Sketch2Anim: Towards Transferring Sketch Storyboards into 3D Animation

LEI ZHONG, University of Edinburgh, United Kingdom
CHUAN GUO, Snap Inc., United States
YIMING XIE, Northeastern University, United States
JIAWEI WANG, University of Edinburgh, United Kingdom
CHANGJIAN LI, University of Edinburgh, United Kingdom

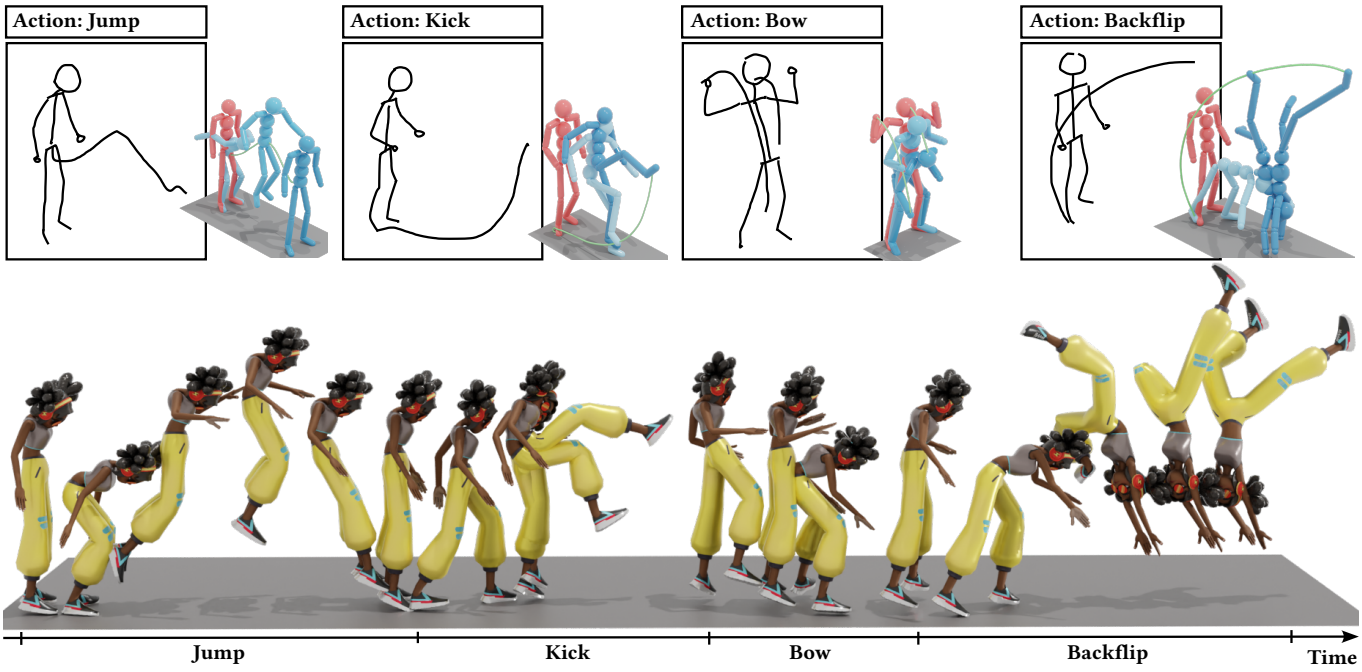


Fig. 1. **Storyboard animation**. Given a storyboard (top row) composed of a few sketch keyposes, joint trajectories, and the corresponding action words, we propose *Sketch2Anim*, a novel approach that automatically transfers each sketch frame into a 3D motion clip, where the keypose is highlighted in red, and the trajectory is highlighted in green. All these clips are then blended into a complete and coherent animation (bottom row) depicting the storyboard.

Storyboarding is widely used for creating 3D animations. Animators use the 2D sketches in storyboards as references to craft the desired 3D animations through a trial-and-error process. The traditional approach requires exceptional expertise and is both labor-intensive and time-consuming. Consequently, there is a high demand for automated methods that can directly translate 2D storyboard sketches into 3D animations. This task is underexplored to date and inspired by the significant advancements of motion diffusion models, we propose to address it from the perspective of conditional motion synthesis. We thus present *Sketch2Anim*, composed of two key modules for sketch constraint understanding and motion generation. Specifically, due to the large domain gap between the 2D sketch and 3D motion, instead of directly conditioning on 2D inputs, we design a 3D conditional motion generator that simultaneously leverages 3D keyposes, joint trajectories, and action words, to achieve precise and fine-grained motion control. Then, we invent a neural mapper dedicated to aligning user-provided 2D sketches with their corresponding 3D keyposes and trajectories in a shared embedding space, enabling, *for the first time*, direct 2D control of motion generation. Our approach successfully transfers storyboards into high-quality 3D motions and inherently supports direct 3D animation editing, thanks to the flexibility of our multi-conditional motion generator. Comprehensive experiments and evaluations, and a user perceptual study demonstrate the effectiveness of our approach.

The code, data, trained models, and sketch-based motion designing interface are at https://zhongleilz.github.io/Sketch2Anim/.

CCS Concepts: • **Computing methodologies** → **Animation**.

Additional Key Words and Phrases: Story Board, Sketch-based Animation, Motion synthesis

Authors' Contact Information: Lei Zhong, University of Edinburgh, Edinburgh, United Kingdom, zhongleilz@icloud.com; Chuan Guo, Snap Inc., New York, United States, guochuan5513@gmail.com; Yiming Xie, Northeastern University, Boston, United States, ymxyimingxie@gmail.com; Jiawei Wang, University of Edinburgh, Edinburgh, United Kingdom, jiaweiwang0222@gmail.com; Changjian Li, University of Edinburgh, Edinburgh, United Kingdom, chjili2011@gmail.com.

ACM Reference Format:
Lei Zhong, Chuan Guo, Yiming Xie, Jiawei Wang, and Changjian Li. 2025. *Sketch2Anim*: Towards Transferring Sketch Storyboards into 3D Animation. *ACM Trans. Graph.* 44, 4 (August 2025), 20 pages. https://doi.org/10.1145/3731167

1 Introduction

> "Animation is not the art of drawings that move but the art of movements that are drawn."
>
> — Norman McLaren

Sketch storyboards are widely used by artists to quickly outline their creative animation ideas [Lasseter 1998; Williams 2012]. As illustrated in Figs. 1 and 2, a storyboard typically consists of a series of 2D sketch frames representing the desired animation clips. Each frame features an abstracted character sketch depicting the keypose, accompanied by joint trajectories that convey the envisioned animation sequence. Additionally, a single action word is often tagged to indicate the semantic and motion space. For instance, as shown in the second frame in Fig. 1, the word "Kick" constrains the motion space, while the keypose describes the starting pose and the trajectory describes the specific leg movement. In the current animation workflow (Fig. 2), animators use these 2D sketches as references and manually align pre-defined 3D character joints to match the 2D keypose, while adding 3D joint motions to follow the 2D trajectories through a trial-and-error process. This workflow requires exceptional expertise developed over years of practice and is both labor-intensive and time-consuming. Therefore, automated methods that can directly translate 2D storyboards into 3D animations are highly demanded.

Earlier research closely relevant to this task is sketch-based motion retrieval [Thorne et al. 2004; Yoo et al. 2014]. They usually extract handcrafted features (*e.g.*, joint angles and bone vectors) from input sketches and search for the closest 3D animation in a pre-collected database. However, these methods are constrained by the scope of pre-existing datasets, resulting in limited scalability and expressiveness. When dealing with multiple sketch types (*e.g.*, keyposes and trajectories), it is often hard to design proper matching score functions, leading to inferior retrieved animations.

More recently, diffusion models have demonstrated remarkable performance in motion generation [Athanasiou et al. 2024; Barquero et al. 2024; Chen et al. 2023; Goel et al. 2024; Tevet et al. 2023]. Inspired by the great advancements, we propose to solve the translation from 2D storyboards to 3D motions from the perspective of conditional motion synthesis. With this key idea, the translation can be reformulated as two sub-problems: i) *how to obtain the accurate 2D keypose and trajectory from the raw sketch of a storyboard* and ii) *how to produce a high-quality motion clip conditioned on the 2D keypose, trajectory, and action word?* This first sub-problem is well-studied, *e.g.*, Sketch2Pose [Brodt and Bessmeltsev 2022] can robustly detect 2D keyposes from sketches and the 2D trajectory can be traced precisely in the user interface. In this paper, we focus on the second sub-problem, *i.e.*, multi-conditional motion generation.

Intuitively, conditioning a motion diffusion model on 2D keyposes, joint trajectories, and action words offers a straightforward approach. However, due to our unique problem setting, the considerable domain gap between 2D inputs and 3D motion makes this

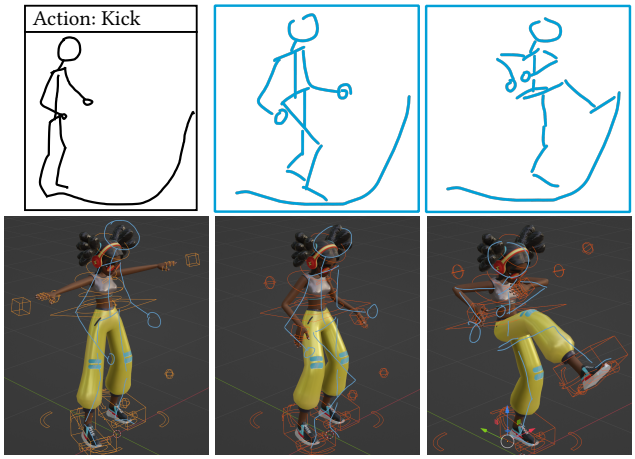


Fig. 2. **Traditional 3D animation workflow.** Given one frame of a sketch storyboard (top-left), animators first imagine the desired keypose sequence guided by the trajectory and the action word. Two such representative keyposes are shown in blue. These keyposes (including the first keypose) are imported into 3D software (*e.g.*, Blender) as keypose and motion references. The animators then manually place the pre-defined 3D joints to match the keypose, while crafting the desired motion to explain the trajectory and action word. After a long trial-and-error process, a high-quality animation is produced. *See the supplemental video for the whole manual process.*

solution infeasible. Specifically, the detected 2D keypose and joint trajectories are the projection of the imagined 3D motion. The former is in the pixel space, while the latter is in the 4D space (*i.e.*, space and time). High-quality motions directly from 2D conditional inputs are hard to learn (see Sec. 7.2). In contrast, 3D keyposes and trajectories are more informative, providing concrete guidance for motion generation. However, even with well-defined 3D keypose and trajectory, the previous conditional motion generation methods are incapable of consuming all of them, since they are typically limited to handling one or two control signals, such as text or text paired with 3D trajectories [Xie et al. 2024], 3D indoor scene [Yi et al. 2024], or 3D contacting objects [Diller and Dai 2024; Li et al. 2025]. Controlling models to simultaneously adhere to more than two conditions, such as action words, keyposes, and trajectories remains a significant challenge.

To address the above challenges, we propose a novel approach, dubbed *Sketch2Anim*, composed of two core modules. Our design is inspired by the two key operators in the traditional animation workflow - lifting the 2D keypose and trajectory to 3D by 2D-3D alignment and crafting the vivid motion based on the lifted constraints. Specifically, instead of mathematically lifting the 2D keypose and trajectory into 3D, we design a neural mapper (Sec. 5) dedicated to aligning 2D keypose and trajectory to their corresponding 3D keypose and trajectory in the shared embedding space. This shared embedding space enables seamless acceptance of 2D keyposes and joint trajectories as input during inference, and more informative and precise 3D keypose and trajectory as conditions during training.

Our second module is a multi-conditional motion generator (Sec. 4), where we successfully inject both the keypose and trajectory controls on top of the action word. Although the 3D keypose and trajectory can be combined through joint positions and treated as

a single control signal, they actually emphasize different motion properties. The keypose primarily imposes local static constraints at specific timesteps, while a joint trajectory conveys both global and local dynamic movements. To better balance their respective influences during training, we propose a trajectory ControlNet combined with a trajectory-aware keypose adapter. The mechanism is that the adapter that is positioned between the trajectory ControlNet and the motion diffusion model, refines the output features from the trajectory ControlNet using the keypose condition and integrates the refined residual features into the diffusion model. This design enables the ControlNet to focus on dynamic motion control, while the adapter refines the local pose at a specific timestep, effectively minimizing interference between the two conditions. We finally blend the animation clips from each keyframe into a complete and coherent 3D animation explaining the storyboard. With the new approach, we are able to translate the 2D sketch storyboard directly into its 3D animation, and in the meantime, it naturally supports 3D motion editing, benefiting from our motion generator.

We evaluate our method on a few synthetic benchmarks and real user sketches, and in both cases, *Sketch2Anim* successfully transfers sketch storyboards into high-quality animations. Extensive experiments and ablation studies validate the effectiveness of our method.

In summary, the principal contributions of this work include:

- We introduce a neural mapper to align the embedding space of 2D and 3D keyposes and trajectories. This shared embedding space allows us to leverage 3D control signals as 2D surrogates during training and then seamlessly use 2D control signals during inference.
- We present a keypose adapter integrated with the trajectory ControlNet, simultaneously controlling the motion diffusion model to effectively adhere to both keypose and joint trajectory constraints.
- Combined together, these technical contributions enable the *first* approach that adapts the motion diffusion model to generate 3D animations directly from 2D storyboards, and an accompanying sketch-based animation design system.

2 Related Work

In this section, we summarize prior works from sketch-based 3D character animation and human motion synthesis that are closely related to our task.

Sketch-based 3D Character Animation. Sketch-based 3D character animation provides intuitive tools for artists to create lifelike animations directly from sketches [Choi et al. 2016; Peng et al. 2021; Zhou et al. 2024]. Before attending to 3D animation, early works study the problem of inferring static 3D poses from various types of sketches, such as 2D stick figures [Davis et al. 2006; Hahn et al. 2015; Lin et al. 2010], lines of action [Guay et al. 2013], custom sketches [Brodt and Bessmeltsev 2022], and silhouette sketches [Bessmeltsev et al. 2016]. The estimated 3D pose is then used to pose a pre-defined 3D character. For instance, The Line of Action [Guay et al. 2013] introduces a sketch-based interface that allows users to draw a single aesthetic line of action to pose an existing 3D character by solving an optimization. Lin *et al.* [2010] propose to use stick figure sketches for sitting pose design, while Hahn *et al.* [2015] extend the usability to general characters with their rough stick-figure sketch. More recently, Brodt *et al.* [2022] propose Sketch2Pose, which can handle more complicated sketches. Their method first predicts 2D joints and body part contacts, and optimizes the 3D pose of an SMPL model to conform to the sketch. Based on our experiments, their 2D joint detection in our storyboard scenario is robust, but their 3D keypose is less satisfactory. Thus, as a pre-processing step, we use their method to obtain the 2D joints for our approach.

Other than 3D poses, as a step forward, some works start exploring the dynamic motion modeling from 2D sketches, *e.g.*, simple 2D paths [Igarashi et al. 1998; Thorne et al. 2004] and space-time curves [Guay et al. 2015]. MotionDoodles [Thorne et al. 2004] presents a sketch-based interface that allows users to create character motions by drawing gesture sketches. They build a gesture vocabulary to drive the character to perform desired motions. Yoo *et al.* [2014] propose a pipeline for generating 3D human character motions by retrieving motion sequences from a database. Their input sketch consists of character strokes, motion curves, and rotation curves, and is analogous to ours, especially since the latter two curves are similar to trajectory strokes. However, these methods are retrieval-based and depend highly on predefined motion templates or databases, limiting their adaptability to complex or unseen motions. In contrast, our system overcomes this limitation by synthesizing novel motions through a motion diffusion model.

