VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF SCIENCE

THANH HOANG-MINH

OPENHUMAN: A CONVERSATIONAL GESTURE SYNTHESIS SYSTEM BASED ON EMOTION AND SEMANTICS

MASTER'S THESIS

Ho Chi Minh City – 2024

VIETNAM NATIONAL UNIVERSITY, HO CHI MINH CITY

UNIVERSITY OF SCIENCE

THANH HOANG-MINH

OPENHUMAN: A CONVERSATIONAL GESTURE SYNTHESIS SYSTEM BASED ON EMOTION AND SEMANTICS

Major: Computer Science

Program Code: 8480101

ACADEMIC SUPERVISOR

Assoc. Prof. Dr. Ngoc Quoc Ly

Ho Chi Minh City - 2024

DECLARATION

I hereby declare that this Master's thesis in Computer Science, titled *OpenHuman: A Conversational Gesture Synthesis System Based on Emotion and Semantics*, is my own scientific work carried out under the supervision of Dr. Ngoc Quoc Ly.

All research results presented in this thesis are entirely truthful and accurate.

Ho Chi Minh City, day...month...year...

CANDIDATE

(Signature and full name)

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to Mr. Ngoc Quoc Ly for his dedicated guidance, for sharing his knowledge and experience, and for offering valuable solutions throughout the process of completing this graduation thesis.

My heartfelt thanks also go to the lecturers of the Faculty of Information Technology at the University of Science – Vietnam National University, Ho Chi Minh City, who have imparted invaluable knowledge to me during my studies.

I would also like to thank the researchers whose work I have cited and built upon to complete this thesis.

Finally, I am deeply grateful to my family, friends, and others who have always supported and encouraged me during the course of this thesis.

Once again, my sincere thanks to all!

Table of Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

LIST OF SYMBOLS AND ABBREVIATIONS

Symbol	Full Description
θ	Trainable weights to be learned
θ'	Learned weights used during sampling
ϵ	Noise added to the image
$\hat{\epsilon}$	Predicted noise
ϵ_θ or f_θ	Noise prediction function in DDPM with parameter θ, where f_θ and ϵ_θ are equivalent
N, M, D	Numbers indicating the dimensions of matrices or vectors
$t : 1 \to T$	Noise step t from $t = 1$ to $t = T$
n_{joins}	Number of joints
$\mathcal{N}(a\mathbf{x}, b^2)$	The function $f(x) = a\mathbf{x} + b\epsilon$ with $\epsilon \in \mathcal{N}(0, \mathbf{1})$ is equivalent to the normal distribution $\mathcal{N}(a\mathbf{x}, b^2)$
$\mathcal{N}(0, \mathbf{I})$	Standard normal distribution with mean 0 and variance 1
$\mathcal{N}(\mathbf{x}_t; \mu_\theta, \Sigma_\theta)$	Normal distribution where $\mathbf{x}_t$ is the output, μ_θ is the mean, and Σ_θ is the variance parameterized by θ
$\mathbf{x}_t^{1:M}$	Gesture data at time step t, from frame 1 to frame M
$\hat{\mathbf{x}}_0$	Predicted data by the model
$f(x\|y)$	Conditional probability function: given y, find the probability of x
$q(\mathbf{x}_t\|\mathbf{x}_{t-1})$	Conditional probability function in the forward diffusion process, given $\mathbf{x}_{t-1}$, compute $\mathbf{x}_t$
$p_\theta(\mathbf{x}_{t-1}\|\mathbf{x}_t)$	Conditional probability function in the denoising process: given $\mathbf{x}_t$, compute $\mathbf{x}_{t-1}$ with learnable parameter θ
$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t)$	Mean of $\mathbf{x}_t$ at time step t after passing through the model with parameter θ
$\mathbb{E}$	Expectation
$\mathcal{L}^t$	Loss function at time step t
G_θ	The OHGesture model to be trained with parameter θ
$\prod_{t=1}^{T}$	Product operator: $\prod_{t=1}^{T} a_t = a_1 \cdot a_2 \cdot \ldots \cdot a_T$

GLOSSARY OF TERMS

Abbreviation	**Full Meaning**
DDPM	Denoising Diffusion Probabilistic Models
OHGesture	Proposed model of the thesis
Sampling/Inference	The process of sampling or inference in the model
KL	Kullback–Leibler Divergence
MAE	Mean Absolute Error
MSE	Mean Squared Error
Forward Diffusion Process	A noise-adding process without weight-based training

THESIS INFORMATION

Thesis title: OpenHuman: A conversational gesture synthesis system based on emotions and semantics

Speciality: Computer Science

Code: 8480104

Name of Master Student: Hoàng Minh Thanh

Academic year: 31

Supervisor: Associate Professor Ly Quoc Ngoc

At: VNUHCM - University of Science

1. SUMMARY:

Along with the explosion of hardware advancements, large language models, text-based AI systems such as ChatGPT, CharacterAI, Gemini, and the development of computer graphics, the current bottleneck in creating digital humans lies in generating character movements corresponding to text or speech inputs.

One major challenge in gesture generation is the lack of sufficient and high-quality gesture data, as well as the lack of contextual consistency in gestures, making system development more difficult. Among current approaches, diffusion models achieve the best results due to their ability to generalize and handle imbalanced feature distributions and low-data-density regions effectively.

To generate gestures corresponding to speech and emotional information, this thesis employ a conditional diffusion model. The conditions include initial gestures, emotions, speech, and corresponding text. This thesis extends the DiffuseStyleGesture model to propose the **OHGesture** model. The thesis method converts speech into text as an additional semantic feature for training. We leverage the Unity codebase of the DeepPhase model for rendering and visualizing character movements. Lastly, this thesis make all our source code publicly available to encourage further development of gesture generation systems by the research community.

2. NOVELTY OF THESIS:

First, through actual gesture generation results, this thesis demonstrate that the proposed OHGesture model achieves high-quality gesture generation with synchronization between gestures, speech, and emotions.

Second, by integrating text to supplement semantic features, this thesis provide an additional mechanism to help the model understand specific contexts during gesture

generation.

Third, this research contributes a novel method for constructing standardized evaluation systems for gesture generation models. By combining data from multiple linguistic sources and utilizing human evaluations, this thesis establish an objective and comprehensive benchmarking platform.

Finally, this thesis utilize animated character visualization, source code hosted on GitHub, and pretrained models on Huggingface. The proposed model achieves favorable results even with inputs outside the training dataset.

3. APPLICATIONS/ APPLICABILITY/ PERSPECTIVE:

The OHGesture model is capable of generating gestures from emotion labels, speech, and corresponding text. This enables the development of human-interactive systems by integrating it with other text-based processing agents like Character.AI or the ChatGPT API. Additionally, we envision creating a store similar to the App Store but dedicated to virtual humans. Each virtual human, acting as an app, could be developed by various contributors and serve diverse applications such as virtual girlfriends, education, and customer assistance.

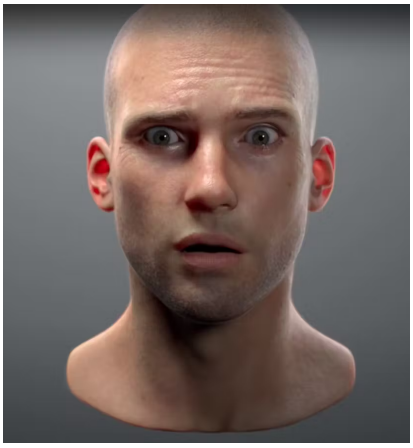
Currently, the model treats gesture sequences as images for denoising, which limits its performance. We plan to improve it by employing Fast Fourier Transform (FFT) to extract phase features of movements, aiming to develop a better gesture generation system.

SUPERVISOR	**Master STUDENT**
(Signature, full name)	(Signature, full name)

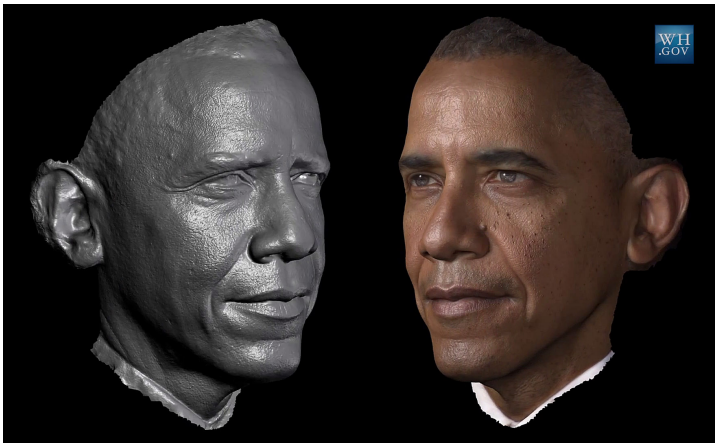
CERTIFICATION
CERTIFICATION UNIVERSITY OF SCIENCE
PRESIDENT

Chapter 1. INTRODUCTION

1.1 General Background



(a) CGI technology with realistic digital humans [23]



(b) Illustration of the reconstruction of President Obama's face [34]

Figure 1.1: Illustration of computer graphics techniques in constructing digital humans

Every day, billions of people around the world look at RGB screens, where the display results are outputs of various software systems. Thus, pixel-level rendering on the screen and how to simulate it most realistically has been a topic of computer graphics research since the 1960s, especially the simulation of humans. As early as 2014, 3D artists were able to create hyper-realistic digital characters like in Figure 1.1a, even when computing hardware was not as advanced as today.

Today, computer graphics technology is capable of simulating many objects to a hyper-realistic level, including complex ones like water, roads, bread, and even the human body and face down to fine details such as pores, acne, and eye patterns. In 2015, using 3D scanning techniques to capture all facial angles and light reflections, researchers in [34] managed to reconstruct the entire face of President Obama on a computer with high accuracy and near-indistinguishability (Figure 1.1b).

Artificial intelligence (AI) has made remarkable advances in recent years, not only in research but also in real-world applications, such as ChatGPT, MidJourney, and across many vertical and horizontal domains. While computer graphics can now construct hyper-realistic faces, generating gestures still relies heavily on motion capture using sensors, and building AI systems that can learn from data remains challenging.

Modern AI systems can now generate human-like text and speech. However, one of the biggest obstacles in building digital humans is gesture generation. Therefore, the aim of this thesis is to build an emotional and semantic-conditioned conversational gesture generation system using both text and speech as input.

1.2 Research Motivation

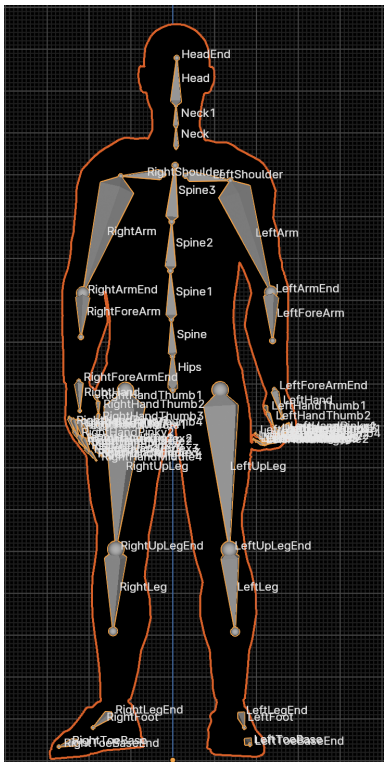
Conversational gesture generation benefits many fields such as animation, filmmaking, video games, education, and virtual reality applications. Traditionally, gesture motion is captured by hiring actors equipped with motion tracking devices and surrounding sensors to record movements with high realism. However, these recorded motions are replayed statically and lack flexibility between actions. Therefore, applying AI to learn motion from data and generate new motions will be a revolution in the motion capture industry.

In 2011, a group of researchers [5] demonstrated a correlation between human speech and gestures, forming the foundation to learn and represent gestures from speech data. With the success of natural language models in text processing and the hyper-realistic simulation of human faces, computer graphics has achieved significant progress. Combined with the ease and accuracy of human speech synthesis, gesture generation through AI has become one of the main bottlenecks in developing interactive virtual assistants.

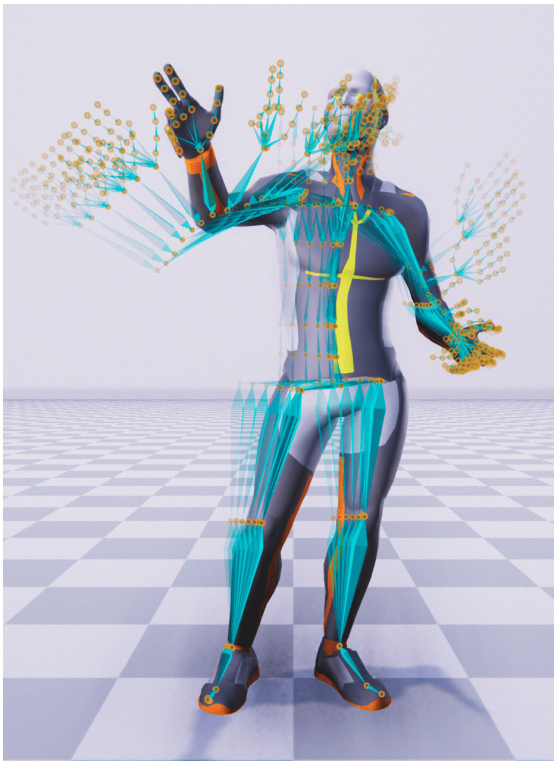
1.3 Problem Data

1.3.1 Skeleton Structure of Gestures

In this thesis, a gesture is defined as the movement of a character's entire body over time, as shown in Figure 1.2a, captured frame by frame. In computer graphics, character motion is represented as bone-specific movements, including hands, legs, head, spine, etc. The full character skeleton structure is presented in Appendix Section B.1.



(a) Skeleton and joint names of a frame



(b) Motion sequence consisting of 6 past and 6 future gestures

Motion data is captured via motion capture systems using cameras and specialized sensors. The output is typically stored in BVH (Biovision Hierarchy) files.

A BVH file consists of two main parts: 'HIERARCHY' and 'MOTION'. The 'HIERARCHY' section is structured as a tree containing the skeleton's initial positions and names. The 'MOTION' section contains the movement data for the entire skeleton frame-by-frame. Each BVH file includes frame rate (fps) and total frame count. Details are presented in Appendix Section B.2.

1.3.2 Motion Structure of Gestures

Gesture motion data, as illustrated in Figure 1.2b, or the 'MOTION' section of a BVH file, contains position and rotation information per frame. Each frame is a skeleton of 75 bones: $\{\mathbf{b}_1, \mathbf{b}_2, \cdots, \mathbf{b}_{75}\}$, where each bone has position $\{p_x, p_y, p_z\}$ and rotation $\{r_x, r_y, r_z\}$.

The output of gesture generation is a sequence of bone rotations per frame. The generated gestures are evaluated based on naturalness, human-likeness, and contextual appropriateness.

In this thesis, the skeleton's position and rotation data are preprocessed into a feature vector $\mathbf{g} \in \mathbb{R}^D$ with $D = 1141$. The learning data becomes $\mathbf{x} \in \mathbb{R}^{M \times D}$. The preprocessing pipeline is detailed in Appendix B.

1.4 Problem Statement

The ultimate goal is to produce a sequence of gestures that reflect the motion of the skeleton frame by frame. This can be approached via classification, clustering, or regression. In this thesis, gesture generation is framed as a regression problem: given an input gesture sequence, predict the next sequence.

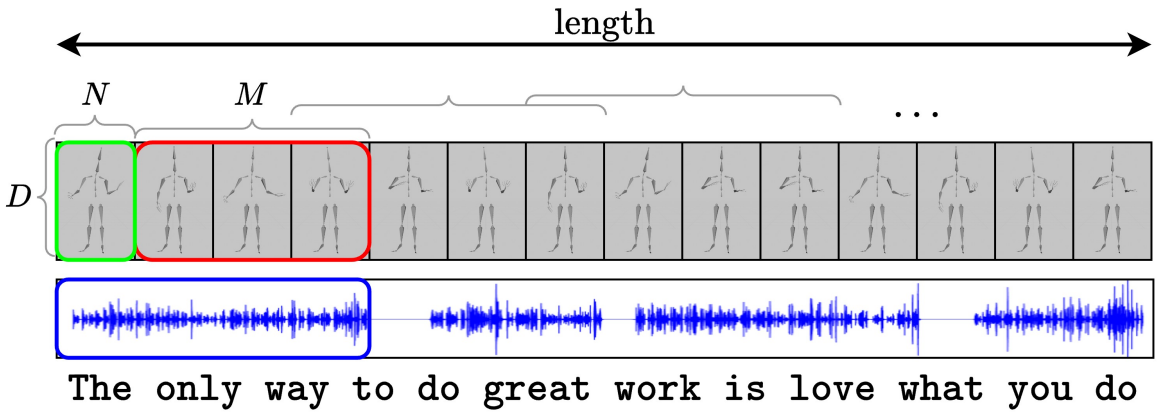


Figure 1.3: A gesture sequence: the first N frames are used as seed gesture $\mathbf{s}$, and the remaining M frames are to be predicted

Each gesture sequence is labeled with an emotion. A key novelty of this thesis is pairing the gesture sequence with both the original speech and the corresponding text (transcribed from the speech).

The objective is to build a model that predicts M future frames from the given inputs: seed gesture $\mathbf{s} \in \mathbb{R}^{1:N \times D}$, speech $\mathbf{a}$, text $\mathbf{v}$, and emotion $\mathbf{e}$.

The model prediction is $\hat{\mathbf{x}} \in \mathbb{R}^{1:M \times D}$, which is compared against ground-truth gesture $\mathbf{x} \in \mathbb{R}^{1:M \times D}$.

Input

- Seed gesture sequence: $\mathbf{s} \in \mathbb{R}^{1:N \times D}$
- Speech signal: $\mathbf{a}$
- Text: $\mathbf{v}$
- Emotion: $\mathbf{e}$

  (`Happy`, `Sad`, `Neutral`, `Angry`, `Old`,

`Funny`)

Predicted Output

- Predicted gesture sequence: $\hat{\mathbf{x}} \in \mathbb{R}^{1:M \times D}$

Ground Truth

- Ground truth gesture: $\mathbf{x} \in \mathbb{R}^{1:M \times D}$

1.5 Challenges

There are several challenges in building a model that can learn human-like conversational gesture patterns:

1. *Limited and low-quality data:* Creating large-scale, high-quality datasets for motion capture is extremely costly in the industry.

2. *Inconsistent context between modalities:* Text datasets are more abundant than speech, and speaker attribution is often missing. Synchronization between speech and emotional tone is also lacking. Additionally, training texts span many unrelated topics.

3. *Imbalanced feature distributions:* Current datasets are biased toward English-speaking gestures, with imbalanced gesture distributions between speaking, questioning, and silent states.

4. *High computational cost due to multimodal input:* The model must encode text, speech, and 3D pose data, increasing the computational load during both training and inference. Reducing input information also degrades performance.

5. *Sequential preprocessing steps:* Although human-computer interaction is most effective through speech and keyboard input, processing the text and speech input for gesture generation must be done sequentially. In real-world applications, inference latency is critical, and users cannot wait long. Rendering the gestures on screen must also be optimized for speed.

1.6 Expected Contributions

- From existing datasets, speech is transcribed into text and used as additional semantic features for training.

- Based on the DiffuseStyleGesture model, the thesis extends the conditional denoising process to include text features.

- The thesis uses Unity for rendering, data extraction, and visualization of gesture generation results.
- The thesis develops a rendering pipeline and demonstrates the system with Unity.

Chapter 2. RELATED WORK

The gesture generation problem, like other machine learning tasks, has been studied and developed alongside both traditional and modern machine learning methods. These include rule-based and data-driven approaches. This thesis first presents the characteristics of gestures Section 2.1, serving as the foundation for learning the relationship between gestures, text, and speech. In section Section 2.3, the thesis introduces the common stages across different gesture generation methods.

In section Section 2.2, the thesis presents the methods that have been applied in gesture generation. A comparison of these methods is provided, leading to the motivation for adopting diffusion models for this task.

Section Section 2.4 explains how diffusion models have recently been applied to gesture generation.

2.1 Gesture Characteristics

According to linguistics, gestures can be categorized into six main groups: adaptors, emblems, deictics, iconics, metaphorics, and beats [15], [41]. Among them, beat gestures do not carry direct semantic meaning but play an important role in synchronizing rhythm between speech and gesture [25], [41]. However, the rhythm between speech and beat gestures is not fully synchronized, making the temporal alignment between them complex to model [33], [6], [28], [55].

Gestures interact with various levels of information in speech [41]. For instance, emblematic gestures like a thumbs-up are usually linked to high-level semantic content (e.g., "good" or "excellent"), while beat gestures often accompany low-level prosodic features such as emphasis. Previous studies typically extract features from the final layer of the speech encoder to synthesize gestures [1], [6], [29], [38], [56]. However, this approach may blend information from different levels, making it hard to disentangle rhythm and semantics.

