Abstract

Interactive motion synthesis is essential in creating immersive experiences in entertainment applications, such as video games and virtual reality. However, generating animations that are both high-quality and contextually responsive remains a challenge. Traditional techniques in the game industry can produce high-fidelity animations but suffer from high computational costs and poor scalability. Trained neural network models alleviate the memory and speed issues, yet fall short on generating diverse motions. Diffusion models offer diverse motion synthesis with low memory usage, but require expensive reverse diffusion processes. This paper introduces the Accelerated Auto-regressive Motion Diffusion Model (AAMDM), a novel motion synthesis framework designed to achieve quality, diversity, and efficiency all together. AAMDM integrates Denoising Diffusion GANs as a fast Generation Module, and an Auto-regressive Diffusion Model as a Polishing Module. Furthermore, AAMDM operates in a lower-dimensional embedded space rather than the full-dimensional pose space, which reduces the training complexity as well as further improves the performance. We show that AAMDM outperforms existing methods in motion quality, diversity, and runtime efficiency, through comprehensive quantitative analyses and visual comparisons. We also demonstrate the effectiveness of each algorithmic component through ablation studies.

1. Introduction

The landscape of interactive motion synthesis, particularly in the realm of video games, has seen a notable expansion. Today’s AAA titles boast tens of thousands of unique characters in real-time, all needing to be contextually animated [18]. Therefore, the efficiency of motion synthesis has emerged as a critical focus of research in the field of computer animation. Motion Matching [25], a prevalent technique for industry-grade animation, was first developed by UbiSoft for the game “For Honor” [1]. The main objective of Motion Matching (MM) is to identify the most contextually suitable animation in a large dataset based on manually defined motion features. This approach, while capable of yielding responsive high-quality animations, is computationally intensive and scales poorly with respect to the size of the dataset.

Alternatively, trained neural networks have emerged to reduce the memory footprints and enhance runtime performance. However, these models possess their own challenges, such as unstable convergence at training time and compromised motion quality at testing time. Recently, diffusion-based generative models have revolutionized content creation, thanks to their power to create diverse high-quality content with lean memory demands. However, standard diffusion models are often impractical for time-critical applications, due to their poor run-time performance caused by expensive reverse diffusion processes.

We introduce the Accelerated Auto-regressive Motion
Diffusion Model (AAMDM), a novel framework crafted to generate diverse high-fidelity motion sequences without the need for prolonged reverse diffusion. Diffusion-based transition models naturally produce diverse multi-modal motion would be too slow for interactive applications. To overcome this challenge, our AAMDM framework mainly adopts two synergistic modules: a Generation Module, for rapid initial motion drafting using Denoising Diffusion GANs; and a Polishing Module, for quality improvements using an Auto-regressive Diffusion Model with just two additional denoising steps. Another distinctive feature of AAMDM is its operation in a learned lower-dimensional latent space rather than the traditional full pose space, further accelerating the training process.

We evaluate our algorithm on the LaFAN1 [13] dataset and demonstrate its capability of synthesizing diverse high-quality motions at interactive rates. Our method outperforms a number of baseline algorithms, such as LMM [25], MotionVAE [31], and AMDM [50], using various quantitative evaluation metrics. Furthermore, we conduct an analysis on an artificial multimodal dataset. This analysis confirms that our model can successfully capture the multi-modal transition model and is better suited for diverse and intricate motion synthesis tasks. Finally, we perform ablation studies to justify various design choices within our framework.

In summary, our primary contributions are as follows:

• We introduce AAMDM, a novel diffusion-based framework capable of generating extended motion sequences at interactive rates. The key idea is to combine the strengths of Denoising Diffusion GANs and Auto-regressive Diffusion Models in a compact embedded space.

• We conduct thorough comparative analyses between AAMDM and various established benchmarks using multiple metrics for measuring motion quality, diversity, and runtime efficiency. Together with our ablation studies, we provide a deep understanding of our algorithm with respect to alternative prior arts.

• We showcase novel high-quality multi-modal motions synthesized from our model, some impossible to achieve by previous methods, such as following a user-controlled root trajectory with diverse arm movements.

2. Related Work

2.1. Data-Driven Kinematic Motion Synthesis

The quest to create virtual characters that move naturally stands as a fundamental challenge in computer animation. Graph-based approach structures motion data into a graph and employs search algorithms to retrieve contextually appropriate animations [4, 22, 26, 27, 35, 46]. It offers high-fidelity motion, but its scalability is curtailed by substantial memory demands and search times.