Closest to our work, the concurrent study DoodleYourMotion [Wu et al. 2024] proposes a sketch-conditioned motion generation method. Their approach utilizes a silhouette sketch image as a control signal to guide the motion diffusion model and introduces a sketch-aware local attention mechanism to enhance interactions between local sketch patches and motion keyframes. However, by relying solely on the sketch image as the control signal, their method is constrained to influencing only the keyframes in the generated motion, lacking the ability to effectively control the motion's dynamics. As a comparison, our sketch system exploits more simple stick figure style character sketches to ease user burden and supports trajectory strokes to enable finer-grained motion control.

In parallel, another line of research aims to animate static images or drawings [Dvorožňák et al. 2020; Smith et al. 2023; Weng et al. 2019; Yang et al. 2022; Zhou et al. 2024]. They first model a textured 3D mesh from the input and further model the 3D motion either by rigging the character to the pre-defined animation-ready 3D assets or by manually dragging the mesh parts in their interface. However, their main focus is 3D character modeling, while we concentrate on creating high-quality 3D animations from sketch storyboards by understanding the motion cues in the input sketches.

Human Motion Synthesis. Human motion synthesis focuses on generating diverse and realistic human-like movements. Traditional approaches rely on concatenation and retrieval techniques, including motion graphs [Heck and Gleicher 2007; Kovar et al. 2023; Shin and Oh 2006] and motion matching [Büttner and Clavet 2015]. Recently, learning-based methods have been widely adopted to generate high-quality and realistic motions, utilizing models such as GANs [Ghosh et al. 2021; Li et al. 2022], autoregressive GPTs [Guo et al. 2022b; Jiang et al. 2023; Zhang et al. 2023b, 2024], generative

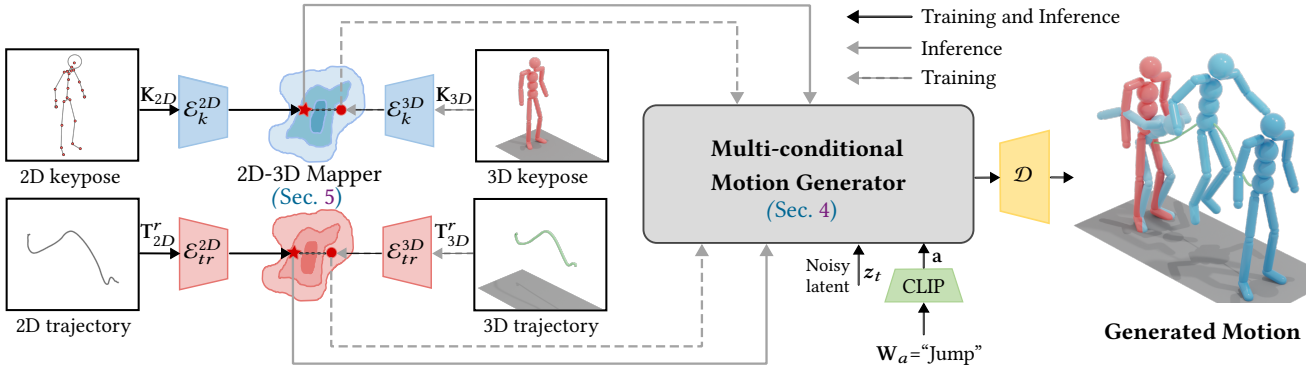


Fig. 3. **Overview of *Sketch2Anim*.** Our pipeline consists of two core modules - the multi-conditional motion generator (Sec. 4) and the 2D-3D neural mapper (Sec. 5). Instead of directly lifting the 2D keypose and trajectory into their 3D counterparts, we train a neural mapper dedicated to aligning the two domains in the embedding space. Because of this shared embedding, it enables the employment of more informative and precise 3D keyposes and trajectories as the motion conditions in the motion generator, while exploiting the 2D keypose and trajectory detected from the sketch storyboard at inference time. The legend indicates the data flow at training and inference of both modules. See the following sections for detailed technical designs.

masked Transfomers [Guo et al. 2024; Li et al. 2024b; Pinyoanuntapong et al. 2024b], and diffusion models [Chen et al. 2023; Tevet et al. 2023; Zhang et al. 2022; Zhou et al. 2025].

Driven by the efficacy of diffusion models for conditioning, several works utilize diffusion models to enable motion in-betweening [Agrawal et al. 2024; Cohan et al. 2024], human-object interactions [Li et al. 2025, 2023], and trajectories control [Dai et al. 2025; Karunratanakul et al. 2023; Wan et al. 2023; Xie et al. 2024]. To harness the capability of pre-trained motion diffusion models, OmniControl [Xie et al. 2024] employs a ControlNet [Zhang et al. 2023a] to guide these models in generating motions that adhere to 3D joint trajectories. Similarly, SMooDi [Zhong et al. 2025] employs ControlNet to steer these models for stylized motion generation. However, both methods utilize ControlNet for a single condition. When handling multiple conditions, how to effectively ensemble multiple ControlNets remains an open problem, even in the image and video domains. In this work, we propose a trajectory-aware keypose adapter designed to work alongside the trajectory ControlNet, guiding the pre-trained diffusion model to accommodate multiple conditions.

In terms of multiple conditions, SKEL-Betweener [Agrawal et al. 2024] is recently proposed and able to accept both the 3D keypose and 3D trajectory as conditions in its motion in-betweening task to synthesize novel motions using their skeletal transformer. It is worth noting that their start and end keyposes are in the 3D format and well-placed spatially with correct orientations. However, in our unique setting, it is difficult to obtain the 3D keypose and trajectory from 2D sketches, and impossible to derive the global placement information from the individually drawn storyboard frames, which differentiates us from them, making their techniques inapplicable.

3 Overview

Our novel approach produces high-quality 3D animations directly from a 2D sketch storyboard.

Storyboard. Figs. 1 and 2 display examples of 2D storyboards. To be clear, in our setting, for each frame, we only consider *character strokes* indicating the static pose, *joint trajectory strokes* (if any)

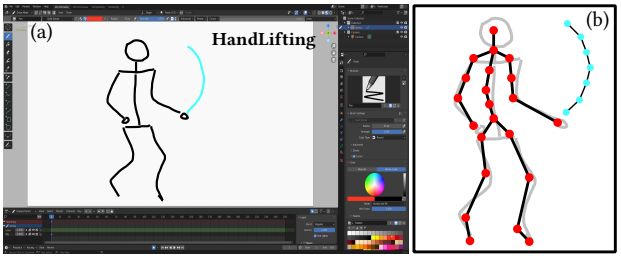


Fig. 4. **User interface and joint detection.** (a) In the interface, the white canvas is the main drawing area. Users can sketch or type the action word. We use different pens for character strokes and trajectories. (b) We use Sketch2Pose to detect the 2D joint points (red points), which serve as our input, instead of the raw sketch in (a); and we draw the line segments between joint points only for intuitive visualization. The uniformly re-sampled trajectory points are drawn in cyan.

expressing dynamic movements, and the *action word* describing the motion content, excluding decorative sketches depicting the scene.

We have developed a user interface as a Blender add-on (see Fig. 4(a)). Character strokes in black and joint trajectories in cyan are traced separately in a *known* camera view. Users can either sketch the action word or type the text. For simplicity, the character is drawn in the stick-figure style with circles and line segments indicating the head, body, and limbs. Storyboard frames are drawn sequentially from the 3/4 view without global placements, and users finally execute *Sketch2Anim* to obtain the complete animation.

Input. Suppose there are f sketches in the storyboard, and for each of them, the character strokes are rendered into a rasterized image. The image is first fed into Sketch2Pose [Brodt and Bessmeltsev 2022] to obtain a set of 2D joint points. Following the joint setting in the SMPL body [Loper et al. 2023], we simply augmented the point set to obtain the input 2D joints indicating the static keypose $\{(x_j, y_j)\}_1^{J=22}$, where J is the number of joints. As for the joint trajectory, the user is able to draw multiple strokes for multiple joints. We trace and downsample each stroke, and attach it to the corresponding joint by closest points checking, resulting in a set of 2D trajectory points $\{(x_i^j, y_i^j)\}_{i=1}^{t_r}$. Note that, the superscript j

represents the joint index, and t_r varies based on the length of the trajectory stroke. An example of joints and trajectory points is shown in Fig. 4(b). Lastly, the action word $\mathbf{W}_a$ is pure text input. If users sketch the word, we use an OCR [Bautista and Atienza 2022] approach to detect it.

Output. Given the 2D joint points, 2D trajectory points, and the action word from a single sketch frame, the algorithm generates a 3D motion clip $\boldsymbol{m}$. By default, it consists of $N = 40$ frames (*i.e.*, 2*s*) in our setting. A final 3D motion by blending the f motion clips is produced, denoted as $\mathcal{M}$. For visualization purposes, we occasionally employ special characters (*i.e.*, 'Michelle' and 'Clair' from Mixamo characters in Figs. 1 and 11, respectively) by retargeting [Nkeeline 2024]. Unless otherwise stated, a default capsule-bone human model is exploited throughout the paper.

Feature representation. We adopt a unified matrix representation for the joint (*i.e.*, keypose), trajectory, and motion. Specifically, the 2D keypose (22 2D joints) is filled into the *first* or *last* motion frame, denoted as $\mathbf{K}_{2D} \in \mathbb{R}^{N\times J\times 2}$, and its 3D counterpart is represented as $\mathbf{K}_{3D} \in \mathbb{R}^{N\times J\times 3}$. Similarly, joint positions along successive frames represent the trajectory of a specific joint. The trajectory points for the joint j are filled into the *first* or *last* t_r motion frames of the j-th joint, denoted as $\mathbf{T}^r_{2D} \in \mathbb{R}^{N\times J\times 2}$, while the 3D counterpart is denoted as $\mathbf{T}^r_{3D} \in \mathbb{R}^{N\times J\times 3}$. Note that, $\mathbf{T}^r_{2D}$ and $\mathbf{T}^r_{3D}$ can contain more than one trajectory. To be more clear, if the trajectory stroke starts from its respective joint, the keypose condition is placed at the $1^{st}/40$ frame of the motion clip, while the trajectory is placed at the first $t_r/40$ frames of the motion clip. Otherwise, if the trajectory stroke ends at its respective joint, the keypose condition is at the $40^{th}/40$ frame and the trajectory is placed at the last $t_r/40$ frames.

Following HumanML3D [Guo et al. 2022a], we adopt a redundant motion representation that includes both local and global positions and rotations. The representation is also in a unified matrix format. Thus, the motion clip $\boldsymbol{m}$ and the final motion $\mathcal{M}$ is in the space of $\mathbb{R}^{N\times D}$ and $\mathbb{R}^{f\times N\times D}$, respectively. $D = 263$ is the total dimension of the joint-based motion representation.

The algorithm. An overview of our algorithm composed of two core modules is displayed in Fig. 3. The first module (Sec. 4) is a multi-conditional motion generator, which is built upon the motion diffusion model [Chen et al. 2023]. It performs diffusion and denoising steps within a pre-trained latent space. During training, we use the 3D joint points (*i.e.*, the keypose) and 3D trajectory points as the surrogates for the 2D inputs. Particularly, the designed trajectory ControlNet and the keypose adapter are integrated into the motion diffusion model, enabling robust and precise multiple motion control. The second module (Sec. 5) exploits a pair of encoders to align the 2D and 3D keypose and trajectory in the shared embedding space, bridging two domains. In our implementation, the algorithm runs sequentially for each sketch frame, and the resulting motion clip is finally blended (Sec. 6).

4 Multi-conditional Motion Generator

The core of our system is a multi-conditional motion latent diffusion model in which the diffusion process operates within the latent space. Fig. 5 displays an overview of this module.

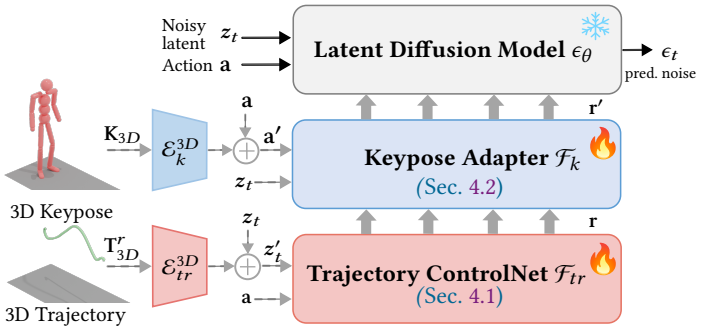


Fig. 5. **The structure of our motion generator.** A trajectory ControlNet is exploited to incorporate the trajectory control, while a keypose adapter is placed between the ControlNet and the diffusion model to inject the keypose condition. The same color and style of the common elements and data flow as in Fig. 3 are used. Refer to Sec. 4 for a detailed explanation.

Let ϵ_θ denote the latent denoiser (a UNet parameterized by θ), and $\{z_t\}^T_{t=0}$ denote the sequence of noisy latents, where z_T is a Gaussian noise. Given a 3D keypose $\mathbf{K}_{3D}$, joint trajectories $\mathbf{T}^r_{3D}$, and an action word embedding $\mathbf{a}$ (obtained from $\mathbf{W}_a$ via CLIP [Radford et al. 2021]), the denoising process at step t ($0 < t \leq T$) is defined as $\epsilon_t = \epsilon_\theta(z_t, t, \mathbf{a}, \mathbf{K}_{3D}, \mathbf{T}^r_{3D})$. A cleaner noisy latent z_{t-1} is obtained by subtracting ϵ_t from z_t. This denoising step is repeated for T iterations until a clean latent z_0 is generated. Finally, z_0 is decoded by a motion decoder $\mathcal{D}$ into a realistic motion sequence $\boldsymbol{m} \in \mathbb{R}^{N\times D}$ that adheres to the given keypose and joint trajectory conditions. In the following, we introduce our novel technical design to effectively inject these conditions.

4.1 Trajectory ControlNet

Inspired by the recent success of ControlNet in motion control [Diller and Dai 2024; Xie et al. 2024; Zhong et al. 2025], we design a trajectory ControlNet $\mathcal{F}_{\text{tr}}$ to condition the model on joint trajectories. Specifically, it consists of a trainable copy of the Transformer encoder from the latent diffusion model and each layer is appended with a zero-initialized linear layer $\mathcal{Z}$. An independent trajectory encoder $\mathcal{E}^{3D}_{tr}$ is employed to extract the trajectory embedding from the joint trajectory $\mathbf{T}^r_{3D}$. Considering that controlling joint trajectories requires refining both global spatial and temporal information, we incorporate the joint trajectory embedding into the noised latent z_t to create the conditioned noised latent z'_t, which is subsequently fed into the trajectory ControlNet to predict the residual feature $\mathbf{r}$.

Formally, the whole process can be written as:

$$z'_t = z_t + \mathcal{E}^{3D}_{tr}(\mathbf{T}^r_{3D}), \quad \mathbf{r} = \mathcal{F}_{\text{tr}}(z'_t, t, \mathbf{a}). \tag{1}$$

As training progresses, the residual features $\mathbf{r}$ are applied to the corresponding layers in the motion diffusion model ϵ_θ, implicitly controlling the generated motion.