As shown in linguistic studies [25], [36], [46], gestures in daily communication can be decomposed into a limited number of semantic units with various motion variations. Based on this assumption, speech features are divided into two types: high-level features representing semantic units and low-level features capturing motion variations. Their relationships are learned across different layers of the speech encoder. Experiments demonstrate that this mechanism can effectively disentangle features at different levels and generate gestures that are both semantically and stylistically appropriate.

2.2 Overview of Gesture Generation Methods

2.2.1 Rule-Based Methods

2.2.1.1 General Principle

These methods rely on clearly defined rules, which are manually crafted to determine how the system processes inputs to produce outputs.

2.2.1.2 Methods

Representative rule-based methods include the *Robot behavior toolkit* [22] and *Animated conversation* [9]. These approaches typically map speech to gesture units using handcrafted rules. Rule-based systems allow for straightforward control over model outputs and provide good interpretability. However, the cost of manually designing these rules is prohibitive for complex applications requiring the processing of large-scale data.

2.2.2 Statistical Methods

2.2.2.1 General Principle

These methods rely on data analysis, learning patterns from datasets, and using probabilistic models or mathematical functions for prediction. The approach involves optimizing model parameters to fit the data.

2.2.2.2 Methods

Like rule-based methods, data-driven methods also map speech features to corresponding gestures. However, instead of manual rules, they employ automatic learning based

on statistical data analysis.

Representative statistical approaches include *Gesture controllers* [30], *Statistics-based* [54], which use probabilistic distributions to find similarities between speech and gesture features. *Gesture modeling* [36] constructs probabilistic models to learn individual speaker styles.

2.2.3 Deep Learning Methods

General Principle of Deep Learning Methods

These methods utilize multi-layer perceptrons (MLPs) to automatically extract features from raw data and learn complex data representations through parameter optimization.

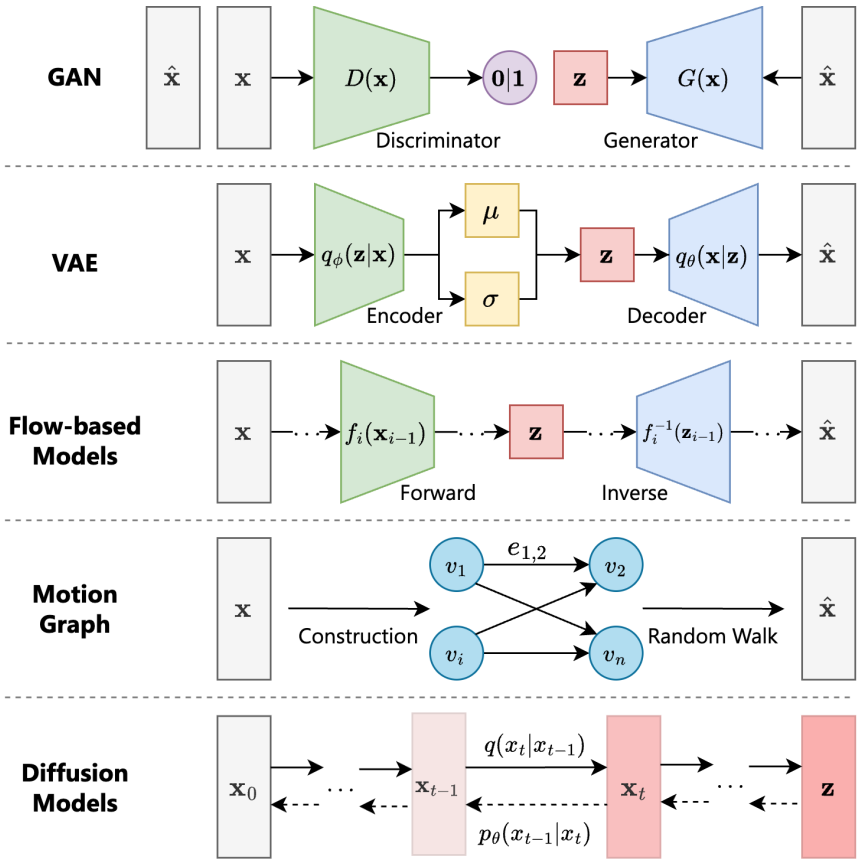


Figure 2.1: Overview of different generative models.

As illustrated in Figure 2.1, deep learning-based gesture generation methods can be divided into two main groups: likelihood-based models and implicit generative models [42].

2.2.3.1 Likelihood-Based Models

General Principle of Likelihood-Based Models

These models work by maximizing the likelihood of observed data given model parameters θ. The objective is to find the optimal parameters θ' by modeling the probability $p(\mathbf{x})$ of the data, where $\mathbf{x}$ represents the gesture sequence.

Methods

The application of deep learning to gesture generation has evolved alongside the development of deep learning models, including RNNs, LSTMs, and Transformers. Representative likelihood-based methods include:

- *Gesticulator* [28], which uses a Multilayer Perceptron (MLP) to encode text and audio features, with BERT-derived vectors used as text features.
- *HA2G* [32] builds a hierarchical Transformer-based model to learn multi-level correlations between speech and gestures, from local to global features.
- *Gesture Generation from Trimodal Context* [55] uses an RNN architecture and treats gesture generation as a translation task in natural language processing.
- *DNN* [12] combines LSTM and GRU to build a classifier neural network that selects appropriate gesture units based on speech input.
- *Cascaded Motion Network (CaMN)* [31] introduces the BEAT dataset and a waterfall-like model. Speaker information, emotion labels, text, speech, and gesture features are processed through layers to extract latent vectors. In the fusion stage, CaMN combines features sequentially: speaker and emotion features are merged first, followed by integration with latent vectors of text, speech, and gestures.
- **Motion Graph**: In *Gesturemaster* [58], a semantic graph is constructed where words in a sentence are connected based on semantic relationships. The model then selects the most relevant nodes and edges in the graph to represent gestures.

2.2.3.2 Implicit Generative Models

Principle

Implicit generative models learn the data distribution without explicitly modeling the probability density function $p(\mathbf{x})$. Instead of directly computing $p(\mathbf{x})$, the model learns a mapping $G_\theta : \mathcal{Z} \to \mathcal{X}$ by matching the distributions between real and generated data $\mathbf{x} = G_\theta(\mathbf{z}), \quad \mathbf{z} \sim p_z(\mathbf{z})$. Here, $\mathcal{Z}$ denotes the input noise space, typically with a simple distribution p_z (e.g., Gaussian, Uniform), and $\mathcal{X}$ is the space of real data, which in this case is gesture sequences $\mathbf{x}$.

In gesture generation, to incorporate conditions such as speech or text labels, implicit generative models often introduce a condition $\mathbf{c}$ representing the context of the task, resulting in the conditional generation: $\mathbf{x} = G_\theta(\mathbf{z}, \mathbf{c})$. This context may include speech, text, speaker ID, style, emotion, or initial gestures.

A typical example is the generative adversarial network (GAN) and Diffusion model, where data is synthesized by transforming an initial simple distribution (e.g., Gaussian) into the target data distribution.

Methods

Representative implicit generative models include:

- *MoGlow* [19] uses Normalizing Flows to maintain motion consistency and compatibility, while providing control via input parameters. This allows users to generate new motions or modify existing ones by adjusting control parameters.

- **GAN**: *GRU-based WGAN* [48] utilizes Wasserstein GAN to evaluate and improve the quality of gesture synthesis. The model focuses on optimizing the Wasserstein loss, mitigating mode collapse typically seen in traditional GANs. GRUs process speech data and convert it into usable gesture features, which are then fed into the WGAN for evaluation and refinement.

- **VAE**: *FreeMo* [49] employs a VAE to decompose gestures into pose modes and rhythmic motions. Gesture poses are randomly sampled using conditional sampling in the VAE's latent space, while rhythmic motions are generated from...

- **VQ-VAE**: *Rhythmic Gesticulator* [3] preprocesses speech segments based on beats, dividing speech into smaller parts and representing them as blocks with normalized dimensions. Similar to the VQ-VAE approach, normalized gesture sequences are quantized into discrete gesture lexicons. The model learns the gesture vocabulary conditioned on the previous gesture, gesture style, and speech. It then reconstructs the gesture sequence via denormalization. Unlike generative

models like GANs or Diffusion, VQ-VAE focuses more on compression rather than direct generation.

- **Diffusion**: Diffusion models focus on generating new data from noisy inputs by progressively denoising toward the original data. Diffusion-based approaches will be presented in section Section 2.4.

2.2.4 Comparison of Gesture Generation Methods

Representative Method	**Advantages**	**Disadvantages**	**Method Type**
Robot behavior toolkit [22]	- Easy to understand and implement. - Highly interpretable and controllable. - Effective for simple cases or small datasets.	- Does not generalize well to complex data. - Labor-intensive rule construction.	Rule-based model
MLP [28], RNN [6], [32], [18], [55], CNN [17], Transformer [7]	- Able to estimate data probability density. - Scalable and learns well from large datasets.	- Sensitive to noise. - Performs poorly on low-frequency data. - Low diversity in generation.	Likelihood-based model
DiffusionStyle-Gesture [50], MDM [44], Motiondiffuse [57]	- Generates high-quality data. - Flexible and diverse outputs. - Covers low-density regions of data space.	- Requires complex configuration for optimal performance. - Random noise leads to different outputs each time. - Slow sampling process.	Implicit generative model

Table 2.1: Comparison of advantages and disadvantages of different methods

2.3 Common Stages in Deep Learning Approaches for Gesture Generation

As presented in Section 1.3, gestures consist of sequences of 3D point coordinates. For each dataset, the number of bones per frame may vary.

Deep learning approaches to gesture generation are implemented using various techniques. However, this thesis generalizes the process into the following main stages, illustrated in Figure 2.2:

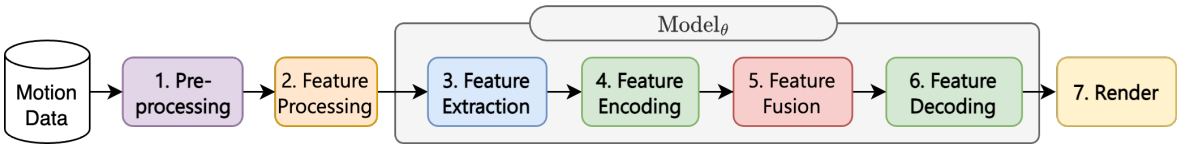


Figure 2.2: Common stages in gesture generation models.

1. **Preprocessing**: In the preprocessing stage, data such as speech segments, gesture sequences, and text are read and digitized into vectors or matrices that represent raw data information. Depending on the specific learning method, the selected initial data features may vary.

2. **Feature Processing**: In this stage, raw data such as speech and text are embedded into feature vectors. Different methods use different embedding models. The way gesture sequences are represented as feature vectors also varies across methods.

3. **Feature Extraction**: This stage uses linear transformation layers or CNN layers to extract features from the data. Text and speech features, after being processed, may be further passed through feature extraction layers to generate representation vectors corresponding to the input modalities.

4. **Feature Encoding**: In this stage, gesture, emotion, and speech vectors are encoded into a lower-dimensional latent space to facilitate learning the correlations among modalities in the feature fusion stage.

5. **Feature Fusion**: In this stage, features from speech, text, gestures, and other information are combined, typically using concatenation, fully connected layers, or operations such as vector addition or subtraction in the latent space.

6. **Feature Decoding**: In this stage, latent vectors are decoded or upsampled back to their original dimensionality.

7. **Rendering**: Once the output vectors are restored to their original size, they are converted back into BVH files and rendered using software such as Blender or Unity to visualize character motion.

2.4 Diffusion-based Model

2.4.1 General Principle of Diffusion-based Methods

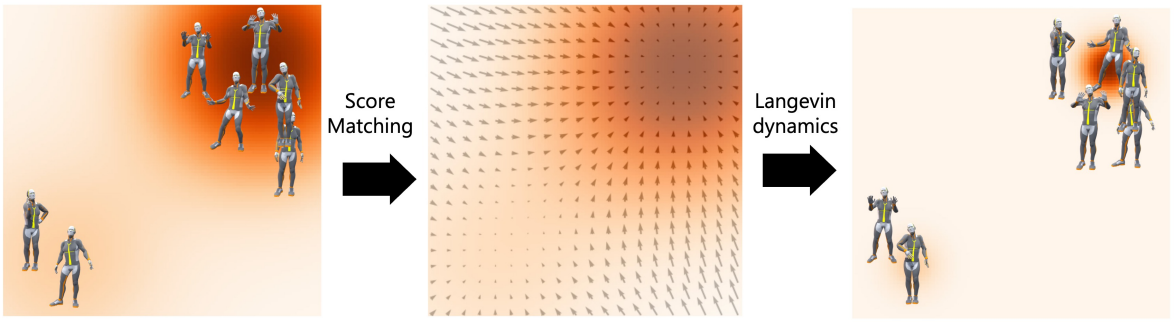


Figure 2.3: Illustration of the general principle of the Diffusion model.

Similar to the methods in the class of Implicit Generative Models (Subsubsection 2.2.3.2), the core principle of Diffusion models is to learn a function f_θ that represents the drift process from a standard normal distribution $\mathcal{N}(0, \mathbf{I})$ to the data distribution $\mathbf{x}_0 \sim p(\mathbf{x})$. Sampling is performed step by step from this reverse process to generate new data, as illustrated in Figure 2.3.

This process is executed over T steps, with noise added at each step gradually reduced so that the model can navigate toward high-density regions of the data distribution.

2.4.2 Typical Methods

2.4.2.1 Typical Methods

- *MotionDiffuse* [57] employs a conditional Diffusion model, with conditions based solely on text, excluding audio. Additionally, the model predicts noise rather than directly predicting the original gesture sequence. MotionDiffuse utilizes Self-Attention and Cross-Attention layers to model the correlation between textual features and gesture features during *Stage 5. Feature Fusion* (Figure 2.2).

- *Flame* [24] applies a Diffusion model with a Transformer-based architecture. In *Stage 2. Feature Processing* (Figure 2.2), it uses the pre-trained RoBERTa model

to embed the text into textual feature vectors, which serve as the conditioning input. During *Stage 5. Feature Fusion* (Figure 2.2), the text is used as the `CLS` token prepended to the gesture sequence before passing through the Transformer Decoder. Similar to other methods, the model predicts the added noise rather than the original gesture sequence.

- *DiffWave* [27] is a noise-predicting Diffusion model in which the time steps pass through multiple Fully Connected layers and a Swish activation function before feature fusion. It uses a dilated convolutional architecture inherited from WaveNet. DiffWave enables better representation of speech, improving the effectiveness of conditioning for the Diffusion model.

- *Listen, Denoise, Action* [2] builds upon DiffWave [27], replacing the dilated convolution layers with a Transformer, and integrating Conformer modules to enhance model performance.

- *DiffSHEG* [10] employs a Diffusion model; in *Stage 2. Feature Processing*, it uses HuBERT to encode the audio signal. The model treats facial expressions as a signal for gesture generation and achieves real-time fusion of both facial expressions and gestures.

- *GestureDiffuCLIP* [4] uses a Diffusion model conditioned on text, leveraging Contrastive Learning with CLIP to integrate text features and control gesture styles. Similar to other prompt-based approaches such as StableDiffusion or Midjourney, it treats text as prompts to learn gestures from descriptive sentences.

- *Freetalker* [52] trains a Diffusion model on multiple datasets to generate speaker-specific gestures conditioned on speech and text. Unlike Transformer-based methods, Freetalker employs an Attention-based Network to model the correlation between textual, auditory, and gesture features during *Stage 5. Feature Fusion* (Figure 2.2).

2.4.2.2 Inherited Method in This Thesis

- *MDM* [44] applies conditional Diffusion to gesture generation, using CLIP (Contrastive Language–Image Pre-training) embeddings of descriptive text as conditions. MDM adopts a Transformer-based architecture to reconstruct original

gesture data. Like other text-based Diffusion approaches, in *Stage 3. Feature Extraction* (Figure 2.2), the input text is randomly masked to hide certain segments, enabling the model to learn the importance of each part for various gestures.

In *Stage 5. Feature Fusion* (Figure 2.2), the text is prepended as a `CLS` token to the gesture sequence before passing through the Transformer Encoder, where self-attention models the relationship between the text and each gesture frame. MDM predicts the original gesture data rather than noise.

- **DiffuseStyleGesture** [50] extends *MDM* [44] by incorporating audio, initial gestures, and style as conditioning inputs. In *Stage 1. Preprocessing* (Figure 2.2), the model processes coordinate vectors to obtain feature vectors of dimension $D = 1141$ per frame. In *Stage 2. Feature Processing* (Figure 2.2), DiffuseStyleGesture uses WavLM for audio embedding. In *Stage 5. Feature Fusion* (Figure 2.2), it improves upon MDM by applying Cross-Local Attention prior to the Transformer Encoder.

2.4.3 Suitability of Diffusion Models for Gesture Generation

Given that gesture data consist of joint coordinates and rotational angles, they require high detail to ensure realistic character motion. However, data in extreme parameter cases are sparse. Thanks to their ability to capture fine-grained details and generalize to rare cases, Diffusion models are well-suited to address these challenges, as discussed in Section 1.5. A comparative evaluation of Diffusion models and other approaches is shown in Table 2.1.

This thesis selects Diffusion as the core model for the gesture generation task. Specifically, it adopts the **DiffuseStyleGesture** [50] model and integrates text as a semantic feature during *Stage 5. Feature Fusion* (Figure 2.2) in the gesture generation pipeline to propose the novel model **OHGesture**.

Chapter 3. METHOD PROPOSAL

The nature of neural network-based methods is to estimate the probability density of data, which requires normalization. In contrast, Diffusion models learn to estimate the gradients of data distribution [42] and do not require normalization over the entire dataset, resulting in better performance than neural network methods that do not use diffusion. Diffusion models can approximate data distributions even in low-density regions and generate highly detailed results.

This thesis builds upon the DiffuseStyleGesture model [53], with the main improvement being the integration of speech. Speech is transcribed into text, and the text is embedded to obtain textual semantic feature vectors, which are then used as conditioning vectors in the conditional diffusion process, as described in Subsection 3.2.2. In addition, in *Stage 7. Rendering* (Figure 2.2), Unity is used to visualize the generated gestures.

The main contributions of the thesis are presented in Section 6.4.

The thesis first describes the diffusion model in Section 3.1, followed by the proposed OHGesture model in Section 3.2.

3.1 Basic Diffusion Model and Improvements

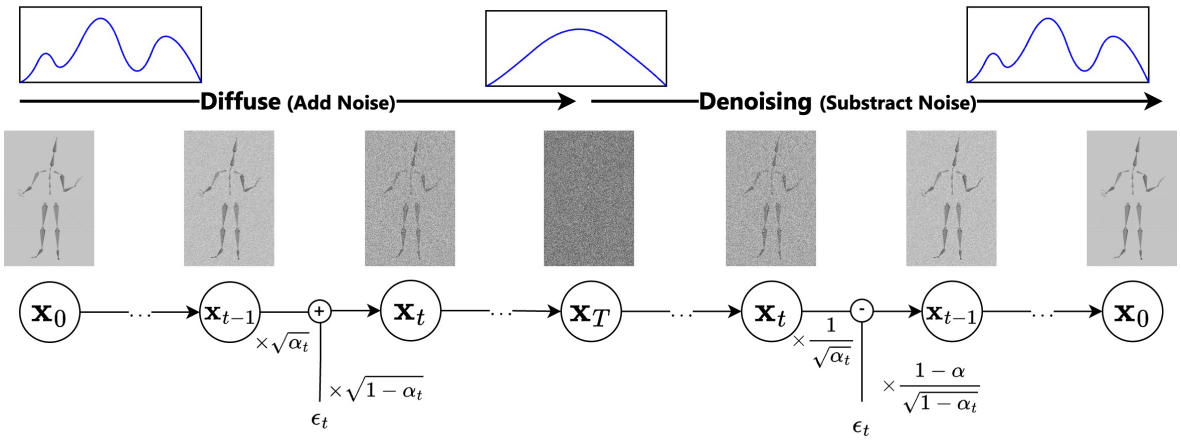


Figure 3.1: The **non-training** Diffusion and Denoising Process

In neural network methods like ResNet or InceptionNet, the goal is to learn the weight θ of a function $f_\theta(x)$ by minimizing a loss function $\mathcal{L}_{\text{loss}}$ between a label y and its prediction $\hat{y}$. Once training is complete, the learned weight θ' can be used to predict a new sample x' through $f_{\theta'}$ to obtain prediction y'.

Similarly to VAE, where the input matrix is encoded into a latent vector z and then decoded back into a matrix of the original size, the diffusion model breaks the learning process into T steps. At each step t, the forward diffusion process $q(\mathbf{x})$ from $1 \to T$ adds Gaussian noise $\epsilon_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, where $\mathbf{I}$ is the identity matrix of the standard Gaussian distribution with $\mathbb{E}[\epsilon] = 0$ and $\text{Var}(\epsilon) = 1$.