Statistical methods have been devised to encapsulate motions within numerical models, such as linear, kernel-based, and neural network categories. Linear models represent poses with low-dimensional vectors, but they often fail to encompass the full spectrum of human movement [5, 21, 47]. Kernel-based models, including Radial Basis Functions (RBF) and Gaussian Processes (GP), embrace the non-linearity in motion data [11, 28, 36, 37, 40, 45, 59]. However, these methods are memory-intensive, especially when managing large covariance matrices.

The neural network paradigm has gained prominence for its scalability and efficiency at runtime [8, 12, 14, 31, 39, 41, 42, 54, 55, 60]. Innovative neural architectures have been proposed to better capture motion sequences within datasets, such as those adjusting weights according to a phase variable [19], employing gating mechanisms [64], and extracting periodic latent features [55]. Nevertheless, these models predominantly focus on locomotion and character’s leg movement, leaving room for broader exploration.

2.2. Generative Diffusion Model

Generative diffusion models are a groundbreaking class of algorithms that learn to replicate data distributions through the reverse of diffusion processes [16, 51, 52]. In conditional generation scenarios, innovations such as classifier-guided diffusion [7] and classifier-free guidance [15] have been introduced, offering fine-tuned control over the balance between diversity and fidelity. Applications of diffusion models span across image and video synthesis to robotics [2, 15–17, 20, 23, 38, 53, 58].

Recent adaptations of diffusion models for motion synthesis have been particularly promising, with efforts aimed at generating 3D human motion from textual descriptions [24, 56, 65]. Enhancements to these models have come through novel architectural designs [56], the integration of geometric losses [56], and the incorporation of physical guidance mechanisms [63]. Additionally, the synthesis of human dance motions from audio signals has been explored, with models using auditory cues to direct the generative process [3, 6, 34, 57]. However, the latency inherent in diffusion models, often taking considerable time to generate brief motion clips, precludes their application in real-time settings. The work of Shi et al. [50] represents a significant stride towards curtailing inference times through a reduced number of denoising steps.

2.3. Accelerating Diffusion Model

The typically slow sampling speeds of diffusion models are primarily attributed to the extensive series of denoising steps required. A range of strategies has been suggested to expedite this process, such as the application of knowledge distillation techniques [33], the employment of adaptive noise scheduling [48], and the design of single-step
denoising distributions as conditional energy-based models [9]. Integrating reinforcement learning with diffusion models has also been proposed to decrease the number of reverse diffusion steps needed [50]. Nevertheless, such models have often had to contend with either diminished sample quality or expensive multiple generation steps. The introduction of Denoising Diffusion GANs [62] is a notable innovation, integrating the strengths of diffusion models with Generative Adversarial Networks to concurrently address sample quality, generation speed, and mode coverage [10]. In this work, we have employed this technique to enhance the diffusion process for fast motion synthesis.

3. Method

The architecture of our Accelerated Auto-regressive Motion Diffusion Model (AAMDM) is illustrated in Figure 2. AAMDM incorporates three key components: transition in a low-dimensional embedded space, a Generation module with Denoising Diffusion GANs for efficient draft generation, and a Polishing module with Auto-regressive Diffusion Mode for refining the draft.

In the following subsections, we will first explain the construction of the low-dimensional embedded space (Section 3.1). Then, we will describe the foundation of the Polishing (Section 3.2) and Generation modules (Section 3.3), namely the auto-regressive diffusion model and denoising diffusion GANs. Next, we will provide the design of the Generation and Polishing modules (Section 3.4), followed by an explanation of how the sampling procedure is guided to follow user’s commands (Section 3.5). Finally, we will provide a model representation (Section 3.6).

3.1. Constructing Embedded Space

Current learning-based methods for motion synthesis typically target to capture pose transitions in the full-body space, which often complicates learning and violates kinematic constraints intrinsic to the character’s morphology. We introduce a compact embedded vector \( xz \in XZ \) to replace a full-body pose \( y \in Y \), where \( x \) denotes an engineered feature and \( z \) a learned latent vector.