4.2 Trajetory-aware Keypose Adapter

Unlike the motion in-betweening task [Agrawal et al. 2024; Cohan et al. 2024; Harvey et al. 2020; Starke et al. 2023], where a globally placed start and end keyposes are known, our single keypose in each

sketch frame is drawn independently without the global arrangement. In essence, the keypose of our task is a representative pose during the action's execution, providing a more visual and concrete interpretation of the action word. Moreover, the keypose does not provide global positional information. Instead, it relies solely on its relative timing in relation to the joint trajectories.

In order to inject the keypose information, a straightforward approach is to introduce a keypose ControlNet dedicated to learning keypose constraints. This keypose ControlNet is trained jointly with the trajectory ControlNet, and their residual features are combined before being incorporated into the motion diffusion model. However, the two controls focus on distinct aspects: the trajectory captures dynamic features and global temporal relationships, while the keypose emphasizes static features and local pose constraints. This divergence creates different levels of learning complexity for each condition, making it difficult to balance their loss weights effectively during training (see Sec. 7.2). Moreover, fusing the residual features from multiple ControlNets remains an open challenge, not only in the motion domain [Bian et al. 2024] but also in the image generation domain [Mou et al. 2024; Qin et al. 2023; Zhao et al. 2024].

To address these challenges, we propose a trajectory-aware keypose adapter, $\mathcal{F}_{\text{k}}$, that works alongside the trajectory ControlNet $\mathcal{F}_{\text{tr}}$ to simultaneously inject both constraints. The keypose adapter accepts the residual output features from the trajectory ControlNet, embedding the keypose condition into the trajectory residual features to achieve deeper interaction and fusion between the two conditions. Specifically, the keypose adapter $\mathcal{F}_{\text{k}}$ is also a trainable copy of the Transformer encoder of the latent diffusion model. We exploit a keypose encoder $\mathcal{E}_k^{3D}$ to extract embeddings from the 3D keypose $\mathbf{K}_{3D}$. Particularly, the keypose complements the action description by providing a detailed interpretation of how the action is executed, we therefore integrate the keypose embedding with the action embedding $\mathbf{a}$ to form a grounded action embedding $\mathbf{a}'$:

$$\boldsymbol{a}' = \boldsymbol{a} + \mathcal{E}_k^{3D}(\mathbf{K}_{3D}). \tag{2}$$

Since the trajectory control has already established a global dynamic motion pattern, the keypose adapter only needs to add local static constraints on top of it. Therefore, the adapter takes the grounded action embedding $\mathbf{a}'$, the noised latent z_t, and the residual features $\mathbf{r}$ from the trajectory ControlNet as inputs. It learns new residual feature corrections $\mathbf{r}'$ for corresponding layers in the motion diffusion model, ensuring the generated motions adhere to both keypose and trajectory constraints. The process can be expressed as:

$$\mathbf{r}' = \mathcal{F}_{\text{k}}(z_t, t, \mathbf{r}, \boldsymbol{a}'), \tag{3}$$

$$\epsilon_t = \epsilon_\theta(z_t, t, \mathbf{a}) + \mathcal{Z}(\mathbf{r}'). \tag{4}$$

Through iterative denoising, we obtain the latent code z_0, which is decoded into a 3D motion using the motion decoder $\mathcal{D}$.

4.3 Training

Following [Zhang et al. 2023a], we freeze the parameters of the motion diffusion model and solely train the trajectory ControlNet and keypose adapter using the standard noise reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \mathbb{E}_{\epsilon,z}\left[\left\|\epsilon_\theta(z_t, t, \mathbf{a}, \mathbf{T}_{3D}^r, \mathbf{K}_{3D}) - \epsilon\right\|_2^2\right], \tag{5}$$

where $\epsilon \sim \mathcal{N}(0, \mathbf{I})$ represents the ground-truth noise added to z_0. However, solely relying on reconstruction supervision in the latent space is insufficient for the model to effectively learn accurate conditioning on the joint trajectory and keypose. Therefore, we calculate the L_2 distance between the joint trajectories of the generated clean motion $\hat{\mathbf{x}}_0 \in \mathbb{R}^{N\times D}$ at the denoising step t and the ground-truth motion $\mathbf{x}_0 \in \mathbb{R}^{N\times D}$:

$$\mathcal{L}_{\text{tr}} = \mathbb{E}_{\hat{\mathbf{x}}_0}\left[\frac{\sum_i \sum_j m_{ij}\|R(\hat{\mathbf{x}}_0)_{ij} - R(\mathbf{x}_0)_{ij}\|_2^2}{\sum_i \sum_j m_{ij}}\right], \tag{6}$$

where $R(\cdot)$ converts the redundant motion representation into its joint global absolute locations in the space of $\mathbb{R}^{N\times J\times 3}$, and $m_{ij} \in \{0, 1\}$ is the binary joint trajectory mask at frame i for the joint j. The generated motion $\hat{\boldsymbol{x}}_0$ is obtained by first converting the denoising output latent z_t into the predicted clean latent as shown below:

$$\hat{z}_0 = \frac{z_t - \sqrt{1-\alpha_t}\,\epsilon_\theta(z_t, t, \mathbf{a}, \mathbf{T}_{3D}^r, \mathbf{K}_{3D})}{\sqrt{\alpha_t}}, \tag{7}$$

where α_t denotes the pre-defined noise scale in the forward process of the diffusion model. The predicted clean latent $\hat{z}_0$ is then fed into the motion decoder $\mathcal{D}$ to obtain the generated motion. Similarly, we apply the keypose constraint in the space of $\mathbb{R}^{N\times J\times 3}$ to ensure that the generated motion adheres to the keypose constraints:

$$\mathcal{L}_{\text{key}} = \mathbb{E}_{\hat{\mathbf{x}}_0}\left[\frac{\sum_i m'_i\|R'(\hat{\mathbf{x}}_0)_i - R'(\mathbf{x}_0)_i\|_2^2}{\sum_i m'_i}\right], \tag{8}$$

where $m_i \in \{0, 1\}$ is the binary keypose mask at frame i. To enforce the matching of the ground truth and generated keyposes without considering their global placement, in $R'(\cdot)$, we therefore first use $R(\cdot)$ to convert the motion to the global absolute locations ($\mathbb{R}^{N\times J\times 3}$), then subtract the root position at frame i from each joint's position.

Overall, the training loss function for the trajectory ControlNet and keypose adapter is defined as follows:

$$\mathcal{L}_{\text{gen}} = \mathcal{L}_{\text{recon}} + \lambda_{\text{tr}}\mathcal{L}_{\text{tr}} + \lambda_{\text{key}}\mathcal{L}_{\text{key}}. \tag{9}$$

5 2D-3D Trajectory and Keypose Alignment

Having the 3D conditional motion generator, instead of formally lifting the 2D keypose and trajectory into the 3D space, we propose aligning them in their respective embedding spaces. Specifically, we train a separate 2D trajectory encoder $\mathcal{E}_{tr}^{2D}$ and keypose encoder $\mathcal{E}_k^{2D}$ to align with their frozen 3D counterparts, $\mathcal{E}_{tr}^{3D}$ and $\mathcal{E}_k^{3D}$ (Sec. 4) after training the motion diffusion model. Leveraging a shared embedding space between 2D and 3D conditions, we can seamlessly extract 2D embeddings from the storyboard for the conditional motion diffusion model to generate motions.

Training. Given a batch of paired 2D and 3D embeddings for keyposes and trajectories, denoted as $\mathcal{B}_k = (\mathbf{s}_{i,k}^{3D}, \mathbf{s}_{i,k}^{2D})_{i=1}^B$ for keyposes and $\mathcal{B}_{tr} = (\mathbf{s}_{i,tr}^{3D}, \mathbf{s}_{i,tr}^{2D})_{i=1}^B$ for trajectories (B is the batch size), we enforce the paired 2D and 3D embeddings to be as close as possible:

$$\mathcal{L}_{\text{match}} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{y\in\{tr,k\}}\|\mathbf{s}_{i,y}^{3D} - \mathbf{s}_{i,y}^{2D}\|_2^2. \tag{10}$$

While the above loss encourages the embeddings of 2D and 3D pairs to be close, it does not guarantee the motions generated based on 2D conditions are realistic. Therefore, we introduce a noise reconstruction loss for 2D conditions as a motion regularizer:

$$\mathcal{L}_{\text{recon}} = \mathbb{E}_{\epsilon,z}\left[\left\|\epsilon_\theta(z_t, t, \mathbf{a}, \mathbf{T}^r_{2D}, \mathbf{K}_{2D}) - \epsilon\right\|_2^2\right]. \tag{11}$$

This term couples alignment with generation, yielding 2D embeddings that both align with their 3D embeddings and are seamlessly utilized by the diffusion model for plausible motion generation.

Additionally, inspired by CLIP-style contrastive learning [Radford et al. 2021], we exploit a complementary contrastive learning loss to improve the training efficiency. The training objective is to maximize the similarity between the embeddings $(\mathbf{s}^{3D}_{i,k}, \mathbf{s}^{2D}_{i,k})$ for keyposes and $(\mathbf{s}^{3D}_{tr}, \mathbf{s}^{2D}_{tr})$ for trajectories, while minimizing the similarity of incorrect pairs. The learning object can be expressed by:

$$\mathcal{L}_{\text{contrast}} = -\frac{1}{B}\sum_{i=1}^{B}\sum_{y\in\{tr,k\}} \log \frac{\exp(\text{sim}(\mathbf{s}^{3D}_{i,y}, \mathbf{s}^{2D}_{i,y})/\tau_1)}{\sum_{j=1}^{B}\exp(\text{sim}(\mathbf{s}^{3D}_{i,y}, \mathbf{s}^{2D}_{j,y})/\tau_1)}, \tag{12}$$

where $\text{sim}(\cdot,\cdot)$ is the cosine similarity function, and τ_1 is a temperature hyperparameter.

The overall training loss function for aligning the embeddings is defined as follows:

$$\mathcal{L}_{\text{align}} = \mathcal{L}_{\text{match}} + \lambda_r \mathcal{L}_{\text{recon}} + \lambda_c \mathcal{L}_{\text{contrast}}. \tag{13}$$

6 Inference Motion Generation

Having the aligned embedding spaces, during inference, we directly input the obtained 2D keypose and joint trajectories (Sec. 3) to the multi-conditional motion diffusion model. Furthermore, we apply classifier-free guidance [Ho and Salimans 2022], and the predicted noise $\epsilon_\theta(z_t, t, \mathbf{a}, \mathbf{T}^r_{2D}, \mathbf{K}_{2D})$ is computed as follows:

$$\epsilon_\theta(z_t, t, \mathbf{a}) + w_c\big(\epsilon_\theta(z_t, t, \mathbf{a}, \mathbf{T}^r_{2D}, \mathbf{K}_{2D}) - \epsilon_\theta(z_t, t, \mathbf{a})\big), \tag{14}$$

where w_c is the scale factor used to control the strength of the conditional guidance.

Additionally, inspired by classifier guidance [Dhariwal and Nichol 2021] and loss-guided diffusion [Song et al. 2023; Wang et al. 2023], we employ inference guidance to enhance the accuracy of following 2D joint trajectories. The idea is to minimize the discrepancy between the projected trajectories of the generated motion and the 2D trajectories. See supplementary for the pseudo-codes and inference guidance details.

Motion Blending. After generating a sequence of motion clips from storyboard frames, we apply a post-processing step to create seamless transitions between adjacent clips. Briefly, given two adjacent motion segments, we first linearly blend them to initialize a transition motion, and then refine it using a deterministic DDIM inversion [Song et al. 2020] followed by a guided denoising process. We iteratively run the above blending method on all the motion clips within a storyboard composing them into a complete animation. The implementation details are provided in the supplementary.

7 Results and Discussion

With our new *Sketch2Anim* system, we have sketched several storyboards with varying motion complexity. Examples are shown in Figs. 1 and 6, with rough sketches (*e.g.*, irregular strokes and disproportional body parts), demonstrating vivid motions from 'Bow' to the sophisticated 'Backflip' and 'Kneel'. It is worth noting that the third sketch frame in Fig. 1 and the first sketch in Fig. 6 include two trajectories for two different joints. For better visualization, we only show the motion clips before blending in Fig. 6. Please refer to the *supplemental video* for complete animations. Extensive comparisons (Sec. 7.1) and ablation study (Sec. 7.2) validate the effectiveness and core technical design choices of our approach. A user perceptual study (Sec. 7.3) also confirms our superior performance. We highly encourage the reader to check the supplemental video, which better demonstrates our approach.

Dataset. We train and evaluate our system on the HumanML3D dataset [Guo et al. 2022a], containing 14,646 motions with 44,970 corresponding motion annotations. Following the processing approach outlined in Guo et al. [2022a], we preprocess the HumanML3D dataset to obtain the redundant motion representations (Sec. 3). Given a motion, we further process it to select a representative keypose corresponding to the detected action word, and project the 3D motion to obtain the 2D keypose and trajectory with data augmentation (*i.e.*, camera view augmentation, joint perturbation, and body proportion perturbation). Examples of our synthetic dataset and its augmented variations are shown in Fig. 7, and more details of the dataset processing can be found in the supplementary.

Evaluation metrics. We quantitatively evaluate the performance of our framework from three aspects - *motion realism*, *control accuracy*, and *text-motion matching*. To assess motion realism, we use the Frechet inception distance (FID) to compare the feature distributions of generated motions with real motions. Furthermore, given the prevalent foot skating issues in kinematics-based motion generation methods, we incorporate the foot skating ratio [Karunratanakul et al. 2023] as an additional metric to enhance the evaluation of motion quality. To evaluate the control accuracy, we measure the alignment of both the 2D and 3D conditional keyposes and trajectories, respectively. Note that the resulting 2D keypose and trajectory are obtained by projecting the generated motion. For the keypose, we measure the mean per joint position error, denoted as MPJPE-2D and MPJPE-3D. For the trajectory, we similarly measure the average error of the keypose joints, denoted as Avg. Err.-2D and Avg. Err.-3D, respectively. Although we do not input the 3D conditions at testing time, we report 3D metrics as a complementary measurement. For text-motion matching, we use motion-retrieval precision (R-precision) to measure the relevancy of the generated motion to its action word. Additionally, we compute multi-modal distance (MM Dist), defined as the average Euclidean distance between the generated motion feature and the corresponding text features.