The process of adding noise from left to right is called $q(\mathbf{x})$, while the denoising process from right to left is $p(\mathbf{x})$. Note that ϵ_t is randomly sampled at each step t, but once sampled, it is fixed. As illustrated in Figure 3.1, if we add a noise ϵ_t during diffusion and subtract exactly the same noise during denoising, the result $\mathbf{x}_0$ is **identical** to the original input.

However, during denoising, instead of subtracting the actual noise ϵ_t, the model uses a function f_θ to **predict the added noise** during diffusion. Then, $\mathbf{x}_T$ is iteratively denoised by subtracting the predicted noise to obtain $\hat{\mathbf{x}}_{T-1}$, continuing until $\hat{\mathbf{x}}_0$ is reached.

In the basic Diffusion model or Denoising Diffusion Probabilistic Models (DDPM [20]), the denoising process goes from $T \to 1$, and the goal is to learn the weights θ

of the noise prediction function f_θ (also denoted as ϵ_θ). After training, the learned weights θ' are used to predict the noise $\hat{\epsilon}$. Then, the noise is subtracted from the noisy image $\mathbf{x}_t$ to get $\mathbf{x}_{t-1}$, with additional noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ added to maintain gesture diversity. This process is repeated until reaching the final prediction $\hat{\mathbf{x}}_0$.

3.1.1 Forward Diffusion Process

Given the input data $\mathbf{x}_0 \sim q(x)$ from the real dataset, at each step t, noise is added to $\mathbf{x}_0$ with the noise-to-signal ratio controlled by a factor β:

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\boldsymbol{\epsilon}_{t-1} \tag{3.1}$$

The forward process runs from $1 \to T$, with $\beta_t \in (0, 1)$ for each t.

Since a function of the form $f(x) = ax + b\epsilon$, with $\epsilon \sim \mathcal{N}(0, \mathbf{I})$, implies $f(x) \sim \mathcal{N}(ax, b^2)$, the forward process can be reformulated as:

$$\begin{aligned} q(\mathbf{x}_t|\mathbf{x}_{t-1}) &= \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \\ q(\mathbf{x}_{1:T}|\mathbf{x}_0) &= \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \end{aligned} \tag{3.2}$$

Here, $\sqrt{1-\beta_t}$ gradually decreases the contribution of $\mathbf{x}_t$, while β_t increases the noise component. Typically, $\beta_1 < \beta_2 < \cdots < \beta_T$. As $T \to \infty$, $\mathbf{x}_T$ approaches a pure Gaussian distribution: $q(\mathbf{x}_T) = \mathcal{N}(0, \mathbf{I})$ [47].

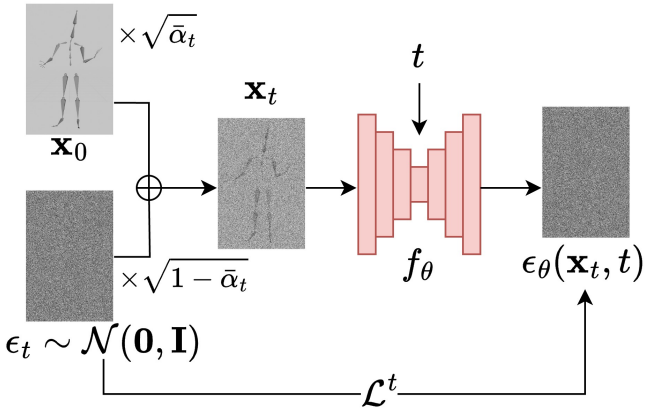


Figure 3.2: Noise addition during training

Since $\boldsymbol{\epsilon}_{t-1}, \boldsymbol{\epsilon}_{t-2}, \cdots \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and fixed for each t, $\mathbf{x}_t$ can be computed directly from $\mathbf{x}_0$. Define $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$, then Equation 3.1 becomes:

$$
\begin{aligned}
\boldsymbol{x}_t &= \sqrt{\alpha_t}\boldsymbol{x}_{t-1} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t}\left(\sqrt{\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2}\right) + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1})}\boldsymbol{\epsilon}_{t-2} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_{t-1} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{\alpha_t(1-\alpha_{t-1}) + (1-\alpha_t)}\boldsymbol{\epsilon}_{t-2} \\
&= \sqrt{\alpha_t\alpha_{t-1}}\boldsymbol{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\boldsymbol{\epsilon}_{t-2} \\
&= \dots \\
&= \sqrt{\prod_{i=1}^{t}\alpha_i}\boldsymbol{x}_0 + \sqrt{1-\prod_{i=1}^{t}\alpha_i}\boldsymbol{\epsilon}_0 \\
&= \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_0 \\
&\sim \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)\,\mathbf{I}\right)
\end{aligned}
\tag{3.3}
$$

The evolution of $\sqrt{\alpha}$ and $\sqrt{1-\alpha}$ over diffusion steps is shown in Appendix Appendix A.

3.1.2 Denoising Process

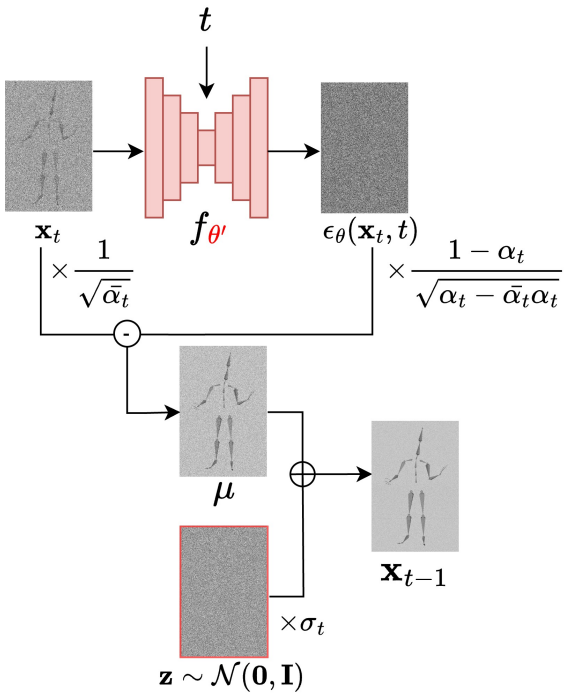


Figure 3.3: Denoising during inference

The denoising process $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ starts from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. A neural network $f_\theta(\mathbf{x}_t, t)$ is used to predict the noise $\hat{\epsilon}_t = f_\theta(\mathbf{x}_t, t)$.

The denoising distribution has mean and variance:

$$
\begin{aligned}
p_\theta(\mathbf{x}_{0:T}) &= p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \\
p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) &= \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))
\end{aligned}
\tag{3.4}
$$

where $\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} f_\theta(\mathbf{x}_t, t) \right)$.

3.1.3 Training Process in the Basic Diffusion Model

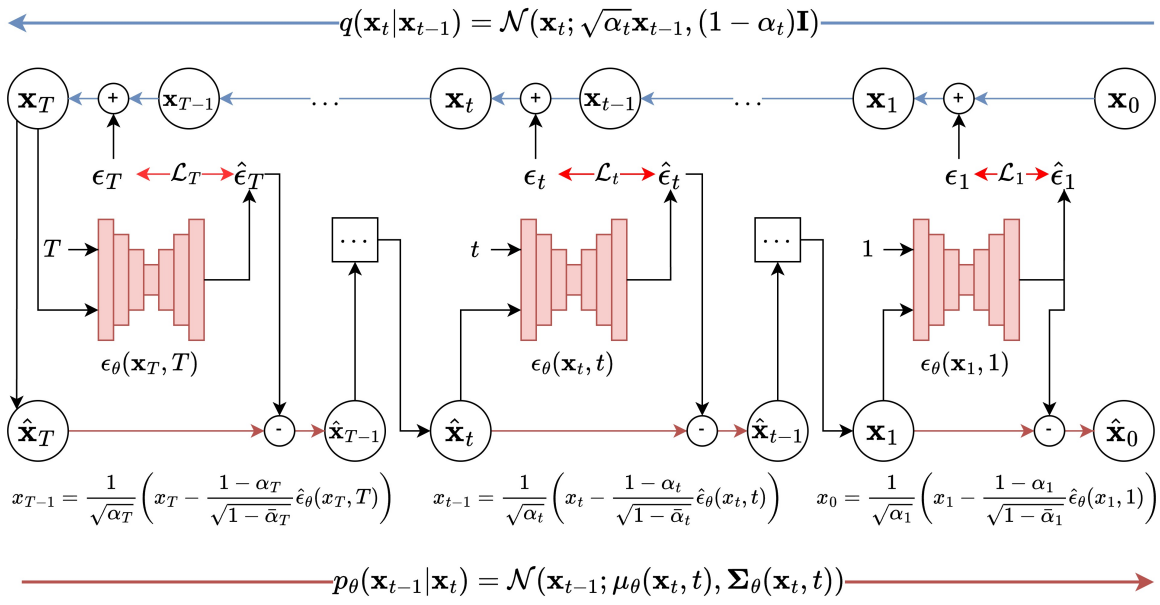


Figure 3.4: Basic diffusion model

The diffusion model learns the parameters θ of the noise prediction function $f_\theta(\mathbf{x}_t, t)$ (or ϵ_θ). During denoising, we minimize the loss between the predicted noise $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ and actual noise $\boldsymbol{\epsilon}_t$ at each step t:

$$
\begin{aligned}
\mathcal{L}^t &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t} \left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2 \right] \\
&= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t} \left[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2 \right]
\end{aligned}
\tag{3.5}
$$

The total loss is $\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}^t$.

Here, $f_\theta(x_t, t)$ or ϵ_θ is a U-Net model used to encode and decode the data for noise prediction. The computation process is illustrated in Figure 3.4.

Algorithm 1 DDPM Training Algorithm

1. Precompute $\sqrt{\alpha_t}$, $\sqrt{1-\alpha_t}$, and $\sqrt{\bar{\alpha}_t}$ for $t = 1 \to T$. Define noise schedule $\{\alpha_t \in (0,1)\}_{t=1}^{T}$ with $\alpha_1 < \alpha_2 < \cdots < \alpha_T$.
2. Sample label $\mathbf{x}_0$ from the normalized data distribution.
3. Generate random noise $\boldsymbol{\epsilon}_t$ for each step $t = 1 \to T$, where $\boldsymbol{\epsilon}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$.
4. Apply forward diffusion to get $\mathbf{x}_t$:
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t$$
5. Randomly select $t \in [1, T]$.
6. Feed $\mathbf{x}_t$ and t into the model to predict noise: $\hat{\boldsymbol{\epsilon}} = \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$.
7. Compute the gradient:
$$\nabla_{\theta_t} \left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right\|^2$$
And loss:
$$\mathcal{L}^t = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}_t}\left[\left\| \boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t) \right\|^2 \right]$$
8. Repeat step 6 until convergence to obtain optimal weights θ'.

3.1.4 Basic Sampling Process in Diffusion Models

After obtaining the weights θ', the denoising function is used to denoise from noise $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The transformation from pure noise $\mathbf{x}_T$ to the prediction $\hat{\mathbf{x}_0}$ is as follows:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} f_{\theta'}(\mathbf{x}_t, t)\right) + \sqrt{1-\alpha_t}\tilde{\epsilon}_t \tag{3.6}$$

Note that ϵ_t is fixed and randomly generated noise created before the training process, and reused during the forward diffusion process Subsection 3.1.2 in formula Equation 3.1. As shown in Figure 3.5, the error ϵ_t corresponds to the noise at each step t, and the loss function $\mathcal{L}^t$ is computed per time step t.

Following the training process, the DDPM sampling algorithm begins from pure noise, i.e., $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$, where the initial data is entirely noise. The values $\sqrt{\alpha_t}$, $\sqrt{1-\alpha_t}$, and $\sqrt{\bar{\alpha}_t}$, obtained from the training phase, are used during sampling to reconstruct the original data $\mathbf{x}_0$. The next step is to compute the noise adjustment coefficient σ_t, based on the noise schedule α_t defined during training. These values influence the amount of noise added in the reverse sampling process.

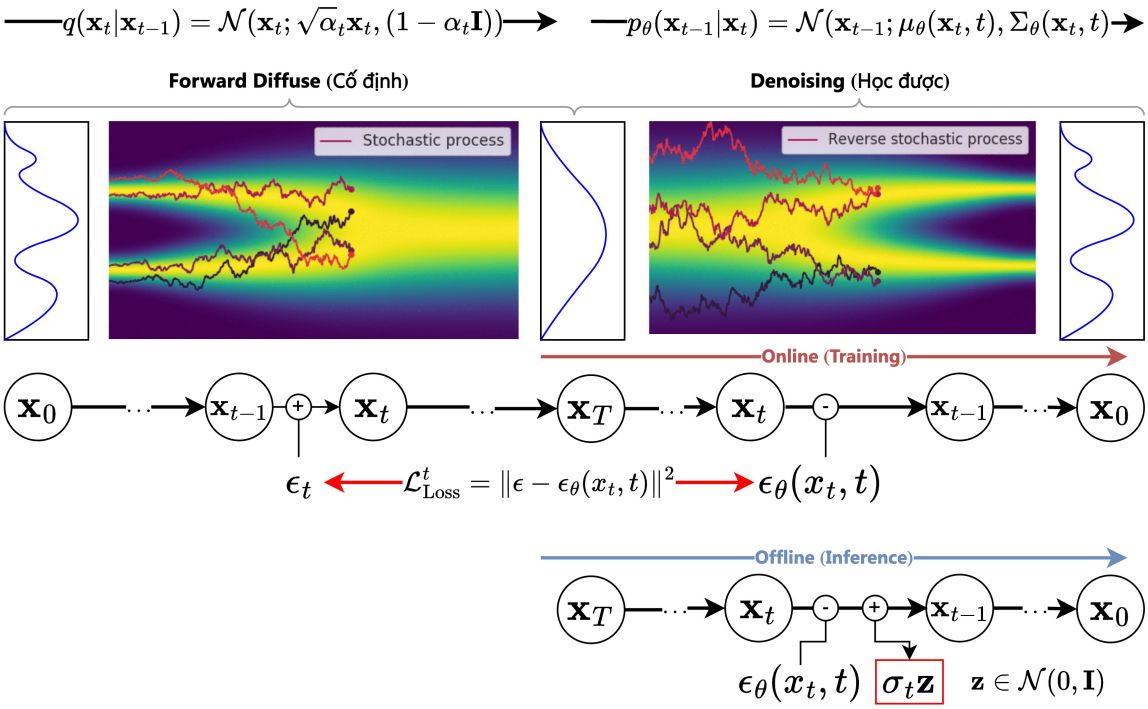


Figure 3.5: Training and Sampling process in a standard Diffusion model

The sampling process (Algorithm 2) proceeds from step T down to 1, and at each step, a random noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$ is generated and added to the predicted result. At each step t, the model predicts the noise $\boldsymbol{\epsilon}_{\theta'}$ based on the noisy data $\mathbf{x}_t$ and time step t, then uses this prediction to compute the value μ, an estimate of $\mathbf{x}_0$. Finally, a noise term $\sigma_t\mathbf{z}$ is added to μ to obtain $\hat{\mathbf{x}}_{t-1}$, the noisy data at step $t-1$. This process continues until $t = 1$, at which point $\hat{\mathbf{x}}_0$ — the final prediction of the original data — is obtained through denoising.

The most important aspect of the denoising sampling process is to add a noise term $\mathbf{z} \in \mathcal{N}(0, \mathbf{I})$, with $\mathbf{z}$ sequentially added at each step t using the scaling factor σ_t. The goal of noise ϵ is to shape the marginal noise distribution so that model f_θ (or ϵ_θ) can learn to denoise, whereas $\mathbf{z}$ is used to increase diversity in generation and improve stability during sampling. The decay of σ_t is detailed in Appendix Section A.3.

3.1.5 Improved Diffusion Model with $\mathbf{x}_0$ Prediction Instead of ϵ_t

Based on Equation 3.3, it is clear that given $\mathbf{x}_t$, we can infer $\mathbf{x}_0$. Conversely, given $\mathbf{x}_0$, we can explicitly infer $\mathbf{x}_t$ for any t by adding the noise ϵ_t from the forward diffusion

Algorithm 2 Sampling algorithm in DDPM

1. Start with noise: $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Retrieve values $\sqrt{\alpha_t}$, $\sqrt{1-\alpha_t}$, and $\sqrt{\bar{\alpha}_t}$ from the training process.
3. Compute the noise adjustment coefficient σ_t from α_t at each step $t : 1 \to T$:
$$\sigma_t = \sqrt{\frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}(1-\alpha_t)}$$
4. For each t, iterate **sequentially** from $[T, \ldots, 1]$.
5. Generate random noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$.
6. Feed $\mathbf{x}_t$ into the model to infer noise: $\boldsymbol{\epsilon}_{\theta'} = \boldsymbol{\epsilon}_{\theta'}(\mathbf{x}_t, t)$.
7. Use the predicted noise to subtract from $\mathbf{x}_t$ at step t:
$$\mu = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_{\theta'}(\mathbf{x}_t, t)\right)$$
8. Add noise: $\hat{\mathbf{x}}_{t-1} = \mu + \sigma_t \mathbf{z}$.
9. When $t = 1$, obtain $\hat{\mathbf{x}}_0$ from the denoising process.

process.

From this observation, the authors in [37] proposed an improvement to DDPM: instead of using the neural network $f_\theta(\mathbf{x}_t, t)$ to predict ϵ_t as in Figure 3.4, the function $f_\theta(\mathbf{x}_t, t)$ is repurposed to directly predict $\mathbf{x}_0$, after which noise is added using Equation 3.3.

As shown in Figure 3.6, we start with forward diffusion from $t : 1 \to T$ to obtain $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$. After acquiring $\mathbf{x}_T$, we feed it into f_θ to predict $\hat{\mathbf{x}}_0$, and from $\hat{\mathbf{x}}_0$ we add noise to obtain $\hat{\mathbf{x}}_{t-1}$. This continues until $t = 1$, when $\hat{\mathbf{x}}_0$ is ultimately obtained. We can compare the two methods as follows:

- ϵ **objective**: the model predicts the noise. Begin with forward noise injection to get $\mathbf{x}_T$, then use $\mathbf{x}_T \in \mathcal{N}(0, \mathbf{I})$ as input for denoising. During denoising, the model predicts the noise $\hat{\epsilon}_t$ added in the forward step, i.e., ϵ_t, and minimizes the error between the predicted and actual forward noise.

- $\mathbf{x}_0$ **objective**: similarly, the model applies forward diffusion to get $\mathbf{x}_T$, then uses $\mathbf{x}_T \in \mathcal{N}(0, \mathbf{I})$ for denoising. The model directly predicts $\mathbf{x}_0$, then reintroduces

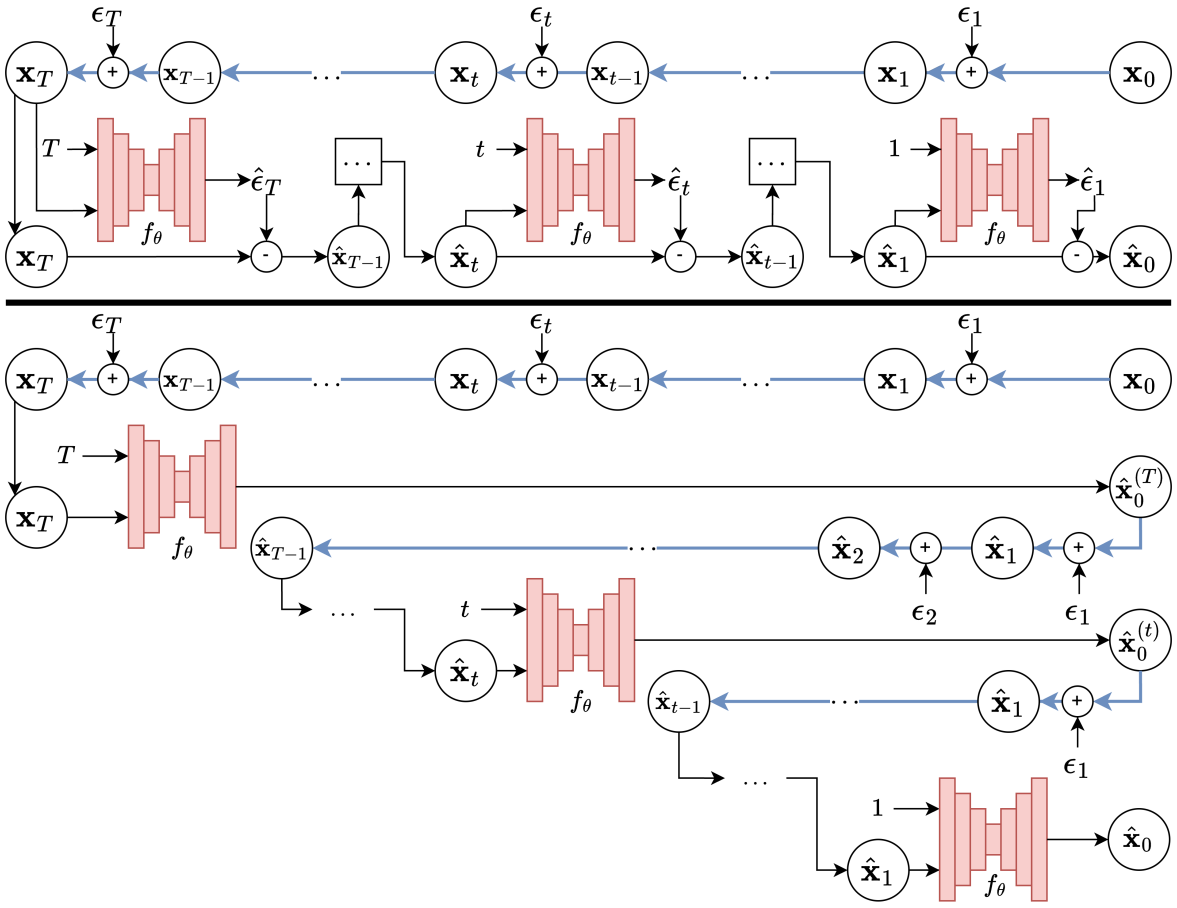


Figure 3.6: Comparison between ϵ objective (top) and $\mathbf{x}_0$ objective (bottom)

noise to compute $\mathbf{x}_{t-1}$, and continues using $\mathbf{x}_{t-1}$ as input to predict $\mathbf{x}_0$ again.

