ground-truth human motions that would affect overall quality scores of their edited versions.

5.3.4 Execution Engine Ablation Study. We measure motion quality over an ablation of our execution engine. We start with 100 real mocap sequences in AMASS (AMASS-Source). We pair each with 1-3 MEOs and edit AMASS-Source using ablated versions of the engine: 1) ENG, our proposed engine, 2) ENG-SS, where diffusion model G is trained to infill the entire body instead of our two-stage

Table 2: Quantitative evaluation with automated metrics. Both versus MoMask-Edit and MDM-Edit, our system scores more favorably on edit fidelity and G-MPJPE. We find our evaluation to be statistically significant with pairwise comparisons using Wilcoxon's signed rank test. Ours vs MDM: p<0.03,Z=24 for fidelity and p<0.0003,Z=10 for GMPJPE. Ours vs MoMask: p<0.02,Z=10 for fidelity and p<0.0002,Z=8 for GMPJPE.

	MDM-Edit	Ours	MoMask-Edit	Ours
Fidelity (↑)	0.588	0.82	0.6	0.882
G-MPJPE (↓)	0.247	0.08	0.181	0.063

process, and 3) ENG-Interp, where we use spline interpolation instead of G. We compare FID_g for each engine's generations.

ENG produces motion distributions that more closely match those of source motions. Edited motions should preserve the overall structure of the source, so we expect FID_g of edited motions to *match* FID_g of AMASS-Source, rather than improve upon it. Indeed, AMASS-Source scores 4.33 and ENG only observes marginal increase to 4.95. Ablations degrade the FID_g score: ENG-SS drops the FID_g to 5.25 and as we expect, ENG-Interp's spline interpolation produces the least "human-like" results with FID_g of 8.05.

6 DISCUSSION AND LIMITATIONS

Limitations. MEOs are limited to kinematic constraints; physics-informed edits like, "jump more forcefully" are not handled by our system. Extending the execution engine to support these edits are exciting future directions. In our system, source motion context and keyframes act as a condition for the diffusion model, but should not necessarily be considered as "hard" spatiotemporal constraints, e.g., editing of joint positions can result in extra displacement of a joint, which in turn should require more transition time to avoid velocity inconsistencies. We are eager to explore methods to improve motion quality and make the execution engine more robust to such input. Currently, our system's frame-picking is largely based on joint extrema; more sophisticated methods for motion understanding [Endo et al. 2023] could make this more flexible.

In conclusion, we have demonstrated a system for editing motions with text, by first translating text instructions into keyframe-like "constraints". Our system can iteratively edit motions from a variety of sources: mocap datasets [Mahmood et al. 2019], modern generative models [Guo et al. 2023; Tevet et al. 2023], extracted from video [Wang et al. 2023a], etc. We are excited about expanding the scope user inputs to the system to more than just text-based instruction and adding new MEO operators, which we believe can specify many edits, e.g., stylistic changes and physically-informed objectives. Extending the system in this manner would provide new ways for users to direct characters.

ACKNOWLEDGMENTS

Purvi Goel is supported by a Stanford Interdisciplinary Graduate Fellowship. Kuan-Chieh Wang was supported by Stanford Wu-Tsai Human Performance Alliances while at Stanford University. We thank the anonymous reviewers for constructive feedback; Vishnu Sarukkai, Sarah Jobalia, Sofia Di Toro Wyetzner for proofreading; Haotian Zhang, David Durst, and James Hong for helpful discussions. Our codebase was built with invaluable help from James Burgess.

REFERENCES

Kfir Aberman, Yijia Weng, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. 2020. Unpaired Motion Style Transfer from Video to Animation. *ACM Transactions on Graphics (TOG)* 39, 4 (2020), 64.

Maneesh Agrawala. 2023. Unpredictable Black Boxes are Terrible Interfaces. https://magrawala.substack.com/p/unpredictable-black-boxes-are-terrible.

Andreas Aristidou, Daniel Cohen-Or, Jessica K. Hodgins, Yiorgos Chrysanthou, and Ariel Shamir. 2018. Deep Motifs and Motion Signatures. *ACM Trans. Graph.* 37, 6, Article 187 (dec 2018), 13 pages. https://doi.org/10.1145/3272127.3275038

Omri Avrahami, Dani Lischinski, and Ohad Fried. 2022. Blended Diffusion for Text-Driven Editing of Natural Images. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 18208–18218.

Federica Bogo, Angjoo Kanazawa, Christoph Lassner, Peter Gehler, Javier Romero, and Michael J. Black. 2016. Keep it SMPL: Automatic Estimation of 3D Human Pose and Shape from a Single Image. In *Computer Vision – ECCV 2016 (Lecture Notes in Computer Science)*. Springer International Publishing.