Implementation Details. All models are implemented in Pytorch and trained and tested on a single NVIDIA RTX4090 GPU. Training the trajectory ControlNet and keypose adaptor requires approximately 1000 epochs (12 hours), while aligning the 2D and 3D keypose and trajectory embedding spaces takes around 100 epochs (3 hours). AdamW [Loshchilov and Hutter 2017] optimizer is exploited for both trainings with the learning rate of $1e^{-5}$. All hyperparameters in the training are set to 1. The weight of classifier-free guidance, w_c,

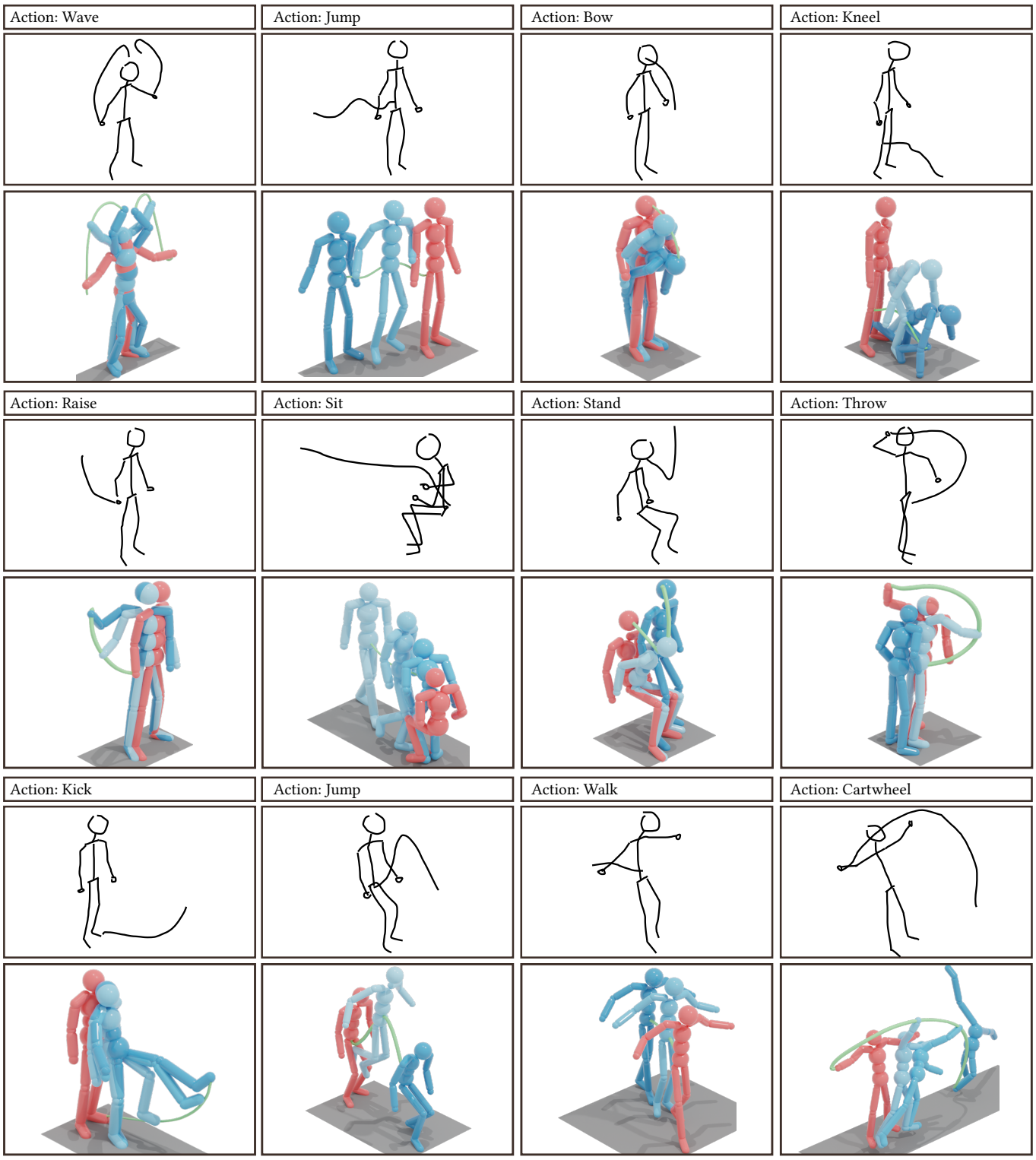


Fig. 6. **Result Gallery**. Three more storyboards are successfully transferred into their high-quality animations. Only the motion clips before blending are shown for clear visualization of keyposes, trajectories, and motion. The complete animation after blending can be seen in the supplemental video. For each clip, we highlight the keypose in red, while the trajectories are spatial curves shown in green. The motions of "Kneel", "Cartwheel", and "Sit" are usually hard to generate, and our results are of high quality conforming with the sketches.

Table 1. Quantitative analysis of *Sketch2Anim* (Ours) and three baseline models defined in Sec. 7.1 on the HumanML3D dataset. Evaluation metrics on motion realism, control accuracy, and text-motion match are presented. Following OmniControl [Xie et al. 2024], we report both the average error of all joints (Average) and their random combination (Cross). The best results are highlighted.

Condition	Method	Realism		Control Accuracy				Text-Motion Matching	
		FID ↓	Foot Skating ↓	MPJPE-2D ↓	MPJPE-3D ↓	Avg. Err.-2D ↓	Avg. Err.-3D ↓	MM Dist ↓	R-precision (Top-3) ↑
Average	Motion Retrieval	0.690	**0.0640**	0.0570	0.0760	0.290	0.410	4.060	0.640
	Lift-and-Control	0.979	0.0885	0.0537	0.0712	0.261	0.340	3.297	0.752
	Direct 2D-to-Motion	2.553	0.112	0.0403	0.0552	0.193	0.275	3.723	0.687
	Ours	**0.525**	0.103	**0.0360**	**0.0478**	**0.0867**	**0.134**	**3.077**	**0.802**
Cross	Motion Retrieval	**0.103**	**0.0668**	0.0549	0.0730	0.3071	0.423	3.405	0.724
	Lift-and-Control	0.738	0.101	0.0505	0.0671	0.209	0.283	3.135	0.778
	Direct 2D-to-Motion	2.310	0.123	0.0403	0.0555	0.189	0.266	3.606	0.709
	Ours	0.577	0.102	**0.0329**	**0.0462**	**0.0792**	**0.132**	**3.042**	**0.796**

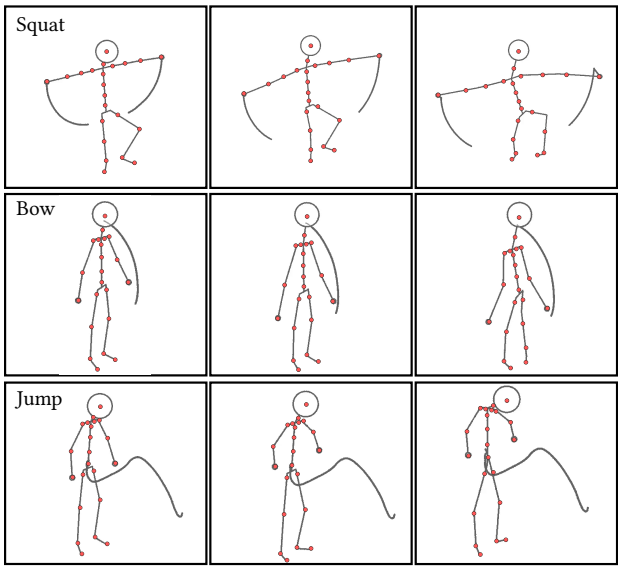


Fig. 7. **Synthetic data and its augmentation.** Three examples from our dataset are shown in the three rows. As introduced in Sec. 3, our input is the 2D joint points (*i.e.*, red dots), and we draw the stick figure for better visualization. Column (a) demonstrates input data with perfect joint points and body part proportions. While in column (b), random perturbations are applied to the body part proportions (*e.g.*, the enlarged neck in "Bow" and arms in "Squat"), resulting in disproportional keyposes. A further random perturbation is added to a few joint points in (c) to increase the deviation of joints to mimic the detected joints from user raw sketches.

is set to 7.5. We update z_t using inference guidance for 4 iterations at each denoising step. At inference time, on average, the algorithm takes around 0.5s to produce a motion clip with 40 frames. The blending of two motion clips takes around 1s. Detailed running time breakdowns can be found in the supplementary.

7.1 Comparison

Instead of evaluating the full algorithm consisting of motion generation and blending, we only consider the core problem of motion clip generation from a single storyboard frame in our comparison. Formally, given the 2D conditional keypose, trajectories, and action word, we compare the generated motion with competitors.

Competitive methods. Since no existing method directly addresses our conditional motion generation task, we developed three baseline approaches for comparison.

- Motion retrieval: since sketch-based motion retrieval works [Thorne et al. 2004; Yoo et al. 2014] do not have available code, we thus adapt a more recent state-of-the-art learning-based motion retrieval method TMR [Petrovich et al. 2023], which only takes as input the text description to identify the closest motion clip from the dataset. We further develop it to accept additional 2D keyposes and joint trajectories. The resulting method serves as a simple yet effective benchmark for comparison.
- Lift-and-control: we trained an additional network to lift 2D keyposes and trajectories to the 3D space. These 3D representations are then fed into the multi-conditional motion diffusion model for motion synthesis. The lifting network is implemented based on MotionBERT [Zhu et al. 2023].
- Direct 2D-to-motion: instead of using 3D inputs as surrogates in the motion generator, a multi-conditional motion diffusion model is trained directly on 2D keyposes, joint trajectories, and action words.

For more details on the implementation of these baselines, please refer to the supplementary.

Evaluation. Table 1 compares *Sketch2Anim* with three baseline methods on motion realism, control accuracy, and text-motion matching metrics on our testing dataset. Following OmniControl [Xie et al. 2024], we report the metrics in terms of the average error over all joints in the first section (denoted as Average), and also the mean error on cross-combinations of all joints, with one combination randomly sampled per test example in the second section (denoted as Cross). The Motion Retrieval method (1^{st} and 5^{th} rows) nearly achieves the best performance on realism metrics, as they utilize real motion clips retrieved from pre-collected datasets. However, their reliance on the diversity and scope of the pre-collected data limits their performance in both control accuracy and text-motion matching metrics. Compared to the Direct 2D-to-Motion method (3^{rd} and 7^{th} rows), *Sketch2Anim* demonstrates substantial improvements in control accuracy, leading to more precise and higher quality motions (*e.g.*, 5 times lower FID scores). The advancements highlight the significance of using 3D inputs as surrogates to guide motion

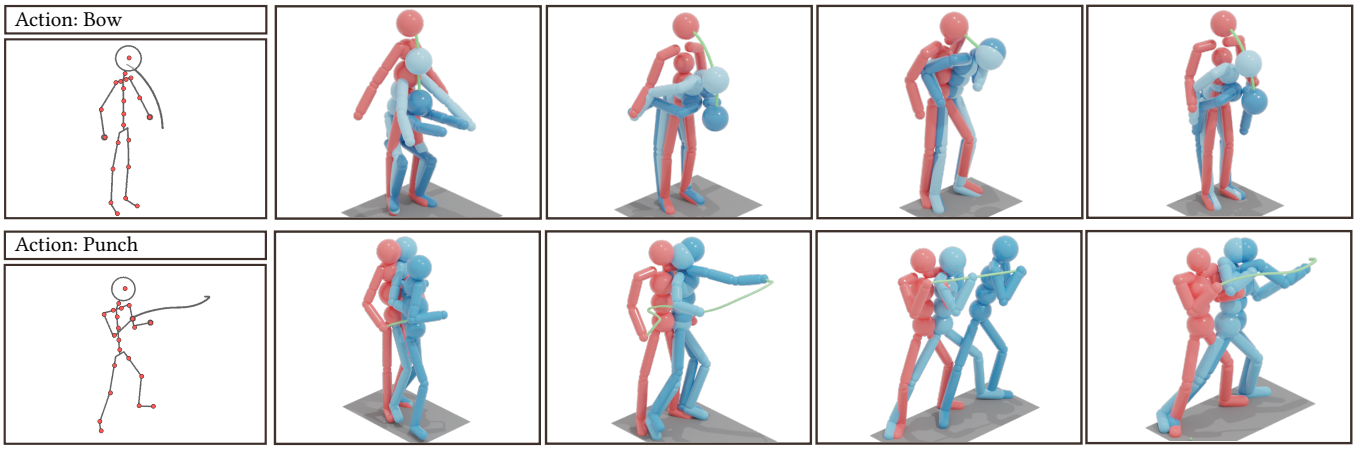


Fig. 8. **Visual comparison.** Given two frames of the storyboard (*i.e.*, "Kick" and "Punch"), corresponding results from competitors and ours are shown. Our results faithfully adhere to each frame, while others either have inaccurate joint trajectories or deviated keyposes, leading to unexpected motions.

synthesis. Similarly, as displayed in the 2^{nd} and 6^{th} rows, the Lift-and-Control method demonstrates reasonable performance across all metrics but is consistently inferior compared to *Sketch2Anim*. Our improvements underscore the advantages of aligning 2D-3D trajectory and keypose embeddings, which effectively capture both spatial and temporal features from the 2D inputs while mitigating potential errors introduced by the lifting step.

The visual comparison is presented in Fig. 8. The Motion-Retrieval method can obtain a motion in high realism, but it fails to conform with the sketch pose and trajectory. Note that the punching hand of the retrieved motion is even wrong for the action. Both the keypose and trajectory of the resulting motions from the Lift-and-Control method are incorrect, due to the inherent inaccuracy of lifting 2D to 3D. The Direct 2D-to-Motion method produces the worst results, either producing weird motions (*e.g.*, "Bow") or failing to match the action word. This is mainly because of the considerable domain gap between 2D sketches and 3D animations. Our results outperform all competitors significantly, the animation is of high quality in terms of motion realism, control accuracy, and text-motion matching. *These visual differences are best viewed in our supplemental video.*

7.2 Ablation Study

We conducted a series of ablation studies to validate the key design choices of our 3D conditional motion generator, with primary focuses on the network modules, leaving the analysis of loss terms in the supplementary. Note that we use the 3D keypose and trajectory in this experiment only to validate the motion generator.

Single ControlNet. As stated in Sec. 3, the static and local keypose and the dynamic and global trajectory have the same point-based representation. Theoretically, it is possible to assemble them together in a unified matrix, and then feed it to a single ControlNet as the condition for motion generation.

As can be seen in Tab. 2, this scheme achieves the worst performance across almost all realism and control accuracy metrics. This significant performance degradation (*e.g.*, more than 50% drop off) highlights the limitations of a single ControlNet, which struggles to encode both keypose and trajectory conditions uniformly, especially because keyposes are significantly sparser than trajectories.

Double ControlNet. Rather than using an adapter, a separate ControlNet offers a more straightforward approach for incorporating the keypose condition. In this configuration, the keypose feature is still combined with the action word, but the keypose ControlNet operates independently, without receiving residual features from the trajectory ControlNet, keeping the two modules parallel. The residual features generated by the two ControlNets are added together and subsequently fed into the motion diffusion model.

As shown in Tab. 2, the double ControlNet scheme performs better than a single ControlNet but worse than ours. We observe that the two ControlNets compete with each other, resulting in oscillatory training curves. In contrast, the adapter focuses on refining the keypose at a specific timestep, leveraging the globally coherent motion provided by the trajectory. On the other hand, the convergence speeds of the two conditions differ significantly, with keyposes converging nearly ten times faster than joint trajectories. This disparity poses a significant challenge in dynamically balancing the loss weights for keypose and joint trajectory during training.

Global keypose control. In Eq. 2, we add the keypose embedding to the action word embedding, because we consider keypose as a detailed interpretation of how the action is executed. To validate the effectiveness of this design, we evaluate the performance of a variant solution where the keypose embedding is added to the noise latent, enabling more global control. Formally, we replace the original Eqs. 2 and 3 with $\boldsymbol{z}'_t = \boldsymbol{z}_t + \mathcal{E}^{3D}_k(\mathbf{K}_{3D})$ and $\mathbf{r}' = \mathcal{F}_k(\boldsymbol{z}'_t, t, \mathbf{r}, \boldsymbol{a})$, while keeping the remaining processes the same.

As shown in Tab. 2, the full method outperforms the global keypose control scheme on almost all metrics. Notably, the full method achieves 10% and 4.9% improvements in FID under the Average and the Cross evaluation modes, highlighting the effectiveness of keypose instantiation for interpreting action words. Furthermore, in terms of trajectory control accuracy (Avg. Err.-3D), the global keypose control scheme shows degradations of 17.2% and 11.0% under

Table 2. Ablation studies on multi-conditional motion diffusion model design using the HumanML3D dataset. Sec. 7.2 defines all the variants, and we use the *3D keypose* and *3D trajectory* as conditions in this experiment, thus only reporting the control accuracy in its 3D format. Best results are highlighted.