3.1.6 Conditional Diffusion Models

To control the generation process under different conditions, generation must be conditioned on c—that is, we aim to model $p(\mathbf{x}|c)$ given condition c. The authors in [14] proposed using a separate function f_ϕ to train the conditional component. However, this approach presents difficulties when the conditions change, and combining or updating weights in a separate model is challenging for scalability.

As shown in Figure 3.5, to allow inference to generate different $\mathbf{x}_0$ samples controllably based on condition c, a guidance term must be added at each step t. The authors proposed Classifier-Free Diffusion Guidance [21], in which the result $\mathbf{x}_0$ is updated by combining the conditional and unconditional outputs:

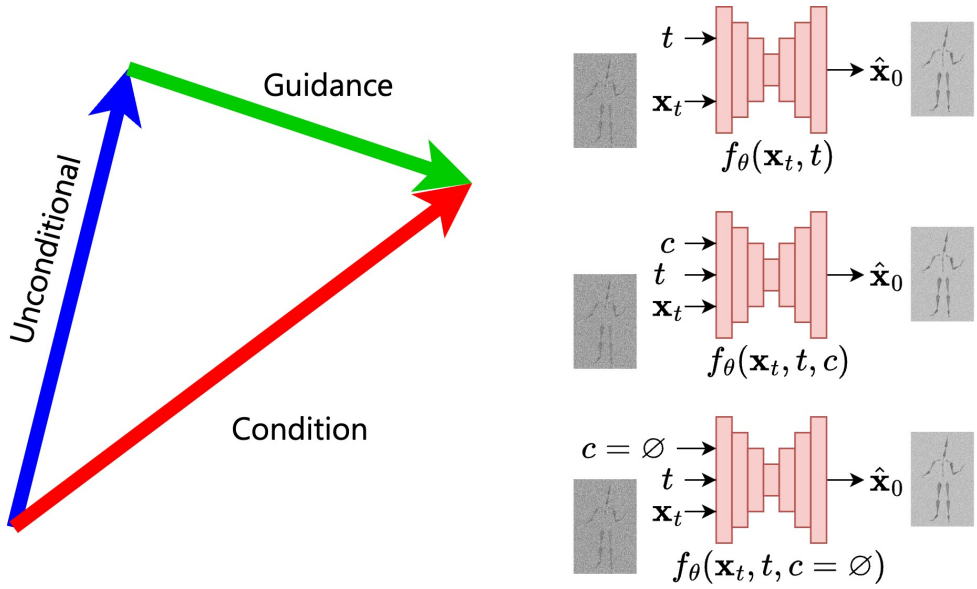


Figure 3.7: Conditional Diffusion via Guidance Vector

$$\hat{\mathbf{x}}_{0\gamma,c,c_\varnothing} = \gamma \cdot f_\theta\left(\mathbf{x}_t, t, c\right) + (1-\gamma) \cdot f_\theta\left(\mathbf{x}_t, t, c_\varnothing\right) \tag{3.7}$$

Here, c is the condition, $c_\varnothing$ is the null (empty) condition, and the conditional generation is controlled by the parameter γ: the larger γ is, the closer the result aligns with condition c; conversely, a smaller γ leads to a result closer to the unconditional output.

As illustrated in Figure 3.7, the top function f_θ is unconditional, the middle one is conditional diffusion, and the bottom one is diffusion under an empty condition.

3.2 Proposed OHGesture Model

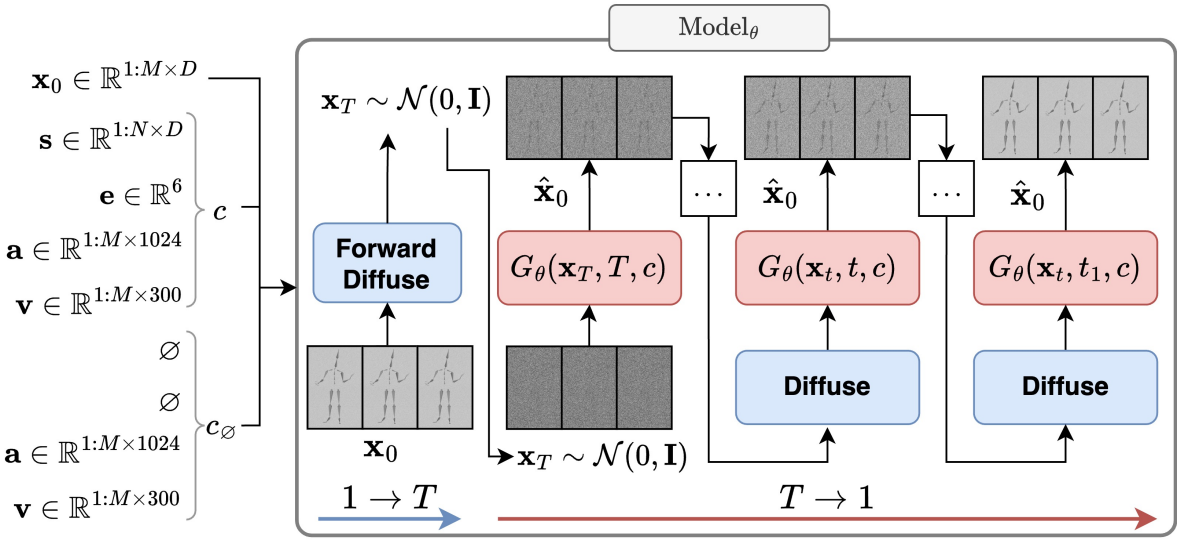


Figure 3.8: Overview of the OHGesture model

The proposed model **OHGesture** (**O**pen **H**uman **Gesture** Generation) in this thesis is based on the **DiffuseStyleGesture** model [50], which applies the Diffusion model [20] with conditional guidance [21] (Classifier-Free Diffusion Guidance) to control features during the denoising process.

The similarities and differences of applying the diffusion model to the gesture generation task compared to image generation are as follows:

Similarities

- Uses the Diffusion model (Section 3.1) on gesture data $\mathbf{x}^{1:M\times D}$, with M temporal frames and $D = 1141$ representing motion coordinates per frame (analogous to image width and height).
- Uses conditional Diffusion (Subsection 3.1.6) with $\mathbf{x}_0$ objective (Subsection 3.1.5).
- In stages *4. Feature Encoding* and *6. Feature Decoding* in Figure 2.2, the model uses a latent vector of dimension 256.

Differences

- Conditional gesture generation:
  - Emotional condition: $c = [\mathbf{s}, \mathbf{e}, \mathbf{a}, \mathbf{v}]$ and $c_\varnothing = [\varnothing, \varnothing, \mathbf{a}, \mathbf{v}]$.

  - Emotion state interpolation between $\mathbf{e}_1, \mathbf{e}_2$ using: $c = [\mathbf{s}, \mathbf{e}_1, \mathbf{a}, \mathbf{v}]$ and $c_\varnothing = [\mathbf{s}, \mathbf{e}_2, \mathbf{a}, \mathbf{v}]$.

- In stage *5. Feature Fusion* Figure 2.2, the model uses Self-Attention: learning the relationship between emotions, seed gestures, and each frame (similar to DALL-E 2's text-image alignment).

- In stage *5. Feature Fusion* Figure 2.2, the model concatenates speech and text (analogous to ControlNet's pixel-wise condition).

Here, $\mathbf{x}_0$ is a sequence of M gesture frames $\mathbf{x} \in \mathbb{R}^{1:M \times D}$ ($D = 1141$), with condition $c = [\mathbf{s}, \mathbf{e}, \mathbf{a}, \mathbf{v}]$ including seed gesture $\mathbf{s}$, emotion $\mathbf{e}$, speech $\mathbf{a}$ corresponding to the gesture, and text $\mathbf{v}$.

The model's objective is to learn parameters θ of the generative function G_θ with inputs being the noisy gesture matrix $\mathbf{x}_t \in \mathbb{R}^{1:M \times D}$, timestep t, and condition c. An overview of the proposed **OHGesture** model is illustrated in Figure 3.8. As with standard diffusion models, it includes two processes: the diffusion process q and the denoising process p_θ with weights θ. The *1. Preprocessing* stage will be presented in Section 4.2.

3.2.1 Feature Processing Stage

In the *Stage 2: Feature Processing* step (Figure 2.2), the goal is to convert raw input data into matrices or vectors suitable for input into the model.

- **Text** $\mathbf{v} \in \mathbb{R}^{1:M \times 300}$: As discussed in Subsection 2.4.2, methods such as *MDM* [44] and *DiffuseStyleGesture+* [53] use gesture-descriptive prompts, similar to those used in Midjourney, as input for the model. However, such prompts are mainly used to cluster gestures rather than align them semantically.

  In contrast, this thesis treats text as a semantic feature that aligns specific segments of text with corresponding gesture segments for digital human generation. Therefore, the contribution in this stage is to use speech data that has already been preprocessed (see Section 4.2) to obtain transcribed text. The FastText model [8] is used to embed this text into vectors, which are then temporally aligned with the number of gesture frames, resulting in a text matrix $\mathbf{v} \in \mathbb{R}^{1:M \times 300}$. For segments without text, zero vectors are used; for segments with vocabulary, the corresponding embedding is assigned for each gesture frame.

- **Speech $\mathbf{a} \in \mathbb{R}^{1:M\times1024}$**: All speech data in '.wav' format is downsampled to 16 kHz. The speech corresponding to the gesture segment (4 seconds) is extracted as a waveform vector $\mathbf{a} \in \mathbb{R}^{64000}$. Following DiffuseStyleGesture, this thesis uses the pre-trained WavLM Large model [11] to embed the raw waveform into a high-dimensional vector representing acoustic features. Linear interpolation is then applied to temporally align the latent features from WavLM at 20 fps, resulting in the speech matrix $\mathbf{a} \in \mathbb{R}^{1:M\times1024}$.

- **Emotion $\mathbf{e} \in \mathbb{R}^6$**: Emotion is represented by one of six classes: `Happy`, `Sad`, `Neutral`, `Old`, `Relaxed`, and `Angry`. Each label is encoded using one-hot encoding to form the vector $\mathbf{e} \in \mathbb{R}^6$.

- **Seed Gesture $\mathbf{s} \in \mathbb{R}^{1:N\times D}$**: This is the initial gesture sequence composed of $N = 8$ frames, with each frame containing joint data for 75 joints. These are processed according to the formula in Equation 4.1 to yield a $D = 1141$-dimensional vector.

- **Ground Truth Gesture $\mathbf{x}_0 \in \mathbb{R}^{1:M\times D}$**: This is the ground-truth gesture sequence of $M = 80$ frames (corresponding to 4 seconds at 20 fps), which is pre-processed to form the matrix $\mathbf{x}_0 \in \mathbb{R}^{1:M\times D}$.

3.2.2 Feature Extraction Stage

In the *Stage 3: Feature Extraction* step (Figure 2.2), the goal is to convert the input matrices into latent vectors that capture the semantic content of each modality. This is done by passing the feature data through linear transformation layers or Multilayer Perceptrons (MLPs).

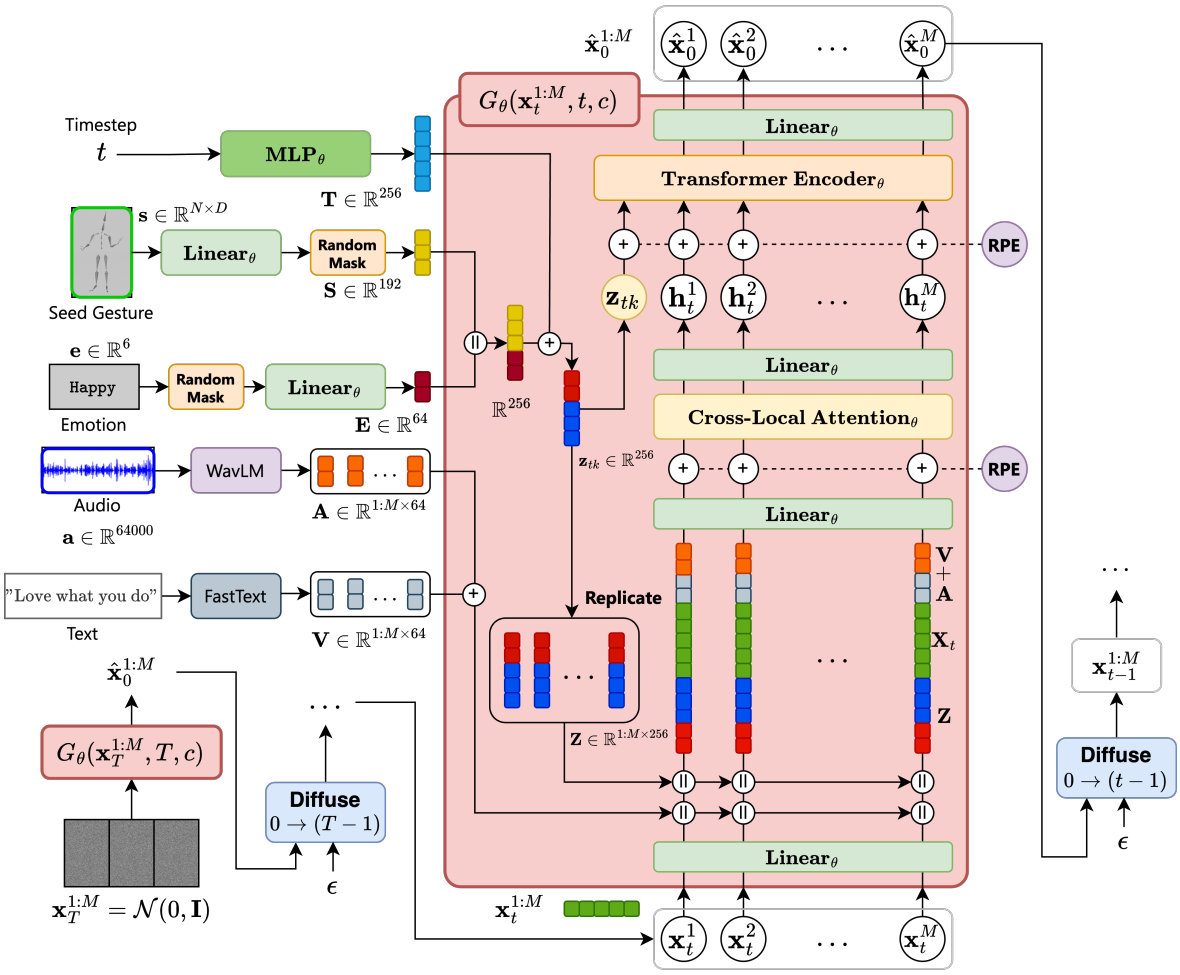


Figure 3.9: The Model Architecture in OHGesture

- **Timestep $\mathbf{T} \in \mathbb{R}^{256}$**: The timestep at each process step $t \in [0, T]$ enables the model to generalize the denoising process across different time steps. The objective is for the model to learn how the values should change with respect to t in order to accurately predict $\mathbf{x}_0$. The timestep t is initialized using sinusoidal positional encoding: $\text{PE}(t) = \left[\sin\left(\frac{t}{10000^{2i/d}}\right), \cos\left(\frac{t}{10000^{2i/d}}\right)\right]$, and then passed through a Multilayer Perceptron (MLP) to obtain the vector $\mathbf{T} \in \mathbb{R}^{256}$.

- **Speech $\mathbf{A} \in \mathbb{R}^{1:M \times 64}$**: The speech feature matrix $\mathbf{a} \in \mathbb{R}^{1:M \times 1024}$ is passed through a linear layer to reduce its dimensionality to a 64-dimensional feature vector, resulting in the matrix $\mathbf{A} \in \mathbb{R}^{1:M \times 64}$.

- **Text $\mathbf{V} \in \mathbb{R}^{1:M \times 64}$**: After preprocessing described in Section 4.2, the text is aligned to match the number of frames M, resulting in matrix $\mathbf{v} \in \mathbb{R}^{1:M \times 300}$. It is then passed through a linear transformation to reduce feature dimensionality, yielding the final matrix $\mathbf{V} \in \mathbb{R}^{1:M \times 64}$, aligned with the speech matrix.

- **Seed Gesture $\mathbf{S} \in \mathbb{R}^{192}$**: The seed gesture $\mathbf{s} \in \mathbb{R}^{1:N \times D}$ is processed through a linear transformation layer to obtain the vector $\mathbf{S} \in \mathbb{R}^{192}$. During training, $\mathbf{S}$ is also passed through a random masking layer to randomly mask segments of gesture frames in N frames. This allows the model to learn how missing frames impact the final predicted gestures.

- **Emotion $\mathbf{E} \in \mathbb{R}^{64}$**: The emotion vector $\mathbf{e} \in \mathbb{R}^{6}$ is passed through a linear transformation layer to produce the feature vector $\mathbf{E} \in \mathbb{R}^{64}$. This is designed to be concatenated with the seed gesture vector $\mathbf{S} \in \mathbb{R}^{192}$ to form a 256-dimensional latent vector.

- **Noisy Gesture $\mathbf{x}_T \in \mathbb{R}^{1:M \times D}$**: During training, $\mathbf{x}_t$ is the noisy gesture that shares the same dimensionality as the original gesture $\mathbf{x}_0$ and is sampled from a standard normal distribution $\mathcal{N}(0, \mathbf{I})$. The initial noisy gesture $\mathbf{x}_T$ is drawn from a Gaussian distribution, and the subsequent $\mathbf{x}_t$, $t < T$ are produced through iterative noising steps, as illustrated in Figure 3.8.

3.2.3 Feature Encoding Stage

In the *Stage 4: Feature Encoding* step (Figure 2.2), the objective is to reduce the dimensionality of the input data to a lower latent space, in order to alleviate computational overhead and avoid the explosion of processing complexity.

The primary data used in the diffusion process is the gesture sequence $\mathbf{x}_t \in \mathbb{R}^{1:M \times D}$. As illustrated in Figure 3.9, the gesture sequence with size $M \times D$ is passed through a linear transformation layer Linear_θ to produce the matrix $\mathbf{X} \in \mathbb{R}^{1:M \times 256}$. This dimensionality reduction of $\mathbf{x}$ is performed prior to passing the gesture sequence through the Cross-Local Attention and Transformer Encoder layers, in order to compute correlations across multiple modalities.

3.2.4 Feature Fusion Stage

In the *Stage 5: Feature Fusion* step (Figure 2.2), the goal is to compute inter-feature correlations using concatenation, addition, or attention-based mechanisms.

First, the seed gesture vector $\mathbf{S} \in \mathbb{R}^{192}$ and the emotion vector $\mathbf{E} \in \mathbb{R}^{64}$ are concatenated to form a single vector of size 256, since 256 is the chosen hidden dimensionality

for computing feature correlations. This vector is then added to the timestep vector $\mathbf{T}$ to form the final vector $\mathbf{z}_{tk} \in \mathbb{R}^{256}$.

$$\mathbf{z}_{tk} = \text{concat}(\mathbf{E} \mid\mid \mathbf{S}) + \mathbf{T} \tag{3.8}$$

The feature fusion process of $\mathbf{z}_{tk}$ is illustrated in Figure 3.11a.