An autoencoder is employed to learn the optimal embedded space, where an Encoder network \( E^{AE}(y) \rightarrow z \) maps pose vectors to latent vectors, and a Decoder network \( D^{AE}(xz) \rightarrow \hat{y} \) reconstructs poses from the encoded vectors. On top of learned features \( z \), we extract manual feature \( x \) as well. The networks are trained jointly to minimize both the perceptual discrepancy losses, \( L_{vel}^{D,E} \) and \( L_{vel}^{D,E} \), and a regularization loss \( L_{reg}^{D,E} \).

\[
\begin{align*}
L_{vel}^{D,E} & = ||\hat{y} \odot y|| + ||F(\hat{y}) \odot F(y)|| \\
L_{vel}^{D,E} & = || \frac{F(\hat{y}_0) \odot F(y_1)}{\delta n} - \frac{F(y_0) \odot F(y_1)}{\delta n} || \\
L_{reg}^{D,E} & = ||z||^2
\end{align*}
\]

Here, \( F \) indicates the forward kinematics function that converts joint rotations into joint positions, and the operator \( \odot \) calculates the difference between two poses. \( y_0 \) and \( y_1 \) represent two consecutive frames of a motion sequence, \( \delta n \) denotes the time interval between frames. \( w_{vel}, w_{vel}^{D,E}, w_{reg} \) are weights for balancing different loss terms. Once we construct the embedded vector space, we can learn an embedded state transition model \( S(xzn_{n-1}) \rightarrow xzn_n \) instead of the full pose transition model \( S(y_{n-1}) \rightarrow \hat{y}_n \).

3.2. Auto-regressive Diffusion Model (ADM)

Character animations are intrinsically multi-modal. For a given pose, there may be multiple follow-up poses at the next moment. The transition from \( S(xzn_{n-1}) \rightarrow xzn_n \) is essentially a many-to-many mapping. Neural network models that use a Mean Square Error (MSE) based loss to train, such as Learned Motion Matching [25], are unable to capture these many-to-many transitions, since MSE losses work on one-to-one mappings. Therefore we employ a diffusion model as our backbone model. Our diffusion model
follows the structure of DDPM [16]. For each forward diffusion step, a small noise vector is added on top of the future embedded vector \( \mathbf{x}_{n} \):

\[
q(\mathbf{x}_{n}^{t}|\mathbf{x}_{n}^{t-1}) = N(\sqrt{\alpha^t}\mathbf{x}_{n}^{t-1}, (1-\alpha^t)I) \quad (5)
\]

The reverse diffusion phase \( p(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t}) \) generates embedded vector \( \tilde{\mathbf{x}}_{n} \) by gradually removing the noise on top of \( \mathbf{x}_{n} \). In our setting, the reverse diffusion model \( G_{ADM}^{\text{RG}} \) follows the formulation of [43, 56] and directly predicts the embedded vector rather than the added noise as in [16, 50]. The previous vector \( \mathbf{x}_{n-1} \) is used as a condition term:

\[
\tilde{\mathbf{x}}_{n}^{0} = G_{ADM}^{\text{RG}}(\mathbf{x}_{n}^{0}, \mathbf{x}_{n-1}, t) \quad (6)
\]

The predicted \( \tilde{\mathbf{x}}_{n}^{0} \) is then used as a condition \( \mathbf{x}_{n-1}^{0} = \tilde{\mathbf{x}}_{n}^{0} \) for generating the next \( \mathbf{x}_{n}^{0} \), where \( n' = n + 1 \).

To ensure high-quality generation over a long horizon, the loss for training \( G_{ADM}^{\text{RG}} \) measures the difference between the auto-regressively generated \( h \)-length embedded vector and the ground truth value. Specifically, to generate an embedded vector sequence, we start with the trajectory \( \mathbf{x}_{0:h} \) and add forward diffusion noise. This can be done in an auto-regressive manner using Equation 6 starting from the initial condition of \( (\mathbf{x}_{0}, \mathbf{x}_{1}^{t}, t_{1}) \) until \( (\mathbf{x}_{n-1}^{0}, \mathbf{x}_{n}^{h}, t_{h}) \). The loss is designed as:

\[
L_{\text{ADM}}^{\text{vel}} = \frac{1}{N} \sum_{n} \mathbb{E}_{\mathbf{x}_{n}} [\log p(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t})] \quad (7)
\]

\[
L_{\text{vel}}^{\text{vel}} = \frac{1}{N} \mathbb{E}_{\mathbf{x}_{n}} [\log p(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t})] \quad (8)
\]

\[
L_{\text{GADM}} = w_{\text{ADM}}^{\text{vel}} L_{\text{vel}}^{\text{vel}} + w_{\text{ADM}}^{\text{ADM}} L_{\text{ADM}}^{\text{ADM}} \quad (9)
\]

Here \( L_{\text{ADM}}^{\text{vel}} \) encourages the reconstruction of the trajectory, and \( L_{\text{ADM}}^{\text{ADM}} \) aims to imitate the velocities.

Although this basic diffusion model can produce high-quality samples and achieve improved mode coverage, the sampling process is time consuming primarily due to the iterative nature of diffusion and denoising.