Condition	Method	Realism		Control Accuracy		Text-Motion Matching	
		FID ↓	Foot Skating Ratio ↓	MPJPE-3D ↓	Avg. Err.-3D ↓	MM Dist ↓	R-Precision (Top-3) ↑
Average	Single ControlNet	0.746	0.107	0.0566	0.257	3.077	0.787
	Double ControlNet	0.608	**0.0933**	0.0428	0.246	3.046	0.790
	Global Keypose	0.462	0.0958	0.0413	0.190	2.932	0.806
	Full Method	**0.446**	0.0953	**0.0399**	**0.162**	**2.884**	**0.814**
Cross	Single ControlNet	0.612	0.107	0.0541	0.210	3.007	0.796
	Double ControlNet	0.433	0.0953	0.0418	0.199	2.950	0.803
	Global Keypose	0.389	0.114	0.0402	0.172	2.903	0.793
	Full Method	**0.370**	**0.0959**	**0.0397**	**0.155**	**2.897**	**0.815**

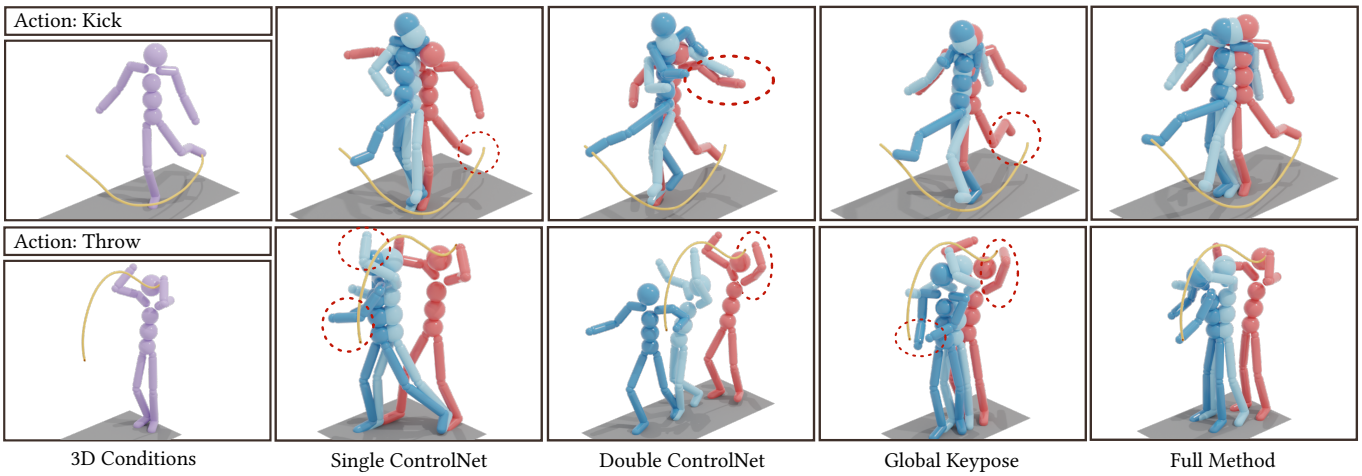


Fig. 9. **Visual results of the ablation study.** Given the conditional 3D keypose (*i.e.*, the purple character) and trajectory (*i.e.*, the yellow curve), resulting motions from alternative methods and ours are shown. Note that the input 3D trajectory is overlaid with the produced motions to help better spot the trajectory deviation. The top example demonstrates the poor matching ability of alternative methods regarding the trajectory (see the circled deviations), while the second example mainly displays the inaccurate keyposes from alternative methods.

the Average and Cross modes, respectively, indicating interference from the keypose on trajectory control.

A separate visual comparison is presented in Fig. 9, where we demonstrate the resulting motions from all method variants, given the same 3D keyposes and trajectories as the conditions. The alternative technical designs fail to explain the sketch keypose or trajectory, producing unsatisfactory motions, which is consistent with the metrics in Tab. 2.

7.3 User Evaluation

We conduct user perceptual studies using pairwise comparisons. In each test, participants are presented with a sketch storyboard frame and a pair of motion clips - one produced by *Sketch2Anim* and the other by a competing method (*i.e.*, Motion Retrieval, Lift-and-Control, and Direct 2D-to-Motion). For each pair of motions, participants are asked to evaluate three aspects - motion realism, trajectory accuracy, and keypose accuracy, as shown in the legend in Fig. 10. In total, we have invited 58 participants, and the percentage of times our approach is preferred over the competing methods is displayed in Fig. 10. Our *Sketch2Anim* performs favorably against Direct 2D-to-Motion for all three aspects, with 88%, 87%, and 90%,

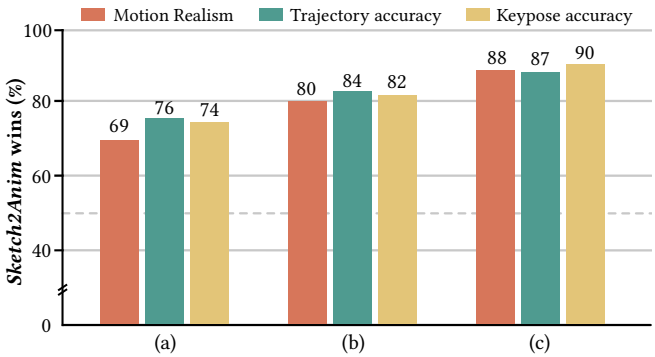


Fig. 10. **User perceptual study results.** The percentage of times our approach is preferred over (a) Motion Retrieval, (b) Lift-and-Control, and (c) Direct 2D-to-Motion is reported. When choosing from a pair of generated motions, users are asked to evaluate three aspects - motion realism, trajectory accuracy, and keypose accuracy. The higher the percentage (50% is a tie), the better our results.

respectively. Similarly, compared with Lift-and-Control, although the percentage our method wins drops around 6%, it is still considerably high (over 82%). Lastly, compared to Motion Retrieval, our

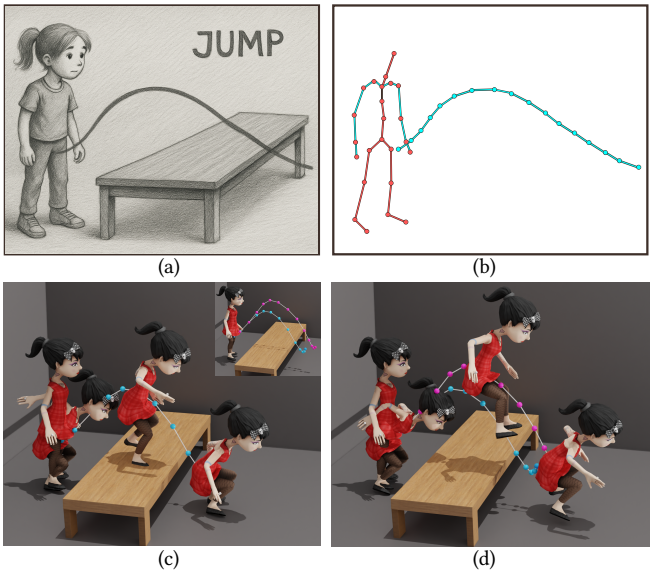


Fig. 11. **3D editing.** Given one frame of a real-world storyboard (a), we preprocess it to obtain the keypose (*i.e.*, 2D joint points in red) and the trajectory (*i.e.*, curve points in cyan) (b). Our *Sketch2Anim* produces the animation conditioned on the action word, 2D keypose, and the 2D trajectory (c). Note that we manually model the room with a table for visualization purposes, and the 3D trajectory from the resulting motion is highlighted with cyan points. If the user is unsatisfied with the motion (*e.g.*, the foot penetrates the table), the 3D trajectory can be further edited by dragging a few sample points to create a new 3D trajectory (purple points at the top-right corner in (c)), and the updated motion (d) is then re-generated based on the new 3D trajectory condition.

approach is preferred for motion realism only 69% of the time. While this is above 50% (indicating no tie), the advantage is not substantial. This outcome is understandable, as the retrieved motions from the dataset are captured motions with fewer artifacts, such as foot skating. On the aspects of trajectory and keypose accuracies, our method outperforms Motion Retrieval more than 74% of times. Overall, the user perceptual study validates our superior performance and is consistent with observations in the comparison.

7.4 Application and Discussion

Real-world storyboard and 3D editing. As stated in Sec. 3, our interface supports simple stick-figure style line segments to ease user burden. However, our algorithm is not limited to only handling simple sketches. As shown in Fig. 11 (a), given a real-world sketch storyboard, our pre-processing step can faithfully extract keypose points and trajectory points (Fig. 11 (b)). These detected 2D points are conditions consumed by the motion generator to produce the vivid motion (Fig. 11 (c)).

As a by-product, our conditional motion generator inherently supports 3D motion editing. As illustrated in Fig. 11 (c), after generating the 3D motion from the storyboard frame, the corresponding 3D keyposes and trajectories can be effortlessly extracted. If users are unsatisfied with the results, our interface allows point-based keypose and trajectory editing by directly dragging and repositioning the points (see the edited trajectory points in purple), which is very similar to Agrawal et al. [2024]. Thanks to the technical design

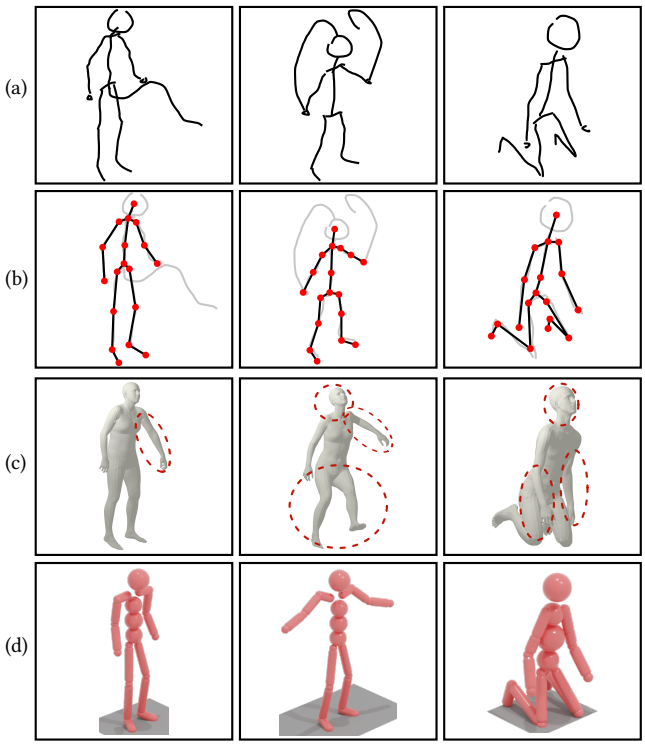


Fig. 12. **Joint detection.** Given user rough sketches (a) with crooked and wobbly strokes, and disproportionally placed body parts, we pre-process them using Sketch2Pose to obtain the 2D joints (b). Their 2D joint detection is robust, but their lifted 3D keyposes (c) are problematic. See the highlighted incorrect parts, *e.g.*, heads, arms, and legs. Taking as input their 2D joints as a condition, we can produce high-quality motions, whose corresponding 3D keyposes (d) faithfully conform with input sketches.

of the motion generator, without any modification, our motion generator can take as input the user-edited keypose and trajectories as new conditions to update the motion (Fig. 11 (d)).

Robustness of joint detection. In Sec. 3, the pre-processing with Sketch2Pose is introduced. Our method is highly dependent on the success of joint detection, especially since the user's raw sketches are often irregular, rough, and have disproportionate body parts. We thus particularly validate the robustness of joint detection from Sketch2Pose in our storyboard sketch scenario. A few examples are presented in Fig. 12. Even though the strokes are crooked and wobbly (Fig. 12(a)), Sketch2Pose can successfully and robustly detect the joints (Fig. 12(b)), but their 3D keypose estimation (Fig. 12(c)) is not reliable, which is the main reason we only take as input their 2D keypose instead of the lifted 3D keypose in our algorithm. As a comparison, we extract 3D keyposes from our generated motions corresponding to input sketches and display them in Fig. 12(d), where our 3D keyposes conform to input sketches faithfully.

Impact of conditions. We have ablated our generator design and the training and inference loss terms in Sec. 7.2 and the supplementary, respectively. Here, we further validate the impact of our conditions. To this end, we start with an action word to generate a motion and gradually add or change one condition each time when

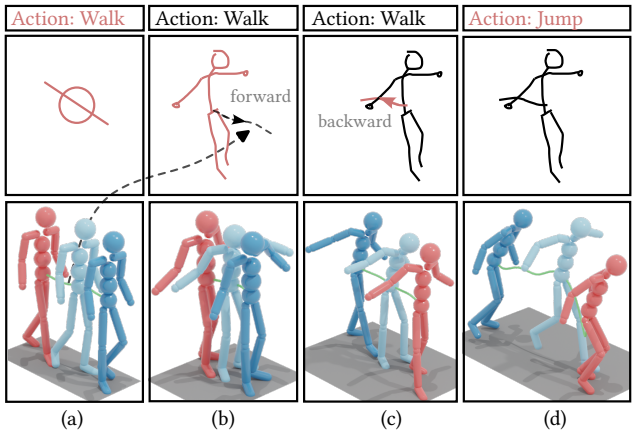


Fig. 13. **Impact of conditions.** From left to right, we gradually add or change one condition for our method when generating motions at inference time. We highlight the change with the pink color. (a) Only the action word "Walk" is input as the condition. (b) The action word and the generated forward root joint trajectory are re-used. A new sketch keypose is provided. (c) The forward trajectory is replaced by a backward trajectory. (d) The action word is replaced by "Jump".

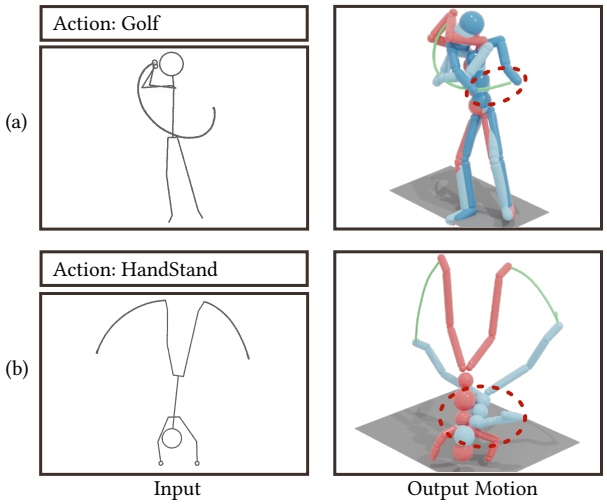


Fig. 14. **Limitations.** (a) Our method does not consider character-object interaction, thus the two hands do not hold the golf club at the ending keypose. (b) Without the physical constraints, even if the foot trajectories are correct, the body folds and floats in the air at the end of the motion. The errors are highlighted with dashed red circles.

generating new motions at inference time. The results can be seen in Fig. 13, where in subfigure (a), given only the action word "Walk" as the condition, our method produces a satisfactory walk motion with two hands down. We then re-use the action word as well as the generated root joint trajectory in (a) and take as input a new sketch keypose (Fig. 13(b)) to produce an updated motion with the two hands up conforming with the sketch. Note that the root joint trajectory in Fig. 13(a) is forward, see the dashed lines. We further change the root joint trajectory to a new backward sketch (Fig. 13(c)), and the generated motion is successfully adjusted. Lastly, we keep the sketch keypose and the backward trajectory, but change the action word from "Walk" to "Jump" (Fig. 13(d)). The resulting motion is a mild jumping, constrained by the relatively flat sketch trajectory. Because of the conflicts between the trajectory and the action word, deviations of keypose and trajectory from the sketches are reasonably observed, while the motion is still satisfying.