3.2.4.1 Frame-wise Feature Integration

Next, $\mathbf{z}_{tk} \xrightarrow{\text{replicate}} \mathbf{Z}$ is replicated M times to match the dimensionality of M frames, resulting in the matrix $\mathbf{Z} \in \mathbb{R}^{1:M \times 256}$, as shown in Figure 3.9.

The speech feature matrix $\mathbf{A}$ and the text feature matrix $\mathbf{V}$ are then added together to produce a matrix that combines both modalities. This combined matrix is then concatenated with the gesture feature matrix $\mathbf{X}$. Finally, the result is concatenated with matrix $\mathbf{Z}$ to obtain the full feature matrix $\mathbf{M}$.

$$\mathbf{M} = \text{concat}(\mathbf{Z} \mid\mid \text{ concat}(\mathbf{X} \mid\mid (\mathbf{V} + \mathbf{A}))) \tag{3.9}$$

The matrix $\mathbf{M} \in \mathbb{R}^{1:M \times P}$, as shown in Equation 3.9, represents the frame-wise feature matrix from frame 1 to frame M, where each frame has dimensionality P, which is the sum of all concatenated feature vectors. Given $\mathbf{X} \in \mathbb{R}^{1:M \times 256}$, $\mathbf{Z} \in \mathbb{R}^{1:M \times 256}$, and $\mathbf{A}, \mathbf{V} \in \mathbb{R}^{1:M \times 64}$, the resulting dimensionality is $P = 256 + 256 + 64$.

Subsequently, the matrix $\mathbf{m}$ is passed through a linear transformation to reduce its dimensionality from P to 256, resulting in the matrix $\mathbf{m}_t \in \mathbb{R}^{1:M \times 256}$.

$$\mathbf{m}_t = \text{Linear}_\theta(\mathbf{M}) \tag{3.10}$$

3.2.4.2 Attention Mechanism in the Feature Integration Process

In the proposed model, the attention mechanism [45] is employed to integrate features. The objective of applying attention is to capture the correlation between individual frames in the sequence. The attention mechanism is utilized in both the Cross-Local Attention and the Self-Attention layers within the Transformer Encoder.

The attention mechanism is formulated as follows:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{Mask}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T + \mathbf{Mask}}{\sqrt{C}}\right)\mathbf{V} \tag{3.11}$$

In the Attention formula above, $\mathbf{Q}$ (Query), $\mathbf{K}$ (Key), and $\mathbf{V}$ (Value) are matrices obtained by passing the input through linear transformation matrices: $\mathbf{Q} = X\mathbf{W}_Q$, $\mathbf{K} = X\mathbf{W}_K$, $\mathbf{V} = X\mathbf{W}_V$. The input X is a matrix representing a sequence of M frames, where each frame is a concatenated vector composed of various feature vectors, including seed gestures, text, speech, emotion, and the gesture $\mathbf{x}_t$ that the thesis aims to denoise. The term $\sqrt{C}$ is a normalization **c**onstant that accounts for the dimensionality of the matrices.

The Local-Cross Attention mechanism is controlled to focus only on local motion features of gestures and features in neighboring frames.

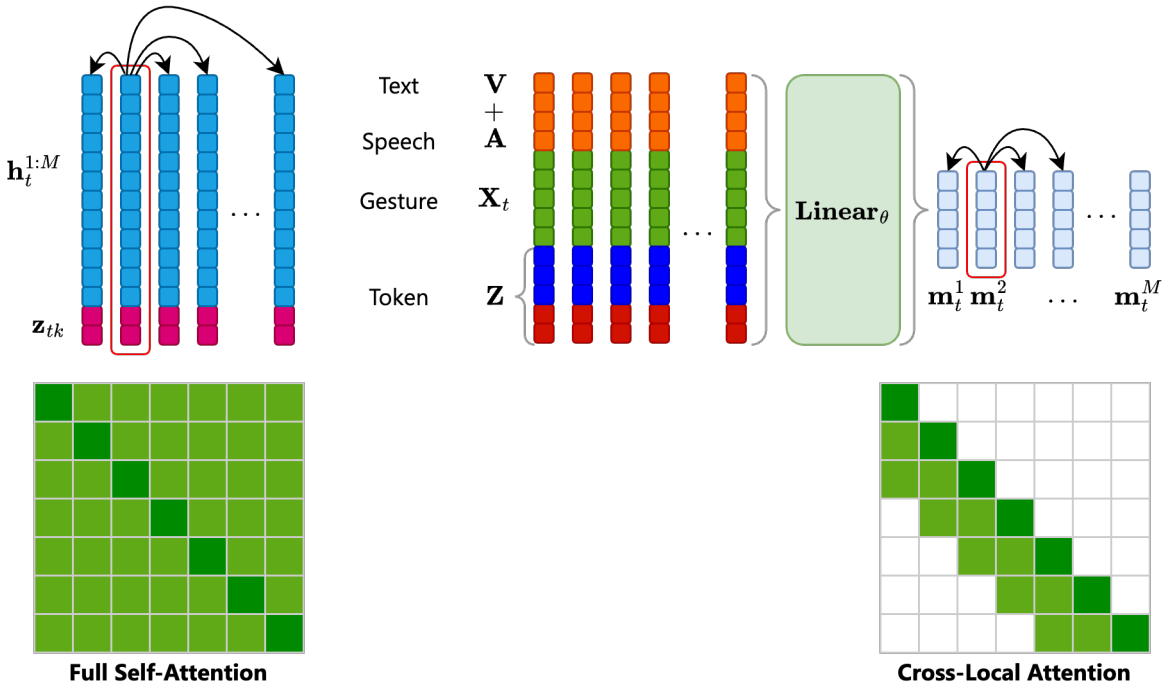


Figure 3.10: Attention Mechanism in Transformer Encoder and Cross-Local Attention

The Attention mechanism functions like a dictionary, where the final retrieved information is the matrix $\mathbf{V}$ (value), while $\mathbf{Q}$ (query) represents the keyword being searched for, and $\mathbf{K}$ (key) is the set of keywords in the lookup dictionary. The Attention process computes the similarity between $\mathbf{Q}$ and $\mathbf{K}$ to determine the weights for the values in $\mathbf{V}$.

The final result is a weighted combination of the values in $\mathbf{V}$, where the values corresponding to keys most similar to the query receive higher weights. $\mathbf{M}$ is the mask used to perform local attention. The Cross-Local Attention mechanism is illustrated on the right in Figure 3.10, while the Transformer Encoder layer uses Self-Attention

shown on the left.

3.2.4.3 Combining Local Features with Cross-Local Attention

The matrix $\mathbf{m}_t \in \mathbb{R}^{1:M\times 256}$ is then passed through the Cross-Local Attention layer to compute the correlations between local features.

$$\mathbf{h}_t = \text{Linear}_\theta(\text{Cross-Local Attention}(\mathbf{m}_t)) \tag{3.12}$$

Cross-local Attention is performed with $\mathbf{Q} = \mathbf{K} = \mathbf{V} = \mathbf{m}_t$. Following the idea of the Routing Transformer method [39], Cross-local Attention highlights the importance of constructing intermediate feature-vector representations before passing them through the Transformer Encoder layer, as shown in Figure 3.9. The feature vectors are augmented with a relative position-encoding vector **RPE** (**R**elative **P**osition **E**ncoding) to preserve temporal ordering before entering the Cross-Local Attention layer.

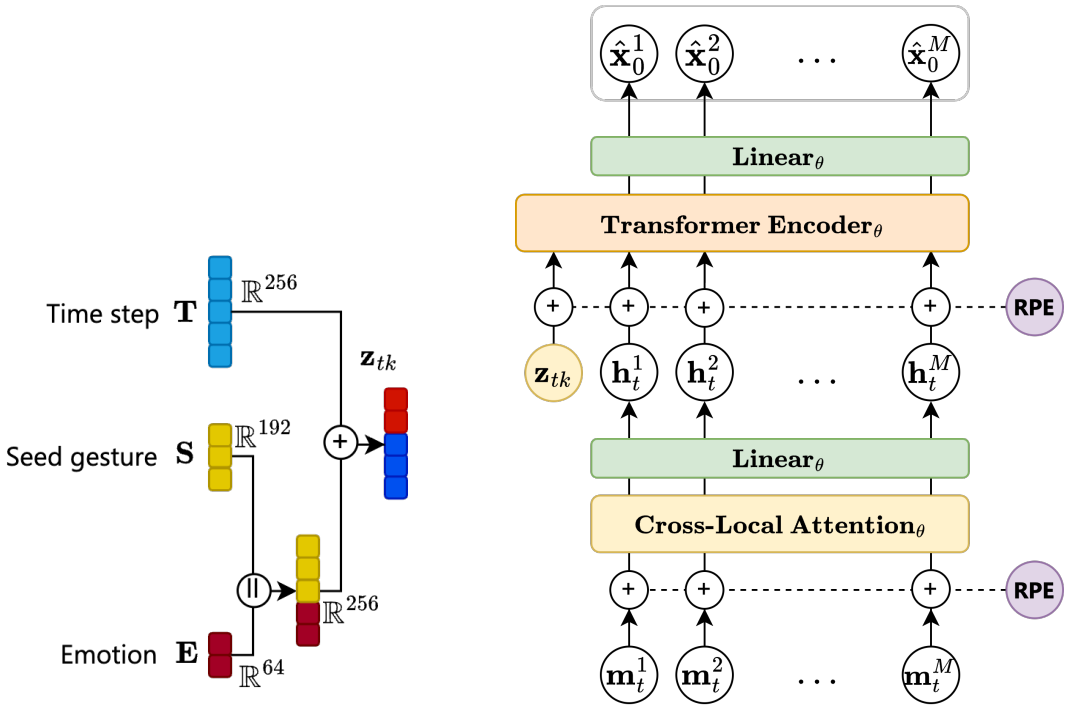
After Cross-Local Attention, the model is forwarded through a linear layer, as in Equation 3.12, to align with the M frames and obtain the matrix $\mathbf{h}_t \in \mathbb{R}^{1:M\times D}$.

3.2.4.4 Global Feature Fusion with the Transformer Encoder

Following MDM [44], the vector $\mathbf{z}_{tk}$ is the first token that encodes information for the entire frame sequence, analogous to the `CLS` token in BERT [13], which summarizes an entire text segment. Here, the thesis uses $\mathbf{z}_{tk}$, with $\mathbf{z}_{tk} \in \mathbb{R}^{256}$ (Equation 3.8), as the first token representing global features for the whole sequence of M frames.

$$\mathbf{X}_0 = \text{Transformer Encoder}\Big(\text{concat}(\mathbf{z}_{tk} \parallel \mathbf{h}_t^{1:M})\Big) \tag{3.13}$$

The vectors $\mathbf{h}_t$ represent the sequence of M frames. Similar to Reformer [26], before entering the Self-Attention layer of the Transformer Encoder, the model employs Relative Position Encoding (RPE) instead of absolute position encoding, improving efficiency on long sequences. Within the Transformer Encoder layer [45], relationships among the data sequences are computed. The Transformer Encoder applies the same self-attention mechanism as in Equation 3.11 but without the **Mask**, enabling correlations across the entire sequence to be captured.



(a) Feature concatenation for $\mathbf{z}_{tk}$ **(b)** Frame-wise feature-fusion process

3.2.5 Feature Decoding Stage

In stage *6. Feature Decoding* (Figure 2.2), once feature correlations are computed, the goal is to upsample the data back to its original dimensionality.

As illustrated in Figure 3.9, the latent matrix $\mathbf{X}_0$, after passing through the Transformer Encoder to capture correlations among heterogeneous data types, is fed into a linear projection layer $\hat{\mathbf{x}}_0 = \text{Linear}_\theta(\mathbf{X}_0)$ to restore the latent matrix to its original size, yielding $\hat{\mathbf{x}}_0 \in \mathbb{R}^{1:M \times D}$.

The final rendering step is presented in Section 4.4.

3.2.6 Emotion Control in Gesture Generation

The preceding steps enable the model to learn gesture generation. To incorporate emotions across different contexts, each emotion is parameterized and varied so that the predictions faithfully express the designated affect.

Analogous to conditional denoising models [21, 44], the thesis uses the condition vector $c = [\mathbf{s}, \mathbf{e}, \mathbf{a}, \mathbf{v}]$, where $\mathbf{s}$ is the seed gesture, $\mathbf{e}$ the emotion, $\mathbf{a}$ the associated speech, and $\mathbf{v}$ the text. The conditional diffusion model injects c at every timestep t in the denoising network $\text{G}_\theta(\mathbf{x}_t, t, c)$, with $c_\varnothing = [\varnothing, \varnothing, \mathbf{a}, \mathbf{v}]$ (unconditional) and

$c = [\mathbf{s}, \mathbf{e}, \mathbf{a}, \mathbf{v}]$ (conditional). A random mask applied to the seed-gesture and emotion vectors conveniently switches labels, allowing optimization under diverse conditions.

$$\hat{\mathbf{x}}_{0\,c,c_\varnothing,\gamma} = \gamma\, G(\mathbf{x}_t, t, c) + (1-\gamma)\, G(\mathbf{x}_t, t, c_\varnothing) \tag{3.14}$$

Classifier-free guidance [21] further enables interpolation between two emotions $\mathbf{e}_1$ and $\mathbf{e}_2$ by setting $c = [\mathbf{s}, \mathbf{e}_1, \mathbf{a}, \mathbf{v}]$ and $c_\varnothing = [\mathbf{s}, \mathbf{e}_2, \mathbf{a}, \mathbf{v}]$:

$$\hat{x}_{0\,\gamma,c_1,c_2} = \gamma\, G(x_t, t, c_1) + (1-\gamma)\, G(x_t, t, c_2).$$

3.2.7 Training Procedure

Algorithm 3 Training in OHGesture

1. Pre-compute γ, $\sqrt{\alpha_t}$, $\sqrt{1-\alpha_t}$, $\sqrt{\bar{\alpha}_t}$, and random noise $\boldsymbol{\epsilon}_t$ for each timestep $t : 1 \to T$. Define the noise schedule $\{\alpha_t \in (0,1)\}_{t=1}^{T}$.
2. Sample the initial label $\mathbf{x}_0$ from the normalized data distribution.
3. Randomly generate Bernoulli masks $c_1 = [\mathbf{s}, \mathbf{e_1}, \mathbf{a}, \mathbf{v}]$, $c_2 = [\mathbf{s}, \mathbf{e_2}, \mathbf{a}, \mathbf{v}]$, or $c_2 = [\varnothing, \varnothing, \mathbf{a}, \mathbf{v}]$.
4. Add noise to obtain the noisy gesture $\mathbf{x}_t$:
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\, \mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\, \boldsymbol{\epsilon}_t.$$
5. Sample t **uniformly** from $[1, T]$.
6. Given $\mathbf{x}_t$, t, and masks c_1, c_2, predict the gesture sequence:
$$\hat{\mathbf{x}}_{0\,\gamma,c_1,c_2} = \gamma\, G_\theta(\mathbf{x}_t, t, c_1) + (1-\gamma)\, G_\theta(\mathbf{x}_t, t, c_2).$$
7. Compute the loss and gradient to update θ:
$$\mathcal{L}^t = \mathbb{E}_{t,\mathbf{x}_0,\boldsymbol{\epsilon}_t}\big[\text{HuberLoss}(\mathbf{x}_0, \hat{\mathbf{x}}_0)\big].$$
8. Repeat from step 6 until convergence, obtaining the optimal parameters θ'.

Algorithm 3 trains the OHGesture model by first computing the required values and hyper-parameters—γ, $\sqrt{\alpha_t}$, $\sqrt{1-\alpha_t}$, $\sqrt{\bar{\alpha}_t}$, and $\boldsymbol{\epsilon}_t$—for every timestep t ($1 \ldots T$). The initial label $\mathbf{x}_0$, representing the ground-truth gesture, is drawn from the normalized data distribution. Random Bernoulli masks c_1 and c_2 emulate different conditions (gesture, emotion, speech, or text), with one mask possibly lacking emotion

information. Noise is then added to create the noisy gesture $\mathbf{x}_t$. A timestep t is sampled uniformly, and $\mathbf{x}_t$ with the masks is fed into the model to predict the original gesture sequence as a weighted combination of conditional outputs. The Huber loss between ground-truth and prediction is used to update θ. This cycle repeats until the model converges, yielding the optimal parameters θ'.

3.2.8 Sampling Process

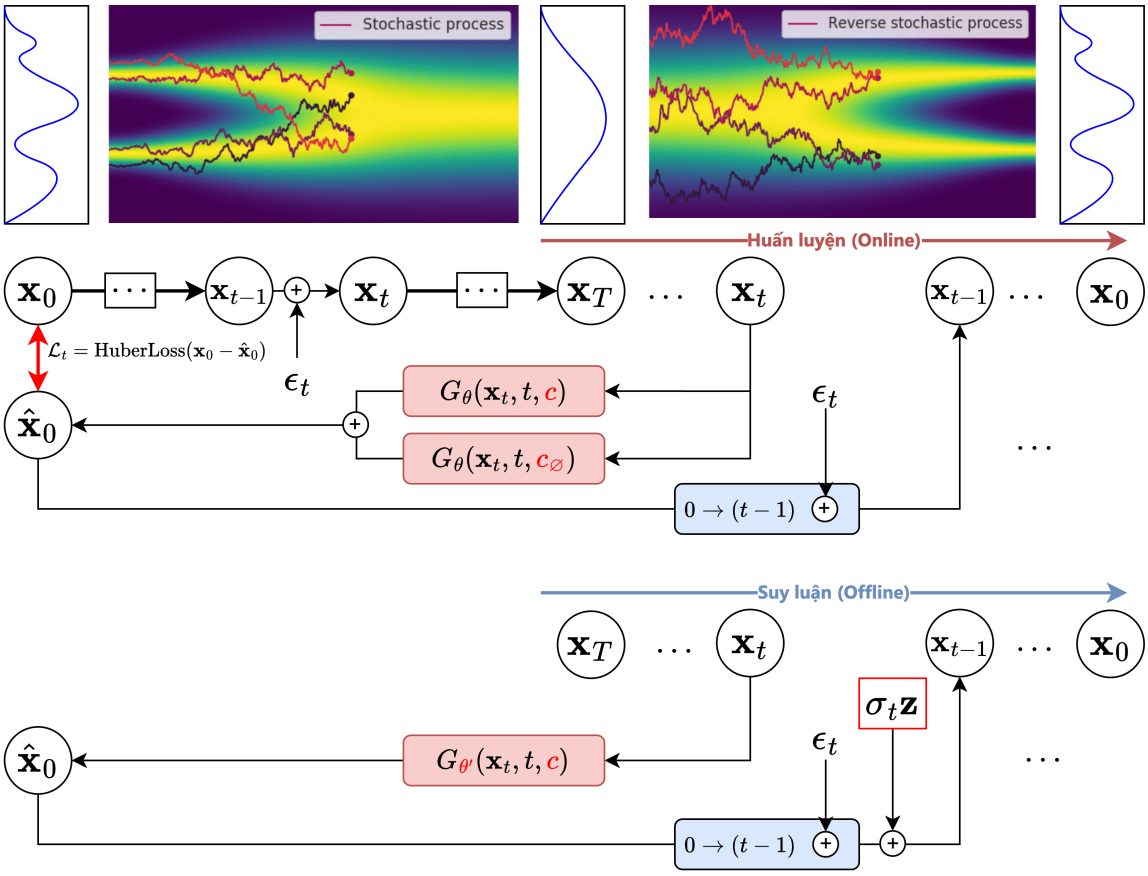


Figure 3.12: Offline (Training) and Online (Inference) Phases

To generate gestures of arbitrary length, the original sequence is segmented into clips of length M. During training, the seed gesture can be chosen by randomly selecting a gesture from the dataset or by averaging the clipped segments—here, the mean rotation angles are used. Generated frames are processed sequentially, with the last $N = 8$ frames taken as the seed for the next iteration. For each clip, the gesture $\mathbf{x}_t$ is denoised via $\hat{\mathbf{x}}_0 = G_{\theta'}(\mathbf{x}_t, t, c)$; noise is re-added to obtain $\mathbf{x}_{t-1}$, and the procedure repeats until $t = 1$, yielding $\mathbf{x}_0$.

Algorithm 4 Sampling in OHGesture

1. Initialize with noise: $\mathbf{x}_T \sim \mathcal{N}(0, \mathbf{I})$.
2. Retrieve $\sqrt{\alpha_t}$, $\sqrt{1-\alpha_t}$, and $\sqrt{\bar{\alpha}_t}$ from training; precompute σ_t from α_t for each timestep $t : 1 \rightarrow T$.
3. Split each 4-second speech segment into $\mathbf{a} \in \mathbb{R}^{64000}$. The initial seed gesture $\mathbf{s}$ is the data mean and is later updated from the inferred gesture segment. Select the desired emotion, obtain the transcript $\mathbf{v}$ from speech $\mathbf{a}$, and form the condition $c = [\mathbf{s}, \mathbf{e}, \mathbf{a}, \mathbf{v}]$.
4. For each timestep, take t **sequentially** from $[T, \ldots, 1]$.
5. Sample random noise $\mathbf{z} \sim \mathcal{N}(0, \mathbf{I})$.
6. Infer $\hat{\mathbf{x}}_0^{(t)} = G_{\theta'}(\mathbf{x}_t, t, c)$.
7. Diffuse $\hat{\mathbf{x}}_0^{(t)}$ from step $0 \rightarrow t$ to obtain $\hat{\mathbf{x}}_{t-1}^{(t)}$.
8. Add noise: $\hat{\mathbf{x}}_{t-1} = \hat{\mathbf{x}}_{t-1}^{(t)} + \sigma_t \mathbf{z}$.
9. Return to step 4. When $t = 1$, output the denoised gesture $\hat{\mathbf{x}}_0$.