### 3.3. Fast Generation via Denoising Diffusion GANs

The Diffusion Model typically involves multiple steps to generate solid predictions. This is based on the assumption that the denoising follows a Gaussian distribution [61]. However, this assumption is only valid when a small amount of noise is eliminated at each denoising step. As a result, it takes numerous steps to generate a high-quality prediction from pure noise. To minimize the number of steps in the reverse process and therefore accelerating the generating process, an alternative approach is to utilize a non-Gaussian multimodal distribution.

Our AAMDDM utilizes Denoising Diffusion-GANs (DD-GANs) as proposed by Xiao et al. [61]. This method formulates the reverse diffusion process using a multimodal distribution. It achieves this by parameterizing the reverse diffusion process as conditional GANs. The reverse diffusion generator, denoted as \( G_{GAN} \), takes an additional latent variable \( r \) as a conditional term, in addition to \( (\mathbf{x}_{n}, \mathbf{x}_{n-1}, t) \):

\[
\tilde{\mathbf{x}}_{n}^{0} = G_{GAN}(\mathbf{x}_{n}^{0}, \mathbf{x}_{n-1}, r^{t}, t). \quad (10)
\]

We use \( G_{GAN}(\sim) \) as an abbreviation of \( G_{GAN}(\mathbf{x}_{n}^{t}, \mathbf{x}_{n-1}, r^{t}, t) \), which can be trained by minimizing the KL divergence between two distributions: \( D_{KL}(p(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t})||q(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t}, \mathbf{x}_{n-1})) \):

\[
L_{G_{GAN}} = -\mathbb{E}_{p(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t}, \mathbf{x}_{n-1})} [\log (D_{GAN}(\sim))] \quad (11)
\]

This objective can then be converted to training a diffusion-step-dependent discriminator network \( D_{GAN}(\mathbf{x}_{n}^{t-1}, \mathbf{x}_{n}^{t}, \mathbf{x}_{n-1}, t) \) to distinguish if \( \mathbf{x}_{n}^{t-1} \) is diffused from the original data \( \mathbf{x}_{n} \) or generated fake data \( \tilde{\mathbf{x}}_{n}^{0} \), and the generator is trained to disguise the discriminator. We use \( D_{GAN}(\sim) \) as an abbreviation of \( D_{GAN}(\mathbf{x}_{n}^{t-1}, \mathbf{x}_{n}^{t}, \mathbf{x}_{n-1}, t) \).

\[
L_{D_{GAN}} = -\mathbb{E}_{p(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t}, \mathbf{x}_{n-1})} [\log (D_{GAN}(\sim))] - \mathbb{E}_{p(\mathbf{x}_{n}^{t-1}|\mathbf{x}_{n}^{t}, \mathbf{x}_{n-1})} [\log (1 - D_{GAN}(\sim))] \quad (12)
\]

DD-GANs offer high sampling speed while maintaining excellent mode coverage and output quality in a single motion step. However, when used in an autoregressive generation setting, we have observed that DD-GANs often lead to unstable training, resulting in deteriorated motion quality. To address this issue, we propose combining ADM and DD-GANs to achieve fast and high-quality sampling.

### 3.4. Combining ADM and DD-GANs

The combination of ADM and DD-GANs is based on the insight that the diffusion process transitions from generating samples from noise at early stages to making small adjustments in the prediction at late stages. To achieve higher quality output, the generation of single motion steps is divided into two sub-steps: Generation and Polishing. The Generation module utilizes DD-GANs to generate a draft prediction in a few steps, while the Polishing module refines the output from the Generation module using ADM.

The process begins with a random noise input, \( \mathbf{x}_{n}^{0} \sim N(0, I) \), and the Generation module goes through \( T_{GAN} \) to 3 reverse diffusion steps using \( G_{GAN} \) for \( \mathbf{x}_{n}^{T_{GAN}} \). The generated \( \mathbf{x}_{n}^{T_{GAN}} \) is then passed to the Polishing module, where \( G_{ADM} \) refines the result using \( T_{ADM} = 2 \) steps. The total number of generation steps, \( T_{AA} = T_{GAN} + T_{ADM} = 5 \). Finally, the generated \( \tilde{\mathbf{x}}_{n}^{0} \) replaces \( \mathbf{x}_{n}^{0} \) for the next step prediction. The Generation Module and Polishing Module are trained separately using Equation 7, 12, and 11. For a more detailed sampling and training procedure, refer to the supplementary materials.
3.5. Motion Control with User Commands