Tim Brooks, Aleksander Holynski, and Alexei A. Efros. 2023. InstructPix2Pix: Learning to Follow Image Editing Instructions. In *CVPR*.

Ginger Delmas, Philippe Weinzaepfel, Francesc Moreno-Noguer, and Grégory Rogez. 2023. PoseFix: Correcting 3D Human Poses with Natural Language. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 15018–15028.

Mark Endo, Joy Hsu, Jiaman Li, and Jiajun Wu. 2023. Motion Question Answering via Modular Motion Programs. *ICML* (2023).

Yao Feng, Jing Lin, Sai Kumar Dwivedi, Yu Sun, Priyanka Patel, and Michael J. Black. 2023. PoseGPT: Chatting about 3D Human Pose. arXiv:2311.18836 [cs.CV]

Michael Gleicher. 1997. Motion Editing with Spacetime Constraints. In *Proceedings of the 1997 Symposium on Interactive 3D Graphics* (Providence, Rhode Island, USA) *(I3D '97)*. Association for Computing Machinery, New York, NY, USA, 139–ff. https://doi.org/10.1145/253284.253321

Michael Gleicher. 2001. Motion Path Editing. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics (I3D '01)*. Association for Computing Machinery, New York, NY, USA, 195–202. https://doi.org/10.1145/364338.364400

Tao Gong, Chengqi Lyu, Shilong Zhang, Yudong Wang, Miao Zheng, Qian Zhao, Kuikun Liu, Wenwei Zhang, Ping Luo, and Kai Chen. 2023. MultiModal-GPT: A Vision and Language Model for Dialogue with Humans. arXiv:2305.04790 [cs.CV]

Deepak Gopinath and Jungdam Won. 2020. fairmotion - Tools to load, process and visualize motion capture data. Github. https://github.com/facebookresearch/fairmotion

Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. 2023. MoMask: Generative Masked Modeling of 3D Human Motions. (2023). arXiv:2312.00063 [cs.CV]

Chuan Guo, Shihao Zou, Xinxin Zuo, Sen Wang, Wei Ji, Xingyu Li, and Li Cheng. 2022. Generating Diverse and Natural 3D Human Motions From Text. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5152–5161.

Sehoon Ha and C. Karen Liu. 2015. Iterative Training of Dynamic Skills Inspired by Human Coaching Techniques. *ACM Trans. Graph.* 34, 1, Article 1 (dec 2015), 11 pages. https://doi.org/10.1145/2682626

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. 2022. Prompt-to-prompt image editing with cross attention control. (2022).

Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising Diffusion Probabilistic Models. *arXiv preprint arxiv:2006.11239* (2020).

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. 2022. Language Models as Zero-Shot Planners: Extracting Actionable Knowledge for Embodied Agents. *CoRR* abs/2201.07207 (2022). arXiv:2201.07207 https://arxiv.org/abs/2201.07207

Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. 2023. MotionGPT: Human Motion as a Foreign Language. *arXiv preprint arXiv:2306.14795* (2023).

Zhongyu Jiang, Zhuoran Zhou, Lei Li, Wenhao Chai, Cheng-Yen Yang, and Jenq-Neng Hwang. 2024. Back to optimization: Diffusion-based zero-shot 3d human pose estimation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*. 6142–6152.

Korrawe Karunratanakul, Konpat Preechakul, Emre Aksan, Thabo Beeler, Supasorn Suwajanakorn, and Siyu Tang. 2023. Optimizing Diffusion Noise Can Serve As Universal Motion Priors. In *arxiv:2312.11994*.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2023. *Motion Graphs* (1 ed.). Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3596711.3596788

Sumith Kulal, Jiayuan Mao, Alex Aiken, and Jiajun Wu. 2021. Hierarchical Motion Understanding via Motion Programs. In *CVPR*.

Jehee Lee and Sung Yong Shin. 1999. A Hierarchical Approach to Interactive Motion Editing for Human-like Figures. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 39–48. https://doi.org/10.1145/311535.311539

Jiaman Li, Jiajun Wu, and C Karen Liu. 2023. Object motion guided human motion synthesis. *ACM Transactions on Graphics (TOG)* 42, 6 (2023), 1–11.