Algorithm 4 starts by initializing the noisy gesture $\mathbf{x}_T$ from $\mathcal{N}(0, \mathbf{I})$. The values $\sqrt{\alpha_t}$, $\sqrt{1-\alpha_t}$, and $\sqrt{\bar{\alpha}_t}$ obtained during training, together with σ_t, are employed at each timestep $(1 \ldots T)$. Each 4-second speech segment is represented by $\mathbf{a}$, and the seed gesture $\mathbf{s}$ is taken as the data mean or from the previously inferred segment. The desired emotion and the transcript form the condition $c = [\mathbf{s}, \mathbf{e}, \mathbf{a}, \mathbf{v}]$. The algorithm proceeds sequentially from T to 1: random noise $\mathbf{z}$ is generated, the model predicts $\hat{\mathbf{x}}_0^{(t)}$ from $\mathbf{x}_t$, t, and c, then $\hat{\mathbf{x}}_{t-1}^{(t)}$ is computed and updated with noise. This loop continues until $t = 1$, after which the algorithm outputs the final denoised gesture $\hat{\mathbf{x}}_0$.

Chapter 4. EXPERIMENTS

4.1 Dataset

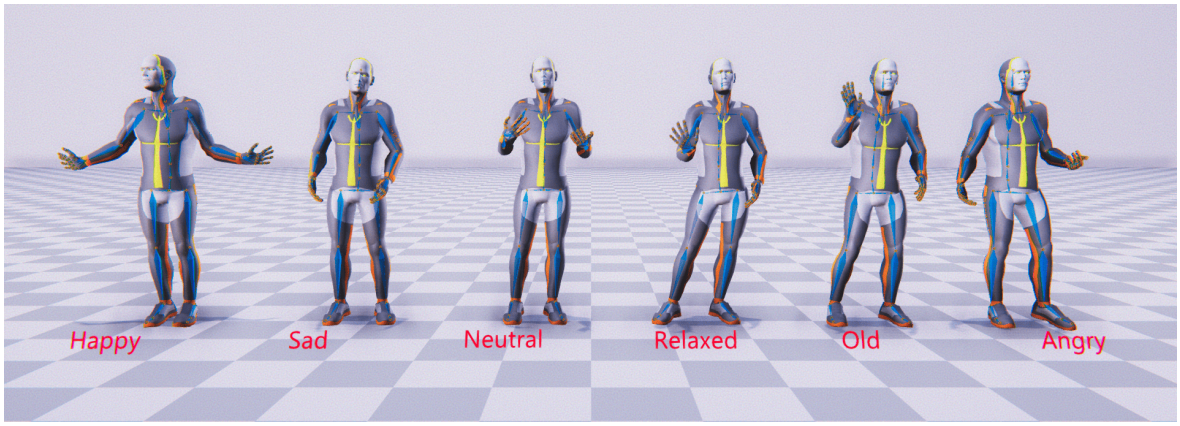


Figure 4.1: Illustration of 6 gestures: `Happy`, `Sad`, `Neutral`, `Old`, `Relaxed`, and `Angry`

This thesis uses the ZeroEGGS dataset [16], a motion capture dataset designed for research and development of gesture generation models. It includes 67 monologue clips performed by a female motion capture actor, with a total duration of 135 minutes. The monologues are annotated with 6 different emotions: `Happy`, `Sad`, `Neutral`, `Old`, `Relaxed`, and `Angry`, enabling simulation of various emotional states in gestures and body movements. ZeroEGGS provides a rich platform for studying the integration of speech and dynamic gestures, supporting the development of models capable of adapting gestures according to emotions and text semantics.

4.2 Data Preprocessing

In stage 1. Data Preprocessing (Figure 2.2), gesture, speech, and text data are read and processed to be represented as vectors or matrices containing information derived from raw data.

For **text data**: the thesis uses the `nltk` library for tokenization and `contractions` to normalize contracted words.

One contribution of the thesis is to convert the speech data available in ZeroEGGS using Adobe Speech To Text, then align the phonetic timestamps using the Montreal Forced Aligner [40] with an English phoneme dictionary to match the gesture frame rate, generating TextGrid files. From these TextGrids containing word-level timing information, the thesis uses `gensim` to generate word2vec embeddings.

Gesture data consists of BVH (BioVision Motion Capture) files captured via motion capture systems. BVH files include two components: Hierarchy and Motion. Specifically:

- `HIERARCHY`: defines a skeletal tree containing 75 bones $\{\mathbf{b}_1, \mathbf{b}_2 \cdots \mathbf{b}_{75}\}$, each with an initial position (`OFFSET`) and `CHANNELS` parameters specifying the type and order of rotation angles (`Zrotation`, `Yrotation`, `Xrotation`) and position (`Xposition`, `Yposition`, `Zposition`), which are defined in the `MOTION` section. The first bone (usually `Hips`) is the root bone $\mathbf{b}_{\text{root}}$, used to define the T-pose via forward kinematics as the initial pose of the skeleton before applying motion.

- `MOTION`: a sequence of frames, each containing motion data representing changes of all 75 bones as defined by the `CHANNELS` in the `HIERARCHY`.

The model in the thesis converts Euler rotation angles to quaternion rotation angles, where a quaternion is a 4-dimensional vector.

$$\mathbf{g} = \left[\mathbf{p}_{\text{root}}, \mathbf{r}_{\text{root}}, \mathbf{p}'_{\text{root}}, \mathbf{r}'_{\text{root}}, \mathbf{p}_{\text{joins}}, \mathbf{r}_{\text{joins}}, \mathbf{p}'_{\text{joins}}, \mathbf{r}'_{\text{joins}}, \mathbf{d}_{\text{gaze}}\right] \tag{4.1}$$

Here, each $\mathbf{g} \in \mathbb{R}^{1141}$ includes:

- $\mathbf{p}_{\text{root}} \in \mathbb{R}^3$: coordinates of the root joint
- $\mathbf{r}_{\text{root}} \in \mathbb{R}^4$: rotation (quaternion) of the root joint
- $\mathbf{p}'_{\text{root}} \in \mathbb{R}^3$: velocity of the root position
- $\mathbf{r}'_{\text{root}} \in \mathbb{R}^3$: angular velocity of the root rotation
- $\mathbf{p}_{\text{joins}} \in \mathbb{R}^{3n_{\text{join}}}$: positions of other joints
- $\mathbf{r}_{\text{joins}} \in \mathbb{R}^{6n_{\text{join}}}$: joint rotations on the X and Y planes

- $\mathbf{p}'_{\text{joins}} \in \mathbb{R}^{3n_{\text{join}}}$: velocity of joint positions
- $\mathbf{r}'_{\text{joins}} \in \mathbb{R}^{3n_{\text{join}}}$: angular velocity of joint rotations
- $\mathbf{d}_{\text{gaze}} \in \mathbb{R}^3$: gaze direction

The original gesture sequences in Euler angles are converted to radians, then converted from Euler to Quaternion as detailed in Section B.3.

Speech data: $\mathbf{a}_{\text{raw}} \in \mathbb{R}^{\text{length}}$ is raw speech sampled at 16000 Hz, trimmed into segments $\mathbf{a} \in \mathbb{R}^{64000}$ corresponding to 4 seconds. The thesis uses `ffmpeg-normalize` to normalize volume to a level lower than the original.

Emotion: Emotion data is represented using a predefined one-hot encoded vector. During sampling, the filename encodes the target emotion.

All data is stored using the `h5` format.

4.3 Training Process

The entire model training process was conducted over approximately two weeks with the following parameters: number of training steps $T = 1000$, using an Nvidia 3090 GPU. The learning rate was set to 3×10^{-5}, batch size was 640, and a total of $43,853$ samples were trained. At each step, t is randomly sampled and input to f_θ to predict $\mathbf{x}_0$. The emotional control parameter was set to $\gamma = 0.1$. The probability of applying random masking to the emotion and initial gesture matrices was 10%, using a Bernoulli distribution to randomly hide/reveal these matrices.

The β parameter was scheduled linearly from $0.5 \rightarrow 0.999$.

The HuberLoss$(\mathbf{x}_0, \hat{\mathbf{x}}_0)$ is computed as follows:

- If $|\mathbf{x}_0 - \hat{\mathbf{x}}_0| \leq \delta$ then $\mathcal{L}_{\delta,\mathbf{x}_0,\hat{\mathbf{x}}_0} = \frac{1}{2}(\mathbf{x}_0 - \mathbf{x}_0)^2$: Below the threshold, the loss is computed as squared distance (similar to MSE), which is sensitive to small errors and provides smooth gradients.
- If $|\mathbf{x}_0 - \hat{\mathbf{x}}_0| > \delta$ then $\mathcal{L}_{\delta,\mathbf{x}_0,\hat{\mathbf{x}}_0} = \delta \cdot |\mathbf{x}_0 - \mathbf{x}_0| - \frac{1}{2}\delta^2$: Above the threshold, the loss behaves like MAE, reducing sensitivity to large errors and improving robustness against outliers.

The training process is implemented in the open-source repository: Github/OHGesture [1].

[1] https://github.com/hmthanh/OHGesture

4.4 Rendering Process in Unity

To visualize the gesture generation process from model output, the thesis uses Unity in stage *7. Rendering* (Figure 2.2), extending code from the DeepPhase model [43]. The generated output is in BVH (BioVision Motion Capture) format. In Unity, the author adds C-Sharp scripts to render gestures based on coordinates and labels, with bone positions and rotations represented using quaternions.

Rendering details are presented in Appendix C.

The Unity project source code is available at: Github/DeepGesture-Unity [2].

[2] https://github.com/DeepGesture/deepgesture-unity

Chapter 5. RESULTS AND EVALUATION

5.1 Evaluation Methods

The evaluation process is conducted through two main metrics: Mean Opinion Scores (MOS) and Fréchet Inception Distance (FID).

5.1.1 Evaluation Based on Human Perception

5.1.1.1 Mean Opinion Scores (MOS)

Currently, there is no standard metric for gesture generation, especially for gesture generation from speech. Therefore, this thesis relies on subjective human evaluation to conduct experimental assessments. Similar to previous methods [56], [29], speech-driven gesture generation models still lack objective metrics that consistently reflect human subjective perception [2].

MOS is measured through three criteria:

- Human-likeness
- Gesture-Speech Appropriateness
- Gesture-Style Appropriateness

One of the contributions of this thesis is the development of the GENEA Leaderboard [1] [35]. This system includes HEMVIP (**H**uman **E**valuation of **M**ultiple **V**ideos in **P**arallel), which is used to compare visual generation results between videos rendered by different models.

Within the GENEA group (**G**eneration and **E**valuation of **N**on-verbal Behaviour for **E**mbodied **A**gents), we hire evaluators on Prolific and conduct a user study based on the rendered video results. Participants rate the results as *Left Better*, *Equal*, or

[1] https://genea-workshop.github.io/leaderboard

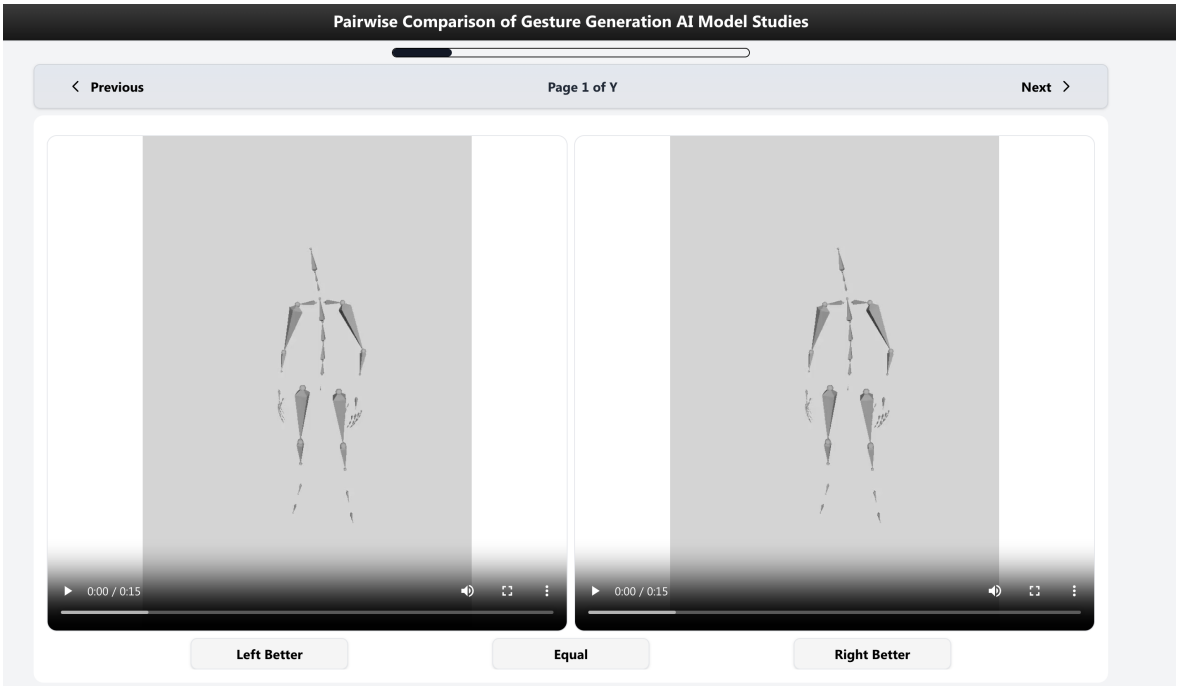


Figure 5.1: HEMVIP system used to evaluate rendering results of two models

Right Better. The updated scores are -1, 0, and 1 for each model, including ground truth data. The comparative results for all models are updated using the Elo rating system.

The source code of the program is available at github.com/hemvip.github.io [2].

5.1.2 Evaluation Based on Quantitative Metrics

5.1.2.1 Mean Square Error (MSE)

The mean square error between the predicted gesture sequence $\hat{\mathbf{y}}_i^{1:M\times D}$ and the ground-truth gesture sequence $\mathbf{y}_i^{1:M\times D}$ is computed as follows:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left\|\mathbf{y}_i^{1:M\times D} - \hat{\mathbf{y}}_i^{1:M\times D}\right\|^2 \tag{5.1}$$

Where:

- n is the number of data samples.
- $\mathbf{y}_i^{1:M\times D}$ is the ground truth of the i^{th} sample, with M being the number of frames and D the number of data dimensions.

[2]HEMVIP 2 https://github.com/hemvip/hemvip.github.io

- $\hat{\mathbf{y}}_i^{1:M\times D}$ is the predicted value of the i^{th} sample, having the same size $M \times D$.
- $\left\|\mathbf{y}_i^{1:M\times D} - \hat{\mathbf{y}}_i^{1:M\times D}\right\|^2$ is the squared norm of the difference between the ground truth and the predicted matrix.

MSE measures the average squared difference between the actual and predicted gesture sequences. The smaller the value, the more accurate the model's predictions. Evaluation results are presented in Subsubsection 5.2.2.1.

5.1.2.2 Fréchet Gesture Distance (FGD)

Similar to image generation methods that use the Fréchet Inception Distance (FID) to measure distributional differences between real and generated data, the Fréchet Gesture Distance (FGD) measures the similarity in distribution between generated gesture sequences $\hat{\mathbf{y}}_i^{1:M\times D}$ and real gesture sequences $\mathbf{y}_i^{1:M\times D}$:

$$\text{FGD} = \|\hat{\mu} - \mu\|^2 + \text{Tr}\left(\Sigma + \hat{\Sigma} - 2\sqrt{\Sigma\hat{\Sigma}}\right) \tag{5.2}$$

Where n is the number of samples, and the parameters are defined as:

- $\mu = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y}_i^{1:M\times D}$ and $\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n} \hat{\mathbf{y}}_i^{1:M\times D}$ are the mean feature vectors of the real dataset $\mathbf{y}_i^{1:M\times D}$ and generated dataset $\hat{\mathbf{y}}_i^{1:M\times D}$, respectively.
- $\Sigma = \frac{1}{n-1}\sum_{i=1}^{n}\left(\mathbf{y}_i^{1:M\times D} - \mu\right)\left(\mathbf{y}_i^{1:M\times D} - \mu\right)^T$ and $\hat{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n}\left(\hat{\mathbf{y}}_i^{1:M\times D} - \hat{\mu}\right)\left(\hat{\mathbf{y}}_i^{1:M\times D} - \hat{\mu}\right)^T$ are the covariance matrices of the features from the real and generated datasets, respectively.
- $\text{Tr}(\cdot)$ denotes the matrix trace operator, which sums the diagonal elements.
- $\sqrt{\Sigma\hat{\Sigma}}$ denotes the matrix square root of the product of the two covariance matrices.

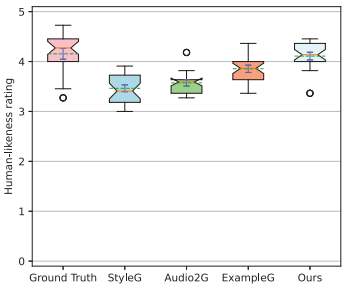
A low FGD score indicates that the distribution of generated gestures is close to the real gestures, while a high FGD suggests a large distributional difference and therefore lower gesture generation quality. In this thesis, the evaluation is performed on the predicted gesture sequence $\hat{\mathbf{x}}^0 \in \mathbb{R}^{1:M\times D}$ and the ground-truth gesture sequence $\mathbf{x}_0 \in \mathbb{R}^{1:M\times D}$.

5.2 Evaluation Results

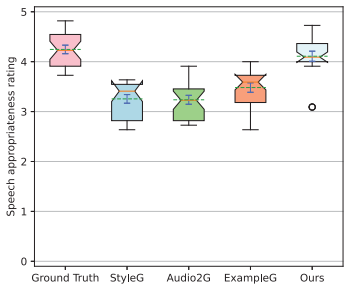
5.2.1 User Study Evaluation Results

5.2.1.1 MOS Evaluation Results

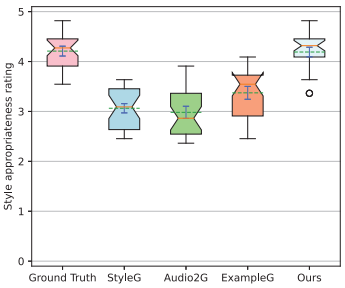
This thesis reuses the evaluation results of the baseline model **DiffuseStyleGesture** [50] for human perception metrics, as gesture generation remains a nascent field and the cost of evaluating multiple models is high. Therefore, this thesis does not include evaluation results for the proposed **OHGesture** model.



(a) Human-likeness



(b) Speech Appropriateness



(c) Style Appropriateness

To understand the visual performance of the proposed method, this thesis conducts a user study comparing gestures generated by the proposed method with those from real motion capture data. The duration of the evaluated video clips ranges from 11 to 51 seconds, with an average length of 31.6 seconds — longer than clips used in the GENEA evaluation [56] (8–10 seconds), as longer durations can provide clearer and more convincing results [51]. Participants rated each video on a scale from 5 to 1, labeled `excellent`, `good`, `fair`, `poor`, and `bad`.

5.2.2 Quantitative Evaluation Results

5.2.2.1 Evaluation Results using MSE

In this thesis, the predicted gesture sequence is segmented over M frames. Mean Square Error (MSE) is applied to the gesture sequence $\mathbf{x}^{1:M\times D}$.

Name	Human likeness ↑	Gesture-speech appropriateness ↑
Ground Truth	4.15 ± 0.11	4.25 ± 0.09
Ours	**4.11 ± 0.08**	**4.11 ± 0.10**
− WavLM	4.05 ± 0.10	3.91 ± 0.11
− Cross-local attention	3.76 ± 0.09	3.51 ± 0.15
− Self-attention	3.55 ± 0.13	3.08 ± 0.10
− Attention + GRU	3.10 ± 0.11	2.98 ± 0.14
+ Forward attention	3.75 ± 0.15	3.23 ± 0.24

Table 5.1: Evaluation results using MOS

Emotion	Neutral	Sad	Happy	Relaxed	Elderly	Angry
DiffuseStyleGesture	75.04	51.40	110.18	130.83	116.03	78.53
ZeroEGG	136.33	81.22	290.47	140.24	102.44	181.07
Proposed Model						
OHGesture	161.22	89.58	279.95	156.93	99.86	215.24

Table 5.2: Mean Square Error results across 6 emotion categories

5.2.2.2 Evaluation Results using FGD

This thesis proposes Fréchet Gesture Distance (FGD), a gesture-based variant of FID, and develops the open-source tool GestureScore [3]. In GestureScore, an Inception V3 model is implemented to encode the frame sequence $\mathbf{x}^{1:M\times D}$ into a latent feature vector of size 32×32, which is then used as input to Equation 5.2. The following Table 5.3 shows the FGD evaluation results of the OHGesture model using GestureScore.