To generate the motions that follow the user’s commands, for single pose transition, AAMDM guides the motion generation process through a guided diffusion method proposed by Rempe et al. [44]. Given the user’s query \( \mathbf{x}_n \), at each diffusion steps with noise vector \( \mathbf{xz}_{n}^{0} \), we perturb the generated vector \( \mathbf{xz}_{n}^{0} \) to obtain the guided vector \( \mathbf{xz}_{n}^{0,*} \):

\[
\mathbf{xz}_{n}^{0,*} = \mathbf{xz}_{n}^{0} - \epsilon \alpha t \nabla_{\mathbf{xz}_{n}} J(\mathbf{xz}_{n}^{0}, \mathbf{x}_n).
\]

Here, \( J \) is an objective function measuring distance between the generated feature vector and user’s query. \( \epsilon \) is a step parameter, \( \alpha t \) is the noise parameter in diffusion model.

3.6. Model Representation

Character Representation The pose vector, denoted as \( y \), captures all the character’s pose information in a single frame of the animation. It is defined as \( y = \{ y^{t}, y^{r}, \hat{y}^{t}, \hat{y}^{r}, \hat{r}^{t}, \hat{r}^{r} \} \), where \( y^{t} \) and \( y^{r} \) represent joint local translations and rotations, \( \hat{y}^{t} \) and \( \hat{y}^{r} \) represent joint local translational and rotational velocities, and \( \hat{r}^{t} \) and \( \hat{r}^{r} \) represent root translational and rotational velocities. The total dimension of \( y \) is 338. Additionally, we define \( x = \{ t^{i}, t^{d} \} \), where \( t^{i} \in \mathbb{R}^{6} \) and \( t^{d} \in \mathbb{R}^{6} \) represents the 2D future trajectory positions and facing direction projected on the ground, 20, 40, and 60 frames in the future local to the character. The latent vector \( z \) has a dimension of 52, and thus \( \mathbf{xz} \in \mathbb{R}^{64} \).

Neural Network Structure The encoder network \( E^{AE} \), the decoder network \( D^{AE} \), ADM generator \( G^{GAN} \), DD-GANs generator \( G^{GAN} \) and DD-GANs discriminator \( D^{GAN} \) are all fully connected neural network. The details are presented in the supplementary material.

4. Experiments

We conducted a series of experiments to evaluate the performance of the proposed method, AAMDM. Firstly, AAMDM is quantitatively compared against several baseline methods using different evaluation metrics. Subsequently, we conducted additional experiments on an artificial multi-modal dataset for detailed discussion. Lastly, we performed ablation studies to justify design choices. Overall, the results demonstrate that AAMDM can efficiently generate high-quality motions with long horizons auto-regressively. The motions can be seen in the supplementary video.

Implementation Details We implemented our motion generation framework in Pytorch and conducted experiments on a PC equipped with an NVIDIA GeForce RTX 3080 Ti and AMD Ryzen 9 3900X 12-Core Processor. For all networks, training was performed for 1M iterations using the RAdam optimizer [32] with a batch size of 64 and a learning rate of 0.0001. We trained the Encoder \( E^{AE} \) and Decoder \( D^{AE} \) first to construct the embedded vector space \( \mathbf{XZ} \), then we trained the Polishing module and the Generation module. Both Polishing and Generation modules were trained with a window size of 10 frames. The total training procedure took around 20 hours.

Dataset We utilized the Ubisoft LaForge Animation Dataset (“LaFAN1”) [13] for evaluation. LaFAN1 is a collection of high-quality human character animations, encompassing a wide range of motions. Our dataset comprised 25 motion clips from LaFAN1, featuring 100,000 pose transitions, and had a total duration of 26.67 minutes.

4.1. Baseline Comparison

We compared AAMDM with the following baselines:

- **Learned Motion Matching (LMM):** LMM is an interactive motion synthesis method proposed by Kolsi et al. [25]. Similar to our method, LMM uses an embedded vector space. It comprises three networks: Projector that maps the human input vector \( \mathbf{x} \) to the embedded vector \( \mathbf{xz} \) for addressing user’s command, Decompressor that reproduces the pose vector \( y \) from \( \mathbf{xz} \), and Stepper that maps \( \mathbf{xz}_{n-1} \) to \( \mathbf{xz}_{n} \) for learning the pose transition. Both Stepper and Projector are trained using MSE based loss. Unlike LMM that treats user commands and pose transition separately, AAMDM fuses these two requirement using the guided diffusion process.

- **Motion VAE (MVAE):** MVAE [31] is based on an autoregressive conditional variational autoencoder. Given the current pose, MVAE predicts a distribution of possible next poses, as it is conditioned on a set of stochastic latent variables. The key distinction between AAMDM and MVAE is that the former models transitions using a diffusion-based model, whereas the latter employs a VAE.