Limitations. Our method has a few limitations. Firstly, we do not consider character-object interaction in our algorithm design. Thus, in Fig. 14(a), even though the storyboard frame demonstrates the animation of playing golf, the two hands of the ending keypose are separated, without holding the golf club. Secondly, our algorithm does not incorporate physical constraints, leading to physically incorrect keyposes. For instance, in Fig. 14(b), the body folds and floats in the air at the end of the "HandStand" motion. Extra physical constraints (*e.g.*, a loss term) extracted from the action word might solve this problem. Thirdly, although we have validated the robustness of joint detection, we could not explore all the variants of user sketches. Clearly, Sketch2Pose [Brodt and Bessmeltsev 2022] might fail if the drawing deviates too much from a reasonable human character. This can be addressed by re-drawing some strokes or manually positioning misplaced joint points in the interface.

8 Conclusion and Future Work

We have presented *Sketch2Anim*, the *first* approach for transferring a sketch storyboard into its high-quality 3D animation. We solve this problem from the perspective of conditional motion generation, and the key idea is to exploit the informative 3D keypose and trajectory during training of the motion diffusion model to enable precise control, while directly inputting the 2D keypose and trajectory during inference. This is achieved by our dedicated neural mapper to align the 2D keypose and trajectory to their 3D counterparts in the shared embedding spaces. We have extensively evaluated the superior performance of our approach with a comparison, an ablation study, and a user perceptual study. We further demonstrated the robustness and flexibility of our method on real-world sketch storyboards and keypose- or trajectory-based motion editing applications.

Future work. There are a few inspiring directions worth exploring in the future.

(1) Speed control: besides keypose and trajectory strokes, over-sketched speed lines are also widely used in storyboard sketches [Choi et al. 2012]. For example, a dense speed line pattern conveys a fast motion, providing hints for the animation timeline control. We plan to include a speed line detection and understanding module and inject the extra constraints into the motion generation model. See more discussion about speed control in the supplementary.
(2) Scene recovery: industrial storyboards usually contain scene strokes representing the surrounding objects and environment (see Fig. 11(a)). It is beneficial if the scene reconstruction and character animation can be considered together with mutual spatial constraints to each other. For instance, if the table in the room is generated together with the motion, it can provide a minimum height constraint for the foot when jumping to prevent penetration and collision between the foot and the table.

Acknowledgments

The authors would like to thank the reviewers for their valuable suggestions, Shuyuan Zhang for crafting the animation in Fig. 2 and the video using the traditional 3D animation workflow in Blender, and Adrien Bousseau, Hakan Bilen for proofreading earlier drafts of the paper. CL was supported by a gift from Adobe. YX was supported by the Apple Scholars in AI/ML PhD fellowship.

References

Dhruv Agrawal, Jakob Buhmann, Dominik Borer, Robert W Sumner, and Martin Guay. 2024. SKEL-Betweener: a Neural Motion Rig for Interactive Motion Authoring. *ACM Transactions on Graphics (TOG)* (2024).

Nikos Athanasiou, Alpár Cseke, Markos Diomataris, Michael J Black, and Gül Varol. 2024. MotionFix: Text-driven 3d human motion editing. In *SIGGRAPH Asia 2024 Conference Papers*.

German Barquero, Sergio Escalera, and Cristina Palmero. 2024. Seamless human motion composition with blended positional encodings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Darwin Bautista and Rowel Atienza. 2022. Scene text recognition with permuted autoregressive sequence models. In *European conference on computer vision*. Springer.

Mikhail Bessmeltsev, Nicholas Vining, and Alla Sheffer. 2016. Gesture3D: posing 3D characters via gesture drawings. *ACM Transactions on Graphics (TOG)* (2016).

Yuxuan Bian, Ailing Zeng, Xuan Ju, Xian Liu, Zhaoyang Zhang, Wei Liu, and Qiang Xu. 2024. MotionCraft: Crafting Whole-Body Motion with Plug-and-Play Multimodal Controls. *arXiv preprint arXiv:2407.21136* (2024).

Kirill Brodt and Mikhail Bessmeltsev. 2022. Sketch2Pose: estimating a 3D character pose from a bitmap sketch. *ACM Transactions on Graphics (TOG)* (2022).

Michael Büttner and Simon Clavet. 2015. Motion Matching - The Road to Next Gen Animation. https://www.youtube.com/watch?v=z_wpgHFSWss&t=658s. Presented at Nucl.ai 2015.

Xin Chen, Biao Jiang, Wen Liu, Zilong Huang, Bin Fu, Tao Chen, and Gang Yu. 2023. Executing your commands via motion diffusion in latent space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Byungkuk Choi, Roger Blanco i Ribera, John P Lewis, Yeongho Seol, Seokpyo Hong, Haegwang Eom, Sunjin Jung, and Junyong Noh. 2016. SketchiMo: sketch-based motion editing for articulated characters. *ACM Transactions on Graphics (ToG)* (2016).

Myung Geol Choi, Kyungyong Yang, Takeo Igarashi, Jun Mitani, and Jehee Lee. 2012. Retrieval and visualization of human motion data via stick figures. In *Computer Graphics Forum*. Wiley Online Library.

Setareh Cohan, Guy Tevet, Daniele Reda, Xue Bin Peng, and Michiel van de Panne. 2024. Flexible motion in-betweening with diffusion models. In *ACM SIGGRAPH 2024 Conference Papers*.

Wenxun Dai, Ling-Hao Chen, Jingbo Wang, Jinpeng Liu, Bo Dai, and Yansong Tang. 2025. Motionlcm: Real-time controllable motion generation via latent consistency model. In *European Conference on Computer Vision*. Springer, 390–408.

James Davis, Maneesh Agrawala, Erika Chuang, Zoran Popović, and David Salesin. 2006. A sketching interface for articulated figure animation. In *Acm siggraph 2006 courses*. 15–es.

Prafulla Dhariwal and Alexander Nichol. 2021. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems* (2021).

Christian Diller and Angela Dai. 2024. Cg-hoi: Contact-guided 3d human-object interaction generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Marek Dvorožňák, Daniel Sỳkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. 2020. Monster mash: a single-view approach to casual 3D modeling and animation. *ACM Transactions on Graphics (ToG)* (2020).

Rinon Gal, Yael Vinker, Yuval Alaluf, Amit Bermano, Daniel Cohen-Or, Ariel Shamir, and Gal Chechik. 2024. Breathing Life Into Sketches Using Text-to-Video Priors. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. AllenNLP: A Deep Semantic Natural Language Processing Platform. arXiv:arXiv:1803.07640

Anindita Ghosh, Noshaba Cheema, Cennet Oguz, Christian Theobalt, and Philipp Slusallek. 2021. Synthesis of compositional animations from textual descriptions. In *Proceedings of the IEEE/CVF international conference on computer vision*.

Purvi Goel, Kuan-Chieh Wang, C Karen Liu, and Kayvon Fatahalian. 2024. Iterative motion editing with natural language. In *ACM SIGGRAPH 2024 Conference Papers*.

Martin Guay, Marie-Paule Cani, and Rémi Ronfard. 2013. The line of action: an intuitive interface for expressive character posing. *ACM Transactions on Graphics (TOG)* (2013).

Martin Guay, Rémi Ronfard, Michael Gleicher, and Marie-Paule Cani. 2015. Space-time sketching of character animation. *ACM Transactions on Graphics (ToG)* (2015).

Chuan Guo, Yuxuan Mu, Muhammad Gohar Javed, Sen Wang, and Li Cheng. 2024. Momask: Generative masked modeling of 3d human motions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

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

Chuan Guo, Xinxin Zuo, Sen Wang, and Li Cheng. 2022b. Tm2t: Stochastic and tokenized modeling for the reciprocal generation of 3d human motions and texts. In *European Conference on Computer Vision*.

Fabian Hahn, Frederik Mutzel, Stelian Coros, Bernhard Thomaszewski, Maurizio Nitti, Markus Gross, and Robert W Sumner. 2015. Sketch abstractions for character posing. In *Proceedings of the 14th ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. 185–191.

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

Rachel Heck and Michael Gleicher. 2007. Parametric motion graphs. In *Proceedings of the 2007 symposium on Interactive 3D graphics and games*.

Jonathan Ho and Tim Salimans. 2022. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598* (2022).

Takeo Igarashi, Rieko Kadobayashi, Kenji Mase, and Hidehiko Tanaka. 1998. Path drawing for 3D walkthrough. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*.

Biao Jiang, Xin Chen, Wen Liu, Jingyi Yu, Gang Yu, and Tao Chen. 2023. Motiongpt: Human motion as a foreign language. *Advances in Neural Information Processing Systems* (2023).

Roy Kapon, Guy Tevet, Daniel Cohen-Or, and Amit H Bermano. 2024. MAS: Multi-view Ancestral Sampling for 3D motion generation using 2D diffusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Korrawe Karunratanakul, Konpat Preechakul, Supasorn Suwajanakorn, and Siyu Tang. 2023. Guided Motion Diffusion for Controllable Human Motion Synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Lucas Kovar, Michael Gleicher, and Frédéric Pighin. 2023. Motion graphs. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*.

John Lasseter. 1998. Principles of traditional animation applied to 3D computer animation. In *Seminal graphics: pioneering efforts that shaped the field*.

Jiaman Li, Alexander Clegg, Roozbeh Mottaghi, Jiajun Wu, Xavier Puig, and C Karen Liu. 2025. Controllable human-object interaction synthesis. In *European Conference on Computer Vision*.

Jiaman Li, C Karen Liu, and Jiajun Wu. 2024a. Lifting Motion to the 3D World via 2D Diffusion. *arXiv preprint arXiv:2411.18808* (2024).

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

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)* (2022).

Zhe Li, Weihao Yuan, Yisheng He, Lingteng Qiu, Shenhao Zhu, Xiaodong Gu, Weichao Shen, Yuan Dong, Zilong Dong, and Laurence T. Yang. 2024b. LaMP: Language-Motion Pretraining for Motion Generation, Retrieval, and Captioning. In *arXiv 2410.07093*.

Juncong Lin, Takeo Igarashi, Jun Mitani, and Greg Saul. 2010. A sketching interface for sitting-pose design. In *Proceedings of the Seventh Sketch-Based Interfaces and Modeling Symposium*.

Dong C Liu and Jorge Nocedal. 1989. On the limited memory BFGS method for large scale optimization. *Mathematical programming* (1989).

Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J Black. 2023. SMPL: A skinned multi-person linear model. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *ArXiv* (2017).

Chong Mou, Xintao Wang, Liangbin Xie, Yanze Wu, Jian Zhang, Zhongang Qi, and Ying Shan. 2024. T2i-adapter: Learning adapters to dig out more controllable ability for text-to-image diffusion models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Nkeeline. 2024. Keemap-Blender Rig Retargeting Addon. https://github.com/nkeeline/Keemap-Blender-Rig-ReTargeting-Addon.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

Yichen Peng, Zhengyu Huang, Chunqi Zhao, Haoran Xie, Tsukasa Fukusato, and Kazunori Miyata. 2021. Sketch-based human motion retrieval via shadow guidance. In *2021 Nicograph International (NicoInt)*.

Mathis Petrovich, Michael J Black, and Gül Varol. 2022. TEMOS: Generating diverse human motions from textual descriptions. In *European Conference on Computer Vision*. Springer.

Mathis Petrovich, Michael J Black, and Gül Varol. 2023. TMR: Text-to-motion retrieval using contrastive 3D human motion synthesis. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Ekkasit Pinyoanuntapong, Muhammad Usama Saleem, Korrawe Karunratanakul, Pu Wang, Hongfei Xue, Chen Chen, Chuan Guo, Junli Cao, Jian Ren, and Sergey Tulyakov. 2024a. ControlMM: Controllable Masked Motion Generation. *arXiv preprint arXiv:2410.10780* (2024).

Ekkasit Pinyoanuntapong, Pu Wang, Minwoo Lee, and Chen Chen. 2024b. Mmm: Generative masked motion model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 1546–1555.

Can Qin, Shu Zhang, Ning Yu, Yihao Feng, Xinyi Yang, Yingbo Zhou, Huan Wang, Juan Carlos Niebles, Caiming Xiong, Silvio Savarese, et al. 2023. Unicontrol: A unified diffusion model for controllable visual generation in the wild. *arXiv preprint arXiv:2305.11147* (2023).

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*. PMLR, 8748–8763.

Gaurav Rai and Ojaswa Sharma. 2024. Enhancing Sketch Animation: Text-to-Video Diffusion Models with Temporal Consistency and Rigidity Constraints. *arXiv preprint arXiv:2411.19381* (2024).

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

Hyun Joon Shin and Hyun Seok Oh. 2006. Fat graphs: constructing an interactive character with continuous controls. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*.

Harrison Jesse Smith, Qingyuan Zheng, Yifei Li, Somya Jain, and Jessica K Hodgins. 2023. A method for animating children's drawings of the human figure. *ACM Transactions on Graphics* (2023).

Jiaming Song, Chenlin Meng, and Stefano Ermon. 2020. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502* (2020).

Jiaming Song, Qinsheng Zhang, Hongxu Yin, Morteza Mardani, Ming-Yu Liu, Jan Kautz, Yongxin Chen, and Arash Vahdat. 2023. Loss-guided diffusion models for plug-and-play controllable generation. In *International Conference on Machine Learning*. PMLR.

Paul Starke, Sebastian Starke, Taku Komura, and Frank Steinicke. 2023. Motion in-betweening with phase manifolds. *Proceedings of the ACM on Computer Graphics and Interactive Techniques* (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

Matthew Thorne, David Burke, and Michiel Van De Panne. 2004. Motion doodles: an interface for sketching character motion. *ACM Transactions on Graphics (ToG)* (2004).

Weilin Wan, Zhiyang Dou, Taku Komura, Wenping Wang, Dinesh Jayaraman, and Lingjie Liu. 2023. Tlcontrol: Trajectory and language control for human motion synthesis. *arXiv preprint arXiv:2311.17135* (2023).

Zhenzhi Wang, Jingbo Wang, Dahua Lin, and Bo Dai. 2023. InterControl: Generate Human Motion Interactions by Controlling Every Joint. *arXiv preprint arXiv:2311.15864* (2023).

Chung-Yi Weng, Brian Curless, and Ira Kemelmacher-Shlizerman. 2019. Photo wake-up: 3d character animation from a single photo. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*.

Richard Williams. 2012. *The animator's survival kit: a manual of methods, principles and formulas for classical, computer, games, stop motion and internet animators*. Macmillan.

Zizhao Wu, Qin Wang, Xinyang Zheng, Jianglei Ye, Ping Yang, Yunhai Wang, and Yigang Wang. 2024. Doodle Your Motion: Sketch-Guided Human Motion Generation. *IEEE Transactions on Visualization and Computer Graphics* (2024).

Yiming Xie, Varun Jampani, Lei Zhong, Deqing Sun, and Huaizu Jiang. 2024. OmniControl: Control Any Joint at Any Time for Human Motion Generation. In *The Twelfth International Conference on Learning Representations*.

Ji Yang, Xinxin Zuo, Sen Wang, Zhenbo Yu, Xingyu Li, Bingbing Ni, Minglun Gong, and Li Cheng. 2022. Object Wake-up: 3D Object Rigging from a Single Image. In *European Conference on Computer Vision*.