- **Feature Vectors:** The BVH files are used to convert the entire skeleton of each frame into a feature vector of size $D = 1141$, as described in Equation 4.1.
- **Rotations:** From the resulting BVH file, this thesis extracts the rotation angles, $D = 225$ ($225 = 75 \times 3$), from the gesture sequence of length M frames for evaluation in the OHGesture row below.

[3]Github/GestureScore: https://github.com/GestureScore/GestureScore

Name	FGD on Feature Vectors	FGD on Raw Data
Ground Truth	-	-
Ours		
OHGesture (Feature D=1141)	2.058	9465.546
OHGesture (Rotations)	3.513	9519.129

Table 5.3: Evaluation results of Fréchet Gesture Distance (FGD) on $\mathbf{x}^{1:M\times D}$ (from frame 1 to frame M, with D features per frame)

5.3 Building and Standardizing a Gesture Generation Evaluation System

Currently, gesture generation is an active research area with many different models. However, there is no shared evaluation metric. Traditional metrics such as FID (Fréchet Inception Distance) or IS (Inception Score) fail to capture the human-likeness, speech appropriateness, and style appropriateness of generated gestures. Moreover, models are trained and evaluated on different datasets, making it difficult to compare results and determine which model is superior or state-of-the-art. This lack of a standardized evaluation protocol hampers progress in gesture generation research.

To address this, the thesis proposes building an online leaderboard system [35] GENEA Leaderboard [4], which ranks gesture generation models. The thesis collects and processes gesture data from multiple languages and datasets, standardizes them into a unified dataset, and invites authors of various models to train and infer on this standardized set. The resulting generated gestures are then evaluated by hired participants through Prolific.

The thesis is also building an online system hemvip/hemvip.github.io [5] to support the evaluation process via Prolific-based crowd-sourcing. Evaluation results for the OHGesture model will be added based on this system.

Through this evaluation system, the research aims to establish a common benchmark, thereby fostering advancement in gesture generation.

[4]GENEA Leaderboard: https://genea-workshop.github.io/leaderboard/

[5]HEMVIP2 https://github.com/hemvip/hemvip.github.io

Chapter 6. CONCLUSION

6.1 Achieved Results

This thesis has successfully developed and evaluated the gesture generation model OHGesture, a system capable of producing highly natural gestures that convey a human-like impression. A notable highlight of the model is its precise synchronization between gestures and the emotional content of the input speech, with the ability to generalize beyond the training data. This means the diffusion-based model is not solely dependent on the learned gesture data but also demonstrates strong generalization capabilities, enabling it to generate gestures for low-probability speech and contextual inputs.

Another significant aspect of this research is the expansion of input modalities. The thesis does not limit itself to gestures, speech, and emotion labels but also integrates text-to-speech tools to convert speech into text. By incorporating textual features, the model can better capture the semantic aspects of gestures, providing additional context and enabling the system to produce more appropriate and context-aware gestures.

6.2 Strengths and Limitations of the Model

The OHGesture model offers several significant advantages, contributing meaningfully to the development of more natural and flexible human-machine interaction systems. However, there remain some limitations that require future improvements for enhanced effectiveness.

Strengths:

- **High realism:** Based on the gesture generation results, the OHGesture model produces gestures with a high degree of human-likeness. The generated gestures

reflect the nuances and rhythm of speech, enabling the system to synchronize effectively with the emotional and semantic content of the speech.

- **Good generalization ability:** Thanks to the denoising model's capability to cover low-probability data points, the model can infer gestures for situations and emotional states not present in the training dataset, demonstrating potential for deployment in diverse real-world contexts.
- **Controllability over multiple attributes:** The diffusion model allows control over various emotional states and supports interpolation between different emotions.

Limitations:

- The model currently lacks real-time inference capabilities and requires multiple steps to generate the final output.
- The feature representation with $D = 1141$ is processed as an image, which does not fully capture motion characteristics.
- Dependence on high-quality input data: The model requires clear and high-quality speech input to ensure accurate gesture generation. When the input speech is noisy or contains ambiguous emotional variations, the accuracy of the generated gestures may degrade.

6.3 Future Research and Development Directions

Looking ahead, there are several promising directions to improve and expand the OHGesture gesture generation model to better meet practical requirements and enhance the system's applicability. Key directions include:

- **Optimizing the model for real-time inference:** Currently, the model requires speech sequences to be segmented, and the generated results must be imported into Unity for rendering. The thesis aims to develop real-time systems in the future to enable interactive applications and enhance the model's practicality.

- **Optimizing the sampling process and reducing sampling steps:** At present, the gesture generation process requires a relatively high number of sampling steps, which impacts the system's speed and efficiency. Optimizing the process to reduce sampling steps without degrading gesture quality would enable faster responses suitable for real-time applications.

- **Integrating and experimenting with new embedding techniques:** Using novel embedding methods to diversify the input information may help the model better understand and reflect the context and emotions of speech. This development path would also enhance the system's ability to generate semantically appropriate gestures across different languages.

- **Expanding to new languages:** Currently, the model primarily operates with English speech data. Extending the model to support gesture generation for various languages and cultures would be a significant advancement, making the system more diverse and widely applicable.

- **Combining with the DeepPhase model [43] to enable real-time gesture generation:** The thesis aims to integrate OHGesture with the DeepPhase model to develop systems capable of real-time gesture responses, suitable for natural interaction scenarios such as human-machine dialogue and voice-controlled systems. The goal is to learn phase-related motion features to extract motion characteristics more effectively, instead of treating features as image-like representations as done in the current model.

- **Improving objective evaluation with automatic metrics:** To reduce reliance on subjective evaluations, it is necessary to develop and incorporate reliable automatic evaluation methods, enabling the model to self-assess and adjust based on objective indicators.

6.4 Thesis Contributions

In this thesis, the OHGesture gesture generation system was developed, with the following key contributions:

- **Development of a gesture generation model based on Diffusion:** The OHGesture system is designed to generate gestures synchronized with input

speech and accurately reflecting emotion. The model also possesses generalization capabilities, allowing gesture generation even for speech samples outside the training data, thereby achieving high realism.

- **Open-sourcing code and models on public platforms:** To encourage community adoption and improvement, the thesis provides source code on GitHub, with extensions and releases available at Github/OHGesture [1] and a pretrained version on Huggingface at huggingface.co/openhuman/openhuman [2], enabling other researchers to easily access, reproduce, and extend the system.

- **Integration of text and transcribed speech in gesture generation:** Since the ZeroEGGS dataset includes only speech, gesture, and emotion labels, this thesis uses Azure and Google APIs to transcribe the speech files into text. This enriches the model's input with textual features, giving the system additional context to produce more semantically appropriate gestures.

- **Contribution to standardized evaluation systems:** We developed an online ranking system GENEA Leaderboard [3] [35] for gesture generation models. The GENEA Leaderboard collects and processes gesture data from multiple languages and datasets into a unified benchmark, allowing comparative evaluation across various models. Human evaluators are used to assess the models, providing more accurate evaluations of gesture generation results compared to previous metrics, which fail to capture the complexity and diversity of speech-related motion. This creates a unified data foundation that promotes consistent evaluation within the gesture generation research community.

- **Development of a Unity-based visualization tool:** Existing gesture visualization systems rely on Blender and do not render gestures effectively. By extending the source code of the DeepPhase model [43], we developed a Unity-based rendering system Github/DeepGesture-Unity [4].

- **Development of gesture evaluation using FGD (Fréchet Gesture Distance):** Based on the FGD source code [55], this thesis builds GestureScore and

[1] GitHub source code: https://github.com/hmthanh/OHGesture
[2] HuggingFace: https://huggingface.co/openhuman/openhuman
[3] GENEA Leaderboard: https://genea-workshop.github.io/leaderboard/
[4] Unity-based gesture generation rendering system: https://github.com/DeepGesture/deepgesture-unity

trains a new model to evaluate distribution differences in joint rotation angles between predicted and ground-truth data. The code is available at GestureScore [5] and the pretrained evaluation model is available on Huggingface [6].

- **Outlining future development directions:** Based on the diffusion model and a deep understanding of the gesture generation process, the thesis proposes integrating phase-based gesture generation models with advanced processing and feature extraction algorithms to optimize the quality and contextual alignment of generated gestures. This development direction opens up opportunities for significant improvements in gesture interaction with complex contextual elements such as facial expressions, prosody, and emotional dynamics, laying the groundwork for advancements in human-machine communication and related fields.

[5] Github/GestureScore: https://github.com/GestureScore/GestureScore

[6] Huggingface/GestureScore: https://huggingface.co/GestureScore/GestureScore

6.5 Closing Remarks

Through experiments and analysis of gesture generation results, the OHGesture model developed in this thesis—an extension of the DiffuseStyleGesture model—demonstrates its ability to generate realistic gestures not only for data within the training set but also for out-of-distribution voices, such as that of Steve Jobs. This proves the potential of diffusion models for generating gestures in low-probability data contexts.

Furthermore, the thesis contributes modified source code on GitHub, including rendering and data processing pipelines based on Unity, providing a solid foundation for future research and improvements to the OHGesture model. The integration of textual input into the gesture generation process also represents a breakthrough, paving the way for applications in domains requiring more natural and effective human-computer interaction.

REFERENCES

[1] Simon Alexanderson, Gustav Eje Henter, Taras Kucherenko, and Jonas Beskow. Style-controllable speech-driven gesture synthesis using normalising flows. *Computer Graphics Forum*, 39(2):487–496, 2020.

[2] Simon Alexanderson, Rajmund Nagy, Jonas Beskow, and Gustav Eje Henter. Listen, denoise, action! audio-driven motion synthesis with diffusion models. *CoRR*, abs/2211.09707, 2022.

[3] Tenglong Ao, Qingzhe Gao, Yuke Lou, Baoquan Chen, and Libin Liu. Rhythmic gesticulator: Rhythm-aware co-speech gesture synthesis with hierarchical neural embeddings. *ACM Trans. Graph.*, 41(6):209:1–209:19, 2022.

[4] Tenglong Ao, Zeyi Zhang, and Libin Liu. Gesturediffuclip: Gesture diffusion model with clip latents. *ACM Transactions on Graphics (TOG)*, 42(4):1–18, 2023.

[5] Kirsten Bergmann, Volkan Aksu, and Stefan Kopp. The relation of speech and gestures: Temporal synchrony follows semantic synchrony. In *Proceedings of the 2nd Workshop on Gesture and Speech in Interaction (GeSpIn 2011)*, 2011.

[6] Uttaran Bhattacharya, Elizabeth Childs, Nicholas Rewkowski, and Dinesh Manocha. Speech2affectivegestures: Synthesizing co-speech gestures with generative adversarial affective expression learning. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2027–2036, 2021.

[7] Uttaran Bhattacharya, Nicholas Rewkowski, Abhishek Banerjee, Pooja Guhan, Aniket Bera, and Dinesh Manocha. Text2gestures: A transformer-based network for generating emotive body gestures for virtual agents. In *2021 IEEE virtual reality and 3D user interfaces (VR)*, pages 1–10. IEEE, 2021.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017.

[9] Justine Cassell, Catherine Pelachaud, Norman Badler, Mark Steedman, Brett Achorn, Tripp Becket, Brett Douville, Scott Prevost, and Matthew Stone. Animated conversation: rule-based generation of facial expression, gesture & spoken intonation for multiple conversational agents. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 413–420, 1994.

[10] Junming Chen, Yunfei Liu, Jianan Wang, Ailing Zeng, Yu Li, and Qifeng Chen. Diffsheg: A diffusion-based approach for real-time speech-driven holistic 3d expression and gesture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7352–7361, 2024.

[11] Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, Jian Wu, Long Zhou, Shuo Ren, Yanmin Qian, Yao Qian, Jian Wu, Michael Zeng, Xiangzhan Yu, and Furu Wei. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, October 2022.

[12] Chung-Cheng Chiu, Louis-Philippe Morency, and Stacy Marsella. Predicting coverbal gestures: A deep and temporal modeling approach. In *Intelligent Virtual Agents: 15th International Conference, IVA 2015, Delft, The Netherlands, August 26-28, 2015, Proceedings 15*, pages 152–166. Springer, 2015.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[14] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[15] Paul Ekman and Wallace V Friesen. The repertoire of nonverbal behavior: Categories, origins, usage, and coding. *semiotica*, 1(1):49–98, 1969.

[16] Saeed Ghorbani, Ylva Ferstl, Daniel Holden, Nikolaus F. Troje, and Marc-André

Carbonneau. Zeroeggs: Zero-shot example-based gesture generation from speech, 2022.

[17] Ikhsanul Habibie, Weipeng Xu, Dushyant Mehta, Lingjie Liu, Hans-Peter Seidel, Gerard Pons-Moll, Mohamed Elgharib, and Christian Theobalt. Learning speech-driven 3d conversational gestures from video. In *Proceedings of the 21st ACM International Conference on Intelligent Virtual Agents*, pages 101–108, 2021.

[18] Dai Hasegawa, Naoshi Kaneko, Shinichi Shirakawa, Hiroshi Sakuta, and Kazuhiko Sumi. Evaluation of speech-to-gesture generation using bi-directional lstm network. In *Proceedings of the 18th International Conference on Intelligent Virtual Agents*, pages 79–86, 2018.

[19] Gustav Eje Henter, Simon Alexanderson, and Jonas Beskow. Moglow: Probabilistic and controllable motion synthesis using normalising flows. *ACM Transactions on Graphics (TOG)*, 39(6):1–14, 2020.

[20] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[21] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[22] Chien-Ming Huang and Bilge Mutlu. Robot behavior toolkit: generating effective social behaviors for robots. In *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, pages 25–32, 2012.

[23] Chris Jones. Ed - realistic digital human, 2014. Accessed: 2024-11-18.

[24] Jihoon Kim, Jiseob Kim, and Sungjoon Choi. Flame: Free-form language-based motion synthesis & editing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 8255–8263, 2023.

[25] Michael Kipp. *Gesture generation by imitation: From human behavior to computer character animation*. Universal-Publishers, 2005.

[26] Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

[27] Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. *arXiv preprint arXiv:2009.09761*, 2020.

[28] Taras Kucherenko, Patrik Jonell, Sanne Van Waveren, Gustav Eje Henter, Simon Alexandersson, Iolanda Leite, and Hedvig Kjellström. Gesticulator: A framework for semantically-aware speech-driven gesture generation. In *Proceedings of the 2020 international conference on multimodal interaction*, pages 242–250, 2020.

[29] Taras Kucherenko, Patrik Jonell, Youngwoo Yoon, Pieter Wolfert, and Gustav Eje Henter. A large, crowdsourced evaluation of gesture generation systems on common data: The genea challenge 2020. In *26th international conference on intelligent user interfaces*, pages 11–21, 2021.

[30] Sergey Levine, Philipp Krähenbühl, Sebastian Thrun, and Vladlen Koltun. Gesture controllers. In *Acm siggraph 2010 papers*, pages 1–11. ACM, 2010.

[31] Haiyang Liu, Zihao Zhu, Naoya Iwamoto, Yichen Peng, Zhengqing Li, You Zhou, Elif Bozkurt, and Bo Zheng. Beat: A large-scale semantic and emotional multimodal dataset for conversational gestures synthesis. In *European conference on computer vision*, pages 612–630. Springer, 2022.

[32] Xian Liu, Qianyi Wu, Hang Zhou, Yinghao Xu, Rui Qian, Xinyi Lin, Xiaowei Zhou, Wayne Wu, Bo Dai, and Bolei Zhou. Learning hierarchical cross-modal association for co-speech gesture generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10462–10472, 2022.

[33] Evelyn McClave. Gestural beats: The rhythm hypothesis. *Journal of psycholinguistic research*, 23(1):45–66, 1994.

[34] Adam Metallo, Vincent Rossi, Jonathan Blundell, Günter Waibel, Paul Graham, Graham Fyffe, Xueming Yu, and Paul Debevec. Scanning and printing a 3d portrait of president barack obama. In *SIGGRAPH 2015: Studio*, pages 1–1. ACM, 2015.

[35] Rajmund Nagy, Hendric Voss, Youngwoo Yoon, Taras Kucherenko, Teodor Nikolov, Thanh Hoang-Minh, Rachel McDonnell, Stefan Kopp, Michael Neff, and

Gustav Eje Henter. Towards a genea leaderboard–an extended, living benchmark for evaluating and advancing conversational motion synthesis. *arXiv preprint arXiv:2410.06327*, 2024.

[36] Michael Neff, Michael Kipp, Irene Albrecht, and Hans-Peter Seidel. Gesture modeling and animation based on a probabilistic re-creation of speaker style. *ACM Transactions On Graphics (TOG)*, 27(1):1–24, 2008.

[37] Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pages 8162–8171. PMLR, 2021.

[38] Shenhan Qian, Zhi Tu, Yihao Zhi, Wen Liu, and Shenghua Gao. Speech drives templates: Co-speech gesture synthesis with learned templates. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 11077–11086, 2021.

[39] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021.

[40] Michael Saxon, Ayush Tripathi, Yishan Jiao, Julie M Liss, and Visar Berisha. Robust estimation of hypernasality in dysarthria with acoustic model likelihood features. *IEEE/ACM transactions on audio, speech, and language processing*, 28:2511–2522, 2020.

[41] Thomas A Sebeok and Jean Umiker-Sebeok. *Advances in visual semiotics: The semiotic web 1992-93*, volume 118. Walter de Gruyter, 2011.

[42] Yang Song. Score-based generative modeling: A brief introduction, 2021. Accessed: 2024-11-18.

[43] Sebastian Starke, Ian Mason, and Taku Komura. Deepphase: Periodic autoencoders for learning motion phase manifolds. *ACM Transactions on Graphics (TOG)*, 41(4):1–13, 2022.

[44] Guy Tevet, Sigal Raab, Brian Gordon, Yonatan Shafir, Daniel Cohen-Or, and Amit H Bermano. Human motion diffusion model. *arXiv preprint arXiv:2209.14916*, 2022.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[46] Rebecca A Webb. *Linguistic features of metaphoric gestures*. University of Rochester, 1997.

[47] Lilian Weng. What are diffusion models? *lilianweng.github.io*, Jul 2021.

[48] Bowen Wu, Chaoran Liu, Carlos T Ishi, and Hiroshi Ishiguro. Probabilistic human-like gesture synthesis from speech using gru-based wgan. In *Companion Publication of the 2021 International Conference on Multimodal Interaction*, pages 194–201, 2021.

[49] Jing Xu, Wei Zhang, Yalong Bai, Qibin Sun, and Tao Mei. Freeform body motion generation from speech. *arXiv preprint arXiv:2203.02291*, 2022.

[50] Sicheng Yang, Zhiyong Wu, Minglei Li, Zhensong Zhang, Lei Hao, Weihong Bao, Ming Cheng, and Long Xiao. Diffusestylegesture: Stylized audio-driven co-speech gesture generation with diffusion models. *arXiv preprint arXiv:2305.04919*, 2023.

[51] Sicheng Yang, Zhiyong Wu, Minglei Li, Mengchen Zhao, Jiuxin Lin, Liyang Chen, and Weihong Bao. The reprgesture entry to the genea challenge 2022. In *Proceedings of the 2022 International Conference on Multimodal Interaction*, pages 758–763, 2022.

[52] Sicheng Yang, Zunnan Xu, Haiwei Xue, Yongkang Cheng, Shaoli Huang, Mingming Gong, and Zhiyong Wu. Freetalker: Controllable speech and text-driven gesture generation based on diffusion models for enhanced speaker naturalness. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7945–7949. IEEE, 2024.

[53] Sicheng Yang, Haiwei Xue, Zhensong Zhang, Minglei Li, Zhiyong Wu, Xiaofei Wu, Songcen Xu, and Zonghong Dai. The diffusestylegesture+ entry to the genea challenge 2023. In *Proceedings of the 2023 International Conference on Multimodal Interaction*, 2023.

[54] Yanzhe Yang, Jimei Yang, and Jessica Hodgins. Statistics-based motion synthesis for social conversations. In *Computer Graphics Forum*, volume 39, pages 201–212. Wiley Online Library, 2020.

[55] Youngwoo Yoon, Bok Cha, Joo-Haeng Lee, Minsu Jang, Jaeyeon Lee, Jaehong Kim, and Geehyuk Lee. Speech gesture generation from the trimodal context of text, audio, and speaker identity. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020.