- **Autoregressive Motion Diffusion Model (AMDM):** AMDM [50] is an autoregressive diffusion model-based framework for motion synthesis. There are three main differences between AAMDM and AMDM. First, AMDM accelerates the diffusion process by simply taking fewer reverse diffusion steps, while AMDM leverages DD-GANs. Second, AMDM operates in the full pose space, whereas AAMDM learns transitions in an embedded space. Third, AMDM predicts noise at each reverse diffusion step, while AMDM directly predicts the target vector as Ramesh et al. [43]. We implemented two versions of AMDM, named AMDM5 and AMDM200, to indicate the use of 5 and 200 diffusion steps, respectively.

4.1.1 Evaluation of Random Motion Synthesis

We first evaluated the performance of these methods in random motion generation over the following metrics:
Table 1. In our quantitative analysis, we demonstrate that the AAMDM framework is capable of generating motions of a quality comparable to that of AMDM200, while significantly outperforming other methods in both random sampling and user control scenarios. Meanwhile, the result also indicates that AAMDM is approximately 40 times faster than AMDM200.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DIV</th>
<th>FID</th>
<th>FFR</th>
<th>FPS</th>
<th>TE-UC</th>
<th>FID-UC</th>
<th>FFR-UC</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAMDM(Ours)</td>
<td>11.574</td>
<td>14.051</td>
<td>0.131</td>
<td>173</td>
<td>0.034</td>
<td>15.367</td>
<td>0.143</td>
</tr>
<tr>
<td>LMM [25]</td>
<td>5.374</td>
<td>49.706</td>
<td>0.194</td>
<td>812</td>
<td>0.021</td>
<td>55.674</td>
<td>0.233</td>
</tr>
<tr>
<td>MVAE [31]</td>
<td>7.223</td>
<td>22.981</td>
<td>0.312</td>
<td>703</td>
<td>0.027</td>
<td>47.453</td>
<td>0.349</td>
</tr>
<tr>
<td>AMDM5 [50]</td>
<td>8.134</td>
<td>18.741</td>
<td>0.214</td>
<td>192</td>
<td>0.054</td>
<td>25.943</td>
<td>0.256</td>
</tr>
<tr>
<td>AMDM200 [50]</td>
<td>11.165</td>
<td>12.132</td>
<td>0.129</td>
<td>4.72</td>
<td>0.012</td>
<td>14.254</td>
<td>0.133</td>
</tr>
</tbody>
</table>

Diversity (DIV): Diversity measures the distributional spread of the generated motions in the character pose space. This metric, adopted from several previous works [29, 30, 56, 57], assesses how well the generated motion matches the distribution of the ground truth dataset. We follow the implementation used in MDM [56], computing Diversity using 1,000 frames from each generated motion clip. A good Diversity score should be close to that of the motion dataset.

Frechet Inception Distance (FID): FID evaluates the difference between the distributions of generated and ground truth motions. FID serves as an indicator of the overall quality of generated motions in many prior works [56, 57].

Footskating Frame Ratio (FFR): FFR quantifies the realism of generated motion, particularly focusing on foot-ground contact. We measured foot skating artifacts as described in Zhang et al. [64]. A lower FFR score indicates better physical plausibility of the generated motions.

Frames Per Second (FPS): FPS is a measure of the efficiency of motion generation methods in creating new frames. Higher FPS values indicate faster frame generation rates, essential for interactive applications.

The qualitative results are summarized in Table 1. Notably, AAMDM achieve similar performance as AMDM200 while surpasses the other baselines in all motion quality metrics with more than 40 times faster than AMDM200. This demonstrates AAMDM’s capability to efficiently generate high-quality character animations.

LMM’s motion quality was generally found to be inferior to AAMDM, as reflected in the FID and DIV metrics. This discrepancy is likely due to LMM’s training with MSE loss, presupposing a one-to-one mapping. However, this assumption may not be valid in datasets with multiple possible transitions from a single pose. Figure 3 provides an example. A more detailed discussion on this aspect will be presented in a subsequent section. However, LMM showed a higher FPS score, attributed to its single feed-forward operation, compared to AAMDM’s five feed-forward operations.

In motion quality evaluation, MVAE slightly outperformed LMM with scores of 7.223 and 22.981 in DIV and FID respectively. MVAE’s better quality can be linked to its use of VAE for handling multiple mappings in pose transitions. Although MVAE offered improved training stability and performance, AAMDM still outperformed in these metrics. MVAE also exhibited faster performance than AAMDM due to its single-step feedforward process.