Hongwei Yi, Justus Thies, Michael J. Black, Xue Bin Peng, and Davis Rempe. 2024. Generating Human Interaction Motions in Scenes with Text Control. *arXiv:2404.10685* (2024).

Innfarn Yoo, Juraj Vanek, Maria Nizovtseva, Nicoletta Adamo-Villani, and Bedrich Benes. 2014. Sketching human character animations by composing sequences from large motion database. *The Visual Computer* (2014).

Jianrong Zhang, Yangsong Zhang, Xiaodong Cun, Yong Zhang, Hongwei Zhao, Hongtao Lu, Xi Shen, and Ying Shan. 2023b. Generating human motion from textual descriptions with discrete representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 14730–14740.

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023a. Adding Conditional Control to Text-to-Image Diffusion Models. In *IEEE International Conference on Computer Vision (ICCV)*.

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).

Yaqi Zhang, Di Huang, Bin Liu, Shixiang Tang, Yan Lu, Lu Chen, Lei Bai, Qi Chu, Nenghai Yu, and Wanli Ouyang. 2024. Motiongpt: Finetuned llms are general-purpose motion generators. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 7368–7376.

Shihao Zhao, Dongdong Chen, Yen-Chun Chen, Jianmin Bao, Shaozhe Hao, Lu Yuan, and Kwan-Yee K Wong. 2024. Uni-controlnet: All-in-one control to text-to-image diffusion models. *Advances in Neural Information Processing Systems* (2024).

Lei Zhong, Yiming Xie, Varun Jampani, Deqing Sun, and Huaizu Jiang. 2025. Smoodi: Stylized motion diffusion model. In *European Conference on Computer Vision*.

Jie Zhou, Chufeng Xiao, Miu-Ling Lam, and Hongbo Fu. 2024. DrawingSpinUp: 3D Animation from Single Character Drawings. In *SIGGRAPH Asia 2024 Conference Papers*.

Wenyang Zhou, Zhiyang Dou, Zeyu Cao, Zhouyingcheng Liao, Jingbo Wang, Wenjia Wang, Yuan Liu, Taku Komura, Wenping Wang, and Lingjie Liu. 2025. Emdm: Efficient motion diffusion model for fast and high-quality motion generation. In *European Conference on Computer Vision*. Springer, 18–38.

Wentao Zhu, Xiaoxuan Ma, Zhaoyang Liu, Libin Liu, Wayne Wu, and Yizhou Wang. 2023. Motionbert: A unified perspective on learning human motion representations. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*.

Supplemental Material

In this supplementary material, we provide additional details about dataset processing, user evaluation, the implementation of our approach and all competitors. Furthermore, we include more ablation study results to analyze the impact of each loss term. We provide a *supplemental video*, which we encourage the reviewers to watch since motion is critical in our results, and this is hard to convey in a static document.

A More ablation studies

In this section, we evaluate the impact of each loss term, when training both the multi-conditional motion diffusion model and the neural mapper to align 2D-3D trajectory and keypose embeddings, respectively. *Because inference guidance can be seen as a post-optimization process, neither variant employs inference guidance in these ablation studies.*

Firstly, we exclude the spatial keypose and trajectories constraint term when training our multi-conditional motion diffusion model, denoted as w/o $\mathcal{L}_{\text{key}}\&\mathcal{L}_{\text{tr}}$. As shown in Table A1, although introducing spatial constraints affects the FID metric, it significantly improves control accuracy, with MPJPE-3D improving by 18.37% and Avg. Err-3D improving by **46.58%** in the Average setting, and by 16.07% and **42.59%**, respectively, in the Cross setting. The trade-off between control constraints and FID metrics is also reported in [Dai et al. 2025; Zhong et al. 2025]. For the text-motion matching metric, it degrades a little bit.

Then, we exclude each loss term in the alignment process. First, we exclude the *reconstruction* term in the loss function, denoted as w/o $\mathcal{L}_{\text{recon}}$. Comparing the results in the 1^{st} and 4^{rd} rows in both the Average and Cross settings in Tab. A2, we observe that removing $\mathcal{L}_{\text{recon}}$ leads to a considerable degradation in the FID metric, with a **32.81%** and **23.12%** drop for the Average and Cross settings, respectively. The other two metrics degrade as well. When We exclude the *match* term (denoted as w/o $\mathcal{L}_{\text{match}}$), we observe similar performance change, e.g., the large FID degradation, and the mild drop of the text-motion matching metric. Both the *recosntruction* and *match* terms play essential roles in the alignment training. The *contrast* loss complements the above two terms and further improves the performance to some extent.

B Dataset Processing

We train and evaluate our system on the HumanML3D dataset [Guo et al. 2022a], containing 14,646 motions with 44,970 corresponding motion annotations. Following the processing approach outlined in [Guo et al. 2022a], we preprocess the HumanML3D dateset to obtain the redundant motion representations. To effectively select a representative keypose from a given motion sequence, we first use NLP tools [Gardner et al. 2017] to extract the *first* verb as the action word from the text description and form a simplified text description with the template of *'a person [action_word]'*. Next, we slide a window over the motion sequence to compute similarity scores between each motion segment and the simplified sentence using TMR [Petrovich et al. 2023]. The motion segment with the highest similarity score is identified as the candidate group of keyposes, from which individual keyposes are randomly selected during training. During the evaluation, we exploit the first keypose from the candidate group as the keypose.

To obtain 2D keyposes and joint trajectories resembling user sketches extracted from storyboards, we begin by orthographically projecting the 3D motion to generate the corresponding 2D motion. From this projected data, we extract the 2D keyposes and joint trajectories. Specifically, we focus on six commonly used end-effector joints - *pelvis, left foot, right foot, head, left wrist*, and *right wrist*, to extract trajectories. In order to mimic the irregularity of real user sketches at testing time, we employ the following three data augmentation strategies:

- *Camera view augmentation*: The camera parameter p comprises the scale s and the Euler angles $v = (v^{\text{pitch}}, v^{\text{yaw}}, v^{\text{roll}})$. During training, v^{roll} is fixed at 0, while v^{pitch} is randomly sampled from $[0°, 30°]$, v^{yaw} from $[-45°, 45°]$, and s from $[0.8, 1.2]$. This procedure is crucial for training the network to consistently map similar poses, despite variations in 2D scale and viewpoint, to the same (local) 3D representation.
- *Random joint perturbation*: Since user sketches often contain rough strokes (*e.g.*, crooked lines), we add Gaussian noise with a standard deviation of 0.02 to the 2D joints to better match real-world drawing scenarios.
- *Body proportion perturbation*: other than the irregularity of a single stroke, the user sketches might have incorrect body proportions, we thus randomly scale selective body parts: leg, spine, and arm, in the 2D projected keypose by a factor s, sampled from the range $[0.6, 1.6]$.

C The details of Inference Guidance

The core of inference guidance is an analytic function. We optimize this analytic function using the second-order optimizer L-BFGS [Liu and Nocedal 1989], which better aligns with the desired trajectory and achieves quicker convergence compared to first-order based methods [Pinyoanuntapong et al. 2024a; Xie et al. 2024]:

$$\epsilon_\theta(\boldsymbol{z}_t, t, \mathbf{a}, \mathbf{T}^r_{2D}, \mathbf{K}_{2D}) = \epsilon_\theta(\boldsymbol{z}_t, t, \mathbf{a}, \mathbf{T}^r_{2D}, \mathbf{K}_{2D}) - \tau_2 \cdot \mathbf{H}^{-1} \nabla_{\boldsymbol{z}_t} G(\boldsymbol{z}_t, t, \boldsymbol{v}, \mathbf{T}^r_{2D}), \tag{15}$$

$$G(\boldsymbol{z}_t, t, \boldsymbol{v}, \mathbf{T}^r_{2D}) = \frac{\sum_{i,j} m_{ij} \| P(R(\hat{\mathbf{x}}_0)_{ij}, \boldsymbol{v}) - \mathbf{T}^r_{2D} \|_2^2}{\sum_{i,j} m_{ij}}, \tag{16}$$

where τ_2 controls the strength of the guidance, and $P(\boldsymbol{x}, \boldsymbol{v})$ represents the projection of the motion $\boldsymbol{x}$ under the camera view $\boldsymbol{v}$. The term $\mathbf{H}^{-1}$ denotes the approximate inverse Hessian matrix used in the L-BFGS optimizer [Liu and Nocedal 1989]. Because the generated motion aligns closely with the desired 2D keyposes, we do not adopt similar inference guidance for the keypose condition.

Algorithm pseudo-code. We list the pseudo-code of our algorithm during inference in Algo. 1. Note that the subscript $2D$ and the guidance term show the full algorithm inference (Sec. 6 in the paper), while the subscript $3D$ indicates the training/inference process of the 3D conditioned motion generator (Sec. 4 in the paper).

Evaluation. In this work, we employ inference guidance as a post-optimization step to ensure the generated motion better follows the given 2D joint trajectory. As illustrated in the 1^{st} and 3^{rd} rows in Tab. A3, using the second-order gradient significantly improves the

Table A1. Ablation study of the loss terms of our multi-conditional motion diffusion model using the HumanML3D dataset. Refer to the main paper for the definition of metrics and the Average and Cross evaluation settings.

Joint	Method	Realism		Control Accuracy		Text-Motion Matching	
		FID ↓	Foot skating ratio ↓	MPJPE-3D ↓	Avg. Err.-3D ↓	MM Dist ↓	R-precision (Top-3) ↑
Average	w/o $\mathcal{L}_{\text{key}}$&$\mathcal{L}_{\text{tr}}$	**0.291**	**0.0858**	0.0490	0.307	**2.908**	**0.823**
	Ours	0.451	0.094	**0.040**	**0.164**	2.934	0.810
Cross	w/o $\mathcal{L}_{\text{key}}$&$\mathcal{L}_{\text{tr}}$	**0.271**	**0.0895**	0.0473	0.270	**2.858**	**0.824**
	Ours	0.370	0.0959	**0.0397**	**0.155**	2.897	0.815

Table A2. Ablation study of the loss terms of our neural mapper to align 2D-3D embeddings using the HumanML3D dataset. Refer to the paper for the definition of the reported metrics and the Average and Cross evaluation settings. Note that the statistics of Ours' are different with the numbers in Table 1 in the main paper, because we did not include inference guidance in this experiment.

Joint	Method	Realism		Control Accuracy				Text-Motion Matching	
		FID ↓	Foot Skating ↓	MPJPE-2D ↓	MPJPE-3D ↓	Avg. Err.-2D ↓	Avg. Err.-3D ↓	MM Dist ↓	R-precision (Top-3) ↑
Average	w/o $\mathcal{L}_{\text{recon}}$	0.768	0.120	0.0391	0.0539	0.184	0.266	3.252	0.754
	w/o $\mathcal{L}_{\text{contrast}}$	0.544	**0.0994**	0.0370	0.0509	0.162	0.232	3.208	0.743
	w/o $\mathcal{L}_{\text{match}}$	0.722	0.117	0.0369	0.0509	0.172	0.247	3.232	0.757
	Ours	**0.516**	0.108	**0.0361**	**0.0498**	**0.158**	**0.227**	**3.128**	**0.769**
Cross	w/o $\mathcal{L}_{\text{recon}}$	0.653	0.116	0.0384	0.0528	0.181	0.254	3.141	0.781
	w/o $\mathcal{L}_{\text{contrast}}$	0.534	**0.0998**	0.0370	0.0508	0.160	0.235	2.984	0.794
	w/o $\mathcal{L}_{\text{match}}$	0.663	0.115	0.0365	0.0502	0.173	0.243	3.138	0.780
	Ours	**0.502**	0.107	**0.0357**	**0.0493**	**0.159**	**0.225**	**2.888**	**0.815**

Algorithm 1: *Sketch2Anim*'s inference

Require: A motion diffusion model M with parameters θ_M, a Trajectory ControlNet $\mathcal{F}_{tr}$ with parameters θ_{tr}, and a KeyPose Adapter $\mathcal{F}_k$ with parameters θ_k. The inputs include 2D/3D keyposes $\mathbf{K}_{2D/3D}$, 2D/3D joint trajectories $\mathbf{T}^r_{2D/3D}$, camera view $\boldsymbol{v}$, and action word embeddings $\mathbf{a}$, which are obtained from the action word $\mathbf{W}_a$ via CLIP.
$\boldsymbol{z}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$; // Sample from pure Gaussian distribution
for $t = T$ *to* 1 **do**
  $\{\boldsymbol{r}\} \leftarrow \mathcal{F}_{tr}(\boldsymbol{z}_t, t, \boldsymbol{a}, \mathbf{T}^r_{2D/3D}; \theta_{tr})$; // Trajectory ControlNet
  $\{\boldsymbol{r}'\} \leftarrow \mathcal{F}_k(\boldsymbol{z}_t, t, \{\boldsymbol{r}\}, \boldsymbol{a}, \mathbf{K}_{2D/3D}; \theta_k)$; // Keypose Adapter
  $\epsilon_t \leftarrow M(\boldsymbol{z}_t, t, \boldsymbol{a}, \{\boldsymbol{r}'\}; \theta_M)$; // Motion diffusion model
  // Inference guidance
  if *input condition is 2D* **then**
    for $k = 1$ *to* K **do**
      $\epsilon_t = \epsilon_t - \tau \cdot \mathbf{H}^{-1} \nabla_{\boldsymbol{z}_t} G(\boldsymbol{z}_t, t, \mathbf{T}^r_{2D}, \boldsymbol{v})$; // 2D guidance
    end
  end
  $\boldsymbol{z}_{t-1} \sim \mathcal{S}(\boldsymbol{z}_t, \epsilon_t, t)$; // $\mathcal{S}(\cdot,\cdot,\cdot)$: DDIM sampling [2021]
end
$\boldsymbol{x}_0 = \mathbf{D}(\boldsymbol{z}_0)$
return $\boldsymbol{x}_0$

trajectory control accuracy, with Avg. Err.-2D decreasing by 45.13% in the Average setting and 50.19% in the Cross setting, and Avg. Err.-3D decreasing by 40.97% and 41.33%, respectively. Moreover, previous methods [Karunratanakul et al. 2023; Pinyoanuntapong et al. 2024a; Xie et al. 2024] primarily rely on first-order gradient-based approaches to ensure that the generated motion adheres to the given trajectories. We thus report the metrics by using first-order gradients when applying inference guidance. The results in Tab. A3 demonstrate that second-order methods achieve better performance than first-order methods across all metrics.

D The details of Motion Blending

In this section, we describe the full details of our inversion-based motion blending and evaluate its performance by a statistical comparison.