[56] Youngwoo Yoon, Pieter Wolfert, Taras Kucherenko, Carla Viegas, Teodor Nikolov, Mihail Tsakov, and Gustav Eje Henter. The genea challenge 2022: A large evaluation of data-driven co-speech gesture generation. In *Proceedings of the 2022 International Conference on Multimodal Interaction*, pages 736–747, 2022.

[57] Mingyuan Zhang, Zhongang Cai, Liang Pan, Fangzhou Hong, Xinying Guo, Lei Yang, and Ziwei Liu. Motiondiffuse: Text-driven human motion generation with diffusion model. *arXiv preprint arXiv:2208.15001*, 2022.

[58] Chi Zhou, Tengyue Bian, and Kang Chen. Gesturemaster: Graph-based speech-driven gesture generation. In *Proceedings of the 2022 International Conference on Multimodal Interaction*, pages 764–770, 2022.

LIST OF THE MASTER'S PUBLISHED WORKS

1. Towards a GENEA Leaderboard – an Extended, Living Benchmark for Evaluating and Advancing Conversational Motion Synthesis [35].

Appendix A. PARAMETERS IN DIFFUSION

A.1 Variation of $\sqrt{\alpha}$ and $\sqrt{1-\alpha}$

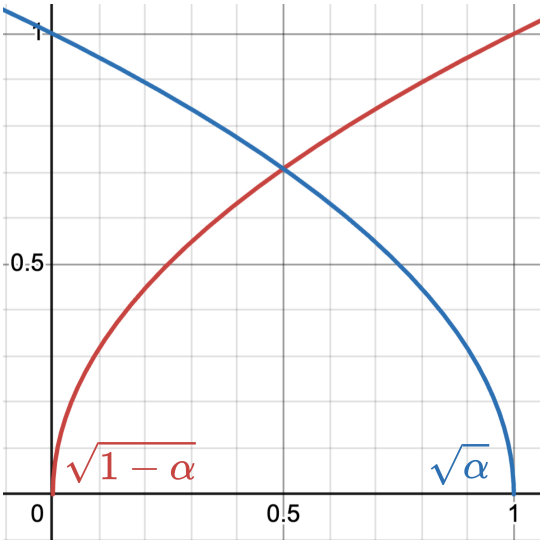


Figure A.1: Variation of $\sqrt{\alpha}$ and $\sqrt{1-\alpha}$

During the diffusion process, the system aims to gradually reduce the presence of $\mathbf{x}$ and increase the presence of noise ϵ_t. With $\mathbf{x}_t = \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_t$, the coefficients $\sqrt{\alpha_t}$ and $\sqrt{1-\alpha_t}$ control this process as follows:

- $\sqrt{\alpha_t}$: Initially has a high value but gradually decreases to reduce the influence of $\mathbf{x}_t$ in the combined result with noise.
- $\sqrt{1-\alpha_t}$: Initially has a small value but gradually increases at each step, with the goal that eventually the noise dominates and becomes fully noise.

Both of these quantities change over time during the diffusion process, determining how much noise is added at each step t and how much of the previous state is retained at each step.

A.2 Variation of $\sqrt{\bar{\alpha}}$ and $\sqrt{1-\bar{\alpha}}$

$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$. $\sqrt{1-\bar{\alpha}_t} = \sqrt{1-\prod_{i=1}^{t} \alpha_i}$. Both $\sqrt{\bar{\alpha}}$ and $\sqrt{1-\bar{\alpha}}$ play an important role in the training and gesture generation processes. They help control the level of noise added and the gestures generated. It can be observed that $\sqrt{\bar{\alpha}}$ gradually decreases while $\sqrt{1-\bar{\alpha}}$ increases.

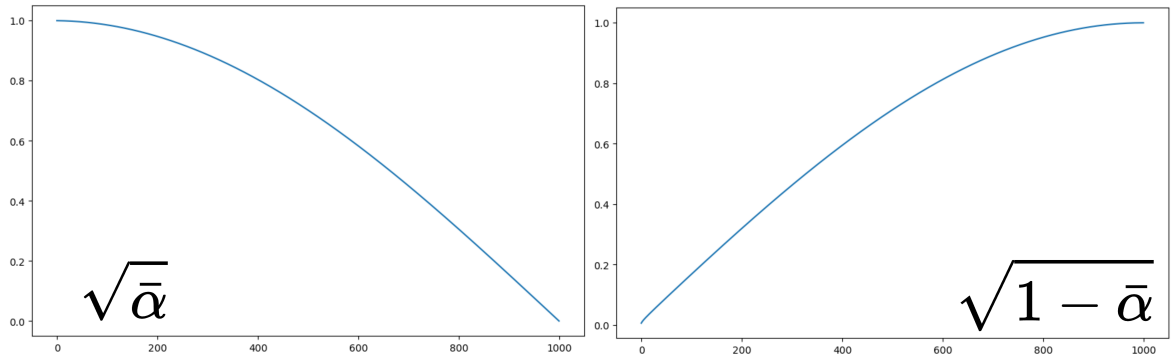


Figure A.2: Variation of $\sqrt{\bar{\alpha}}$ and $\sqrt{1-\bar{\alpha}}$

A.3 Variation of σ_t

During the sampling process, the goal of decreasing σ_t is to allow the model to navigate toward high-density data regions at each step t.

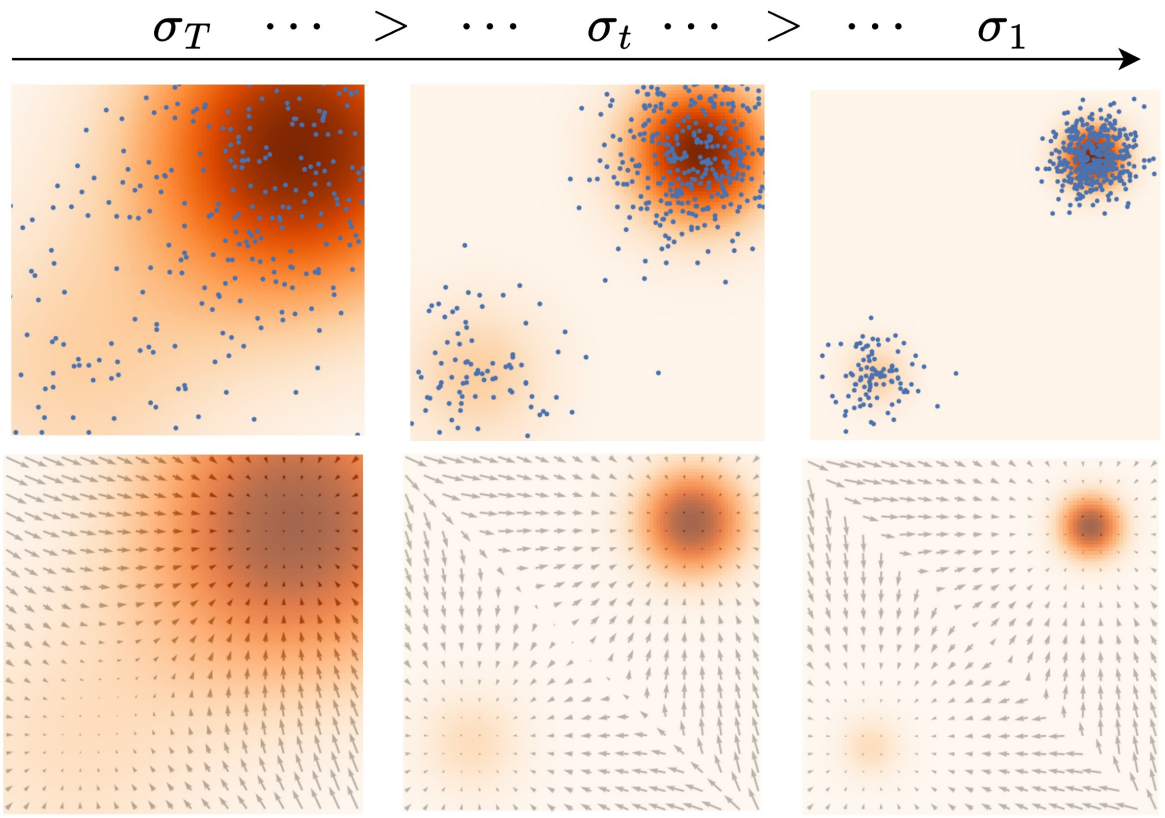


Figure A.3: Variation of the coefficient σ_t during the sampling process

Appendix B. BVH DATA PROCESSING PIPELINE

B.1 Skeleton Structure of a Character

Some skeleton joint names from the 75 motion skeletons include:

`Hips`, `Spine`, `Neck`, `Head`, `RightShoulder`, `RightArm`, `RightForeArm`, `RightHand`, `LeftShoulder`, `LeftArm`, `LeftForeArm`, `LeftHand`, `RightUpLeg`, `RightLeg`, `RightFoot`, `RightToeBase`, `LeftUpLeg`, `LeftLeg`, `LeftFoot`, `LeftToeBase`, ...

Figure B.1: Character skeleton

B.2 Structure of a BVH File

A BVH (Biovision Hierarchy) file is a data format that contains information about the skeleton structure and motion data of bones in a skeletal system. A BVH file consists of two main parts: the skeleton hierarchy declaration and the bone motion data.

- **HIERARCHY**:
  - Defines the components and names of the skeleton joints, as well as the initial positions of the joints in the T-pose (motion capture actors extend their arms horizontally to form a "T").
  - Defines the parent-child relationships from the root node to the leaf nodes of the skeleton, typically with the root node being the spine (`Spine`).
  - Specifies the data to be recorded such as position or rotation angles along X, Y, Z axes of each joint over time.
- **MOTION**: A sequence of movements frame by frame, where each frame contains bone movement data as defined in the HIERARCHY section (e.g., rotation angles or positions).

```
HIERARCHY
ROOT Hips
{
    OFFSET 0.00352400006 86.606987 0
    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
    JOINT Spine
    {
        OFFSET 3.72965547e-17 8.81424618 -2.08010697
        CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
        JOINT Spine1
        {
            OFFSET -1.01898064e-17 8.84414101 -1.06827795e-17
            CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
            JOINT Spine2
            {
                OFFSET -7.98921476e-17 11.6822681 -3.7201687e-17
                CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
                JOINT Spine3
                {
                    OFFSET -1.73472348e-17 11.7467003 -3.81639165e-16
                    CHANNELS 6 Xposition Yposition Zposition Zrotation Yrotation Xrotation
                    JOINT Neck
```

(a) HIERARCHY in BVH file

```
MOTION
Frames: 7286
Frame Time: 0.016667
0.00352400006 86.606987 0 0.490480989 0.0279319994 -6.49757195
0.00352400006 86.606987 -0.0283540003 0.48171699 -0.0194809996
0.00352400006 86.606987 -0.0546680018 0.47310701 -0.0689229965
0.00352400006 86.6056061 -0.0783089995 0.464415014 -0.12521399
0.00352400006 86.6043854 -0.100622997 0.455056995 -0.185530007
0.00352400006 86.6031876 -0.122184001 0.444750011 -0.249752 -6
0.00352400006 86.6019974 -0.142823994 0.433441013 -0.317882001
0.00352400006 86.6007996 -0.163385004 0.420841008 -0.38962099
0.00352400006 86.5995941 -0.185066 0.405855 -0.464527011 -6.83
0.00352400006 86.5983505 -0.207151994 0.389432997 -0.541544974
0.00352400006 86.5970688 -0.229480997 0.372554004 -0.614641011
0.00352400006 86.5951233 -0.25286001 0.353704005 -0.684701025
0.00352400006 86.5928802 -0.277058005 0.333912998 -0.756413996
0.00352400006 86.5905991 -0.30133 0.314761996 -0.830637991 -7.
0.00352400006 86.588295 -0.325839996 0.296337992 -0.905332029
0.00352400006 86.5859909 -0.350551009 0.278358996 -0.979005992
0.00352400006 86.5836182 -0.375133991 0.262282014 -1.05174804
5.09999991e-05 86.5807953 -0.39993 0.249116004 -1.12400305 -7.
```

(b) MOTION in BVH file

$\mathbf{rotation}_i^{\text{local}} = \{\alpha, \beta, \gamma\}$ represents the rotation angles around the Z, Y, and X axes, respectively. The combined rotation in Euler space is:

$$R = R_Z(\alpha)R_Y(\beta)R_X(\gamma) \tag{B.1}$$

Where:

1. **Rotation matrix around axis** Z:

$$R_Z(\alpha) = \begin{bmatrix} \cos(\alpha) & -\sin(\alpha) & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

2. **Rotation matrix around axis** Y:

$$R_Y(\beta) = \begin{bmatrix} \cos(\beta) & 0 & \sin(\beta) \\ 0 & 1 & 0 \\ -\sin(\beta) & 0 & \cos(\beta) \end{bmatrix}$$

3. **Rotation matrix around axis** X:

$$R_X(\gamma) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\gamma) & -\sin(\gamma) \\ 0 & \sin(\gamma) & \cos(\gamma) \end{bmatrix}$$

To compute the motion coordinates of a character, the following operation is applied:

$$\mathbf{position}_{\text{global}} = R \cdot \mathbf{position}_{\text{local}} + \mathbf{t} \tag{B.2}$$

B.3 Conversion from Euler Angles to Quaternions

To avoid Gimbal lock, Euler angle data must be converted into quaternion representation. Each bone's rotation from Euler angles in the ZYX order is represented as a quaternion $q = (q_w, q_x, q_y, q_z)$, with components calculated as follows:

First, compute the cos and sin values of half the rotation angles for each axis:

- $c_\alpha = \cos\left(\frac{\alpha}{2}\right), \quad s_\alpha = \sin\left(\frac{\alpha}{2}\right)$
- $c_\beta = \cos\left(\frac{\beta}{2}\right), \quad s_\beta = \sin\left(\frac{\beta}{2}\right)$
- $c_\gamma = \cos\left(\frac{\gamma}{2}\right), \quad s_\gamma = \sin\left(\frac{\gamma}{2}\right)$

Based on the values above, the quaternion components are computed as:

- $q_w = c_\alpha c_\beta c_\gamma + s_\alpha s_\beta s_\gamma$

- $q_x = c_\alpha c_\beta s_\gamma - s_\alpha s_\beta c_\gamma$
- $q_y = c_\alpha s_\beta c_\gamma + s_\alpha c_\beta s_\gamma$
- $q_z = s_\alpha c_\beta c_\gamma - c_\alpha s_\beta s_\gamma$

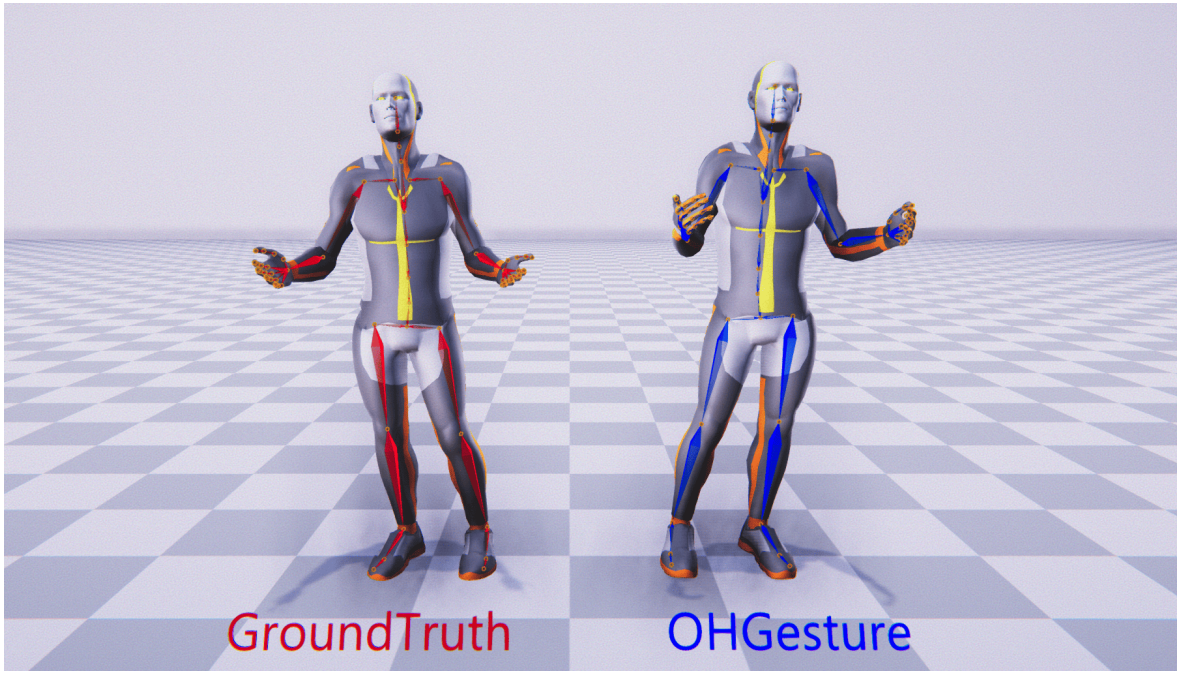
With the computed quaternion q, the global position of the bone $\mathbf{p}_{\text{global}}$ is determined by rotating the local position $\mathbf{p}_{\text{local}}$ using the formula:

$$\mathbf{p}_{\text{global}} = q \cdot \mathbf{p}_{\text{local}} \cdot q^{-1} + \mathbf{t} \tag{B.3}$$

where $\mathbf{t}$ is the origin position of the bone in global space.

Appendix C. ILLUSTRATION OF GESTURE INFERENCE RESULTS

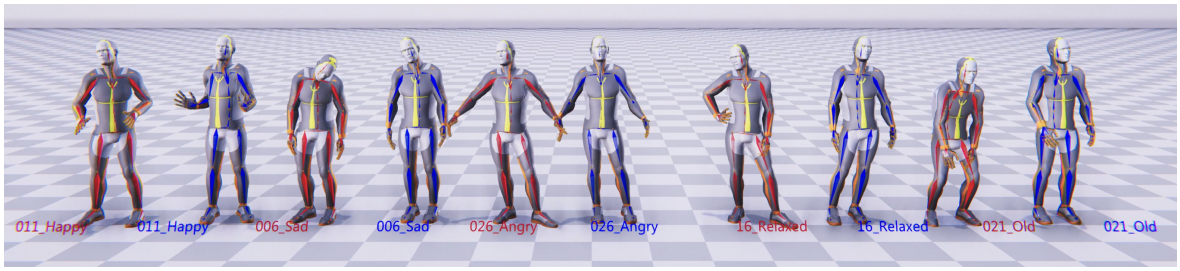
C.1 Comparison between Ground Truth Gestures and Predicted Gestures



Click the image to watch the video

The results show both the ground truth gestures and the model's predicted gestures at frame 3821, inferred from the speech sample `003_Neutral_2_x_1_0`.

C.2 Illustration of Different Emotions in Gestures



Click the image to watch the video

Generated or inferred results from the OHGesture model with different emotions. Red indicates the ground truth from the ZeroEGGS dataset, while blue shows the output generated by the OHGesture model.

Click the image to watch the video

C.3 Illustration of Gesture Generation with Out-of-Training Speech

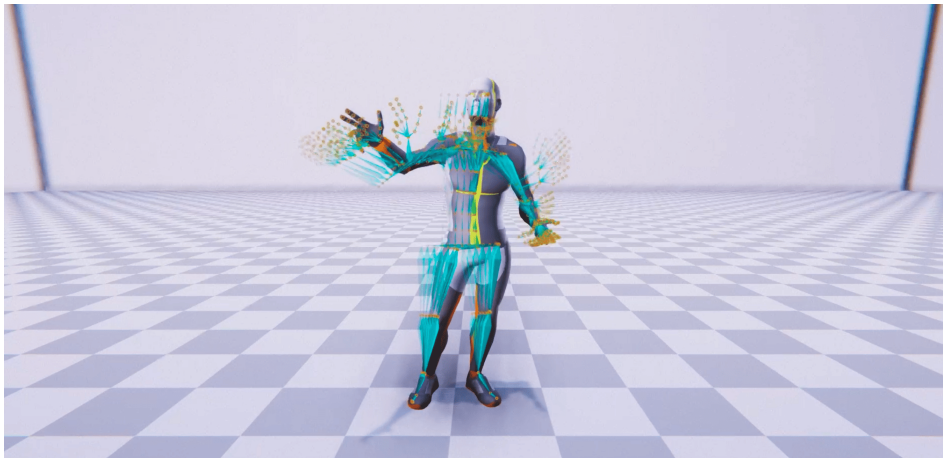
Click the image to watch the video

Illustration of gesture generation corresponding to Steve Jobs' speech.

Click the image to watch the video

Illustration of gesture generation with synthesized speech from Microsoft Azure introducing the topic.

C.4 Illustration of Character Motion



Click the image to watch the video

Illustration of gestures extracted from 6 frames before and 6 frames after the target gesture.