In comparison between AMDM5 and AAMDM, both methods used 5 diffusion steps which led to similar FPS scores (173 vs 192). However, the diffusion steps in AMDM5 were modeled using a Gaussian distribution, which is typically effective when the total number of denoising steps is in the order of hundreds. As AMDM5 utilized only five steps, this assumption did not hold and it led to compromised motion quality. On the other hand, AAMDM leveraged DD-GANs to model multimodal transitions, which reduced the number of steps required for generating a new frame without sacrificing motion quality.

AMDM200 with more diffusion steps is better aligned with the Gaussian distribution assumption, which is connected to highly improved motion quality metrics. However, this increase in diffusion steps comes at the cost of efficiency. As the number of steps rises, the generation speed decreases. This trade-off highlights the balance between motion quality and generative efficiency, with AMDM200
favoring the former at the expense of the latter.

4.1.2 Evaluation of Interactive Synthesis

We evaluated the performance of these methods in an interactive motion synthesis scenario. The experiment involved interactively controlling the character’s moving direction while allowing the arms to move freely. Our evaluation employed the following metrics, with ‘-UC’ denoting ‘Under Control’:

- **Tracking Error (TE-UC)**: The TE-UC metric assesses the method’s ability to follow user commands $\bar{x}$. It is defined as the discrepancy between the user’s command and the generated motion $|\bar{x} - \hat{x}|$. A lower TE-UC value signifies better alignment with user input, reflecting superior performance.

- **Frechet Inception Distance (FID-UC)**: The FID-UC is used to measure the similarity between the motion dataset and the generated trajectories. A lower FID-UC indicates a higher quality of the generated motion.

- **Footskating Frame Ratio (FFR-UC)**: This metric evaluates the realism of the motion when the character is under user control. It assesses aspects such as naturalness and adherence to physical constraints. Lower FFR-UC scores suggest more physically plausible and realistic motion generation.

In user control scenarios, our results demonstrate that the AAMDM framework consistently outperformed baseline methods across nearly all metrics evaluated. Compared with Learned Motion Matching (LMM), AAMDM addresses several key issues inherent in the LMM’s approach. LMM employs a projector network trained with an MSE loss to interpret user commands, which leads to two primary issues. Firstly, multiple candidate poses could potentially match the user command, but LMM’s projector network struggles to handle multi-modal transitions. Secondly, the projector network often ignores the character’s current pose, necessitating blending techniques to ensure smooth transitions. MVAE faces challenges in training to capture all the possible transitions, resulting in a quality of motion that does not match that of AAMDM. Similarly, AMDM5’s reduces the number of diffusion steps, which breaks the Gaussian distribution assumption and consequently downgrades the motion quality. Although AMDM200 provides higher-quality generation due to more diffusion steps, its low speed (4.72 FPS) is not suitable for any interactive applications.

4.2. Additional Studies on Artificial Dataset

In addition to the previous experiment, we conducted an additional study to analyze the effectiveness of various methods on a many-to-many transition dataset. For this purpose, we created a 2D “Squ-9” dataset characterized by its multi-modal dynamics where any given point in three by three Gaussian distributions can transit to any other Gaussian distributions in the next time step. By learning this dataset, we evaluated the effectiveness of each method to capture this many-to-many dynamics. The comparative results are visually depicted in Figure 4.

In our results, our AAMDM captured all the possible modes while preserving sample quality. In contrast, LMM struggled to represent the dataset’s many-to-many vector transitions, resulting in a singular vector cluster at each step. MVAE showed an improvement in mode coverage, yet it cannot illustrate all possible modes. Among other diffusion model-based approaches, AMDM5 exhibited better transitions but their qualities are still worse than AAMDM. Although AMDM200 produced results of comparable quality to AAMDM, it required 40 times more inference time.

4.3. Ablation Studies

We provided additional insights of AAMDM by conducting three ablation studies summarized in Table 2.
Polishing Steps In our study, we investigated the impact of the number of polishing steps \((T_{\text{ADM}})\) on the generation process. Specifically, we denote \(T_{\text{ADM}} = 0\) as the scenario where no polishing module is used, meaning the output from the generation module is directly utilized for future frame generation. In our experiments, settings with \(T_{\text{ADM}} > 0\) exhibited significant performance enhancements compared to the \(T_{\text{ADM}} = 0\) scenario as when \(T_{\text{ADM}} = 0\), the framework was unable to generate reliable long-horizon trajectory due to the diverges of the character’s pose. This suggests that relying solely on denoising diffusion GANs may not yield high-quality outputs for long-horizon generation. In contrast, additional polishing steps markedly improved the output quality, making it more suitable for long-horizon predictions. Furthermore, results indicate a positive correlation between the number of polishing steps \(T_{\text{ADM}}\) and the output quality. However, it is important to note that increasing \(T_{\text{ADM}}\) also leads to longer sampling times.