Ruilong Li, Shan Yang, David A. Ross, and Angjoo Kanazawa. 2021. AI Choreographer: Music Conditioned 3D Dance Generation with AIST++. arXiv:2101.08779 [cs.CV]

Jacky Liang, Wenlong Huang, Fei Xia, Peng Xu, Karol Hausman, Brian Ichter, Pete Florence, and Andy Zeng. 2023. Code as Policies: Language Model Programs for Embodied Control. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. https://doi.org/10.1109/icra48891.2023.10160591

Naureen Mahmood, Nima Ghorbani, Nikolaus F. Troje, Gerard Pons-Moll, and Michael J. Black. 2019. AMASS: Archive of Motion Capture as Surface Shapes. In *International Conference on Computer Vision*. 5442–5451.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. 2022. SDEdit: Guided Image Synthesis and Editing with Stochastic Differential Equations. arXiv:2108.01073 [cs.CV]

Lea Müller, Vickie Ye, Georgios Pavlakos, Michael Black, and Angjoo Kanazawa. 2023. Generative Proxemics: A Prior for 3D Social Interaction from Images. *arXiv preprint arXiv:2306.09337* (2023).

Boris N. Oreshkin, Florent Bocquelet, Félix G. Harvey, Bay Raitt, and Dominic Laflamme. 2022. ProtoRes: Proto-Residual Network for Pose Authoring via Learned Inverse Kinematics. In *International Conference on Learning Representations*.

Jia Qin, Youyi Zheng, and Kun Zhou. 2022. Motion In-Betweening via Two-Stage Transformers. *ACM Trans. Graph.* 41, 6, Article 184 (nov 2022), 16 pages. https://doi.org/10.1145/3550454.3555454

Davis Rempe, Zhengyi Luo, Xue Bin Peng, Ye Yuan, Kris Kitani, Karsten Kreis, Sanja Fidler, and Or Litany. 2023. Trace and Pace: Controllable Pedestrian Animation via Guided Trajectory Diffusion. In *Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jiawei Ren, Mingyuan Zhang, Cunjun Yu, Xiao Ma, Liang Pan, and Ziwei Liu. 2023. InsActor: Instruction-driven Physics-based Characters. *NeurIPS* (2023).

Vishnu Sarukkai, Linden Li, Arden Ma, Christopher Ré, and Kayvon Fatahalian. 2023. Collage Diffusion. arXiv:2303.00262 [cs.CV]

Yonatan Shafir, Guy Tevet, Roy Kapon, and Amit H Bermano. 2023. Human motion diffusion as a generative prior. *arXiv preprint arXiv:2303.01418* (2023).

Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. arXiv:2303.11366 [cs.AI]

Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. ProgPrompt: Generating Situated Robot Task Plans using Large Language Models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. 11523–11530. https://doi.org/10.1109/ICRA48891.2023.10161317

Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. ViperGPT: Visual Inference via Python Execution for Reasoning. *Proceedings of IEEE International Conference on Computer Vision (ICCV)* (2023).

Guy Tevet, Sigal Raab, Brian Gordon, Yoni Shafir, Daniel Cohen-or, and Amit Haim Bermano. 2023. Human Motion Diffusion Model. In *The Eleventh International Conference on Learning Representations*. https://openreview.net/forum?id=SJ1kSyO2jwu

Jonathan Tseng, Rodrigo Castellon, and C Karen Liu. 2022. EDGE: Editable Dance Generation From Music. *arXiv preprint arXiv:2211.10658* (2022).

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2023b. Voyager: An Open-Ended Embodied Agent with Large Language Models. arXiv:2305.16291 [cs.AI]

Kuan-Chieh Wang, Zhenzhen Weng, Maria Xenochristou, João Pedro Araújo, Jeffrey Gu, Karen Liu, and Serena Yeung. 2023a. NeMo: Learning 3D Neural Motion Fields From Multiple Video Instances of the Same Action. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 22129–22138.

Dong Wei, Xiaoning Sun, Huaijiang Sun, Bin Li, Sheng liang Hu, Weiqing Li, and Jian-Zhou Lu. 2023a. Understanding Text-driven Motion Synthesis with Keyframe Collaboration via Diffusion Models. *ArXiv* abs/2305.13773 (2023). https://api.semanticscholar.org/CorpusID:258841591

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023b. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL]

Andrew P. Witkin and Zoran Popovic. 1995. Motion warping. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995). https://api.semanticscholar.org/CorpusID:1497012

Wilson Yan, Yunzhi Zhang, Pieter Abbeel, and Aravind Srinivas. 2021. VideoGPT: Video Generation using VQ-VAE and Transformers. arXiv:2104.10157 [cs.CV]

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL]

Wenjie Yin, Hang Yin, Kim Baraka, Danica Kragic, and Mårten Björkman. 2023. Dance Style Transfer with Cross-modal Transformer. In *2023 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*. 5047–5056. https://doi.org/10.1109/WACV56688.2023.00503

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

Mingyuan Zhang, Huirong Li, Zhongang Cai, Jiawei Ren, Lei Yang, and Ziwei Liu. 2023. FineMoGen: Fine-Grained Spatio-Temporal Motion Generation and Editing. *NeurIPS* (2023).