Implementation. Given two adjacent motion clips, $\hat{\mathbf{x}}^p_0$ and $\hat{\mathbf{x}}^q_0$, we select a segment of length l at the junction of their start and end as the motion transition. We initialize a transition motion $\mathbf{x}^l_0$ using linear blending and compose it with adjacent clips to form $\mathbf{x}^{p+q}_0$. Subsequently, we apply the deterministic DDIM inversion process [Song et al. 2020] to obtain the noised latent code $\boldsymbol{z}^{\text{Inv}}_T$ for the concatenated motion. The reverse process can be represented at step t as:

$$\boldsymbol{z}_{t+1} = \sqrt{\frac{\omega_{t+1}}{\omega_t}} \left(\boldsymbol{z}_t + \left(\sqrt{\frac{1}{\omega_{t+1}}} - 1 \right) - \left(\sqrt{\frac{1}{\omega_t}} - 1 \right) \right) \cdot \varepsilon_\theta(\boldsymbol{z}_t; t, \boldsymbol{c}, \emptyset), \tag{17}$$

where ω represents the noise scale. $\boldsymbol{z}^{Inv}_T$ can be obtained at the last reverse step T. During inference, we denoise $\mathbf{z}^{Inv}_T$ using a combination of classifier-free guidance and second-order inference guidance as follows:

$$\epsilon_\theta(\boldsymbol{z}_t, t, \mathbf{a}_{p+q}) = \epsilon_\theta(\boldsymbol{z}_t, t, \mathbf{a}_{p+q}) - \tau_3 \cdot \mathbf{H}^{-1} \nabla_{\boldsymbol{z}_t} G_m(\boldsymbol{z}_t, t, \hat{\mathbf{x}}^p_0, \hat{\mathbf{x}}^q_0). \tag{18}$$

Here, $\mathbf{a}_{p+q}$ represents the action words that combines the p part and q part. τ_3 adjusts the strength of inference guidance. $\mathbf{H}^{-1}$ is

Table A3. Ablation study of the inference guidance using the HumanML3D dataset.

Joint	Method	Realism		Control Accuracy				Text-Motion Matching	
		FID ↓	Foot Skating ↓	MPJPE-2D ↓	MPJPE-3D ↓	Avg. Err.-2D ↓	Avg. Err.-3D ↓	MM Dist ↓	R-precision (Top-3) ↑
Average	w/o inf. guidance	**0.516**	0.108	0.0361	0.0498	0.158	0.227	3.128	0.769
	w/ 1^{st} order grad.	0.520	0.105	0.0361	0.0490	0.102	0.183	3.093	0.797
	w/ 2^{nd} order grad.	0.525	**0.103**	**0.0360**	**0.0478**	**0.0867**	**0.134**	**3.077**	**0.802**
Cross	w/o inf. guidance	**0.502**	**0.107**	0.0357	0.0493	0.159	0.225	**2.888**	**0.815**
	w/ 1^{st} order grad.	0.596	0.104	0.0350	0.0471	0.0981	0.184	2.906	0.801
	w/ 2^{nd} order grad.	0.577	0.102	**0.0329**	**0.0462**	**0.0792**	**0.132**	3.042	0.796

Table A4. Quantitative comparison of motion blending methods on the HumanML3D subset.

Method	FID ↓	AUJ ↓	PJ (GT = 0.0330)
SQUAD	1.0015	0.5589	0.0002
DoubleTake	**0.3156**	0.2823	0.0182
Ours	0.4204	**0.2075**	**0.0217**

the approximate inverse Hessian matrix used in the L-BFGS optimizer [Liu and Nocedal 1989]. The function G_m is used to measure the motion similarity in their global joint space, defined as:

$$\|R(\text{slice}(\hat{\mathbf{x}}_0^{Inv}, p)) - R(\hat{\mathbf{x}}_0^{p})\|_2^2 + \|R(\text{slice}(\hat{\mathbf{x}}_0^{Inv}, q)) - R(\hat{\mathbf{x}}_0^{q})\|_2^2,$$

where $\text{slice}(\mathbf{x}; i)$ represents the extraction of the segment i from the motion $\mathbf{x}$, and $\hat{\mathbf{x}}_0^{Inv}$ is obtained similarly as in Eq. 8 in the main paper.

We take the final denoising step of $\hat{\mathbf{x}}^{Inv}$ as the result of motion blending between adjacent motions. We iteratively run the above blending method on all the motion clips within a storyboard composing them into a complete animation.

Performance evaluation. To evaluate its effectiveness, we conducted a quantitative comparison against a diffusion-based baseline DoubleTake [Shafir et al. 2023] and SQUAD interpolation. The evaluation was performed on a curated subset of the HumanML3D dataset, comprising 1,219 samples (25% of the test set). This subset was selected by filtering for text descriptions containing two distinct action words, indicating the presence of multiple motion phases. For each sample, we identified two key motion segments and masked out the **middle 20 frames** between the corresponding representative keyposes to define the transition region. Each method was then tasked with generating the blended motion. The resulting motions were compared against ground-truth transitions using the following widely adopted metrics:

- **FID**: Fréchet Inception Distance, for assessing motion realism.
- **AUJ**: Area Under the Jerk Curve, measuring average motion smoothness.
- **PJ**: Peak Jerk, capturing extreme motion fluctuations. The mean PJ value of the ground truth samples in the test set is 0.0330. The metric close to this value is better.

The evaluation results are reported in Tab. A4. Quantitative results indicate that our method surpasses SQUAD in both realism and smoothness and closely matches the performance of DoubleTake across all metrics.

E More Implementation Details

Model Details. Our pre-trained motion diffusion model is based on MLD [Chen et al. 2023]. Both the trajectory ControlNet and the keypose adapter are composed of four Transformer encoder blocks. For the text input, we first combine the action word W_a with the template *'a person [action_word]'* to form a simplified text description. We then leverage a CLIP model [Radford et al. 2021] to encode the text into embeddings $\boldsymbol{a}$ and use linear layers to project the timestep t into time embeddings. These text embeddings are added to the time embeddings and concatenated with the noisy latent $\boldsymbol{z}_t$. The trajectory encoders (*i.e.*, $\mathcal{E}_{tr}^{3D}$ and $\mathcal{E}_{tr}^{2D}$), and and keypose encoders (*i.e.*, $\mathcal{E}_{k}^{3D}$ and $\mathcal{E}_{k}^{2D}$), as illustrated in Figs. 3 and 4 in the main paper, primarily consist of a single Transformer encoder designed to encode the trajectory and keypose into their corresponding embeddings.

Evaluation Details. We follow the evaluation protocol for the *Cross* setting from OmniControl [Xie et al. 2024], assessing multiple combinations of joints. A total of 63 combinations are randomly sampled during the evaluation process.

Inference Time. To evaluate the inference efficiency of our approach and baseline methods, we report the average inference time of generating a single motion clip, measured in seconds, on a single NVIDIA RTX4090 GPU. Specifically, our *Sketch2Anim* takes $0.427s$ to generate a motion clip, while the running time is $0.0415s$, $0.288s$, and $0.427s$ for Motion Retrieval, Lift-and-Control, Direct 2D-to-Motion methods, respectively. Since inference guidance is a post-optimization strategy, it takes around $0.058s$. The Direct 2D-to-Mition method employs the same inference pipeline as ours.

F Design Details of Competitors

Since no existing framework directly addresses the problem of translating sketch storyboards into 3D animation, we developed three baseline approaches based on the latest works. To ensure a fair comparison, all baseline methods use the same input and are trained and evaluated on the same dataset. Additionally, *Lift-and-Control* and *Direct 2D-to-Motion* primarily adopt the same model architecture as *Sketch2Anim*, with only key components replaced. Fig. A1 shows more visual comparison examples.

Motion Retrieval. This retrieval baseline is primarily based on TMR [Petrovich et al. 2023], retaining the same text encoder, motion encoder, and motion decoder. We use the same 2D trajectory and keypose encoder as *Sketch2Anim* to extract trajectory and keypose embeddings, which are then concatenated with the text embedding.

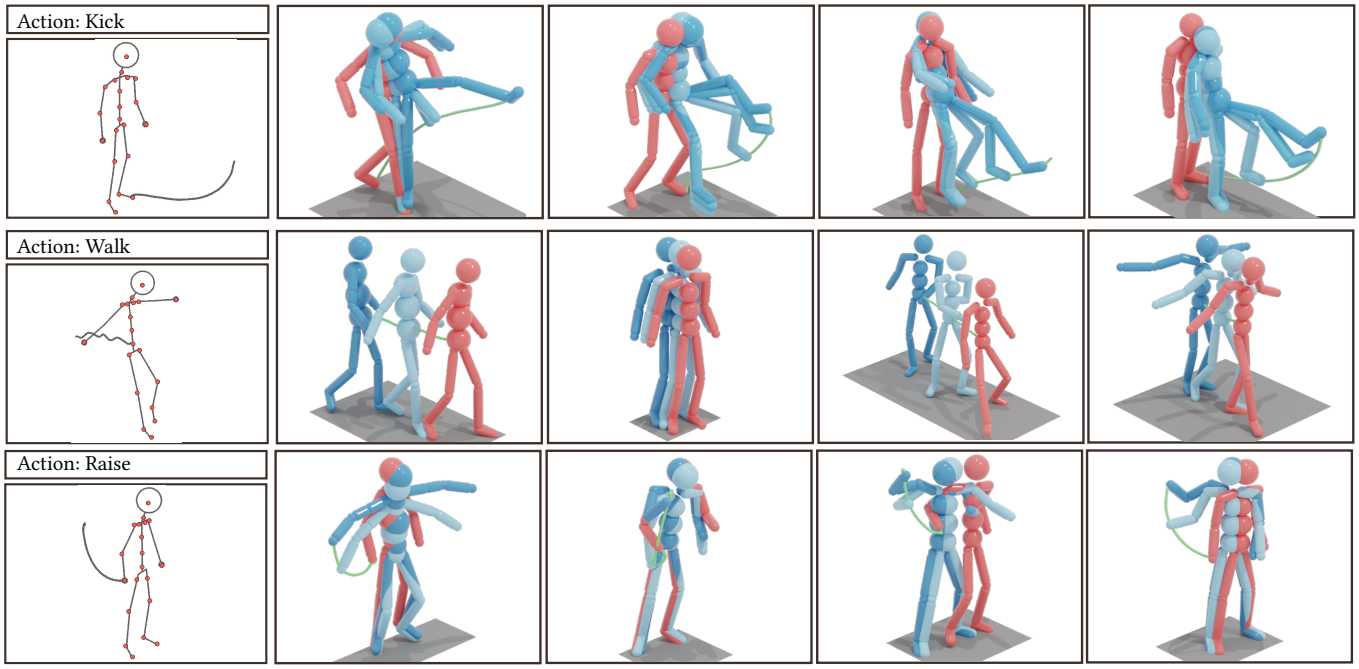


Fig. A1. **More Visual comparison.** Check either the keypose (red) or the trajectory (green) for the differences.

The TEMOS [Petrovich et al. 2022] losses and the InfoNCE loss [Oord et al. 2018] are employed to train this baseline using the AdamW optimizer [Loshchilov and Hutter 2017] with a learning rate of 10^{-4} and a batch size of 32.

Lift-and-Control. This baseline first lifts the 2D keyposes and trajectories to 3D, then employs the same multi-conditional motion diffusion model as *Sketch2Anim* for motion synthesis. The lifting component is based on MotionBERT [Zhu et al. 2023]. We separately project the 2D keyposes and trajectories into high-dimensional features and incorporate learnable spatial and temporal encodings. Subsequently, we apply the DSTformer module, as described in MotionBERT, to lift the keyposes and trajectories to 3D. For training, we adopt the pretraining loss, with the only modification being the separation of the 2D re-projection loss into keypose and trajectory components. The lifting network is trained for 1000 epochs with a learning rate of 5×10^{-4} and a batch size of 64 using the AdamW optimizer [Loshchilov and Hutter 2017].

Direct 2D-to-Motion. This baseline differs from *Sketch2Anim* by directly leveraging 2D keyposes and trajectories to train the multi-conditional motion diffusion model. The training strategy remains unchanged, using the same number of epochs and loss function, but replacing the 3D input with 2D input, as detailed in Section 4.3.

G More Discussion

Lifting 2D Animation Sequence. Given a storyboard frame, an alternative solution is to first generate a 2D animation sequence and then elevate it to its corresponding 3D motion. There are two possible ways to generate a complete 2D animation sequence:

- The first solution involves using sketch image animation methods [Gal et al. 2024; Rai and Sharma 2024] to generate a sketch image animation sequence, estimating the 2D poses, and then lifting them to 3D (*e.g.*, using Sketch2Pose [Brodt and Bessmeltsev 2022]). However, as shown in Fig.A2, our experiments reveal that existing sketch image animation methods [Gal et al. 2024] fail to produce structurally consistent human sequences due to the lack of human body priors. Extending these methods to incorporate human body priors, such as SMPL or human skeletons, falls outside the scope of this paper.
- The second solution is to build a 2D motion diffusion model, similar to [Kapon et al. 2024; Li et al. 2024a], conditioned on 2D keyposes and joint trajectories. After generating the 2D motion from the 2D keyposes and trajectories, the results can then be lifted to the full 3D motion. We did not evaluate this solution and leave it for future work.

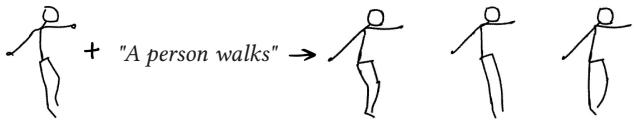
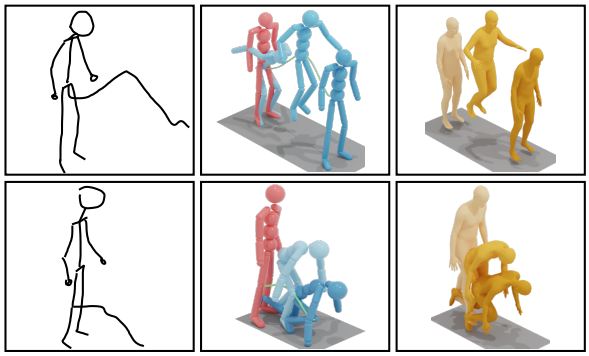


Fig. A2. **Lifting 2D sequence**. Given a sketch keypose and the action word from a storyboard (left), we exploit the SoTA method [Gal et al. 2024] to produce the animated animation sequence (right).

Speed Control. In future work, we have discussed the potential solution to handle over-sketched speed lines for motion speed control. A more straightforward way to achieve speed control in our method is non-uniform trajectory sampling. Given a fixed motion



(a) Sketch frame (b) Capsule-bone model (c) SMPL model

Fig. A3. **Motion Visualization with different human models**.

frame rate, a faster trajectory means fewer sampled points, covering a longer trajectory in a shorter time (fewer motion frames), and vice versa. In our interface, we can additionally record the user's drawing speed, sample trajectory points accordingly (instead of uniform sampling), and sequentially assign these points to corresponding motion frames. In this way, the resulting motion has the desired speed control as the drawing speed.

SMPL model visualization. As stated in Sec. 3 in the main paper, we use a capsule-bone human model in our visualization. However, other models, *e.g.*, , the SMPL human model, are also feasible for visualization purposes. Fig. A3 displays two examples visualized by both the capsule-hone human and the SMPL human, demonstrating consistent and high-quality resulting motion.

H User Evaluation Details

We conduct user studies using pairwise comparisons. In each test, participants watched two motions, generated by our model and a competitor, given the same sketch keypose, trajectory, and action word. Participants were asked to choose their preferred motion based on three evaluation criteria. A total of 21 pairs were tested, evenly divided into three groups for comparison with three competitors. We invited 58 participants to complete the survey, which was conducted online using Google Forms.

The three key criteria are motion realism, keypose accuracy, and trajectory accuracy, ensuring a comprehensive assessment of the motion quality. For motion realism, participants are asked to assess how natural and realistic the generated motion appeared, focusing on the smoothness and fluidity of movements. For keypose accuracy, they evaluate how closely the generated animation matched the keypose provided in the input sketch. The corresponding keypose in the motion sequence is highlighted in red as in the paper and video. For trajectory accuracy, participants are asked to evaluate how well the animation followed the input joint trajectory, observing whether the motion adhered to the expected direction and maintained a consistent flow. Each animation pair was presented side by side, allowing participants to compare the results objectively. For a given input, the participants should indicate their preference on all three criteria.