Generation Steps In our second study, we examined the effects of the number of generation steps. Theoretically, increasing \(T_{\text{GAN}}\) should reduce the amount of noise that needs to be removed at each denoising step, potentially simplifying the training process. However, our results show that a specific value of \(T_{\text{GAN}}\), \(T_{\text{GAN}} = 3\) and \(T_{\text{GAN}} = 4\), yielded the highest overall motion quality, yet \(T_{\text{GAN}} = 3\) was more efficient. Although \(T_{\text{GAN}} = 2\) achieved the best performance in the DIV metric, we observed a few cases of divergence in the motion, which resulted in worse performance in FID compared to \(T_{\text{GAN}} = 3\). When \(T_{\text{GAN}} = 10\), the learning task should be easier since the distance between each diffusion step is smaller. However, our results show that the performance was the worst. We hypothesize that this is because we utilized a simple MLP network; thus, it may not be adequately equipped to handle larger values of \(T_{\text{GAN}}\) for effectively training the denoising diffusion GANs.

### Importance of Embedded Transition Space

In this analysis, we explored the advantages of learning transitions in an embedded space \(XZ\) as opposed to the full pose space \(Y\). Our results suggest that utilizing the full pose space yields inferior outcomes compared to an embedded space. We attribute this finding to two primary factors. Firstly, learning in a higher-dimensional space, like the full pose space, is inherently more challenging than in a lower-dimensional space, particularly under multimodal distribution conditions. Secondly, as discussed in the previous section, \(AAMDM\) does not employ a complex neural network architecture or specialized techniques for constructing the latent space in Denoising Diffusion GANs. MLP networks used in our framework may not be sufficiently robust to capture transitions in larger spaces effectively. This limitation further supports the advantage of using an embedded space for learning transitions.

### Table 2. Ablation study results. *The default parameters.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>DIV →</th>
<th>FID ↓</th>
<th>FPS ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>14.533</td>
<td>0.000</td>
<td>N/A</td>
</tr>
<tr>
<td>1</td>
<td>10.612</td>
<td>16.332</td>
<td>215</td>
</tr>
<tr>
<td>2*</td>
<td>11.574</td>
<td>14.051</td>
<td>173</td>
</tr>
<tr>
<td>10</td>
<td>12.041</td>
<td>11.779</td>
<td>47</td>
</tr>
<tr>
<td>2</td>
<td>12.415</td>
<td>28.476</td>
<td>211</td>
</tr>
<tr>
<td>2*</td>
<td>11.574</td>
<td>14.051</td>
<td>173</td>
</tr>
<tr>
<td>2</td>
<td>11.313</td>
<td>14.534</td>
<td>135</td>
</tr>
<tr>
<td>2</td>
<td>9.775</td>
<td>16.312</td>
<td>58</td>
</tr>
<tr>
<td>wo/ Emb</td>
<td>56.341</td>
<td>128.412</td>
<td>146</td>
</tr>
</tbody>
</table>

We have introduced a novel framework for motion synthesis: Accelerated Auto-regressive Motion Diffusion Model (AAMDM). AAMDM is designed to efficiently generate high-quality animation frames for interactive user engagement. This is achieved by several technical components: the use of a low-dimensional embedded space for compact representation, Denoising Diffusion GANs for fast approximations, and the Diffusion Model for robust and accurate long-horizon synthesis. Our benchmarking of \(AAMDM\) against various baseline methods has demonstrated its superior capabilities in motion synthesis. We have also investigated the nuances of different autoregressive motion synthesis methods, providing valuable insights into this domain. Additionally, our ablation studies have validated the design choices made for \(AAMDM\) and identified the influence of various hyperparameters on the overall system performance.

In the future, we plan to explore several research directions. One notable challenge is the trade-off between the motion quality and the computational cost. Future work could explore advanced techniques such as parallel computing and the use of temporal information to accelerate the generation process. In addition, the model performance can be further improved by introducing more sophisticated methods to structure the latent space of the denoising diffusion GANs, such as a structured matrix-Fisher distribution [49]. Finally, it will be interesting to improve the controllability of the framework by introducing a learning-based control mechanism rather than relying on gradient-based sampling guidance.

5. Discussion and future work
References


